Channel estimation for IEEE 802.11p vehicular communication using deep learning

SA Ngorima

orcid.org/ 0000-0002-0775-3529

Thesis accepted in fulfilment of the requirements for the degree *Doctor of Philosophy in Engineering with Computer and Electronic Engineering* at the North-West University

Promoter:	Prof ASJ Helberg
Co-promoter:	Prof MH Davel

Graduation:	July 2026
Student number:	36450057

Declaration

I, Simbarashe Aldrin Ngorima, hereby declare that the dissertation entitled "Channel estimation for IEEE 802.11p vehicular communication using deep learning" is my own original work and has not already been submitted to any other university or institution for examination.

S.A. Ngorima

Student number: 36450057

Signed on the 28th day of November 2025 at Potchefstroom.

Acknowledgements

This thesis is the product of several years of work, and it would not have been possible without the support and encouragement of many people, to whom I owe a great deal.

My deepest thanks go to my supervisors, Prof. Albert Helberg and Prof. Marelie Davel. Prof. Helberg grounded the work in the realities of communications engineering and kept the broader picture in view. Prof. Davel pushed me toward greater rigour and clarity of thought, and taught me to question my own results before anyone else could. Their patience, careful reading, and steady guidance shaped both this thesis and the researcher I have become.

I am grateful to the MUST Deep Learning research group at North-West University for providing an environment in which this research could be carried out.

To my girlfriend, thank you for your steady love and encouragement, for believing in me on the days I struggled to believe in myself, and for carrying this journey alongside me.

To my parents, specifically my mother and my father, thank you for your prayers and your belief in me from the very beginning, carried all the way from Zimbabwe.

Above all, I give thanks to God, from whom all wisdom and strength come, and by whose grace this work was begun and completed. To Him be the glory.

Abstract

Vehicular communication systems require precise channel estimation in dynamic propagation environments. Traditional methods struggle in high-mobility settings due to rapid Doppler shifts and multipath effects. Recent deep learning (DL) approaches demonstrate improved accuracy but face challenges in generalisation across diverse operating conditions, inference latency, and interpretability.

This thesis investigates DL-based channel estimation for IEEE 802.11p vehicular communications through three main contributions. First, different DL architectures are proposed and evaluated. A novel architecture that combines data pilot-aided (DPA) estimation with non-causal one-dimensional convolutions was found to achieve superior bit error rate (BER) performance with a median single-frame inference latency of 4.16 ms, satisfying the receiver processing budget for IEEE 802.11p continuous frame reception. The architecture employs a processing order where DPA first provides temporal tracking anchor points, whilst the temporal convolutional network (TCN) corrects errors through dilated convolutions.

Secondly, training strategies that improve generalisation across diverse noise levels and propagation environments are investigated. Mixed signal-to-noise ratio (SNR) training substantially improves performance at low SNR across all evaluated architectures. Multi-channel training enables a single model trained on combined data from three propagation models to generalise effectively to unseen environments, eliminating the need for channel-specific estimators.

Thirdly, the Relevance-based Explainable Analysis for Channel estimation (REACH) framework extends layer-wise relevance propagation (LRP) for interpretability-driven optimisation. For feedforward networks, REACH enables 43.6% parameter reduction without significant performance degradation. For multi-channel TCNs, cross-channel relevance correlation exceeds 0.81, explaining the strong generalisation to unseen propagation environments. Analysis of cross-channel important features reveals that only 19.6% of input features are critical across six channel models.

Combined, these contributions establish the DPA-TCN architecture as an effective estimator for IEEE 802.11p systems, combining efficient single-frame inference, generalisation across diverse conditions, and interpretability-driven optimisation.

Keywords: *Vehicular communications, IEEE 802.11p, Channel estimation, Deep learning, Generalisation, Mixed-SNR training, Multi-channel training, Interpretability*

Contents

List of Figures

List of Tables

List of Acronyms

AGC	Automatic Gain Control
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BSM	Basic Safety Message
CCH	Control Channel
CDP	Constructed Data Pilots
CEPT	Conference of Postal and Telecommunications Administrations
CFO	Carrier Frequency Offset
CFR	Channel Frequency Response
CNN	Convolutional Neural Network
CP	Cyclic Prefix
CPU	Central Processing Unit
CSI	Channel State Information
C-V2X	Cellular Vehicle-to-Everything
DeepLIFT	Deep Learning Important FeaTures
DL	Deep Learning
DNN	Deep Neural Network
DPA	Data-Pilot Aided
DSRC	Dedicated Short-Range Communications
EDCA	Enhanced Distributed Channel Access
FBF	Frame-by-Frame
FCC	Federal Communications Commission

FFT	Fast Fourier Transform
FLOPs	Floating Point Operations
FNN	Feedforward Neural Network
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
GRACE	Gradient-based XAI Channel Estimation
GRU	Gated Recurrent Unit
ICI	Inter-Carrier Interference
ID	In-Distribution
IDFT	Inverse Discrete Fourier Transform
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
ISI	Inter-Symbol Interference
ITS	Intelligent Transportation Systems
LFSR	Linear Feedback Shift Register
LOS	Line-of-Sight
LRP	Layer-wise Relevance Propagation
LS	Least Squares
LSTM	Long Short-Term Memory
LTS	Long Training Symbol
LTV	Linear Time-Variant
MAC	Medium Access Control
MACC	Multiply-Accumulate Operation
MAE	Mean Absolute Error
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MMSE	Minimum Mean Squared Error
MPDU	MAC Protocol Data Unit
MSE	Mean Square Error
NLOS	Non-Line-of-Sight

NMSE	Normalised Mean Square Error
NN	Neural Network
OFDM	Orthogonal Frequency Division Multiplexing
OOD	Out-of-distribution
PDP	Power Delay Profile
PHY	Physical Layer
PPDU	Physical Layer Protocol Data Unit
PRR	Packet Reception Ratio
PSD	Power Spectral Density
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
R2V	Roadside-to-Vehicle
REACH	Relevance Explainability Analysis for Channel Estimation
ReLU	Rectified Linear Unit
RMS-DS	Root Mean Square Delay Spread
RNN	Recurrent Neural Network
RSU	Road Side Unit
RTV-EX	Roadside-to-Vehicle Expressway
RTV-SS	Roadside-to-Vehicle Suburban Street
RTV-UC	Roadside-to-Vehicle Urban Canyon
SBS	Symbol-by-Symbol
SCH	Service Channel
SF	SIGNAL Field
SISO	Single-Input Single-Output
SNR	Signal-to-Noise Ratio
SOTA	State-of-the-Art
STA	Spectral Temporal Averaging
STS	Short Training Symbol
TA	Time Averaging
TCN	Temporal Convolutional Network

TDL	Tapped Delay Line
TRFI	Time-domain Reliable Frequency-domain Interpolation
US	Uncorrelated Scattering
UTC	Coordinated Universal Time
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VTV	Vehicle-to-Vehicle
VTV-EX	Vehicle-to-Vehicle Expressway
VTV-SDWW	Vehicle-to-Vehicle Same Direction With Wall
VTV-UC	Vehicle-to-Vehicle Urban Canyon
WAVE	Wireless Access in Vehicular Environments
WSS	Wide-Sense Stationary
WSSUS	Wide-Sense Stationary Uncorrelated Scattering
XAI-CHEST	Explainable AI for Channel Estimation

List of Symbols

$\mathbf{b}^{(\ell)}$	Bias vector at layer ℓ
f_d	Maximum Doppler shift
f_θ	Neural mapper with parameters θ
$f^\star$	Unknown optimal mapping $\mathbf{Y} \mapsto \mathbf{H}$
$\mathbf{H} \in \mathbb{C}^{K \times I}$	True channel grid $\{H[k, n]\}$
$\mathbf{H}_n$	Channel frequency response vector for the n-th OFDM symbol
$\widehat{\mathbf{H}}$	Estimated channel grid
$\widehat{\mathbf{H}}^{\mathrm{LS}}$	LS channel estimate from the LTS preamble
$\mathbf{h}^{(\ell)}$	Hidden activation vector at layer ℓ
I	Number of OFDM data symbols in the frame
K	FFT size (subcarriers per OFDM symbol)
$\mathbf{K}$	Key matrix in self-attention
K_d	Number of data subcarriers
K_p	Number of pilot subcarriers
k	Subcarrier index
$N_n[k]$	Complex AWGN sample on subcarrier k of OFDM symbol n
N	Number of training pairs $(\mathbf{X}'^{(i)}, \mathbf{H}^{(i)})$
N_{CP}	Cyclic prefix length (samples)
n	OFDM symbol index
$\mathbf{Q}$	Query matrix in self-attention
R_i	Relevance score for neuron/feature i (LRP)
$\mathbf{R}_{\mathrm{in}}$	Input relevance tensor (heatmap)
T_c	Coherence time
$\mathbf{V}$	Value matrix in self-attention

$\mathbf{W}^{(\ell)}$	Weight matrix at layer ℓ
$X_n[k]$	Transmitted symbol on subcarrier k of OFDM symbol n
$X_T[k]$	Known training symbol during LTS-based estimation
$\mathbf{X}'$	Neural network input feature tensor (distinct from transmitted symbols X)
$Y_n[k]$	Received frequency-domain signal on subcarrier k of OFDM symbol n
$\mathbf{Y} \in \mathbb{C}^{K \times I}$	Received frequency-domain grid $\{Y_n[k]\}$
$\mathcal{I}_{\text{interp}}(\cdot)$	Interpolation operator (linear, Lagrange, cubic spline)
$\mathcal{K}_0$	Set of null subcarriers
$\mathcal{K}_{\text{act}}$	Set of active subcarriers: $\mathcal{K}_{\text{act}} = \mathcal{K}_d \cup \mathcal{K}_p$
$\mathcal{K}_d$	Set of data subcarriers
$\mathcal{K}_p$	Set of pilot subcarriers
$\mathcal{L}$	Loss function (e.g., MSE, NMSE)
$\delta^{(\ell)}$	Dilation factor at layer ℓ (TCN)
ε	Approximation tolerance or stabilisation constant
κ	Convolutional kernel size
$\sigma^{(\ell)}(\cdot)$	Activation function applied at layer ℓ
σ_n^2	Noise variance per complex subcarrier
$\Re\{\cdot\}$	Real part operator
$\Im\{\cdot\}$	Imaginary part operator
$\|\cdot\|_F$	Frobenius norm
$\odot$	Elementwise (Hadamard) product

Chapter 1

Introduction

This chapter introduces the study and motivates its relevance for vehicular communication systems. We define the scope of the study, establish research questions, and provide an overview of our approach to developing accurate and efficient channel estimation solutions.

1.1 Background and motivation

Vehicular communication systems are a critical enabler of intelligent transportation systems, autonomous driving, and intelligent traffic management [1]. These applications fall under ultra-reliable low-latency communications (URLLC), which impose strict constraints on end-to-end delay and communication reliability. Safety-critical Vehicle-to-Everything (V2X) services, in particular, aim for end-to-end latencies below a few milliseconds and system-level packet-error-rate targets to ensure reliable operation in highly dynamic vehicular environments [2].

While achieving full vehicle-to-everything safety depends heavily on higher-layer protocols and burst-sensitive metrics, robust physical-layer performance is a foundational prereq-

uisite. Accurate channel estimation represents a necessary, but not sufficient, condition for meeting these broader system requirements. Validating link-level estimator performance and processing latency is therefore fundamental, as it dictates how effectively the underlying wireless channel can support the stringent reliability and latency budgets of the overall network.

The IEEE 802.11p standard, designed specifically for vehicular communications, employs an Orthogonal Frequency Division Multiplexing (OFDM) Physical Layer (PHY) inherited from IEEE 802.11a [3]. In vehicular environments, the challenge becomes particularly acute due to rapid vehicle mobility, which introduces time-varying Doppler shifts, whilst multipath propagation creates frequency-selective fading [4]. Classical channel estimators based on Least Squares (LS) and interpolation-based averaging degrade severely in fast time-varying channels because they assume quasi-static behaviour within an OFDM symbol and fixed correlation structures, assumptions that break down under high mobility and cause estimation errors [5].

Deep Learning (Deep Learning (DL)) has emerged as a transformative technology in various domains, demonstrating excellent pattern recognition capabilities and adaptability to complex nonlinear relationships. In wireless communications, DL approaches have shown promise for channel estimation by learning complex mappings from received signals to channel responses [6]–[9]. However, integrating DL into safety-critical vehicular systems introduces important challenges: many models exceed the computational limits of onboard hardware, their inference times may not satisfy millisecond-level latency constraints, and limited interpretability complicates verification and certification for deployment in unpredictable operating conditions [10]. These limitations motivate the development of channel estimation methods that maintain high link-level accuracy while operating with low computational latency for practical vehicular deployment.

This research investigates DL-based channel estimation for IEEE 802.11p vehicular communications, with the aim of improving the accuracy of the estimation under fast time-varying channel conditions while ensuring real-time efficiency.

1.2 Problem statement

Traditional channel estimation methods, while mathematically well-founded, face limitations when applied to high-mobility vehicular communication scenarios. They assume quasi-static or slowly varying channels, making them inadequate for environments where channel conditions change rapidly due to vehicle mobility, multipath propagation, and dynamic interference patterns [5]. As a result, these approaches struggle to maintain the accuracy required for reliable IEEE 802.11p-based V2X communications.

DL approaches demonstrate strong potential for improving channel estimation accuracy. However, several key research challenges remain:

1. **Architecture selection**: Several DL approaches have been proposed, but comparative evaluation of these approaches under consistent vehicular channel conditions is lacking. The trade-offs between estimation accuracy and inference latency remain unclear.

2. **Training strategy limitations**: Current DL approaches typically train DL estimators on single vehicular channel models and under high SNR conditions. The impact of training across diverse channel types and SNR levels on a model's generalisation to an unseen propagation environment remains under-explored.

3. **Real-time latency constraints**: Vehicular channels can exhibit short coherence times in dynamic highway and urban scenarios [11]. Meeting such system-wide constraints requires the physical layer to deliver channel estimates rapidly to maintain link reliability. This poses strict latency requirements specifically on DL inference, situating estimation latency as a critical bottleneck within the broader end-to-end communication delay. Therefore, careful architecture selection and optimisation are critical for a practical DL estimator.

4. **Interpretability to support efficient model design**: DL estimators provide limited insight into which input features are most relevant for accurate channel prediction and which internal parameters contribute meaningfully to model per-

formance. This lack of interpretability hinders principled reduction of redundant features and unnecessary model components, restricting opportunities to improve computational efficiency while maintaining estimation accuracy.

These challenges motivate an investigation into DL–based channel estimation for IEEE 802.11p, focusing on improving generalisation, reducing inference latency, and leveraging interpretability to guide efficient model design.

1.3 Project scope

To maintain focus and ensure that the research questions are addressed effectively, this study restricts its scope to the following aspects:

- **PHY layer channel estimation**: Since channel estimation occurs in the PHY layer of wireless communication systems, this work focuses exclusively on simulation and analysis at the PHY layer. Cross-layer optimisation and higher-layer network protocols are outside the scope of this work.
- **Vehicular wireless standard**: The research is confined to IEEE 802.11p, a widely adopted standard for Vehicle-to-Vehicle (V2V) communications, ensuring relevance to Intelligent Transportation Systems (ITS).
- **Propagation environments**: The study considers six vehicular channel scenarios that reflect a range of mobility and propagation conditions relevant to V2X communications. These include three Roadside-to-Vehicle (R2V) environments (Roadside-to-Vehicle Suburban Street (RTV-SS), Roadside-to-Vehicle Expressway (RTV-EX), Roadside-to-Vehicle Urban Canyon (RTV-UC)) and three V2V environments (Vehicle-to-Vehicle Same Direction With Wall (VTV-SDWW), Vehicle-to-Vehicle Urban Canyon (VTV-UC), Vehicle-to-Vehicle Expressway (VTV-EX)).
- **Dataset generation**: The scope of this study is restricted to synthetically gener-

ated data. All datasets are simulated using MATLAB[1] in accordance with vehicular channel specifications [12], allowing for controlled evaluation without the variability of real-world hardware impairments.

- **System configuration**: The study focuses on single-antenna Single-Input Single-Output (SISO) systems, excluding multi-antenna Multiple-Input Multiple-Output (MIMO) configurations and cooperative communication techniques.
- **Neural Network (NN) architectures**: The research investigates DL models relevant to sequence learning, specifically Feedforward Neural Networks (FNNs), Recurrent Neural Networks (RNNs) (Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs)), and Temporal Convolutional Networks (TCNs) for channel estimation.
- **Performance analysis**: The evaluation validates link-level model accuracy and communication performance using Normalised Mean Square Error (NMSE) and Bit Error Rate (BER), along with computational efficiency metrics such as Floating Point Operations (FLOPs), parameter count, and inference latency. While these classical averaged physical-layer metrics do not capture application-level burst errors, they serve as the necessary foundational benchmark for assessing the core viability of the proposed estimators.

1.4 Research questions

This thesis addresses four fundamental research questions:

RQ1: Limitations of classical estimation: *What are the performance limitations of conventional channel estimation techniques (LS, STA, TRFI, DPA) in high-mobility vehicular channels?*

RQ2: Architecture selection and performance evaluation: *Which DL architectures (FNN, LSTM, GRU, TCN) achieve better channel estimation performance for IEEE*

[1]MATLAB, version R2022b, The MathWorks, Inc., Natick, Massachusetts, United States

802.11p vehicular communications, and what are the trade-offs between estimation accuracy and inference latency?

RQ3: Generalisation through training data composition: *How does training data composition, specifically, mixed-SNR versus high-SNR training, and multi-channel versus single-channel training, affect model generalisation to unseen SNR levels and propagation environments?*

RQ4: Interpretability for model optimisation: *Can interpretability methods contribute to channel estimation and also support effective complexity reduction?*

1.5 Objectives

To address these research questions, the thesis pursues the following objectives:

O1: Simulation framework and baseline establishment Develop a reproducible simulation framework for IEEE 802.11p vehicular channels to enable benchmarking:

- Implement a compliant IEEE 802.11p PHY layer simulation to generate realistic datasets under diverse vehicular propagation conditions (V2V and R2V).
- Implement standard classical estimators (LS, STA, TRFI, CDP, and DPA) to serve as performance baselines.
- Establish the upper-bound performance limits and failure modes of these classical techniques in high-mobility scenarios.

O2: Architecture evaluation and training strategy optimisation Apply and evaluate DL architectures and training strategies for IEEE 802.11p vehicular channel estimation:

- Implement and optimise selected architectures from the literature: FNNs, LSTMs, GRUs, and TCNs architectures.

- Compare high-SNR versus mixed-SNR training (0–40 dB) for robustness under noise conditions.
- Evaluate single-channel versus multi-channel training for generalisation to unseen propagation environments.
- Conduct performance comparison using BER, NMSE, and inference latency metrics across the vehicular channel models.

O3: Performance-latency trade-off analysis Analyse computational feasibility of DL architectures for real-time vehicular deployment:

- Quantify inference latency, model parameters, and FLOPs for each architecture.
- Identify architectures that satisfy the coherence time (sub-millisecond latency) requirements whilst maintaining estimation accuracy.

O4: Interpretability-driven model compression for reduced inference latency

- Apply Layer-wise Relevance Propagation (LRP) to perform feature attribution on trained models.
- Use insights to reduce model complexity through selective feature removal and parameter pruning.
- Achieve lower computational complexity and faster inference time while maintaining acceptable BER performance.

1.6 Research methodology

This research employs an experimental methodology to ensure reproducible results.

Simulation environment: MATLAB is used to implement the IEEE 802.11p PHY layer specifications with vehicular channel models. Datasets are generated in-house covering six propagation scenarios (VTV-SDWW, RTV-SS, VTV-EX, VTV-UC, RTV-UC, RTV-EX).

Experimental Design: The methodology employs controlled experiments with systematic variation of key parameters, such as SNR levels, channel models, architectural configurations; and the use of random seeds to ensure reproducibility.

Comparative analysis: Performance is benchmarked against classical methods (LS, STA, TRFI) and existing DL approaches using BER, NMSE, model parameters, FLOPs, and inference latency as metrics.

Interpretability analysis: LRP is applied to trained models to identify important features and to inform input feature selection and model compression.

1.7 Thesis overview

This thesis presents a systematic investigation into DL-based channel estimation for vehicular communication systems. It is organised into seven interconnected chapters:

Chapter 1: Introduction establishes the research context, motivation, and objectives.

Chapter 2: Background. presents a detailed coverage of the fundamentals of channel estimation, vehicular communication systems, classical estimators, and existing DL approaches. This chapter establishes the theoretical foundation.

Chapter 3: Simulations and dataset generation. describes the simulation setup for implementing IEEE 802.11p and vehicular channel modelling, creating datasets for training and evaluating DL models.

Chapter 4: Deep learning architectures for channel estimation. presents and evaluates the selected sequential DL architectures for channel estimation.

Chapter 5: Robustness through noise-aware training. investigates the effect of training data composition on model robustness. Mixed-SNR training and multi-channel training are evaluated.

Chapter 6: Interpretability and complexity aware design. introduces the REACH framework, applying LRP for feature-level interpretability. Cross-channel relevance analysis identifies key features, enabling model compression through informed feature selection and parameter pruning.

Chapter 7: Conclusion. synthesises contributions and identifies future research directions. Each chapter builds on the previous one, progressing from architecture evaluation to optimisation, and finally to interpretability-driven model compression.

1.8 Publications

The research presented in this thesis has contributed to several peer-reviewed publications, each addressing specific aspects of DL-based vehicular channel estimation:

Peer-reviewed journal papers:

1. **Ngorima, S.A.**, Helberg, A., Davel, M.H. (2023). "Sequence Based Deep Neural Networks for Channel Estimation in Vehicular Communication Systems." *Artificial Intelligence Research. SACAIR 2023. Communications in Computer and Information Science*, vol. 1976, Springer, Cham.

2. **Ngorima, S.A.**, Helberg, A., Davel, M.H. (2024). "Neural Network-Based Vehicular Channel Estimation Performance: Effect of Noise in the Training Set." *Artificial Intelligence Research. SACAIR 2024. Communications in Computer and Information Science*, vol. 2326, Springer, Cham.

Peer-reviewed conference papers:

1. **Ngorima, S.A.**, Helberg, A., Davel, M.H. (2024). "A Data Pilot-Aided Temporal Convolutional Network for Channel Estimation in IEEE 802.11p Vehicle-to-Vehicle Communications." In *Proceedings of the Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2024*.

2. **Ngorima, S.A.**, Helberg, A., Davel, M.H. (2025). "Simplified Temporal Convolutional Based Channel Estimation for a WiFi Vehicular Communication Channel." In *Proceedings of the 2025 IEEE 3rd Wireless Africa Conference (WAC)*, Pretoria, South Africa, pp. 1–5.

The full published versions of these papers are provided in Appendices A.3.1 to A.3.4 in accordance with the ethical requirements of this study.

Chapter 2

Background

This chapter presents the background on DL-based channel estimation in IEEE 802.11p vehicular communications. It outlines the characteristics of wireless propagation in vehicular environments, describes the IEEE 802.11p OFDM system model, and reviews classical and DL-based channel estimation methods.

2.1 Introduction

Vehicular communication systems rely on accurate channel estimation to maintain communication quality under complex propagation conditions and high mobility. This chapter presents the theoretical and technical background for the DL-based estimation methods investigated in this thesis.

We begin with the fundamentals of wireless propagation in vehicular environments and then describe the IEEE 802.11p system model. IEEE 802.11p specifies the PHY/Medium Access Control (MAC) for vehicular ad hoc networks operating in the 5.9 GHz band. Throughout this thesis, the term IEEE 802.11p refers to the PHY/MAC parameters

relevant to channel estimation such as subcarrier spacing and OFDM symbol structure, while Dedicated Short-Range Communications (DSRC) and Wireless Access in Vehicular Environments (WAVE) denote the broader vehicular communication stack into which IEEE 802.11p is incorporated.

Next, we review classical channel estimation methods and discuss their limitations under vehicular conditions. This is followed by an overview of recent DL-based channel estimation strategies that aim to improve accuracy and robustness. We then introduce interpretability techniques used in DL models to enhance trust and support model optimisation. Finally, we present the performance metrics and evaluation criteria used in this thesis.

2.2 Wireless communication fundamentals

Wireless communication in vehicular environments introduces propagation challenges different from static scenarios, as the dynamic nature of vehicle mobility and the complexity of scattering environments create time-varying channels that challenge traditional estimation approaches. This section outlines the main mechanisms and models that characterise vehicular channels.

2.2.1 Multipath propagation and fading mechanisms

In vehicular communications, electromagnetic waves propagate through environments with multiple scatterers, including buildings, vehicles, vegetation, and terrain features. Consequently, the transmitted signal reaches the receiver via several mechanisms: direct path transmission, specular reflection from smooth surfaces resulting in moderate delay and attenuation, diffuse scattering from rough surfaces generating weaker and more dispersed signal components, and diffraction around obstacles causing amplitude reduction and phase rotation. Each mechanism introduces unique delays, attenuations, and phase shifts [13].

Figure 2.1 shows the propagation mechanisms in a typical vehicular scenario. The direct path (direct signal), when present, provides the strongest and least delayed signal. Reflected signals from buildings and road surfaces arrive with delays corresponding to the additional path length. Diffracted components, bent around obstacles, experience amplitude reduction and phase rotation. The superposition of these multipath components at the receiver produces the complex fading patterns that characterise vehicular channels.

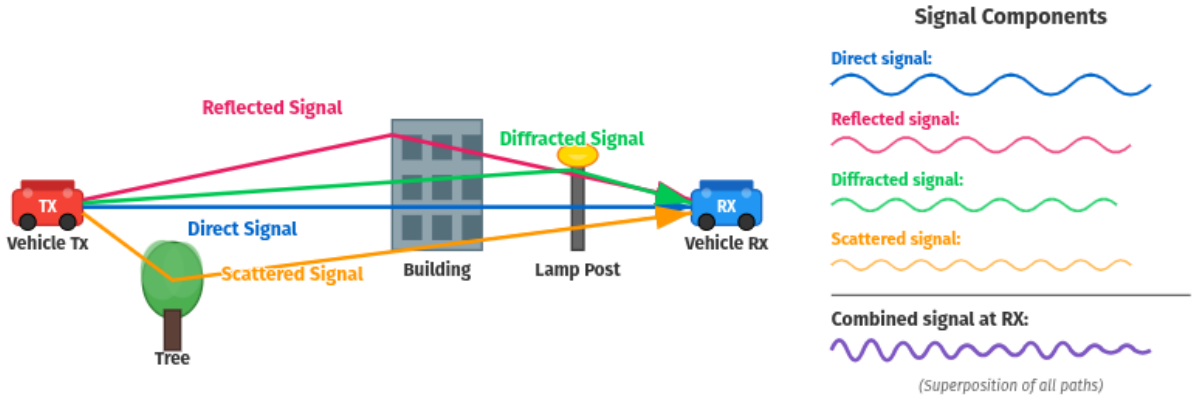


Figure 2.1: Multipath propagation in a vehicular V2V scenario.

Linear time-variant channel model

Vehicular channels are well described by Linear Time-Variant (LTV) models [14], as the mobility of transmitter, receiver, and surrounding scatterers leads to rapidly changing multipath conditions. In the continuous-time baseband domain, the channel impulse response is:

$$h(t,\tau) = \sum_{p=0}^{L-1} \alpha_p(t)\,\delta(\tau - \tau_p(t)), \tag{2.1}$$

where $\alpha_p(t)$ and $\tau_p(t)$ represent the complex gain and delay of the p-th propagation path at time t.

The received signal $r(t)$ is the convolution of the transmitted signal $s(t)$ with the channel impulse response plus additive noise:

$$r(t) = \int_{-\infty}^{\infty} h(t,\tau)s(t-\tau)d\tau + n(t), \tag{2.2}$$

where $h(t, \tau)$ represents the time-varying channel impulse response, τ denotes the propagation delay, and $n(t)$ is Additive White Gaussian Noise (AWGN) with power spectral density $N_0/2$.

In discrete time, sampled at period T_s, the impulse response becomes:

$$h[l, n] = \sum_{p=0}^{L-1} \alpha_p[n]\delta[l - \tau_p[n]], \tag{2.3}$$

where $\alpha_p[n] = |\alpha_p[n]|e^{j\phi_p[n]}$ represents the complex path gain, n is the time index, and $\tau_p[n]$ is the discrete normalised delay. Typical vehicular channels contain $L \approx$ 6-12 resolvable multipath components [15].

The frequency domain representation, obtained via the discrete Fourier transform, is:

$$H[k, n] = \sum_{p=0}^{L-1} \alpha_p[n]e^{-j2\pi k\tau_p[n]/K}, \tag{2.4}$$

where $k \in \{0, 1, \ldots, K-1\}$ indexes the subcarriers and K is the DFT size. In OFDM systems such as IEEE 802.11p, each subcarrier therefore experiences a distinct, time-varying channel gain $H[k, n]$.

The Wide-Sense Stationary Uncorrelated Scattering (WSSUS) assumption [14] is often applied to describe fading statistics, offering manageable correlation characteristics for multipath channels. Section 2.2.5 discusses WSSUS in detail.

2.2.2 Doppler effects in high mobility scenarios

Mobility in vehicular environments introduces Doppler frequency shifts that significantly alter the received signal spectrum. For a scatterer at angle θ relative to the direction of motion, the Doppler shift is given by

$$f_d = \frac{v\cos(\theta)f_c}{c}, \tag{2.5}$$

where v is the relative velocity, f_c is the carrier frequency (5.9 GHz for IEEE 802.11p) and c the speed of light. Maximum Doppler shift occurs when $\theta = 0°$:

$$f_{d,\max} = \frac{vf_c}{c}. \tag{2.6}$$

Several Doppler spectrum models have been proposed to characterise the frequency-domain distribution of time variation in wireless channels, including Jakes, Gaussian, and measured vehicular spectra. Among these, the classical Jakes model [16, pp. 19–22] is widely adopted as a reference because of its analytical tractability and realistic representation of isotropic scattering. Assuming isotropic scattering, the Doppler PSD follows the classical Clarke-Jakes model [16], resulting in the characteristic U-shaped spectrum shown in Fig. 2.2:

$$S_D(f_d) = \begin{cases} \dfrac{\sigma_h^2}{\pi f_{d,\max}\sqrt{1-(f_d/f_{d,\max})^2}}, & |f_d| \le f_{d,\max}, \\ 0, & \text{elsewhere}, \end{cases} \tag{2.7}$$

where σ_h^2 denotes the average channel power. This spectrum exhibits singularities at the edges $|f_d| = f_{d,\max}$.

The inverse Fourier transform of the Doppler spectrum yields the temporal autocorrelation function [14]:

$$r_h(\Delta t) = \sigma_h^2 J_0(2\pi f_{d,\max}\Delta t), \tag{2.8}$$

where $J_0(\cdot)$ is the zeroth-order Bessel function of the first kind. The coherence time T_c, which quantifies how rapidly the channel varies, can be defined as the time at which this autocorrelation drops to a specified threshold (commonly 0.5). For the Jakes model, this leads to the widely used approximation [16]:

$$T_c \approx \frac{9}{16\pi f_{d,\max}} = \frac{0.423}{f_{d,\max}}. \tag{2.9}$$

Table 2.1 summarises the maximum Doppler shifts and the corresponding coherence times for typical vehicular speeds at the IEEE 802.11p carrier frequency of 5.9 GHz. A large T_c corresponds to a slowly varying channel with small Doppler spread, while a small T_c indicates rapid time variation.

As shown in Table 2.1, the coherence time T_c decreases with increasing velocity. In IEEE 802.11p, the duration of the OFDM symbol is $T_{\text{sym}} \approx 8$ μs, which remains much shorter than T_c even at highway speeds. Thus, the channel can be regarded as quasi-static

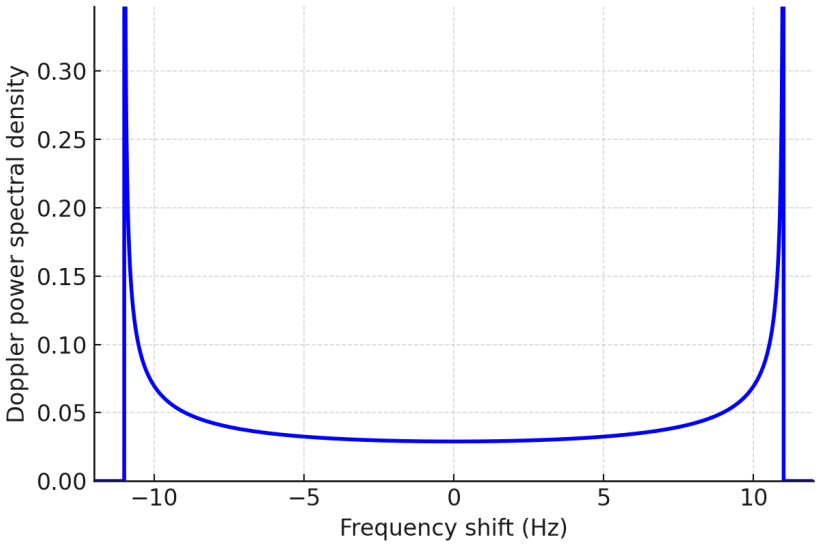


Figure 2.2: Doppler PSD of the classical Jakes model under isotropic scattering, showing the characteristic U-shaped form.

Table 2.1: Maximum Doppler shifts and coherence times for IEEE 802.11p at 5.9 GHz, using $T_c \approx 0.423/f_{d,\max}$.

Velocity (km/h)	$f_{d,\max}$ **(Hz)**	T_c **(ms)**
30	164	2.58
60	328	1.29
120	655	0.65
200	1093	0.39

within a single OFDM symbol. However, Doppler spread still destroys the orthogonality between subcarriers, resulting in Inter-Carrier Interference (ICI). Moreover, the reduced coherence time at high mobility degrades the reliability of pilot-based channel tracking across consecutive symbols. These limitations motivate the use of advanced tracking and learning-based channel estimation methods in vehicular environments.

2.2.3 Delay spread and frequency selectivity

Multipath components with different delays create frequency selectivity in the channel response. The Root Mean Square Delay Spread (RMS-DS) is defined as [4, Ch. 5]

$$\tau_{\text{rms}} = \sqrt{\frac{\sum_p \tau_p^2 |\alpha_p|^2}{\sum_p |\alpha_p|^2} - \left(\frac{\sum_p \tau_p |\alpha_p|^2}{\sum_p |\alpha_p|^2} \right)^2}, \tag{2.10}$$

where τ_p and α_p denote the delay and complex gain of the p-th multipath component, respectively. The RMS-DS determines the channel coherence bandwidth, defined as the frequency range over which the channel response is highly correlated [4, Ch. 5]:

$$B_c \approx \frac{1}{2\pi\tau_{\text{rms}}}. \tag{2.11}$$

When B_c is comparable to or smaller than the signal bandwidth, the channel exhibits frequency-selective fading, meaning that different frequency components experience uncorrelated fading. In such cases, adjacent subcarriers in OFDM systems experience different channel gains, necessitating more sophisticated channel estimation approaches beyond simple per-subcarrier methods. Vehicular channels typically exhibit significant frequency selectivity, particularly in urban environments where dense multipath propagation results in large delay spreads [12], [17].

2.2.4 Doubly selective channel model

Vehicular wireless channels are generally doubly selective, which means that they exhibit variations in both time and frequency domains. As established in Section 2.2.1, the LTV channel model (2.3) captures both effects: multipath propagation (indexed by l) causes frequency selectivity, while temporal variation (indexed by n) introduces time selectivity through Doppler effects [4].

The degree of selectivity depends on the relationship between channel parameters and signal characteristics. Frequency selectivity becomes significant when the coherence bandwidth B_c defined in (2.11) is comparable to or smaller than the signal bandwidth. Time

selectivity occurs when the coherence time T_c as defined in (2.9) approaches or becomes shorter than the transmission duration.

For IEEE 802.11p, the OFDM symbol duration ($T_{\text{sym}} = 8\ \mu$s) is typically much shorter than the coherence time in vehicular scenarios, allowing the channel to be treated as quasi-static within each symbol. However, significant variation occurs across consecutive symbols, particularly at highway speeds. This temporal variation introduces ICI and degrades pilot-based channel tracking across frames [12], [17].

The received OFDM symbol at subcarrier k and symbol index i, using the frequency-domain model from (2.4), can be expressed as:

$$\tilde{y}_i[k] = H[k,i]\,\tilde{x}_i[k] + \tilde{e}_{i,d}[k] + \tilde{v}_i[k], \tag{2.12}$$

where $H[k,i]$ is the channel frequency response, $\tilde{x}_i[k]$ is the transmitted symbol, $\tilde{e}_{i,d}[k]$ represents Doppler-induced interference between adjacent subcarriers, and $\tilde{v}_i[k]$ is AWGN.

2.2.5 Statistical channel models

For analysis and simulation, vehicular channels are often approximated by statistical models. A widely adopted framework is the WSSUS model [14], which assumes local stationarity over short time intervals (on the order of tens of milliseconds, depending on scenario) and uncorrelated multipath components in delay [4], [18].

Wide-Sense Stationary Uncorrelated Scattering assumptions

Under WSSUS [14] as formulated in [4, Ch. 6], the channel $h(t,\tau)$ satisfies two key properties:

1. **WSSUS:** first and second order statistics are time-invariant over the local stationarity interval, such that the autocorrelation function

$$\mathbb{E}\{h(t,\tau)\} = \mu_h(\tau), \quad R_h(\Delta t,\tau) = \mathbb{E}\{h(t,\tau)h^*(t+\Delta t,\tau)\}, \tag{2.13}$$

where $\mu_h(\tau)$ denotes the mean channel impulse response at delay τ, and $R_h(\Delta t, \tau)$ is the channel autocorrelation function evaluated at delay τ and time lag $\Delta t = t_2 - t_1$. The operator $(\cdot)^*$ represents the complex conjugate, and $\mathbb{E}\{\cdot\}$ denotes the statistical expectation. Under the Wide-Sense Stationary (WSS) assumption, R_h depends only on the time difference (Δt) rather than on the absolute time instants, consistent with [4, Ch. 6].

2. **Uncorrelated Scattering (US):** Multipath components with different delays are uncorrelated:

$$\mathbb{E}\{h(t, \tau_1)h^*(t, \tau_2)\} = P_h(\tau_1)\,\delta(\tau_1 - \tau_2). \tag{2.14}$$

The channel is then characterised by the scattering function $S_h(\tau, f_d)$, the 2-D Doppler delay power spectrum. For a sparse discrete-path representation,

$$S_h(\tau, f_d) = \sum_{p=0}^{L-1} P_p\,\delta(\tau - \tau_p)\,S_{D,p}(f_d), \tag{2.15}$$

where P_p is the average power of path p with delay τ_p and Doppler PSD $S_{D,p}(f_d)$, describing the distribution of channel power over Doppler frequencies [4, Ch. 6].

These assumptions enable tractable statistical characterisation through the scattering function, which describes the distribution of channel power across delay and Doppler frequency. For the Tapped Delay Line (TDL) models used in vehicular simulations, WSSUS implies that different taps are statistically independent and that each tap follows a Doppler spectrum (typically Jakes-type) as discussed in Section 2.2.2 [4], [18].

2.2.6 Channel amplitude distributions

The small-scale fading amplitude follows different statistical distributions depending on the propagation environment. Two cases are particularly relevant for vehicular communications:

2.2.6.1 Rayleigh fading

In Non-Line-of-Sight (NLOS) propagation environments with a large number of independent scatterers and no dominant path, the complex channel coefficient h at a given subcarrier and time instant can be modelled as a zero-mean circularly symmetric complex Gaussian random variable. The channel envelope $r = |h|$ then follows a Rayleigh distribution [4, Ch. 5]:

$$p_R(r) = \frac{r}{\sigma^2} \exp\Big(- \frac{r^2}{2\sigma^2} \Big), \quad r \geq 0, \tag{2.16}$$

where σ^2 is the variance of each real and imaginary component ($\mathbb{E}\{|h|^2\} = 2\sigma^2$). The phase $\phi = \angle h$ is uniformly distributed over $[0, 2\pi)$, and the instantaneous power $|h|^2$ follows an exponential distribution [4, Ch. 5].

2.2.6.2 Rician fading

In environments where a dominant Line-of-Sight (LOS) component is present in addition to diffuse multipath, the channel envelope follows a Rician distribution [4, Sec. 5.5.2]:

$$p_{\mathrm{Ric}}(r) = \frac{r}{\sigma^2} \exp\Big(- \frac{r^2 + A^2}{2\sigma^2} \Big) I_0\Big(\frac{rA}{\sigma^2} \Big), \tag{2.17}$$

where A is the LOS amplitude, σ^2 is the variance per dimension of the scattered components, and $I_0(\cdot)$ denotes the modified Bessel function of the first kind of zero order [4, Ch. 5]. The Rician K-factor quantifies the ratio of LOS to scattered power:

$$K = \frac{A^2}{2\sigma^2}. \tag{2.18}$$

As $K \to 0$, the Rician distribution converges to the Rayleigh case, corresponding to a purely NLOS environment with no dominant LOS component, while larger K corresponds to stronger LOS dominance [4, Ch. 5]. Figure 2.3 illustrates the Rayleigh and Rician fading distributions for different K-factors, highlighting how increasing K skews the envelope toward higher amplitudes due to the stronger LOS component.

These statistical models, together with the WSSUS framework, provide tractable abstractions of vehicular channels for analysis and the generation of training data in deep

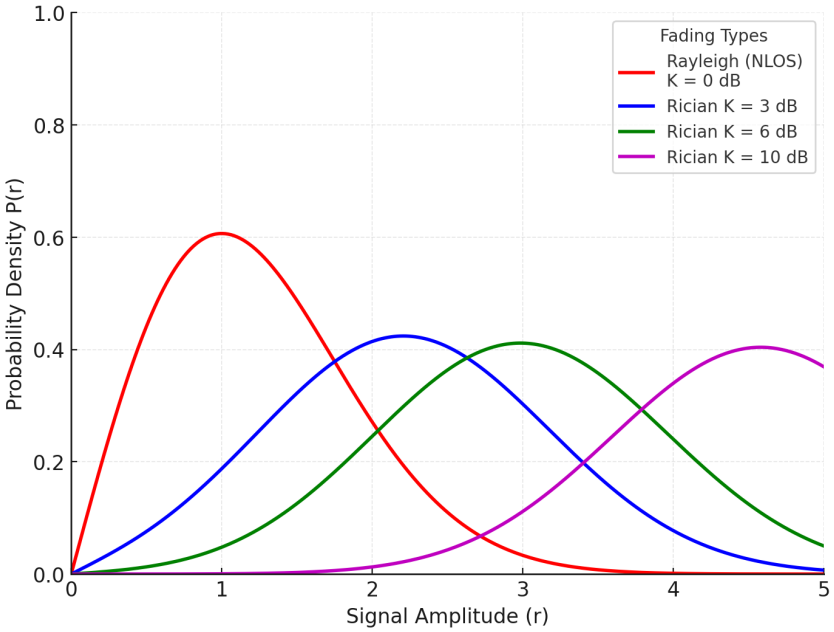


Figure 2.3: Rayleigh and Rician fading distributions for different K-factors.

learning-based channel estimation [4], [18].

2.3 Vehicular communication systems

ITS transform vehicles into connected nodes within a coordinated communication framework. The characteristics of the vehicular channel such as doubly selective fading (Subsection 2.2.4), high Doppler spreads (Subsection 2.2.2), and time-varying multipath (Subsection 2.2.1), directly impact the design of the V2X system and motivate the deep learning-based channel estimators developed in this thesis.

2.3.1 Vehicle-to-Everything (V2X) paradigms

V2X comprises four primary link types with different propagation and estimation challenges (Fig. 2.4): V2V, Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P) and Vehicle-to-Network (V2N).

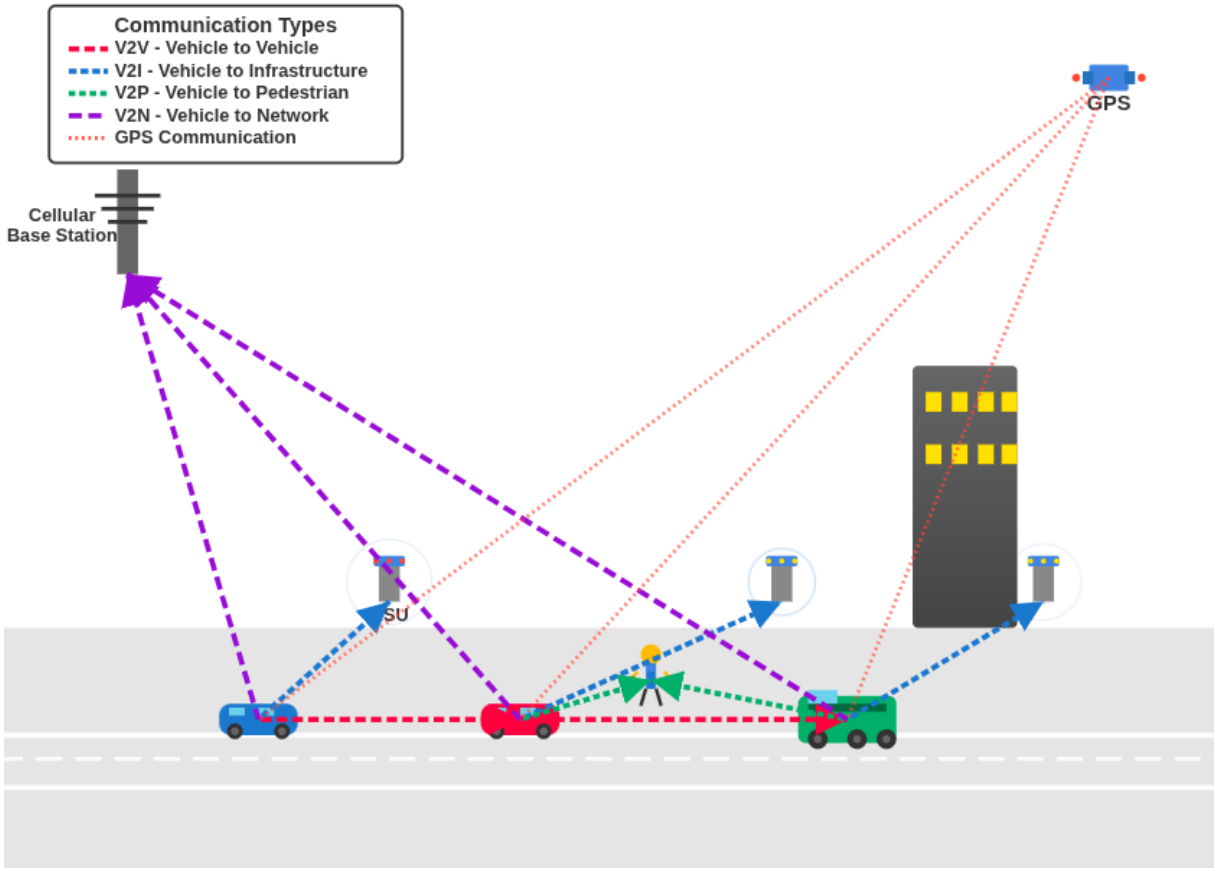


Figure 2.4: V2X communication paradigms.

Vehicle-to-Vehicle (V2V) (dual mobility). Both endpoints move, resulting in an increased effective Doppler shift and a rapidly changing geometry. With relative speed v_{rel}, the maximum Doppler shift is

$$f_{d,\max}^{\text{V2V}} = \frac{v_{\text{rel}} f_c}{c}. \tag{2.19}$$

For opposing traffic at 120 km/h each ($v_{\text{rel}} = 66.7$ m/s) and $f_c = 5.9$ GHz, this yields $f_{d,\max} \approx 1.31$ kHz. The corresponding coherence time $T_c \approx 0.32$ ms is much longer than the OFDM symbol duration ($T_{\text{sym}} \approx 8$ μs), allowing quasi-static approximation within symbols but requiring tracking across consecutive symbols [17].

Vehicle-to-Infrastructure (V2I) (single-sided mobility). Vehicles communicate with Road Side Units (RSUs) installed at elevated positions (3-6 m) at intersections and highway access points. LOS components are common, delay spreads are moderate, and

the Doppler shift is lower than V2V for the same vehicle speed.

Vehicle-to-Pedestrian (V2P) (human-carried devices). Connections between vehicles and pedestrian devices (smartphones, wearables) are dominated by the vehicle's motion rather than the pedestrian's. At $f_c = 5.9$ GHz, a walking speed of 1 m/s produces only about 20 Hz Doppler shift, while a vehicle at 30 m/s induces nearly 600 Hz [4, Ch. 6]. Body shadowing and occlusions cause power fluctuations and intermittent connectivity [17].

Vehicle-to-Network (V2N) (cellular backhaul). Connectivity leverages cellular infrastructure (4G/5G) for wide-area communication, enabling cloud-based services and network-assisted positioning. Since only the vehicle is in motion, the Doppler shift is equal to $f_d = vf_c/c$, yielding $f_d \approx 650$ Hz at 5.9 GHz and 120 km/h, about half that of a V2V link under the same conditions. The resulting coherence time, $T_c \simeq 0.423/f_d \approx 0.65$ ms, implies that the channel remains approximately constant over several OFDM symbols but changes across successive subframes [4, Ch. 6]. Path loss and shadowing dominate large-scale variation, while small-scale fading follows cellular channel statistics under vehicular mobility.

2.3.2 Safety requirements and metrics

Safety-related V2X applications impose stringent latency and reliability constraints that directly affect the PHY-layer design of IEEE 802.11p and Cellular Vehicle-to-Everything (C-V2X) systems. Two major standardisation frameworks define these requirements:

(i) SAE J2945/1 (U.S.) specifies the on-board broadcast of Basic Safety Messages (BSMs) at 10 Hz (every 100 ms), with congestion control based on transmit power and rate adaptation [19].

(ii) ETSI TR 102 638 (Europe) defines similar message classes such as Cooperative Awareness Messages (CAMs) and Decentralised Environmental Notification Messages

(DENMs) [20].

Although the application layer disseminates situational awareness updates at 10 Hz, the underlying wireless channel varies much faster. As shown in Section 2.2.2, coherence times at highway speeds are 0.3-0.6 ms, several hundred times shorter than the BSM interval. Thus, reliable PHY-layer operation demands accurate channel estimation and equalisation at the OFDM symbol level ($T_{\text{sym}} \approx 8\ \mu$s) despite the slower messaging cadence.

The standards further target high packet reception ratios and strict end-to-end latencies for safety-critical events such as emergency braking and collision warnings. While the channel estimators developed in this thesis are evaluated using link-level physical-layer metrics rather than burst-sensitive application-level metrics, meeting these physical-layer processing constraints is a crucial prerequisite. The latency introduced by channel estimation acts as a fundamental bottleneck within the wider end-to-end communication delay. Therefore, while low-latency, high-reliability channel estimation is a necessary condition for vehicular safety, it provides the foundational link-level robustness upon which higher-layer protocols rely, rather than acting as a sufficient guarantee of system-wide safety.

Representative end-to-end Quality of Service (QoS) targets derived from SAE J2945/1 [19] and ETSI TR 102 638 [20] are summarised in Table 2.2. These parameters describe the latency, update rate, and reliability requirements for the core V2X safety applications. Reliability is typically quantified using the Packet Reception Ratio (PRR), defined as the probability that a transmitted message is correctly received within a specified distance (commonly 300 m).

2.3.3 Standardised vehicular channel models

Accurate channel models support the analysis, simulation and DL training. We adopt TDL models derived from extensive measurements (the Atlanta campaign [12]), which capture the doubly selective behaviour and WSSUS characteristics described in Sections 2.2.4 and 2.2.5.

Table 2.2: Representative QoS targets for selected V2X applications.

Application	Latency (ms)	Update Rate (Hz)	Reliability/Notes
Emergency braking (EEBL)	< 100	10	PRR $\gtrsim 90\%$ @ 300 m
Collision/hazard warning	< 100	10	PRR $\gtrsim 90\%$ @ 300 m
Cooperative awareness	100-300	1-10	Situational awareness
Traffic optimisation	≤ 500	0.1-1	Infrastructure-assisted
Infotainment	Seconds	On-demand	Throughput-driven

IEEE 802.11p reference models

We employ six measured V2X scenarios from standardised vehicular channel measurement campaigns [12]. These models represent two main categories: V2V and R2V communications across diverse environments. The V2V scenarios (Vehicle-to-Vehicle (VTV) prefix) model communication between moving vehicles, whilst R2V scenarios (RTV prefix) model communication from fixed roadside units to mobile vehicles. Table 2.3 summarises the key characteristics of all six models, including the number of taps, vehicle velocity, maximum Doppler shift, average path gains, and path delays.

The six models span diverse propagation environments:

- **RTV-SS:** Suburban road side to vehicle with moderate mobility (32 to 48 km/h, 500 Hz Doppler shift) and intermediate delay spread (up to 700 ns).
- **RTV-EX:** High-mobility scenario (104 km/h, 700 Hz Doppler shift) with roadside infrastructure communicating to vehicles on expressways with relatively short delay spread (401 ns), reflecting open propagation environments.
- **RTV-UC:** Dense urban canyon (32-48 km/h, 300 Hz Doppler shift) with delay spread (501 ns) due to strong multipath from surrounding buildings.
- **VTV-UC:** V2V communication in urban canyons where both the transmitter and the receiver are mobile. The relative motion between vehicles produces 500 Hz Doppler shift despite moderate individual speeds (32–48 km/h), with delay spread

Table 2.3: Vehicular channel model characteristics following Jakes' Doppler spectrum [12].

Channel model	**Taps**	**Velocity [km/h]**	**Max Doppler [Hz]**	**Average path gains [dB]**	**Path delays [ns]**
Roadside-to-Vehicle (R2V) Scenarios					
RTV-SS	12	32–48	500	[0, 0, −9.3, −9.3, −14, −14, −18, −18, −19.4, −24.9, −27.5, −29.8]	[0, 1, 100, 101, 200, 201, 300, 301, 400, 500, 600, 700]
RTV-EX	12	104	700	[0, 0, 0, −9.3, −9.3, −9.3, −20.3, −20.3, −21.3, −21.3, −28.8, −28.8]	[0, 1, 2, 100, 101, 102, 200, 201, 300, 301, 400, 401]
RTV-UC	12	32–48	300	[0, 0, 0, −11.5, −11.5, −11.5, −19, −19, −25.6, −25.6, −28.1, −28.1]	[0, 1, 2, 100, 101, 102, 200, 201, 300, 301, 500, 501]
Vehicle-to-Vehicle (V2V) Scenarios					
VTV-UC	12	32–48	500	[0, 0, −10, −10, −10, −17.8, −17.8, −17.8, −21.1, −21.1, −26.3, −26.3]	[0, 1, 100, 101, 102, 200, 201, 202, 300, 301, 400, 401]
VTV-SDWW	12	104	1150	[0, 0, −11.2, −11.2, −19, −21.9, −25.3, −25.3, −24.4, −28, −26.1, −26.1]	[0, 1, 100, 101, 200, 300, 400, 401, 500, 600, 700, 701]
VTV-EX	11	104	1200	[0, 0, 0, −6.3, −6.3, −25.1, −25.1, −25.1, −22.7, −22.7, −22.7]	[0, 1, 2, 100, 101, 200, 201, 202, 300, 301, 302]

up to 401 ns.

- **VTV-SDWW:** High-speed V2V scenario (104 km/h) on expressways with centre dividing walls. The wall creates additional reflections, producing the highest Doppler shift (1150 Hz) and delay spread to 701 ns.
- **VTV-EX:** Open expressway V2V (104 km/h, 1200 Hz Doppler shift) with the highest Doppler shift due to direct relative motion but shorter delay spread (302

ns) from sparse scattering.

These profiles exhibit clustered delays around discrete values (0, 100, 200, 300 ns) representing distinct scattering clusters, with path powers generally decaying with delay [12].

2.3.4 Dedicated Short-Range Communications (DSRC) architecture

The 5.9 GHz band dedicated to ITS is regulated regionally by bodies such as the U.S. Federal Communications Commission (FCC) and the European Conference of Postal and Telecommunications Administrations (CEPT). In the United States, the historical allocation spans 5.850-5.925 GHz (75 MHz) and is divided into seven 10 MHz channels [21]. Channel 178 (5.890 GHz) serves as the Control Channel (CCH), while Channels 172, 174, 176, 180, 182, and 184 are designated as Service Channels (SCHs), as shown in Fig. 2.5. Adjacent 10 MHz channels can be bonded to form a 20 MHz operational mode. However, 10 MHz operation is typically preferred in vehicular environments because the doubled OFDM symbol duration improves robustness against multipath delay spread and Doppler effects [22]. For comparison, the European ITS-G5 system operates over 5.875-5.905 GHz (30 MHz), divided into three 10 MHz channels.

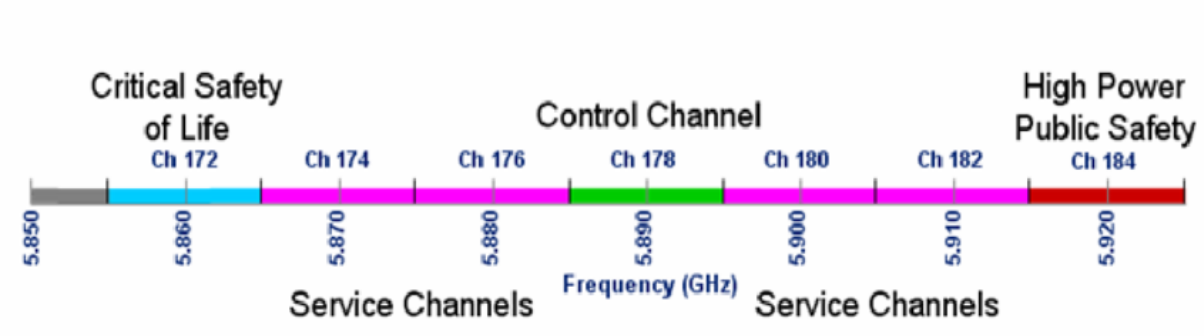


Figure 2.5: DSRC spectrum allocation in the 5.9 GHz band (U.S. FCC plan).

Multichannel coordination and timing. IEEE 1609.4 defines a time-division channel coordination mechanism in which all stations synchronise to a global 10 Hz (100 ms)

synchronisation interval derived from Coordinated Universal Time (UTC). Each interval is subdivided into a CCH period and an SCH period, typically 50 ms each, separated by short guard intervals to allow for radio retuning and context switching [21]. During the CCH period, vehicles exchange high-priority safety beacons such as BSMs along with control and management frames. During the SCH period, application data and infotainment traffic are carried on the selected SCHs. These alternating intervals ensure the temporal separation between safety-critical and non-safety communications, as shown in Fig. 2.6. The 100 ms synchronisation cadence determines how often transceivers must re-acquire

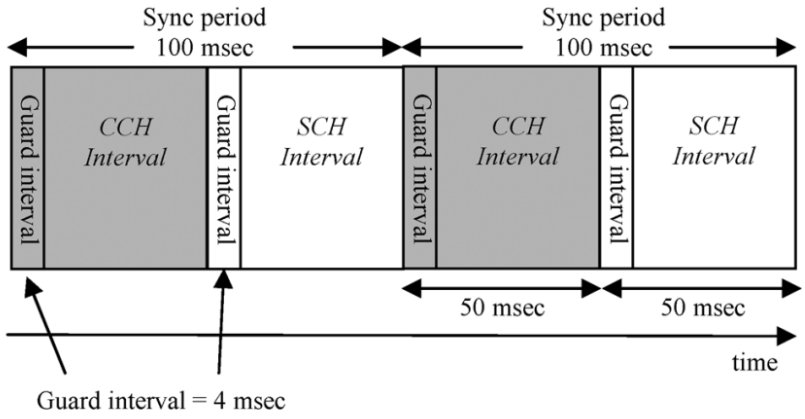


Figure 2.6: Multichannel coordination in DSRC showing the 100 ms synchronisation interval alternating between CCH and SCH periods with guard intervals for channel switching (adapted from [21]).

or track the channel state. Given that vehicular channel coherence times can be of milliseconds at highway speeds, the retuning process introduces an inherent need for rapid channel re-estimation and temporal tracking across successive CCH/SCH transitions.

Medium Access: Enhanced Distributed Channel Access (EDCA). Medium access in DSRC is governed by IEEE 802.11e Enhanced Distributed Channel Access (EDCA), which prioritises traffic using four Access Categories (AC) with distinct Arbitration Inter-Frame Spaces (AIFS) and contention window (CW) parameters [23]. The mean channel access delay for a given AC can be approximated by

$$\mathbb{E}[T_{\text{access}}] \;=\; \text{AIFS}[AC] \;+\; \frac{\text{CW}[AC]}{2}\, T_{\text{slot}}, \tag{2.20}$$

where AIFS[AC] is the arbitration inter-frame space, CW[AC] the contention window, and T_{slot} the slot duration. Safety beacons are assigned to the highest-priority access category (typically `AC_VO`), which uses the smallest AIFS and contention window to minimise latency and ensure timely channel access for safety-critical transmissions.

2.4 OFDM and IEEE 802.11p system

OFDM forms the PHY layer of IEEE 802.11p and mitigates the impact of multipath fading while maintaining spectral efficiency. IEEE 802.11p is an amendment to IEEE 802.11 (derived from 802.11a) that adapts the PHY and MAC layers for vehicular delay and Doppler conditions in DSRC and ITS applications [21], [22]. This section reviews the OFDM mathematical framework, key IEEE 802.11p parameter adaptations, and the transceiver and frame structures that define the channel estimation challenges studied in this thesis.

2.4.1 OFDM principles

This subsection outlines the core signal-processing concepts behind OFDM, focusing on subcarrier orthogonality, discrete-time modulation using the Inverse Fast Fourier Transform (IFFT), and the use of the Cyclic Prefix (CP) to preserve circular convolution in multipath channels.

2.4.1.1 Orthogonality and spectral efficiency

An OFDM signal divides the available bandwidth into N orthogonal subcarriers, each modulated by a low-rate data stream. Let the useful symbol duration be T_u and the subcarrier spacing $\Delta f = 1/T_u$. The subcarrier spacing is given by

$$\Delta f \;=\; \frac{1}{T_u} \;=\; \frac{B_s}{N}, \tag{2.21}$$

where B_s is the sampling bandwidth associated with the N-point Fast Fourier Transform (FFT) grid. In time-invariant channels, subcarriers remain orthogonal; however, time selectivity due to Doppler shifts perturbs this orthogonality, introducing ICI [24].

The continuous-time baseband OFDM waveform (before CP insertion) can be expressed as

$$s(t) = \sum_{i=0}^{\infty} \sum_{k=0}^{N-1} X_i[k]\, e^{j2\pi k \Delta f\,(t - iT_s)}\, \mathrm{rect}\left(\frac{t - iT_s}{T_u}\right), \tag{2.22}$$

where $s(t)$ is the transmitted baseband waveform, $X_i[k]$ denotes the complex data symbol on the k-th subcarrier of the i-th OFDM symbol, and $T_s = T_u + T_{\mathrm{CP}}$ is the total symbol duration including the CP. The rectangular pulse $\mathrm{rect}(\cdot)$ defines each symbol over $[0, T_u)$, ensuring non-overlapping time intervals [24].

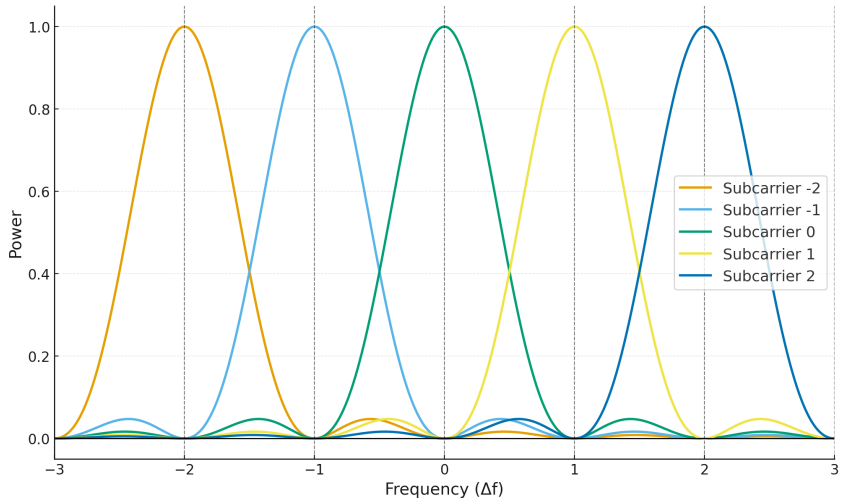


Figure 2.7: OFDM subcarrier orthogonality.

Figure 2.7 shows the principle of subcarrier orthogonality in OFDM systems. The frequency response of each subcarrier follows a sinc^2 power spectrum, with its main lobe peak aligned to the zero-power crossings of adjacent subcarriers. However, in high-mobility vehicular channels, Doppler spread distorts these spectra, shifting and broadening the lobes, breaking orthogonality and introducing ICI that degrades system performance [4], [25].

2.4.1.2 Discrete implementation via Inverse Fast Fourier Transform (IFFT)

In practical systems, OFDM modulation is implemented digitally via an Inverse Discrete Fourier Transform (IDFT). Let the frequency domain vector $\mathbf{X} = [X[0], X[1], \ldots, X[N-1]]^T$ contain the complex constellation symbols mapped onto the N subcarriers (data, pilot, and null tones as shown in Fig. 2.8). The discrete-time OFDM signal is then

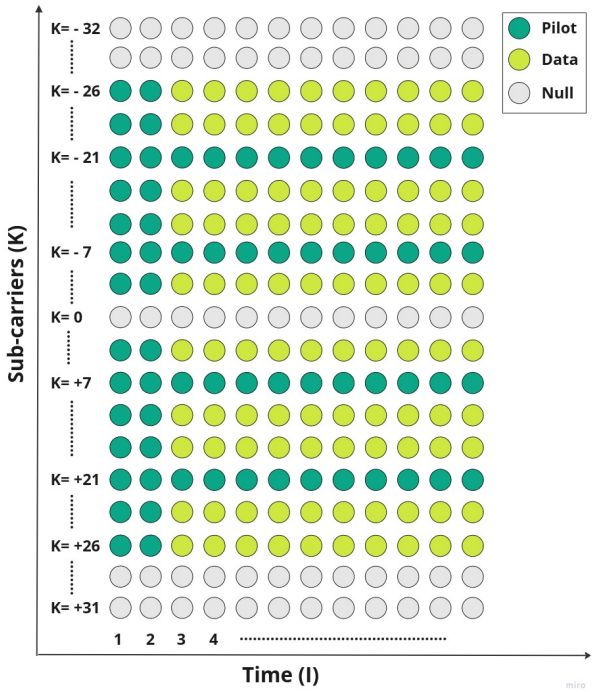


Figure 2.8: IEEE 802.11p OFDM frame in the frequency domain [6].

generated using an N-point IFFT [4, Ch. 19] as

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi}{N}kn}, \quad n = 0, \ldots, N-1, \tag{2.23}$$

which converts the frequency-domain symbols to the time-domain sequence transmitted over the channel. At the receiver, an N-point FFT recovers the subcarrier-domain representation $X[k]$ from $x[n]$.

The computational complexity of a direct N-point DFT is $\mathcal{O}(N^2)$, whereas the Cooley Tukey FFT algorithm [26] reduces this to $\mathcal{O}(N \log_2 N)$ by recursively decomposing the transform into smaller subtransforms [27]. This reduction is essential for real-time

embedded implementations; the required IFFT size N is determined by the number of subcarriers and thus directly impacts the computational load.

2.4.1.3 Cyclic prefix and multipath mitigation

The CP extends the OFDM symbol by copying the last N_{CP} samples to its front, effectively transforming a linear convolution into a circular convolution, as shown in Figure 2.9. This eliminates Inter-Symbol Interference (ISI) provided that

$$T_{\mathrm{CP}} > \tau_{\max}, \tag{2.24}$$

where $\tau_{\max}$ is the maximum channel delay spread. For IEEE 802.11p, $T_{\mathrm{CP}} = 1.6\,\mu$s accommodates the delay spreads typically observed in vehicular environments [21].

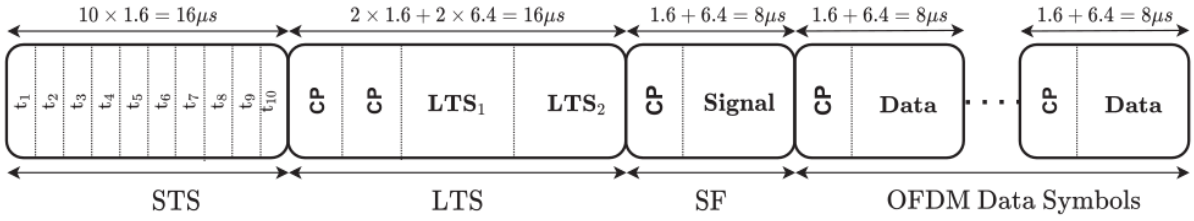


Figure 2.9: IEEE 802.11p frame structure in time domain [17].

2.4.2 IEEE 802.11p physical layer specifications

IEEE 802.11p operates on a 10 MHz channel bandwidth, which is half that of IEEE 802.11a, thereby doubling the time-domain parameters (symbol and CP durations) and halving the subcarrier spacing to improve robustness against Doppler shift [21], [22]. The key parameters of the PHY-layer are summarised in Table 2.4.

The supported coding rates (1/2, 2/3, and 3/4) define the ratio of information bits to encoded bits produced by the convolutional encoder. Higher code rates provide higher throughput but reduced error protection, directly influencing link reliability [21]. The

Table 2.4: Key IEEE 802.11p PHY parameters

Parameter	Value
Channel bandwidth	10 MHz
FFT size (N_{FFT})	64
Subcarrier spacing	156.25 kHz
Total / active subcarriers	64 / 52
Data / pilot / null subcarriers	48 / 4 / 12
Useful symbol duration (T_u)	6.4 μs
Cyclic prefix duration (T_{CP})	1.6 μs
Total symbol duration (T_{sym}=T_u+T_{CP})	8.0 μs
Pilot subcarriers	$\{-21, -7, 7, 21\}$
Modulation	BPSK, QPSK, 16-QAM, 64-QAM
Convolutional coding rate	1/2, 2/3, 3/4

achievable data rate is given by

$$R_b = \frac{K_{\mathrm{d}} \cdot m \cdot r}{T_{\mathrm{sym}}}, \tag{2.25}$$

where K_{d} is the number of data subcarriers, m is the number of bits per symbol (modulation order), and r is the coding rate [28].

2.4.3 Transceiver architecture and signal processing chain

Figure 2.10 shows the complete IEEE 802.11p transceiver architecture, with each processing block serving specific functions in the communication chain. The IEEE 802.11p baseband chain (Fig. 2.10) follows the conventional OFDM transceiver pipeline with adaptations for vehicular environments.

Transmitter processing. The transmitter scrambles input bits to ensure uniform power spectral density, applies convolutional coding (constraint length 7, rate 1/2 base code with puncturing for rates 2/3 and 3/4), interleaves coded bits across subcarriers to

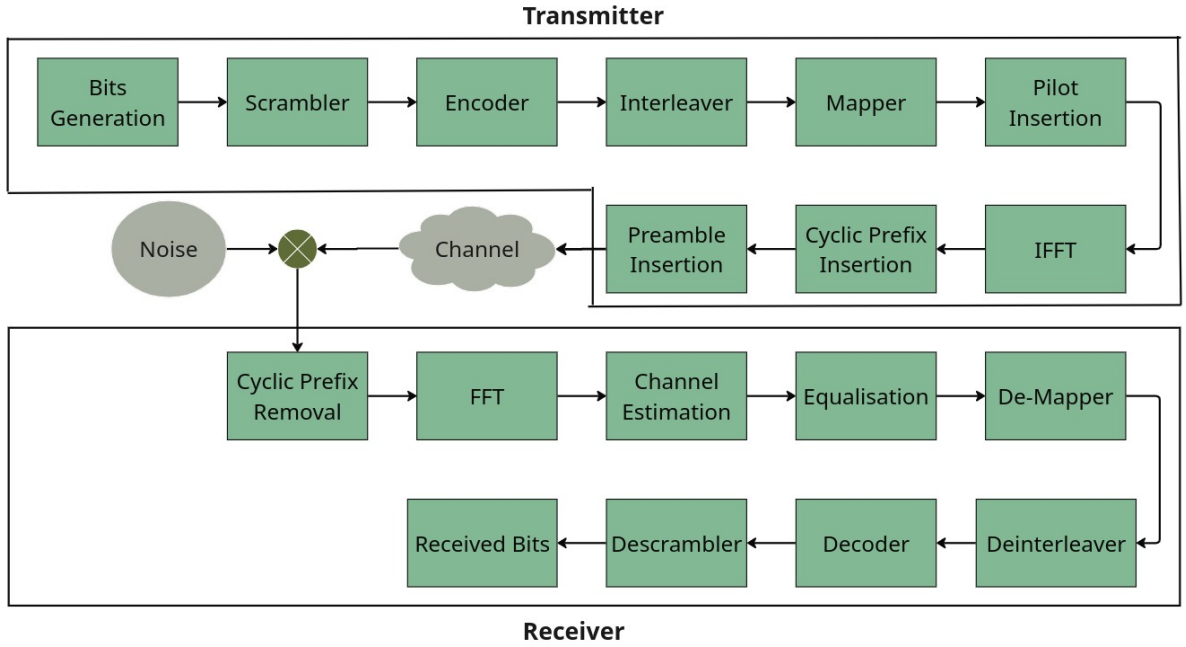


Figure 2.10: IEEE 802.11p baseband transceiver chain (PHY) [21].

protect against burst errors, maps them to complex Quadrature Amplitude Modulation (QAM) symbols using Gray coding, inserts four pilot tones at subcarrier indices $\{-21, -7, 7, 21\}$ for phase tracking and channel estimation, and applies a 64-point IFFT with CP insertion to generate the transmitted time-domain waveform [21], [28].

Receiver processing. The receiver performs Automatic Gain Control (AGC), packet detection, timing and frequency synchronisation, CP removal, 64-point FFT, channel estimation, equalisation, QAM demapping, deinterleaving and Viterbi decoding [21]. The accuracy of channel estimation directly impacts equalisation performance and is the primary focus of this thesis.

2.4.4 Frame structure and preamble design

An IEEE 802.11p Physical Layer Protocol Data Unit (PPDU) comprises a preamble, a SIGNAL symbol, and a DATA field [21], [28]. The preamble enables packet detection, AGC, synchronisation, and initial channel estimation in the receiver front-end. Figure 2.9

shows the complete frame structure.

Short Training Symbol (STS). The Short Training Symbol (STS) comprises ten repetitions of a 16-sample waveform, totalling 16 μs. It supports packet detection, diversity selection, and coarse Carrier Frequency Offset (CFO) and timing estimation.

Long Training Symbol (LTS). The Long Training Symbol (LTS) comprises a 3.2 μs CP followed by two identical 6.4 μs OFDM symbols (total 16 μs), each carrying a known Binary Phase Shift Keying (BPSK) pattern on the 52 active subcarriers. After CP removal and FFT at the receiver, the two LTS observations $Y_{T1}[k]$ and $Y_{T2}[k]$ enable per-subcarrier LS channel estimation by averaging, providing the initial Channel State Information (CSI) for data symbol equalisation [21], [28].

Signal Field (SF). The SIGNAL Field (SF) is a single BPSK-modulated OFDM symbol (rate 1/2 coding) that conveys the Modulation and Coding Scheme (MCS) and payload length for the subsequent DATA symbols [28].

DATA field and pilot structure. Following the SF, the DATA field comprises N_{SYM} OFDM symbols carrying the encoded MAC Protocol Data Unit (MPDU). Four pilot subcarriers at indices $\{-21, -7, 7, 21\}$ are embedded in each DATA symbol to enable phase tracking, residual CFO correction, and pilot-aided channel tracking across symbols. The pilot symbols follow a pseudorandom polarity pattern (defined by the IEEE 802.11 scrambler with initial state) to avoid spectral lines and improve tracking robustness [28].

The pilot spacing of 14 subcarriers corresponds to $14 \times 156.25\,\mathrm{kHz} \approx 2.19\,\mathrm{MHz}$, which is comparable to or may exceed the coherence bandwidth observed in vehicular channels (Section 2.2.3). When the coherence bandwidth falls below this spacing, interpolation of CSI across subcarriers becomes unreliable, motivating the use of learning-based channel estimation approaches that exploit both spatial (frequency) and temporal structure [21].

2.4.5 Signal model

This subsection presents the discrete-time baseband signal model that serves as the analytical framework for channel estimation throughout this thesis. We adopt the standard IEEE 802.11 OFDM model [24], [27], [28] and assume perfect timing and carrier synchronisation for analytical tractability. The SF is omitted as our focus is on the DATA symbol channel estimation.

We consider a PPDU comprising two LTS followed by N_{SYM} OFDM data symbols (Fig. 2.9). The number of data symbols depends on the payload size, modulation order, and coding rate. For example, an Ethernet payload of 1500 bytes may require tens to hundreds of OFDM symbols depending on the selected MCS [21], [28].

2.4.5.1 Frequency-domain symbol structure

Let $K = 64$ denote the FFT size for IEEE 802.11p operation on a 10 MHz channel [28]. The subcarrier allocation comprises $K_{\mathrm{d}} = 48$ data subcarriers, $K_{\mathrm{p}} = 4$ pilot subcarriers, and $K_0 = 12$ null subcarriers (DC and guard tones). The frequency-domain OFDM symbol at time index n is thus

$$X_n[k] = \begin{cases} X_n^{\mathrm{d}}[k], & k \in K_{\mathrm{d}}, \\ X_n^{\mathrm{p}}[k], & k \in K_{\mathrm{p}}, \\ 0, & k \in K_0. \end{cases} \tag{2.26}$$

where $\mathcal{K}_{\mathrm{d}}$, $\mathcal{K}_{\mathrm{p}}$, and $\mathcal{K}_0$ denote the data set, pilot, and null subcarrier indices, respectively.

2.4.5.2 Time-domain transmission

The corresponding time-domain OFDM symbol is generated using a K-point IDFT, typically implemented by an IFFT:

$$x_n[m] = \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} X_n[k]\, e^{j2\pi mk/K}, \quad m = 0, \ldots, K-1. \tag{2.27}$$

This unitary normalisation ($1/\sqrt{K}$) preserves the signal energy between time and frequency domains (Parseval's theorem)[27, Ch. 5]. A CP of $N_{\mathrm{CP}} = 16$ samples (1.6 μs) is prepended to each symbol to mitigate ISI, as specified in the IEEE 802.11p PHY [28].

2.4.5.3 Channel model and received signal

After removal of the CP, the received time-domain OFDM symbol can be expressed as a discrete LTV convolution:

$$y_n[m] = \sum_{l=0}^{L-1} h[l, n]\, x_n[m - l] + w_n[m], \quad m = 0, \ldots, K-1, \tag{2.28}$$

where $y_n[m]$ is the m-th time-domain sample of the n-th received OFDM symbol, $h[l, n]$ is the l-th tap of the multipath channel impulse response at symbol index n, L is the number of channel taps, and $w_n[m]$ represents complex AWGN with variance σ_w^2. This model is consistent with the WSSUS framework and TDL representation discussed in Sections 2.2.5 and 2.3.3, where each tap varies over time according to its Doppler spectrum as characterised by the scattering function in Equation (2.15) [4, Ch. 6]. The convolution in (2.28) captures the joint effects of multipath delay spread (indexed by l) and Doppler frequency shift (variation with n), both characteristic of vehicular propagation.

Frequency domain input and output relation. Applying a K-point FFT to (2.28) yields the frequency-domain relationship

$$Y_n[k] \;=\; \sum_{q=0}^{K-1} C_{kq}[n]\; X_n[q] \;+\; V_n[k], \tag{2.29}$$

where $Y_n[k]$ is the received symbol at subcarrier k and OFDM symbol index n, $C_{kq}[n]$ denotes the subcarrier coupling coefficient induced by time selectivity, and $V_n[k]$ is the FFT of the noise sequence.

Under the quasi-static within-symbol assumption (Section 2.2.4), valid when the Doppler spread f_{D} is small compared to the subcarrier spacing $\Delta f = 156.25$ kHz, inter-carrier coupling becomes negligible ($C_{kq}[n] \approx 0$ for $q \neq k$), resulting in

$$Y_n[k] \;\approx\; H[k, n]\, X_n[k] \;+\; V_n[k], \tag{2.30}$$

where $H[k,n] \triangleq C_{kk}[n]$ represents the Channel Frequency Response (CFR) at subcarrier k and OFDM symbol n. This per-subcarrier model forms the basis for most conventional channel estimation and equalisation approaches.

In mobility scenarios, such as vehicular communications with large Doppler spreads, the off-diagonal coefficients $C_{kq}[n]$ become significant, introducing ICI:

$$Y_n[k] = H[k,n]\, X_n[k] + \sum_{q \neq k} C_{kq}[n]\, X_n[q] + V_n[k]. \tag{2.31}$$

This general model captures both flat and frequency-selective fading effects in time-varying vehicular channels and serves as the reference for subsequent channel estimation analyses presented.

2.5 Channel estimation fundamentals

This section establishes the background required for analysing and designing high performance channel estimators. We begin by formulating the channel estimation objective in the OFDM frequency domain, followed by a discussion structured by the type of information used: we begin with classical methods that rely solely on the preamble, followed by tracking techniques that use pilots and data within the frame. Finally, we discuss advanced methods designed to mitigate error propagation in data-aided tracking.

2.5.1 Problem formulation

Channel estimation aims to reconstruct the channel response so that the receiver can compensate for path loss, shadowing, multipath fading, and Doppler shifts. In an OFDM system with perfect within-symbol synchronisation, a per-subcarrier diagonal model [4] is given by Equation (2.30).

Given observations $\{Y_n[k]\}_{k \in \mathcal{K}_{\text{act}}}$ on the active subcarriers $\mathcal{K}_{\text{act}}$, the channel estimate $\widehat{H}[k,n]$ is often obtained by minimising Mean Square Error (MSE) over the active sub-

carriers:

$$\widehat{\mathbf{H}}_n = \arg\min_{\tilde{\mathbf{H}}} \sum_{k \in \mathcal{K}_{\text{act}}} \mathbb{E}\Big[\big|Y_n[k] - \tilde{H}[k]\, X_n[k]\big|^2\Big], \tag{2.32}$$

with $\mathbf{H}_n := \{H[k,n]\}_{k \in \mathcal{K}_{\text{act}}}$. Estimation quality limits post-equalisation SNR, which in turn determines BER for the chosen modulation [27].

2.5.2 Preamble-based estimators

Preamble-based estimators rely exclusively on the known Long Training Symbol (LTS) located at the start of the packet. They assume the channel remains static for the duration of the frame and do not exploit pilots or data symbols embedded in the payload.

Least Squares (LS) Estimation. The LS estimation method is the most widely adopted channel estimation technique in wireless communications due to its computational simplicity and implementation efficiency [29], [30]. It estimates the channel frequency response by minimising the squared error between the received and expected pilot symbols without requiring statistical knowledge of the channel or noise.

According to the IEEE 802.11 OFDM PHY specification [28], two identical LTS are transmitted at the start of each frame. Each LTS is first constructed in the frequency domain across the 52 active subcarriers (48 data and 4 pilot tones) and then converted to the time domain using a K-point IFFT, as described in Section 2.4.4. At the receiver, after coarse and fine frequency-offset correction, the CP is removed, and a K-point FFT is applied to recover the frequency-domain LTS observations $Y_{T1}[k]$ and $Y_{T2}[k]$ on each active subcarrier $k \in \mathcal{K}_{\text{act}}$. The per-subcarrier LS channel estimate is then obtained by averaging over the two LTS symbols:

$$\widehat{H}_{\text{LS}}[k] \;=\; \frac{Y_{T1}[k] + Y_{T2}[k]}{2\, X_T[k]}, \qquad k \in \mathcal{K}_{\text{act}}. \tag{2.33}$$

where $X_T[k]$ is the known frequency-domain training symbol defined by the standard. Under AWGN with variance σ_n^2 per complex subcarrier, the mean-squared estimation

error is

$$\mathrm{MSE}_{\mathrm{LS}} = \sigma_n^2 / \left(2\,|X_T[k]|^2\right) \tag{2.34}$$

which doubles if only one LTS is used. The LS method exhibits linear complexity $\mathcal{O}(|\mathcal{K}_{\mathrm{act}}|)$, and is computed once per frame. The resulting channel estimate $\widehat{H}_{\mathrm{LS}}[k]$ is then applied to all subsequent OFDM data symbols within that frame for equalisation. Although computationally efficient, the LS estimator does not exploit any statistical knowledge of the channel or noise and assumes that it remains constant over the two training symbols and the subsequent data duration. In high-mobility vehicular scenarios where the coherence time is comparable to or shorter than the frame duration, this assumption no longer holds, leading to rapid degradation in estimation accuracy [4, Ch. 19.4.2].

Minimum Mean Squared Error (MMSE) Estimation. The MMSE estimator refines the LS estimate by incorporating second-order statistics of the channel and noise to minimise the expected estimation error [29], [31]. Unlike LS, which assumes no prior statistical information, the MMSE estimator exploits the spatial or frequency correlation of the channel to achieve improved accuracy. In the per-subcarrier (diagonalised) OFDM model, a linear MMSE refinement of the preamble-based LS estimate is expressed as

$$\widehat{\mathbf{H}}_{\mathrm{MMSE}} = \mathbf{R}_{HH}\left(\mathbf{R}_{HH} + \frac{\sigma_n^2}{|X_T|^2}\,\mathbf{I}\right)^{-1} \widehat{\mathbf{H}}_{\mathrm{LS}}, \tag{2.35}$$

where $\mathbf{R}_{HH} = \mathbb{E}[\mathbf{H}\mathbf{H}^H]$ denotes the channel covariance matrix across active subcarriers, and $\sigma_n^2/|X_T|^2$ represents the LS error variance per subcarrier. For unit-magnitude training symbols, this term reduces to σ_n^2. The corresponding mean-squared estimation error is

$$\mathrm{MSE}_{\mathrm{MMSE}} = \mathrm{tr}\left(\mathbf{R}_{HH} - \mathbf{R}_{HH}\left(\mathbf{R}_{HH} + \frac{\sigma_n^2}{|X_T|^2}\,\mathbf{I}\right)^{-1} \mathbf{R}_{HH}\right). \tag{2.36}$$

While MMSE estimation achieves lower error variance than LS, it requires accurate knowledge of the channel covariance matrix and noise variance, as well as a matrix inversion whose computational complexity scales as $\mathcal{O}(|\mathcal{K}_{\mathrm{act}}|^3)$. Consequently, its practical implementation in vehicular OFDM systems is limited, since the underlying covariance statistics may vary rapidly and become outdated within a single frame [32]. In such cases, LS or adaptive low-rank approximations of MMSE are often preferred trade-offs between accuracy and complexity.

Maximum Likelihood (ML). The ML approach selects the channel that maximises the likelihood of the received signal given the known pilot/data symbols [33]. Under the per-subcarrier (diagonal) OFDM model with AWGN, the Maximum Likelihood (ML) solution is equivalent to the LS estimate and therefore offers no performance advantage. In more general receiver models that account for ICI, residual carrier-frequency offset, or coloured noise, the ML estimator can, in principle, outperform LS [34]. However, it requires joint optimisation over both the channel and impairment parameters, which is computationally intensive and highly sensitive to model mismatch.

Due to these complexity and robustness constraints, especially in fast-varying vehicular channels, this thesis adopts the LS estimator as the practical baseline for subsequent analysis.

2.5.3 Pilot-based and data-aided estimation

Most practical OFDM systems employ pilot-based estimation, where a subset of subcarriers carries known symbols for channel observation [29]. The channel frequency response at data subcarriers is then reconstructed by interpolation between the pilot subcarriers. The pilot pattern for IEEE 802.11p was defined and discussed in Section 2.4.4. Unlike the preamble-based methods, the techniques in this subsection utilises the pilots and data subcarriers distributed throughout the time-frequency grid. This allows the receiver to update the channel estimate for every OFDM symbol, accommodating time-varying channel conditions that preamble-only methods cannot capture.

2.5.3.1 Pilot interpolation

Using the four pilot subcarriers $\{-21, -7, 7, 21\}$ shown in Fig. 2.8, the pilot-only baseline estimates the channel on data subcarriers by interpolating $\widehat{H}[k]$ between adjacent pilot estimates. Common approaches include linear and higher-order (Lagrange) interpo-

lation [35]:

$$\text{Linear:} \quad \widehat{H}[k] = \widehat{H}[k_p] + \frac{k - k_p}{k_{p+1} - k_p}\big(\widehat{H}[k_{p+1}] - \widehat{H}[k_p]\big), \tag{2.37}$$

$$\text{Lagrange:} \quad \widehat{H}[k] = \sum_{i=0}^{N_p-1} \widehat{H}[k_{p,i}] \prod_{\substack{j=0 \\ j\neq i}}^{N_p-1} \frac{k - k_{p,j}}{k_{p,i} - k_{p,j}}, \tag{2.38}$$

where $k_{p,i}$ denotes the i-th pilot subcarrier index and $N_p = 4$ for IEEE 802.11p. Linear interpolation is computationally efficient but assumes a slowly varying frequency response between pilots. Lagrange interpolation captures higher-order variations, but its sensitivity to noise and estimation errors at pilot locations limits its robustness in high-mobility environments [29].

2.5.3.2 Data-Pilot Aided (DPA) estimation

Data-aided methods exploit information from demapped data subcarriers in addition to pilots. Data-Pilot Aided (DPA) assumes short-term temporal correlation across adjacent OFDM symbols in vehicular channels [6]. For each OFDM symbol n the steps are:

1. **Initial estimation (LS on the preamble):**

$$\widehat{H}_0^{\mathrm{DPA}}[k] \;=\; \widehat{H}_{\mathrm{LS}}[k], \qquad k \in K_{\mathrm{act}}\ (:= K_d \cup K_p). \tag{2.39}$$

2. **Equalisation and demapping for OFDM symbol n:**

$$d_n[k] \;=\; \begin{cases} X_n^{\mathrm{p}}[k], & k \in K_p \quad \text{(known pilot)}, \\ \mathfrak{D}\left(\dfrac{Y_n[k]}{\widehat{H}_{n-1}^{\mathrm{DPA}}[k]}\right), & k \in K_d \quad \text{(nearest-constellation demap)}. \end{cases} \tag{2.40}$$

3. **Channel update using pilots and demapped data:**

$$\widehat{H}_n^{\mathrm{DPA}}[k] \;=\; \frac{Y_n[k]}{d_n[k]}, \qquad k \in K_{\mathrm{act}}. \tag{2.41}$$

Although DPA improves channel tracking between pilots, it remains sensitive to noise and susceptible to error propagation across symbols. In this context, tracking refers to the

ability of the estimator to follow the temporal evolution of the channel response across consecutive OFDM symbols within a frame. Due to imperfect symbol decisions and time selectivity, estimation errors accumulate over time [7]. The accumulated estimation error can be expressed as:

$$\epsilon_n \;=\; \epsilon_{n-1} + \Delta\epsilon_n \;\approx\; \epsilon_0 + n\,\bar{\Delta}\epsilon, \tag{2.42}$$

where $\bar{\Delta}\epsilon$ is the average incremental error per symbol.

2.5.4 Advanced classical methods

The basic DPA method discussed in Section 2.5.3 is susceptible to error propagation, where a single decision error corrupts subsequent estimates. The methods presented here, Spectral Temporal Averaging (STA) [36], Constructed Data Pilots (CDP) [37], and Time-domain Reliable Frequency-domain Interpolation (TRFI) [38], address this limitation by introducing reliability metrics and consistency checks to filter the channel update, thereby reducing the propagation of errors across the frame.

2.5.4.1 Spectral Temporal Averaging (STA)

STA [36] leverages both time and frequency correlation across OFDM symbols to refine channel estimates. It applies frequency-domain averaging to smooth variations between adjacent subcarriers, followed by time-domain averaging to track gradual changes across symbols. In practice, STA is applied to initial estimates (e.g., from DPA), excluding edge subcarriers to avoid boundary effects, thereby improving the estimation accuracy in fast-varying channels.

Frequency domain averaging is applied first as follows:

$$\widehat{H}_n^{\mathrm{FD}}[k] = \sum_{\lambda=-\beta}^{\beta} \omega_\lambda\, \widehat{H}_n^{\mathrm{DPA}}[k+\lambda], \qquad \omega_\lambda = \frac{1}{2\beta+1}. \tag{2.43}$$

Next, time averaging is employed to calculate the final STA channel estimate:

$$\widehat{H}_n^{\mathrm{STA}}[k] = \left(1 - \tfrac{1}{\alpha}\right) \widehat{H}_{n-1}^{\mathrm{STA}}[k] + \tfrac{1}{\alpha}\, \widehat{H}_n^{\mathrm{FD}}[k], \qquad \widehat{H}_0^{\mathrm{STA}}[k] = \widehat{H}_0^{\mathrm{DPA}}[k]. \tag{2.44}$$

Here, β controls the frequency smoothing window and $\alpha \geq 1$ sets the temporal smoothing strength (larger α gives stronger smoothing but slower tracking). STA effectively reduces noise at the cost of reduced tracking capability for rapid channel variations.

2.5.4.2 Constructed Data Pilots (CDP)

CDP reduces error propagation in data-aided estimation by exploiting the temporal correlation between adjacent OFDM symbols [37]. It updates the current channel only when a consistency check across time passes. Otherwise, it reuses the last reliable estimate. CDP operates on data subcarriers while treating pilot subcarriers as always reliable.

Let $\mathcal{K}_d$ and $\mathcal{K}_p$ denote the data and pilot subcarrier sets, respectively. Initialise with the preamble-based LS estimate

$$\widehat{H}_0^{\mathrm{CDP}}[k] = \widehat{H}^{\mathrm{LS}}[k], \quad k \in \mathcal{K}_d \cup \mathcal{K}_p.$$

For $n \geq 1$, form the DPA update $\widehat{H}_n^{\mathrm{DPA}}[k]$ and perform a temporal consistency test using the *previous* received symbol $Y_{n-1}[k]$:

$$Y_{n-1}^{\mathrm{eq}'}[k] = \frac{Y_{n-1}[k]}{\widehat{H}_n^{\mathrm{DPA}}[k]}, \tag{2.45}$$

$$Y_{n-1}^{\mathrm{eq}''}[k] = \frac{Y_{n-1}[k]}{\widehat{H}_{n-1}^{\mathrm{CDP}}[k]}, \tag{2.46}$$

and demap to nearest constellation points with the slicer $D(\cdot)$:

$$d'_{n-1}[k] = D\big(Y_{n-1}^{\mathrm{eq}'}[k]\big), \tag{2.47}$$

$$d''_{n-1}[k] = D\big(Y_{n-1}^{\mathrm{eq}''}[k]\big). \tag{2.48}$$

On data subcarriers $k \in \mathcal{K}_d$, update by

$$\widehat{H}_n^{\mathrm{CDP}}[k] = \begin{cases} \widehat{H}_{n-1}^{\mathrm{CDP}}[k], & d'_{n-1}[k] \neq d''_{n-1}[k] \quad \text{(inconsistent; keep past)}, \\ \widehat{H}_n^{\mathrm{DPA}}[k], & d'_{n-1}[k] = d''_{n-1}[k] \quad \text{(consistent; accept new)}. \end{cases} \tag{2.49}$$

On pilot subcarriers $k \in \mathcal{K}_p$, set $\widehat{H}_n^{\mathrm{CDP}}[k] = \widehat{H}_n^{\mathrm{DPA}}[k]$.

The consistency test uses $\widehat{H}_n^{\mathrm{DPA}}$ to validate decisions on symbol $n-1$, introducing a one-symbol latency but preserving causality in streaming receivers. CDP is lightweight, requiring only two equalisations and two hard decisions per data subcarrier ($k \in \mathcal{K}_d$) and is robust to isolated decision errors. However, at low SNR or with high-order constellations, the consistency test is less reliable, limiting gains.

2.5.4.3 Time-domain Reliable Frequency-domain Interpolation (TRFI)

TRFI [38] improves DPA estimation by reconstructing data subcarriers considered unreliable by interpolating the frequency domain using a set of reliable subcarriers, rather than freezing them (i.e., retaining the previous estimate without update). The method operates in two stages: first, subcarriers are classified as reliable or unreliable based on a temporal consistency test. Second, reliable subcarriers accept the DPA update while unreliable ones are reconstructed via interpolation from the reliable set.

Classification. Let $\widehat{H}_n^{\mathrm{DPA}}[k]$ be the DPA update on symbol n (Eqs. (2.40)-(2.41)) and $\widehat{H}_{n-1}^{\mathrm{TRFI}}[k]$ the previous TRFI estimate. Equalise the previous received symbol $Y_{n-1}[k]$ twice:

$$Y_{n-1}^{\mathrm{eq}\,\prime}[k] = \frac{Y_{n-1}[k]}{\widehat{H}_n^{\mathrm{DPA}}[k]}, \tag{2.50}$$

$$Y_{n-1}^{\mathrm{eq}\,\prime\prime}[k] = \frac{Y_{n-1}[k]}{\widehat{H}_{n-1}^{\mathrm{TRFI}}[k]}. \tag{2.51}$$

The equalised symbols are then demapped to the nearest constellation points using the hard decision operator $D(\cdot)$:

$$d'_{n-1}[k] = D\left(Y_{n-1}^{\mathrm{eq}\,\prime}[k]\right), \tag{2.52}$$

$$d''_{n-1}[k] = D\left(Y_{n-1}^{\mathrm{eq}\,\prime\prime}[k]\right). \tag{2.53}$$

The reliable and unreliable data subcarrier sets are declared as:

$$\mathcal{K}_{\mathrm{rel},\,n-1} = \{k \in \mathcal{K}_d :\ d'_{n-1}[k] = d''_{n-1}[k]\}, \tag{2.54}$$

$$\mathcal{K}_{\mathrm{unrel},\,n-1} = \mathcal{K}_d \setminus \mathcal{K}_{\mathrm{rel},\,n-1}. \tag{2.55}$$

The pilot subcarriers are included in the reliable pool:

$$\mathcal{K}^{\star}_{\mathrm{rel},\,n-1} = \mathcal{K}_{\mathrm{rel},\,n-1} \cup \mathcal{K}_p. \tag{2.56}$$

This classification scheme introduces only one symbol latency and preserves causality.

Update. For subcarriers deemed reliable ($k \in \mathcal{K}^{\star}_{\mathrm{rel},\,n-1}$), TRFI directly adopts the current DPA estimate:

$$\widehat{H}_n^{\mathrm{TRFI}}[k] = \widehat{H}_n^{\mathrm{DPA}}[k]. \tag{2.57}$$

For unreliable subcarriers ($k \in \mathcal{K}_{\mathrm{unrel},\,n-1}$), the channel response is reconstructed by interpolating the neighbouring reliable subcarriers on the same OFDM symbol:

$$\widehat{H}_n^{\mathrm{TRFI}}[k] = \mathcal{I}_{\mathrm{cubic}}\Big(k;\ \{(m, \widehat{H}_n^{\mathrm{TRFI}}[m]) : m \in \mathcal{K}^{\star}_{\mathrm{rel},\,n-1}\}\Big), \tag{2.58}$$

where $\mathcal{I}_{\mathrm{cubic}}(\cdot)$ denotes one-dimensional cubic spline interpolation along the subcarrier index, using the reliable subcarriers $\mathcal{K}^{\star}_{\mathrm{rel},\,n-1}$ (pilots included) as reference points [39]. The spline uses natural end conditions, i.e., zero second derivative at the first and last reference points. If fewer than three reliable neighbours are available, the method reverts to linear interpolation.

TRFI typically outperforms both STA and CDP because unreliable data subcarriers are reconstructed rather than frozen [38]. Performance is high at moderate-to-high SNR and for higher-order QAM. Performance degrades when the reliable set becomes sparse (low SNR or pronounced frequency selectivity), since the cubic interpolant becomes underconstrained. Moreover, TRFI still inherits demapping errors from the DPA step. Complexity is modest (one extra equalisation/slicing pass and a 1-D spline per symbol).

2.5.5 Integration approaches in channel estimation

In this subsection, we briefly discuss how the channel estimation methods can be integrated into practical OFDM receivers. Let $\mathbf{Y} \in \mathbb{C}^{K \times I}$ represent the received time-frequency grid over K subcarriers and I OFDM data symbols, where K is the FFT size and I is the number of data symbols per frame. Let $\mathbf{y}_n \in \mathbb{C}^K$ represent the received signal on the

n-th OFDM symbol, and $\mathbf{H} \in \mathbb{C}^{K \times I}$ the true channel grid. Each integration approach provides different trade-offs between computational complexity, latency, and estimation accuracy.

2.5.5.1 Symbol by symbol estimation

In Symbol-by-Symbol (SBS) approaches, the channel is estimated independently for each received symbol using current and past observations:

$$\widehat{\mathbf{h}}_n = f_{\mathrm{SBS}}\big(\mathbf{y}_n, \mathbf{y}_{n-1}, \ldots, \mathbf{y}_{n-W}\,;\,\theta\big), \tag{2.59}$$

where $W \geq 0$ is the observation window size and θ represents the learnable parameters of the neural network estimator f_{SBS}. SBS processing is attractive for vehicular applications due to its causal operation (enabling low latency), memory efficiency (requiring only a buffer of $W+1$ symbols), and ability to adapt to fast channel changes [6]. The limitation is the lack of future context, which can reduce accuracy in rapidly time-varying channels.

2.5.5.2 Frame by frame estimation

Frame-by-Frame (FBF) methods process an entire frame of OFDM symbols jointly, exploiting both past and future context as follows:

$$\widehat{\mathbf{H}} = f_{\mathrm{FBF}}\big(\mathbf{Y}\,;\,\theta\big). \tag{2.60}$$

2.6 Deep learning for channel estimation

DL has become a powerful tool for channel estimation in wireless communication systems. In practice, DL is realised through Neural Networks (NNs) with multiple hidden layers, which serve as the computational mechanism for learning hierarchical representations from data. A variety of NN architectures exist, each designed to capture different types of data dependencies, and the choice of structure influences learning capability, generalisation performance, and computational efficiency. Unlike classical estimators that rely on

statistical assumptions, NNs learn complex nonlinear mappings directly from data, enabling improved adaptability under dynamic and non-stationary conditions. This section outlines the key concepts and architectures that underpin DL-based channel estimation, with a focus on vehicular and high-mobility environments.

2.6.1 Motivation

As discussed in the previous sections, conventional channel estimation techniques rely on pilot-based interpolation and explicit channel models. While effective under static and low-mobility conditions, these estimators degrade in environments where channel statistics vary rapidly. Their performance depends strongly on accurate prior knowledge of noise variance and channel correlation, which is often not available or difficult to model.

DL provides an alternative by learning the mapping between received signals and channel states directly from data. This approach removes the need for explicit channel modelling and enables generalisation across diverse propagation conditions. NNs can approximate nonlinear functions that capture complex relationships between pilots, noise, and channel coefficients, outperforming classical estimators, especially when the underlying channel distribution deviates from assumed models [17].

In addition, DL methods scale naturally with data volume and can exploit spatio-temporal features in high-mobility scenarios. Once trained and optimally tuned, they can perform inference rapidly [40]. Positioning this inference latency within the wider context of the communication system, highly efficient DL estimators prevent physical-layer bottlenecks, thereby supporting the strict end-to-end delay requirements of the full V2X stack. These advantages motivate the investigation of DL architectures for channel estimation, particularly within the IEEE 802.11p framework, where high Doppler shifts and multipath fading challenge traditional estimators.

2.6.2 Neural network architectures

This subsection provides a brief overview of commonly used NN architectures and their core modelling principles, focusing on those most relevant to channel estimation.

2.6.2.1 Feedforward neural networks

Deep Neural Networks (DNNs) form the foundation of learning-based channel estimation by mapping received observations to channel estimates through a sequence of linear transformations and nonlinear activations [41]. This layered structure allows the network to learn the relationship between the received OFDM symbols and the underlying channel response.

For a network with L layers, the forward propagation is given by

$$\mathbf{h}^{(\ell)} = \sigma^{(\ell)}\big(\mathbf{W}^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}\big), \qquad \ell = 1, \ldots, L, \tag{2.61}$$

where $\mathbf{h}^{(0)}$ denotes the input vector, $\mathbf{h}^{(L)}$ the output of the final layer (representing the estimated channel $\widehat{\mathbf{h}}$), and $\sigma^{(\ell)}(\cdot)$ the activation function at layer ℓ.

Parameters and training. The trainable parameters are collected as $\theta = \{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{L}$, where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $\mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}$ represent the weights and biases of layer ℓ, respectively. The total number of parameters scales as $\sum_{\ell=1}^{L}(n_\ell n_{\ell-1}+n_\ell)$, which increases with network depth and width. These parameters are optimised during supervised training as discussed in Section 2.6.3.

Common activation functions. The choice of activation function $\sigma^{(\ell)}$ significantly impacts both the network's expressivity and training dynamics [42]. Common choices

include:

$$\text{ReLU: } \sigma(x) = \max(0, x) \quad \text{(computationally efficient, induces sparsity)} \tag{2.62}$$

$$\text{Sigmoid: } \sigma(x) = \frac{1}{1 + e^{-x}} \quad \text{(bounded output, smooth gradients)} \tag{2.63}$$

$$\text{Tanh: } \sigma(x) = \tanh(x) \quad \text{(zero-centered, symmetric)} \tag{2.64}$$

Rectified Linear Unit (ReLU) is popular in deep architectures due to its ability to mitigate vanishing gradient problems [43], although it may suffer from dead neuron problems where some units become permanently inactive. Sigmoid and tanh functions provide smooth, bounded outputs but can lead to gradient saturation in very deep networks [44].

2.6.2.2 Convolutional networks

Convolutional Neural Networks (CNNs) extends the concept of fully-connected networks by introducing convolutional layers that exploit local spatial or frequency correlations in the input data [45], [46, Ch. 9]. Unlike single-channel 2D convolution, a convolutional layer in deep learning operates on 3D tensors. Let the input layer ℓ be a tensor $\mathbf{F}^{(\ell-1)}$ with spatial dimensions $H \times W$ and C_{in} input channels (feature maps). The layer produces an output tensor $\mathbf{F}^{(\ell)}$ with C_{out} output channels. The value of the k-th output feature map $\mathbf{F}_k^{(\ell)}$ (where $k = 1, \ldots, C_{\text{out}}$) is computed by summing the convolutions of the input maps with their corresponding unique kernels:

$$\mathbf{F}_k^{(\ell)} = \sigma^{(\ell)} \left(\sum_{c=1}^{C_{\text{in}}} \mathbf{W}_{k,c}^{(\ell)} * \mathbf{F}_c^{(\ell-1)} + b_k^{(\ell)} \right), \tag{2.65}$$

where $*$ denotes the 2D convolution operator, $\mathbf{W}_{k,c}^{(\ell)}$ is the 2D kernel connecting the c-th input channel to the k-th output channel, $b_k^{(\ell)}$ is the scalar bias for the k-th output map, and $\sigma^{(\ell)}(\cdot)$ is the nonlinear activation function.

This summation over C_{in} allows the network to combine information from different input features, such as mixing the real and imaginary components of a received signal, to extract complex spatial-spectral patterns useful for channel estimation.

CNNs reduce the number of trainable parameters compared to fully-connected networks

because the same kernel weights are shared across spatial positions. This weight sharing enhances generalisation and allows deeper architectures without excessive complexity. Pooling layers are sometimes inserted between convolutions to reduce feature map dimensions and improve robustness to small frequency or time shifts.

2.6.2.3 Recurrent architectures for temporal dynamics

Recurrent Neural Networks (RNNs) model temporal dependencies by maintaining a hidden state that evolves with each input in a sequence, enabling representation of past information during sequential prediction [47]. LSTM networks [48] improve upon standard RNNs by introducing gating mechanisms that mitigate vanishing gradients during long-term learning [44].

Consider an LSTM layer ℓ processing a sequence of inputs derived from the feature tensor $\mathbf{X}'$. Let $\mathbf{x}'_n$ denote the input feature vector at OFDM symbol index n. The layer maintains a hidden state vector $\mathbf{h}_n^{(\ell)}$ and a cell state vector $\mathbf{c}_n^{(\ell)}$. Their recursive updates are governed by the following equations:

$$\mathbf{f}_n = \sigma\Big(\mathbf{W}_f^{(\ell)}[\mathbf{h}_{n-1}^{(\ell)}; \mathbf{x}'_n] + \mathbf{b}_f^{(\ell)}\Big), \tag{2.66}$$

$$\mathbf{i}_n = \sigma\Big(\mathbf{W}_i^{(\ell)}[\mathbf{h}_{n-1}^{(\ell)}; \mathbf{x}'_n] + \mathbf{b}_i^{(\ell)}\Big), \tag{2.67}$$

$$\tilde{\mathbf{c}}_n = \tanh\Big(\mathbf{W}_c^{(\ell)}[\mathbf{h}_{n-1}^{(\ell)}; \mathbf{x}'_n] + \mathbf{b}_c^{(\ell)}\Big), \tag{2.68}$$

$$\mathbf{c}_n^{(\ell)} = \mathbf{f}_n \odot \mathbf{c}_{n-1}^{(\ell)} + \mathbf{i}_n \odot \tilde{\mathbf{c}}_n, \tag{2.69}$$

$$\mathbf{o}_n = \sigma\Big(\mathbf{W}_o^{(\ell)}[\mathbf{h}_{n-1}^{(\ell)}; \mathbf{x}'_n] + \mathbf{b}_o^{(\ell)}\Big), \tag{2.70}$$

$$\mathbf{h}_n^{(\ell)} = \mathbf{o}_n \odot \tanh(\mathbf{c}_n^{(\ell)}). \tag{2.71}$$

Here, $\sigma(\cdot)$ denotes the logistic sigmoid activation, $\odot$ represents the elementwise (Hadamard) product, and $[\cdot;\cdot]$ denotes vector concatenation. The learnable parameters for layer ℓ consist of the weight matrices $\mathbf{W}_{f,i,c,o}^{(\ell)}$ and bias vectors $\mathbf{b}_{f,i,c,o}^{(\ell)}$, which correspond to the collective parameters $\mathbf{W}^{(\ell)}$ and $\mathbf{b}^{(\ell)}$ defined in the symbol list.

The forget ($\mathbf{f}_n$), input ($\mathbf{i}_n$), and output ($\mathbf{o}_n$) gates regulate the flow of information, allowing the network to capture temporal correlations across the OFDM frame index n.

2.6.2.4 Temporal convolutional networks (TCNs)

TCNs provide a feedforward alternative to recurrent architectures for modelling temporal dependencies in sequential data. The formulation presented in this section primarily follows the generic TCN architecture proposed by Bai *et al.* [49], which adapts the dilated causal convolutions originally introduced in WaveNet [50].

TCNs replace recurrence with a hierarchy of convolutions that are both *causal* and *dilated*. Causality ensures that predictions at time index n depend only on inputs up to and including n, strictly preventing information leakage from the future [49]. Dilation refers to inserting gaps between kernel elements, which enlarges the receptive field without increasing the number of trainable parameters [51].

Causal dilated convolutions. For an input feature sequence $\mathbf{X}' = [\mathbf{x}'_1, \ldots, \mathbf{x}'_I]$, a temporal convolutional layer at depth ℓ computes the hidden activation $\mathbf{h}_n^{(\ell)}$ at time index n. To capture long-range dependencies efficiently, the network employs a dilation factor $\delta^{(\ell)}$ that typically increases exponentially with depth (for example, $\delta^{(\ell)} = 2^{\ell-1}$). The operation is defined as:

$$\mathbf{h}_n^{(\ell)} = \sigma^{(\ell)}\left(\sum_{i=0}^{\kappa-1} \mathbf{W}_i^{(\ell)}\, \mathbf{h}_{n-\delta^{(\ell)}i}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \tag{2.72}$$

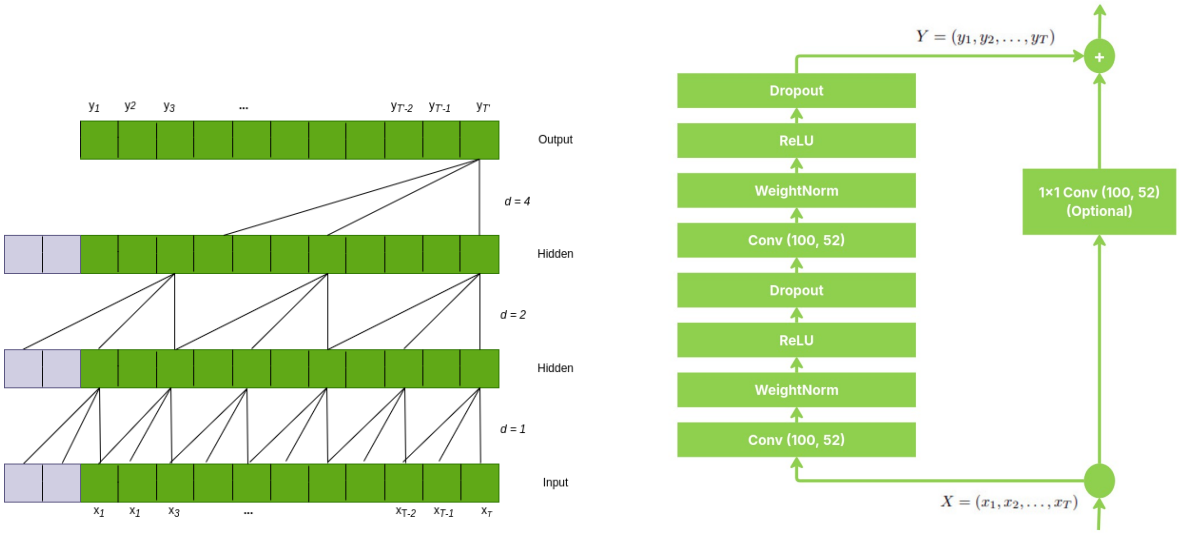
where κ is the kernel size, $\sigma^{(\ell)}(\cdot)$ is the activation function, and $\mathbf{h}_n^{(0)} = \mathbf{x}'_n$ corresponds to the input feature vector. The term $\mathbf{W}_i^{(\ell)}$ denotes the weight matrix corresponding to the i-th kernel tap, and $\mathbf{b}^{(\ell)}$ is the bias vector. The indices $n - \delta^{(\ell)}i$ ensure that the convolution looks backward in time. If $n - \delta^{(\ell)}i < 1$, the term $\mathbf{h}^{(\ell-1)}$ is assumed to be zero (left-padding), preserving the causal structure required for real-time processing [50].

Residual blocks. To support deep architectures, TCNs employ residual blocks, a concept originally introduced for image recognition in ResNet [9]. A residual block contains

a sequence of transformations typically dilated causal convolutions followed by weight normalisation and activation functions denoted here as $\mathcal{F}(\cdot)$. The block output is the sum of the input and the transformed path:

$$\mathbf{o}_n = \sigma(\mathbf{h}_n + \mathcal{F}(\mathbf{h}_n)) \,. \tag{2.73}$$

The residual connection allows the network to learn modifications to the identity mapping rather than the full transformation, which stabilises gradients during backpropagation and enables the training of very deep networks [9]. When the input and output dimensions differ, a 1×1 convolution is applied to the input branch to match dimensions before addition as shown in Figure 2.11 (b).



(a) Dilated causal convolutions with increasing dilation factors across layers expand the receptive field exponentially.

(b) Residual block with optional 1×1 convolution for dimension matching when input and output sizes differ.

Figure 2.11: TCN architectural components for temporal sequence modelling [49].

TCN advantages. Compared to RNNs and LSTMs, the TCN architecture offers several specific advantages for channel estimation: (1) Parallelism, as the entire frame can be processed simultaneously during training (unlike the sequential processing of RNNs); (2) Flexible receptive fields, which can be tuned via the dilation factor $\delta^{(\ell)}$ and kernel size κ to cover the coherence time of the channel; and (3) Stable gradients, attributed to the absence of recurrent loops and the use of residual connections [49].

2.6.2.5 Transformer architectures

Transformer networks model sequential data through self-attention mechanisms that capture global dependencies without recurrence [52]. Each element in the input sequence attends to all others, enabling the network to learn relationships across arbitrary temporal or spectral distances. To retain ordering information, positional encodings are added to the input features before self-attention is applied.

Given an input feature sequence $\mathbf{X}' = [\mathbf{x}'_1, \ldots, \mathbf{x}'_N]$, the transformer projects it into three learned representations: queries $\mathbf{Q} \in \mathbb{R}^{N \times d_k}$, keys $\mathbf{K} \in \mathbb{R}^{N \times d_k}$, and values $\mathbf{V} \in \mathbb{R}^{N \times d_v}$. For one attention head, the attention output is computed as

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2.74}$$

where the softmax function normalises the dot-product similarities between queries and keys to yield attention weights that determine the contribution of each value vector. Multi-head attention repeats this operation across multiple subspaces to capture diverse dependency patterns.

The resulting attended features are then processed by a feedforward layer to generate the output representation:

$$\widehat{\mathbf{H}} = f_\theta(\mathbf{A}) = \phi\big(\mathbf{W}_2\,\sigma(\mathbf{W}_1\mathbf{A} + \mathbf{b}_1) + \mathbf{b}_2\big), \tag{2.75}$$

where $\sigma(\cdot)$ and $\phi(\cdot)$ denote nonlinear activation functions, and $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ are learned parameters of the feedforward block. Together, these components define the end-to-end mapping $\mathbf{X}' \mapsto \widehat{\mathbf{H}}$.

2.6.3 Training strategies

Having established the neural architectures in the previous section, this subsection describes the strategies used to train and optimise their parameters. Training deep NN estimators requires defining the learning objective, optimisation algorithm, and regularisation techniques that ensure stable convergence and generalisation to unseen channel conditions.

The discussion begins with the supervised learning formulation adopted throughout this thesis, followed by practical optimisation and regularisation approaches.

2.6.3.1 Supervised learning

The neural estimator f_θ maps the input feature grid $\mathbf{X}'$ to the predicted channel $\widehat{\mathbf{H}}$, where $\mathbf{X}'$ denotes the extracted features and $\mathbf{H}$ the ground-truth channel grid. Training follows a supervised regression setup using N labelled samples $\{(\mathbf{X}'^{(i)}, \mathbf{H}^{(i)})\}_{i=1}^{N}$. The trainable parameters θ are obtained by minimising the regularised empirical risk:

$$\theta^\star = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(f_\theta(\mathbf{X}'^{(i)}),\, \mathbf{H}^{(i)}\big) \;+\; \lambda\, R(\theta), \tag{2.76}$$

where $\lambda \geq 0$ controls the regularisation strength and $R(\theta)$ penalises model complexity, commonly through weight decay [46, Ch. 7], where $\lambda \geq 0$ controls the regularisation strength and $R(\theta)$ penalises model complexity [46, Ch. 7].

Loss functions. The loss function quantifies the discrepancy between the predicted and true channel responses and guides the optimisation of network parameters. During training, gradients of the loss with respect to the network weights determine how each parameter is updated to reduce prediction error, as described by the gradient descent rule in Section 2.6.3.2. The choice of $\mathcal{L}$ thus directly shapes the learning dynamics and the invariances captured by the model [46, Ch. 5]. For channel estimation, commonly used losses include MSE, NMSE, and Mean Absolute Error (MAE):

$$\text{MSE:} \quad \mathcal{L}_{\text{MSE}} = \frac{1}{\|\mathbf{M}\|_0} \big\|(\widehat{\mathbf{H}} - \mathbf{H}) \odot \mathbf{M}\big\|_F^2, \tag{2.77}$$

$$\text{NMSE:} \quad \mathcal{L}_{\text{NMSE}} = \frac{\big\|(\widehat{\mathbf{H}} - \mathbf{H}) \odot \mathbf{M}\big\|_F^2}{\big\| \mathbf{H} \odot \mathbf{M} \big\|_F^2 + \varepsilon}, \tag{2.78}$$

$$\text{MAE:} \quad \mathcal{L}_{\text{MAE}} = \frac{1}{\|\mathbf{M}\|_0} \big\|(\widehat{\mathbf{H}} - \mathbf{H}) \odot \mathbf{M}\big\|_1, \tag{2.79}$$

where $\|\mathbf{M}\|_0$ counts the active subcarrier-symbol pairs and $\varepsilon > 0$ prevents division by zero.

MSE provides strong gradients and accelerates convergence but may overemphasise large errors due to the quadratic penalty term [46, Ch. 6]. NMSE normalises the error by the channel power, promoting scale invariance across varying SNR levels. MAE offers robustness to statistical outliers, but often results in slower convergence near the solution due to its constant gradient [53, Ch. 1]. During training, the loss is computed for each sample, and the gradient (of the loss with respect to the parameters) averaged over a mini-batch to approximate the true gradient. This stochastic approach ensures that each weight update reflects a diverse set of channel realisations (e.g., varying Doppler shifts and SNR levels), thereby promoting robust generalisation.

2.6.3.2 Optimisation and regularisation

Training deep networks involves iteratively adjusting parameters to minimise the loss defined above. This process, known as optimisation, determines how the model learns from data and how effectively it converges toward a solution that generalises beyond the training set.

Parameter updates typically follow stochastic gradient descent (SGD) [54]:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}_t, \tag{2.80}$$

where η_t is the learning rate at iteration t, and $\nabla_\theta \mathcal{L}_t$ denotes the gradient of the loss with respect to the parameters. Gradients are computed by backpropagation [55], which propagates the loss derivatives backward through the network layers according to Equation (2.61).

Adaptive Optimisers. While SGD provides a basic update rule, its convergence speed and stability depend strongly on the learning rate. Adaptive optimisers such as Adam [56] improve performance by maintaining running averages of the first and second moments

of the gradients:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \qquad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^{\odot 2}, \tag{2.81}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \tag{2.82}$$

where $\mathbf{g}_t = \nabla_\theta \mathcal{L}_t$, and $\hat{\mathbf{m}}_t$, $\hat{\mathbf{v}}_t$ are bias-corrected moment estimates. These adaptive updates adjust learning rates per parameter, improving convergence in noisy or non-stationary training conditions. Learning-rate schedules and gradient clipping further stabilise training [57].

Regularisation. Regularisation mitigates overfitting and improves generalisation by constraining model capacity and stabilising optimisation [46, Ch. 7]. The most widely used techniques include weight decay, dropout, early stopping, and normalisation. Weight decay adds an L_2 penalty term $\lambda\|\theta\|_2^2$ to the loss in Equation (2.76), where $\theta = \{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{L}$ denotes all trainable weights and biases, and $\|\cdot\|_2$ represents the Euclidean norm over these parameters [58]. This penalty discourages large parameter magnitudes and promotes simpler models. Dropout randomly deactivates units with probability p during training, encouraging redundancy and robustness to noise [59]. Early stopping halts training once the validation loss ceases to improve, preventing overfitting to the training set [60]. Normalisation methods such as batch, layer, and group normalisation rescale intermediate activations to maintain consistent statistics across layers [61]–[64]. Specifically, batch normalisation operates over each mini-batch, layer normalisation acts across features within a sample, and group normalisation normalises within feature groups of a sample. Layer and group variants are preferred for small batches or recurrent models where batch statistics are unreliable [63].

2.6.4 Deep learning approaches to channel estimation

This subsection reviews the evolution of DL architectures for channel estimation. While the primary focus of this thesis is on high-mobility vehicular scenarios, we first contextualise these methods within the broader landscape of DL-based PHY layer communications.

The discussion is organised into three categories: (i) image-based and CNN architectures, (ii) hybrid model-driven approaches that refine classical estimators, and (iii) recurrent architectures specifically designed for time-varying vehicular channels. Research in this area has progressed rapidly, producing architectures that are different in design, computational requirements, and suitability for real-time deployment. The following subsections summarise the main studies from these categories and outline their main design characteristics and performance trends.

2.6.4.1 Image-based estimators

The initial wave of DL channel estimation treated the time-frequency grid of the channel response as a 2D image, leveraging the powerful feature extraction capabilities of CNNs originally developed for computer vision.

Ye *et al.* [41] pioneered this approach with a coarse-to-fine scheme, using a deep CNN to interpolate channel values from pilots. Their work demonstrated that CNNs could implicitly learn the frequency-selective properties of the channel without requiring explicit knowledge of the channel statistics. Building on this, researchers in [65] proposed a straightforward image-restoration framework. They treated the pilot positions as a low-resolution image and employed a super-resolution CNN (SRCNN) to reconstruct the full high-resolution channel grid.

While these image-based methods outperform classical LS estimation in static or low-mobility environments, they often struggle in high-mobility vehicular scenarios [40]. Standard 2D kernels assume local correlation in both time and frequency; however, in fast-varying channels, the Doppler effect disrupts the local temporal correlation, rendering static image-processing techniques less effective. This limitation motivated the shift toward methods that explicitly account for temporal dynamics or leverage classical estimators.

2.6.4.2 Hybrid methods

To improve robustness in dynamic environments, hybrid approaches (often termed model-driven DL) combine classical signal processing with NNs. Rather than learning the channel from scratch (black-box estimation), these methods use a classical estimator to provide a coarse initial estimate, which is then refined by a lightweight NN.

Gizzini *et al.* [6], [66] combined classical estimators (STA and TRFI) with FNNs that refined their outputs via learned nonlinear mappings. The classical estimators first produced coarse, noise-affected estimates, which were then refined by the NN to correct interpolation errors and improve robustness. Their BER performance results confirmed that neural post-processing can effectively enhance classical estimators, validating the use of DL methods for vehicular channel estimation. We will refer to these FNN-based estimators as STA-DNN and TRFI-DNN estimators going forward.

Autoencoder-based denoising networks [67] extended these ideas by learning representations of channel frequency responses in an unsupervised manner. The autoencoder was used as a post-processing stage following the DPA estimator. In this cascaded architecture, the DPA estimator first generates a coarse, noisy channel grid, which is then fed as input to the autoencoder. The network learns to project this noisy input onto a compact, low-dimensional representation, effectively filtering out noise and interpolation errors before reconstructing the final clean channel response.

With these hybrid estimators demonstrating the effectiveness of neural refinement for channel estimation, recent studies have shifted toward architectures that model temporal dependencies across OFDM symbols or frames. The next subsection discusses recurrent estimators that exploit this temporal correlation to improve channel tracking in vehicular environments.

2.6.4.3 Recurrent estimators

Recurrent models address the temporal evolution of vehicular channels by capturing dependencies across consecutive OFDM symbols within a frame, enabling the network to track channel variations and exploit temporal correlation structure to improve estimation accuracy.

Pan *et al.* [7] employed a two-gate LSTM network with 128 hidden units per layer and a fully connected layer (LSTM-DNN-DPA) to transform the hidden state into channel estimates. Their approach achieved improved performance over the STA-DNN [9] and TRFI-DNN [66].

Gizzini *et al.* [8] further enhanced this design by combining a three-gate LSTM with DPA and temporal averaging (TA) as post processing additional methods (LSTM-DPA-TA). This configuration further improved estimation accuracy under high mobility conditions, although with increased computational cost. Building on this concept, Gizzini *et al.* [68] proposed the GRU-DPA-TA framework, to replace the three-gate LSTM unit with a two-gate GRU structure. This reduces the number of trainable parameters by roughly 25% while maintaining comparable temporal modelling capability. The framework demonstrated effectiveness with lower computational demands than the LSTM-based estimators, making it attractive for real-time vehicular implementations.

Hou *et al.* [69] also explored a GRU-based channel estimator that refines DPA outputs through a GRU followed by a fully connected layer that maps recurrent outputs to the CFR. This design also achieved improved BER performance over STA-DNN [9], TRFI-DNN [66], and LSTM-DNN [7], without a significant increase in computational cost.

Recurrent models such as LSTM-DPA-TA, and GRU-DPA-TA have therefore gained attention as practical baselines for DL-based channel estimation. They balance modelling power and computational efficiency, aligning with the objective of enabling deployable neural estimators for vehicular communication systems.

2.6.4.4 Attention-based architectures

Recent literature has increasingly explored transformer-based architectures, which rely on self-attention mechanisms rather than recurrence to model temporal dependencies. Unlike RNNs, which process symbols sequentially, transformers process the entire time-frequency grid in parallel, using self-attention to weigh the importance of every pilot symbol against every other symbol in the frame [70].

For example, researchers in [71] proposed *Channelformer*, an encoder-decoder architecture that utilises multi-head attention to capture long-range dependencies across the OFDM slot. Similarly, Liu *et al.* [72] introduced *CE-ViT*, a vision transformer approach that treats the channel grid as a sequence of image patches to achieve robustness in high-mobility scenarios. However, this global context comes at a high computational cost. The self-attention mechanism scales quadratically with sequence length ($\mathcal{O}(N^2)$), creating memory and compute bottlenecks on embedded hardware. Making such architectures viable for online deployment requires aggressive structural pruning, removing up to 70% of parameters [71].

Given that standard vehicular hardware operates under strict power and latency constraints, the quadratic overhead of full self-attention is often prohibitive. In contrast, recurrent architectures (and convolutional alternatives) scale linearly with sequence length ($\mathcal{O}(N)$) [40]. Therefore, this thesis adopts the recurrent estimators, specifically LSTM-DPA-TA [8], and GRU-DPA-TA [68], as the primary DL benchmarks.

2.7 Interpretability and feature attribution

As discussed in previous sections, DL models can significantly enhance channel estimation accuracy. However, DL estimators remain largely opaque and in safety-critical vehicular systems, accuracy alone is insufficient. Estimators must also be *transparent* and *predictable*. Understanding when and why a model fails, whether its behaviour is stable across varying SNR, Doppler shift, or delay spread conditions, and how its decisions align

with physical channel properties is essential for reliable deployment.

2.7.1 Definitions

Interpretability entails providing human-understandable explanations for the internal processes that govern the behaviour and specific decisions of a model [73]. While often used interchangeably with explainability, the former typically focuses on understanding internal mechanisms, whereas the latter concerns post-hoc justification of individual predictions [74]. In this thesis, *interpretability* is used to denote methods that identify which input features influence the model's output and also which model parameters are relevant.

Interpretability techniques are generally categorised as intrinsic or post-hoc [75, Ch. 2.2]. Intrinsic interpretability is achieved when the model structure itself is transparent (e.g., linear models, decision trees, or explicit attention mechanisms). Post-hoc approaches, conversely, analyse a trained model without modifying its internal structure. In channel estimation, post-hoc techniques are particularly valuable as they allow for the analysis of high-performance, complex architectures without sacrificing estimation accuracy.

2.7.2 Feature attribution methods

Feature attribution quantifies the contribution of individual input features to a specific prediction [76]. In OFDM-based estimators, input to the DL model can be represented as

$$\mathbf{X}' \in \mathbb{R}^{K \times I \times C}, \tag{2.83}$$

where K is the number of subcarriers, I the number of OFDM symbols, and C the number of feature channels. These channels may include the real and imaginary components, pilot-data masks, and classical estimates such as DPA or STA.

Attribution methods produce a relevance tensor $\mathbf{R}_{\text{in}} \in \mathbb{R}^{K \times I \times C}$ of the same shape as $\mathbf{X}'$.

Aggregating over feature channels yields a two-dimensional heat map:

$$\mathbf{R}[k,n] = \sum_{c=1}^{C} |\mathbf{R}_{\text{in}}[k,n,c]|, \tag{2.84}$$

highlighting influential time-frequency positions. To evaluate the relative importance of each modality m (pilots, data, or classical estimates), the modality relevance ratio can be defined as:

$$r_m = \frac{\sum_{c\in\mathcal{C}_m}\sum_{k,n}|\mathbf{R}_{\text{in}}[k,n,c]|}{\sum_{c=1}^{C}\sum_{k,n}|\mathbf{R}_{\text{in}}[k,n,c]|}. \tag{2.85}$$

Gradient-based saliency maps. Gradient-based saliency computes the gradient of a scalar output with respect to the input, visualising $\nabla_{\mathbf{X}'} f(\mathbf{X}')$ as a heat map to identify which input regions influence the prediction most [77]. For channel estimation, the scalar output is typically chosen as the mean squared error or a specific channel coefficient $\widehat{H}[k_0, n_0]$ at a target location. Saliency maps highlight which time-frequency positions (k, n) exhibit the strongest local sensitivity. However, gradients reflect only the local linear approximation at the current input and can become noisy or vanish in saturated regions of activation functions, limiting their reliability in highly nonlinear networks and dynamic vehicular channels [78].

DeepLIFT. Deep Learning Important FeaTures (DeepLIFT) addresses gradient saturation by backpropagating activation differences relative to a baseline input [79]. This approach provides stable relevance scores even when gradients vanish but depends strongly on the baseline choice [80].

SHAP. SHapley Additive exPlanations (SHAP) computes feature contributions using Shapley values from cooperative game theory [76]. It provides theoretically grounded explanations but is computationally expensive. Also, its independence assumption among features is often violated in OFDM grids where subcarriers and symbols are highly correlated [81].

Perturbation-based methods. Perturbation (occlusion) methods evaluate feature relevance by masking inputs and observing prediction changes [82]. While model-agnostic, they require multiple forward passes and are therefore computationally costly for high-dimensional OFDM data [83].

Layer-wise Relevance Propagation (LRP). LRP decomposes the prediction of a model by backpropagating relevance scores from the output to the input through a conservation principle [84]. For a scalar prediction $s = f(\mathbf{X}')$, LRP computes input relevances $\{R_i\}$ that satisfy the conservation property:

$$s = \sum_i R_i, \tag{2.86}$$

where i indexes all input features (spanning grid locations and input channels). The relevance at each layer is redistributed to the previous layer using propagation rules that preserve the total relevance. A commonly used stabilised z^β-rule propagates relevance through a linear layer with inputs a_i and weights w_{ij} as:

$$R_i = \sum_j \frac{a_i w_{ij}}{\sum_k a_k w_{kj} + \varepsilon \,\mathrm{sign}\big(\sum_k a_k w_{kj}\big)} R_j, \tag{2.87}$$

where $\varepsilon > 0$ is a small stabilisation constant that prevents division by zero, j indexes neurons in the current layer, and R_j is the relevance of neuron j [85]. This rule is applied recursively from the output layer to the input layer to obtain the input relevance tensor $\mathbf{R}_{\text{in}}$. For convolutional layers, the same propagation rule applies with w_{ij} interpreted as convolutional kernel weights connecting input position i to output position j [84]. The conservation property in Equation (2.86) ensures that the total relevance distributed across input features equals the scalar prediction, providing a physically interpretable budget of contributions over the active subcarrier set $\mathcal{K}_{\text{act}}$ [85].

Overall, each attribution method exhibits different trade-offs for wireless channel estimation applications. Gradient-based saliency is computationally efficient, but suffers from noise. DeepLIFT addresses saturation but introduces dependence on and sensitivity to the selection of the baseline. SHAP assumes feature independence, which is not typically

true. LRP offers a practical middle ground, providing efficient computation through back-propagation and conservation-based attributions between output and input contributions.

2.7.3 Applications in wireless communication

Interpretability has recently gained traction in wireless research, leading to frameworks such as Explainable AI for Channel Estimation (XAI-CHEST) [10] and Gradient-based XAI Channel Estimation (GRACE) [86].

The XAI-CHEST framework introduced by Gizzini *et al.* [10] employs the perturbation-based approach described in Section 2.7.2 to identify relevant input features for channel estimation. XAI-CHEST trains a separate interpretability network that learns to generate noise masks for input subcarriers. The framework applies these learned noise masks with varying intensity levels to perturb the channel estimator's inputs during training. Subcarriers that tolerate high noise levels without degrading the prediction accuracy of the utility network are classified as irrelevant, while noise-sensitive subcarriers are identified as relevant [10]. The interpretability network is optimised using a custom loss function that balances two objectives: maximising the noise applied to input features while maintaining the utility network's estimation accuracy. This approach reveals which input subcarriers the estimators rely upon for channel prediction in doubly selective fading environments. Identifying that only a subset of subcarriers is necessary for an accurate estimate, XAI-CHEST enables the input dimension to be reduced [10]. However, as with perturbation-based methods generally, XAI-CHEST has a high computational overhead due to repeated perturbation evaluations and auxiliary training.

To address these computational limitations, the GRACE framework [86] applies LRP to provide efficient gradient-based explanations of deep channel estimators. GRACE leverages the LRP conservation principle of Equation (2.86) to decompose channel predictions into subcarrier-level contributions through a single backward pass, identifying which time-frequency positions most influence the estimate. Using internal network weights and activations rather than external perturbations, GRACE is faster than XAI-CHEST. Empirical

results show that restricting estimation to highly relevant features identified by GRACE improves BER performance relative to perturbation-based selection [86].

Together, these frameworks illustrate complementary philosophies: perturbation-based methods offer model-agnostic interpretability, while gradient-based approaches provide efficient internal insights. While frameworks such as XAI-CHEST have successfully demonstrated that interpretability can guide input dimension reduction and architectural simplification [10], they rely on training auxiliary interpretability networks, which introduces training overhead. Consequently, a general framework that integrates *efficient* gradient-based relevance insights directly into the model design process-to guide pruning and lightweight architecture search without secondary models-remains an open challenge.

2.8 Performance metrics for channel estimation

Evaluating channel estimators requires metrics that reflect both estimation accuracy and deployment feasibility. This section defines the key indicators used in this thesis: NMSE to measure channel-centric accuracy, BER to measure system-level performance, and model parameters, FLOPs and inference time to measure computational efficiency.

2.8.1 Normalised Mean Square Error (NMSE)

The NMSE quantifies the accuracy of channel estimation by measuring the mean square error between estimated and true channel coefficients, normalised by the power of the true channel [87, Sec. 2.4]:

$$\text{NMSE} \;=\; \frac{\mathbb{E}\Big[\, \|\widehat{\mathbf{H}} - \mathbf{H}\|_F^2 \,\Big]}{\mathbb{E}[\, \|\mathbf{H}\|_F^2 \,]}, \tag{2.88}$$

where $\mathbf{H} \in \mathbb{C}^{K\times I}$ and $\widehat{\mathbf{H}} \in \mathbb{C}^{K\times I}$ are the true and estimated channel grids and $\|\cdot\|_F$ denotes the Frobenius norm. In practice, we compute a masked empirical estimate over the active set $\mathcal{K}_{\text{act}}$ (active data and pilot subcarriers). NMSE is typically reported in decibels as $\text{NMSE}_{\text{dB}} = 10\log_{10}(\text{NMSE})$.

2.8.2 Bit Error Rate (BER)

BER measures end-to-end communication system performance by calculating the proportion of incorrectly decoded information bits after demodulation and decoding [27, Ch. 5]:

$$\text{BER} = \frac{\text{number of bit errors}}{\text{total number of transmitted bits}}. \tag{2.89}$$

Unlike NMSE, which measures only the estimation accuracy, BER reflects the practical impact of channel estimation errors on the entire receiver chain, including equalisation, demodulation, and forward error correction decoding [88, Ch. 8]. BER is a good performance metric for wireless systems, as it directly quantifies the reliability of information delivery.

The relationship between estimation quality and BER depends on the modulation scheme, coding rate, and receiver architecture. For uncoded systems with perfect equalisation, an approximate relationship can be expressed as [89]:

$$\text{BER} \approx Q\left(\sqrt{\frac{2\,E_b/N_0}{1+\text{NMSE}}}\right), \tag{2.90}$$

where $Q(\cdot)$ represents the Gaussian Q-function defined as $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-t^2/2}dt$, and E_b/N_0 denotes the bit energy to noise spectral density ratio. For the coded OFDM systems considered in this thesis, BER is measured empirically by counting bit errors after Viterbi decoding [88, Ch. 8 and Ch. 13].

2.8.3 Computational complexity metrics

Beyond estimation accuracy and communication reliability, the practical feasibility of a deep channel estimator depends on its computational efficiency. Real-time vehicular communication imposes tight latency budgets and hardware constraints in embedded Central Processing Units (CPUs), Graphics Processing Units (GPUs), or Field-Programmable Gate Arrays (FPGAs). Because physical-layer processing time directly impacts the broader end-to-end system delays, DL inference must execute rapidly enough

to avoid creating computational bottlenecks. To assess deployment suitability, we report both analytic and empirical metrics.

2.8.3.1 Model parameters

The total number of trainable parameters quantifies the memory footprint required to store the network [46, Ch. 6]:

$$N_{\text{params}} = \sum_{\ell=1}^{L} \big(n_\ell n_{\ell-1} + n_\ell\big), \tag{2.91}$$

where L is the number of layers, n_ℓ is the number of neurons in layer ℓ, and the two terms represent the weight matrices $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and bias vectors $\mathbf{b}^{(\ell)} \in \mathbb{R}^{n_\ell}$, respectively. For convolutional layers, the parameter count is $N_{\text{params}} = C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w + C_{\text{out}}$, where C_{in} and C_{out} are the input and output channel dimensions, and $k_h \times k_w$ is the kernel size [90].

2.8.3.2 Floating Point Operations (FLOPs)

FLOPs measure the computational complexity required for a single forward pass through the DL network [91]. For a fully connected layer with input dimension n_{in} and output dimension n_{out}, the FLOPs count is:

$$\text{FLOPs}_{\text{FC}} = 2n_{\text{in}}n_{\text{out}}, \tag{2.92}$$

accounting for one multiplication and one addition per weight. For a convolutional layer with output spatial dimensions $H_{\text{out}} \times W_{\text{out}}$, kernel size $k_h \times k_w$, and C_{in} input channels and C_{out} output channels, the FLOPs count is [92]:

$$\text{FLOPs}_{\text{conv}} = 2H_{\text{out}}W_{\text{out}}C_{\text{out}}C_{\text{in}}k_hk_w. \tag{2.93}$$

FLOPs are hardware-agnostic and useful for comparisons, but do not predict latency, which also depends on memory bandwidth, parallelism, cache locality, and kernel optimisations [93]–[95].

It is important to note that the FLOPs calculations reported in this thesis focus exclusively on the NN inference, excluding the computational cost of classical preprocessing or post-processing methods such as DPA or Time Averaging (TA). This distinction is made because the computational overhead of these classical methods is almost negligible compared to that of the DL model. For example, complexity analysis in [6] demonstrates that initial channel estimation (DPA) requires approximately 87.8% fewer real-valued operations than the hybrid STA-DNN architecture, justifying this simplification.

2.8.3.3 Inference time

Inference time quantifies the end-to-end latency for processing one OFDM frame on the target hardware platform [96]. Unlike analytic measures such as FLOPs, it captures framework overheads, memory access delays, and kernel execution efficiency, providing a realistic indicator of deployability.

2.9 Chapter summary

This chapter introduced the theoretical background and related studies underpinning this research. It outlined the signal model for vehicular OFDM systems and reviewed classical and advanced channel estimation techniques. The fundamentals of DL related to this work were discussed, including core network architectures and training strategies. A review of recent DL approaches for channel estimation was presented. The chapter concluded with an overview of interpretability techniques and performance metrics used to evaluate learning-based estimators. The next chapter will discuss the development of the dataset used in this work.

Chapter 3

Simulation and dataset generation

This chapter presents the simulation framework and dataset generation procedures that form the foundation for all DL-based channel estimation investigations in this thesis. We detail the implementation of the IEEE 802.11p PHY layer, instantiation of vehicular channel models, data preprocessing strategies, and reproducibility protocols used to ensure reliable and verifiable results.

3.1 Overview

To evaluate DL architectures for channel estimation in vehicular scenarios, realistic propagation data is required. This chapter describes our simulation methodology for generating such datasets. Building on the theoretical foundations presented in Section 2.2 to Section 2.4, we simulate the IEEE 802.11p physical layer under six vehicular channel models, enabling systematic evaluation of DL architectures under controlled, yet realistic conditions.

We first present the simulation setup (Section 3.2) and data preprocessing pipeline (Sec-

tion 3.3), and then describe the final dataset in Section 3.5.

3.2 Simulation

This section describes the simulation of the IEEE 802.11p PHY layer and vehicular channel models introduced in Chapter 2. The simulation environment is implemented in MATLAB[1]. The data generation process leverages and adapts the open-source framework established by Gizzini *et al.*[2].

The simulation uses baseband processing to generate channel observations for supervised learning, isolating channel propagation effects from hardware impairments while maintaining full compliance with standard specifications. Section 3.2.1 presents the PHY simulation, Section 3.2.2 describes the vehicular channel model simulation, and Section 3.2.3 details the frame generation process.

3.2.1 Physical layer configuration

The simulation implements all parameters specified in Table 2.4, including 10 MHz channel bandwidth with 64-point FFT, 52 active subcarriers comprising 48 data and 4 pilot tones, 156.25 kHz subcarrier spacing, and 8μs OFDM symbol duration with 1.6μs CP.

3.2.1.1 Modulation and coding

The modulation scheme is fixed at 16QAM throughout all experiments (except in Chapter 6 where 64QAM is employed), providing 4 bits per symbol and balancing spectral efficiency with robustness under the high-Doppler shift conditions characteristic of vehicular channels [21]. Although IEEE 802.11p supports BPSK, Quadrature Phase Shift

[1] MathWorks MATLAB R2022b.

[2] `https://github.com/abdulkarimgizzini/Deep-Learning-Based-Channel-Estimation-Schemes-for-IEEE-802.11p-Standard`

Keying (QPSK), 16QAM, and 64QAM, the selection of 16QAM achieves reasonable data rates whilst maintaining acceptable error performance in doubly selective fading environments.

The transmitter processing follows the chain detailed in Section 2.4.3, employing rate-1/2 non-punctured convolutional coding to prioritise link reliability over throughput. At the receiver, the convolutional code is decoded using the Viterbi algorithm with a traceback length of 34 bits. Each transmitted frame contains two LTS preamble symbols followed by 50 data-carrying OFDM symbols (structure detailed in Section 2.4.4), providing sufficient temporal duration (416 μs total) to observe channel evolution whilst ensuring computational tractability for large-scale dataset generation.

3.2.1.2 Scrambling and interleaving

Scrambling and interleaving follow the IEEE 802.11p specifications detailed in Section 2.4.3. The scrambler employs the 7-bit Linear Feedback Shift Register (LFSR) polynomial $S(x) = x^7 + x^4 + 1$ with initialisation vector [1011101] (decimal 93). The two-stage interleaving process disperses burst errors across the frequency-selective channel: matrix interleaving arranges coded bits into a $16 \times N_{\text{col}}$ structure where $N_{\text{col}} = (n_{\text{bits}} \times K_d \times I)/16$, and block interleaving applies a fixed pseudo-random permutation vector of length $n_{\text{bits}} \times K_d \times I = 9600$ bits. This permutation vector is generated deterministically using seed = 1 and applied uniformly across all frames, enabling the receiver to perform the inverse permutation during deinterleaving. Table 3.1 summarises the simulation-specific configuration parameters that extend beyond the standard IEEE 802.11p specifications, documenting the choices made for dataset generation and experimental consistency.

3.2.2 Vehicular channel model simulation

The simulation instantiates the six vehicular channel models specified in Section 2.3.3 and detailed in Table 2.3, implementing the TDL structure with time-varying tap coefficients following Jakes' Doppler spectrum. The six models comprise VTV-SDWW, VTV-EX,

Table 3.1: Simulation configuration parameters.

Parameter	**Value**
OFDM symbols per frame (I)	50
Preamble structure	2 × LTS
Modulation scheme	16QAM (fixed)
Coding rate (R)	1/2 (non-punctured)
Scrambler seed	93 (decimal)
Interleaver permutation seed	1 (deterministic)
Channel realisations per model	20,000
Training/test split	18,000 / 2,000
SNR range (E_b/N_0)	0–40 dB (5 dB steps)

VTV-UC, RTV-SS, RTV-EX, and RTV-UC, covering diverse vehicular scenarios from urban canyon environments to high-speed expressway communications with vehicle velocities ranging from 48 km/h to 104 km/h and maximum Doppler frequencies from 300 Hz to 1 200 Hz.

Each channel model is characterised by its number of propagation paths (11–12 taps), path gains, path delays, and Doppler characteristics as specified in Table 2.3. The maximum Doppler frequency for each model follows $f_D = v f_c / c$ where v is the vehicle velocity, $f_c = 5.9$ GHz is the carrier frequency, and $c = 3 \times 10^8$ m/s is the speed of light, as discussed in Section 2.2.2. The channel generation uses MATLAB's `comm.RayleighChannel` system object, which implements the WSSUS assumptions described in Section 2.2.5 and produces Rayleigh fading statistics as specified in Section 2.2.6.

Channel realisation procedure

Each frame is transmitted through a unique channel realisation generated using deterministic seed assignment. For a dataset of N_{frames} frames indexed by $i = 1, 2, \ldots, N_{\text{frames}}$, the seed of frame i is simply i. This one-to-one mapping ensures reproducible channel

generation whilst providing statistical independence across frames, as each seed produces a unique combination of tap gains, phases, and Doppler shifts drawn from the statistical distributions defined by the channel model.

The channel generation procedure for each frame operates as follows. First, the TDL channel model is initialised with seed i and sampling frequency $f_s = K \cdot \Delta f = 10$ MHz. Second, a time-domain channel impulse response $h(t, \tau)$ spanning the frame duration is generated according to the LTV model described in Section 2.2.1:

$$h(t, \tau) = \sum_{l=0}^{L-1} h_l(t)\, \delta(\tau - \tau_l), \tag{3.1}$$

where $h_l(t)$ are the time-varying complex path gains driven by Jakes' Doppler processes, τ_l are the discrete path delays specified in Table 2.3, and L is the number of taps. Third, the transmitted time-domain frame (including preamble and all CPs) is convolved with the time-varying impulse response. Fourth, complex Gaussian noise with variance $\sigma^2 = N_0$ is added, where N_0 is determined by the target SNR following the relationship established in Section 2.4.5. Finally, the received signal is processed by removing the CPs and applying a 64-point FFT to obtain the frequency-domain signal.

This procedure produces the frequency-domain received signal $\mathbf{Y} \in \mathbb{C}^{K \times (I+2)}$ and the corresponding true frequency-domain channel matrix $\mathbf{H} \in \mathbb{C}^{K \times (I+2)}$ for supervised learning, where $I = 50$ data symbols and 2 preamble symbols span a total frame duration of $(I + 2) \times 8\ \mu\text{s} = 416\ \mu\text{s}$. During this interval, the channel evolves according to the time-varying TDL model, with temporal correlation governed by the Doppler spectrum and coherence time characteristics discussed in Section 2.2.2.

Unlike conventional simulation approaches, which transmit multiple frames through a single channel realisation, our framework generates a unique channel instance for each frame. This design choice provides several advantages for evaluating DL models. First, statistical independence between frames eliminates temporal correlation in training samples, preventing the network from learning frame-to-frame dependencies, which is particularly critical in the high-mobility scenarios considered in this work, where rapid channel evolution renders inter-frame correlations unreliable. Second, diverse coverage of the channel

model's statistical distribution ensures that training and testing sets span the full range of propagation conditions rather than being concentrated around particular realisations. Third, testing on independently generated channels provides rigorous evaluation of the estimator's ability to generalise beyond training conditions, as each test frame represents a unique channel instance.

3.2.3 Frame generation

This subsection describes the complete frame generation process from information bits to transmitted OFDM symbols, followed by the channel estimation procedure at the receiver.

3.2.3.1 Transmitter processing chain

Each frame is generated following the IEEE 802.11p transmitter architecture described in Section 2.4.3. The processing chain comprises: scrambling (Section 3.2.1.2), rate-1/2 convolutional encoding (Section 2.4.3), two-stage interleaving (Section 3.2.1.2), 16QAM mapping with Gray coding, and subcarrier allocation following the structure in Section 2.4.5.

The frequency-domain symbols undergo 64-point IFFT with CP insertion (Section 2.4.1.3) to form time-domain OFDM symbols. The complete transmitted frame comprises a two-symbol preamble (Section 2.4.4) followed by 50 data symbols:

$$\mathbf{s}_{\text{frame}} = [\mathbf{x}_{\text{preamble}}, \mathbf{x}_{\text{CP},1}, \mathbf{x}_1, \ldots, \mathbf{x}_{\text{CP},I}, \mathbf{x}_I], \tag{3.2}$$

where $\mathbf{x}_i \in \mathbb{C}^K$ is the IFFT output for data symbol i and $\mathbf{x}_{\text{CP},i} \in \mathbb{C}^{K_{\text{CP}}}$ is its cyclic prefix, resulting in a frame duration of 416 μs.

3.2.3.2 Receiver processing chain

The receiver performs CP removal and 64-point FFT to obtain frequency-domain samples. LS channel estimation is applied using the two LTS preamble symbols as described in Section 2.4.4. Each LTS comprises a known BPSK sequence on the 52 active subcarriers.

After CP removal and FFT, the two preamble observations are averaged to obtain the LS estimate:

$$\widehat{H}_{\mathrm{LS}}[k] = \frac{Y_{\mathrm{preamble},1}[k] + Y_{\mathrm{preamble},2}[k]}{2 \cdot P[k]}, \quad k \in \mathcal{K}_{\mathrm{act}}, \tag{3.3}$$

where $Y_{\mathrm{preamble},j}[k]$ denotes the received frequency-domain sample at subcarrier k for the j-th preamble symbol, $P[k]$ is the known transmitted preamble value, and $\mathcal{K}_{\mathrm{act}}$ denotes the set of 52 active subcarrier indices. Averaging across the two preambles reduces noise variance by 3 dB compared to single-symbol estimation [29], improving the accuracy of the initial CSI. This LS estimate serves as both the baseline estimator for performance comparisons and the initial input to DL-based refinement architectures evaluated in subsequent chapters.

3.3 Data preprocessing

This section describes how raw simulation data is prepared for NN training. Section 3.3.1 presents the data structure and Section 3.3.2 describes the complex-to-real conversion.

3.3.1 Data structure

The training data and the testing data are generated at certain SNR levels. For each channel realisation at each SNR point specified, the simulation collects four quantities at the receiver after FFT processing. The received symbols $\mathbf{Y} \in \mathbb{C}^{K\times(I+2)}$ contain frequency-domain observations spanning preamble and data symbols. The true channels $\mathbf{H} \in \mathbb{C}^{K\times(I+2)}$ provide the ground truth frequency response at each subcarrier and symbol. The LS estimates $\widehat{\mathbf{H}}_{\mathrm{LS}} \in \mathbb{C}^{K_{\mathrm{act}}\times 1}$ derived from preamble averaging serve as baseline estimates, where $K_{\mathrm{act}} = 52$ denotes the number of active subcarriers. The data symbols $\mathbf{X} \in \mathbb{C}^{K\times I}$ contain known data and pilot values for $I = 50$ OFDM symbols.

After extracting the preamble, we retain only the active subcarriers for the data symbols,

resulting in $\mathbf{Y} \in \mathbb{C}^{K_{\text{act}} \times I}$ and $\mathbf{H} \in \mathbb{C}^{K_{\text{act}} \times I}$ where $K_{\text{act}} = 52$ active subcarriers and $I = 50$ data symbols per frame. This reduction focuses the learning task on subcarriers that carry information while excluding null subcarriers.

3.3.2 Complex-to-real transformation

NNs typically operate on real-valued inputs. We transform complex-valued channel data into real-valued representations by separating each complex quantity into real and imaginary components and stacking them as feature channels along the subcarrier dimension, following the standard approach in the literature [6]–[8], [17], [66], [68]:

$$\mathbf{X}' = \begin{bmatrix} \Re(\mathbf{Y}) \\ \Im(\mathbf{Y}) \end{bmatrix} \in \mathbb{R}^{2K_{\text{act}} \times I}. \tag{3.4}$$

The target output for supervised learning excludes pilot subcarriers, focusing only on the data subcarrier positions:

$$\mathbf{H}_{\text{target}} = \begin{bmatrix} \Re(\mathbf{H}[\mathcal{K}_d, :]) \\ \Im(\mathbf{H}[\mathcal{K}_d, :]) \end{bmatrix} \in \mathbb{R}^{2K_d \times I} = \mathbb{R}^{96 \times 50}, \tag{3.5}$$

where $\mathbf{H}[\mathcal{K}_d, :]$ selects the $K_d = 48$ data subcarrier rows from the true channel matrix. This design choice trains the network to estimate channels at data positions, with pilot-position estimates derived separately from the known pilot symbols or classical methods.

This method of stacking the real and imaginary parts of the complex data along the subcarrier dimension is used for the FNNs, LSTM-based and the GRU models. For the TCN models, we introduce an alternative method to convert complex data to real-valued data. This method is outlined in the next chapter (Section 4.6.1.1).

3.4 Dataset validation and reproducibility

This section describes the procedures used to validate that the training and test subsets are representative and free from duplicated samples, and to ensure that the datasets can

be reproduced exactly. These procedures support the claim that the DL models evaluated in this thesis learn generalisable channel estimation behaviour rather than relying on memorisation. All datasets used throughout this thesis employ a random splitting strategy.

The dataset generation procedures follow standardised protocols to ensure reproducibility. Channel generation employs the deterministic seed assignment specified in Section 3.2.2, creating a one-to-one mapping between frame index and random seed. This mapping ensures that frame index i always produces identical channel realisations when using the same channel model parameters from Table 2.3, carrier frequency $f_c = 5.9$ GHz, and sampling rate $f_s = 10$ MHz. The interleaving permutation vector used in frame generation (Section 3.2.3) employs a fixed seed which is stored in the dataset metadata, ensuring identical permutations across all frames during both interleaving and deinterleaving.

Simulation scripts and dataset generation code are maintained in a public Git repository with documentation.[3] The repository includes instructions describing execution procedures, dependency requirements, and configuration parameters required to reproduce the datasets.

3.5 Dataset description

This section describes the final datasets used for model training and evaluation. A separate dataset is generated for each of the six vehicular channel models from Section 3.2.2, providing distinct propagation environments. Section 3.5.1 presents the dataset composition, and Section 3.5.2 describes the statistical properties of the generated data. To validate the dataset's quality, Sections 3.5.3 and 3.5.4 analyse feature-level consistency using empirical distributions and the Kolmogorov-Smirnov test. Finally, Section 3.5.5 evaluates the independence of training and test samples to confirm dataset integrity.

[3] `https://github.com/SimbaAl/PhD_XAI_Codebase`

3.5.1 Dataset composition

For each channel model, 20,000 independent OFDM frames are generated following the procedure in Section 3.2.2. Each frame corresponds to a unique channel realisation with its own random seed according to Section 3.2.2. Specifically, a realisation is defined by a distinct combination of time-varying multipath fading coefficients, Doppler shift trajectories, and a complex Gaussian noise instance determined by an SNR value set to a specific SNR level for training data generation and drawn uniformly from the range $[0, 40]$ dB for the testing data.

To facilitate both supervised training and performance evaluation, each dataset is packaged as a tuple $\{\mathbf{Y}, \mathbf{H}, \mathbf{X}, \widehat{\mathbf{H}}_{\text{pre}}^{\text{LS}}\}$ containing the following tensors:

1. **Received Signal (Y):** The frequency-domain received grid of size $K \times (I + 2)$, which serves as the primary observation for the estimator.
2. **True Channel (H):** The ground-truth frequency response grid of size $K \times (I + 2)$, serving as the target label for supervised loss calculation.
3. **Transmitted Symbols (X):** The grid of transmitted QAM symbols of size $K \times (I + 2)$, required for computing the BER during the evaluation phase.
4. **Coarse Channel Estimates ($\widehat{\mathbf{H}}_{\text{coarse}}$):** This set includes the initial preamble-based LS estimate ($\widehat{\mathbf{H}}_{\text{pre}}^{\text{LS}}$) and, for hybrid model configurations, the time-varying classical estimates (e.g., DPA, STA, TRFI) derived from $\mathbf{Y}$. These serve as the input features for hybrid DL architectures that aim to refine coarse classical predictions.

The datasets are split into 18,000 training and 2,000 test frames using random allocation. This split ensures that the training and testing sets sample the full range of channel realisations while maintaining statistical independence between subsets.

From this point on, the same six datasets: VTV-SDWW, VTV-EX, VTV-UC, RTV-SS, RTV-EX, and RTV-UC, and the same training and test partitions are used throughout Chapters 4 and 5. In Chapter 6, the dataset is adapted to 64QAM modulation.

3.5.2 Temporal and spectral correlation structure

The dataset exhibits specific statistical patterns arising from the channel generation process. Consecutive frames have statistically independent channel realisations because each frame uses a unique random seed. Frame ordering contains no information about channel evolution and does not reflect physical propagation patterns.

Within each frame, channel coefficients are expected to display structured correlation across both frequency and time, and these relationships are expected to be nonlinear and complex. Frequency selectivity causes adjacent subcarriers to experience correlated fading due to multipath delay spread characterised by the Power Delay Profile (PDP) in Table 2.3. The degree of frequency correlation depends on the coherence bandwidth relative to subcarrier spacing, as discussed in Section 2.2.3.

Time variation creates correlation between adjacent OFDM symbols within a frame, governed by the Jakes' Doppler spectrum and coherence time characteristics from Section 2.2.2. The normalised Doppler products ranging from 0.125 to 0.499 indicate that significant temporal evolution occurs across the 50 symbols within each frame. This temporal correlation motivates the use of architectures that can exploit temporal structure, such as RNNs and CNNs.

3.5.3 Feature distribution analysis

We first examined whether individual real-valued feature dimensions are statistically consistent across the dataset partitions. A feature is defined as either the real or imaginary component of a channel coefficient at a specific subcarrier-symbol index, yielding $2K_{\text{act}}I = 5\,200$ features per frame.

Figure 3.1 compares empirical cumulative distribution functions (CDFs) for eight random features in the training and test sets. The near-perfect overlap indicates strong similarity in feature statistics. Additionally, Figure 3.2 confirms that the magnitude distributions follow Rayleigh characteristics expected from the vehicular fading models, with no abnor-

mal outliers or skewness. These observations support the use of standard z-normalisation, in which each feature is transformed to zero mean and unit variance by subtracting the feature mean and dividing by the feature standard deviation. The mean and standard deviation are computed over the training set only and then applied to the validation and test sets, ensuring correct statistical isolation.

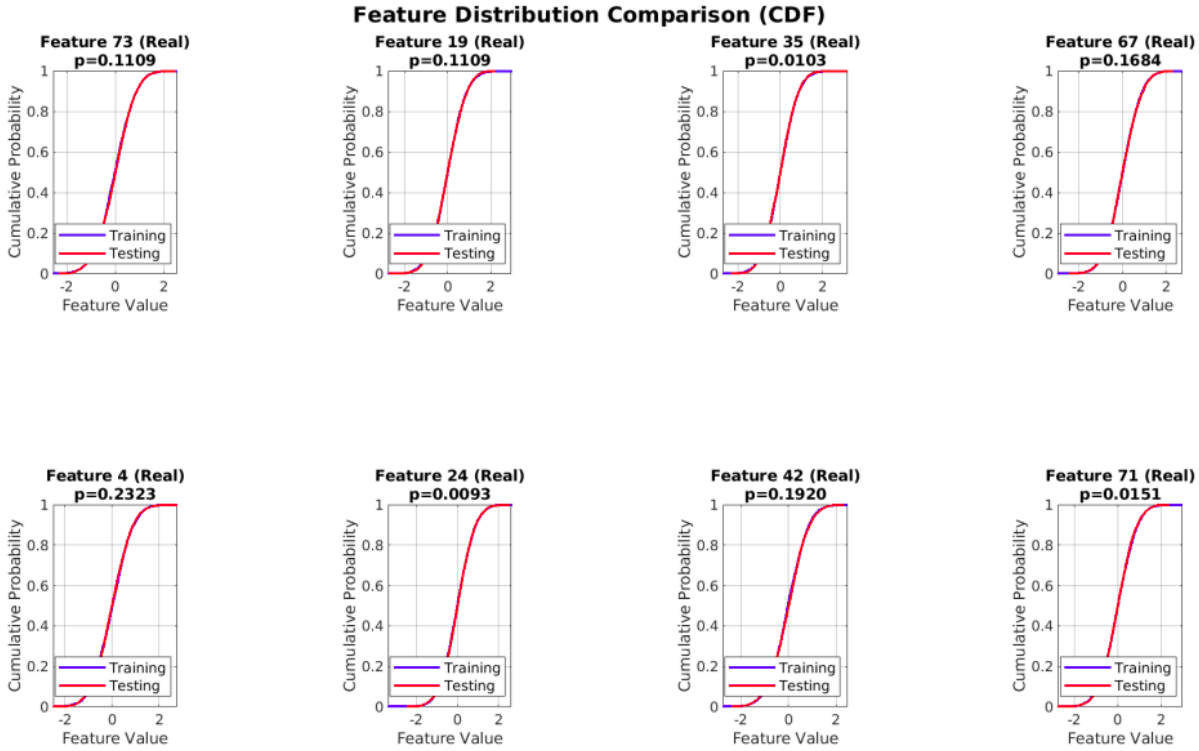


Figure 3.1: CDFs of 8 random features from the training and test sets. The overlap indicates distributional consistency, where p represents the probability that the two samples are drawn from the same underlying distribution.

3.5.4 Kolmogorov-Smirnov feature-level consistency

To statistically verify that the two subsets originate from the same population, we applied the Kolmogorov-Smirnov (KS) test to each feature dimension. The KS test fails to reject the null hypothesis (identical distributions) for 85.3% of features ($p > 0.05$), indicating strong alignment between the training and test subsets. Features with $p < 0.05$

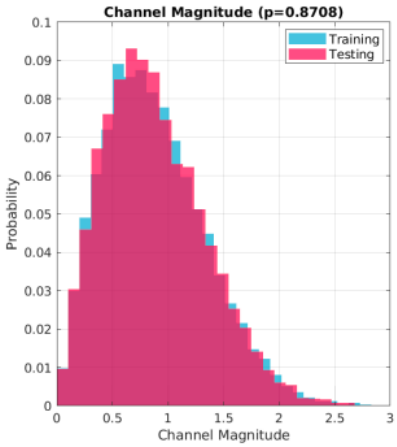


Figure 3.2: Distribution of channel magnitude for training and testing sets. The close match reflects consistent sampling of channel attenuation statistics across dataset splits.

are sparsely distributed and do not correspond to any specific subcarrier or temporal structure, meaning that no systematic distribution shift is introduced by the split.

3.5.5 Train/test sample independence

To ensure that DNNs cannot benefit from channel memorisation, we evaluated redundancy across dataset partitions. For each test sample, we computed its Euclidean distance to the closest training sample using the concatenated $(\mathbf{Y}, \mathbf{H})$ representation, where $\mathbf{Y}$ denotes the received frequency-domain grid and $\mathbf{H}$ represents the true channel response grid. The resulting nearest-neighbour distances in Figure 3.3 show a clear separation between all sample pairs, with a minimum observed distance of 8.37. Given our similarity threshold of 1.0, this confirms that no test sample is an exact or near-duplicate of any training sample. This confirms that the train/test separation provides statistically independent channel conditions, enabling reliable generalisation evaluation.

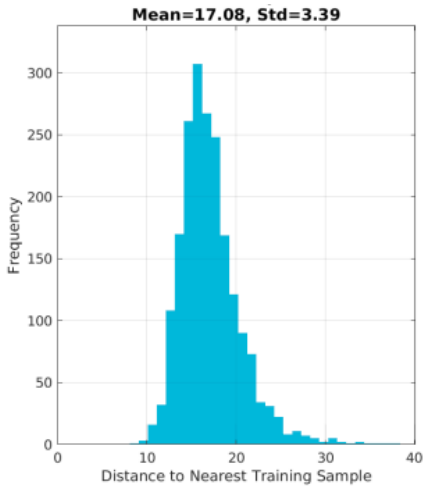


Figure 3.3: Nearest-neighbour distances between test samples and the closest training samples. The high separation values confirm that no duplicated channel realisations exist across the dataset split.

3.6 Chapter summary

This chapter describes the data-generation process for evaluating DL channel estimators. The simulation implements IEEE 802.11p with six vehicular channel models, producing 20,000 frames per model, of which 18,000 are used for training and 2,000 for testing. Preprocessing converts complex observations into real-valued inputs for NNs, with deterministic procedures that ensure reproducibility.

Statistical validation confirmed the integrity of this partitioning strategy: KS tests demonstrated distributional consistency between training and test sets, while Euclidean distance analysis verified that no test sample is a duplicate of a training sample.

All datasets used in this work are generated following this process, except for one of the datasets used in Chapter 6, where the same process is used but some of the parameters are changed. (The modulation scheme is changed to 64QAM and vehicle mobility is changed to 200 km/h.) The datasets developed in this chapter enable the evaluation of DL architectures for channel estimation in the chapters that follow.

Chapter 4

Deep learning architectures for channel estimation

This chapter presents DL architectures to improve channel estimation performance in vehicular communication systems. We introduce a dual-cell LSTM and a TCN model.

4.1 Introduction

This chapter evaluates DL architectures for channel estimation using the datasets generated as described in Chapter 3. Building on the estimators and theoretical foundations from Chapter 2, we investigate four architecture families that integrate NNs with classical methods to improve estimation accuracy. The chapter is organised as follows. To establish a fair comparison, we first identify two primary baselines derived from the literature review. We select FNN-based hybrid estimators (Section 4.3) to represent established low-complexity methods that improve the estimation of classical estimators. Complementing these, we select the recurrent LSTM-DPA-TA architecture (Section 4.4) as the baseline for sequential channel estimation. Against these baselines, we introduce and evaluate

two new estimators designed to address specific limitations in current vehicular channel estimation. Section 4.5 introduces a new estimator (dual-cell LSTM architecture), while Section 4.6 introduces an estimator based on TCNs. Finally, a comparative analysis of the DL estimators across various channel models is presented in Section 4.7. Section 4.8 summarises the findings.

4.2 Experimental setup

To ensure a fair comparison between the proposed architectures and the baselines, we established a unified experimental framework. This section details the experimental configuration, datasets, and hyperparameter optimisation methodology used to evaluate all the architectures discussed in this chapter.

4.2.1 Dataset configuration

All models developed and evaluated in this chapter use datasets generated according to the procedure described in Chapter 3. The channel models used in this chapter are VTV-SDWW and VTV-EX. The other channel models are analysed in the next chapter. The properties of these channel models are described in Section 2.3.3. Unless otherwise specified, the modulation scheme used in this chapter is 16QAM.

4.2.2 Hyperparameter optimisation protocol

To ensure fair comparison across architectures, we employ a consistent hyperparameter optimisation methodology using Optuna [97], a Bayesian optimisation framework that implements the Tree-structured Parzen Estimator (TPE) sampling algorithm. This approach provides efficient exploration of the hyperparameter space based on trial performance [98]. To maximise computational efficiency, the framework applies adaptive pruning via median stopping rules, automatically terminating trials that fail to meet performance thresh-

olds early in the training process. Beyond simple accuracy maximisation, the protocol addresses the trade-off between performance and efficiency by integrating the NSGA-II sampler [99]. This genetic algorithm enables simultaneous multi-objective optimisation, identifying a set of solutions that balance minimal validation loss with reduced model complexity.

Common hyperparameters. Across all NN architectures, we optimise the following hyperparameters:

- **Learning rate:** Searched in log-space $[10^{-5}, 10^{-1}]$ to span multiple orders of magnitude.
- **Learning rate scheduling:** StepLR scheduler with optimised step size and gamma decay.
- **Regularisation:** Dropout probability and weight decay (where applicable).
- **Batch size:** Is fixed at 128 for all the architectures.
- **Optimiser:** Adam with default PyTorch parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$).

Architecture-specific hyperparameters. Each architecture family requires additional parameters:

- **FNNs:** Hidden layer sizes.
- **RNNs:** Hidden state size, number of recurrent layers.
- **TCNs:** Number of resource blocks, number of layers, kernel size, dilation pattern.

Training protocol. For baseline construction, we conduct hyperparameter optimisation with the following protocol:

- **Definition of search space:** A wide range of values are defined for the hyperparameters being optimised, for example, learning rate, batch size, regularisation (weight decay, dropout) and architectural dimensions. Post-trial, if a value was found on the edge of the range, the range was extended.
- **Optimisation trials:** 50-180 trials depending on architecture complexity.
- **Epochs per trial:** Early stopping (patience = 20 epochs) was used to control the number of epochs per trial.
- **Validation metric:** Minimum validation loss across training epochs.
- **Random seeds:** Unless specified, each hyperparameter configuration during optimisation is evaluated with three different random seeds to reduce sensitivity to weight initialisation.
- **Final training:** Best configuration trained using early stopping with patience =20.

All the models evaluated in this chapter were trained at 40 dB SNR, following the protocol established in previous work [8], [68], where models learn channel dynamics from high SNR samples. The impact of this design choice on generalisation to lower SNRs is investigated in Chapter 5.

Experiment tracking. All experiments are tracked using Weights & Biases[1], recording training curves, hyperparameter configurations, and performance metrics to ensure reproducibility.

4.2.3 Performance evaluation

Model performance is evaluated using the metrics defined in Section 2.8:

- **NMSE** (Section 2.8.1): Quantifies channel estimation accuracy by measuring the normalised mean squared error between estimated and true channel coefficients.

[1] `https://wandb.ai`

- **BER** (Section 2.8.2): Measures end-to-end communication performance after channel equalisation, demodulation, and decoding.

Both metrics are computed over a 0-40 dB SNR range to assess performance under varying noise conditions. All results are presented on logarithmic scales for BER and NMSE to clearly show the performance of the evaluated estimators across the SNR range.

4.3 Feedforward networks

FNNs refine classical channel estimates by learning nonlinear correction patterns. This section evaluates FNN-based estimators that post-process STA and TRFI outputs, following the hybrid architecture framework introduced in Section 2.6.4.2.

4.3.1 Architecture

As discussed in Section 2.6.4.2, Gizzini *et al.* [6], [66] proposed two FNN architectures: STA-DNN and TRFI-DNN. These networks are designed to address the specific limitations of classical estimators by learning to map the noise-corrupted, interpolated estimates to a more accurate channel response.

STA limitations. The STA estimator applies temporal and frequency averaging with fixed coefficients α and β (see Section 2.6.4.2). These fixed coefficients degrade performance in high-mobility conditions because they cannot adapt to varying Doppler profiles [68]. An FNN can learn data-driven corrections to compensate for this mismatch.

TRFI limitations. The TRFI estimator uses frequency-domain interpolation between pilot subcarriers, but suffers from interpolation errors at low SNR when pilot estimates are unreliable [66]. An FNN can learn to suppress these interpolation artefacts through supervised training on ground-truth channels.

Network architecture. Both STA-DNN and TRFI-DNN follow the same architectural template, consisting of three hidden layers with 15 units each and ReLU activation:

- **Input:** Real-valued stacking of classical estimate $\mathbf{x}'_n = f_{\mathbb{R}}(\widehat{\mathbf{H}}_n^{\mathrm{STA}})$ or $f_{\mathbb{R}}(\widehat{\mathbf{H}}_n^{\mathrm{TRFI}}) \in \mathbb{R}^{2K_{\mathrm{act}}}$.
- **Hidden layers:** Three fully connected layers with 15 units each, ReLU activation, and dropout.
- **Output layer:** Linear projection to refined channel estimate $\widehat{\mathbf{H}}_n^{\mathrm{NN}} \in \mathbb{R}^{2K_{\mathrm{act}}}$.
- **Target:** Ground-truth channel $\mathbf{H}_n^{\mathrm{target}} = f_{\mathbb{R}}(\mathbf{H}_n) \in \mathbb{R}^{2K_{\mathrm{act}}}$.
- **Weight initialisation:** Truncated normal distribution with mean 0 and standard deviation 0.05 for all layers.

The network minimises the MSE loss between the network output $\widehat{\mathbf{H}}_n^{\mathrm{NN}}$ and the ground-truth target $\mathbf{H}_n^{\mathrm{target}}$:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{H}_n^{\mathrm{target}} - \widehat{\mathbf{H}}_n^{\mathrm{NN}} \right\|_2^2, \tag{4.1}$$

where N denotes the total number of training samples in the dataset, with each sample corresponding to one OFDM symbol. It is important to note that the FNN models discussed in this section use a single OFDM symbol as input, following the approach outlined in [6]. In contrast, other architectures evaluated in this study, such as Dual-cell LSTM, LSTM-DPA-TA, GRU-DPA-TA, and DPA-TCN, process the input as a whole frame, which consists of 50 OFDM symbols and 52 subcarriers.

4.3.2 Hyperparameter optimisation

Following the framework in Section 4.2.2, we optimise FNN hyperparameters using Optuna. The network architecture follows the original design from Gizzini *et al.* [6], [66], with three hidden layers of 15 units each, which has been demonstrated to be sufficient for a similar task. We fix the batch size to 128 and optimise the remaining hyperparameters

over 100 trials of 100 epochs each. The search space and optimal values are detailed in Table 4.1.

Table 4.1: FNN hyperparameter search space and optimal values (100 trials, 100 epochs each; final model trained for 250 epochs).

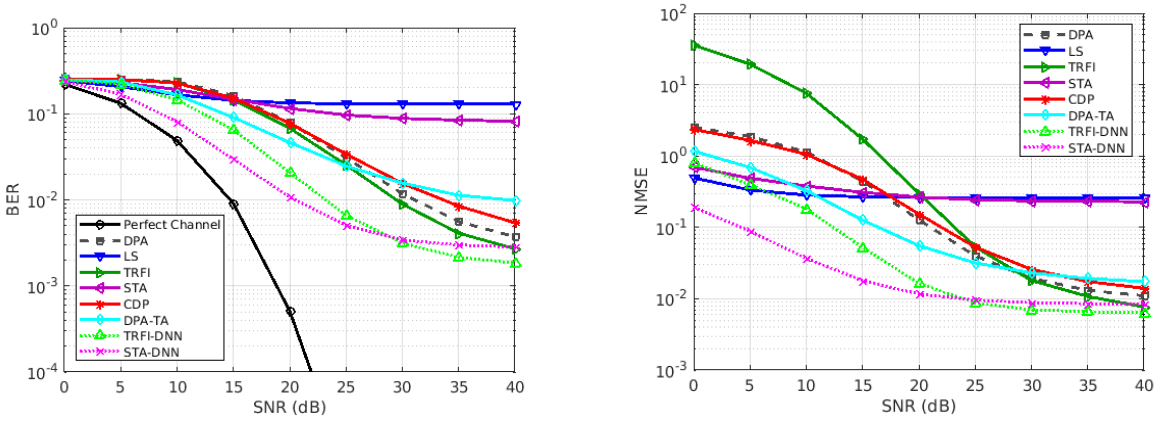
Hyperparameter	**Search space**	**Best value**
Learning rate	$[10^{-5}, 10^{-2}]$	5×10^{-4}
Dropout probability	[0.0, 0.5]	0.1
Weight decay	$[10^{-6}, 10^{-3}]$	10^{-5}
StepLR step size	[5, 20]	10
StepLR gamma	[0.5, 0.9]	0.7

The best configuration uses a learning rate of 5×10^{-4} with StepLR scheduling (step size = 10, gamma = 0.7), moderate dropout (0.1), and minimal weight decay (10^{-5}) for regularisation. The fixed three-layer architecture with 15 hidden units per layer reflects the limited complexity of the correction task: the FNN only needs to learn local nonlinear mappings to compensate for systematic errors in STA and TRFI estimates, not long-range temporal dependencies.

4.3.3 Results and discussion

Figure 4.1(a) and Figure 4.1(b) compare the FNN estimators with the classical channel estimators. Perfect Channel is the theoretical lower bound assuming perfect CSI, serving as the reference to assess estimation losses. Therefore, the closer the BER of an estimator to the perfect channel, the better the performance. LS estimation, which uses only the preamble symbols to estimate the channel once per frame, demonstrates performance degradation with a BER plateau above 10^{-1}. This occurs because the initial estimate becomes outdated as the channel varies throughout the frame in high-mobility scenarios. The STA estimator aims to improve the initial channel estimation through frequency-domain averaging and time-domain averaging using the coefficient α, which

shows a performance degradation around 10^{-1}. This is due to its fixed averaging parameters, which cannot adapt to time-varying channels. CDP, which implements a reliability test by comparing demapping results from the current and previous channel estimates and updates only when estimates are reliable, achieves a BER performance of approximately 10^{-2} at 40 dB. However, the binary reliability decision still allows some error propagation. TRFI extends CDP by using cubic interpolation to estimate unreliable subcarriers based on reliable ones. Despite this, TRFI shows better performance than CDP from 20 dB. The DPA estimator, which updates the channel estimate per symbol using demapped symbols, shows better performance than LS, CDP, and STA. Because it suffers from error propagation, where demapping errors compound across symbols, its performance is limited. DPA-TA, which aims to improve the DPA estimates by averaging in the temporal domain, shows a performance improvement to the DPA estimate in low SNRs, but at 30 dB, DPA performs better than it.



(a) BER performance.

(b) NMSE performance.

Figure 4.1: Performance of FNN-based estimators compared with classical channel estimation methods under the VTV-SDWW channel model.

STA-DNN and TRFI-DNN, which apply FNNs as post-processors to the classical estimators (STA and TRFI), show noticeable improvements over classical estimators across all SNRs. However, from 30 dB, TRFI-DNN starts to perform better than STA-DNN. The NMSE results showed that STA-DNN and TRFI-DNN outperform their classical counterparts (STA and TRFI), confirming that adding the NN effectively reduces estimation

error. In low dB (0 to 5 dB), the TRFI-DNN shows performance worse than that of the LS estimator. This is because its classical counterpart (TRFI) also has a very high NMSE in this SNR range, which then makes it very difficult for the NN to correct this high NMSE.

4.4 Recurrent baselines

FNNs process each OFDM symbol independently, ignoring the temporal correlation which exists across OFDM symbols in a single frame. RNNs address this limitation by modelling the channel as a time series, using memory states to capture temporal evolution in fast-fading vehicular environments.

4.4.1 Architecture

We implemented two RNN architectures from prior work: LSTM-DPA-TA [8] and GRU-DPA-TA [68] as our baselines. Both architectures extend the DPA estimator from Section 2.5.3.2 with recurrent processing, but differ in their gating mechanisms. LSTMs maintain separate memory and hidden states with three gates (input, forget, output), while GRUs combine these into a single state with two gates (reset, update). RNNs treat the wireless channel as a time series by estimating the current channel response from past OFDM symbols to exploit temporal correlation in fast-fading vehicular environments. Unlike the FNN estimators, these sequential architectures model time evolution through memory states, improving robustness against rapid Doppler variations.

Input construction. Let n index OFDM symbols and k index active subcarriers. The input at symbol n is constructed from previous estimates:

$$\bar{H}_n[k] = \begin{cases} \widehat{H}_{n-1}[k], & k \in \mathcal{K}_d, \\ \widehat{H}^{\mathrm{LS}}_{n-1}[k], & k \in \mathcal{K}_p, \end{cases} \tag{4.2}$$

where $\mathcal{K}_d$ and $\mathcal{K}_p$ denote the data and pilot subcarrier sets, respectively, with $\widehat{H}_0[k] = \widehat{H}^{\mathrm{LS}}[k]$ for initialisation. Each complex input vector $\bar{\mathbf{H}}_n \in \mathbb{C}^{K_{\mathrm{act}}}$ is converted into a real-valued representation

$$\mathbf{x}'_n = [\Re\{\bar{\mathbf{H}}_n\}, \Im\{\bar{\mathbf{H}}_n\}] \in \mathbb{R}^{2K_{\mathrm{act}}}, \tag{4.3}$$

following the complex-to-real transformation in Section 3.3.2.

Recurrent processing. The recurrent unit, denoted $\psi(\cdot)$, models temporal dependencies as

$$\mathbf{s}_n = \psi(\mathbf{s}_{n-1}, \mathbf{x}'_n; \Theta), \qquad \widehat{\mathbf{H}}_n^{\mathrm{RNN}} = \Omega(\mathbf{s}_n), \tag{4.4}$$

where $\psi(\cdot)$ represents either an LSTM or GRU cell with parameters Θ, $\mathbf{s}_n$ is the hidden state at symbol n, and $\Omega(\cdot)$ is the output layer that maps the hidden state to the channel estimate.

DPA refinement. DPA is used as a refinement step to refine the RNN output, exploiting the temporal correlation across adjacent OFDM symbols. For each data subcarrier $k \in \mathcal{K}_d$, the received signal is first equalised using the previous RNN estimate:

$$\widehat{H}_n^{\mathrm{eq}}[k] = \frac{Y_n[k]}{\widehat{H}_{n-1}^{\mathrm{RNN}}[k]}, \tag{4.5}$$

where $Y_n[k]$ denotes the received symbol on subcarrier k of the n-th OFDM symbol. The equalised signal is then demapped to the nearest constellation point:

$$\widehat{d}_n[k] = \mathcal{D}\big(\widehat{H}_n^{\mathrm{eq}}[k]\big), \tag{4.6}$$

where $\mathcal{D}(\cdot)$ represents the constellation demapping operator. The demapped symbol is then used as a known reference to compute the DPA estimate:

$$\widehat{H}_n^{\mathrm{DPA}}[k] = \frac{Y_n[k]}{\widehat{d}_n[k]}. \tag{4.7}$$

For pilot subcarriers $k \in \mathcal{K}_p$, the known pilot symbols $X_n^{\mathrm{p}}[k]$ are used directly without demapping.

Temporal averaging. To reduce the impact of AWGN, TA is applied to $\widehat{H}_n^{\mathrm{DPA}}[k]$ as follows:

$$\widehat{H}_n^{\mathrm{TA}}[k] = \left(1 - \frac{1}{\alpha}\right) \widehat{H}_{n-1}^{\mathrm{TA}}[k] + \frac{1}{\alpha} \widehat{H}_n^{\mathrm{DPA}}[k], \tag{4.8}$$

where α controls the smoothing strength (typically $\alpha = 2$ [8]).

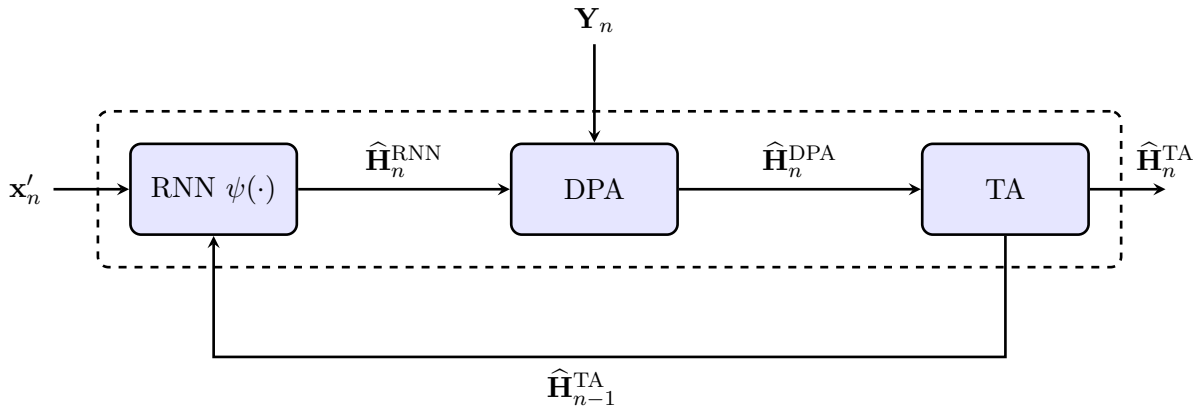


Figure 4.2: RNN-based channel estimation pipeline using DPA refinement and temporal averaging.

Figure 4.2 shows the complete RNN-based channel estimation pipeline. The RNN processes input $\mathbf{x}'_n$ to predict the channel $\widehat{\mathbf{H}}_n^{\mathrm{RNN}}$, which is then used to equalise and demap the received symbol $\mathbf{Y}_n$. The DPA stage refines the estimate by treating the demapped symbols as pseudo-pilots to re-estimate the channel, producing $\widehat{\mathbf{H}}_n^{\mathrm{DPA}}$. Finally, temporal averaging smooths the estimate to obtain $\widehat{\mathbf{H}}_n^{\mathrm{TA}}$, which feeds back to construct the input for the next symbol. LSTM-DPA-TA uses 128 hidden units, while GRU-DPA-TA uses 48 hidden units, following the original configurations [8], [68].

4.4.2 Hyperparameter optimisation

Following the framework in Section 4.2.2, we optimise RNN hyperparameters using Optuna. The baseline architectures follow the original configurations from [8], [68]. The LSTM-DPA-TA uses 128 hidden units with a single LSTMCell layer, while GRU-DPA-TA uses 48 hidden units with a single GRUCell layer. Both architectures map from 104-dimensional inputs (real and imaginary components of 52 active subcarriers) to 96-

dimensional outputs (real and imaginary components of 48 data subcarriers). We fix the batch size to 128 and conduct 80 trials of 200 epochs each with early stopping, optimising the hyperparameters detailed in Table 4.2. The optimal configuration for both architec-

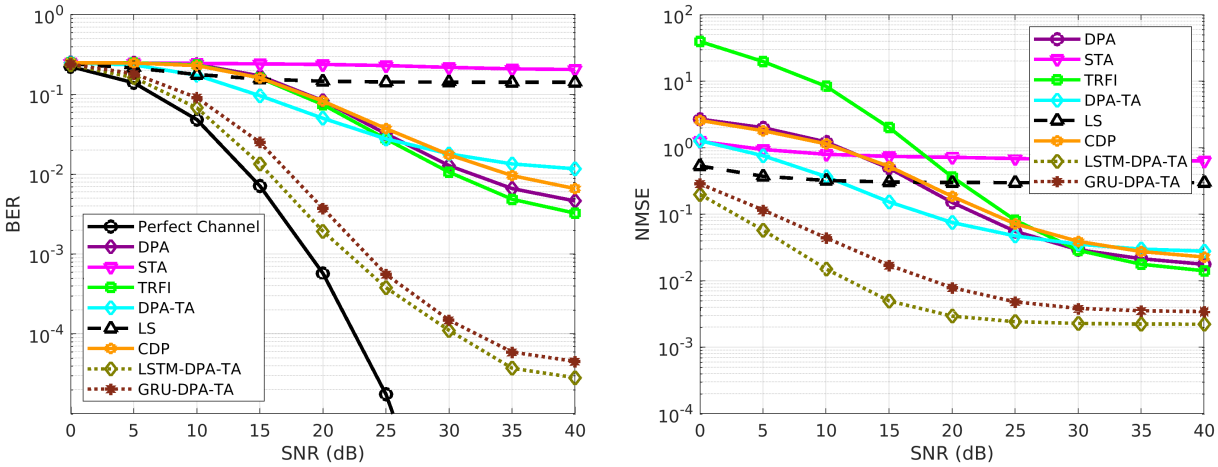
Table 4.2: RNN hyperparameter search space and optimal values (80 trials, 200 epochs each).

Hyperparameter	**Search space**	**LSTM-DPA-TA**	**GRU-DPA-TA**
Learning rate	$[10^{-5}, 10^{-2}]$	10^{-3}	10^{-3}
StepLR step size	[5, 20]	10	20
StepLR gamma	[0.5, 0.9]	0.8	0.9
Gradient clipping	$[10^{-5}, 10^{-2}]$	10^{-4}	—

tures uses the Adam optimiser, a learning rate of 10^{-3} and no weight decay. The LSTM implementation includes gradient clipping at 10^{-4} to prevent exploding gradients during backpropagation through time, while the GRU architecture achieves stable training without gradient clipping. The temporal averaging coefficient is fixed at $\alpha = 2$ for both architectures, following [8].

4.4.3 Results and discussion

Figure 4.3 compares the RNN baselines with FNN and classical estimators under the VTV-EX channel model. Figure 4.3 shows that under the VTV-EX channel model and mobility conditions, the recurrent channel estimators (GRU-DPA-TA and LSTM-DPA-TA) achieve up to two orders of magnitude BER improvement over classical estimators (DPA-TA, TRFI, CDP) for SNR $\geq$ 25 dB. This confirms that sequential learning architectures effectively capture temporal channel evolution that traditional estimators ignore. The NMSE result also shows the better performance of the LSTM-DPA-TA and the GRU-DPA-TA than the FNNs and the classical estimators.



(a) BER performance.

(b) NMSE performance.

Figure 4.3: RNN-based estimators compared with FNN and classical methods under VTV-EX.

4.5 Dual-cell LSTM

Single-cell RNNs process all temporal dynamics through one gating pathway, which can limit channel representation diversity and generalisation [100]. We propose a dual-cell LSTM architecture that extends LSTM-DPA-TA by using two independent LSTM cells operating in parallel on the same input. The outputs are averaged before DPA refinement, introducing recurrent diversity that improves robustness in high-Doppler shift conditions. This approach remains practical for real-time vehicular communication because both LSTM cells operate only within each frame of 50 OFDM symbols and are reinitialised from the preamble at each new frame, keeping memory requirements bounded and computational overhead manageable.

4.5.1 Architecture

The proposed dual-cell design introduces recurrent diversity by allowing two LSTM branches to learn complementary temporal features within each frame of 50 OFDM symbols. Averaging their outputs yields a more stable and generalised estimate, improving robustness in rapidly varying vehicular channels, while both cells maintain independent hidden states that are reinitialised at the start of each new frame.

Input construction. Let n and k denote the OFDM symbol and subcarrier indices, respectively. The received frequency-domain signal on subcarrier k of symbol n is

$$Y_n[k] = \begin{cases} H_n[k]X_n[k] + N_n[k], & k \in \mathcal{K}_d, \\ H_n[k]X_n[k] + N_n[k], & k \in \mathcal{K}_p, \end{cases} \tag{4.9}$$

where $X_n[k]$ and $N_n[k]$ are the transmitted OFDM symbol (or pilot) and complex AWGN, respectively.

Inputs to the NN are constructed from the previous channel estimates on active subcarriers. For data subcarriers, the input uses the most recent network estimate, while for pilot subcarriers, it reuses the LS estimate:

$$\bar{H}_n[k] = \begin{cases} \widehat{H}_{n-1}[k], & k \in \mathcal{K}_d, \\ \widehat{H}^{\mathrm{LS}}_{n-1}[k], & k \in \mathcal{K}_p. \end{cases} \tag{4.10}$$

Since the network operates on real-valued tensors, each complex vector $\bar{\mathbf{H}}_n \in \mathbb{C}^{K_{\mathrm{act}}}$ is transformed into a real-valued representation using the complex-to-real stacking procedure described in Section 3.3.2:

$$\mathbf{x}'_n = \left[\Re\{\bar{\mathbf{H}}_n\},\ \Im\{\bar{\mathbf{H}}_n\}\right] \in \mathbb{R}^{2K_{\mathrm{act}}}. \tag{4.11}$$

Targets are defined analogously by stacking the real and imaginary parts of the true channel response on data subcarriers:

$$\mathbf{t}_n = \left[\Re\{\mathbf{H}_n[\mathcal{K}_d]\},\ \Im\{\mathbf{H}_n[\mathcal{K}_d]\}\right] \in \mathbb{R}^{2K_d}. \tag{4.12}$$

Dual-cell recurrence. The recurrent update consists of two parallel LSTM cells, each with 128 hidden units and independent parameters:

$$\mathbf{s}^{(1)}_n = \psi_1(\mathbf{s}^{(1)}_{n-1}, \mathbf{x}'_n; \Theta_1), \qquad \mathbf{s}^{(2)}_n = \psi_2(\mathbf{s}^{(2)}_{n-1}, \mathbf{x}'_n; \Theta_2), \tag{4.13}$$

where $\psi_1(\cdot)$ and $\psi_2(\cdot)$ denote the two independent LSTM mappings, and Θ_1, Θ_2 are their respective parameter sets. Both cells process the same input sequence $\mathbf{x}'_n$ in the forward

direction but maintain separate hidden states. Each branch produces an intermediate channel estimate through a linear readout $\Omega_1(\cdot)$ and $\Omega_2(\cdot)$:

$$\widehat{\mathbf{H}}_n^{(1)} = \Omega_1(\mathbf{s}_n^{(1)}), \qquad \widehat{\mathbf{H}}_n^{(2)} = \Omega_2(\mathbf{s}_n^{(2)}). \tag{4.14}$$

The final recurrent output is the elementwise average:

$$\widehat{\mathbf{H}}_n^{\mathrm{RNN}} = \tfrac{1}{2}\Big(\widehat{\mathbf{H}}_n^{(1)} + \widehat{\mathbf{H}}_n^{(2)}\Big), \tag{4.15}$$

which is subsequently used for DPA refinement.

DPA refinement. DPA refinement follows the same procedure as the baselines:

$$\widehat{H}_n^{\mathrm{eq}}[k] = \frac{Y_n[k]}{\widehat{H}_{n-1}^{\mathrm{RNN}}[k]}, \qquad \widehat{d}_n[k] = \mathcal{D}\Big(\widehat{H}_n^{\mathrm{eq}}[k]\Big), \qquad \widehat{H}_n^{\mathrm{DPA}}[k] = \frac{Y_n[k]}{\widehat{d}_n[k]}, \tag{4.16}$$

where $\mathcal{D}(\cdot)$ denotes the constellation demapper. For pilot subcarriers $k \in \mathcal{K}_p$, the known pilot symbols $X_n^{\mathrm{p}}[k]$ are used directly without demapping.

4.5.2 Hyperparameter optimisation

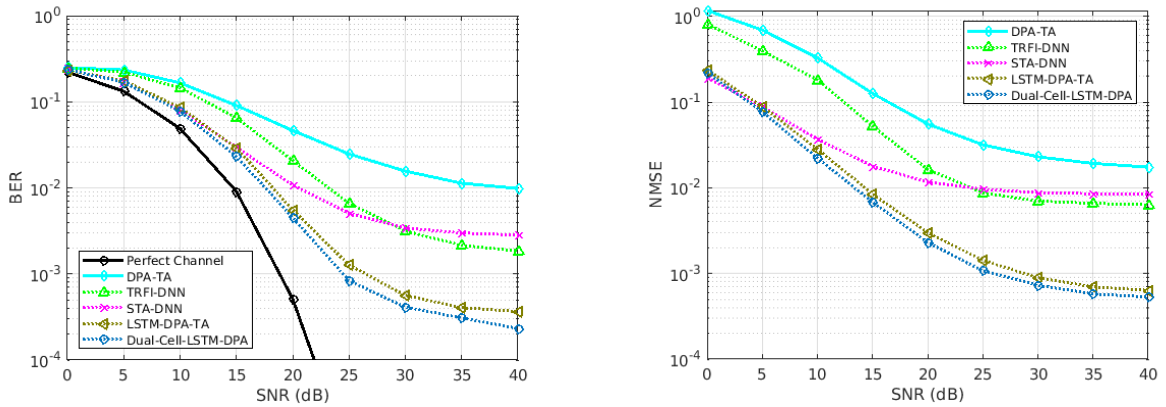
We optimise the dual-cell LSTM hyperparameters following the framework established in Section 4.2.2, with modifications to account for the increased model capacity. The dual-cell architecture uses two independent LSTM cells, each with 128 hidden units, processing 104-dimensional inputs and producing 96-dimensional outputs. The search space and best values are detailed in Table 4.3. The final configuration uses a learning rate of 0.01, StepLR (step size = 9, gamma = 0.6), dropout = 2×10^{-3}, weight decay = 8×10^{-3}, and batch size 128.

4.5.3 Results

We benchmark the proposed dual-cell LSTM (Dual-Cell-LSTM-DPA) against TRFI-DNN, STA-DNN, the baseline LSTM-DPA-TA, and the classical DPA-TA on the VTV-SDWW channel model. Evaluations on the other channel models will be presented later in this chapter.

Table 4.3: Hyperparameter search space and selected values for the dual-cell LSTM (150 trials, 100 epochs each; final model trained for 250 epochs).

Hyperparameter	**Search space**	**Best value**
Learning rate	$[10^{-5}, 10^{-1}]$	10^{-2}
Step size (StepLR)	[1, 10]	9
Gamma (StepLR)	[0.1, 1.0]	0.6
Dropout probability	[0.0, 0.8]	2×10^{-3}
Weight decay	$[10^{-6}, 10^{-2}]$	8×10^{-3}



(a) BER comparison.

(b) NMSE comparison.

Figure 4.4: BER and NMSE performance under VTV-SDWW.

Figure 4.4 shows the BER and NMSE for the VTV-SDWW channel model. At low SNR (0-10 dB), the dual-cell LSTM performs similarly to the baseline (LSTM-DPA-TA). Above 10 dB, the dual-cell LSTM performs better, with the performance gap increasing at higher SNR. For NMSE, the dual-cell LSTM outperforms the baseline across all SNR levels. These gains show that averaging two independent LSTM predictions captures temporal correlations more effectively and reduces error propagation across OFDM symbols. However, this improvement comes at the cost of doubled computational complexity due to the second LSTM cell.

On the other hand, STA-DNN and TRFI-DNN perform worse than the two recurrent

methods (dual-cell-LSTM-DPA and the LSTM-DPA-TA). STA-DNN shows better performance at low SNRs (0 to 15 dB), but its performance decreases as the SNR increases. From 25 dB, we can observe the flattening of its BER graph. This is because STA uses fixed averaging coefficients for spectral and temporal averaging. When the channel is dynamic, these coefficients cause a mismatch with the true channel response, thereby decreasing performance. TRFI-DNN performs poorly at low SNR because interpolation between pilots becomes unreliable when noise dominates the pilot estimates. Figure 4.4(b) shows better NMSE performance for the Dual-cell-LSTM-DPA compared to all evaluated estimators, followed by LSTM-DPA-TA.

4.6 Temporal convolutional networks

While RNN-based estimators demonstrate strong performance by leveraging temporal correlations, they are constrained by their sequential nature, which increases their processing time. This is a constraint in vehicular communication, where channel estimation should be completed within the coherence time [68]. We propose a DPA-TCN architecture that addresses these limitations by using TCNs as a post-processing refinement stage after DPA estimation. Unlike the causal TCN introduced in Section 2.6.2.4, our refinement module employs a **non-causal** TCN variant that exploits **frequency-domain** correlations across subcarriers rather than temporal correlations across OFDM symbols. This architectural choice enables parallel computation across all subcarriers within a frame, potentially reducing inference latency.

4.6.1 Non-causal refinement TCN

The proposed TCN operates as a residual refinement module applied after DPA estimation. Unlike recurrent baselines where the NN precedes DPA, this reversed ordering allows DPA to provide initial channel tracking that exploits temporal correlation between OFDM symbols [6], while the TCN subsequently corrects residual errors by modelling

frequency-domain structure.

Our non-causal TCN employs dilated one-dimensional convolutions with symmetric padding that operate along the subcarrier axis, allowing the receptive field at a given subcarrier to exploit information from both lower and higher frequency neighbouring subcarriers within the same frame. This contrasts with causal TCNs, which restricts each output to depend only on "past" inputs in the sequence.

The TCN refines the DPA estimate by learning and adding correction values:

$$\widehat{\mathbf{H}}_n^{\mathrm{DPA-TCN}} = \widehat{\mathbf{H}}_n^{\mathrm{DPA}} + f_\theta\Big(\widehat{\mathbf{H}}_n^{\mathrm{DPA}}\Big), \tag{4.17}$$

where $\widehat{\mathbf{H}}_n^{\mathrm{DPA}}$ is the initial DPA estimate, $f_\theta(\cdot)$ is the TCN that computes the correction, and $\widehat{\mathbf{H}}_n^{\mathrm{DPA-TCN}}$ is the improved estimate.

The network consists of residual blocks with dilated convolutions, weight normalisation, a global skip connection, and a final 1×1 projection layer. Training minimises the MSE between the refined estimate and the true channel:

$$\mathcal{L} = \frac{1}{I|\mathcal{K}_{\mathrm{act}}|} \sum_{n=1}^{I} \sum_{k\in\mathcal{K}_{\mathrm{act}}} \Big| H_n[k] - \widehat{H}_n^{\mathrm{DPA-TCN}}[k] \Big|^2, \tag{4.18}$$

where I is the number of OFDM data symbols in the frame.

4.6.1.1 Data structuring for frequency-domain processing

To enable the TCN to operate along the frequency (subcarrier) dimension, we introduce a specialised data structuring method. Each complex-valued DPA estimate is decomposed into real and imaginary parts, then interleaved along the OFDM symbol dimension. For each subcarrier k, we define a feature vector $\mathbf{z}_k$:

$$\mathbf{z}_k = [\Re(Y_1[k]), \Im(Y_1[k]), \Re(Y_2[k]), \Im(Y_2[k]), \ldots, \Re(Y_I[k]), \Im(Y_I[k])], \tag{4.19}$$

producing a sequence of length $2I$ for each subcarrier, where $I = 50$ is the number of OFDM symbols in the frame. This interleaving embeds both magnitude and phase evolution across OFDM symbols into the feature representation.

The complete input tensor for the TCN is:

$$\mathbf{X}'_{\text{TCN}} \in \mathbb{R}^{N \times C \times L}, \qquad C = 2I = 100, \quad L = K_{\text{act}} = 52, \tag{4.20}$$

where N is the batch size, C is the feature dimension (100 channels from interleaving real/imaginary parts across 50 symbols), and L is the sequence length (52 active subcarriers). Figure 4.5 illustrates this data structuring approach with a kernel size of 2. Our kernel size will be larger to accommodate more adjacent subcarrier information.

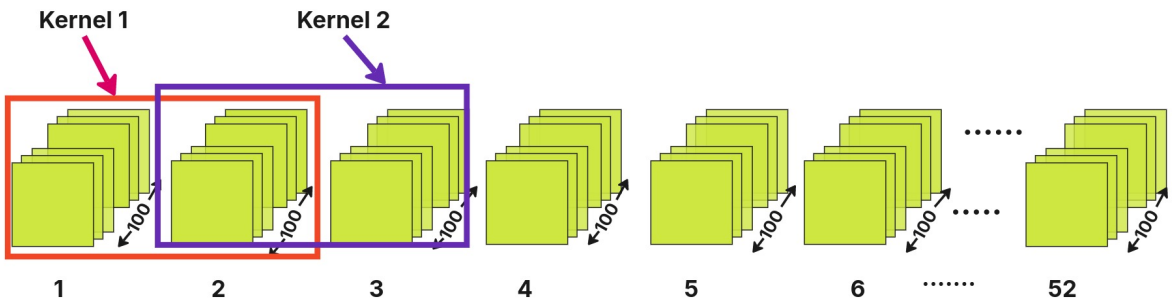


Figure 4.5: TCN data structure: 52 subcarriers treated as sequence dimension, 100 feature channels from interleaved real and imaginary components across 50 OFDM symbols.

4.6.1.2 Processing pipeline

The DPA-TCN processing pipeline consists of:

1. **LS initialization**: $\widehat{H}_{\text{LS}}[k]$ from the preamble provides the initial channel estimate.
2. **DPA channel tracking**: DPA refines the estimate symbol-by-symbol using decision-directed feedback (Section 2.5.3.2), producing $\widehat{\mathbf{H}}_n^{\text{DPA}}$.
3. **TCN refinement**: The non-causal TCN processes the structured DPA estimates to produce the final estimate $\widehat{\mathbf{H}}_n^{\text{DPA-TCN}}$.

4.6.2 Hyperparameter optimisation

We optimise TCN hyperparameters following the protocol in Section 4.2.2. The optimisation differs from previous architectures in two key aspects: (1) we include kernel size as an architecture-specific parameter for convolutional layers, and (2) we exclude weight decay, which we found to be less critical for TCN regularisation. We conduct 80 trials of 200 epochs each. The search space and best values are detailed in Table 4.4. The best

Table 4.4: TCN-DPA hyperparameter search space and best values.

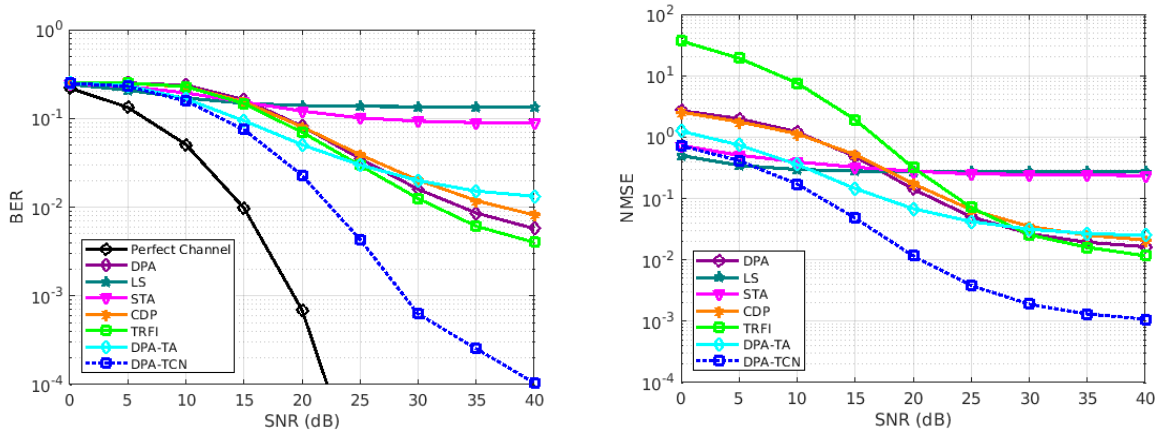
Hyperparameter	**Search space**	**Best value**
Learning rate	$[10^{-5}, 10^{-2}]$	10^{-3}
Number of blocks	[1, 5]	4
Hidden channels	$[64, 100, 128, 256, 300, 512]$	100
Kernel size	[2, 5]	2
Dilation depth	$[2, 3, 4, 5, 6, 7, 8]$	4
Dropout probability	$[10^{-5}, 0.5]$	0.05
StepLR step size	[10, 50]	N/A
StepLR gamma	[0.5, 1.0]	N/A
CosineAnnealingWarmRestarts T_0	[10, 200]	50
CosineAnnealingWarmRestarts T_{mult}	$\{1, 2, 3, 4\}$	2
CosineAnnealingWarmRestarts $\eta_{\min}$	$[10^{-8}, 10^{-4}]$	10^{-6}

configuration uses a dilation depth of 4 layers with small kernels (size 2), indicating that local subcarrier correlations are most informative for channel estimation.

4.6.3 Results

We evaluate the DPA-TCN estimator against classical baselines on the VTV-SDWW channel model at high mobility ($v = 100$ km/h). We also include a variant with temporal

averaging (DPA-TCN-TA) to assess whether additional smoothing provides benefits. Figure 4.6 shows the BER and NMSE results. DPA-TCN performs better than the classical



(a) BER performance.

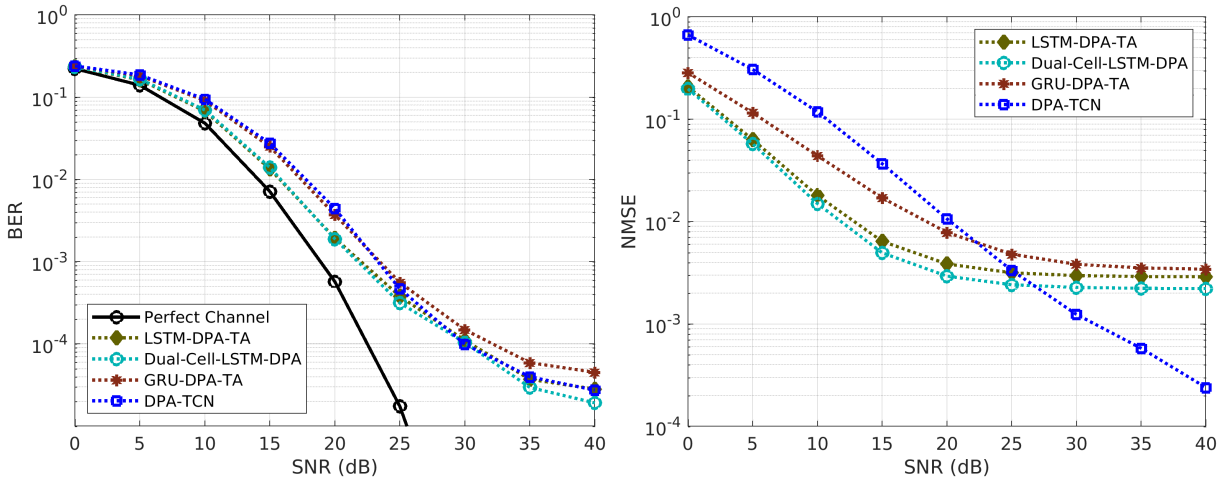
(b) NMSE performance.

Figure 4.6: BER and NMSE performance of DPA-TCN against classical baselines under VTV-SDWW channel model.

estimators, with its performance continuing to improve in high SNR regions (above 15 dB). The NMSE results show a similar trend with the BER performance as the DPA-TCN outperforms all the other evaluated estimators. Comparison with RNN-based estimators (LSTM-DPA-TA, GRU-DPA-TA, dual-cell LSTM) is presented in Section 4.7, where we evaluate all DL estimators using the same channel model to assess their relative performance.

4.7 Comparative analysis

In this section, we evaluate the recurrent estimators (LSTM-DPA-TA, GRU-DPA-TA) and the proposed architectures (Dual-cell LSTM-DPA, and DPA-TCN) on the VTV-EX channel model. All models are trained on a training dataset generated using this channel model and tested using an independent testing dataset also generated with the same channel model. The main aim of this experiment is to evaluate the proposed DL architectures on the same channel model. All models were trained on a dataset generated at 40 dB SNR.



(a) BER performance.

(b) NMSE performance.

Figure 4.7: BER and NMSE performance comparison of the proposed DPA-TCN estimator against classical baselines under the VTV-EX channel model.

As shown in Figure 4.7(a), for many tested SNRs, the proposed Dual-cell-LSTM-DPA and the LSTM-DPA-TA achieve similar BER performance. Divergence of the two models is observed only above 30 dB. TCN-DPA and GRU-DPA-TA show similar performance from 0 to 20 dB, but DPA-TCN performs marginally better from 25 dB. The NMSE results in Figure 4.7(b) show a better performance by Dual-cell-LSTM-DPA, with Dual-cell-LSTM-DPA achieving the best performance of the evaluated estimators.

The observed better NMSE performance of DPA-TCN than the other recurrent models, specifically at high SNRs, but worse than the recurrent evaluated models (Dual-Cell-LSTM-DPA, LSTM-DPA-TA, GRU-DPA-TA) at low SNRs (0 to 25), highlights that the training at high SNR may not have modelled the channel behaviour at low SNR well. This observation motivates investigating how introducing noise during training might improve model generalisation, rather than training exclusively on high-SNR data as commonly done in prior work [6]–[8], [17], [66], [101]. This forms the basis of the next chapter.

4.8 Chapter summary

This chapter evaluated DL architectures for channel estimation in vehicular OFDM systems. Building upon the FNN and RNN introduced earlier, we proposed two new architectures: dual-cell LSTM-DPA and DPA-TCN. The dual-cell LSTM enhances performance by utilising recurrent diversity, making it more effective in dynamic channels, although this improvement comes with increased complexity.

The proposed DPA-TCN architecture leverages the parallelism of temporal convolutions to achieve competitive estimation accuracy while significantly reducing estimation latency. This feature makes the TCN-based approach particularly suitable for low-latency receiver implementations, where the sequential processing of recurrent models, such as LSTM or GRU, may not meet the required inference-time constraints.

Performance analysis revealed a consistent degradation in accuracy at low SNRs, highlighting the sensitivity of DL estimators to training conditions. This finding motivates the next chapter, which will develop a noise-aware training strategy aimed at improving model generalisation under varying noise levels. Thus, the next chapter (Chapter 5) transitions from architectural design to enhancing the generalisation of DL-based channel estimators.

Additionally, all models trained and tested in this chapter used a single-channel model. The exploration of single DL models that can generalise well across different propagation conditions has yet to be conducted, and this will be a focus in the upcoming chapter.

Chapter 5

Robustness through noise-aware training

This chapter investigates the impact of training data composition on model performance and generalisation.

5.1 Introduction

Chapter 4 demonstrated performance improvements through DL architectures. However, a critical question remains: how robust are these models when tested under conditions that differ from training? Traditional approaches train DL channel estimators exclusively on high SNR data (typically 30 or 40 dB), based on the assumption that low noise signals enable better learning of channel characteristics [6]–[8], [66]. The rationale is that, at high SNR, the channel response dominates over noise in the received signal, allowing the network to learn channel transformations without being corrupted by noise. This assumption has not been systematically validated across different architectures and channel conditions.

This chapter investigates how the composition of the training data affects the model's robustness across various vehicular scenarios. We challenge the conventional high SNR training approach by evaluating mixed SNR training, where datasets include samples from multiple SNR levels rather than a single high SNR condition. We further extend this investigation to multi-channel training, where a single model is trained on diverse propagation environments simultaneously. The investigation addresses four key questions: (1) Do models trained exclusively on high SNR data generalise effectively to low SNR conditions? (2) Can mixed SNR training improve robustness without sacrificing high SNR performance? (3) What are the best training strategies for different architectural families and do these strategies generalise across channel conditions? (4) Can a single model trained on multiple channel types generalise to completely unseen propagation scenarios?

We evaluate performance across six vehicular channel models with distinct propagation characteristics: VTV-SDWW, VTV-EX, VTV-UC, RTV-SS, RTV-EX, and RTV-UC.

We first present the high-SNR and mixed-SNR training strategies (Section 5.2) and the experimental setup (Section 5.3). Results and discussion for single-channel training and testing are presented in Sections 5.4 and 5.4.4, respectively. The investigation is then extended to multi-channel training in Section 5.5, followed by a computational and latency analysis in Section 5.6. Section 5.7 summarises the findings. Note that parts of this chapter have been published previously [102], [103].

5.2 Training strategies

Constructing training data for DL channel estimators involves fundamental trade-offs between signal clarity and noise robustness. We investigate two contrasting approaches and their theoretical motivations.

5.2.1 High SNR training

This approach, adopted in previous work such as [6]–[8], [17] to mention a few, assumes that DL models learn optimal channel representations when trained on clean signals where SNR $\geq$ 30 dB. The theoretical justification is that at high SNR, the channel response $H[k, n]$ dominates the received signal $Y[k, n]$, and under these conditions, the network can focus on learning channel transformations without being affected by noise [8]. The assumption is that models trained on clean data can generalise to noisy conditions through inherent robustness.

5.2.2 Mixed SNR training

We propose an alternative approach in which the training data span over a diverse range of SNR. This strategy exposes networks to noise-dominated (SNR $<$ 10 dB) and signal-dominated (SNR $>$ 30 dB) conditions, potentially enabling more robust feature learning. The theoretical motivation comes from data augmentation principles, which hold that introducing variability during training improves model robustness and generalisation [104]. By exposing networks to diverse noise conditions, mixed SNR training acts as a form of augmentation that prevents overfitting to clean signal patterns.

5.2.3 Dataset construction

We construct training and testing datasets for three vehicular channel models (VTV-SDWW, RTV-SS and VTV-EX) outlined in Table 2.3. For each channel model, we construct two training datasets following the simulation procedures and the same frame structure described in Section 3.2. Both datasets use 75% of the frames for training and 25% for validation. Testing uses independent sets to ensure an unbiased evaluation. The same testing sets are used to test all the architectures.

The two dataset types differ in their SNR characteristics and preprocessing approaches:

- The **high SNR dataset** contains 18,000 frames generated at SNR = 40 dB. Data preprocessing follows Section 3.3, stacking real and imaginary parts along the subcarrier dimension as described in Section 3.3.2.
- The **mixed SNR dataset** contains 18,000 frames distributed uniformly across SNR $\in \{0, 5, 10, 15, 20, 25, 30, 35, 40\}$ dB (2,000 frames per SNR level). Data preprocessing follows Section 4.6.1.1, converting complex observations $\mathbf{Y} \in \mathbb{C}^{52\times 50}$ to real-valued representations $\mathbf{X} \in \mathbb{R}^{52\times 100}$ by interleaving the real and imaginary components along the OFDM symbol dimension.

5.3 Experimental setup

This section describes the evaluated architectures and the hyperparameter optimisation procedure used for both training strategies.

5.3.1 Evaluated architectures

LSTM-based architectures have established themselves as the State-of-the-Art (SOTA) for vehicular channel estimation [8], whilst GRU architectures offer a lightweight recurrent alternative [68]. We evaluate whether our proposed Dual-Cell-LSTM-DPA architecture, which extends recurrent processing with two parallel LSTM cells to increase temporal feature diversity, improves upon these established recurrent baselines. To contextualise performance gains and identify which architectural mechanisms contribute to estimation accuracy, we include DPA-TCN as a non-recurrent temporal baseline and two feedforward estimators (STA-DNN, TRFI-DNN). These five architectures provide a representative comparison across convolutional, feedforward, and recurrent DL estimators for vehicular OFDM channel estimation.

The evaluated architectures are:

- **LSTM-DPA-TA** [8] (Section 4.4): SOTA recurrent baseline with DPA post pro-

cessing and TA for noise reduction.

- **GRU-DPA-TA** [68] (Section 4.4): Lightweight recurrent baseline with simplified gating structure.
- **Dual-Cell-LSTM-DPA**: Proposed architecture in Section 4.5 extending recurrent processing with two parallel LSTM cells before DPA refinement.
- **DPA-TCN** estimator introduced in Section 4.6, which refines the DPA output using a non-causal TCN.

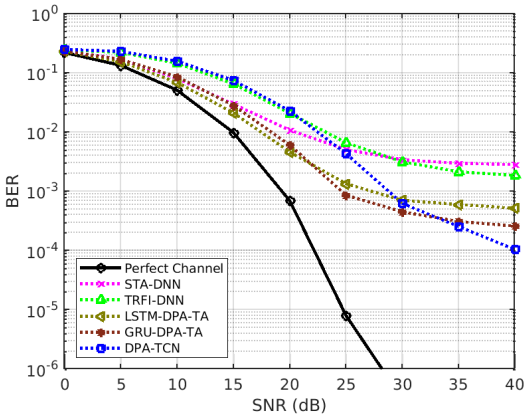
5.3.2 Hyperparameter optimisation

Each architecture undergoes systematic hyperparameter optimisation following the procedure in Section 4.2.2. The only difference is the number of trials used in this optimisation. We conduct 50 trials per model with three random seeds, using validation loss as the optimisation criterion. This optimisation is performed independently for each channel model and training strategy combination. The architectural parameters for LSTM-DPA-TA [8] and GRU-DPA-TA [68] were taken directly from their original papers, as this was shown to be large enough for a similar task. The search spaces and the best configurations for each architecture under both training strategies are outlined in Table A.1.

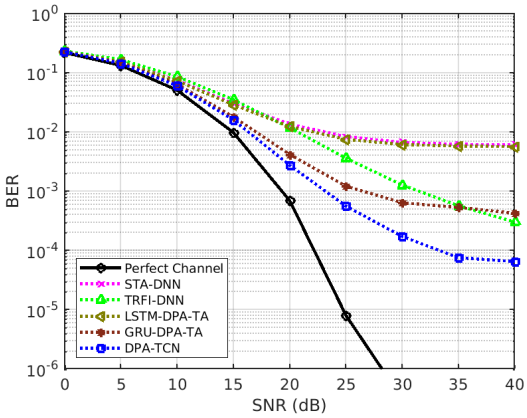
5.4 Results

This section presents comparative performance results across training strategies and channel models, followed by architecture-specific analysis and statistical validation. We use BER of all the estimators considered as the primary metric to analyse the performance of the evaluated DL models. It is important to note that all the results are reported on the testing set, while performance on the validation set is tracked during hyperparameter optimisation. The results reported in this chapter are always for the final hyperparameter-tuned model and reported on the test set.

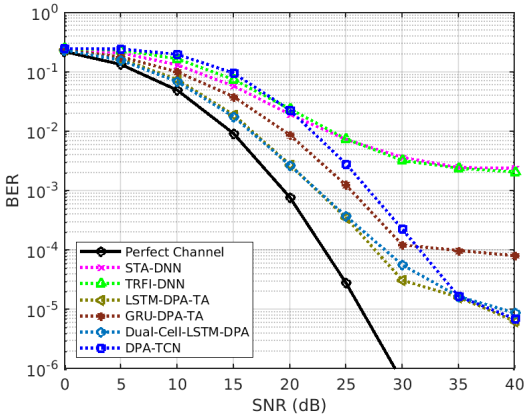
5.4.1 Performance comparison across channel models



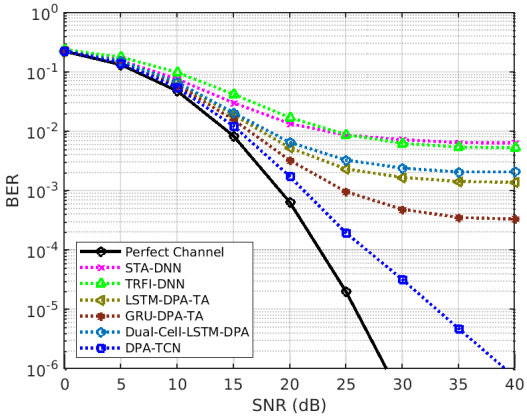
(a) VTV-SDWW: High SNR training.



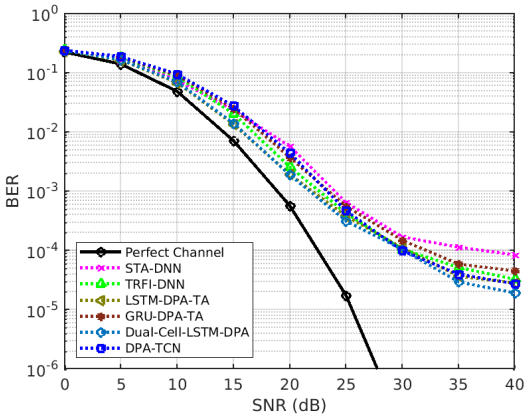
(b) VTV-SDWW: Mixed SNR training.



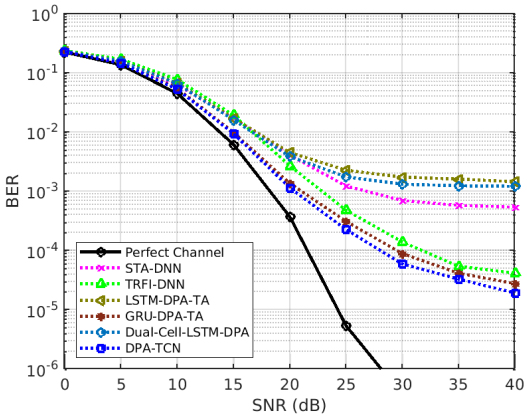
(c) RTV-SS: High SNR training.



(d) RTV-SS: Mixed SNR training.



(e) VTV-EX: High SNR training.



(f) VTV-EX: Mixed SNR training.

Figure 5.1: BER performance comparison across three channel models.

Figure 5.1 shows the BER performance across the evaluated architectures, training strategies, and channel models. The results show striking differences in how architectures respond to the composition of training data, with patterns that remain consistent under different propagation conditions. The ideal curve (black line) shows the optimal performance assuming a perfect channel estimation, serving as the theoretical upper bound. LS estimation provides the classical baseline from Section 2.5.2.

Across all three channel models, several consistent patterns are observed. When models are trained on a mixed-SNR dataset, performance improvement in low-SNR regions across different channel models is observed, with DPA-TCN showing benefits when trained on mixed SNR across all the tested SNR regions. Secondly, recurrent architectures (LSTM-DPA-TA and Dual-Cell-LSTM-DPA) show better performance when trained on a high SNR dataset, although in low SNR regions, their performance improves with a mixed SNR dataset. Third, the degree of performance improvements varies with channel characteristics. RTV-SS shows the most difference between training strategies, indicating that channels with high delay spread benefit the most from noise-aware training. These differences are explored in more detail, architecturally, in Section 5.4.2, and, per SNR analysis, in Section 5.4.3.

5.4.2 Architecture-specific analysis

We analyse the performance characteristics of each architecture under both training strategies across all three channel models, identifying consistent patterns and channel-dependent variations.

5.4.2.1 DPA-TCN

DPA-TCN emerges as the most consistent beneficiary of mixed SNR training across all three channel models. In VTV-SDWW, the architecture achieves substantial BER improvements at low to mid SNR (0–25 dB) while maintaining high SNR performance. This pattern intensifies in RTV-SS, where mixed training produces surprising gains exceeding

10 dB in high-SNR regions. At 40 dB, the BER improves from approximately 10^{-5} with high SNR training to 10^{-6} with mixed training. In VTV-EX, DPA-TCN shows consistent improvements across the SNR range, with greater gains at lower SNR levels. The consistent performance across various channel conditions demonstrates that TCN's dilated causal convolutions effectively learn robust temporal patterns that generalise across both time-varying and frequency-selective fading.

5.4.2.2 GRU-DPA-TA

GRU-DPA-TA shows consistent trends across channel models. At low SNR (0–15 dB), mixed training shows consistent improvements across all three channels, with gains ranging from 0.5 to 1 dB. However, at high SNR (25–40 dB), the architecture shows performance degradation with mixed training, particularly pronounced in RTV-SS and VTV-EX.

This dual behaviour suggests that the GRU architecture benefits from exposure to noisy conditions to a certain extent; however, the gating mechanisms appear to learn suboptimal strategies when trained on mixed data, resulting in reduced performance in high-SNR conditions. The consistency of this pattern indicates that this trade-off is fundamental to the GRU architecture rather than specific to particular propagation conditions.

5.4.2.3 LSTM-DPA-TA and Dual-Cell-LSTM-DPA

Both LSTM-based architectures demonstrate a strong and consistent preference for high-SNR training, especially in high-SNR conditions, across all channel models. At the same time, both architectures show performance improvement in low SNRs when trained on mixed SNRs. This behaviour reflects how LSTM-based models respond to different training strategies. High-SNR training enables the models to learn clear temporal relationships, which aligns with the way LSTM memory cells process sequential information. In contrast, mixed-SNR training introduces noise that interferes with the formation of stable temporal patterns, especially when the model is required to learn temporal and spectral

structure at the same time. As a result, mixed-SNR training offers only limited benefit for LSTM-based architectures, whereas high-SNR training produces more stable and reliable performance.

5.4.2.4 Feedfoward neural networks

The FNN architectures show architecture-dependent responses that vary somewhat with channel characteristics. TRFI-DNN benefits from mixed-SNR training across all three channel models, with improvements most pronounced in RTV-SS, where the reliability-based mechanism of TRFI synergises with learned noise handling. At low to mid SNR (5–20 dB), TRFI-DNN achieves average BER performance improvements compared to high SNR training across all channels.

The reliability metric inherent in TRFI, which identifies trustworthy subcarriers for interpolation, appears to be naturally compatible with noise-aware learning. The NN learns to leverage these reliability indicators more effectively when exposed to diverse noise conditions during training, resulting in robust performance across channel models.

STA-DNN shows more variable behaviour across channels. In VTV-SDWW, it shows a marginal preference for high SNR training. However, in RTV-SS and VTV-EX, mixed training provides slight improvements at very low SNR (0–10 dB) but degradation at high SNR. The spectral-temporal averaging inherent in STA, which assumes relatively stable channel conditions, appears to conflict with noise-aware learning. When trained on mixed-SNR data, the DNN component struggles to reconcile averaged inputs with varying noise characteristics, particularly in frequency-selective environments such as RTV-SS.

5.4.3 Training strategy comparison per SNR

Figures 5.2, 5.3, and 5.4 show the performance differences between the two training strategies for all channel models, highlighting distinct patterns in the advantages specific to each architecture and their consistency across different propagation conditions.

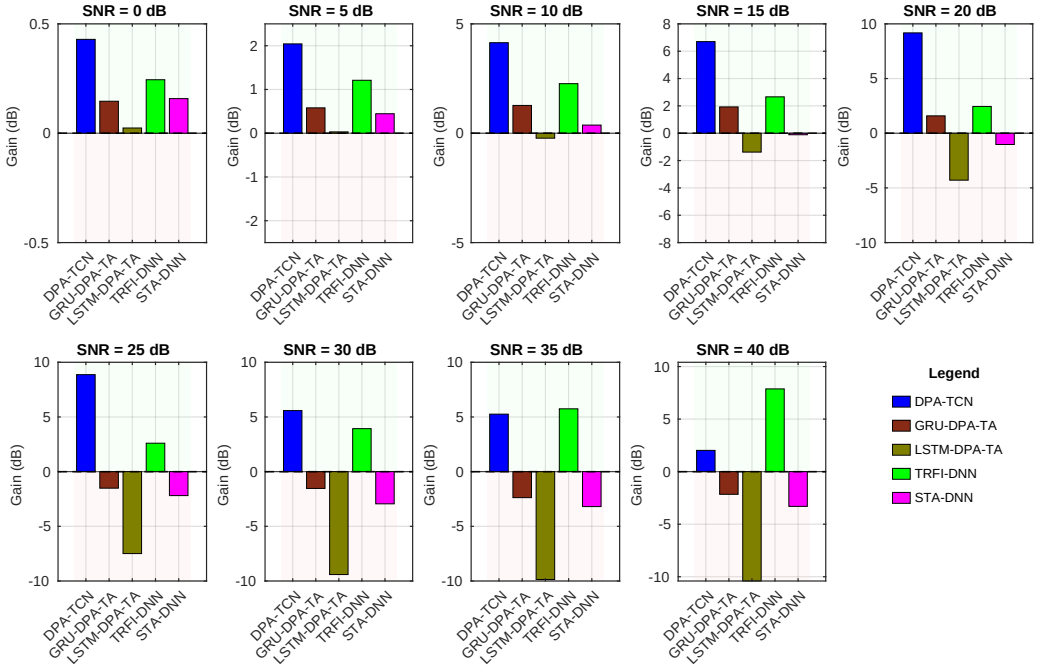


Figure 5.2: Difference in BER between high SNR and mixed-SNR training of the VTV-SDWW channel model, evaluated at different test SNRs. Positive values indicate improvement from mixed training.

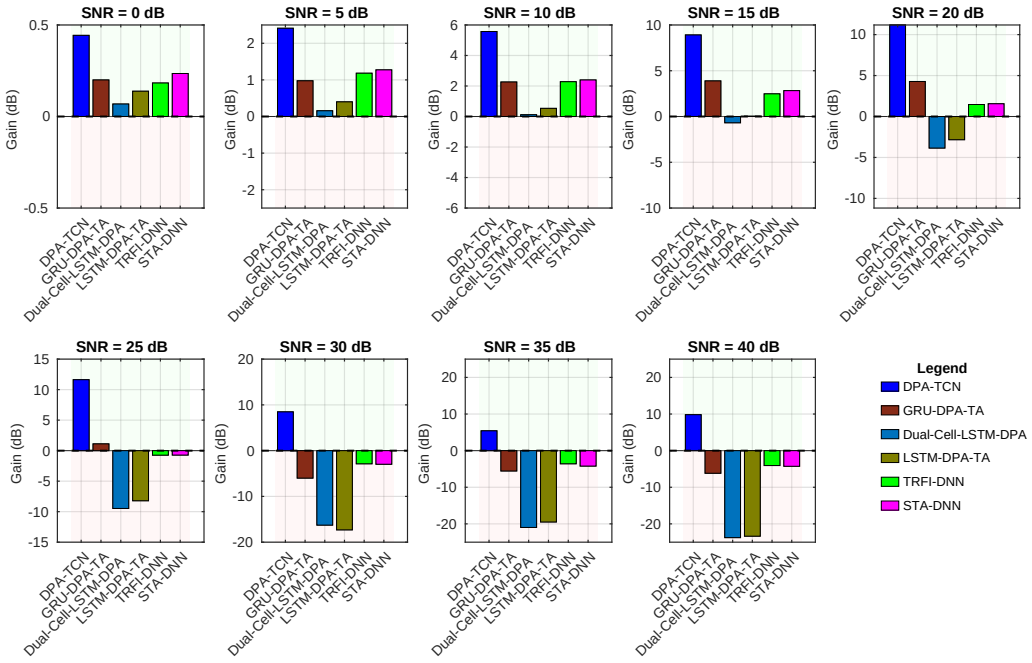


Figure 5.3: Difference in BER between high SNR and mixed-SNR training of the RTV-SS channel model. Positive values indicate improvement from mixed training.

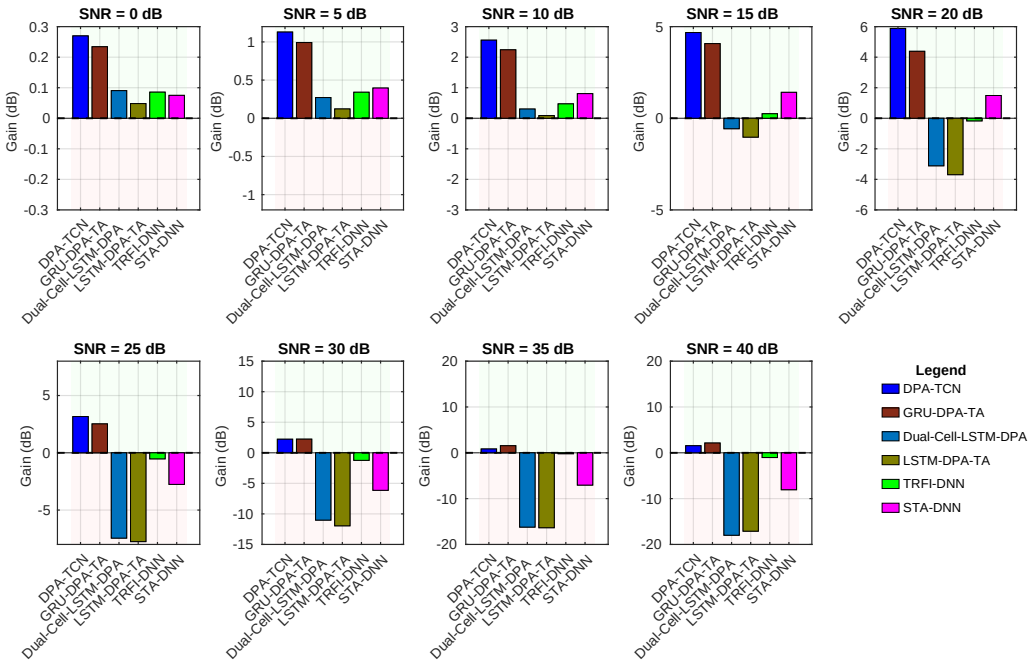


Figure 5.4: Difference in BER between high SNR and mixed-SNR training of the VTV-EX channel model. Positive values indicate improvement from mixed training.

The delta BER analysis also reveals consistent patterns across all three channel models. DPA-TCN shows the most consistent positive delta values across all channels. This pattern is even more pronounced in RTV-SS, where improvements exceed 10 dB at high SNR, demonstrating that frequency-selective channels benefit most dramatically from mixed training.

GRU-DPA-TA shows a consistent biphasic pattern across all channels: positive delta values at low SNR (0–15 dB) with gains of 0.5 to 1 dB, transitioning to negative values at high SNR (25–40 dB) with losses of 2 to 3 dB. The crossover point consistently stays around 20 dB under various propagation conditions, indicating that it is more of an intrinsic architectural characteristic rather than a feature specific to any particular channel.

The LSTM architectures (LSTM-DPA-TA and Dual-Cell-LSTM-DPA) show strongly negative delta values at high SNR across all three channels. This consistent behaviour suggests that these architectures fundamentally require low noise training signals for optimal high-SNR performance.

TRFI-DNN shows positive delta values across all channels at low to mid SNR, where the reliability-based mechanism proves most valuable. STA-DNN shows the most variable pattern, with channel-dependent behaviour ranging from a slight negative preference in VTV-SDWW to mixed performance in RTV-SS and VTV-EX.

The magnitude of delta BER values correlates with channel severity, with RTV-SS showing the largest absolute differences between training strategies. This suggests that channels with high delay spread and frequency-selective fading benefit most from noise-aware training for convolutional architectures, while simultaneously being most detrimental to recurrent architectures when noise is introduced during training.

5.4.4 Discussion

The results across three channel models reveal insights into the relationships among training strategies, architectural characteristics, and propagation conditions. The consistency of patterns across the three channel models VTV-SDWW, RTV-SS, and VTV-EX demonstrates that the findings are not artefacts of a single channel model, but reflect architectural properties that generalise across vehicular communication scenarios.

The better performance observed of mixed SNR training in low SNR regions for all architectures suggests that by exposing the model to training data of a larger domain of SNR values, the networks are better able to interpolate and generalise performance across the varying dynamic range of vehicular channel conditions. The performance improvement achieved by this training strategy differs across architectures. This indicates that while access to a wider SNR domain is the underlying benefit, the specific network structure determines the final result. For instance, the greater robustness of the convolutional DPA-TCN suggests that its inherent structure is better suited to leveraging this broader SNR domain compared to the recurrent architectures.

For vehicular communication systems, where channel conditions vary, maintaining consistent performance across all conditions is essential. Mixed-SNR training provides this consistency with the DPA-TCN, making it preferable for V2V applications. On the other

hand, infrastructure-assisted V2I scenarios with more controlled conditions (less noisy) may benefit from the performance of recurrent architectures trained with a high-SNR strategy.

5.5 Multi-channel training and generalisation

The previous sections showed that the proposed DPA-TCN and the baseline LSTM-DPA-TA exhibit varying performance across channel models, reflecting the unique propagation characteristics of each environment. As summarised in Table 2.3, these channel models differ in delay profiles, Doppler behaviour, and scattering conditions, and vehicles are expected to encounter such variations in real-world scenarios. This raises an important question for better channel estimation: Can one model be trained to perform reliably across multiple propagation scenarios, rather than developing different models for each channel condition?

To explore this, we extend the noise-aware training strategy to multi-channel training. By training on a diverse set of channel conditions, we aim to assess whether channel diversity improves model generalisation to both seen and unseen propagation environments. We investigate this idea by training the DPA-TCN and LSTM-DPA-TA models on a unified dataset combining multiple channel models. We then evaluate the performance of these models on both In-Distribution (ID) channel models, which were present during training, and Out-of-distribution (OOD) channel models, which the models have never encountered. This evaluation assesses both interpolation capability (performance on seen channels) and the extrapolation capability (generalisation to unseen channels).

5.5.1 Multi-channel dataset construction

We construct two unified datasets: one generated at high SNR and one at mixed SNR, aligning with the most effective training strategies identified for LSTM-DPA-TA and DPA-TCN. Each dataset contains 54,000 frames generated from three vehicular channel

models (VTV-SDWW, RTV-SS, and VTV-EX), 18,000 frames per channel. In the mixed-SNR dataset, the 18,000 frames per channel are generated evenly across SNR levels in $0, 5, 10, 15, 20, 25, 30, 35, 40$ dB, resulting in 2,000 frames per SNR per channel, consistent with the mixed-SNR strategy in Section 5.2. Preprocessing follows the interleaving procedure introduced in Section 4.6.1.1. Conversely, in the high SNR dataset, the 18,000 frames per channel are generated at 40 dB, and preprocessing follows the stacking along subcarrier procedure described in Section 5.2.3.

For evaluation, we define two test scenarios:

In-Distribution (ID) channels: frames drawn from the same three channel models used during training (VTV-SDWW, RTV-SS, and VTV-EX), but held out from the training set. These evaluate performance on channel conditions the model has seen during training.

Out-of-distribution (OOD) channels: frames drawn from three channel models that were not used during training (VTV-UC, RTV-UC, and RTV-EX). These evaluate the model's ability to generalise to previously unseen propagation environments.

5.5.2 Experimental configuration

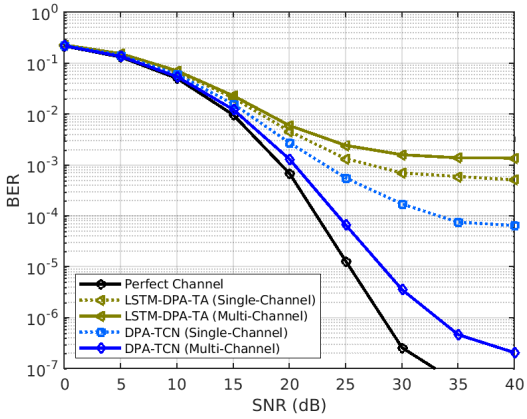
The TCN and the LSTM architectures were optimised for the multi-channel training scenario following the Optuna hyperparameter tuning procedure described in Section 4.2.2. The same procedure is repeated, except that the number of trials used to search for the best parameters is increased. Each trial trained a complete model instance with early stopping based on the validation loss. The search space encompassed architectural, regularisation, and optimisation parameters. A total of 169 trials were executed during the optimisation process. The full search space and the optimal hyperparameters of the TCN are reported in Table 5.1 and for the LSTM are reported in Table 5.2.

Table 5.1: Hyperparameter search space and optimal configuration of the TCN on the multi-channel mixed-SNR dataset.

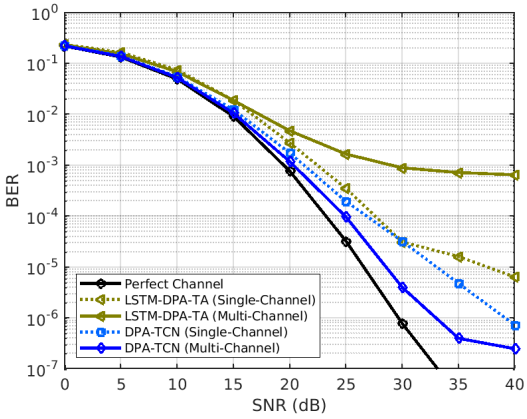
Hyperparameter	**Search Space**	**Optimal Value**
Learning rate	log-uniform: $[10^{-5}, 10^{-2}]$	4.77×10^{-4}
Number of TCN blocks	integer: [1, 8]	5
Kernel size	integer: [2, 5]	4
Dropout rate	uniform: [0, 0.5]	0.228
Step size (scheduler)	integer: [5, 50]	11
Gamma (scheduler decay)	uniform: [0.5, 0.99]	0.948
Dilation depth	integer: [2, 8]	5
Hidden channels	categorical: {64, 100, 128, 256, 512}	256
Batch size	fixed	128

Table 5.2: Hyperparameter search space and optimal parameters for the LSTM model on the multi-channel dataset.

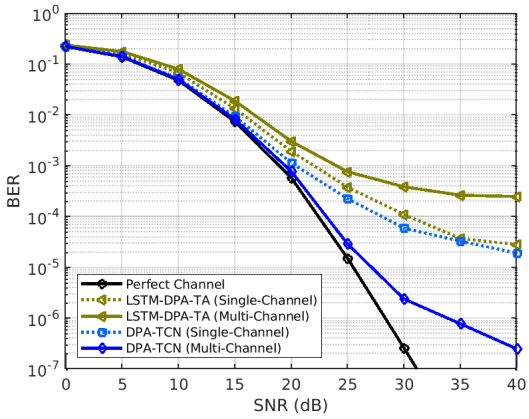
Hyperparameter	**Search Space**	**Optimal Value**
Learning rate	log-uniform: $[10^{-5}, 10^{-1}]$	4.4012×10^{-4}
Dropout	log-uniform: $[10^{-4}, 0.5]$	0.00404
Weight decay	log-uniform: $[10^{-6}, 10^{-3}]$	1.238×10^{-6}
Step size	integer: [5, 30]	27
Gamma	uniform: [0.5, 0.95]	0.8511
Gradient clipping norm	log-uniform: $[10^{-5}, 10^{-2}]$	1.784×10^{-3}
Batch size	fixed	128



(a) VTV-SDWW



(b) RTV-SS

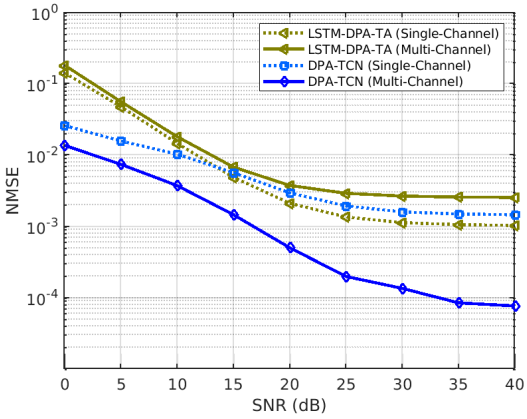


(c) VTV-EX

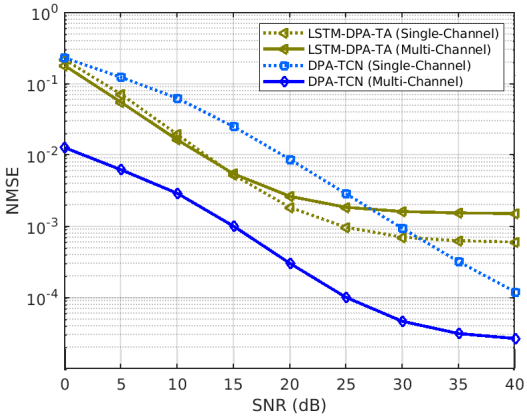
Figure 5.5: BER performance on in-distribution channels.

5.5.3 Results on in-distribution channels

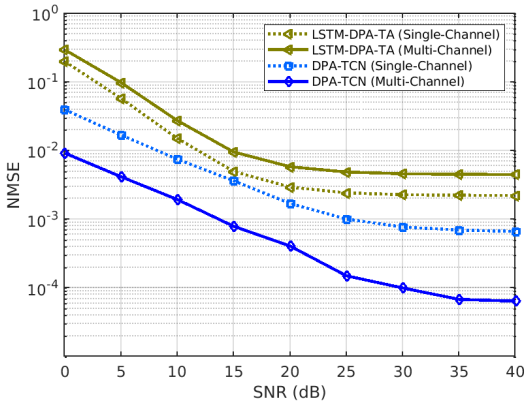
Figure 5.5 presents the BER performance of the DPA-TCN and LSTM-DPA-TA trained on either a single channel model dataset or a multiple channel models dataset (the three ID channel models). As mentioned before, the mixed SNR training is used for the TCN-DPA estimator as it has been demonstrated to perform well with this training strategy. The high SNR training is used for the LSTM-DPA-TA estimator, as it has also been shown to perform well with this strategy. The DPA-TCN trained on a multi-channel dataset is represented by a blue solid line, and the DPA-TCN trained on a single channel is represented by the dotted blue line. The multi-channel LSTM-DPA-TA is represented



(a) VTV-SDWW



(b) RTV-SS



(c) VTV-EX

Figure 5.6: NMSE performance on in-distribution channels.

by the solid olive line and the single channel LSTM-DPA-TA is represented by the dotted olive line.

The DPA-TCN trained on a multi-channel dataset achieves the best performance across all the ID channel models. This performance indicates that the TCN benefits from multi-channel training, with greater advantage observed in the RTV-SS channel model. These performance gains show the model's ability to learn features that work effectively across diverse channels. The NMSE curves (Figure 5.6) confirm that the multi-channel model achieves accurate channel estimates across all three environments.

LSTM-DPA-TA exhibits performance degradation under multi-channel training relative to single-channel training. Across all three ID channels, the multi-channel trained LSTM

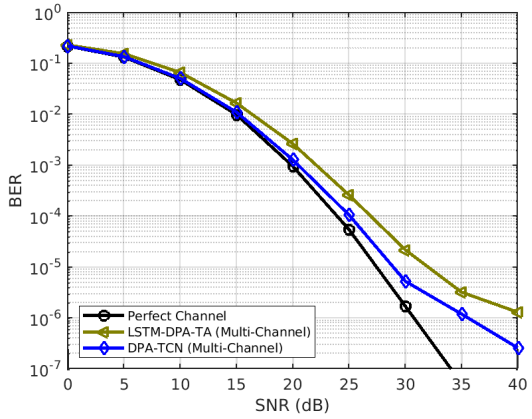
shows performance losses, particularly at high SNR, where the recurrent architecture previously excelled with single high SNR training. This degradation extends the findings of Section 5.4.2.3: LSTM architectures not only struggle with SNR diversity during training but also suffer when exposed to channel diversity.

5.5.4 Results on out-of-distribution channels

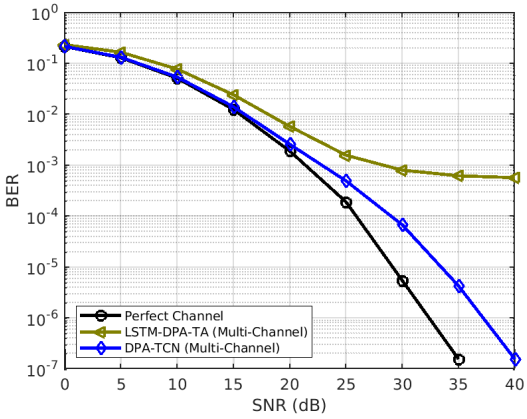
We now evaluate how well the multi-channel trained estimators generalise to unseen channel models by testing them on the OOD channels (VTV-UC, RTV-UC, and RTV-EX), which were not encountered during training. Figure 5.7 presents the BER performance across these three channels, and Figure 5.8 presents the corresponding NMSE performance.

The multi-channel trained DPA-TCN demonstrates superior generalisation to OOD channels across all conditions. Despite never encountering these propagation scenarios during training, the model effectively interpolates between the Doppler shifts and delay-spread characteristics learned from the training channels to handle the new conditions. Both BER and NMSE curves confirm this, with DPA-TCN achieving the lowest NMSE across the entire SNR range.

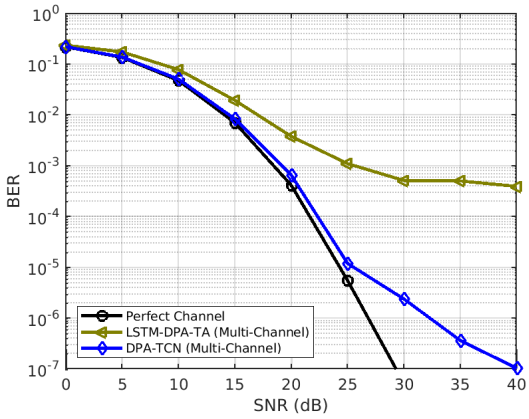
In contrast, the multi-channel LSTM-DPA-TA shows generally weak OOD generalisation, with one notable exception: it performs well across all SNRs on the VTV-UC channel. This can be explained by the characteristics of VTV-UC itself, which is a low-mobility, congested urban environment with vehicle speeds around 48 km/h, corresponding to a Doppler shift of approximately 250 Hz. The resulting temporal dynamics are mild, and LSTM-DPA-TA appears to handle this lower-dynamics regime well, whereas the more aggressive Doppler conditions in RTV-UC and RTV-EX exceed what it has learnt to generalise over.



(a) VTV-UC



(b) RTV-UC

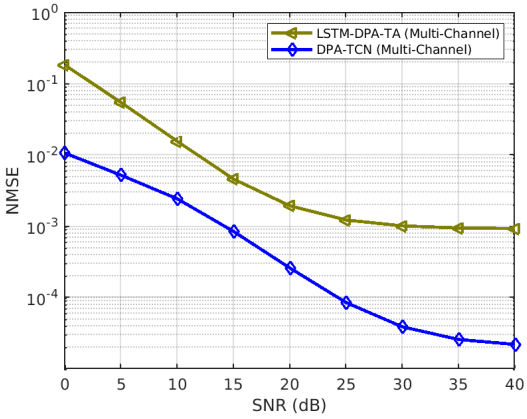


(c) RTV-EX

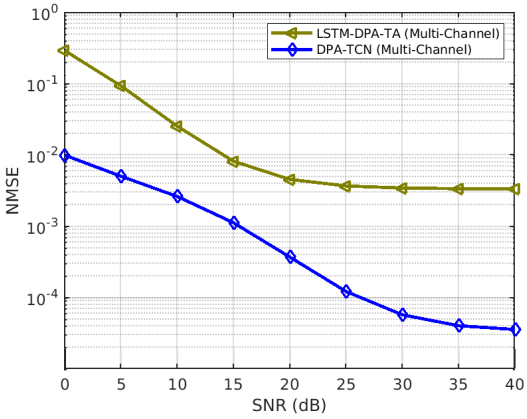
Figure 5.7: BER performance on out-of-distribution channels.

5.6 Computational and latency analysis

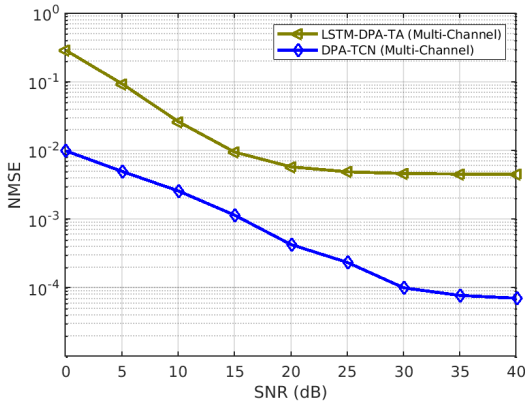
This section evaluates the computational cost and inference latency of the proposed DL estimators compared to the recurrent baselines (LSTM and GRU). In line with the metrics introduced in Section 2.8.3, we combine analytical complexity measures (parameter count and floating-point operations) with empirical GPU latency measurements from the implemented PyTorch models.



(a) VTV-UC



(b) RTV-UC



(c) RTV-EX

Figure 5.8: NMSE performance on out-of-distribution channels.

5.6.1 Measurement methodology

Evaluation environment: All latency measurements were performed on a single NVIDIA A40 GPU (48 GB GDDR6, driver 580.126.09) running Ubuntu 22.04.5 LTS. Models were implemented in PyTorch 2.5.1 (CUDA 11.8 build) with Python 3.11, using FP32 precision and a batch size of 1 (single-frame inference) to reflect realistic real-time deployment conditions. Prior to measurement, a warm-up phase of 50 forward passes was performed to stabilise CUDA kernel caches and clock frequencies. All measurements use `torch.cuda.synchronize()` before and after each inference call to ensure accurate GPU wall-clock timing. Each model was evaluated over 2,000 independent single-frame inference runs, from which the median (p50) and 90th-percentile (p90) latencies were

computed.

For each estimator, we compute the following:

- **Number of trainable parameters**, which determines the model memory footprint. This was computed by summing the number of elements across all tensors that require gradient computation (weights and biases) using PyTorch's internal accounting functionality.
- **FLOPs/frame**. We define this metric as the total floating-point operations required to process one OFDM frame. The value is computed as twice the total number of Multiply-Accumulate Operations (MACCs) executed by the network's layers during a forward pass, since each multiply-accumulate comprises one multiplication and one addition. The MACC count is determined from the layer definitions.
- **Inference latency**, measured using synchronised CUDA wall-clock timing. For each model, we process individual OFDM frames sequentially with batch size 1 and record the distribution of inference durations. We report the **median (p50) latency** and **90th-percentile (p90) latency**, where the p90 latency is defined as the latency value below which 90% of all inference measurements fall.

5.6.2 Computational complexity

Table 5.3 summarises the measured computational complexity of the evaluated estimators. Recurrent models (GRU and LSTM variants) are lightweight in terms of parameters and FLOPs, while the convolutional TCN architecture has a larger footprint due to its wider layers and dilated convolutions.

5.6.3 Inference latency

Table 5.4 presents the measured single-frame latency distributions. The recurrent estimators (GRU-DPA-TA, LSTM-DNN-DPA, LSTM-DPA-TA, and Dual-cell LSTM-DPA)

Table 5.3: Computational complexity of DL channel estimators.

Model	Number of Parameters	FLOPs/Frame
GRU-DPA-TA	26,880	2.74M
LSTM-DPA-TA	132,192	13.41M
Dual-cell LSTM-DPA	264,384	26.83M
DPA-TCN	6.48M	673M

require single-frame inference times between 68 and 135 ms, since their gate updates must execute sequentially across the 50 OFDM symbols of each frame. This sequential latency exceeds the receiver processing budget required for continuous frame reception in IEEE 802.11p.

Table 5.4: Single-frame inference latency. p90 = 90th percentile latency.

Model	Median Latency (ms)	p90 Latency (ms)
GRU-DPA-TA	68.4	69.0
LSTM-DNN-DPA	76.3	76.4
LSTM-DPA-TA	135	137
Dual-cell LSTM-DPA	111	123
DPA-TCN	**4.16**	**4.42**

The proposed DPA-TCN, by contrast, processes all 50 symbols in parallel through dilated convolutions, achieving a median single-frame latency of 4.16 ms and a p90 of 4.42 ms, a 16- to 32-fold reduction relative to the recurrent baselines. This per-frame latency fits within the receiver processing budget for IEEE 802.11p continuous frame reception.

5.7 Chapter summary

This chapter examined how training-data composition, spanning SNR diversity and channel diversity, affects the robustness and generalisation of DL channel estimators. We summarise the main findings below.

1. Mixed-SNR training improves the performance of DL estimators in low SNR conditions, with DPA-TCN benefiting more from this strategy across all the evaluated SNR regions. This trend was consistent, see Figure 5.1 and also the BER difference analysis presented in Figure 5.2 - Figure 5.4.

2. Recurrent architectures (LSTM-DPA-TA, GRU-DPA-TA, Dual-Cell-LSTM-DPA) benefit from mixed-SNR training in low SNRs but overall perform best when trained on high-SNR data. This low SNR (0 to 20) performance improvement behaviour was observed consistently across all evaluated channel models, see the delta BER analysis presented in Figure 5.2 to Figure 5.4.

3. FNN architectures show architecture-dependent behaviour: TRFI-DNN improves with mixed-SNR training, whereas STA-DNN performs best with high-SNR training.

4. Mixed-channel and mixed-SNR training (Section 5.5) resulted in a single DPA-TCN model that generalises effectively to both ID and OOD channel models, removing the need for multiple channel-specific estimators. See Figure 5.5 to Figure 5.8.

5. LSTM-DPA-TA trained on the multi-channel dataset shows poor generalisation to unseen channels.

6. Training-data composition (SNR span and channel diversity) is therefore a key architectural design variable. Models trained without sufficient diversity show degraded performance under mismatched conditions.

7. The computational analysis demonstrated that the DPA-TCN architecture meets the receiver processing budget for IEEE 802.11p continuous frame reception due to its parallel convolutional structure, whereas recurrent architectures have inference latencies one to two orders of magnitude higher (Section 5.6).

In conclusion, the DPA-TCN estimator, when trained on a mixed-channel and mixed-SNR dataset, delivers strong generalisation to both known and unseen vehicular environments, while meeting the receiver processing requirements of IEEE 802.11p V2X operation.

Chapter 6

Interpretability and complexity-aware design

This chapter introduces the REACH framework, an extension of LRP for interpretability-driven optimisation of channel estimation networks. Two case studies apply REACH to feedforward and TCN estimators, identifying redundant parameters and channel-invariant features that explain the strong generalisation reported in Chapter 5.

6.1 Introduction

Previous chapters established that DL achieves better channel estimation performance when compared to classical channel estimators through careful architectural design, mixed-SNR training strategies, and multi-channel exposure during training. The multi-channel trained TCN-DPA model demonstrated strong generalisation across both ID and OOD channel models. What remains unclear is *why* these models perform well and *which* input features drive their predictions.

Section 2.7 introduced LRP as an approach to feature attribution through conservative backward propagation of relevance Equations (2.86) and (2.87). Whilst existing frameworks such as GRACE [86] apply LRP to analyse channel estimators post-hoc, they remain diagnostic tools that explain which features influence predictions without extending to architectural optimisation. This chapter bridges that gap by demonstrating that interpretability can serve dual purposes: understanding model behaviour and prescribing efficient architectures.

We apply LRP to two distinct case studies that address complementary aspects of interpretability based optimisation. The case studies use different network architectures, not to compare performance, but to develop and validate an interpretability methodology under varying levels of complexity.

Case Study I develops the interpretability methodology using a simple architecture. We apply LRP to a three-layer FNN estimator (TRFI-DNN) trained on a single-channel model (VTV-SDWW). The reduced architectural complexity of FNN with its direct input-to-output mappings and the absence of temporal dependencies provides an ideal testbed to establish good relevance analysis procedures. This case study develops the REACH framework, which extends LRP through temporal aggregation, batch processing, and percentile-based feature categorisation. Using this controlled setting, we investigate whether relevance analysis can identify redundant features and network parameters, thereby transforming interpretability from a diagnostic tool into a practical architectural optimisation strategy.

Case Study II applies the developed methodology with the best-performing architecture from previous chapters. We analyse the multi-channel trained TCN-DPA model from Chapter 5 across all six vehicular channels: VTV-SDWW, RTV-SS, and VTV-EX (ID), and VTV-UC, RTV-UC, and RTV-EX (OOD). This case study addresses a fundamentally different question: which input features enable robust generalisation across diverse propagation environments? The TCN architecture with its dilated convolutions and long-range dependencies presents greater interpretability challenges than the feedforward baseline. By applying REACH to this complex architecture, we investigate whether

the interpretability insights from Case Study I generalise to SOTA models and whether cross-channel relevance analysis can reveal features that remain important regardless of propagation characteristics.

Section 6.2 introduces the REACH framework, our proposed extension of LRP that provides systematic aggregation and thresholding procedures specifically designed for channel estimation networks. This framework transforms the LRP foundations from Section 2.7 into a practical methodology for feature selection and architectural optimisation. Section 6.3 presents Case Study I, which applies REACH through application to FNNs for single-channel optimisation, demonstrating feature selection and node pruning capabilities. Section 6.4 presents Case Study II, applying the REACH framework to investigate cross-channel interpretability in the multi-channel trained TCN, identifying cross-channel important features and validating their functional significance through ablation experiments. Section 6.5 summarises the key contributions and their implications for practical deployment.

6.2 REACH methodology

REACH adapts the LRP principles discussed in Section 2.7 to optimise channel estimation networks. The methodology applies the conservation principle of Equation (2.86) and the stabilised propagation rule of Equation (2.87) to decompose network predictions into subcarrier-level contributions. Unlike GRACE [86], which analyses individual frames, REACH aggregates relevance across temporal and batch dimensions to capture channel dynamics in high-mobility scenarios. This spatio-temporal aggregation improves the robustness of the feature relevance maps: by averaging out high-variance relevance contributions from transient noise and instantaneous channel fades, the methodology provides stable and generalisable feature importance estimates that accurately reflect the network's processing strategy across diverse vehicular propagation conditions.

REACH aims to improve confidence in neural predictions by identifying which time-frequency features drive estimation accuracy. It also aims to systematically reduce com-

putational complexity by pruning inputs and architectural components with minimal relevance, thus maintaining performance whilst reducing resource requirements for vehicular deployment.

6.2.1 REACH process flow

The REACH framework operates as a five-stage pipeline that takes a trained channel estimation network as input and produces both feature relevance scores and a compressed model variant as output. Figure 6.1 summarises the process.

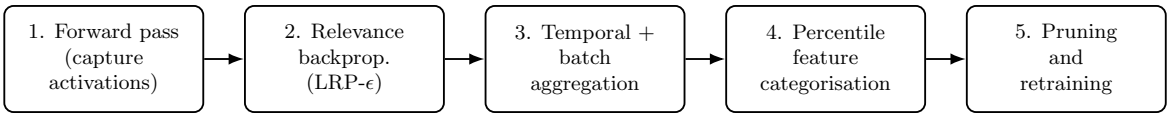


Figure 6.1: The REACH framework as a five-stage process: from a trained network to a relevance-guided compact model.

1. **Forward pass with intermediate state capture.** A batch of OFDM frames is passed through the trained network, with intermediate activations and pre-activation values retained at each layer for use in the backward relevance pass.

2. **Architecture-specific relevance backpropagation.** Output relevance is propagated backward through the network using the LRP-ϵ rule, with adaptations for the layer types present: stabilised propagation for fully connected layers, dilated-convolution-aware propagation for TCN blocks, magnitude-weighted splits at residual connections, and effective-weight handling for weight-normalised layers.

3. **Temporal and batch aggregation.** Per-frame relevance maps are averaged across the temporal dimension ($T = 1$ for symbol-level FNNs, $T = 50$ for frame-level TCNs) and across a batch of B frames ($B = 128$ for FNN, $B = 2{,}000$ for TCN) to produce stable feature importance estimates that are robust to instantaneous channel variations.

4. **Percentile-based feature categorisation.** The aggregated relevance distribution is thresholded using percentile cut-offs (P_{25}, P_{50}, P_{90}) to assign each input feature to one of four categories: Critical, Significant, Moderate, or Marginal.

5. **Architectural pruning.** Features and hidden neurons with relevance below a configurable threshold are removed from the network. The resulting compact architecture is retrained from scratch, and its performance is benchmarked against the original baseline.

The remainder of this section details each stage. Sections 6.2.2 and 6.2.3 cover the architecture-specific relevance backpropagation rules required in stage 2, addressing numerical stability through the ϵ-rule and conservation through normalisation for FNNs, and the handling of dilations, residual connections, and weight normalisation for TCNs. Section 6.2.4 formalises the aggregation in stage 3, Section 6.2.5 details the percentile categorisation in stage 4, and Section 6.2.6 describes the pruning operation in stage 5.

6.2.2 Feedforward network implementation

For feedforward architectures, the forward pass computes activations whilst preserving intermediate outputs needed for relevance distribution. This process, defined by the standard layer-wise transformation (Equation ((2.61)) in Section 2.6.2.1), requires that the pre-activation values are stored. The relevance is then propagated backward through the network using the rule of Equation ((2.87)), distributing the final channel estimate back to the input features.

Deep fading in vehicular channels can produce extremely small activation values, causing numerical instability in the relevance computation. To address this, we implement the ϵ stabilisation term (as introduced in Equation ((2.87)) in Section 2.6.2.1). This stabilisation ensures that the denominators, which correspond to the pre-activation terms (denoted as $\mathbf{z}$ in standard LRP notation), remain bounded away from zero even when activations approach zero. The stabilised relevance computation utilises the ϵ-stabilised pre-activation term $\mathbf{z}_{\text{stable}}$ to compute the relevance factor q_{mn}.

Whilst LRP theory guarantees conservation, floating-point numerical errors can introduce small deviations. We apply a normalisation correction whenever the conservation property is violated beyond a defined error tolerance to strictly maintain the relevance budget:

$$R_m^{(l)} \leftarrow R_m^{(l)} \cdot \frac{\sum_n R_n^{(l+1)}}{\sum_m R_m^{(l)} + \epsilon} \tag{6.1}$$

This scaling maintains the conservation property, ensuring that all relevance from the final estimate is traced back to input features without loss or artificial inflation.

6.2.3 Temporal convolutional network implementation

The TCN architecture introduces additional implementation challenges: temporal convolutions with varying dilation rates, residual skip connections that create multiple paths for relevance flow, and weight normalisation that decouples magnitude from direction.

For 1D convolutional layers with kernel size k, dilation d, and stride s, we adapt Equation (2.87) to account for the receptive field structure:

$$R_i^{(l-1)} = \sum_j \sum_{m=0}^{k-1} \frac{a_{i+md}^{(l-1)} w_{jm}}{\sum_{i'} \sum_{m'} a_{i'+m'd}^{(l-1)} w_{jm'} + \epsilon} R_j^{(l)} \tag{6.2}$$

where m indexes kernel positions. This formulation distributes relevance from output position j back to all input positions within its receptive field, weighted by their contribution to the preactivation.

Residual skip connections create multiple parallel paths from input to output. At each residual block, activations combine through element-wise addition. We employ magnitude-based redistribution to distribute relevance across both paths:

$$R_{\text{residual}} = \frac{|a_{\text{residual}}|}{|a_{\text{residual}}| + |a_{\text{main}}| + \epsilon} R_{\text{out}}, \quad R_{\text{main}} = \frac{|a_{\text{main}}|}{|a_{\text{residual}}| + |a_{\text{main}}| + \epsilon} R_{\text{out}} \tag{6.3}$$

where a_{residual} and a_{main} are the activations from the skip connection and main path, respectively. This rule attributes relevance in proportion to the magnitude of each path's contribution to the combined output.

For weight normalisation, which reparameterises layer weights as $\mathbf{w} = \frac{g}{\|\mathbf{v}\|}\mathbf{v}$, we use the effective weights in the LRP propagation:

$$w_{\text{eff}} = \frac{g}{\|\mathbf{v}\|}\mathbf{v} \tag{6.4}$$

6.2.4 Temporal and batch aggregation

REACH's key departure from single-frame analysis appears in its aggregation strategy. We accumulate relevance scores across T consecutive OFDM frames to capture temporal channel dynamics:

$$R_{\text{temporal}}(k) = \frac{1}{T}\sum_{t=1}^{T}|R_t(k)| \tag{6.5}$$

where $R_t(k)$ represents the relevance score for subcarrier k at time step t. The temporal window T varies by architecture:

- For single-symbol estimators (FNN), $T = 1$, as the model processes inputs instantaneously symbol-by-symbol.
- For sequence-based estimators (TCN), $T = 50$, corresponding to the number of OFDM symbols in a standard IEEE 802.11p frame.

This temporal averaging reduces sensitivity to instantaneous channel variations that might spuriously elevate or suppress individual feature scores.

These temporal scores are further averaged across batch size:

$$R_{\text{final}}(k) = \frac{1}{B}\sum_{b=1}^{B}R_{\text{temporal},b}(k) \tag{6.6}$$

where $B = 128$ in our feedforward network implementation and 2,000 in our TCN implementation. This batch-wise aggregation ensures statistical stability across the range of channel conditions in the training distribution, producing feature importance measures that reflect consistent patterns rather than outliers.

6.2.5 Percentile-based feature categorisation

This step is essential for converting continuous relevance scores into small, high-impact feature sets for subsequent analysis and model optimisation. By identifying features with low or marginal contributions, we can isolate the most impactful input subcarriers to build compact, efficient models that disregard noisy or confusing inputs. Feature categorisation employs distribution statistics rather than fixed thresholds. For a relevance distribution with values $\{R_1, R_2, \ldots, R_N\}$, we compute percentiles P_{25}, P_{50}, and P_{90} corresponding to the 25th, 50th, and 90th percentiles. Features are grouped into four categories based on these thresholds:

$$\text{Critical:} \quad R_i \geq P_{90} \tag{6.7}$$

$$\text{Significant:} \quad P_{50} \leq R_i < P_{90} \tag{6.8}$$

$$\text{Moderate:} \quad P_{25} \leq R_i < P_{50} \tag{6.9}$$

$$\text{Marginal:} \quad R_i < P_{25} \tag{6.10}$$

This percentile-based approach adapts to the natural distribution of relevances, ensuring that thresholds are dynamically set to capture the small number of highly important features and the long tail of low-importance features. Combined categories group features for specific analyses: the Essential category merges Critical and Significant features (top 50%), whilst the Core category includes all except Marginal features (top 75%).

6.2.6 Architectural pruning

The final relevance scores enable threshold-based feature selection and node pruning. For input features, we identify low-contribution subcarriers through:

$$\mathcal{P}_{\text{features}} = \{k : R_{\text{final}}(k) < \tau\} \tag{6.11}$$

where τ represents a predetermined threshold. Subcarriers in $\mathcal{P}_{\text{features}}$ can be excluded from the model input, reducing both preprocessing overhead and the dimensionality of the first network layer.

For hidden neurons, layer-wise pruning identifies nodes contributing minimally to network computation:

$$\mathcal{P}_{\text{layer}}^{(l)} = \left\{ j : \frac{1}{N} \sum_{n=1}^{N} |R_j^{(l,n)}| < \tau_{\text{layer}}^{(l)} \right\} \tag{6.12}$$

where $R_j^{(l,n)}$ denotes relevance of neuron j in layer l for sample n, and $\tau_{\text{layer}}^{(l)}$ sets the layer-specific threshold. This capability transforms interpretability from a diagnostic tool into an optimisation strategy, creating architectures tailored to the relevant feature space rather than processing all available measurements.

6.3 Case Study I: Feedforward neural networks architectural optimisation

This case study applies REACH to FNNs for single-channel deployment scenarios. The objective is twofold: identify which input features contribute most to accurate channel estimation, and determine whether relevance-driven pruning can create compact architectures suitable for resource-constrained vehicular modems. We focus on the TRFI-DNN model, a three-layer fully connected network that refines classical TRFI estimates through learned nonlinear transformations.

6.3.1 Experimental configuration

The evaluation employs the VTV-SDWW channel model under very high mobility ($v =$ 200 km/h) with 64QAM modulation. These parameters exceed the baseline configuration (140 km/h, 16QAM) used in previous chapters, creating a demanding scenario where channel estimation accuracy critically affects BER performance. This choice validates that REACH-guided optimisation maintains effectiveness under severe propagation conditions. The received symbols undergo TRFI preprocessing (Section 2.5.4.3), with real and imaginary parts stacked along the subcarrier dimension (Section 3.3.2), resulting in 104 input features from the 52 active subcarriers as discussed in Section 3.3.2.

Following the hyperparameter optimisation procedure from Section 4.2.2, we use Optuna with Bayesian optimisation to identify the smallest architecture that achieves acceptable performance under the challenging 200 km/h 64QAM conditions. The optimisation tunes hidden layer sizes, learning rates, and training epochs to minimise validation loss whilst preventing overfitting. Starting from this compact baseline, rather than an oversized network, ensures that REACH-guided pruning identifies genuinely redundant parameters. Table 6.1 presents the best configuration of the TRFI-DNN baseline model. Sizes $[28, 19, 30]$ represent the sizes of the nodes in the 3 layers of the TRFI-DNN model. The model employs a truncated normal initialisation with a standard deviation of 0.05 and Adam optimisation with learning rate 2.79×10^{-4} and weight decay 10^{-5}.

Table 6.1: Configuration of the TRFI-DNN model for VTV-SDWW 64QAM under very high mobility.

Parameter	**VTV-SDWW 64QAM**
Input/Output size	104 / 104
Number of hidden layers	3 (Fully-connected)
Neurons per layer	[28, 19, 30]
Activation	ReLU
Optimiser	Adam (lr $= 2.79 \times 10^{-4}$)
Weight decay	10^{-5}
Batch size	128
LR scheduler	ReduceLROnPlateau
Initialisation	Truncated normal ($\sigma = 0.05$)

Following the REACH methodology described in Section 6.2, we calculate relevance scores for each input feature (subcarrier) across all input samples using the LRP epsilon rule. The TRFI-DNN operates on individual OFDM symbols rather than complete frames: each training sample consists of one OFDM symbol with 52 active subcarriers converted to 104 real-valued features by stacking real and imaginary components. This symbol-level processing is necessary for the feedforward architecture, which requires fixed-size 1D input vectors. Each frame contributes 50 training samples (one per OFDM symbol). In total,

we process 450,000 training samples (which is equivalent to 9000 samples) from the training dataset in batches of 128, accumulating relevance contributions through backward propagation. For each sample, we compute the forward pass with LRP tracing enabled, sum the output to create a scalar signal, and backpropagate to collect feature-wise relevances. Temporal aggregation captures in-frame correlations, whilst batch-wise averaging produces stable relevance scores that represent the absolute average contribution of each subcarrier to channel estimation performance.

Rather than employing fixed percentile thresholds, we implement an adaptive threshold-based categorisation strategy that identifies features based on their proportional contribution to the maximum observed relevance. Let R_i denote the average relevance score for feature i, and define the maximum absolute relevance as $R_{\max} = \max_i |R_i|$. We establish threshold levels as fractions of this maximum to define feature categories.

For compact model deployment, we define five feature sets spanning different compression levels (Table 6.2). The percentages indicate the cumulative proportion of total relevance captured by each set, measuring how much of the model's explanatory power is retained after feature selection. The **Critical** set (10 features, 9.6%) targets aggressive compression by selecting only the highest-relevance subcarriers. The Primary set (17 features, 16.3%) corresponds to features with relevance exceeding 10% of maximum, representing the most impactful contributors to channel estimation. The Significant set (20 features, 19.2%) extends beyond the primary threshold to include moderately relevant features. The Core set (41 features, 39.4%) contains all features above 1% of maximum relevance, excluding only those with minimal relevance.

Figure 6.2 shows the relevance distribution across all 104 input features, with features colour-coded according to their assigned categories. The visualisation indicates that high-importance features cluster in specific frequency regions rather than distribute uniformly across the spectrum. The Critical features (red) represent the peaks of the importance distribution, whilst the Primary set (orange) captures additional high-relevance regions. The polynomial trend line (dashed red line) demonstrates a gradual decay in average importance across the feature space, with notable peaks corresponding to frequency bands

Table 6.2: Feature sets for compact model deployment under VTV-SDWW 64QAM.

Category	Count	Percentage	Description
Critical	10	9.6%	Highest-relevance features only
Primary	17	16.3%	Above primary threshold ($R \geq 0.1R_{\max}$)
Significant	20	19.2%	Extended primary set
Core	41	39.4%	All non-negligible ($R \geq 0.01R_{\max}$)
Baseline	**104**	**100%**	**Complete feature set**

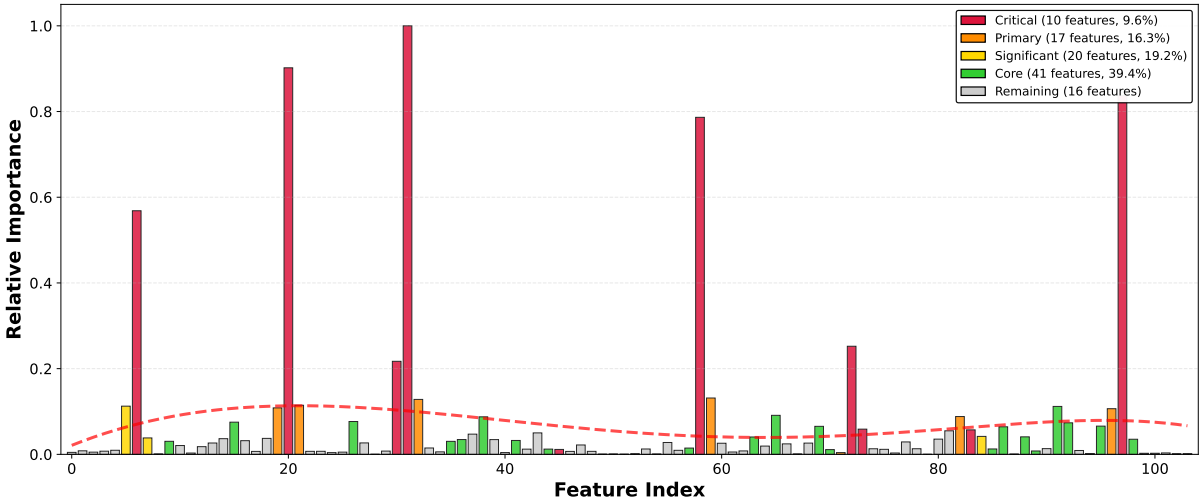


Figure 6.2: Feature importance distribution with category assignments for VTV-SDWW 64QAM, $v = 200$ km/h.

that strongly influence channel estimation accuracy.

6.3.2 Feature selection and retraining

We constructed models for each category by selecting only those features that belong to that category. This results in reduced input dimensions and different architectures for each category model through hyperparameter optimisation. The new architectures for each category ensure that the compact models learn best representations for their respective feature subsets. Table 6.3 presents the best configurations discovered by optimisation of hyperparameters using the same protocol in Section 4.5.2. Three random seeds with 50

trials each, exploring hidden layer sizes from 5 to 120 neurons per layer and learning rates from 10^{-5} to 10^{-2} were employed. Fixed hyperparameters include batch size (128), weight decay (10^{-5}), activation function (ReLU), optimiser (Adam), learning rate scheduling (ReduceLROnPlateau with factor 0.1, patience 20), and early stopping (patience 20).

Table 6.3: Best hyperparameters and validation performance for feature-selective TRFI-DNN models (VTV-SDWW 64QAM).

Category	**Features**	**Architecture**	**Learning Rate**	**Val. Loss**
Critical	10	[26, 11, 29]	1.08×10^{-3}	0.0337
Primary	17	[46, 19, 23]	4.37×10^{-4}	0.0320
Significant	20	[46, 29, 28]	6.87×10^{-4}	0.0310
Core	41	[25, 37, 29]	6.09×10^{-3}	0.0288
Baseline	**104**	**[28, 19, 30]**	2.79×10^{-4}	**0.0312**

The optimisation reveals significant architectural diversity across categories, with configurations adapting to the information content of each feature subset. The Core model demonstrates efficiency, achieving a validation performance of 0.0288. In contrast, the Critical model achieves a higher validation loss (0.0337). The following section will discuss another approach to achieving compact models: node pruning.

6.3.3 Node pruning

The previous sections demonstrate REACH's effectiveness for input feature selection. This section extends the analysis to architectural optimisation through node pruning, evaluating whether LRP can identify redundant neurons in hidden layers. We employ the VTV-SDWW channel under very high mobility conditions with 64QAM modulation to assess whether architectural pruning maintains estimation accuracy whilst reducing computational cost.

6.3.3.1 Pruning methodology

We implement LRP-based analysis on the trained baseline model (104 inputs, three hidden layers with 28, 19, and 30 node sizes, 104 outputs) to assess the importance of both input features and hidden layer nodes. The methodology comprises two distinct but complementary approaches: feature selection and node pruning.

For feature selection, relevance scores rank inputs with predefined thresholds, returning or keeping features whose removal would degrade BER by less than 1%. This process produces five compact feature sets with varying compression levels: Critical (10 features, 9.6%), Significant (20 features, 19.2%), Primary (17 features, 16.3%), and Core (41 features, 39.4%). These feature sets maintain channel estimation capability whilst reducing input dimensionality.

For node pruning, we apply REACH to calculate relevance scores for neurons across the hidden layers.

The relevance computation procedure processes a subset of training samples through the network. For each sample, we perform a forward pass with explanation enabled, sum the output to create a scalar, and backpropagate to collect layer-wise relevances. We then average relevances across all samples to obtain stable importance estimates for each neuron, quantifying their contribution to the model's predictions.

Rather than applying uniform pruning percentages across all layers, we implement an adaptive relevance-based pruning strategy. We sort neurons within each layer by their average absolute relevance scores in descending order.

6.3.3.2 Systematic pruning evaluation

To complement the adaptive relevance-based approach and establish safe pruning boundaries, we conduct a systematic evaluation across uniform pruning percentages. This analysis characterises the relationship between architectural compression and performance

retention.

For neuron significance assessment, we calculate average absolute relevance scores to account for both positive and negative contributions, as both influence network decisions through complementary mechanisms. The average relevance $\bar{R}_j$ for neuron j within a hidden layer is:

$$\bar{R}_j = \frac{1}{N} \sum_{n=1}^{N} |R_j^{(n)}| \tag{6.13}$$

where $R_j^{(n)}$ represents the relevance of neuron j for sample n in a dataset of size N.

We evaluate pruning percentages from 0% to 90% in 9% increments to systematically discover performance degradation patterns. For each pruning level $P\%$ applied uniformly across all hidden layers containing S nodes, the retention count becomes $\lceil S \times (1 - P/100) \rceil$, ensuring at least one node per layer remains even at extreme pruning levels. Neurons are ranked by average absolute relevance in descending order, with top-ranked neurons preserved and lower-ranked ones eliminated uniformly across all layers.

The evaluation protocol follows four steps. First, we initialise pruned architectures based on relevance rankings at each pruning percentage. Second, we train each pruned model from scratch using the hyperparameter optimisation protocol from Section 4.2.2, ensuring fair comparison by providing identical training conditions. Third, we evaluate models on validation data, recording validation loss, MSE and accuracy metrics with 0.05 tolerance. Finally, we assess computational cost through trainable parameter counts and FLOPs estimation for forward passes. Table 6.4 presents evaluation results using validation loss and validation accuracy across pruning levels, showing the trade-off between compression and performance.

Validation performance reveals three distinct pruning regions, as illustrated in Figure 6.3. The moderate region (0–45%) shows a gradual performance decline, maintaining acceptable functionality with validation loss increasing modestly from 0.030 to 0.032 and accuracy remaining above 32%. This region appears to balance efficiency and performance for practical deployment, suggesting that nearly half the neurons can be removed with low impact. The aggressive region (45–63%) exhibits accelerated degradation, indicating the

Table 6.4: Systematic pruning evaluation results for the baseline model under uniform pruning percentages.

Pruning (%)	**Parameters**	**FLOPs**	**Validation Loss** (MSE)	**Validation Accuracy**
0	7,315	14,526	0.030	36.20
9	6,764	13,424	0.030	35.81
18	5,928	11,752	0.031	34.49
27	5,235	10,366	0.031	34.14
36	4,601	9,098	0.032	32.92
45	3,943	7,782	0.032	32.47
54	3,191	6,278	0.034	29.16
63	2,711	5,318	0.037	25.38
72	1,997	3,890	0.036	26.50
81	1,416	2,728	0.039	23.93
90	748	1,392	1.0007	4.03

removal of increasingly critical components. Accuracy drops more sharply from 32.47% to 25.38%, and loss values grow substantially, suggesting these neurons contribute more significantly to model capability. Pruning beyond 63% triggers catastrophic collapse, with 90% pruning producing near-random accuracy (4.03%), confirming inadequate architectural capacity for the estimation task.

The systematic evaluation establishes that approximately 45% represents a good pruning threshold. This finding provides validation for the adaptive relevance-based approach, which achieved layer-specific compression rates (28.6%, 31.6%, 53.3%) that fall within or slightly exceed this threshold whilst maintaining performance through neuron selection. The combination of systematic exploration and relevance-based optimisation demonstrates that LRP-guided node pruning can improve architecture without significant performance loss, offering substantial resource savings suitable for vehicular environments.

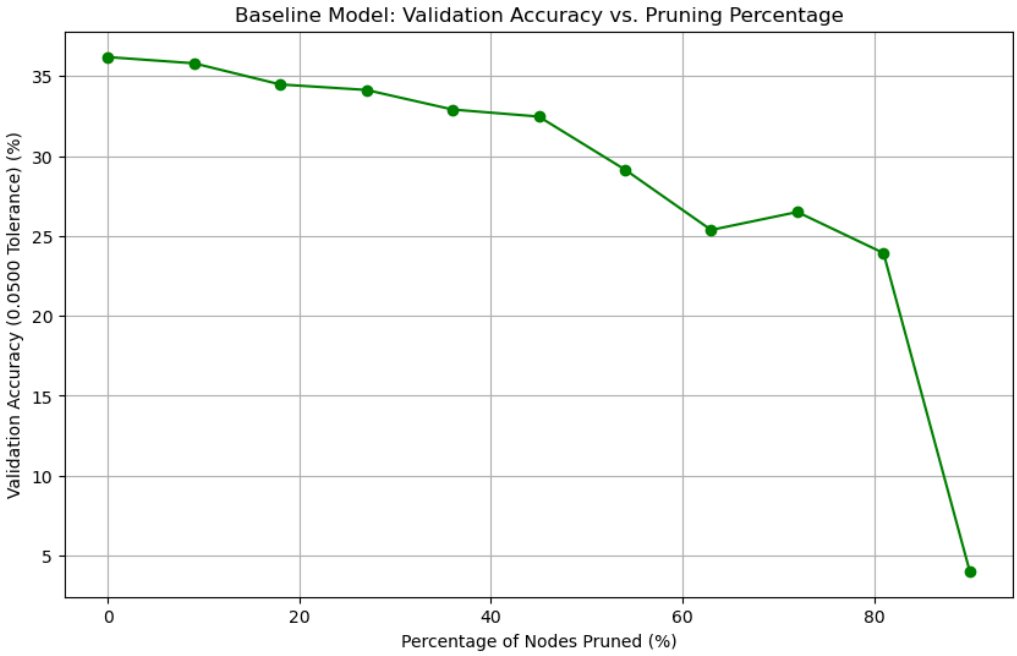


Figure 6.3: Validation accuracy versus pruning percentage for the baseline model under uniform pruning.

6.3.3.3 Relevance-based pruning visualisation

To provide insight into how LRP identifies important neurons across the network architecture, we present detailed visualisations demonstrating the distribution of computational relevance and the layer-specific node relevance. Figure 6.4 presents a comprehensive analysis of relevance patterns across all three hidden layers, showing the distribution of relevance scores, sorted neuron importance, and cumulative relevance curves that justify the pruning decisions.

The analysis reveals distinct relevance patterns across layers. As shown in the histograms (left column), most neurons concentrate near zero relevance with few high-relevance outliers, a pattern consistent across all layers. The sorted relevance plots (middle column) demonstrate rapid relevance decay after top-ranked neurons, justifying aggressive pruning. The cumulative relevance curves (right column) show that retained nodes capture 99.13%, 96.06%, and 84.75% of layer relevance for Layers 0, 2, and 4, respectively.

Layer 0 shows a strong concentration of relevance, with a steep decay curve indicating

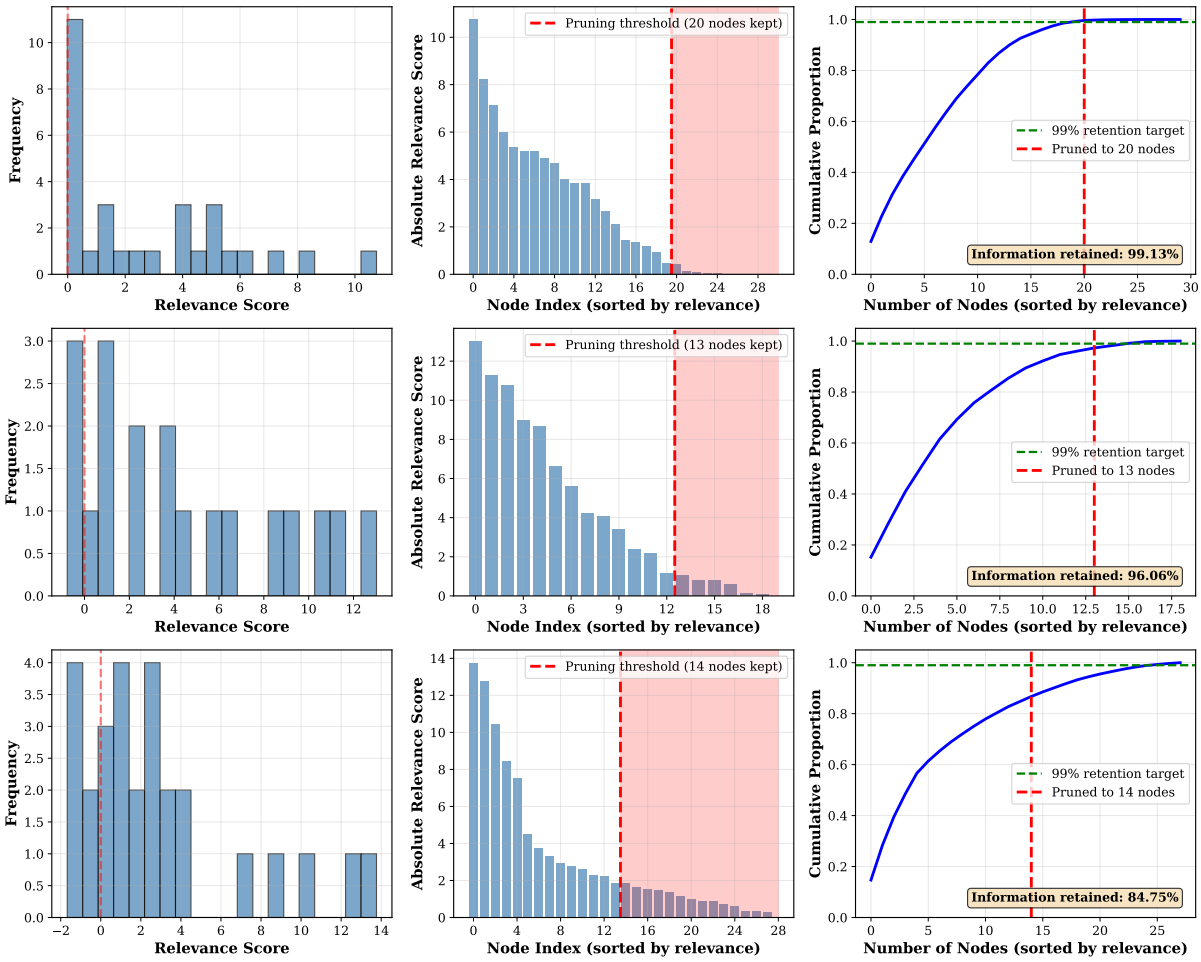


Figure 6.4: Relevance analysis and pruning decisions for hidden layers. Each row represents one layer (Layers 0, 2, and 4 from top to bottom). Left: Histogram of relevance scores. Middle: Sorted absolute relevance with pruning threshold (red line) and pruned neurons (pink region). Right: Cumulative relevance proportion retained.

clear separation between important and redundant nodes. Layer 2 demonstrates more uniform relevance distribution, yet still exhibits significant redundancy with the top 13 nodes capturing over 96% of relevance. Layer 4 shows the most aggressive pruning potential, with 14 nodes retaining nearly 85% of relevance whilst enabling 46.7% compression. Figure 6.5, to Figure 6.7 provide visualisations of node-level importance through relevance heatmaps. Each figure presents two complementary views: a complete heatmap displaying all neurons with black borders indicating retained nodes, and a focused visualisation showing only nodes preserved after pruning, arranged by descending relevance.

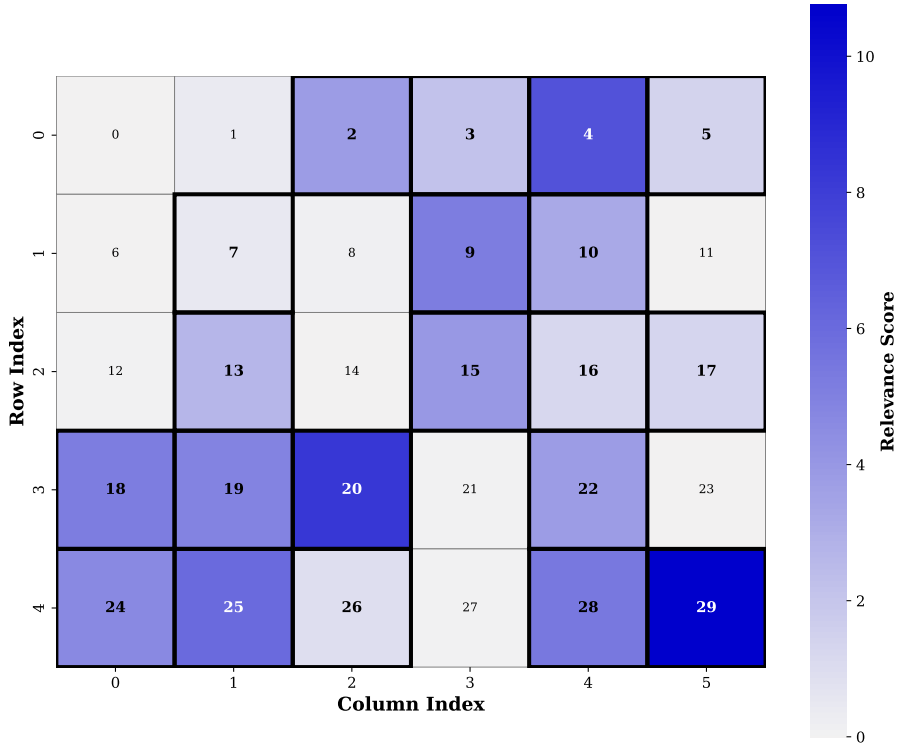


Figure 6.5: Complete node relevance heatmap for Layer 0. The heatmap displays all nodes with colour intensity indicating relevance (blue = higher values). Black borders highlight the 20 nodes out of 28 that were retained in the pruned architecture, representing **71.4**% retention.

These visualisations demonstrate REACH's capability to identify less relevant and more relevant features, enabling informed architectural decisions beyond traditional magnitude-based pruning approaches. The varying retention rates across layers (46.7% to 68.4%) indicate layer-specific importance patterns that uniform pruning strategies would overlook. By adapting compression to the actual relevance distribution in each layer, the approach maintains the neurons that matter most whilst eliminating those with minimal contribution.

Figure 6.8 shows an overview of the adaptive pruning approach across all three hidden layers. The resulting NodePruned architecture indicates that different layers exhibit varying levels of redundancy, with Layer 0 retaining 71.4% of nodes (20 of 28), Layer 2 retaining 68.4% (13 of 19), and Layer 4 retaining only 46.7% (14 of 30). The relevance profiles show that retained nodes consistently exhibit much higher average relevance than those pruned, confirming that low-contribution nodes are effectively removed. Moreover, the

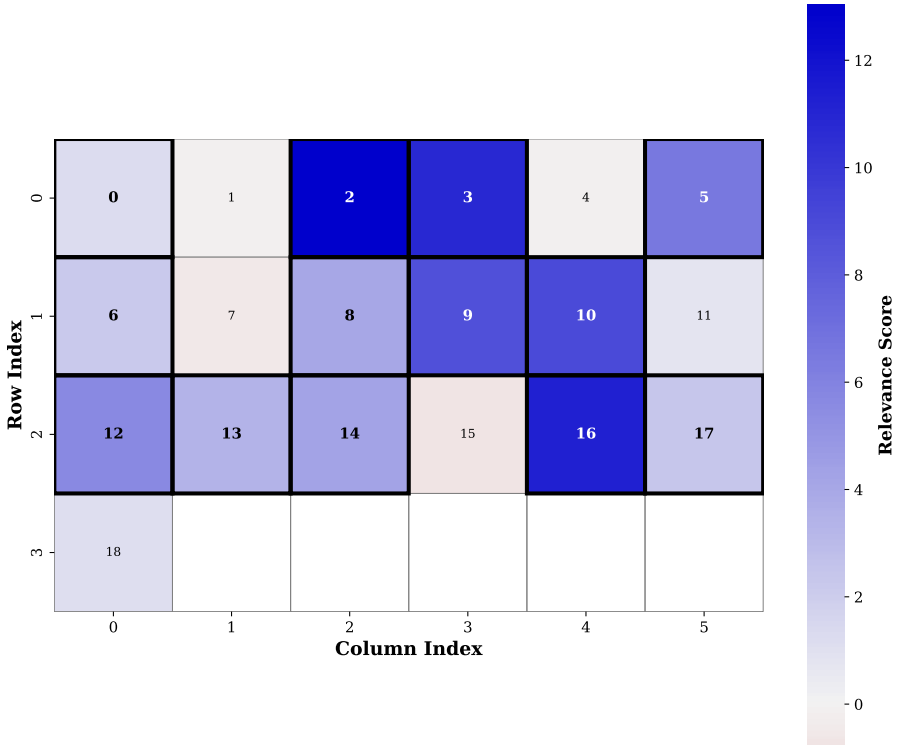


Figure 6.6: Complete node relevance heatmap for Layer 2 (Second Hidden Layer). The heatmap displays all 19 nodes with colour intensity indicating relevance (blue = higher values). Black borders denote the 13 nodes out of 19 nodes retained after pruning, representing **68.4**% retention.

pruning threshold increases across layers, indicating that deeper layers demand higher minimum relevance for retention.

6.3.3.4 BER performance analysis

To validate the practical effectiveness of both feature selection and node pruning, we evaluate BER performance across SNR ranges for all model variants. Figure 6.9 presents BER curves for the feature-selected retrained compact models (Critical, Significant, Primary, Core, Essential) alongside the baseline model under very high mobility VTV-SDWW channel conditions with 64QAM modulation.

From 0 to 20 dB SNR, NodePruned and compact models perform similarly to the baseline model. This shows that a careful design of DL estimators with reduced parameters and

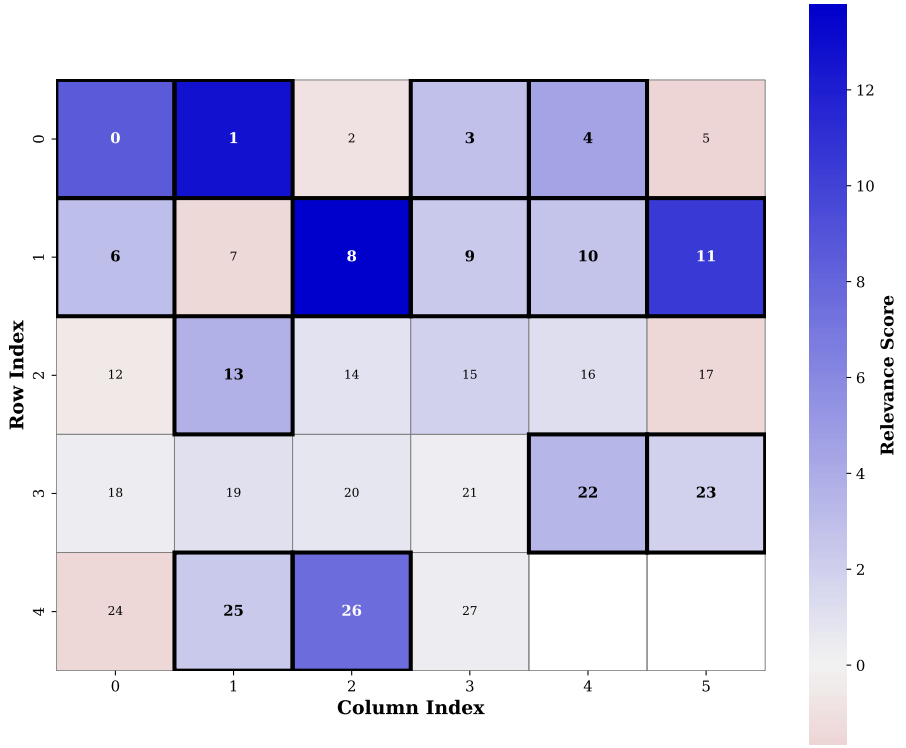


Figure 6.7: Complete node relevance heatmap for Layer 4 (Third Hidden Layer, 30 nodes). The heatmap displays all nodes with colour intensity indicating relevance (blue = higher values). Black borders highlight the 14 highly relevant nodes out of 30 nodes selected for retention, representing **46.7**% retention.

reduced input dimensionality can lead to similar or even better channel estimation in noisy channel conditions. The selected features capture essential information for robust estimation, while some model parameters add complexity but are not useful.

Performance divergence begins noticeably from 25 dB onwards. While the NodePruned, Core, Significant, and Critical categories show a marginal decline in performance compared to the baseline, the Primary category demonstrates better performance than the baseline from this SNR level forward.

6.3.3.5 Model complexity comparison

We quantify the computational complexity of the NodePruned and the derived compact feature set models through the number of parameters. Figure 6.10 shows the compar-

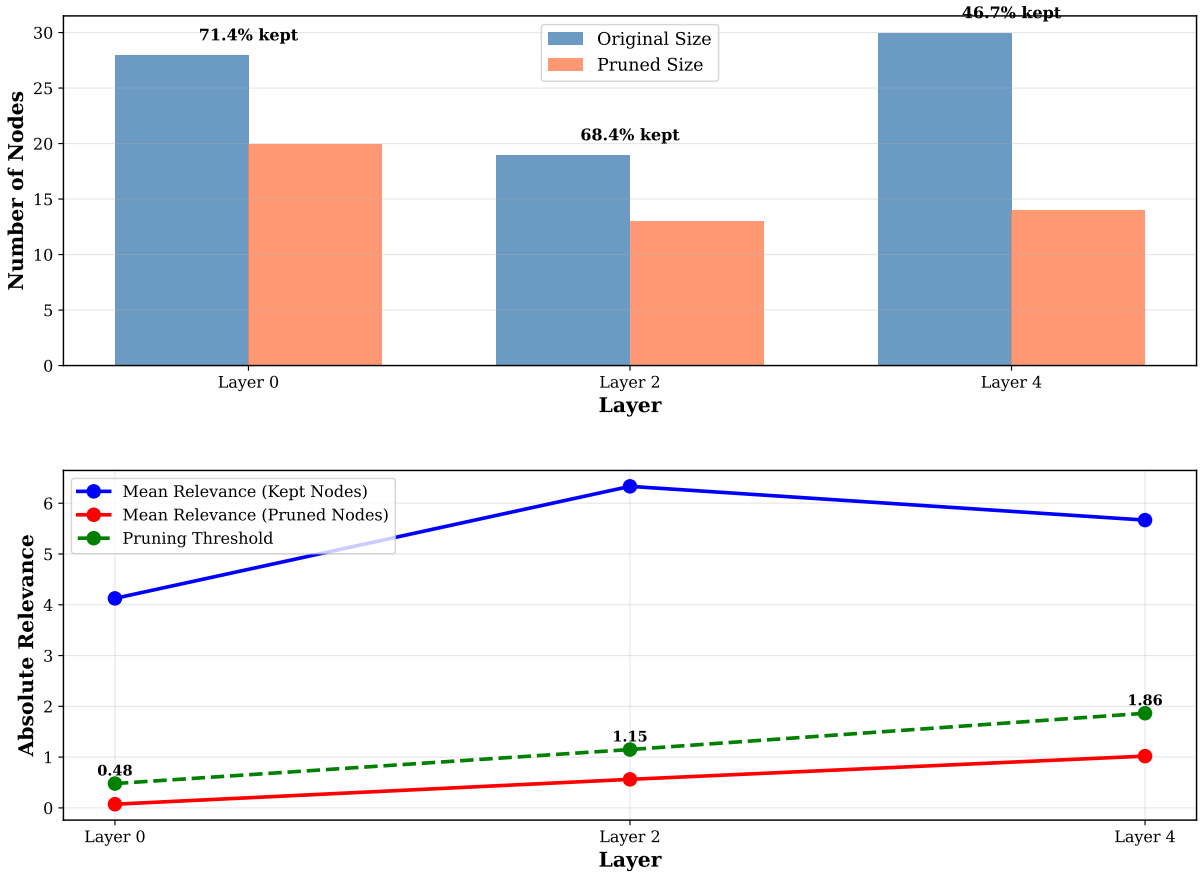


Figure 6.8: Adaptive node pruning across network layers. The top panel shows the original and pruned architectures, showing layer-specific compression rates. The bottom panel shows relevance distributions for retained and removed nodes.

ison of parameter counts across all categories relative to the full baseline model (7,315 parameters). The parameter count clearly shows that compression is achieved through two complementary methods:

1. Node Pruning: The **NodePruned** model, resulting from LRP-based node pruning, is the most compact architecture, requiring only 4,129 trainable parameters. This represents a substantial complexity reduction of **43.55**% compared to the full baseline.

2. Input Dimensionality Reduction: REACH-based feature selection generates compact models by reducing the input dimensionality and architectural modifications via hyperparameter optimisation. These models achieve a significant parameter re-

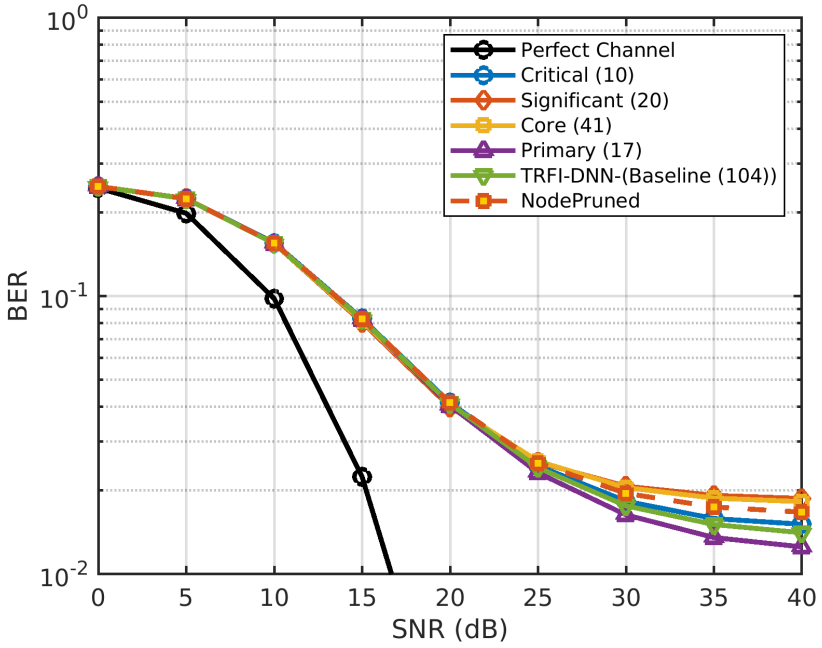


Figure 6.9: BER versus SNR for NodePruned, compact and baseline DNN models under a very high mobility VTV-SDWW channel with 64QAM modulation.

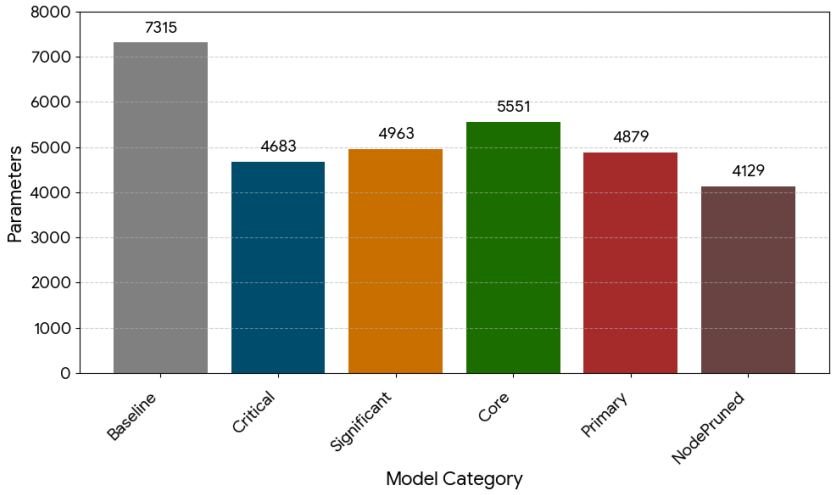


Figure 6.10: Model parameters across model categories.

duction ranging from **24.11**% (Core category, 5,551 parameters) to **35.98**% (Critical category, 4,683 parameters).

The Critical category is the model with the smallest selection of features (**35.98**% reduction), and the Primary category achieves a **33.30**% reduction (4,879 parameters).

These initial results, while derived from a single-channel setup, provide promising evidence for the feasibility of the REACH framework. In the next section, we evaluate the approach to incorporate a more realistic setting and an SOTA DL architecture.

6.4 Case Study II: TCN cross-channel interpretability

Case Study I demonstrated that relevance-driven pruning creates efficient architectures for single-channel scenarios. We now shift focus to a fundamentally different question: in multi-channel training, which features enable robust generalisation across diverse propagation environments? This case study applies LRP to the multi-channel trained TCN-DPA model from Chapter 5, investigating whether interpretability can explain the strong OOD performance observed previously.

The model achieved strong performance on unseen propagation environments (Figure 5.7), suggesting that exposure to diverse channel conditions during training produces representations that transfer beyond the specific scenarios encountered. What remained unclear was *why* this generalisation occurs and *which* learned features enable robust performance across varying Doppler spreads and delay characteristics.

Our analysis addresses three questions. First, do different channel models rely on similar input features, or do each environment develop channel-specific relevance patterns? High similarity would suggest that the model learned channel-invariant representations, whilst divergent patterns would indicate adaptive, channel-dependent feature selection. Second, can we identify a subset of features that exhibit high relevance across all channels? Such cross-channel important features would represent the core information required for reliable estimation regardless of propagation characteristics. Third, do relevance patterns correlate with physical channel properties such as Doppler frequency and delay spread?

6.4.1 Experimental configuration

This section describes the model used, its training, and the relevance calculation.

6.4.1.1 Multi-channel trained model

As mentioned, our analysis employs the TCN-DPA model developed and trained on the multi-channel, mixed-SNR dataset described in Section 5.5. This model was trained on 54,000 frames drawn equally from VTV-SDWW, RTV-SS, and VTV-EX channel models and 16QAM modulation is used. The model accepts inputs of dimension (100, 52), where 100 represents the interleaved real and imaginary components along the OFDM symbol dimension, and 52 corresponds to the active subcarriers. This results in a total input dimensionality of 5200 features (100×52) per frame.

6.4.1.2 Relevance computation methodology

We adapt the REACH-based LRP implementation for temporal convolutions, residual connections, and weight normalisation. For an input $\mathbf{x} \in \mathbb{R}^{100\times52}$, relevance propagates backwards through the network following the conservation principle of Equation ((2.86)). The output relevance initialises to the squared L2 norm of the model output $\mathbf{y}$, providing a scalar target for relevance decomposition.

For each channel model and SNR level, we computed relevance across **2000 test frames** ($\mathbf{N = 2000}$). The resulting frame-wise scores were then processed using the **batch aggregation strategy** detailed in Section 6.2.4 to obtain stable estimates. This produces an average relevance map $\bar{R} \in \mathbb{R}^{100\times52}$ for each channel-SNR combination, indicating which time-frequency positions contribute the most to the channel estimate.

We analyse three SNR levels (0, 20, 40 dB) to capture relevance patterns under noise-dominated (0 to 15 dB), intermediate (20 to 25 dB), and clean conditions (30 to 40 dB). The analysis includes all six channel models investigated in this work, enabling comparison

between ID channels seen during training (VTV-SDWW, RTV-SS, VTV-EX) and OOD channels encountered only during testing (VTV-UC, RTV-UC, RTV-EX).

6.4.2 Cross-channel relevance patterns

To understand cross-channel generalisation, we first assess whether various channel models show similar or different relevance patterns. This subsection explores relevance distributions across six vehicular channels, starting with visualisations of each channel and then measuring cross-channel similarity via correlation analysis.

6.4.2.1 Individual channel analysis

Figure 6.11 shows the relevance map for VTV-SDWW at 0 dB, selected as a representative example. Similar patterns emerge across the 20 and 40 dB SNR levels and also across all six channel models with variations primarily in magnitude rather than structure. The visualisation shows a uniform distribution of relevance across time channels (vertical axis), with distinct vertical bands corresponding to the pilot subcarrier positions marked by cyan dashed lines. The consistent pattern across temporal positions indicates that the TCN aggregates information across the entire OFDM frame rather than relying on specific symbol locations.

The relevance magnitudes vary with the SNR. At 0 dB, the average relevance ranges from 0.62 to 21.45 across channels (OFDM symbol dimension), reflecting the model's intensive feature processing under severe noise conditions. At 20 dB, this range narrows to 0.74 to 6.11, and at 40 dB, it is equal to 0.80 to 6.17.

6.4.2.2 Cross-channel correlation analysis

To quantify similarity in learned feature importance across channels, we compute Pearson correlation coefficients between flattened relevance vectors for all channel pairs. Fig-

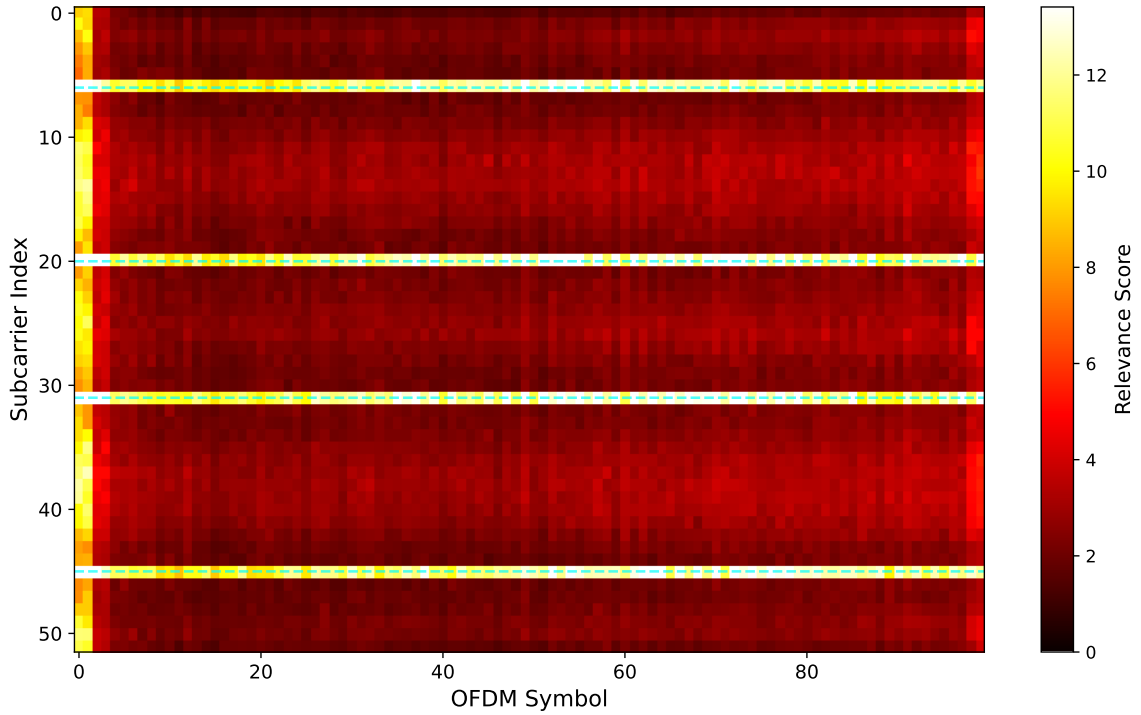


Figure 6.11: Relevance map for VTV-SDWW at 0 dB showing uniform temporal distribution with elevated relevance at pilot positions and at the initial OFDM symbols positions, and some at the last OFDM symbols.

ure 6.12 presents the correlation matrix at 20 dB.

All pairwise correlations lie in the range 0.81 to 0.99 at 20 dB, with even higher similarity at 0 dB (0.95 to 1.00) and 40 dB (0.84 to 0.99). The strongest correlations (0.99) occur between RTV-SS and VTV-UC, two channels with substantially different Doppler characteristics (600 Hz versus 500 Hz) and communication scenarios as expressed by the TDL parameters of each (roadside-to-vehicle versus vehicle-to-vehicle).

High correlations appear not only among ID channels (VTV-SDWW, RTV-SS, VTV-EX) but also between ID and OOD pairs. For instance, VTV-SDWW ID and VTV-UC OOD exhibit a correlation of 0.90, whilst VTV-EX (ID) and RTV-UC (OOD) achieve 0.95. These cross-distribution similarities explain the strong OOD generalisation observed in Chapter 5: the model applies similar feature selection strategies to previously unseen channels, relying on learned patterns that transfer across propagation environments.

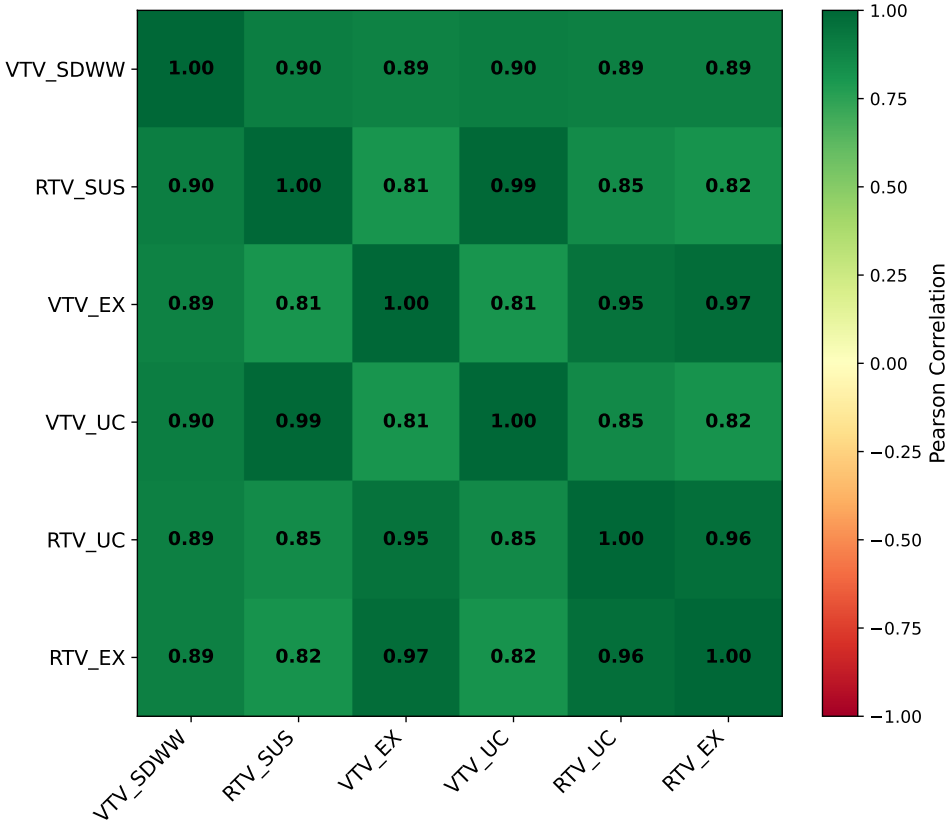


Figure 6.12: Cross-channel relevance correlation matrix at 20 dB showing high similarity (0.81 to 0.99) across all channel models.

The uniformly high correlations at 0 dB (minimum 0.95) show that under severe noise, all channels converge to nearly identical relevance patterns. The model depends on the same robust features when noise dominates. At higher SNR, slight differentiation emerges (minimum 0.81 at 20 dB), indicating that channel-specific refinement becomes possible once noise decreases, while still maintaining substantial overlap in feature prioritisation.

6.4.3 Cross-channel important feature identification

Cross-channel correlation analysis showed consistent relevance patterns (0.81 to 0.99), suggesting that the model prioritises similar features across diverse propagation environments. This subsection identifies which specific features remain essential across all channels and analyses their distribution across temporal and spectral regions.

6.4.3.1 Defining universal features

Cross-channel important features are those that exhibit high relevance across all tested channel models and represent the core input information required for reliable estimation, regardless of propagation characteristics. To identify these features, we apply a percentile-based threshold: for each channel, we mark the features in the top 25% of relevance. A feature qualifies as cross-channel important if it exceeds this threshold in all six channels at least once across all SNR levels.

Figure 6.13 visualises the cross-channel important feature mask. Green pixels indicate positions where relevance consistently ranks high across all channels, whilst red pixels denote channel-specific or low-importance positions.

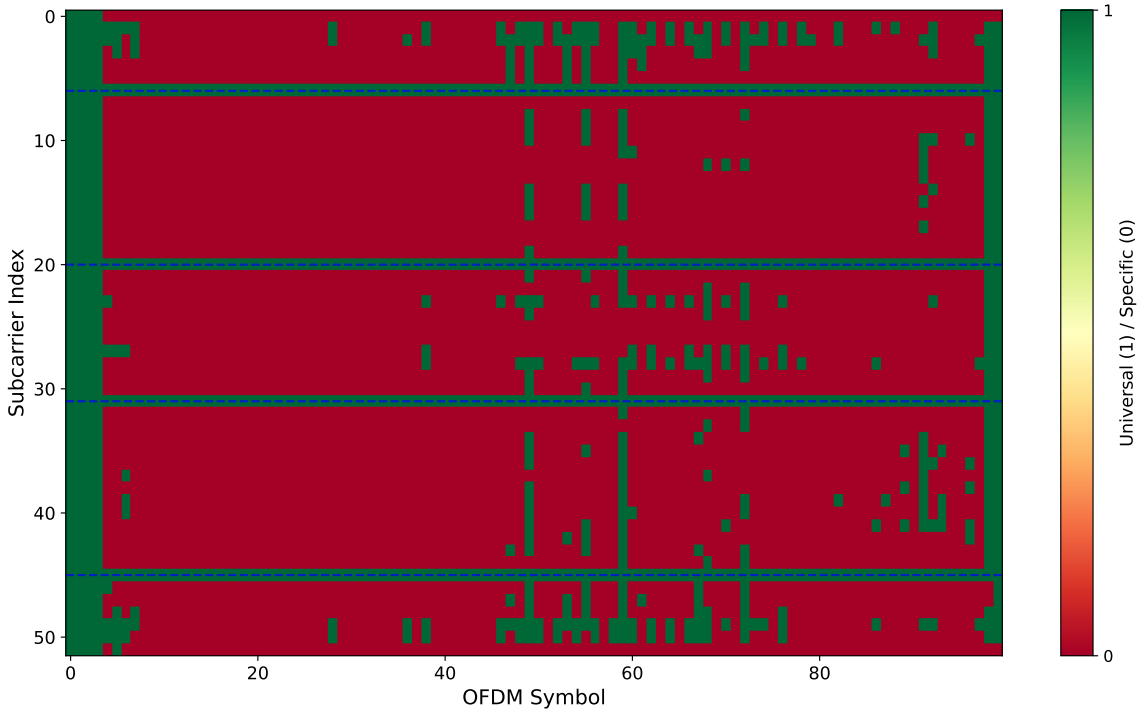


Figure 6.13: Cross-channel important features (green) exhibiting high relevance across all six channel models. Four vertical bands correspond to pilot positions; horizontal patterns indicate temporally critical positions.

The analysis identifies 1020 cross-channel important features out of 5200 total (19.6%). This proportion strikes a balance: the model shares a substantial core of features across

channels, explaining generalisation capability, whilst retaining flexibility for channel-specific adaptation through the remaining 80.4% of features.

Visual inspection reveals two distinct patterns. Four prominent vertical green bands align precisely with pilot subcarrier positions (indices 6, 20, 31, 45), confirming that transmitted pilot symbols constitute primary channel-invariant features. These known reference symbols provide unambiguous channel information across all propagation scenarios, which naturally emerges as universally important.

Beyond pilots, horizontal green bands appear at specific time channel (OFDM symbols) positions, indicating that certain OFDM symbols contribute consistently across channels. We can also observe the concentration of cross-channel important features in early symbols and late positions of the frame. In the middle part of the frame, we can also see a bit of concentration of the cross-channel important features.

6.4.3.2 Temporal and spectral analysis

Table 6.5 provides a detailed breakdown of cross-channel important features by temporal and spectral region. The distribution exhibits clear patterns that reveal how the TCN prioritises information across the OFDM frame.

Temporal distribution shows concentration in early symbols (42.8% of symbols 0–5 are cross-channel). Mid-early symbols (6–15) show the lowest cross-channel density (8.1%), suggesting channel-specific adaptation roles, whilst mid and late symbols exhibit moderate density (18.1% and 19.5%).

Spectral distribution reveals surprising uniformity: edge (20.5%), mid (19.0%), and central (20.5%) subcarriers exhibit similar cross-channel important feature densities.

Subcarrier type breakdown reveals the most striking pattern: all 400 pilot features (100.0%) qualify as cross-channel important, whilst only 620 of 4800 data features (12.9%) achieve cross-channel status. Despite pilots being a small fraction of total features (7.7%), their complete cross-channel classification confirms their fundamental role as known reference

Table 6.5: Temporal and spectral analysis of cross-channel important features at 20 dB SNR.

Feature Category	Total Features	Cross-Channel Important Features (%)
Temporal Regions		
Symbols 0–5 (Early)	624	267 (42.8%)
Symbols 6–15 (Mid-early)	1040	84 (8.1%)
Symbols 16–30 (Mid)	1560	283 (18.1%)
Symbols 31–49 (Late)	1976	386 (19.5%)
Spectral Regions		
Edge Subcarriers (0–5, 47–51)	1100	226 (20.5%)
Mid Subcarriers (6–20, 32–46)	3000	569 (19.0%)
Central SC (21–31)	1100	225 (20.5%)
Subcarrier Type		
Pilot subcarriers	400	400 (100.0%)
Data subcarriers	4800	620 (12.9%)
Total	**5200**	**1020 (19.6%)**

points. However, the 620 cross-channel data features still outnumber pilots.

6.4.3.3 Pilot versus data subcarrier contribution

The model processes two types of subcarriers: four pilots carrying known symbols and 48 data positions with DPA-estimated symbols. Figure 6.14 compares their relative contributions across channels and SNR levels.

Data subcarriers contribute 65–92% of total relevance across all configurations, exceeding pilot contributions (8–35%). This ratio increases with SNR: at 0 dB, pilots account for 27–35% of relevance, declining to 11–14% at 20 dB and 8–10% at 40 dB. As SNR improves and DPA estimates become more reliable, the model places greater trust in the spatial-

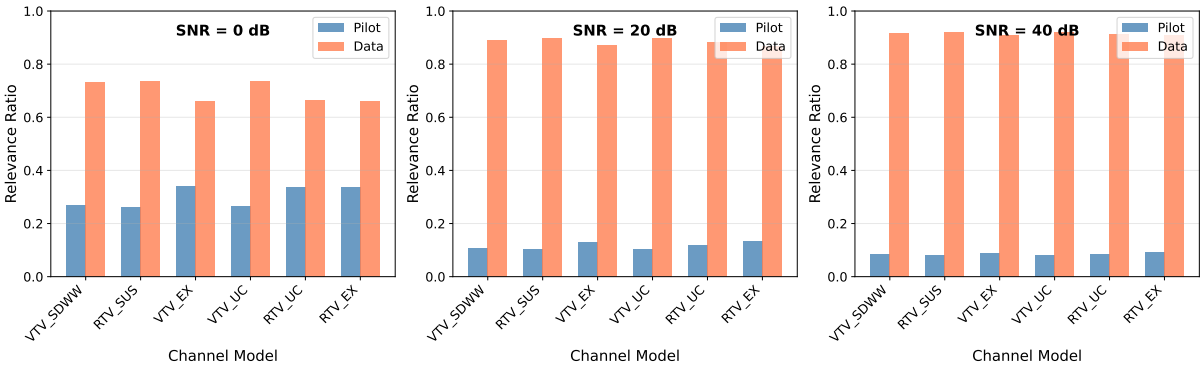


Figure 6.14: Pilot versus data subcarrier relevance contribution across SNR levels. Data subcarriers dominate (65–92%) despite pilots being known references.

frequency information from data positions.

6.4.4 Cross-channel feature validation

To validate that the identified cross-channel relevant features are functionally important rather than simply statistically correlated, we conducted controlled experiments comparing feature-reduced models against the original baseline. This validation addresses a critical question: do cross-channel important features capture essential channel information sufficient for accurate channel estimation?

6.4.4.1 Experimental design

We train two models and compare them against the original multi-channel baseline from Chapter 5:

- **Baseline:** Original multi-channel TCN with full feature set (5,200 features, 100 channels $\times$ 52 subcarriers) (from Chapter 5).
- **cross-channel:** Newly trained model using only the 1020 identified universal features (19.6% of the full set).

The feature selection is implemented through binary masking: features not in the selected set are zeroed in the input, whilst selected features retain their original normalised values. This masking approach enables a direct comparison of information content without architectural changes. All three reduced models use the same optimised hyperparameters as the baseline.

The random feature mask is generated with a fixed seed (42) for reproducibility and represents an unbiased control: if cross-channel important features truly capture important channel properties, the cross-channel model should significantly outperform the random model despite identical feature counts.

6.4.4.2 Training results

The cross-channel model converged after 84 epochs with a validation of 0.0008, compared to the baseline validation MSE of 0.0002. The low loss close to the baseline confirms the model successfully learns from the reduced feature set.

6.4.4.3 Performance evaluation

Figure 6.15 presents NMSE performance across all six channels, and Figure 6.16 shows the averaged comparison between ID and OOD scenarios. The cross-channel model maintains competitive NMSE performance despite using only 19.6% of input features. Across all channels and SNR levels, the NMSE gap between Baseline and the cross-channel model remains within approximately 2–5 dB, demonstrating that the LRP-identified features capture the essential information required for channel estimation. The consistent gap across both ID and OOD channels confirms that the identified cross-channel features generalise beyond the training distribution, corroborating the high cross-channel correlations (0.81–0.99) observed in Section 6.4.2.2.

Figure 6.17 shows the BER performance. The cross-channel model achieves near-identical error rates up to an SNR of 25 dB to the baseline across all channels and then starts to

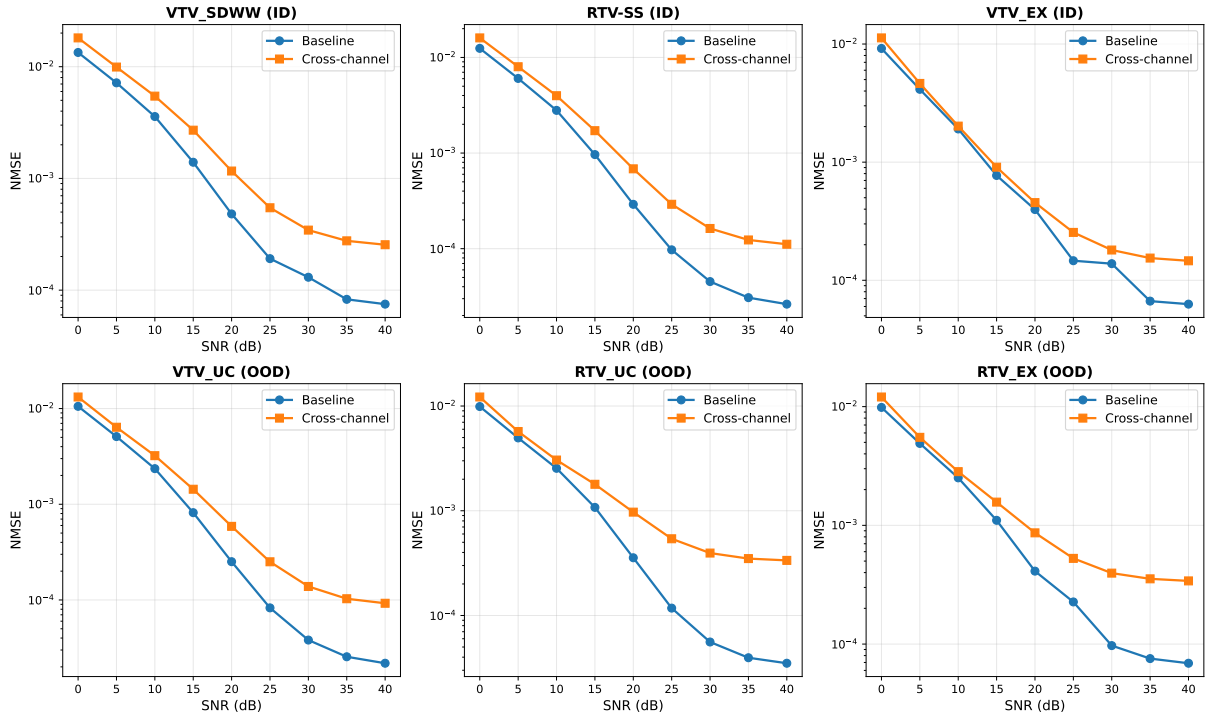


Figure 6.15: NMSE performance comparison between baseline (full features) and cross-channel (19.6% features) models on ID channels (top row) and OOD channels (bottom row).

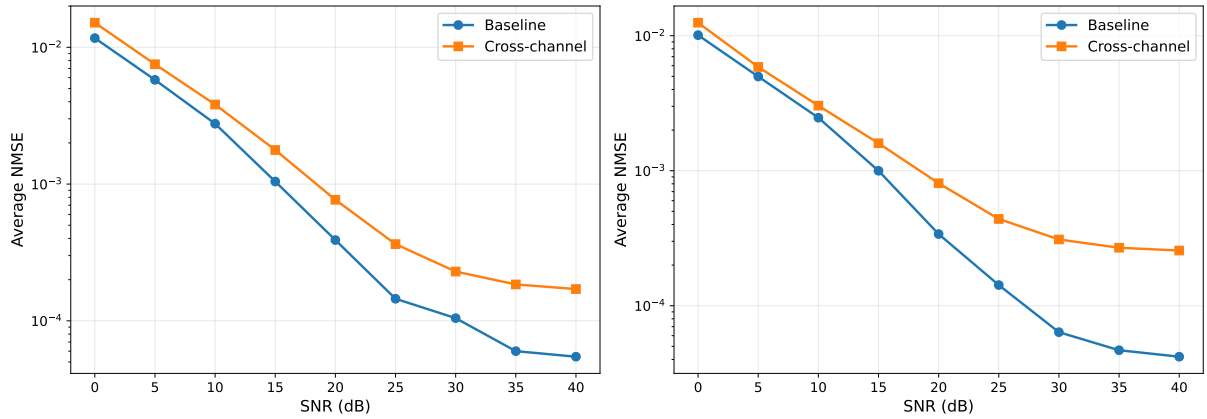
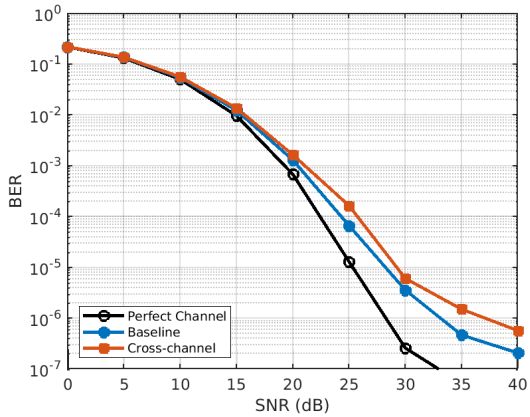
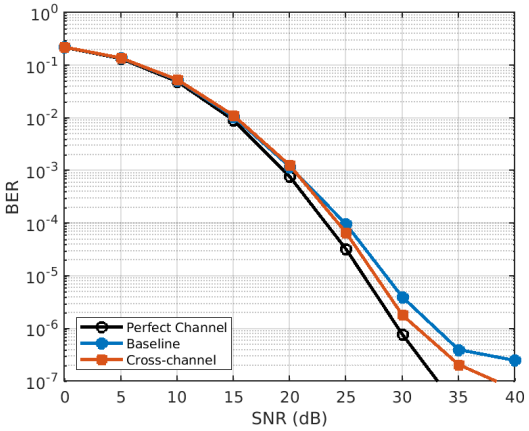


Figure 6.16: Average NMSE performance of the baseline and cross-channel on ID channels (left) vs OOD channels (right).

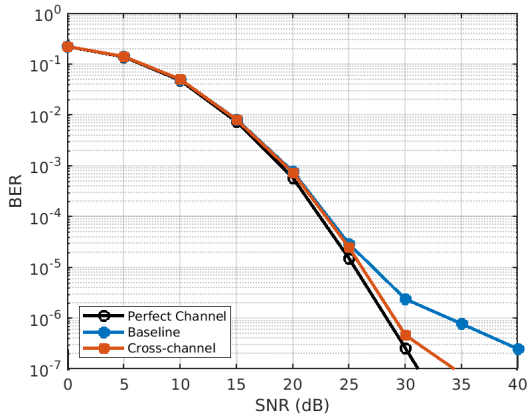
degrade. Despite the 80% reduction in input features, the performance of the cross-channel model is very commendable. In some cases, such as VTV-EX and RTV-SS, the cross-channel model achieves a BER slightly better than that of the baseline. These results validate the effectiveness of the REACH framework as an effective interpretability tool to identify features that are important in DL multi-channel. In some cases, training



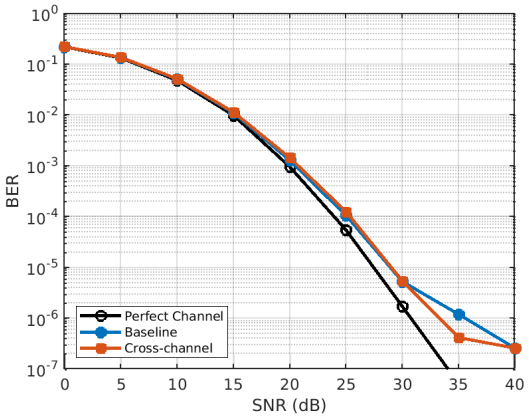
(a) VTV_SDWW (ID)



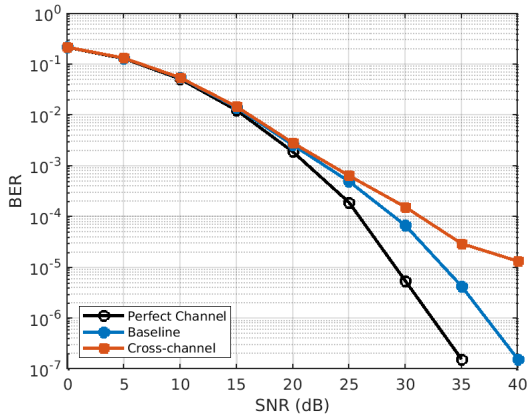
(b) RTV_SS (ID)



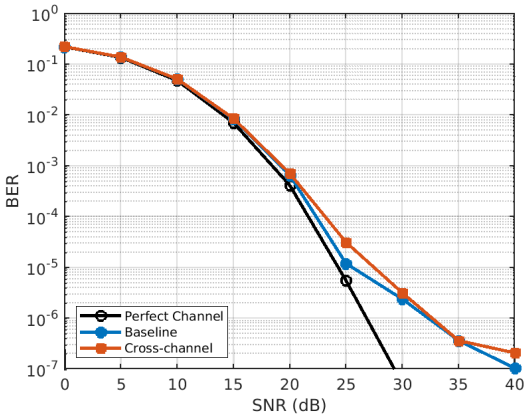
(c) VTV_EX (ID)



(d) VTV_UC (OOD)



(e) RTV_UC (OOD)



(f) RTV_EX (OOD)

Figure 6.17: BER performance comparison between baseline and cross-channel model across all channel models.

with only cross-channel important features actually leads to better results than Baseline, as shown in Figures 6.17(b) and 6.17(c).

6.5 Chapter summary

This chapter applied LRP to analyse and optimise DL-based channel estimators for IEEE 802.11p vehicular communications. We developed the REACH framework, which extends LRP through temporal aggregation, batch processing, and percentile-based feature categorisation.

Here is a summary of our main findings:

1. For single-channel FNN estimators (Case Study I), REACH-based feature categorisation identified substantial input redundancy. Models retrained on different feature subsets achieved BER performance comparable to the full baseline. This indicates that effective channel estimation does not require the entire input space. See Figure 6.9.

2. Node-level relevance analysis revealed substantial hidden layer redundancy in FNNs. The NodePruned model achieved 43.6% parameter reduction (from 7,315 to 4,129) with no significant degradation in BER below an SNR of 25 dB. See Section 6.3.3.

3. For multi-channel TCN estimators (Case Study II), cross-channel relevance correlation exceeded 0.81 across all channel pairs at 20 dB, indicating that the model learns channel-invariant feature representations. This explains the strong OOD generalisation observed in Chapter 5. See Section 6.4.2.2.

4. Cross-channel important feature identification found that only 1,020 features (19.6% of the input space) exhibit high relevance across all six channel models. Models evaluated using only these features achieved the BER performance comparable to that of a full baseline (multi-channel DPA-TCN) on all six channels. See Figure 6.17.

In conclusion, this chapter demonstrates that LRP-based interpretability serves not only as a diagnostic tool but as a practical methodology for model architecture improvement. The REACH framework enables systematic identification of essential features and redundant parameters, as demonstrated by the 43.6% parameter reduction achieved through node pruning in Case Study I.

Chapter 7

Conclusion

This chapter concludes the thesis by evaluating the extent to which the research objectives were achieved. A summary of the research methodology and the main contributions is provided, followed by directions for future work and concluding remarks on the practical viability of DL-based channel estimation for IEEE 802.11p vehicular communications.

7.1 Overview

This work investigated DL-based channel estimation for IEEE 802.11p vehicular communications, with the aim of improving the accuracy of the estimation under IEEE 802.11p channel conditions with high mobility, where conventional estimators could have limited performance. Four objectives were addressed:

1. To develop a reproducible simulation framework for IEEE 802.11p and establish the performance limitations of classical estimators in high-mobility channels.
2. To develop and evaluate DL architectures and training strategies that improve estimation accuracy and generalisation across diverse vehicular environments.

3. To analyse the trade-offs between estimation performance and computational complexity, with particular attention to single-frame inference latency, to assess feasibility for real-time deployment.

4. To develop an interpretability framework that enables systematic model compression through relevance analysis.

This chapter evaluates the extent to which these objectives were achieved. We summarise the contributions of this research and discuss directions for future work.

7.2 Summary of research methodology

The research adopted an experimental, benchmarking-driven methodology. We first implemented an IEEE 802.11p PHY layer simulator in MATLAB, generating synthetic datasets compliant with the standard vehicular channel models specified in [12]. 16QAM was used as the primary modulation scheme, with 64QAM included for a specific high-mobility use case.

Classical estimators (LS, MMSE, DPA, STA, and TRFI) were implemented first to establish performance baselines and identify their failure modes in high-mobility conditions. DPA was retained as a building block for the DL pipeline following [6]–[8]. DL architectures were then developed and evaluated in a stepwise manner, starting from recurrent baselines (LSTM and GRU variants) and progressing to the proposed convolutional architecture. Training strategies were investigated independently of architecture choice, comparing high-SNR training against mixed-SNR training (0–40 dB) and single-channel against multi-channel training. Finally, the REACH interpretability framework was applied to identify relevant features and guide model compression. All architectures were compared on BER, NMSE, and single-frame inference latency measured under a consistent protocol.

7.3 Summary of contributions

This thesis showed that DL-based channel estimation for IEEE 802.11p vehicular communications can achieve robust cross-channel generalisation and efficient single-frame inference. When carefully designed and trained, DL-based estimators offer superior link-level performance compared to classical methods in dynamic vehicular environments. The specific contributions made in the course of this research are the following:

- We reimplemented existing DL architectures from literature (FNNs, LSTMs and GRUs baselines) and then developed two novel architectures (Dual-Cell LSTM-DPA and DPA-TCN) for IEEE 802.11p vehicular communications, presented in Chapter 4. This comparative analysis established the superiority of LSTMs and GRUs over FNN based estimators (STA-DNN, TRFI-DNN).

- The development of a novel non-causal DPA-TCN architecture, presented in Chapter 4. This contribution encompasses two innovations: (1) *Non-causal TCN design*: Unlike the causal TCNs typically used for temporal sequence modelling (Section 2.6.2.4), our TCN employs symmetric padding to enable bidirectional receptive fields across subcarriers, learning early and later frequency-domain correlations. (2) *DPA $\rightarrow$ TCN architecture*: We use DPA estimates as input to the TCN rather than the conventional approach of applying neural networks before DPA. This ordering exploits complementary strengths: whilst DPA provides temporal tracking through decision-directed feedback, it suffers from error propagation where incorrect symbol decisions degrade subsequent estimates. The TCN, with its dilated convolutions and bidirectional receptive field, learns to identify and correct these DPA inconsistencies. This reversed pipeline reduces cumulative error propagation close to the perfect channel (see Figure 5.6 and Figure 5.8) and enables the TCN to process all subcarriers in parallel in a single forward pass, which translates directly into a much lower per-frame inference latency than the sequential recurrent baselines.

- The evaluation of models trained with *mixed-SNR* and *multi-channel* training strategies presented in Chapter 5 showed behaviour that depends strongly on architecture.

LSTM estimators trained on *mixed-SNR* and *multi-channel* datasets underperform relative to their high-SNR single-channel variants at high SNR, whereas DPA-TCN was found to benefit from diverse noise levels (0–40 dB) across all SNR regions (see Figure 5.2 to Figure 5.4 in Section 5.4.3). Multi-channel training extended this idea: the DPA-TCN model trained on combined datasets from multiple channel models (VTV-SDWW, RTV-SS, VTV-EX) generalised effectively to both ID and OOD propagation environments. This eliminated the need for multiple channel-specific estimators and addressed the generalisation limitations observed when training exclusively at high SNR (40 dB), as was common in prior work.

- Inference latency analysis established that the proposed DPA-TCN architecture achieves a median single-frame inference latency of 4.16 ms and a 90th-percentile latency of 4.42 ms on a single NVIDIA A40 GPU. This is a 16- to 32-fold reduction relative to the recurrent baselines (68–135 ms), whose sequential gate updates across the 50 OFDM symbols of each frame accumulate into per-frame latencies that exceed the receiver processing budget for continuous IEEE 802.11p frame reception. The per-frame latency achieved by DPA-TCN fits within the receiver processing budget, and the parallel structure of the network makes it well suited to hardware-accelerated implementation on platforms such as FPGAs or automotive AI accelerators, where convolutional kernels can typically achieve substantially lower inference times than on general-purpose GPUs.

- The development of REACH, an interpretability framework based on LRP, presented in Chapter 6 extends LRP through temporal aggregation, batch processing, and percentile-based feature categorisation, quantifying which temporal and spectral input features contribute most to channel predictions. For single-channel FNN estimators, REACH revealed substantial input redundancy, enabling 43.6% parameter reduction (7,315 to 4,129 parameters) with no significant BER degradation. For the multi-channel DPA-TCN estimator, cross-channel relevance correlation exceeded 0.81, explaining the strong OOD generalisation observed in Chapter 5. Cross-channel important feature identification revealed that only 19.6% of input features are important across six channel models.

These contributions collectively establish the non-causal DPA-TCN architecture, trained with mixed-channel and mixed-SNR strategies and optimised through REACH-guided feature selection, as a strong link-level estimator for IEEE 802.11p vehicular communications. The combination of accurate link-level performance, efficient single-frame inference, robust generalisation, and interpretability addresses the principal challenges identified at the start of this thesis. We note, however, that link-level metrics such as BER and NMSE constitute a necessary but not sufficient condition for the safety-critical requirements of V2X applications. End-to-end validation against burst-error and packet-delivery metrics, under realistic traffic and interference conditions, is needed to establish fitness for safety-critical deployment, and is identified as future work below.

7.4 Further application and future work

This thesis forms the basis for several directions of future research: (1) architectural optimisation of the TCN, (2) hardware implementation and field validation, (3) extension to next-generation vehicular standards, (4) extension to MIMO systems, and (5) application-layer and burst-error evaluation. Specific issues that we would like to address in the future include:

- **Architectural pruning for TCN models**: Chapter 6 demonstrated cross-channel important feature identification through input-level relevance analysis. Extending REACH-based relevance analysis to identify redundant convolutional filters or residual blocks within the TCN architecture itself could enable architectural compression beyond input feature selection, potentially achieving further parameter reduction whilst maintaining the parallel processing advantages.
- **Hardware implementation and field validation**: Whilst this study evaluated computational complexity through FLOPs analysis and measured inference latency in simulation, implementation of the optimised DPA-TCN model on embedded processors (such as vehicular onboard units) is necessary to validate real-time processing

capabilities under physical hardware constraints. Field testing with actual vehicular measurement would validate whether the generalisation benefits observed with channel models transfer to real-world propagation conditions, particularly for scenarios which may not be well-represented in standardised models.

- **Extension to next-generation vehicular standards**: This research focused on IEEE 802.11p OFDM systems with specific frame structures (50 OFDM symbols per frame, 52 active subcarriers). A logical next step is to investigate whether the architectural insights (TCN superiority over RNNs, benefits of non-causal frequency-domain processing), training strategies (mixed-SNR and multi-channel robustness), and interpretability findings (cross-channel important features) generalise to the emerging IEEE 802.11bd amendment and to cellular V2X standards such as LTE-V2X and 5G NR-V2X, which use different physical layer characteristics, pilot patterns, and numerologies.

- **Extension to MIMO systems**: The current work focused on SISO configurations. Future research should investigate the scalability of the proposed non-causal TCN architectures and REACH framework to MIMO systems, where spatial correlation across antenna elements adds another dimension of complexity to the channel estimation task. The frequency-domain processing approach may extend naturally to joint spatial-frequency correlation learning.

- **Application-layer and burst-error evaluation**: This thesis evaluated estimator performance using frame- and SNR-averaged BER and NMSE, which are standard link-level metrics. These averaged metrics do not capture the burstiness of residual errors caused by decision-directed feedback, deep fades, or hidden-terminal collisions, nor do they directly reflect packet-level delivery outcomes. Burst-sensitive and application-layer metrics such as consecutive cooperative awareness message loss periods, inter-reception time distributions as a function of inter-vehicle distance, and packet-level delivery ratios would provide a more direct assessment of whether the proposed estimator benefits safety-critical V2X applications. Coupling the estimator with system-level simulators that generate realistic traffic and interference would be a natural follow-up study.

7.5 Concluding remarks

Reliable channel estimation is an important enabler for vehicular communications. This thesis demonstrates that DL is a practical and effective alternative to classical estimation at the link level, provided that architectural choice, training data composition, and model interpretability are addressed in an integrated manner. The proposed DPA-TCN architecture, trained with mixed-channel and mixed-SNR data and optimised through the REACH interpretability framework, provides a solid link-level foundation for robust vehicular channel estimation. Establishing its fitness for safety-critical deployment requires additional validation at the packet and application layer, as well as on representative hardware, which we identify as priority directions for future work. We hope that the methodologies developed here will be useful beyond IEEE 802.11p and more broadly applicable in environments where accuracy and efficiency are required.

Note on nomenclature: The non-causal Data Pilot-Aided Temporal Convolutional Network (DPA-TCN) architecture introduced in this thesis is also referred to as the DPA-RDCNN in the follow-up publications arising from this work. Both designations describe exactly the same neural network design and processing pipeline.

References

[1] M. N. Tahir and M. Katz, "Performance Evaluation of IEEE 802.11p, LTE and 5G in Connected Vehicles for Cooperative Awareness," *Engineering Reports*, vol. 4, no. 4, e12467, 2022. DOI: 10.1002/eng2.12467.

[2] W. Anwar, N. Franchi, and G. Fettweis, "Physical Layer Evaluation of V2X Communications Technologies: 5G NR-V2X, LTE-V2X, IEEE 802.11bd, and IEEE 802.11p," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, IEEE, 2019, pp. 1–7. DOI: 10.1109/VTCFall.2019.8891313.

[3] Z. Hameed Mir and F. Filali, "LTE and IEEE 802.11p for Vehicular Networking: A Performance Evaluation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–15, 2014.

[4] A. F. Molisch, *Wireless Communications*. John Wiley & Sons, 2012, vol. 34.

[5] K. Zhong, X. Lei, B. Dong, and S. Li, "Channel Estimation in OFDM Systems Operating Under High Mobility Using Wiener Filter Combined Basis Expansion Model," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 186, 2012.

[6] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Deep Learning Based Channel Estimation Schemes for IEEE 802.11p Standard," *IEEE Access*, vol. 8, pp. 113 751–113 765, 2020. DOI: 10.1109/ACCESS.2020.3003286.

[7] J. Pan, H. Shan, R. Li, Y. Wu, W. Wu, and T. Q. S. Quek, "Channel Estimation Based on Deep Learning in Vehicle-to-Everything Environments," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1891–1895, 2021.

[8] A. K. Gizzini, M. Chafii, S. Ehsanfar, and R. M. Shubair, "Temporal Averaging LSTM-based Channel Estimation Scheme for IEEE 802.11p Standard," in *2021 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2021, pp. 1–7. DOI: 10.1109/GLOBECOM46510.2021.9685409.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] A. K. Gizzini, Y. Medjahdi, A. J. Ghandour, and L. Clavier, "Towards Explainable AI for Channel Estimation in Wireless Communications," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 5, pp. 7389–7394, 2024. DOI: 10.1109/TVT.2023.3345632.

[11] L. Cheng, B. Henty, F. Bai, and D. D. Stancil, "Doppler Spread and Coherence Time of Rural and Highway Vehicle-to-Vehicle Channels at 5.9 GHz," in *2008 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2008, pp. 1–6. DOI: 10.1109/GLOCOM.2008.ECP.802.

[12] G. Acosta-Marum and M. A. Ingram, "Six Time- and Frequency-Selective Empirical Channel Models for Vehicular Wireless LANs," in *2007 IEEE 66th Vehicular Technology Conference*, IEEE, 2007, pp. 2134–2138. DOI: 10.1109/VETECF.2007.448.

[13] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM Wireless Communications with MATLAB*. John Wiley & Sons, 2010.

[14] P. Bello, "Characterization of Randomly Time-Variant Linear Channels," *IEEE Transactions on Communications Systems*, vol. 11, no. 4, pp. 360–393, 1963.

[15] G. Acosta-Marum, "Measurement, Modeling, and OFDM Synchronization for the Wideband Mobile-to-Mobile Channel," Ph.D. dissertation, Georgia Institute of Technology, 2007.

[16] W. C. Jakes and D. C. Cox, *Microwave Mobile Communications*. Wiley-IEEE Press, 1994.

[17] A. K. Gizzini, "Advanced Linear and Deep Learning based Channel Estimation Techniques in Doubly Dispersive Environments," Ph.D. dissertation, Cergy Paris CY Université, 2021.

[18] C. F. Mecklenbräuker, A. F. Molisch, J. Karedal, *et al.*, "Vehicular Channel Characterization and Its Implications for Wireless System Design and Performance," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1189–1212, 2011.

[19] V2X Core Technical Committee, *On-Board System Requirements for V2V Safety Communications*, SAE International, 2020. DOI: 10.4271/J2945/1_202004.

[20] European Telecommunications Standards Institute, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions," ETSI, Tech. Rep. TR 102 638 V1.1.1, 2009.

[21] J. B. Kenney, "Dedicated Short-Range Communications (DSRC) Standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011. DOI: 10.1109/JPROC.2011.2132790.

[22] A. M. S. Abdelgader and W. Lenan, "The Physical Layer of the IEEE 802.11p WAVE Communication Standard: The Specifications and Challenges," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2014, pp. 22–24.

[23] S. A. Ahmed, S. H. Ariffin, and N. Fisal, "Overview of Wireless Access in Vehicular Environment (WAVE) Protocols and Standards," *Environment*, vol. 7, no. 8, 2013.

[24] Y. G. Li and G. L. Stuber, "Orthogonal Frequency Division Multiplexing for Wireless Communications," in *OFDM for Wireless Communications*, Springer, 2006. DOI: 10.1007/0-387-30235-4.

[25] T. Wang, J. G. Proakis, E. Masry, and J. R. Zeidler, "Performance Degradation of OFDM Systems Due to Doppler Spreading," *IEEE Transactions on Wireless Communications*, vol. 5, no. 6, pp. 1422–1432, 2006. DOI: 10.1109/TWC.2006.1638663.

[26] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965. DOI: 10.2307/2003354.

[27] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hill, 2008.

[28] IEEE 802 LAN/MAN Standards Committee, *IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE, 2009.

[29] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "Channel Estimation Techniques Based on Pilot Arrangement in OFDM Systems," *IEEE Transactions on Broadcasting*, vol. 48, no. 3, pp. 223–229, 2002.

[30] M.-H. Hsieh and C.-H. Wei, "Channel Estimation for OFDM Systems Based on Comb-Type Pilot Arrangement in Frequency Selective Fading Channels," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 217–225, 1998.

[31] Y. G. Li, "Pilot-Symbol-Aided Channel Estimation for OFDM in Wireless Systems," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 4, pp. 1207–1215, 2000.

[32] M. K. Ozdemir and H. Arslan, "Channel Estimation for Wireless OFDM Systems," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 2, pp. 18–48, 2007.

[33] O. Edfors, M. Sandell, J.-J. Van de Beek, S. K. Wilson, and P. O. Borjesson, "OFDM Channel Estimation by Singular Value Decomposition," *IEEE Transactions on Communications*, vol. 46, no. 7, pp. 931–939, 1998.

[34] M. Morelli, C.-C. J. Kuo, and M.-O. Pun, "Synchronization Techniques for Orthogonal Frequency Division Multiple Access (OFDMA): A Tutorial Review," *Proceedings of the IEEE*, vol. 95, no. 7, pp. 1394–1427, 2007.

[35] Y. Zhao and A. Huang, "A Novel Channel Estimation Method for OFDM Mobile Communication Systems Based on Pilot Signals and Transform-Domain Processing," *IEEE Transactions on Vehicular Technology*, vol. 47, no. 8, pp. 2089–2096, 1997.

[36] J. A. Fernandez, K. Borries, L. Cheng, B. V. K. Vijaya Kumar, D. D. Stancil, and F. Bai, "Performance of the 802.11p Physical Layer in Vehicle-to-Vehicle Environments," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 3–14, 2012. DOI: 10.1109/TVT.2011.2164428.

[37] Z. Zhao, X. Cheng, M. Wen, B. Jiao, and C.-X. Wang, "Channel Estimation Schemes for IEEE 802.11p Standard," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 38–49, 2013. DOI: 10.1109/MITS.2013.2270032.

[38] Y.-K. Kim, J.-M. Oh, Y.-H. Shin, and C. Mun, "Time and Frequency Domain Channel Estimation Scheme for IEEE 802.11p," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 1085–1090. DOI: 10.1109/ITSC.2014.6957832.

[39] R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, 1981. DOI: 10.1109/TASSP.1981.1163711.

[40] A. K. Gizzini and M. Chafii, "A Survey on Deep Learning Based Channel Estimation in Doubly Dispersive Environments," *IEEE Access*, vol. 10, pp. 70 595–70 619, 2022.

[41] H. Ye, G. Y. Li, and B.-H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, 2018. DOI: 10.1109/LWC.2017.2757490.

[42] C. Nwankpa, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," in *Proceedings of the 2nd International Conference on Computational Sciences and Technologies (INCCST 20)*, 2020.

[43] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, PMLR, 2011, pp. 315–323.

[44] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent Is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[45] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation Applied to Handwritten ZIP Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[46] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[47] J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[48] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[49] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv preprint arXiv: 1803.01271*, 2018.

[50] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "WaveNet: A Generative Model for Raw Audio," in *9th ISCA Workshop on Speech Synthesis*, 2016.

[51] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," in *International Conference on Learning Representations (ICLR)*, 2016.

[52] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[53] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4.

[54] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[56] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[57] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *International Conference on Learning Representations*, 2017.

[58] A. Krogh and J. Hertz, "A Simple Weight Decay Can Improve Generalization," *Advances in Neural Information Processing Systems*, vol. 4, 1991.

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[60] L. Prechelt, "Early Stopping – But When?" In *Neural Networks: Tricks of the Trade*, Springer, 2002, pp. 55–69.

[61] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, PMLR, 2015, pp. 448–456.

[62] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[63] Y. Wu and K. He, "Group Normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[64] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How Does Batch Normalization Help Optimization?" *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[65] M. Soltani, V. Pourahmadi, A. Mirzaei, and H. Sheikhzadeh, "Deep Learning-Based Channel Estimation," *IEEE Communications Letters*, vol. 23, no. 4, pp. 652–655, 2019.

[66] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Joint TRFI and Deep Learning for Vehicular Channel Estimation," in *2020 IEEE Global Communications Conference (GLOBECOM) Workshops*, IEEE, 2020, pp. 1–6. DOI: 10.1109/GCWkshps50303.2020.9367412.

[67] S. Han, Y. Oh, and C. Song, "A Deep Learning Based Channel Estimation Scheme for IEEE 802.11p Systems," in *2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–6.

[68] A. K. Gizzini and M. Chafii, "RNN Based Channel Estimation in Doubly Selective Environments," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1–18, 2024. DOI: 10.1109/TMLCN.2023.3332021.

[69] J. Hou, H. Liu, Y. Zhang, W. Wang, and J. Wang, "GRU-Based Deep Learning Channel Estimation Scheme for the IEEE 802.11p Standard," *IEEE Wireless Communications Letters*, vol. 12, no. 5, pp. 764–768, 2023. DOI: 10.1109/LWC.2022.3187110.

[70] Z. Chen, F. Gu, and R. Jiang, "Channel Estimation Method Based on Transformer in High Dynamic Environment," in *2020 International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, 2020, pp. 817–822. DOI: 10.1109/WCSP49889.2020.9299821.

[71] D. Luan and J. S. Thompson, "Channelformer: Attention Based Neural Solution for Wireless Channel Estimation and Effective Online Training," *IEEE Transactions on Wireless Communications*, vol. 22, no. 10, pp. 6562–6577, 2023. DOI: 10.1109/TWC.2023.3244484.

[72] F. Liu, J. Zhang, P. Jiang, C.-K. Wen, and S. Jin, "CE-ViT: A Robust Channel Estimator Based on Vision Transformer for OFDM Systems," in *2023 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2023, pp. 4798–4803. DOI: 10.1109/GLOBECOM54140.2023.10436847.

[73] G. Montavon, W. Samek, and K.-R. Müller, "Methods for Interpreting and Understanding Deep Neural Networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018. DOI: 10.1016/j.dsp.2017.10.011.

[74] Z. C. Lipton, "The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[75] C. Molnar, *Interpretable Machine Learning*. Lulu, 2020.

[76] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[77] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv preprint arXiv:1312.6034*, 2013.

[78] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: Removing Noise by Adding Noise," *arXiv preprint arXiv:1706.03825*, 2017.

[79] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning Important Features Through Propagating Activation Differences," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3145–3153.

[80] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards Better Understanding of Gradient-Based Attribution Methods for Deep Neural Networks," in *International Conference on Learning Representations*, 2018.

[81] Y. Wang, T. Zhang, X. Guo, and Z. Shen, *Gradient Based Feature Attribution in Explainable AI: A Technical Review*, 2024. arXiv: 2403.10415 [cs.AI].

[82] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *European Conference on Computer Vision*, Springer, 2014, pp. 818–833.

[83] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the Visualization of What a Deep Neural Network Has Learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, 2016.

[84] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," *PLOS ONE*, vol. 10, no. 7, e0130140, 2015. DOI: 10.1371/journal.pone.0130140.

[85] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-Wise Relevance Propagation: An Overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Springer, 2019, pp. 193–209. DOI: 10.1007/978-3-030-28954-6_10.

[86] A. K. Gizzini, Y. Medjahdi, and M. B. Mabrouk, "GRACE: Gradient-based XAI Scheme for Channel Estimation in Wireless Communications," in *2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, IEEE, 2024, pp. 572–577. DOI: 10.1109/MeditCom61057.2024.10621232.

[87] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[88] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[89] R. Negi and J. Cioffi, "Pilot Tone Selection for Channel Estimation in a Mobile OFDM System," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 1122–1128, 1998.

[90] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[91] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning Convolutional Neural Networks for Resource Efficient Inference," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[92] Y. He, X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[93] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[94] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical Guidelines for Efficient CNN Architecture Design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[95] A. Canziani, A. Paszke, and E. Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications," *arXiv preprint arXiv:1605.07678*, 2016.

[96] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[97] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-Generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701.

[98] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[99] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[100] J. Y. Choi and B. Lee, "Combining LSTM Network Ensemble via Adaptive Weighting for Improved Time Series Forecasting," *Mathematical Problems in Engineering*, vol. 2018, Art. no. 2470171, 2018. DOI: 10.1155/2018/2470171.

[101] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Enhancing Least Square Channel Estimation Using Deep Learning," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, IEEE, 2020, pp. 1–5.

[102] S. A. Ngorima, A. S. J. Helberg, and M. H. Davel, "Neural Network-Based Vehicular Channel Estimation Performance: Effect of Noise in the Training Set," in *Artificial Intelligence Research*, Springer, 2025, pp. 192–206.

[103] S. A. Ngorima, A. Helberg, and M. Davel, "Simplified Temporal Convolutional-based Channel Estimation for a WiFi Vehicular Communication Channel," in *2025 IEEE 3rd Wireless Africa Conference (WAC)*, IEEE, 2025, pp. 1–7.

[104] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

Supplemental content

A.1 Appendix: Chapter 4

Dual-cell LSTM: Inference Algorithms

Algorithm 1 Test-time Estimation with Dual-Cell LSTM + DPA + (Optional) TA

Require: Trained Θ_f, Θ_b; received grid $\{Y_n[k]\}$ for n=1:T; TA weight α (default α=2)

Ensure: $\{\widehat{H}_n^{\mathrm{TA}}[k]\}_{n=1}^{T}$ on $k \in \mathcal{K}_d$; metrics (e.g., NMSE, BER)

1: Initialise from preambles: $\widehat{H}^{\mathrm{LS}}[k]$ for $k \in \mathcal{K}_{\mathrm{act}}$

2: Set $\widehat{H}_0^{\mathrm{RNN}}[k] \leftarrow \widehat{H}^{\mathrm{LS}}[k], \quad \widehat{H}_0^{\mathrm{TA}}[k] \leftarrow \widehat{H}^{\mathrm{LS}}[k]$

3: **for** $n = 1$ to T **do**

4: **Input construction (active subcarriers):**

$$\bar{H}_n[k] \leftarrow \begin{cases} \widehat{H}_{n-1}^{\mathrm{TA}}[k], & k \in \mathcal{K}_d \\ \widehat{H}^{\mathrm{LS}}[k], & k \in \mathcal{K}_p \end{cases}, \quad \mathbf{x}_n \leftarrow f_{\mathbb{R}}\big(\{\bar{H}_n[k]\}_{k\in\mathcal{K}_{\mathrm{act}}}\big)$$

5: **Dual-LSTM mapping and fusion:**

$$\widehat{\mathbf{H}}_n^{(f)} \leftarrow \Omega_f\big(\psi_f(\mathbf{s}_{n-1}^f, \mathbf{x}_n; \Theta_f)\big); \quad \widehat{\mathbf{H}}_n^{(b)} \leftarrow \Omega_b\big(\psi_b(\mathbf{s}_{n-1}^b, \mathbf{x}_n; \Theta_b)\big)$$

$$\widehat{\mathbf{H}}_n^{\mathrm{RNN}} \leftarrow \tfrac{1}{2}\big(\widehat{\mathbf{H}}_n^{(f)} + \widehat{\mathbf{H}}_n^{(b)}\big)$$

6: **DPA refinement (data subcarriers, causal):**

$$\widehat{H}_n^{\mathrm{eq}}[k] = \frac{Y_n[k]}{\widehat{H}_{n-1}^{\mathrm{RNN}}[k]}; \quad \widehat{D}_n[k] = \mathcal{Q}\big(\widehat{H}_n^{\mathrm{eq}}[k]\big); \quad \widehat{H}_n^{\mathrm{DPA}}[k] = \frac{Y_n[k]}{\widehat{D}_n[k]}, \quad k \in \mathcal{K}_d$$

7: **TA smoothing (optional, data subcarriers):**

$$\widehat{H}_n^{\mathrm{TA}}[k] = \big(1 - \tfrac{1}{\alpha}\big)\, \widehat{H}_{n-1}^{\mathrm{TA}}[k] + \tfrac{1}{\alpha}\widehat{H}_n^{\mathrm{DPA}}[k], \quad k \in \mathcal{K}_d$$

8: **Feedback to next step:** set $\widehat{H}_n^{\mathrm{RNN}}[k] \leftarrow \widehat{H}_n^{\mathrm{TA}}[k]$ (if TA used) **or** $\widehat{H}_n^{\mathrm{RNN}}[k] \leftarrow \widehat{H}_n^{\mathrm{DPA}}[k]$

9: **end for**

10: Compute NMSE and BER from $\{\widehat{H}_n^{\mathrm{TA}}\}_{n=1}^{T}$ (or $\{\widehat{H}_n^{\mathrm{DPA}}\}$ if TA disabled)

A.2 Appendix: Chapter 5

Hyperparameter Optimisation Results

Table A.1 presents the complete hyperparameter search spaces and optimal values for all architectures evaluated in Chapter 5 under both mixed SNR and high SNR training.

Table A.1: Optimised hyperparameters for evaluated architectures under mixed SNR and high SNR training strategies.

Hyperparameter	**Search Space**	**Mixed SNR**	**40 dB**
TCN-DPA			
Learning rate	$[10^{-5}, 10^{-2}]$	0.0005	0.001
Number of Blocks	$[1, 5]$	4	4
Hidden Channels	N/A	100	100
Kernel Size	$[2, 5]$	4	2
Dilation Depth	N/A	4	4
Dropout	$[10^{-5}, 0.5]$	0.05	0.05
StepLR step size	$[10, 50]$	11	N/A
StepLR γ	$[0.5, 1]$	0.948	N/A
CosineAnnealingWarmRestarts T_0	$[10, 200]$	N/A	50
CosineAnnealingWarmRestarts T_{mult}	$\{1, 2, 3, 4\}$	N/A	2
CosineAnnealingWarmRestarts $\eta_{\min}$	$[10^{-8}, 10^{-4}]$	N/A	10^{-6}
Epochs	$[0, 200]$	E Stopped	E Stopped
STA-DNN			
Learning rate	[1e-5 to 1e-2]	0.001	0.001
Number of layers	[1 to 5]	2	3
Size of hidden layer 0	[5 to 30]	29	15
Size of hidden layer 1	[5 to 30]	27	15
Size of hidden layer 2	[5 to 30]	N/A	15
Total training epochs	[50 to 500]	133	300

Continued on next page

Table A.1 – Continued from previous page

Hyperparameter	Search Space	Mixed SNR	40 dB
TRFI-DNN			
Learning rate	[1e-5 to 1e-2]	0.0004	0.001
Number of layers	[1 to 5]	3	3
Size of hidden layer 0	[5 to 30]	23	15
Size of hidden layer 1	[5 to 30]	29	15
Size of hidden layer 2	[5 to 30]	21	15
Total training epochs	[50 to 500]	130	160
LSTM-DPA-TA			
Learning rate	[1e-5 to 1e-1]	0.004	0.001
LSTM size	[64 to 128]	128	128
StepLR step size	[1 to 50]	35	10
StepLR gamma	[0.1 to 1]	0.7	0.8
Training epochs	[50 to 500]	160	500
Batch Size	N/A	128	128
GRU-DPA-TA			
Learning rate	[1e-5 to 1e-1]	0.004	0.001
GRU size	[64 to 128]	48	48
StepLR step size	[1 to 50]	21	10
StepLR γ	[0.1 to 1]	0.9	0.8
Training epochs	[50 to 500]	150	500
Batch Size	N/A	128	128

A.3 Published Papers

A Data Pilot-Aided Temporal Convolutional Network for Channel Estimation in IEEE 802.11p Vehicle-to-Vehicle Communications

Simbarashe Aldrin Ngorima[1,2,3], Albert Helberg[1], Marelie H. Davel[1,2,3]
[1]*Faculty of Engineering, North-West University, South Africa*
[2]*Centre for Artificial Intelligence Research, South Africa*
[3]*National Institute for Theoretical and Computational Sciences, South Africa*
aldringorima@gmail.com
albert.helberg@nwu.ac.za
marelie.davel@nwu.ac.za

***Abstract*—In modern communication systems, having an accurate channel estimator is crucial. However, when there is mobility, it becomes difficult to estimate the channel and the pilot signals, which are used for channel estimation, become insufficient. In this paper, we introduce the use of Temporal Convolutional Networks (TCNs) with data pilot-aided (DPA) channel estimation and temporal averaging (TA) to estimate vehicle-to-vehicle same direction with Wall (VTV-SDWW) channels. The TCN-DPA-TA estimator showed an improvement in Bit Error Rate (BER) performance of up to 1 order of magnitude. Furthermore, the BER performance of the TCN-DPA without TA also improved by up to 0.7 magnitude compared to the best classical estimator.**

***Index Terms*—channel estimation, deep learning, TCN, vehicle-to-vehicle, wireless communications, IEEE 802.11p**

I. Introduction

Beyond 5G networks and the modern Wi-Fi standard used for vehicular communication systems are expected to deliver high data speeds and low latency [1]. However, mobility in wireless channels can cause doubly selective fading, where the channel changes both in the temporal and frequency domains. This is mainly due to multipath propagation, which results in varying delays, Doppler shifts, and attenuation as users move. It is important to accurately estimate the channel because doubly selective fading affects receiver processes such as equalisation, demodulation, and decoding, which directly influences system performance. For reliable communication in mobile scenarios, accurate channel estimation algorithms are vital to handle the doubly selective fading nature of wireless channels.

Pilot signals are usually used to estimate the channel, but in mobility scenarios these pilots become insufficient [2]. To address this, a data pilot-aided (DPA) estimation scheme is generally used, in which data subcarriers are remapped and used for channel estimation [3]. Several classical methods are then built upon DPA. This includes (1) the spectral temporal averaging (STA) scheme [4], where the averaging is performed in the time and frequency domains, (2) the constructed data pilot (CDP) scheme [5] in which the previous symbols are used as preambles to estimate the current symbol, and (3) time domain reliability test frequency domain interpolation (TRFI) [6], where a reliability test is performed on the DPA estimated channel to obtain reliable subcarriers. The reliable channel estimates are interpolated at the unreliable subcarriers.

The process of demapping the data subcarriers in DPA to obtain an initial channel estimate is heavily influenced by the noise level and the accuracy of the previously estimated channel [7]. As the previous data symbols are utilised as the preamble to obtain the current symbol, the error in the DPA increases as we progress from one symbol to the next across the frame, leading to performance degradation in vehicular environments. Given that DPA is the initial step in most classical methods, this error is also present in classical channel estimation methods.

In this work, we propose the use of TCN-based estimators to reduce DPA errors and improve performance in vehicular channel environments. Using a TCN for initial channel estimation, we can achieve more accurate results, which can then be utilised effectively by the DPA method to estimate the current channel state as a post-processing step. In addition, we also test the effectiveness of temporal averaging (TA) in further improving the channel estimate.

The remainder of the paper is structured as follows: Section II provides background information on the IEEE 802.11p standard investigated in this paper. Section II-B offers a review of current channel estimation schemes, while Section III introduces the TCN-based estimator. In Section IV, the results are presented and analysed. Section V concludes the paper.

II. Background

This section summarises the IEEE 802.11p standard specification and examines channel estimation system models and techniques in the context of IEEE 802.11p.

A. System Model

The IEEE 802.11p protocol uses Orthogonal Frequency Division Multiplexing (OFDM) to facilitate data transmission

across radio channels. OFDM divides the available bandwidth (10MHz) into subcarrier frequencies, allowing multiple signals to be sent at the same time [8]. Pilot signals on designated subcarriers track channel conditions to enhance data decoding for efficient communication. Only 52 of the 64 available subcarriers are used for data transmission and pilot symbols, while the remaining subcarriers are used for guard bands and direct current (DC) offset. Fig. 1 shows how these subcarrier indices are arranged in the IEEE 802.11p frame. The four pilot subcarriers are positioned at indices $\{-21, -7, 7, 21\}$, while the remaining 48 indices are designated for data.

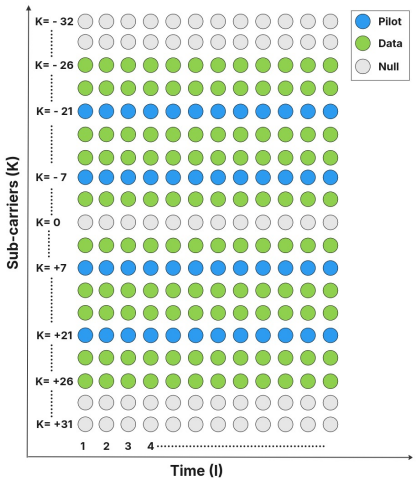


Fig. 1: IEEE 802.11p subcarrier arrangement [9].

In this study, we assume perfect synchronisation and consider a frame structure that begins with two extended preambles, followed by *I* data symbols. The received OFDM symbol $y_i[k]$ at time instant *i* and subcarrier *k* can be represented as in [9]:

$$y_i[k] = h_i[k]x_i[k] + n_i[k], \tag{1}$$

where $x_i[k]$ is the transmitted constellation symbol, $h_i[k]$ is the channel frequency response, $n_i[k]$ is the noise added to the *i-th* OFDM symbol on the *k-th* subcarrier.

B. IEEE 802.11p Channel Estimation Methods

In this section, we provide an overview of the IEEE 802.11p channel estimation methods. All the methods discussed will be evaluated against our proposed approach.

1) LS Estimation: The least squares (LS) estimation method is the widely accepted channel estimation technique in wireless communications to estimate the channel response. It is also a standard method used in IEEE 802.11p. LS utilises the two received preambles as follows [2]:

$$\hat{h}_{\text{LS}}[k] = \frac{y_1^{(p)}[k] + y_2^{(p)}[k]}{2p[k]}, \tag{2}$$

where $y_1^{(p)}[k]$ and $y_2^{(p)}[k]$ represent the received preambles located at the beginning of each frame at subcarrier *k* and $p[k]$ is the predefined preamble that was transmitted.

2) DPA Estimation: In vehicular communications, DPA is considered the initial channel estimation method in vehicular communications because it is assumed that there is a high correlation between the successive OFDM symbols received [9]. In DPA, the LS estimation at the preambles (as described by (2)) is used to obtain the initial channel estimate; the subsequent channel estimate is obtained by equalising the received symbols with the initial channel estimate as described by (4). Equalised symbols are demapped to the closest constellation as described by (5) and the final DPA channel estimate is updated as expressed in (7). According to [2], the remaining OFDM symbols are equalised using the previously estimated channel:

$$y_i^{\text{eq}}[k] = \frac{y_i[k]}{\hat{h}_{i-1}^{\text{DPA}}[k]}, \tag{3}$$

$$\hat{h}_0^{\text{DPA}}[k] = \hat{h}_{\text{LS}}[k], \tag{4}$$

As outlined in [9], the data subcarrier $y_i^{eq}[k]$ is then demapped and mapped to the nearest constellation point using the respective modulation order to determine its closest constellation point $d_i[k]$ which is given by

$$d_i[k] = \Re\left(\frac{y_i[k]}{\hat{h}_{i-1}^{\text{DPA}}[k]}\right), \tag{5}$$

where $\Re(.)$ is a mapping operation to obtain the closest constellation point. The initial channel estimate $\hat{h}_0^{\text{DPA}}[k]$ is derived from the LS estimation, expressed as

$$\hat{h}_0^{\text{DPA}}[k] = \hat{h}_{\text{LS}}[k], \tag{6}$$

and the final DPA channel estimate is subsequently updated as

$$\hat{h}_i^{\text{DPA}}[k] = \frac{y_i[k]}{d_i[k]}. \tag{7}$$

3) STA Estimation: The STA method continuously updates the channel estimate using data symbols, leveraging time and frequency correlation across OFDM symbols. Following [9] the STA channel estimate is obtained by performing all the steps of the DPA procedure, followed by frequency domain averaging of the estimates for all selected subcarriers as described below:

$$\hat{h}_i^{\text{FD}}[k] = \sum_{\lambda=-\beta}^{\beta} \omega_\lambda \hat{h}_i^{\text{DPA}}[k+\lambda], \tag{8}$$

$$\omega_\lambda = \frac{1}{2\beta+1}, \tag{9}$$

where $2\beta + 1$ denotes the number of subcarriers being averaged. Finally, time averaging is performed to calculate the final STA channel estimate as

$$\hat{h}_i^{\text{STA}}[k] = \left(1 - \frac{1}{\alpha}\right)\hat{h}_{i-1}^{\text{STA}}[k] + \frac{1}{\alpha}\hat{h}_i^{\text{FD}}[k] \tag{10}$$

where α is a filtering parameter that varies depending on the characteristics of the channels.

4) CDP Estimation: The CDP method also relies on the high correlation of OFDM symbols. CDP performs all DPA procedures. Then, as described in [9], the previously received symbol is equalised using both the current DPA estimate, $\hat{h}_i^{\text{DPA}}[k]$, and the previous CDP estimate, $\hat{h}_{i-1}^{\text{CDP}}[k]$ as

$$y_{i-1}^{\text{eq}\prime}[k] = \frac{y_{i-1}[k]}{\hat{h}_i^{\text{DPA}}[k]}, \tag{11}$$

$$y_{i-1}^{\text{eq}\prime\prime}[k] = \frac{y_{i-1}[k]}{\hat{h}_{i-1}^{\text{CDP}}[k]}, \tag{12}$$

The equalised symbols $y_{i-1}^{\text{eq}\prime}[k]$ and $y_{i-1}^{\text{eq}\prime\prime}[k]$ are then demapped to their closest constellation $d'_{i-1}[k]$ and $d''_{i-1}[k]$ using the mapping operation $\mathfrak{R}(.)$. As outlined in [9], the final CDP channel estimate $\hat{h}_i^{\text{CDP}}[k]$ is obtained as follows

$$\hat{h}_i^{\text{CDP}}[k] = \begin{cases} \hat{h}_{i-1}^{\text{CDP}}[k], & d'_{i-1}[k] \neq d''_{i-1}[k], \\ \hat{h}_i^{\text{DPA}}[k], & d'_{i-1}[k] = d''_{i-1}[k] \end{cases} \tag{13}$$

If $d'_{i-1}[k] \neq d''_{i-1}[k]$, it indicates that $\hat{h}_i^{\text{DPA}}[k]$ is not reliable and therefore the previous channel estimate should be used instead.

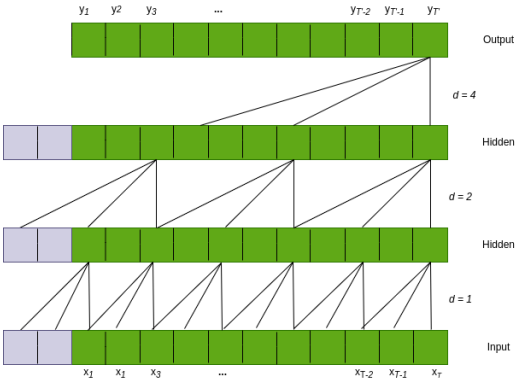
5) TRFI Estimation: The TFRI method is a channel estimation technique designed to enhance the performance of CDP. The steps involved in the TFRI process are similar to the CDP estimator processes. The main difference is that, instead of the comparison step described by (13), TFRI uses the demapped symbols $d'_{i-1}[k]$ and $d''_{i-1}[k]$ calculated after the processes expressed by (11), and (12) to create a reliable subcarrier set and an unreliable set. Then it uses the reliable set to interpolate the channel at unreliable subcarriers based on a reliable test as detailed by the authors in [9].

III. Proposed TCN-DPA Estimator

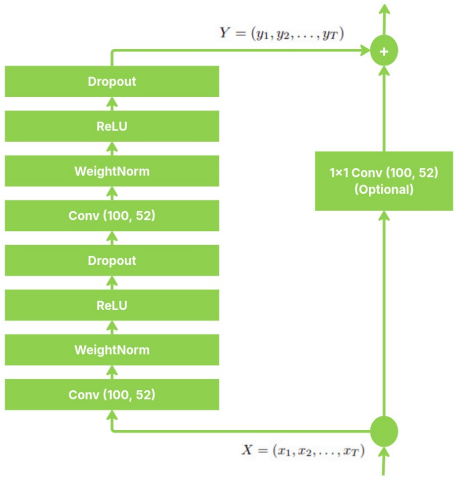
TCNs are designed to handle sequential data, whether it is single-channel or multiple-channel sequential data, making them ideal for capturing temporal dependencies in the received signal. Using 1-dimensional convolutional layers, TCNs can efficiently process sequential data, identifying patterns that classical methods may miss. This section outlines the architecture of the TCN and the proposed TCN-based estimator used in this work.

A. Temporal Convolutional Networks (TCNs)

Convolutional Neural Networks (CNNs), introduced by LeCun [10], are key models in the development of neural networks (NNs). CNNs use convolutional and pooling layers to extract local features and reduce dimensionality. TCNs were developed from CNNs, specifically to handle sequential data. TCNs integrate specialised components such as causal convolution, dilated convolution, and residual connections, as shown in Fig. 2 [11], making them effective in capturing temporal dependencies within the input data. A TCN uses one-dimensional convolution kernels (kernels that move in only one direction, for example from left to right) to analyse sequential data in the temporal domain, as shown in Fig. 2a.



(a) Causal dilated convolutions.



(b) TCN Residual block adapted from [11].

Fig. 2: TCN architectural elements. (a) Causal dilated convolutions and (b) TCN residual block with an optional 1x1 residual connection that can be added when the input and output have different dimensions.

Suppose that we have an input sequence $\{x_0,\ldots,x_T\}$ and our goal is to forecast the corresponding outputs $\{y_0,\ldots,y_T\}$ at each time step. The main constraint is that when predicting the output y_t at any given time t, we can only use the inputs that have been observed up to that point: $\{x_0,\ldots, x_t\}$. To ensure that the output is of the same length as the input, zero padding of length (kernel size - 1) is applied to the hidden layers. A TCN uses causal convolutions to ensure that there is no future information leakage. For causality to be true, y_T should depend only on $\{x_0,\ldots,x_T\}$ and not on any future input beyond time T. A crucial component of a TCN is the residual block [11]. It incorporates a branch that takes the input sequence and applies a series of transformations, denoted as F. When the outputs have dimensions different from the input, a residual branch is present where the outputs are then combined with the original input x through an activation function, as:

$$\text{Output} = \text{Activation}(x + F(x)) \tag{14}$$

The residual connection allows the layers to focus on capturing the desired mapping instead of learning the entire transformation from input to output. The residual block typically contains dilated causal convolution layers and non-linearity layers. One commonly used non-linearity is the Rectified Linear Unit (ReLU), which enhances network sparsity and helps prevent overfitting by setting a portion of the neuron's output to zero. The ReLU function also addresses the issue of gradient vanishing during backpropagation. Its mathematical definition is given as:

$$\mathrm{f}(x, w, b) = \max\left(0, w^T x + b\right) \tag{15}$$

where x represents the input to the neuron, w denotes the weight parameter, and b signifies the bias term. In the residual block, the dropout layer is applied during training after each convolutional layer to prevent overfitting and improve the model's generalisation capabilities.

B. Dataset Generation

This section discusses the characteristics of the generated dataset, the vehicular channel model used, and the structure of the dataset for use by a TCN.

1) Channel Model: The vehicular channel model investigated in this work is specified in [12]. In this study, we investigate the vehicle-to-vehicle same direction with wall (VTV-SDWW) tapped delay line (TDL) vehicular channel model. This model is used for communication between two vehicles moving in the same direction with a central wall separating them while keeping a distance of 300-400 metres between them. A VTV-SDWW TDL mobility scenario is used in which vehicles have a velocity of 100 km/h and a Doppler shift of f_d = 550 Hz. 16QAM modulation is used.

Our dataset consists of 18,000 time-specific samples or frames. Each sample includes a sent and received version of 50 OFDM symbols. Each OFDM symbol consists of a complex value for each of the 52 active subcarriers, where 48 are data subcarriers and 4 are pilot subcarriers. To prepare the data for the NN, we separate the complex-valued received 16 QAM symbol per resource element and concatenate them as shown in Fig. 3, such that the values are interleaved per subcarrier and time slot. This appears as if the number of symbols received per carrier has doubled. Each input sample is now a matrix of real values with dimensions of 52×100 and each output is a matrix of real values with dimensions of 48×100 where the pilot signals are excluded. These samples are grouped into batches.

The input tensor for the TCN is a 3D tensor with dimensions (N, C, T) where N represents the batch size, C represents the number of channels (100 separated and interleaved real and imaginary parts of the complex data) and T represents the number of 'time steps' (in this case, not actual time, but the 52 subcarriers are modelled as if sequential). In our case, with a batch size of 128, the input tensor shape is $(128, 100, 52)$. A kernel size of 2 indicates the length of the convolutional filter. This filter processes two consecutive time steps across

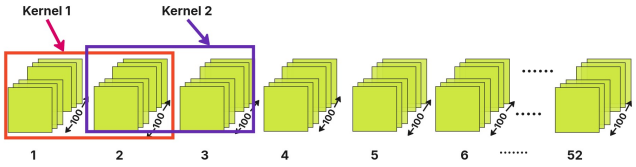


Fig. 3: TCN operation with an input sequence of 52 subcarriers and 100 channels of interleaved real and imaginary parts of the complex data.

100 channels to calculate weighted sums of the input values it overlaps.

The NN is trained at the 40 dB SNR level. This is based on the hypothesis of the authors in [2], [9], [13] that training a NN on a high SNR dataset facilitates the NN to learn the actual channel state and this is important for the NN.

C. TCN-DPA estimator

This section describes the proposed TCN-DPA estimator where the initial input to the TCN is the DPA estimate in the frequency domain (the subcarriers). The DPA process is performed after this TCN process. The previous TCN output, $\hat{h}_{i-1}^{\mathrm{TCN}}[k]$, is used by the DPA process to equalise the current received symbol. The equalised symbol is then mapped to the closest constellation point, acquiring the demapped symbol as expressed in (17). The final channel estimate $\hat{h}_i^{\mathrm{TCN}}[k]$ is updated as:

$$d_i^{\mathrm{TCN}}[k] = \Re\left(\frac{y_i[k]}{\hat{h}_{i-1}^{\mathrm{TCN}}[k]}\right), \tag{16}$$

$$\hat{h}_0^{\mathrm{TCN}}[k] = \hat{h}_{\mathrm{DPA}}[k], \tag{17}$$

$$\hat{h}_i^{\mathrm{TCN}}[k] = \frac{y_i[k]}{d_i^{\mathrm{TCN}}[k]} \tag{18}$$

Fig. 4 shows this iterative process. The next section gives a

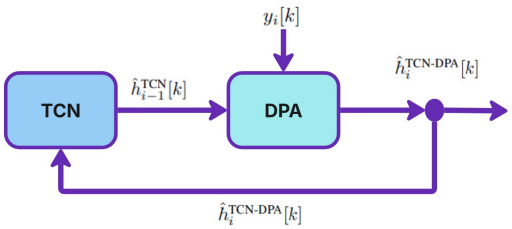


Fig. 4: TCN-DPA estimator process showing how the TCN output is used for the DPA process.

description of the TA process and how it can be added to the TCN-DPA method.

1) TA Processing: Temporal averaging (TA) is a method for noise reduction in which the noise reduction ratio can be determined analytically. As stated in [2], TA can iteratively decrease the Additive White Gaussian Noise (AWGN) power within a frame. In that study, an α value of 2 yielded

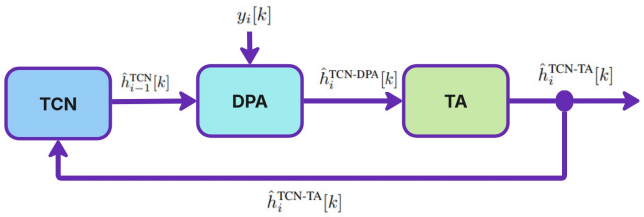


Fig. 5: TCN-DPA-TA estimator process showing the TCN output as input to the DPA and the TA processes and how the TCN-DPA-TA output is used to update the final channel estimate.

commendable results. For simplicity, we also use an α of 2 in this work. The complete TCN-DPA-TA is shown in Fig. 5.

The TA process is applied to the estimated TCN-DPA channel $\hat{h}_i^{\text{TCN-DPA}}[k]$ so that the final channel estimate $\hat{h}_i^{\text{TCN-TA}}[k]$ is obtained as:

$$\hat{h}_i^{\text{TCN-TA}}[k] = \left(1 - \frac{1}{\alpha}\right) \hat{h}_{i-1}^{\text{TCN-TA}}[k] + \frac{1}{\alpha} \hat{h}_i^{\text{TCN-DPA}}[k] \quad (19)$$

where α represents the weight size used to calculate the averages. A higher value of α means that more symbols are used.

2) TCN Hyperparameter Tuning: In this work, we use Bayesian optimisation (BO) to find the best hyperparameter combination to improve model performance. BO constructs a probabilistic model, typically a Gaussian process, to approximate the function based on observed data. Gaussian processes are preferred as surrogate models due to their ability to provide both mean predictions and uncertainty estimates for input parameters [14]. We optimise the learning rate, number of layers, kernel size, dropout, and StepLR parameters (step size and gamma). For hyperparameter tuning, we use Optuna [15] and Weights & Biases (WandB)[1]. Optuna, a BO framework, uses a tree-structured parzen estimator (TPE) to maintain a probabilistic model linking hyperparameters to outcomes. This approach enables efficient exploration of the hyperparameter space, refining it sequentially to improve the probability of reaching the global optimum in fewer trials. WandB is used for logging and monitoring experimental results.

We conduct 80 trials to explore the hyperparameter space comprehensively. Each trial runs for 200 epochs to allow the models to converge and for the effects of hyperparameter changes to become evident. The validation loss is used as an indicator of convergence and the optimal hyperparameters are selected based on the lowest validation loss observed during these trials. A search space for the hyperparameters is defined as shown in Table I. The best hyperparameter combination is obtained and is also shown in the same table.

IV. Analysis and Results

The TCN-DPA architecture consists of a 4-layer TCN residual block with a kernel size of 2 and layer dilations of (1, 2). 12,000 frames are used for training, 4,000 for validation, and 2,000 are used for testing.

[1]https://wandb.ai/

TABLE I: Optimal TCN Hyperparameters.

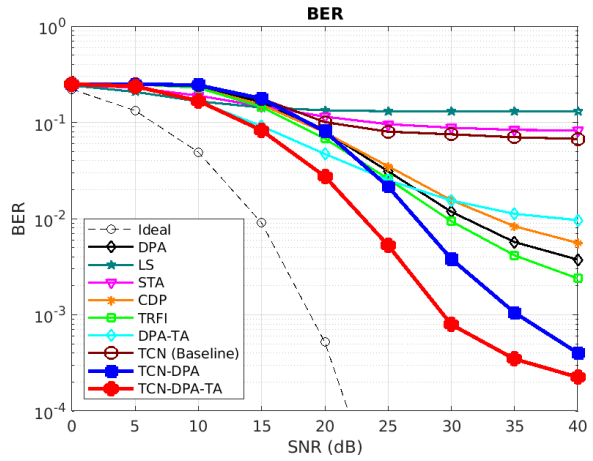
Hyperparameter	Search Space	Optimal Value
Learning Rate	1×10^{-5} to 1×10^{-2}	0.003
Number of Layers	1 to 5	4
Kernel Size	2 to 5	2
Dropout	10^{-5} to 0.5	0.01
StepLR Step Size	10 to 50	17
StepLR Gamma	0.5 to 1	0.8
Epochs	0 to 200	100

A. Bit Error Rate (BER)

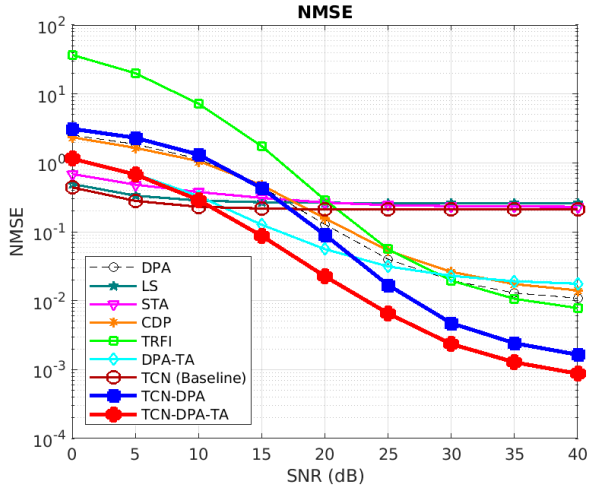
Figure 6a shows the BER of the proposed TCN-DPA with and without TA compared to the classical methods, which are DPA, LS, STA, CDP, TRFI and DPA-TA. We also compare the performance of a TCN without any post-processing which is our baseline TCN to determine the need for DPA as a post-processing method. The baseline TCN performance is poor compared to other estimators. The TCN-based estimators clearly showed better performance than the classical estimators. It is observable that from SNR larger than 20 dB the TCN-DPA estimator starts to outperform the classical estimators, and at 40 dB TCN-DPA obtained BER performance close to 0.7 orders of magnitude lower than TRFI. TCN-DPA-TA outperformed all estimators in all SNRs. At 40 dB, the TCN-DPA-TA is approximately 1 order of magnitude lower than the TRFI, which is the best classical estimator. The performance of the TCN-DPA estimator at SNR levels less than 20 dB is close to that of classical estimators, regardless of the extensive hyperparameter tuning. This necessitates the addition of TA to the TCN-DPA to smooth the channel estimate per subcarrier by considering the current and previous channel estimate values. The improvement in BER performance of TCN-based estimators can be attributed to their ability to effectively capture relationships between adjacent subcarrier frequency DPA estimates. The TCN learns how the channel affects adjacent subcarriers, and using this information the TCN can accurately predict the channel's response for each individual subcarrier. This allows for a more accurate reconstruction of channel state information across the frequency domain. The better BER of TCN-DPA-TA compared to that of TCN-DPA in low SNR regions further demonstrates the effectiveness of TA postprocessing in refining these estimates.

B. Normalised Mean Squared Error (NMSE)

Fig. 6b shows the NMSE performance of the TCN-based estimators compared to the classical methods namely DPA, LS, STA, CDP, TRFI, DPA-TA and a baseline TCN. The TCN-DPA-TA estimator (red line graph) shows even better performance than the TCN-DPA estimator (blue line graph) because of the added temporal averaging process. It is also evident in



(a) BER simulation results of various estimators.



(b) NMSE simulation results of various estimators.

Fig. 6: BER and NMSE simulation results for the VTV-SDWW vehicular channel model.

this figure that the performance of the TCN-DPA estimator at low SNRs is similar to that of the classical estimators, but from 20 dB its performance increases exponentially outperforming all the classical estimators. For the TCN-DPA-TA, it starts outperforming all the classical estimators from 10 dB, which is lower than that of the TCN-DPA.

V. Conclusion

In this paper, we propose the application of a tuned TCN with DPA and TA for channel estimation in VTV-SDWW TDL vehicular communications using adjacent subcarriers as input to the TCN. Our performance comparisons between TCN-based estimators and classical estimators demonstrate the suitability of TCNs for vehicular channels. We introduce a new method of structuring data for NNs, by separating and interleaving the real and imaginary components of complex data along the OFDM symbol dimension. Our experiments show that the BER performance of the TCN-based estimators achieves improvements of up to one order of magnitude for TCN-DPA-TA and up to 0.7 orders of magnitude for TCN-DPA. These results demonstrate the superior channel estimation capabilities of our proposed estimators in vehicular communications. In the future, we plan to conduct a detailed computational complexity analysis of the proposed estimators and investigate them in other vehicular channel models.

Acknowledgment

The authors gratefully acknowledge the financial support of this study by the Telkom CoE at NWU and NITheCS.

References

[1] A. K. Gizzini and M. Chafii, "RNN Based Channel Estimation in Doubly Selective Environments," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 2, pp. 1–18, 2024.

[2] A. K. Gizzini, M. Chafii, S. Ehsanfar, and R. M. Shubair, "Temporal Averaging LSTM-based Channel Estimation Scheme for IEEE 802.11p Standard," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 01–07.

[3] P. Walk, H. Becker, and P. Jung, "OFDM Channel Estimation via Phase Retrieval," in *2015 49th Asilomar Conference on Signals, Systems and Computers*, 2015, pp. 1161–1168.

[4] J. A. Fernandez, K. Borries, L. Cheng, B. V. K. Vijaya Kumar, D. D. Stancil, and F. Bai, "Performance of the 802.11p Physical Layer in Vehicle-to-Vehicle Environments," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 3–14, 2012.

[5] Z. Zhao, X. Cheng, M. Wen, B. Jiao, and C.-X. Wang, "Channel Estimation Schemes for IEEE 802.11p Standard," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 38–49, 2013.

[6] Y.-K. Kim, J.-M. Oh, Y.-H. Shin, and C. Mun, "Time and Frequency Domain Channel Estimation Scheme for IEEE 802.11p," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 1085–1090.

[7] J. Pan, H. Shan, R. Li, Y. Wu, W. Wu, and T. Q. Quek, "Channel Estimation Based on Deep Learning in Vehicle-to-Everything Environments," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1891–1895, 2021.

[8] A. M. Abdelgader and W. Lenan, "The Physical Layer of the IEEE 802.11p WAVE Communication Standard: The Specifications and Challenges," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2014, pp. 22–24.

[9] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Deep Learning Based Channel Estimation Schemes for IEEE 802.11p Standard," *IEEE Access*, vol. 8, pp. 113 751–113 765, 2020.

[10] Y. LeCun, Y. Bengio *et al.*, "Convolutional Networks for Images, Speech, and Time Series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[11] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[12] G. Acosta-Marum and M. A. Ingram, "Six Time- and Frequency-Selective Empirical Channel Models for Vehicular Wireless LANs," in *2007 IEEE 66th Vehicular Technology Conference*, 2007, pp. 2134–2138.

[13] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Joint TRFI and Deep Learning for Vehicular Channel Estimation," in *2020 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2020, pp. 1–6.

[14] R. J. Borgli, H. Kvale Stensland, M. A. Riegler, and P. Halvorsen, "Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization," in *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*, 2019, pp. 1–6.

[15] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-Generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.

Sequence Based Deep Neural Networks for Channel Estimation in Vehicular Communication Systems

SA Ngorima[1,2,3][0000−0002−0775−3529], ASJ Helberg[1][0000−0001−6833−5163], and MH Davel[1,2,3][0000−0003−3103−5858]

[1] Faculty of Engineering, North-West University, South Africa
[2] Centre for Artificial Intelligence Research, South Africa
[3] National Institute for Theoretical and Computational Sciences, South Africa
aldringorima@gmail.com
albert.helberg@nwu.ac.za

Abstract. Channel estimation is a critical component of vehicular communications systems, especially in high-mobility scenarios. The IEEE 802.11p standard uses preamble-based channel estimation, which is not sufficient in these situations. Recent work has proposed using deep neural networks for channel estimation in IEEE 802.11p. While these methods improved on earlier baselines they still can perform poorly, especially in very high mobility scenarios. This study proposes a novel approach that uses two independent LSTM cells in parallel and averages their outputs to update cell states. The proposed approach improves normalised mean square error, surpassing existing deep learning approaches in very high mobility scenarios.

Keywords: Channel estimation · deep learning · dual-cell LSTM · IEEE 802.11p · vehicular channels.

1 Introduction

The international wireless communication standard IEEE 802.11p was introduced for vehicle-to-vehicle communication. However, the original IEEE 802.11p framework did not take into account high mobility and rapid channel changes, which can affect system performance at high speeds [15]. Reliable wireless vehicular communication is challenging due to the need for accurate channel state information (CSI) in high-mobility scenarios, where estimates can quickly become outdated.

IEEE 802.11p employs a data pilot-aided (DPA) approach for channel estimation, with techniques such as spectral temporal averaging (STA) [3] and time-domain reliable test frequency domain interpolation (TRFI) [11] estimators to improve accuracy. However, in high signal-to-noise ratios (SNRs), STA may not effectively account for frequency and time variability while TRFI's assumption of high correlation between successive transmitted signals may not hold true.

The ability of a deep neural network (DNN) to generalize in a robust manner has made it an attractive option to improve channel estimation. DNN algorithms have been investigated as a means of improving traditional channel estimation techniques such as STA and TRFI. These DNN-based postprocessors have excelled at capturing time-varying behaviours in complex propagation environments. An auto-encoder (AE) DNN successfully handled the inherent error propagation challenges in DPA estimation [8]. Another approach combined STA for initial linear channel estimation with a deep neural network (DNN) as a nonlinear postprocessor [5]. In Gizzini et al. [6] a TRFI-DNN was introduced that combines TRFI and a DNN. TRFI-DNN has been shown to be more effective than other DNN methods compared with.

Performance-wise, AE-DNN [8] has a relatively low bit error rate (BER). However, it is computationally complex and does not effectively learn the time-frequency channel correlation. STA-DNN [5] outperforms both AE-DNN and TRFI-DNN [6] in low SNRs of up to 20 dB at a velocity of 120 km/h. However, its performance starts to deteriorate as the SNR increases. TRFI-DNN, on the other hand, outperforms AE-DNN by approximately 3 dB for BER = $10^{-}3$. In scenarios where mobility is higher than 120 km/h, STA-DNN starts to experience an error floor at even lower SNRs of 17 dB. However, the challenge of dealing with a rapidly changing channel remains for both temporal filtering and DNN postprocessing.

To estimate wireless channels in high-mobility situations, we propose using a dual-cell long-short-term memory (LSTM) network. This network can capture short-term temporal dependencies and causal correlations. A DPA block and a temporal averaging (TA) block are used to further reduce the noise in the dual-cell LSTM estimates.

The remainder of the paper is structured as follows: Section 2 reviews IEEE 802.11p and channel estimation, Section 3 introduces the dual-cell LSTM approach, Section 4 describes the experiments, Section 5 presents results and analysis, and Section 6 concludes.

2 Background

This section provides a brief overview of the IEEE 802.11p standard specification. It also discusses channel estimation system models and techniques within the context of IEEE 802.11p, serving as the foundation for improving vehicular communication systems.

2.1 IEEE 802.11p Standard Specifications and Frame Structure

The IEEE 802.11p protocol employs a technique referred to as Orthogonal Frequency Division Multiplexing (OFDM) to transfer data via radio channels. OFDM divides the available bandwidth into several closely spaced subcarrier frequencies, allowing many signals to be sent simultaneously. During transmission, some subcarriers carry pilot signals that are known by both the transmitter and

the receiver. The receivers use pilots to analyse changes in the communication channel. As vehicles travel at different speeds, factors such as distance, barriers, and multipath effects constantly change the way signals propagate. These pilot signals can be used to track time-varying channel conditions and obtain the CSI. This CSI can then be used to improve the decoding of transmitted user data, ensuring efficient communication.

OFDM signal consists of modulation symbols sent on an active subcarrier during a particular time interval. OFDM frames are made up of several payload OFDM signals that are preceded by preamble symbols that fulfil various key roles. The preamble enables the receiver to synchronise, establish an initial coarse channel estimate, and mark the beginning of incoming frames.

In IEEE 802.11p, only 52 of the 64 available subcarriers are used for data transmission and pilot symbols, as illustrated in Figure 1. The rest of the subcarriers are kept for other purposes, such as guard bands and direct current (DC) offset. To ensure accurate channel tracking, pilot signals are inserted into specific subcarriers.

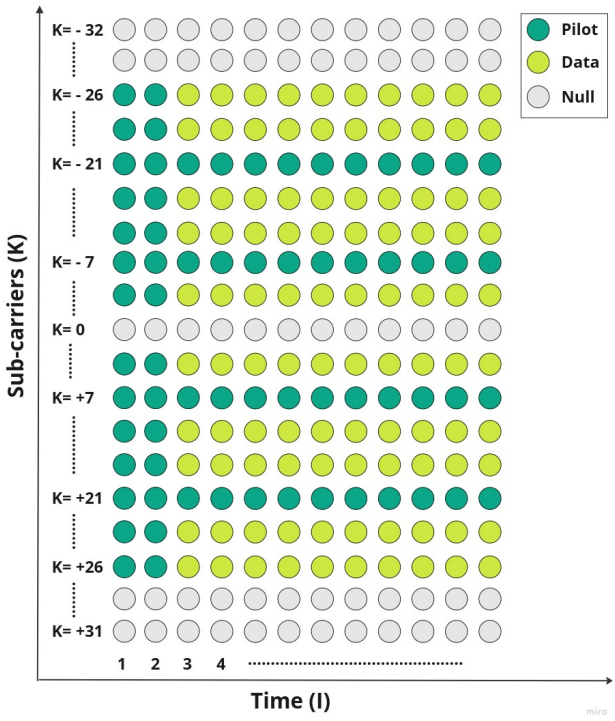


Fig. 1: Subcarrier arrangements in a IEEE802.11p transmission [5].

2.2 Channel Estimation System Model

This study assumes perfect synchronisation and only considers a frame structure comprising two extended preambles at the start, followed by I OFDM data symbols. Following the terminology used in [4], the expression of the transmitted and received OFDM symbol can be represented as follows:

$$\tilde{y}_i[k] = \begin{cases} \tilde{h}_{i,d}[k]\tilde{x}_{i,d}[k] + \tilde{v}_{i,d}[k], & k \in \mathcal{K}_{\mathrm{d}} \\ \tilde{h}_{i,p}[k]\tilde{x}_{i,p}[k] + \tilde{v}_{i,p}[k], & k \in \mathcal{K}_{\mathrm{p}} \end{cases}, \tag{1}$$

where $\tilde{x}_i[k]$ denotes the OFDM symbol transmitted at the i^{th} instance, affected by its corresponding time-varying frequency-domain channel response $\tilde{h}_i[k]$. $\tilde{y}_i[k]$ are the OFDM data symbols received. Furthermore, $\tilde{v}_i[k]$ denotes the frequency-domain equivalent of additive white Gaussian noise (AWGN) characterised by a variance of σ^2. The indices K_d and K_p refer to the sets of subcarriers dedicated to data and pilot signals, respectively.

Thus, the OFDM symbol received ($\tilde{y}_i[k]$) consists of both the data and the pilot subcarrier symbols affected by the channel response, with noise added. Channel estimation is the task of determining $\tilde{h}_i[k]$ for each OFDM symbol.

2.3 IEEE 802.11p Channel Estimation Schemes

Accurate channel estimation in OFDM systems such as IEEE 802.11p relies on well-allocated pilot subcarriers. However, the standard's limited number of pilots may not suffice for tracking channel variations effectively. Two proposed channel estimation schemes tackle this issue: one uses data subcarriers alongside pilots, while the other inserts additional pilots at the cost of a reduced transmission rate. Achieving reliable channel estimation requires a careful balance between accuracy and transmission efficiency.

Recently, models based on recurrent neural networks (RNNs) have shown promise in leveraging sequential dependencies in channel data for channel estimation in vehicular communication [4, 10, 13]. Among RNN architectures, LSTM networks can learn long-term dependencies and have shown excellent results in sequential tasks [9].

In the context of channel estimation, an LSTM has been used to directly map pilot sequences to channel responses [13], while Gizzini et al. proposed a long-short-term memory data pilot-aided temporal averaging (LSTM-DPA-TA) integration to improve estimates in high mobility situations [4]. Hou et al. [10] also employed a single-layer Gated Recurrent Unit (GRU) as a post-processing for the estimation of DPA.

DPA Estimation The DPA estimation technique uses both pilot subcarriers and demapped data subcarriers to estimate the channel for the present OFDM symbol [8]. Following the notation in [4] the demapped data symbol $d_i[k]$ for

each subcarrier $\mathcal{K}$ is given in Equation 2 as follows:

$$d_i[k] = \mathfrak{D}\left(\frac{y_i[k]}{\hat{\tilde{h}}_{\mathrm{DPA}_{i-1}}[k]}\right), \hat{\tilde{h}}_{\mathrm{DPA}_0}[k] = \hat{\tilde{h}}_{\mathrm{LS}}[k], k \in \mathcal{K}_{\mathrm{on}}, \tag{2}$$

Here, $\mathfrak{D}(.)$ signifies the process of demapping to the closest constellation point based on the chosen modulation order, $\mathcal{K}_{\mathrm{on}}$ are the active subcarriers, $\tilde{h}_{\mathrm{LS}}[k]$ is the channel estimated using the Least Squares (LS) method from the received preambles, namely $y_1^{(p)}[k]$ and $y_2^{(p)}[k]$ such that:

$$\tilde{h}_{\mathrm{LS}}[k] = \frac{y_1^{(p)}[k] + y_2^{(p)}[k]}{2p[k]}, k \in \mathcal{K}_{\mathrm{on}}, \tag{3}$$

In this context, $p[k]$ denotes the predetermined frequency-domain preamble sequence. Subsequently, the final updates for the DPA channel estimation are carried out in the following manner:

$$\tilde{h}_{\mathrm{DPA}_i}[k] = \frac{y_i[k]}{d_i[k]}, k \in \mathcal{K}_{\mathrm{on}}. \tag{4}$$

It should be noted that DPA fundamental estimation serves as an initial reference for a majority of IEEE 802.11p channel estimation techniques [10].

TA Processing Temporal averaging (TA) is a noise reduction technique in which the noise reduction ratio can be calculated analytically [4]. The TA method assumes that the noise terms of consecutive OFDM symbols are not correlated. TA analyses variations by averaging the channel estimations across OFDM symbols. Following the notation in [4], consider the estimate of the channel $h_{k,n}$ in the subcarrier k and symbol n. TA computes a moving estimate $\hat{H}_k$ of the channel frequency response h_k as the weighted sum:

$$\hat{H}_{k,n} = (1 - 1/\alpha)\hat{h}_{k-1,n} + (1/\alpha)\hat{h}_{k,n} \tag{5}$$

In this case, α denotes the window size over which the averages are calculated. A larger α incorporates more symbols. TA reduces noise through averaging, making it a good candidate for postprocessing.

STA-DNN Estimator Previous work [5] recognised the problems in only using traditional STA estimates [3] and proposed a hybrid STA-DNN technique to solve them. The typical STA estimator averages the initial estimate of the DPA channel over frequency and time.

To compensate for the time-varying channel conditions, STA uses fixed average window widths and weights [3]. To counteract this, Gizzini et al. [5] suggest creating an STA-DNN estimator by feeding the STA estimate to a deep neural network (DNN). Their evaluation showed that the STA-DNN technique corrects errors and greatly increases the accuracy of the estimation over the traditional STA estimation [5]. Despite these improvements, there is still a significant error in high-mobility vehicle situations at low SNR [4].

LS Channel Estimate In IEEE 802.11p, the basic LS channel estimation technique is used as initial channel estimation. Two successive training symbols T_1 and T_2 are used to obtain the initial frequency response of the channel of the k^{th} subcarrier as follows:

$$\tilde{H}_0(k) = \frac{Y_{T1}(k) + Y_{T2}(k)}{2X_T(k)}, k \in \mathcal{K}_{\text{on}} \tag{6}$$

where $X_T(k)$ is the predefined training symbol of the k^{th} subcarrier, $Y_{T1}(k)$ and $Y_{T2}(k)$ are the symbols in the frequency domain of the receiver. The LS channel estimation approach overlooks time variations, resulting in reduced accuracy as the OFDM symbol index increases. To maintain reliable performance in mobile scenarios, more advanced channel tracking techniques are essential, accounting for dynamic channel changes and ensuring accurate estimation.

The channel estimation schemes discussed are used in this work for comparison purposes. LS estimation is used as an initial estimation approach before our dual-cell LSTM method. This is followed by applying DPA as a post-processing step to refine our dual-cell LSTM estimate. A step-by-step procedure on how these techniques are used in our work is discussed in detail in Section 3.

3 Proposed Dual-Cell LSTM Estimation Method

LSTM networks are designed for sequential data with temporal dependencies. These networks have an architecture that allows them to understand the temporal dependencies in the data, enabling them to predict future data based on past observations [9]. The gated cell structure of LSTMs enables them to learn long-term temporal relations by regulating the input of new information, eliminating unnecessary past information, and updating the hidden state using selected cell state values [14]. LSTMs have been successfully used in different domains and applications such as time series forecasting [12] and speech recognition [7].

This section introduces a novel dual-cell LSTM architecture that uses two independent LSTM cells in parallel to capture sequential information from different temporal perspectives. This enables the model to fuse representations learnt from different, non-overlapping temporal perspectives at each timestep. The method also integrates DPA and TA as post-processing methods to reduce noise.

The dual-cell structure is distinct from a bidirectional LSTM as it is composed of two parallel LSTMs operating in the forward direction, while a bidirectional LSTM comprises two LSTMs, one processing the sequence in a forward direction and the other in a backward direction. Our hypothesis is that this dual-cell architecture is capable of capturing a more complete image of channel dynamics compared to other LSTM structures. The proposed protocol aims to efficiently address channel estimation in highly dynamic environments, such as vehicular communication.

As illustrated in Figure 2, the inputs (x_{t-1}, x_{t-2}, x_{t-3}, etc.) are supplied directly to the independent LSTM cells at the same time. Following the typical

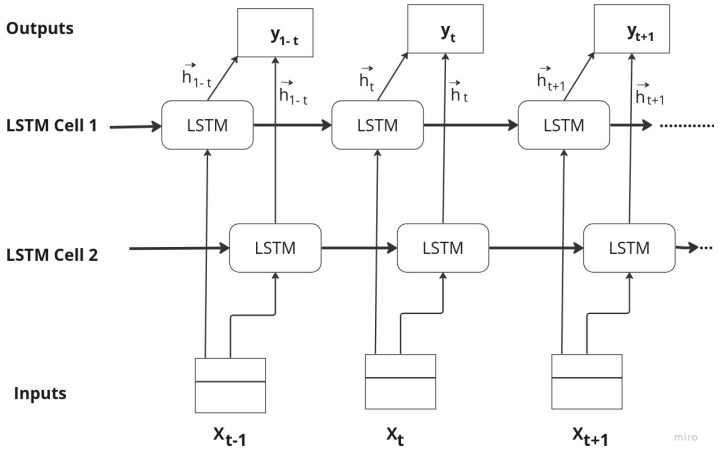


Fig. 2: Schematic representation of the dual-cell LSTM estimator. The architecture consists of two parallel LSTM cell chains, with each cell processing sequential inputs from x_{t-1} to x_t. The outputs from both chains are fused to produce the final output.

LSTM design [9], each LSTM cell consists of an input gate, a forget gate, an output gate, and a cell memory block. For both cells, the number of hidden units is set to H. At each time step t, the same input sequence x_t is fed into both cells to update their hidden states ($h_{1,t}$ and $h_{2,t}$) and cell states ($c_{1,t}$ and $c_{2,t}$). This processing takes place in parallel, without information exchange between cells. Finally, a combined output y_t is produced by averaging the hidden state outputs of both cells and passing them through a linear layer. This captures representations from the two processing streams, although independently and simultaneously. Averaging the outputs of the two cells has various benefits:

- Combining the representations learned by each cell helps to prevent overfitting. Cells may capture somewhat different features of the input sequence.
- Averaging provides a type of ensembling in which the combination of numerous models (cells) outperforms isolated ones.
- By averaging time-distributed outputs at each step, the model may use information processed in both cells at the same time.

3.1 Dual-Cell LSTM Estimation Algorithm

This section presents the algorithm for the proposed dual-cell LSTM model. (Also see Algorithm 1.)

The Dual-Cell LSTM model for the channel estimation method can be broken down into four phases:

- First, an LS channel estimation is performed. Initial channel estimation is obtained as illustrated in Section 2.3. This forms the input dataset ($\tilde{\tilde{x}}_i$) used

Algorithm 1 Dual-Cell LSTM Model for Channel Estimation

Data: HLS_Structure, True_Channel_Structure,
Step 1: LS estimation to obtain the initial channel estimation that is used as the training dataset (HLS_Structure)
Step 2: Load dataset
- Load True_Channel_Structure
- Load HLS_Structure

Step 3: Prepare input and target matrices
- Construct Dataset_X by combining real and imaginary parts of HLS_Structure.
- Construct Dataset_Y by combining real and imaginary parts of True_Channel_Structure.

Step 4: Initialize dual LSTM cell model
- Create two LSTM cells: cell1 and cell2
- Initialise hidden and cell states of both cells

Step 5: Train model
while not converged **do**
Step 5.1: Forward pass
- Feed Dataset_X to both cells
- Get outputs out1 and out2 from cell1 and cell2
- Calculate out_avg = (out1 + out2)/2
- Calculate loss between out_avg and Dataset_Y

Step 5.2: Backpropagate loss
- Update cell parameters using backpropagation

end while
Step 6: Channel estimation on new data
- Forward pass new samples through trained model
- Get channel estimates $\hat{est} = out\_avg$

Step 7: Post-processing
- Perform DPA estimation on $\hat{est}$ to improve estimates
- Perform TA on DPA outputs
- Final estimated channel

Step 8: Calculate evaluation metrics
- Calculate NMSE, BER., on test dataset

Step 9: Repeat step 4 for the required number of epochs
Step 10: Save trained model

to train the model. Following the notation presented in [4], the input $\bar{x}_i$ can be obtained as follows:

$$\bar{x}_i = \begin{cases} \hat{\hat{h}}_{\mathrm{LSTM}_{i-1,d}}[k], & k \in \mathcal{K}_{\mathrm{d}} \\ \hat{\hat{h}}_{i-1,p}[k], & k \in \mathcal{K}_{\mathrm{p}} \end{cases}. \tag{7}$$

The input $\tilde{\bar{x}}_i$ is computed by transforming LS estimate $\bar{x}_i$ from complex values to real values. $\hat{\hat{h}}_{i-1,p}[k]$ is the LS estimated channel at the K_p subcarriers. $\tilde{\bar{x}}_i$ is input to the LSTM as:

$$\hat{\hat{h}}_{\mathrm{LSTM}_{i.d}} = \Omega_{\mathrm{LSTM}}(\tilde{\bar{x}}_i,\ \Theta), \tag{8}$$

where Ω_{LSTM} denotes the LSTM processing unit and Θ denotes the overall weights.

– Second, the LSTM model is trained to estimate the channel characteristics from the received data.
– Third, the DPA post-processing technique is used to improve estimations, as follows:

$$d_{\mathrm{LSTM}_i}[k] = \mathfrak{D}\left(\frac{y_i[k]}{\hat{\hat{h}}_{\mathrm{LSTM}_{i-1}}[k]}\right), \hat{\hat{h}}_{\mathrm{LSTM}_0}[k] = \hat{\hat{h}}_{\mathrm{LS}}[k], \tag{9}$$

$$\hat{\hat{h}}_{\mathrm{LSTM-DPA}_i}[k] = \frac{y_i[k]}{d_{\mathrm{LSTM}_i}[k]}. \tag{10}$$

– Fourth, the TA technique is the final post-processing method: TA is applied to the estimated channel $\hat{\hat{h}}_{\mathrm{LSTM-DPA}_i}[k]$ to further reduce the impact of AWGN noise as follows:

$$\hat{\hat{h}}_{\mathrm{DNN-TA}_{i,d}} = \left(1 - \frac{1}{\alpha}\right)\hat{\hat{h}}_{\mathrm{DNN-TA}_{i-1,d}} + \frac{1}{\alpha}\hat{\hat{h}}_{\mathrm{LSTM-DPA}_{i,d}}. \tag{11}$$

A fixed α value of 2 is used this is based on the analysis done in [4].

Finally, estimates are evaluated using normalised mean squared error (NMSE) and bit error rate (BER) metrics.

4 Experimental Setup

This section discusses the channel model employed for data generation, the data preparation for LSTM and the hyperparameter optimisation process.

4.1 Channel Model

Vehicular channel models, widely examined in the literature [4–6], originate from channel measurements in metropolitan Atlanta, Georgia, USA [1]. We conducted an analysis of the vehicle-to-vehicle same direction with wall (VTV-SDWW) tapped delay line (TDL) vehicular channel model, which is used for communication between two vehicles travelling in the same direction with a central wall between them and maintaining a distance of 300-400 metres between them.

Two mobility scenarios based on the VTV-SDWW TDL were adopted for comparison: (i) high mobility (V = 100 km/h, Doppler shift f_d = 550 Hz) and (ii) very high mobility (V = 200 km/h, f_d = 1,100 Hz). Simulation parameters included a frame size of 50 OFDM symbols, 16QAM modulation, and convolutional channel coding at a half-code rate. The dataset consisted of 12,000 training samples, 4,000 validation samples, and 2,000 testing samples. During training, a training SNR level of 40 dB is used to improve the generalization of the model. Each simulation generated a packet of 50 OFDM symbols sent across a simulated wireless channel, capturing varying channel conditions. In this study, a 'packet sample' represents one 50-OFDM symbol packet, serving as a single observation for estimation.

4.2 Data Preparation for LSTM

The dataset that was created according to the IEEE 802.11p standard (see Section 2). We excluded the signal field from the dataset, assuming optimal receiver synchronisation, for the sake of simplicity. As a result, the emphasis is on the presence of two extended training symbols within each transmitted frame, followed by a set of I Orthogonal Frequency Division Multiplexing (OFDM) data symbols. The received OFDM symbol is represented by Equation 1.

We constructed $Dataset\_X$ and $Dataset\_Y$ matrices for the input and target data, respectively. For training data, we obtained a real-valued training dataset $Dataset\_X$ by concatenating the real and imaginary parts of the propagating channel ($HLS\_Structure$) into one vector. $Dataset\_Y$ is the target output matrix created similarly to $Dataset\_X$ by concatenating the real and imaginary parts of the actual or ground truth channel for specific positions, 1 to 48 for real and 49 to 96 for imaginary.

$Dataset\_X$ has dimensions $12,000 \times 50 \times 104$. Similarly, $Dataset\_Y$ has dimensions $12,000 \times 50 \times 96$. The first dimension is the size of the dataset. The second dimension denotes the number of OFDM symbols per frame, while the third is the sequence length, which is the size of the input data positions if it is $Dataset\_X$ or output if its $Dataset\_Y$. Our input size is 104 and our output size is 96 (as above).

4.3 Hyperparameter Optimisation

The hyperparameters of the dual-cell LSTM model were optimised using Optuna to find the values that minimise loss of validation during training. Optuna is a Bayesian optimisation framework that generates hyperparameter configurations using a tree-structured parzen estimator (TPE) to maintain a probabilistic model that links hyperparameters to measurable outcomes [2]. It terminates underperforming tests early to maximise search efficiency. The procedure is repeated until either the desired convergence is achieved or the maximum number of trials is reached.

We optimised a set of hyperparameters, namely the learning rate, the step size of the optimiser, the gamma of the *StepLR* scheduler, the batch size, the weight decay and the dropout probability. The optimisation procedure was carried out as follows:

1. A dual LSTM model with fixed input size (104) was defined. This is the total number of input subcarriers used in this work ($K_{on} * 2$ where $K_{on} = 52$ active subcarriers). The LSTM size was 128.
2. The non-dominated Sorting Genetic Algorithm II (NSGA-II) sampler created an Optuna study for multi-objective optimisation.
3. An objective function trained the dual-cell LSTM for 100 epochs with specified hyperparameters (Table 1).
4. Optuna recommended hyperparameters within defined limits, and Adam optimiser initialised and trained the dual-cell LSTM.

5. Training and validation losses were tracked for each epoch, with the average validation loss used.
6. The study ran 150 optimisation trials to identify optimal hyperparameters based on the lowest validation loss.
7. The loss curve, weight, and bias plots were logged to the weights and bias[4] for visualisation.
8. Optimal trial hyperparameters (learning rate, step size, gamma, dropout, weight decay) were determined on the basis of the lowest validation loss.

The final model is trained for 250 epochs using a batch of 128. A summary of the parameters of the dual cell LSTM model used in this work is given in Table 1.

Table 1: Optuna Hyperparameter optimisation: 150 trials and 100 epochs

Hyperparameters	**Search space**	**Final value**
Learning rate	[1e-5, 1e-1]	0.01
Step size	[1, 10]	9
gamma	[0.1, 1]	0.6
Dropout probability	[0.0, 0.8]	0.002
Weight decay	[1e-6, 1e-2]	0.008
Batch size	[16, 32, 48, . . . ,128]	128

5 Analysis and Simulation Results

We evaluate the following channel estimation methods, STA-DNN [5], LSTM-DNN-DPA [13], LSTM-DPA-TA [4], DPA-TA and the proposed dual-cell LSTM estimator against a perfect channel by calculating BER and NMSE in MATLAB. The hidden layer dimensions of the STA-DNN were 15-15-15 and 128-40 for the LSTM-DNN.

1. **LSTM-DNN-DPA:** This method involves cascading an LSTM and a multilayer perceptron network (MLP) to estimate the channel and DPA as a post processor [13]. However, overfitting was not evaluated in this work, as cascaded models possess a high risk of overfitting due to an increased number of parameters.
2. **LSTM-DPA-TA (128):** This approach uses an LSTM model for channel tracking followed by DPA and TA for noise reduction. Overall performance is based on the precision of the initial LSTM estimate. The noise mitigation ratio of TA processing has been analytically derived in [4]. This method was not evaluated against other LSTM architectures.

[4] https://wandb.ai/

3. **LSTM-DPA-TA (64):** This is the same LSTM-based method as above but using a smaller LSTM: size 64 instead of 128 [4]. A smaller LSTM size struggles to learn long-term dependencies in dynamic environments compared to a larger LSTM.
4. **STA-DNN:** This technique utilises DNNs to capture additional temporal and frequency correlations [5]. However, this method does not consider sequential data, thus limiting its capability.
5. **DPA-TA:** We established our lower bound by using only the DPA and TA techniques. The shortcoming of this approach is that it only uses conventional techniques which are limited to learning patterns in data, especially dynamic channels.
6. **Dual-cell LSTM:** We evaluate the performance of our dual-cell LSTM network approach against the other methods in the same scenarios discussed below.

5.1 High Mobility Scenario

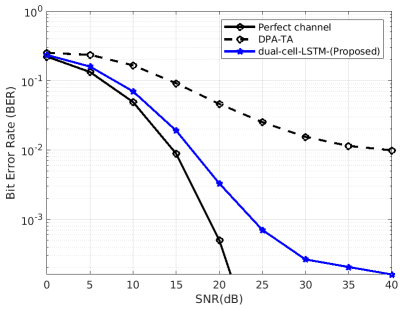
In this section, we evaluate the performance of existing approaches and our proposed dual-cell LSTM method under high mobility conditions, characterised by a velocity (v) of 100 km/h and a Doppler frequency (f_d) of 550Hz using two key metrics; BER and NMSE.

Clarification on Data Comparison: The results of existing methods have been extracted directly from Gizzen et al. [4], while the dataset used to develop the dual-cell LSTM method was generated in-house following the experimental setup detailed in that respective paper. The performance of the proposed method is compared against the stated published performance of the methods presented in [4]. We caution that in this comparison, the datasets generated according to Section 4.2 and the unpublished dataset in [4] may differ, although the described vehicular scenarios were precisely duplicated. To reflect this caution, we present the comparison in separate graphs in Figures 3 and 4.

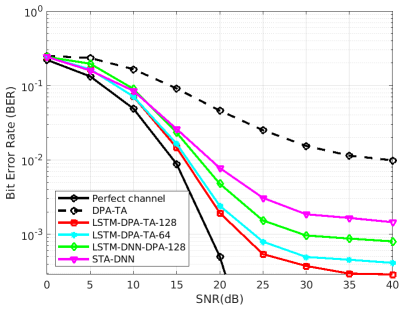
Figures 3a and 3b show the BER performance results of the DNN-based estimators and the classical DPA-TA estimator. Our experiments showed that the dual-cell LSTM was capable of dealing with a variety of SNR levels. At SNR levels between 0 and 12 dB, it performed similarly to LSTM-DPA-TA (128). When the SNR was between 27 and 40 dB, it outperformed all the models tested. Figures 3c and 3d compare the dual-cell LSTM NMSE performance with that of existing techniques. Lower NMSE values indicate a better estimate of the true channel response. At an NMSE of 10^{-2}, the dual-cell LSTM method shows better performance than LSTM-DA-TA (128), LSTM-DA-TA (64) and LSTM-DNN by approximately 7 dB, 8 dB, and 12 dB improvements, respectively. At 35 dB SNR, it reaches an NMSE of $10^{-}3$.

5.2 Very High Mobility Scenario

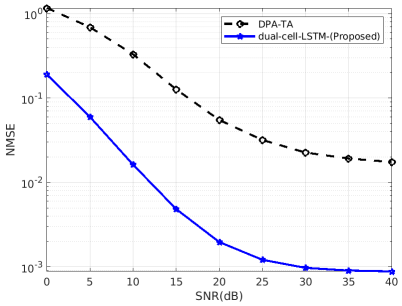
This section evaluates channel estimation methods under very high mobility conditions of 200km/h and a frequency Doppler of 1,100Hz. BER and NMSE



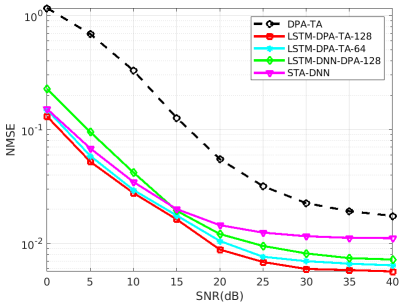
(a) BER performance of the proposed dual-cell LSTM method.



(b) BER performance results of existing methods.



(c) NMSE performance of the proposed dual-cell LSTM method.

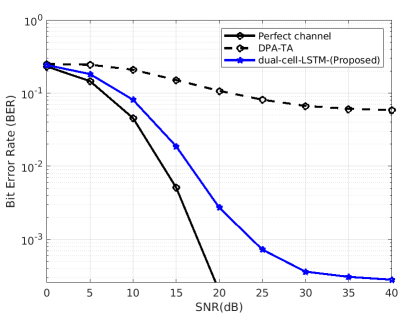


(d) NMSE performance of existing methods.

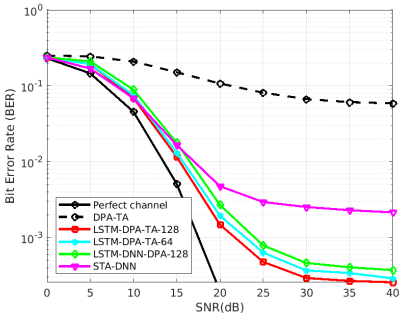
Fig. 3: BER and NMSE results under High Mobility conditions using the VTV-SDWW channel model at $v = 100$ km/h, $f_d = 550$ Hz.

metrics are used to evaluate performance. We again follow the experimental setup of Grizzini et al. [4] to simulate the very high mobility environment, and present results separately. Figures 4a and 4b plot the BER against SNR, where all estimators showed reduced performance due to fast fading that causes a decrease in the temporal correlations of the channel.

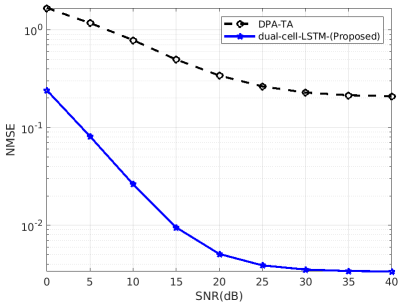
At SNR levels ranging from 0 to 15 dB, the LSTM-DPA-TA (128) approach surpasses the dual-cell LSTM technique by up to 0.5 dB. On the other hand, the BER of feedforward models such as STA-DNN deteriorates significantly more rapidly even at high SNR due to its inability to exploit temporal dependencies in highly dynamic channels. The corresponding NMSE vs SNR graphs are shown in Figures 4c and 4d. Similarly, despite the challenges of extremely high mobility, dual-cell LSTM showed a capability to achieve a very low NMSE by a significant margin compared to the other existing DNN methods.



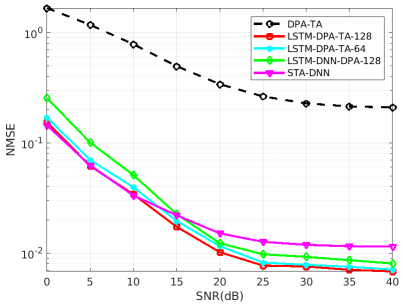
(a) BER performance of the proposed dual-cell LSTM method.



(b) BER performance results of existing methods.



(c) NMSE performance of the proposed dual-cell LSTM method.



(d) NMSE performance of existing methods.

Fig. 4: BER and NMSE performance at Very High Mobility: VTV-SDWW channel model at V = 200km/h, f_d = 1,100Hz.

6 Conclusion

This paper introduced a dual-cell LSTM network model for dynamic channel estimation in vehicular communication systems. A comparison was conducted with results available from the literature, with experimental setups duplicated. Our dual-cell LSTM model showed the potential to achieve very low NMSE between estimated and true channel responses across all tested SNR levels of magnitude close to 1 compared to other sequence models. In terms of BER, the dual LSTM method showed improved performance in high SNRs, starting at 30 dB, performing better than the other methods. This result supports our hypothesis that the combination of two LSTM methods in a parallel ensemble can outperform the existing single LSTM method and the feedforward methods. Future work will include evaluating the complexity and learning capacity of ensemble sequential learning methods.

Acknowledgements The authors are grateful to NITheCS, the Telkom CoE at the NWU and Hensoldt South Africa, who supported this research.

References

1. Acosta-Marum, G., Ingram, M.A.: Six time-and frequency-selective empirical channel models for vehicular wireless lans. IEEE Vehicular Technology Magazine **2**(4), 4–11 (2007)
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 2623–2631 (2019)
3. Fernandez, J.A., Borries, K., Cheng, L., Kumar, B.V., Stancil, D.D., Bai, F.: Performance of the 802.11 p physical layer in vehicle-to-vehicle environments. IEEE transactions on vehicular technology **61**(1), 3–14 (2011)
4. Gizzini, A.K., Chafii, M., Ehsanfar, S., Shubair, R.M.: Temporal averaging lstm-based channel estimation scheme for ieee 802.11 p standard. In: 2021 IEEE Global Communications Conference (GLOBECOM). pp. 01–07. IEEE (2021)
5. Gizzini, A.K., Chafii, M., Nimr, A., Fettweis, G.: Deep learning based channel estimation schemes for ieee 802.11 p standard. IEEE Access **8**, 113751–113765 (2020)
6. Gizzini, A.K., Chafii, M., Nimr, A., Fettweis, G.: Joint trfi and deep learning for vehicular channel estimation. In: 2020 IEEE Globecom Workshops (GC Wkshps. pp. 1–6. IEEE (2020)
7. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. pp. 6645–6649. Ieee (2013)
8. Han, S., Oh, Y., Song, C.: A deep learning based channel estimation scheme for ieee 802.11 p systems. In: ICC 2019-2019 IEEE International Conference on Communications (ICC). pp. 1–6. IEEE (2019)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Hou, J., Liu, H., Zhang, Y., Wang, W., Wang, J.: Gru-based deep learning channel estimation scheme for the ieee 802.11p standard. IEEE Wireless Communications Letters **12**(5), 764–768 (2023). https://doi.org/10.1109/LWC.2022.3187110
11. Kim, Y.K., Oh, J.M., Shin, Y.H., Mun, C.: Time and frequency domain channel estimation scheme for ieee 802.11 p. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 1085–1090. IEEE (2014)
12. Li, Y., Zhu, Z., Kong, D., Han, H., Zhao, Y.: Ea-lstm: Evolutionary attention-based lstm for time series prediction. Knowledge-Based Systems **181**, 104785 (2019)
13. Pan, J., Shan, H., Li, R., Wu, Y., Wu, W., Quek, T.Q.: Channel estimation based on deep learning in vehicle-to-everything environments. IEEE Communications Letters **25**(6), 1891–1895 (2021)
14. Staudemeyer, R.C., Morris, E.R.: Understanding lstm - a tutorial into long short-term memory recurrent neural networks. ArXiv **abs/1909.09586** (2019), https://api.semanticscholar.org/CorpusID:202712597
15. Zhao, Z., Cheng, X., Wen, M., Jiao, B., Wang, C.X.: Channel estimation schemes for ieee 802.11 p standard. IEEE intelligent transportation systems magazine **5**(4), 38–49 (2013)

Neural Network-based Vehicular Channel Estimation Performance: Effect of Noise in the Training Set

SA Ngorima[1,2,3][0000−0002−0775−3529], ASJ Helberg[1][0000−0001−6833−5163], and MH Davel[1,2,3][0000−0003−3103−5858]

[1] Faculty of Engineering, North-West University, South Africa
[2] Centre for Artificial Intelligence Research, South Africa
[3] National Institute for Theoretical and Computational Sciences, South Africa
aldringorima@gmail.com
albert.helberg@nwu.ac.za

Abstract. Vehicular communication systems face significant challenges due to high mobility and rapidly changing environments, which affect the channel over which the signals travel. To address these challenges, neural network (NN)-based channel estimation methods have been suggested. These methods are primarily trained on high signal-to-noise ratio (SNR) with the assumption that training a NN in less noisy conditions can result in good generalisation. This study examines the effectiveness of training NN-based channel estimators on mixed SNR datasets compared to training solely on high SNR datasets, as seen in several related works. Estimators evaluated in this work include an architecture that uses convolutional layers and self-attention mechanisms; a method that employs temporal convolutional networks and data pilot-aided estimation; two methods that combine classical methods with multilayer perceptrons; and the current state-of-the-art model that combines Long-Short-Term Memory networks with data pilot-aided and temporal averaging methods as post processing. Our results indicate that using only high SNR data for training is not always optimal, and the SNR range in the training dataset should be treated as a hyperparameter that can be adjusted for better performance. This is illustrated by the better performance of some models in low SNR conditions when trained on the mixed SNR dataset, as opposed to when trained exclusively on high SNR data.

Keywords: Channel estimation · deep learning · neural networks · CNN-Transformer · IEEE 802.11p · vehicular channels.

1 Introduction

The integration of network infrastructure with artificial intelligence (AI) is becoming more prevalent due to the increasing demands of data-intensive applications and the need for more efficient and reliable communication systems. Incorporating AI into these systems promises to improve network management, resource allocation, and overall system performance.

This convergence is notably evident in vehicular communication, which is an important component of intelligent transportation systems. Vehicular communication systems face unique problems due to their high mobility, rapidly varying environments, and significant latency requirements. One of the challenges in this domain is obtaining accurate channel estimates, which is critical to ensuring reliable communication between vehicles and infrastructure. Channel estimation involves determining the characteristics of a communication channel and monitoring its changes to adjust to varying conditions [9].

The dynamic nature of vehicular environments, characterised by rapidly changing channel conditions, poses a challenge to traditional channel estimation methods. These methods include methods such as least squares (LS) and minimum mean square error (MMSE), which rely on mathematically estimating the channel based on the received version of known data sent as a preamble to the signal. Although effective in more stationary environments, these techniques often struggle to keep up with the rapid changes inherent in vehicular channels [4]. This limitation highlights the need for more advanced adaptive approaches to channel estimation in these high-mobility scenarios.

In recent years, researchers have increasingly focused on using data symbols directly to estimate the channel in a method referred to as data pilot-aided (DPA) channel estimation. DPA estimation addresses the challenges posed by dynamic environments, especially in vehicular communication systems, where pilot signals are evidently insufficient. Unlike traditional methods that rely heavily on a fixed number of pilot signals for channel estimation, DPA estimation uses demapped data symbols as additional pilots. This approach allows for more flexible and adaptive tracking of channel variations without the need to allocate more resources for pilot signals, thereby improving efficiency and accuracy in rapidly changing channels. However, the performance of the DPA method can be significantly affected by errors introduced during the demapping process, which may limit its overall effectiveness [12]. To improve efficiency, DPA estimation is typically used with an error compensation scheme to reduce channel distortion and prevent errors from propagating to the next symbols in a frame. Two common error compensation approaches for channel estimation are spectral temporal averaging (STA) [2] which utilises the correlations of data symbols in time and frequency domains to estimate the channel and time-domain reliable test frequency domain interpolation (TRFI) [8], as described in more detail in Section 2.2. However, these methods still struggle in highly dynamic scenarios [3].

Deep learning (DL) has recently gained attention for its ability to improve the accuracy of channel estimates. A method utilising autoencoders and training on a high signal-to-noise ratio (SNR) dataset (SNR=40 dB) was proposed to correct DPA estimate errors in the frequency domain [6]. However, it did not account for temporal changes, limiting its performance. Furthermore, incorporating feedforward networks into classical methods such as STA and TRFI methods, as proposed in [4], has shown promise in improving estimation accuracy. These methods were also trained on high SNR datasets (SNR=30 dB). These methods offer valuable baselines. Recently, a long-short-term memory (LSTM)-

based architecture has been proposed to capture the temporal and frequency characteristics of vehicular channels [3]. This method was trained on a dataset generated at the SNR level of 40 dB and demonstrated significant improvements over previous estimators. However, this method is computationally expensive.

This study introduces the use of a mixed SNR dataset, where the training dataset includes data generated at different SNR levels. We hypothesise that training on a mixed SNR dataset helps an neural network (NN) generalise across a wider range of real-world conditions by exposing it to both noise-dominated and signal-dominated scenarios. This approach contrasts with the methodology adopted in several studies [4, 5, 3, 12], which propose that training the NN-based estimator using a dataset generated only at a high SNR level enhances its ability to generalise in any environment: at high SNR the impact of the channel is greater than the impact of noise, therefore it is expected that the NN develops better channel knowledge.

We evaluate the use of the mixed SNR dataset using a CNN-Transformer-based estimator [11] which is a hybrid model that combines a one-dimensional convolutional neural network (CNN) with a transformer architecture, a TCN-DPA estimator [10] that uses a temporal convolutional network (TCN) to correct the DPA propagation error, two estimators based on concatenating a multilayer perceptron (MLP) with a classical method, in this case STA-MLP [4] and TRFI-MLP [5], and the LSTM-DPA-TA method [3] which combines Long Short-Term Memory (LSTM) networks with Data Pilot-Aided (DPA) estimation and Temporal Averaging (TA).

2 Background

We describe the system model, well-known channel estimation schemes in vehicular communications, and some of the NN-based channel estimation techniques currently used. We also provide a brief description of the IEEE 802.11p physical layer structure that is used in all simulations, along with a description of its specifications.

2.1 System Model

In this paper, we adopt the IEEE 802.11p physical layer structure, which is specifically designed for vehicular communication systems. This section follows a detailed discussion of the IEEE 802.11p specifications as outlined in Ngorima et al. [10]. The IEEE 802.11p standard uses Orthogonal Frequency Division Multiplexing (OFDM) to transfer data by dividing the available bandwidth into several subcarrier frequencies for simultaneous transmission of multiple signals. 52 of the 64 available subcarriers are used for data transmission and pilot symbols. The remaining subcarriers are for guard bands and direct current (DC) offset. To illustrate this structure, Figure 1 shows the IEEE 802.11p OFDM frame, indicating pilot and data subcarriers.

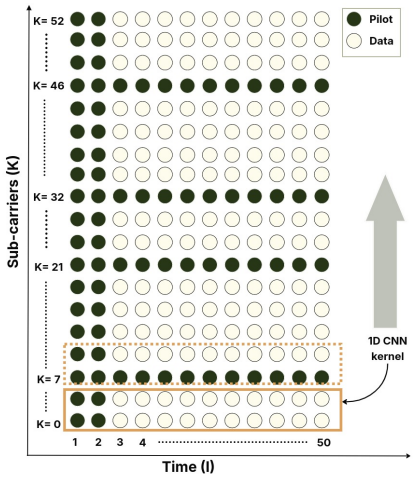


Fig. 1. The IEEE 802.11p frame structure, illustrating the allocation of subcarriers for pilot and data transmission.

In this work, we consider a frame structure that consists of two preamble symbols for signal detection and timing synchronisation, followed by a signal field that contains transmission parameters. The received signal on subcarrier $[k]$ at time i can be represented as

$$Y_i[k] = H_i[k]X_i[k] + N_i[k], \tag{1}$$

where $H_i[k]$, $X_i[k]$ and $N_i[k]$ denote the respective channel response, transmitted signal, and additive Gaussian noise (AWGN), respectively. Note that in vehicular channels H_i varies in both the time and frequency domains.

2.2 Well-known Vehicular Channel Estimation Schemes

This section describes some of the main vehicular channel estimation schemes that have been widely used and studied in the literature. These methods were selected based on their effectiveness in dynamic environments and their foundational role in the development of more advanced NN-based estimators.

DPA Estimation: The first step of the DPA process, as described by Pan et al. [12], is to calculate the initial estimate of the frame. This is done by applying the least squares (LS) estimation to the preamble as shown below:

$$\hat{h}_0[k] = \frac{y_1^{(p)}[k] + y_2^{(p)}[k]}{2p[k]}, \tag{2}$$

where $y_1^{(p)}[k]$ and $y_2^{(p)}[k]$ are the known orthogonal preambles received on the $[k]$ subcarrier and $p[k]$ is the OFDM data symbol transmitted over the k-th subcarrier. The next OFDM symbols in the frame are equalised using the initial estimate as the starting point as follows:

$$y_i^{\text{eq}}[k] = \frac{y_i[k]}{\hat{h}_{i-1}^{\text{DPA}}[k]}, \hat{h}_0^{\text{DPA}}[k] = \hat{h}_0[k]. \tag{3}$$

The equalised symbol is demapped and mapped to its closest constellation point as:

$$d_i[k] = \Re\left(\frac{y_i[k]}{\hat{h}_{i-1}^{\text{DPA}}[k]}\right), \hat{h}_0^{\text{DPA}}[k] = \hat{h}_{\text{LS}}[k], \tag{4}$$

Finally, the DPA channel is updated:

$$\tilde{h}_{\text{DPA}_i}[k] = \frac{y_i[k]}{d_i[k]} \tag{5}$$

STA Estimation Scheme: STA uses both spectral (frequency) and temporal correlations within data symbols to estimate the channel [4]. STA can also be viewed as a recursive filtering process that combines multiple channel estimates into a more accurate estimate. The first step is to calculate a channel estimate in the frequency domain $\hat{h}i^{\text{FD}}[k]$ as follows:

$$\hat{h}i^{\text{FD}}[k] = \sum \lambda = -\beta^{\beta}\omega_\lambda \hat{h}_i^{\text{DPA}}[k+\lambda],\ \omega_\lambda = \frac{1}{2\beta+1}, \tag{6}$$

where $2\beta + 1$ represents the number of subcarriers that are considered. Subsequently, a temporal average is computed to arrive at the final STA channel estimate, $\hat{h}i^{\text{STA}}[k]$:

$$\hat{h}i^{\text{STA}}[k] = \left(1 - \frac{1}{\alpha}\right)\hat{h}i - 1^{\text{STA}}[k] + \frac{1}{\alpha}\hat{h}_i^{\text{FD}}[k], \tag{7}$$

where α is a smoothing parameter that controls the weight given to current and previous estimates.

TRFI Estimation Scheme: TRFI leverages the high correlation within OFDM symbols to improve the accuracy of the channel estimation [5]. The process starts by creating multiple channel estimates for each symbol using the DPA method. The received symbol is equalised using the estimates of the current and previous symbols ($\hat{h}_i^{\text{DPA}}[k]$) and ($\hat{h}_{i-1}^{\text{DPA}}[k]$) respectively.

$$\begin{aligned} y_{i-1}^{\text{eq}'}[k] &= \frac{y_{i-1}[k]}{\hat{h}_i^{\text{DPA}}[k]}, \\ y_{i-1}^{\text{eq}''}[k] &= \frac{y_{i-1}[k]}{\hat{h}_{i-1}^{\text{DPA}}[k]}, \end{aligned} \tag{8}$$

Subsequently, the equalised symbols are remapped to their corresponding constellation points, $d'_{i-1}[k]$ and $d''_{i-1}[k]$. TRFI then categorises subcarriers into reliable and unreliable sets based on a reliability test between the remapped symbols. Subcarriers with consistent remapped symbols, that is, $d'_{i-1}[k] = d''_{i-1}[k]$, are considered reliable, while those with discrepancies are classified as unreliable. TRFI then interpolates channel estimates for unreliable subcarriers by using the reliable subcarriers as anchor points. This interpolation process effectively fills in the gaps in channel knowledge, resulting in a more accurate overall channel estimate. A detailed discussion on the interpolation procedure is given in [4].

2.3 MLP-based Estimators

To enhance the performance of conventional STA and TRFI estimators, Gizzini et al. [4] introduced MLP-based channel estimation methods. In these methods, MLP layers are added to the STA and TRFI processes, allowing the model to learn complex patterns in vehicular channels and thereby improve the accuracy of the estimation. The methods STA-MLP and TRFI-MLP initially conduct DPA estimation, proceed with STA or TRFI estimation and subsequently feed the results into the MLP layers.

2.4 TCN-DPA Estimator

As proposed in [10], the TCN-DPA estimator integrates a Temporal Convolutional Network (TCN) with DPA estimation to enhance channel estimation in dynamic environments. The TCN processes the received OFDM symbols in the frequency domain, treating subcarriers as time steps. The extracted features are then passed to the DPA process, where the previous TCN output, $\hat{h}_{i-1}^{\mathrm{TCN}}[k]$, is used to equalise the current received symbol. After equalisation, the symbol is demapped and remapped to the nearest constellation point to obtain the final channel estimate, $\hat{h}_{i}^{\mathrm{TCN}}[k]$. This estimate is updated iteratively using the DPA process. This approach leverages the ability of the TCN to model long-range dependencies across subcarriers, making it effective for complex channel conditions.

2.5 LSTM-DPA-TA Estimator

The LSTM-DPA-TA estimator [3] integrates an LSTM network with DPA and Temporal Averaging (TA) techniques to enhance channel estimation in vehicular environments. In this approach, the LSTM processes the received OFDM symbols to extract key features, which are then passed into the DPA module. The resulting channel estimates are further refined using the TA technique. In the LSTM-DPA-TA method, OFDM symbols are treated as time steps, with subcarriers considered as features within each time-step sequence. This design enables the model to effectively capture frequency dependencies and improve the accuracy of channel estimation under dynamic conditions.

2.6 CNN-Transformer

The CNN-Transformer is a recently proposed hybrid architecture for vehicular channel estimation in [11]. This approach combines the strengths of CNNs and Transformer networks to analyse vehicular channel data comprehensively. The channel data can be represented in both the time and frequency domains. The frequency domain represents the subcarriers of the OFDM symbols, while the time domain represents the sequence of OFDM symbols transmitted over time. Each OFDM symbol is transmitted across multiple subcarriers. The frequency domain captures the spatial characteristics of the channel across OFDM subcarriers, while the time domain reflects the temporal dynamics as OFDM symbols are transmitted sequentially. In this CNN-Transformer architecture, subcarriers are treated as the 'time steps'. The architecture leverages CNNs to extract local features from subcarriers and uses Transformer layers to capture global dependencies across the frequency domain. The CNN component applies 1D convolutions to capture local patterns, while the Transformer component uses self-attention mechanisms to analyse patterns between subcarriers globally.

3 Methodology

This section outlines the methodological framework used in this study. We start with a detailed description of the datasets used for training and evaluation, specifically focussing on two approaches: the mixed SNR dataset and the high SNR dataset. Following this, we discuss the data preprocessing steps applied to transform complex received symbols into a format suitable for input into NNs.

3.1 Data Preparation

We simulate vehicle-to-vehicle communication following the 'Vehicle-To-Vehicle Expressway Same Direction with Wall' (VTV-SDWW) channel model [7]. Our simulation specifically modelled vehicles moving at 100 km/h, with a Doppler shift of 550 Hz. We used 16QAM (16-Quadrature Amplitude Modulation) modulation for data transmission in this urban environment scenario. This modulation scheme encodes the data by varying the amplitude and phase of the carrier signal.

Mixed SNR Dataset The mixed SNR dataset represents a wide range of noise introduced on the VTV-SDWW channel. For this dataset, we simulate 18,000 time-specific frames with SNR values ranging from 0 to 40 dB in increments of 5 dB. Each SNR level includes 2,000 frames, with 50 OFDM symbols spread across 52 active subcarriers (48 data and 4 pilot subcarriers). During training, each model is exposed to the entire range of SNR levels. The validation set is created by reserving 25% of this training data to monitor the performance of the model during training. An independent test set consisting of 2,000 frames is generated separately from the training sets. This test set includes frames at

SNR levels of 0, 5, 10, 15, 20, 25, 30, 35, and 40 dB, ensuring that the evaluation correctly reflects the performance across different SNR conditions. The same test set is used to test all the models regardless of how they are trained.

High SNR Dataset The high SNR dataset focusses on a fixed SNR level, representing less noisy or nearly ideal channel conditions, as opposed to the mixed SNR dataset. This dataset is used to train models with the assumption that training NNs in an environment with a clearly defined channel can lead to better generalisation even in noisy environments [3–5, 12]. In this work, the high SNR dataset consists of 18,000 time-specific frames, all generated at a 40 dB SNR level, with the same frame structure as the mixed SNR data. As before, 25% of the training data is set aside for validation.

Data preprocessing To preprocess the data for NN models, we transform the complex 16QAM symbols received across subcarriers and time slots into a real-valued format. Each received symbol consists of both real and imaginary components. The following steps are performed to prepare the data:

1. **Separate Real and Imaginary Components:** Decompose each complex 16QAM symbol into its real and imaginary parts. Handle the data as two separate real-valued matrices instead of a single complex-valued matrix. For a given OFDM symbol, where $y[k] = r[k] + j \cdot i[k]$ represents the received complex value at subcarrier k, extract $r[k]$ as the real part and $i[k]$ as the imaginary part.
2. **Interleave Components:** After separation, interleave the real and imaginary components. This means arranging the real and imaginary parts in an alternating sequence across the time slots. By interleaving, a unified structure that preserves the relationship between the real and imaginary parts within the input data is created.
3. **Form the Input Matrix:** Structure the interleaved data into a matrix with dimensions 52×100, where 52 represents the number of subcarriers and 100 represents the sequence of interleaved real and imaginary components across the time slots. This transformation enables the NN to process the complex-valued input as a series of real-valued inputs.
4. **Prepare Input Data for NN models:** The resulting matrix is now a real-valued representation of the original complex data, ready for input into NN models.

The transformed input matrix described above is visually represented in Figure 2. Each block in the figure represents the interleaved real and imaginary components of the received 16QAM symbols for a particular subcarrier across the OFDM symbols. The kernel size of 3 shown in the figure indicates the width of the sliding window used by the CNN during the convolution process across the subcarriers.

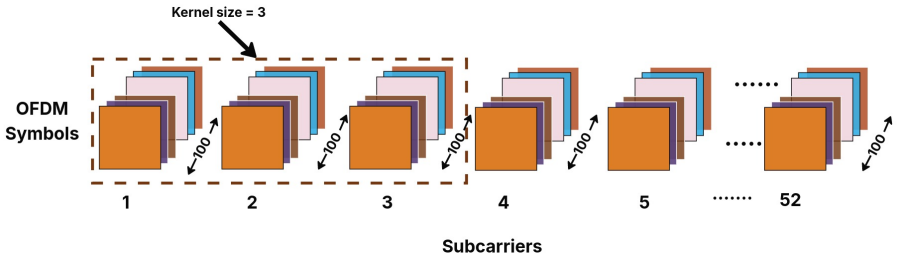


Fig. 2. Transformed IEEE 802.11p frame structure used as input to deep learning models, composed of 100 interleaved complex symbols and 52 subcarriers.

3.2 Hyperparameter Optimisation

We reimplement most of the NN-based estimators discussed in the previous sections to ensure compatibility with our experimental setup and to be able to optimise each individually for the specific datasets used. Specifically, the CNN-Transformer, TCN-DPA, STA-MLP, and TRFI-MLP models were reimplemented from scratch. The LSTM-DPA-TA model is implemented using existing code available online. To verify the accuracy of our reimplementations, we compare the performance of our implementations against the benchmark results reported in the original studies, ensuring that they perform as expected.

The final architecture of each model is determined through a hyperparameter tuning process, to identify the optimal configurations for each model. We optimise the hyperparameters of all models in both datasets using Optuna, a Bayesian optimisation framework that efficiently searches the hyperparameter space using a tree-structured Parzen estimator (TPE) [1]. This method systematically explores different hyperparameter configurations while conserving computational resources by early termination of underperforming trials.

For each model, we conduct multiple trials with different random seeds to ensure that results are not biased by a particular initialisation. Specifically, we conduct 50 trials per model, each with 3 seeds, allowing us to assess the robustness of the selected hyperparameters. The search space for each model was designed to cover a wide range of potential configurations. The key hyperparameters include the learning rate, the number of layers, the layer size (kernel size, number of attention heads, or layer width, depending on model), the learning rate scheduler parameters, and hyperparameters controlling regularisation through dropout. During the optimisation process, we monitor the convergence of validation loss. Early stopping is employed to prevent overfitting, stopping training when the validation loss ceases to improve for a predefined number of epochs.

Table 1 provides a summary of the best hyperparameters obtained for each model across the datasets. In addition to the parameters in the table, batch normalisation is applied to enhance model generalisation. The Adam optimiser and a batch size of 128 was used for TCN-DPA, LSTM-DPA-TA, STA-MLP and

TRFI-MLP. For the CNN-Transformer model, we use the AdamW optimiser and manually set the batch size to 16.

Table 1. Optimised hyperparameters for CNN-Transformer, TCN-DPA, STA-MLP, TRFI-MLP, and LSTM-DPA-TA on the mixed SNR and high SNR datasets.

Hyperparameter	**Search Space**	**Mixed SNR**	**40 dB**
CNN-Transformer			
Learning rate	[1e-5 to 1e-2]	0.001	0.001
Transformer layers	[1 to 4]	2	4
Number of attention heads	[1 to 4]	2	4
Hidden dimension	[64 to 128]	128	128
Dropout rate	[0.001 to 0.3]	0.1	0.25
Number of epochs	[50 to 200]	110	200
Number of CNN layers	[1 to 5]	2	4
CNN kernel size	[2 to 5]	3	3
TCN-DPA			
Learning rate	[1e-5 to 1e-2]	0.0006	0.003
Number of Layers	[1 to 5]	4	4
Kernel Size	[2 to 5]	2	2
Dropout	[10^{-5} to 0.5]	0.17	0.01
StepLR Step Size	[10 to 50]	21	17
StepLR Gamma	[0.5 to 1]	0.9	0.8
Epochs	[0 to 200]	156	100
STA-MLP			
Learning rate	[1e-5 to 1e-2]	0.001	0.001
Number of layers	[1 to 5]	2	3
Size of hidden layer 0	[5 to 30]	29	15
Size of hidden layer 1	[5 to 30]	27	15
Size of hidden layer 2	[5 to 30]	N/A	15
Total training epochs	[50 to 500]	133	300
TRFI-MLP			
Learning rate	[1e-5 to 1e-2]	0.0004	0.001
Number of layers	[1 to 5]	3	3
Size of hidden layer 0	[5 to 30]	23	15
Size of hidden layer 1	[5 to 30]	29	15
Size of hidden layer 2	[5 to 30]	21	15
Total training epochs	[50 to 500]	130	160
LSTM-DPA-TA			
Learning rate	[1e-5 to 1e-1]	0.004	0.01
LSTM size	[64 to 128]	128	128
StepLR step size	[1 to 50]	35	10
StepLR step gamma	[0.1 to 1]	0.7	0.8
Training epochs	[50 to 500]	160	500

4 Results

This section evaluates the performance of channel estimators using BER as the primary metric. We compare the effectiveness of two training approaches: one using a mixed SNR dataset and the other using a high SNR dataset. The evaluated architectures are CNN-Transformer, TCN-DPA, STA-MLP, TRFI-MLP, and LSTM-DPA-TA.

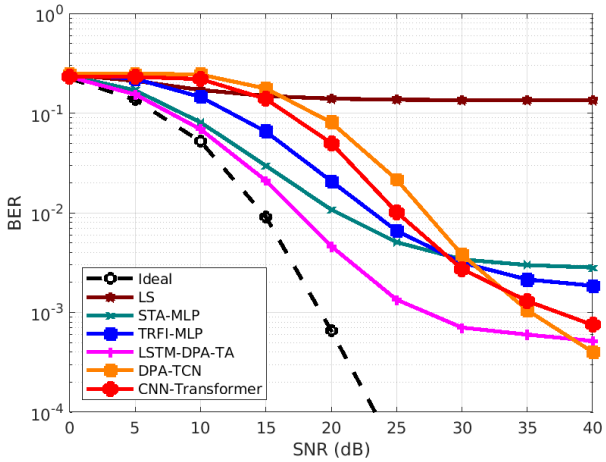
4.1 BER

BER is a crucial performance measure in digital communication. It compares transmitted and received bit sequences to assess system reliability. In channel estimation, a lower BER means more accurate signal reconstruction and fewer errors in decoded data. The following analysis details the BER performance of each model at different levels of SNR, providing a direct comparison between the two training approaches.

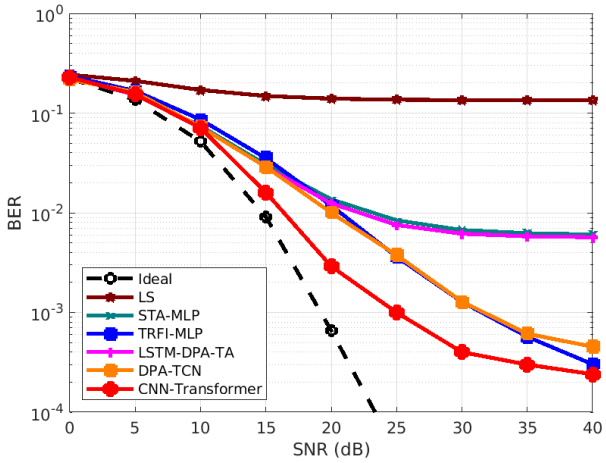
Figure 3 presents the BER performance of the channel estimators trained using a high SNR dataset (40 dB) and a mixed SNR dataset, respectively. The ideal curve, represented by the dotted black line in these figures, represents the best possible performance in channel estimation, where the received signal is assumed to be perfectly equalised without any channel estimation errors. This curve serves as a theoretical lower bound for the BER, showing the performance in a scenario where only additive white Gaussian noise (AWGN) is present and the channel effects are perfectly compensated for. The closer the BER of a model is to the ideal curve, the better its performance in accurately estimating the channel and mitigating the effects of noise. LS serves as the lower performance baseline.

In Figure 3a, we observe several key trends across models trained on an high SNR dataset: In the low SNR range (0-10 dB), the STA-MLP and the LSTM-DPA-TA models perform relatively well, maintaining a lower BER compared to other models. As we move into the mid SNR range (15-25 dB), the CNN-Transformer and DPA-TCN models show an improvement, particularly from 15 dB onwards, with performance becoming more pronounced from 20 dB. This improvement at high SNR highlights that these models have effectively adapted to high SNR conditions. The TRFI-MLP model also improves in this range, although it lags behind the CNN-Transformer and DPA-TCN models, indicating some limitations in its ability to adapt to low SNR conditions. In the high SNR range (30-40 dB), the CNN-Transformer and DPA-TCN models continue to show further significant reductions in BER. The LSTM-DPA-TA model continues to outperform other models across all SNR levels, showing its effectiveness in less noisy conditions. In contrast, STA-MLP shows limited improvement beyond 20 dB, indicating that it struggles to capitalise on the reduced noise at higher SNR levels.

In the mixed SNR training scenario depicted in Figure 3b, we observe that CNN-Transformer and DPA-TCN models exhibit excellent performance across the entire SNR range with CNN-Transformer outperforming all models. The



(a) High SNR training dataset



(b) Mixed SNR training dataset

Fig. 3. Comparison of BER performance for various channel estimators trained on high SNR and mixed SNR datasets.

LSTM-DPA-TA and STA-MLP models perform well at lower SNR levels (0 to 15 dB) but show reduced performance as the SNR increases, indicating diminished capabilities in high SNR conditions when trained on mixed SNR data. We can also observe the improved performance of the TRFI-MLP estimator across the entire SNR range tested compared to its performance when trained on the high SNR dataset. The unsatisfactory performance by LSTM-DPA-TA and STA-MLP

indicates that while these models can handle varying noise conditions, they may not leverage mixed SNR training as effectively as TRFI-MLP, CNN-Transformer and DPA-TCN.

Models trained on mixed SNR data exhibit lower BER even in low-SNR scenarios, compared to those trained on high SNR data. Mixed SNR training appears to be beneficial for models such as TRFI-MLP, TCN-DPA, and CNN-Transformer, which show a significant improvement in low SNR conditions than when trained on high SNR. In contrast, the LSTM-DPA-TA and STA-MLP models demonstrate better performance when trained on a high SNR dataset, suggesting that these models make use of the better channel statistics available at high SNR. This is analysed in more detail below.

4.2 Difference in BER Between Models Trained on Mixed SNR and High SNR Datasets

Figure 4 illustrates the difference in BER between models trained on high SNR datasets and those trained on a mixed SNR dataset. The graph clearly shows how the training dataset influences the performance of various models across different SNR ranges. In the low SNR range (0-10 dB), the CNN-Transformer and TCN-DPA models exhibit the highest positive delta, peaking around 10 dB. This indicates a substantial improvement in performance when these models are trained on a mixed SNR dataset compared to a high SNR dataset. A positive delta suggests that mixed SNR training better equips these models to generalise in low SNR conditions. TRFI-MLP also shows a noticeable positive delta, although less pronounced, indicating that mixed SNR training offers some advantages in low SNR environments. In contrast, the LSTM-DPA-TA and STA-MLP models display a slightly negative delta across this range, implying that training on high SNR datasets offers a marginal performance advantage in these specific models.

As the SNR increases (15-25 dB), the positive delta values for the CNN-Transformer and TCN-DPA models start to decrease but remain positive, particularly around 15 dB. This indicates that while the benefits of mixed SNR training diminish slightly, these models still gain performance advantages in this SNR range. TRFI-MLP continues to maintain positive delta values, demonstrating that mixed SNR training consistently improves its performance at varying SNR levels. Meanwhile, the LSTM-DPA-TA model continues to show a negative delta, particularly at the higher end of the mid-SNR range, around 20-25 dB. Similarly, the STA-MLP model exhibits a slight negative delta in this range.

In the high SNR range (30-40 dB), the delta values for most models, including the CNN-Transformer, TCN-DPA, and TRFI-MLP, approach zero. This suggests that as the SNR increases and the impact of noise decreases, the performance difference between models trained on high SNR and mixed SNR datasets becomes negligible.

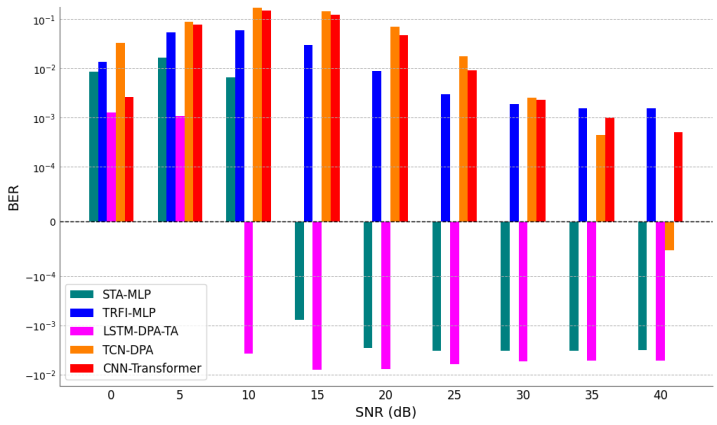


Fig. 4. Difference in BER between models trained on mixed SNR and high SNR datasets.

5 Conclusion

This study investigated the effectiveness of training NN-based channel estimators using mixed SNR datasets compared to high SNR datasets. Our results demonstrate that training with mixed SNR data significantly improves the generalisation of various estimators, especially in low SNR conditions. In particular, models such as the CNN-Transformer, DPA-TCN and TRFI-MLP exhibited substantial improvements in BER across the entire SNR range when trained on mixed SNR data compared to when trained on high SNR. Among the models tested, the CNN-Transformer, when trained on mixed SNR data, outperformed other estimators, including the current state-of-the-art LSTM-DPA-TA. However, it is important to note that some models, such as LSTM-DPA-TA and STA-MLP, showed reduced performance when trained on mixed SNR data. Mixed SNR training improves performance and generalisation across SNR levels for some models. The exact reason why only certain models benefit remains unclear and requires further investigation.

These results indicate the importance of considering the SNR range as an important hyperparameter during training, rather than following the current practice of using only high SNR training data. The channel model used in this study is an example where the impact of mobility on the channel is more significant compared to other channels with lower mobility. As a result, the tested models should be able to generalise well to those channels. This will be verified in future research that will investigate the impact of mixed SNR training on other vehicular channel models.

Acknowledgements The authors are grateful to NITheCS and the Telkom CoE at NWU for their support of this research.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-Generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2623–2631 (2019)
2. Fernandez, J.A., Borries, K., Cheng, L., Vijaya Kumar, B.V.K., Stancil, D.D., Bai, F.: Performance of the 802.11p Physical Layer in Vehicle-to-Vehicle Environments. IEEE Transactions on Vehicular Technology **61**(1), 3–14 (2012). https://doi.org/10.1109/TVT.2011.2164428
3. Gizzini, A.K., Chafii, M., Ehsanfar, S., Shubair, R.M.: Temporal Averaging LSTM-based Channel Estimation Scheme for IEEE 802.11p Standard. In: 2021 IEEE Global Communications Conference (GLOBECOM). pp. 01–07. IEEE (2021)
4. Gizzini, A.K., Chafii, M., Nimr, A., Fettweis, G.: Deep Learning Based Channel Estimation Schemes for IEEE 802.11p Standard. IEEE Access **8**, 113751–113765 (2020)
5. Gizzini, A.K., Chafii, M., Nimr, A., Fettweis, G.: Joint TRFI and Deep Learning for Vehicular Channel Estimation. In: 2020 IEEE Globecom Workshops (GC Wkshps). pp. 1–6. IEEE (2020)
6. Han, S., Oh, Y., Song, C.: A Deep Learning Based Channel Estimation Scheme for IEEE 802.11p Systems. In: ICC 2019-2019 IEEE International Conference on Communications (ICC). pp. 1–6. IEEE (2019)
7. Jiang, D., Delgrossi, L.: IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments. In: VTC Spring 2008 - IEEE Vehicular Technology Conference. pp. 2036–2040 (2008). https://doi.org/10.1109/VETECS.2008.458
8. Kim, Y.K., Oh, J.M., Shin, Y.H., Mun, C.: Time and Frequency Domain Channel Estimation Scheme for IEEE 802.11p. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). pp. 1085–1090. IEEE (2014)
9. Li, Y.: Pilot-Symbol-Aided Channel Estimation for OFDM in Wireless Systems. IEEE Transactions on Vehicular Technology **49**(4), 1207–1215 (2000). https://doi.org/10.1109/25.875230
10. Ngorima, S.A., Helberg, A., Davel, M.H.: A Data Pilot-Aided Temporal Convolutional Network for Channel Estimation in IEEE 802.11p Vehicle-to-Vehicle Communications. In: Southern Africa Telecommunication Networks and Applications Conference (SATNAC) (2024)
11. Ngorima, S.A., Helberg, A., Davel, M.H.: Hybrid CNN-Transformer Model for Channel Estimation in Vehicular Communications. In: Deep Learning Indaba Conference (2024), *Work in Progress*
12. Pan, J., Shan, H., Li, R., Wu, Y., Wu, W., Quek, T.Q.: Channel Estimation Based on Deep Learning in Vehicle-to-Everything Environments. IEEE Communications Letters **25**(6), 1891–1895 (2021)

Simplified Temporal Convolutional-based Channel Estimation for a WiFi Vehicular Communication Channel

Simbarashe Aldrin Ngorima[1,2,3], Albert Helberg[1], Marelie H. Davel[1,2,3]
[1]*Faculty of Engineering, North-West University, South Africa*
[2]*Centre for Artificial Intelligence Research (CAIR), South Africa*
[3]*National Institute for Theoretical and Computational Sciences (NITheCS), South Africa*
36450057@mynwu.ac.za
albert.helberg@nwu.ac.za, marelie.davel@nwu.ac.za

***Abstract*—Channel estimation in vehicular communication is a crucial element in the advancement of intelligent transportation systems. However, the use of pilot signals in the IEEE 802.11p standard is insufficient for accurate channel estimation in high-mobility scenarios. Data pilot-aided (DPA) estimation helps address this, but suffers from demapping errors. We propose a simplified Temporal Convolutional Network-based estimator (DPA-TCN) trained on a mixed signal-to-noise ratio dataset to improve estimation performance and reduce computational complexity. Our DPA-TCN estimator achieves a bit error rate comparable to a state-of-the-art long-short-term memory network with DPA and temporal averaging (LSTM-DPA-TA) while reducing the complexity of the model by approximately 65%.**

***Index Terms*—channel estimation, deep learning, IEEE 802.11p, TCN, vehicle-to-vehicle, wireless communications**

I. Introduction

In intelligent transportation systems (ITS), ensuring reliable and efficient vehicular communication is crucial for enhancing overall transportation safety and performance. Accurate and timely data transmission between vehicles and infrastructure is heavily dependent on precise channel estimation. However, this task is challenging due to the dynamic nature of vehicular channels. The channel in these networks is considered doubly selective, indicating that the channels vary in time and frequency. These variations are caused by two main factors: multipath propagation, where signals reflect off various surfaces, and the high mobility of vehicles, leading to rapid changes in the communication environment. Together, these factors complicate the process of accurate channel estimation, making it a significant challenge in the development of ITS. The IEEE 802.11p standard [1], approved and specified for vehicular communication to support ITS, faces limitations in scenarios that have high mobility. Specifically, the number of pilot subcarriers allocated in this standard tends to be insufficient, and relying solely on these pilots has proven to be less effective [2].

Increasing the number of pilots for transmission reduces spectral efficiency [3]. To address the problem of insufficient pilots, a method known as data pilot-aided (DPA) estimation is commonly used [4]. This technique involves using data subcarriers for channel estimation. However, the presence of noise and multipath effects causes demapping errors which make it difficult to obtain accurate data pilots using only DPA as an estimation method [5].

In recent years, researchers have suggested using estimators that use deep learning (DL) to capture important features in vehicular channels [2], [3], [6], [7]. These estimators are based on the DPA method. Among these DL estimators, LSTM-DPA-TA [8] is one of the state-of-the-art (SOTA) in terms of bit error rate (BER) performance. However, the computational complexity of this estimator restricts the adaptation of this estimator in real-world applications.

In this work, we propose a simplified Temporal Convolutional Network (TCN) estimator that takes DPA estimation as input for the estimation of vehicular channels. Our aim is to address both the computational and BER performance challenges in vehicular channels by leveraging the TCN's parallel nature and one-dimensional convolutional operations. Furthermore, we train the TCN estimator on a mixed SNR dataset that incorporates various channel variations based on results from [9], rather than using a dataset generated solely at high SNR as in previous studies [2], [3], [6], [8], [10]. The DPA process solves the insufficient pilot problem, while the TCN uses the DPA estimates to accurately estimate the channel state. We evaluate our proposed DPA-TCN estimator on a realistic vehicle-to-vehicle same direction with wall (VTV-SDWW) tapped delay line (TDL) vehicular channel model specified in [11]. Performance assessment is based on BER, Normalised Mean Square Error (NMSE) and complexity metrics.

II. System Model

This section provides an introduction to the IEEE 802.11p standard specification and discusses various channel estimation techniques following [12]. We will cover the widely used least squares (LS) method, which is considered a baseline in our work. Furthermore, we will dive into the DPA method, commonly utilised for initial channel estimation in various estimators, to highlight the importance of DL post-processing.

Lastly, we will compare the performance of these methods with the SOTA, LSTM-DPA-TA.

The IEEE 802.11p standard operates in the frequency band ranging from 5.0 GHz to 5.9 GHz, utilising a 10 MHz bandwidth [11]. It employs Orthogonal Frequency Division Multiplexing (OFDM) with 64 subcarriers for data transmission. Of these 64 subcarriers, 52 are used for data and pilots, while the remaining 12 at indices $\{-32,\ldots,-27,0,27,\ldots,31\}$ serve as guard bands and DC offsets. Specifically, four pilot subcarriers are located at indices $\{-21,-7,\ 7,\ 21\}$, with the remaining 48 subcarriers designated for data transmission.

The IEEE 802.11p frame structure includes a preamble for signal detection, timing synchronisation, and channel estimation, followed by a signal field for transmission parameters. This work assumes perfect synchronisation and considers a frame structure with two extended preambles at the start, followed by *I* OFDM data symbols.

Given this structure, the signal received on the subcarrier $[k]$ at time i can be expressed as

$$Y_i[k] = H_i[k]X_i[k] + N_i[k], \tag{1}$$

where $X_i[k]$, $H_i[k]$ and $N_i[k]$ are the transmitted symbol, the channel frequency response and the Gaussian noise, respectively.

In this work, we investigate the VTV-SDWW vehicular channel model [11]. This model simulates communication between two vehicles moving in the same direction, separated by a central wall, at a distance of 300-400 metres. In this scenario, vehicles travel at 100 km/h with a Doppler shift of f_d = 550 Hz, using 16QAM modulation.

A. Least Squares (LS) Estimation

The LS estimation approach is a basic method commonly used in wireless communications. The LS approach uses two adjacent received symbols, as shown below:

$$\hat{H}_{\text{LS}}[k] = \frac{Y_1^{(p)}[k] + Y_2^{(p)}[k]}{2p[k]}, \tag{2}$$

where $p[k]$ is the known preamble, $Y_1^{(p)}[k]$ and $Y_2^{(p)}[k]$ are the preambles received in the $[k]$ subcarrier.

B. DPA

DPA addresses the pilot limitation issue in vehicular communications, without altering the IEEE 802.11p standard frame structure by introducing data pilots. As a result, DPA is recognised as the primary channel estimation method in vehicular communications [2]. First, DPA obtains an initial estimate of the frame using the LS estimate at the preambles (as shown in (2)).

Then, the equalised symbols $Y_i^{\text{eq}}[k]$ are obtained for the subsequent symbols by using the previous estimate to equalise the current symbol as follows:

$$Y_i^{\text{eq}}[k] = \frac{Y_i[k]}{\hat{H}_{i-1}^{\text{DPA}}[k]}, \hat{H}_0^{\text{DPA}}[k] = \hat{H}_{\text{LS}}[k]. \tag{3}$$

The equalised data symbol $y_i^{eq}[k]$ is remapped to the nearest constellation point $d_i[k]$ using the remapping operation $\Re(\cdot)$

$$d_i[k] = \Re\left(\frac{Y_i[k]}{\hat{H}_{i-1}^{\text{DPA}}[k]}\right), \hat{H}_0^{\text{DPA}}[k] = \hat{H}_{\text{LS}}[k], \tag{4}$$

The DPA channel estimate is then updated as follows:

$$\hat{H}_i^{\text{DPA}}[k] = \frac{Y_i[k]}{d_i[k]}. \tag{5}$$

C. LSTM-DPA-TA

The LSTM in this estimator uses the received OFDM symbol as input, performs the DPA process on the LSTM output, and then applies a time-average filter. The LSTM-DPA-TA method was developed in [8] and was evaluated on several channels, including the VTV-SDWW channel used in this work. We are using the LSTM-DPA-TA estimator for comparison purposes with our proposed DPA-TCN estimator.

III. Proposed DPA-TCN Estimator

This section outlines the architecture of the DPA-TCN estimator proposed in this work.

A. Temporal Convolutional Networks (TCNs)

TCNs are designed to handle sequential data by leveraging techniques from Convolutional Neural Networks (CNNs), as introduced by LeCun [13]. TCNs use causal and dilated convolutions along with residual connections to capture sequential dependencies. This is done to capture temporal dependencies within the input data using one-dimensional kernels.

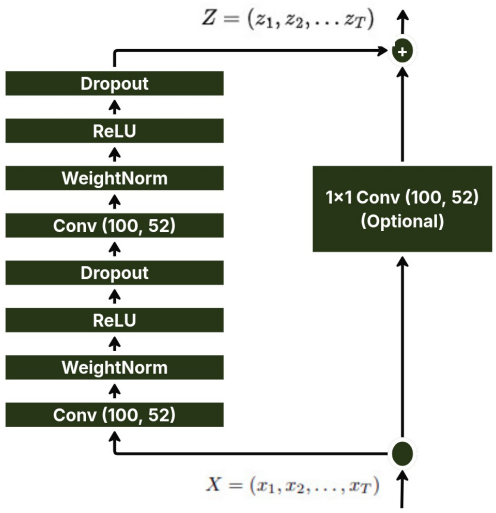


Fig. 1: TCN Residual block where 100 are the interleaved real and imaginary parts considered as features and 52 are complex active subcarriers which are considered as timesteps [14]

The goal is to predict the outputs $z_0, \ldots, z_T$ based on the input sequence $x_0, \ldots, x_T$. To predict z_t at time t, only the inputs $x_0, \ldots, x_t$ are used. To keep the input and output dimensions consistent, TCNs apply padding of length (kernel

size - 1) at the start of the frame or the input sequence. Causal convolutions avoid future leakage by only convolving an output at time t with elements from time t or earlier. An optional residual connection allows the input to be added to the output if the output has dimensions different from the input [14].

Furthermore, the residual block typically includes dilated causal convolution layers, nonlinearity layers, and dropout layers after each convolutional layer to effectively prevent overfitting and enhance the model's generalisation capabilities.

B. Dataset Generation

The dataset for training the proposed DPA-TCN comprises 18,000 time specific frames with SNR levels ranging from 0 dB to 40 dB in 5 dB steps. We generate 2,000 frames at each SNR level, each containing transmitted and received versions of 50 OFDM symbols. Each OFDM symbol has complex values for 52 active subcarriers (48 data and 4 pilot subcarriers). The 16QAM symbols received are separated and interleaved per subcarrier and time slot, resulting in an input matrix of real values with dimensions 52×100, and an output matrix (excluding pilot signals) of 48×100. These samples are batched for processing.

The dataset for training the LSTM-DPA-TA and TCN-DPA estimator is generated as above at a fixed SNR of 40dB.

For all the estimator models, 12,000 frames are used for training, 4,000 frames for validation, and an independent set of 2,000 frames for testing.

Note that in this work, we treat subcarriers as time steps. The input of the TCN is a 3D tensor of shape (N, C, T), where N is the batch size, C is the number of features (100 features) and T is the number of time steps (52 subcarriers). For a batch size of 128, the input tensor shape is $(128, 100, 52)$. A one-dimensional kernel size of 2 processes two consecutive time steps across the input sequence.

C. TCN estimator

This section describes the proposed DPA-TCN estimator where the input is a DPA-estimated frame $\hat{H}_i^{\text{DPA}}[k]$ which consists of $\hat{H}_i^D$ (estimated data subcarriers) and $\hat{H}_i^P$ (estimated pilot subcarriers) obtained following (2) to (5).

The proposed DPA-TCN estimator is shown in Fig. 2. The final channel estimate $\hat{H}_i^{\text{TCN}}[k]$ is updated based on (6):

$$\hat{H}_i^{\text{TCN}}[k] = f_{\text{TCN}}(\hat{H}_{i-1}^D, \hat{H}_{i-1}^P.W), \tag{6}$$

Here, f_{TCN} is a TCN function that processes the input estimates $\hat{H}_{i-1}^D$ and $\hat{H}_{i-1}^P$ to produce the refined channel estimate $\hat{H}_i^{\text{TCN}}[k]$. The function f_{TCN} is parameterised by the weight matrix W, which includes the learnable filters and biases of the TCN's convolutional layers. These weights are optimised during training to minimise the estimation error.

In Fig. 2, the LS block initialises the channel estimation using the preamble, producing $\hat{H}_{\text{LS}}[k]$. The DPA block iteratively refines the estimation by utilising both the received signal $y_i[k]$ and the prior channel estimate $\hat{H}_i^{\text{DPA}}[k]$, resulting in $\hat{H}_i^{\text{DPA}}[k]$. The final TCN block further refines the estimate

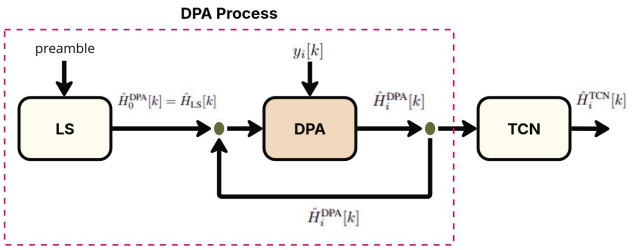


Fig. 2: DPA-TCN estimator process

to obtain the final channel estimate $\hat{H}_i^{\text{TCN}}[k]$. Arrows indicate the flow of data and the iterative feedback loop within the DPA block.

1) Hyperparameter Tuning: We use Bayesian optimisation (BO) to find the best hyperparameter combination for model performance. BO is an iterative algorithm that uses a probabilistic model, commonly a Gaussian process, to approximate an unknown objective function. For the TCN estimator, we optimise the learning rate, number of layers, the size of the kernel, the dropout, and the StepLR parameters (step size and gamma) using Weights & Biases (WandB)[1] and Optuna [15], a BO-backed hyperparameter tuning tool. The results of the experiment are recorded and tracked by WandB.

We conduct 50 trials, each running for 200 epochs, to comprehensively explore the hyperparameter space and observe the effects of hyperparameter changes. Convergence is determined using the validation loss. The hyperparameter search space and the best combination are shown in Table I.

TABLE I: Best TCN Hyperparameters

Hyperparameter	Search Space	Best Value
Learning Rate	1×10^{-5} to 1×10^{-2}	0.0006
Number of Layers	1 to 5	4
Kernel Size	2 to 5	2
Dropout	10^{-5} to 0.5	0.17
StepLR Step Size	10 to 50	21
StepLR Gamma	0.5 to 1	0.9
Epochs	0 to 200	156

IV. Analysis and Results

In this section, the proposed DPA-TCN estimator is compared to other estimators namely the TCN-DPA, LSTM-DPA-TA, LS, DPA estimators as well as the ideal case with perfect channel knowledge. Furthermore, we present a comparative analysis of the computational complexity of these estimators.

Firstly, we perform a hyperparameter search for the LSTM model in the LSTM-DPA-TA estimator on the dataset generated at 40 dB using Optuna and WandB, following the same experimental setup discussed in Section III-C1. The following

[1]https://wandb.ai/

best parameters are obtained for the LSTM model: a learning rate of 0.003, LSTM size = 128, StepLR step size = 12 and StepLR step gamma = 0.9. Then, this LSTM architecture is combined with DPA and TA as specified in [8] for comparison with the proposed DPA-TCN estimator.

A. BER

Fig. 3a shows the BER performance of various estimators. The LS estimator shows very poor performance across all tested SNRs. A standalone DPA estimator achieves better BER performance than LS but falls short when compared to NN-based estimators, especially at SNRs greater than 20 dB. The TCN-DPA trained at 40 dB achieves average performance at low SNRs up to 15 dB, and an exponential improvement from 20 dB to 40 dB. This improved performance in high SNR environments but not in low SNRs indicates the TCN's ability to excel in the conditions in which it is trained, but less so elsewhere.

Our proposed DPA-TCN estimator, trained using a mixed dataset, demonstrates good performance over a wide SNR range. Furthermore, our proposed DPA-TCN estimator achieves an overall BER performance comparable to that of the LSTM-DPA-TA. At SNR levels of 0 to 10 dB, the performance of the two estimators is similar. From 15 dB to 33 dB, the BER using the LSTM-DPA-TA estimator is lower than that when using the TCN estimator. Beyond 33 dB, the TCN estimator performs slightly better than the LSTM-DPA-TA.

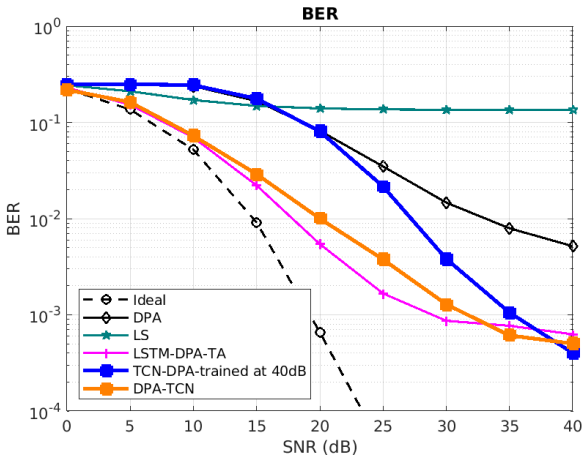
B. NMSE

The proposed DPA-TCN estimator aims to improve channel estimation by enhancing DPA processing. As illustrated in Fig. 3b, the DPA-TCN estimator consistently achieves lower NMSE values compared to the standalone DPA estimator and other methods across all SNR ranges investigated. In contrast, the TCN-DPA trained at 40 dB exhibits performance similar to the initial DPA estimator at low SNRs (0 to 15 dB) but shows exponential improvement starting from 20 dB. The LSTM-DPA-TA estimator outperforms all the models presented with significantly lower NMSE values which are only approached by the other presented models at high SNR levels. Our DPA-TCN outperforms the standalone DPA estimator over most of the SNR range, approaching LSTM-DPA-TA and TCN-DPA performance at very high SNR levels.

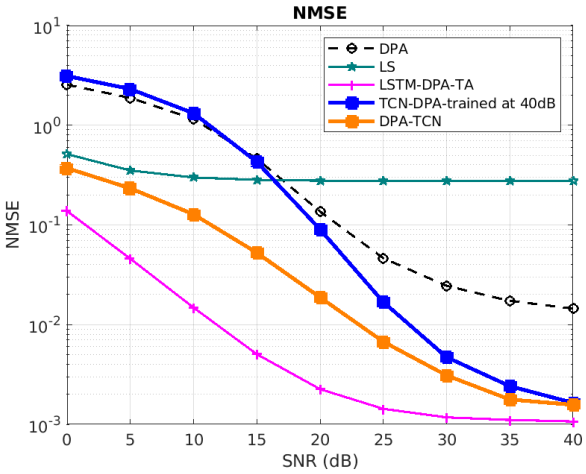
C. Complexity analysis

This section discusses the computational complexity of the estimators presented in this work using real-valued operations as a measure of complexity. Real-valued operations are generally used to analyse the complexity of channel estimation methods [2].

LS estimation requires the addition and division of the received preambles as shown in (2), which is $2X_A$ summations and $2X_A$ divisions where X_A represents the number of active subcarriers. In this case, $X_A = 52$ subcarriers. From equations (3) to (5) DPA requires 2 complex value divisions that require



(a) BER performance of various estimators.



(b) NMSE performance of various estimators.

Fig. 3: BER and NMSE simulation results on VTV-SDWW vehicular channel model.

6 real-valued multiplications, 2 real-valued divisions, 2 real-valued summations, and one real-valued subtraction totaling $26X_A$ operations.

In [8], the authors provide a detailed analysis of the computational complexity of the LSTM-DPA-TA estimator. This analysis includes the real-value operations performed by each LSTM gate and the TA post-processing method. The computational complexity for the LSTM-DPA-TA component is given in [8] as:

$$L_{\text{LSTM}} = 4(L^2) + L(4X_{\text{in}} + 3) + 18X_D + X_{\text{in}} + 13L + 5X_{\text{in}} + 8X_D - 8, \quad (7)$$

where the input size $X_{\text{in}} = 104$, is obtained by stacking the real and imaginary parts of the complex data along the subcarrier dimension, X_D is the input data subcarrier which is also obtained by stacking the real and imaginary parts of the 48 complex data subcarriers, and L represents the unit size of the LSTM, with $L = 128$.

The complexity of a TCN residual block depends primarily on its 1D convolutional layers. Our proposed DPA-TCN has a single residual block and performs a one-dimensional convolution along the subcarrier domain. The residual block has 4 layers and uses a kernel size $K = 2$. Our TCN has the same input (X_I) and output (X_O) sizes, which are equal to the number of active subcarriers $X_A = 52$. The input data contains 100 features, which are also fed into the TCN as X_F.

Each convolutional layer requires $(X_I \times X_F \times K)$ real-valued multiplications and X_O real-valued summations. The total complexity for 4 layers is calculated as follows:

- Total multiplications:

$$N_{\text{mul}} = 4 \times (X_A \times K \times X_F) = 8(X_A X_F). \quad (8)$$

- Total summations:

$$N_{\text{sum}} = 4X_A. \quad (9)$$

Additionally, considering the residual connection, when the input and output dimensions are different, the multiplications for the residual connection equal $X_A \times 1$, and the summations for the residual connection are X_A. Thus, the total real-valued multiplications and summations of our TCN estimator are:

$$N_{\text{mul,sum}} = 8X_A X_F + 6X_A. \quad (10)$$

Equations 8 to 10 quantify the computational complexity of the TCN estimator in terms of real-valued multiplications and summations. The total complexity is dominated by the convolutional operations, which scale linearly with the number of active subcarriers X_A and the number of features X_F.

TABLE II: Total computational complexity

Estimator	Total complexity
LS	208
DPA	1,560
LSTM-DPA-TA	123,944
DPA-TCN (proposed)	43,472

Table II shows the total complexity of the estimators compared in this work. The proposed DPA-TCN estimator has a reduced complexity compared to the LSTM-DPA-TA estimator by approximately 65%. In other words, the TCN estimator is around 3 times less complex than the LSTM-DPA-TA. Its reduced complexity makes it ideal for vehicular communications that require faster processing.

V. Conclusion

In this paper, we proposed a novel DPA-TCN estimator for channel estimation in vehicular communications, specifically designed for the VTV-SDWW channel model. By integrating the advantages of DPA and a TCN, the proposed estimator effectively addresses the challenges of channel estimation in high-mobility environments. Simulation results indicate that our proposed DPA-TCN estimator achieves BER performance comparable to the current state-of-the-art estimator (LSTM-DPA-TA) but with significantly reduced computational complexity in comparison. Future work will investigate other vehicular channel models.

Acknowledgment

The authors thank Telkom CoE at the NWU who supported this research.

References

[1] D. Jiang and L. Delgrossi, "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments," in *2008 VTC Spring - IEEE Vehicular Technology Conference*, 2008, pp. 2036–2040.

[2] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Deep Learning Based Channel Estimation Schemes for IEEE 802.11p Standard," *IEEE Access*, vol. 8, pp. 113 751–113 765, 2020.

[3] J. Pan, H. Shan, R. Li, Y. Wu, W. Wu, and T. Q. Quek, "Channel Estimation Based on Deep Learning in Vehicle-to-Everything Environments," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1891–1895, 2021.

[4] P. Walk, H. Becker, and P. Jung, "OFDM Channel Estimation via Phase Retrieval," in *2015 49th Asilomar Conference on Signals, Systems and Computers*, 2015, pp. 1161–1168.

[5] S. Han, Y. Oh, and C. Song, "A Deep Learning Based Channel Estimation Scheme for IEEE 802.11p Systems," in *2019-2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[6] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, "Joint TRFI and Deep Learning for Vehicular Channel Estimation," in *2020 IEEE Global Communications Conference (GLOBECOM) Workshops*. IEEE, 2020, pp. 1–6.

[7] J. D. Jovane and C.-H. Lee, "Channel Estimation using Temporal Convolutional Networks for V2X Communications," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 565–570.

[8] A. K. Gizzini, M. Chafii, S. Ehsanfar, and R. M. Shubair, "Temporal Averaging LSTM-based Channel Estimation Scheme for IEEE 802.11p Standard," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 01–07.

[9] S. A. Ngorima, A. S. J. Helberg, and M. H. Davel, "Neural Network-Based Vehicular Channel Estimation Performance: Effect of Noise in the Training Set," in *Artificial Intelligence Research*, A. Gerber, J. Maritz, and A. W. Pillay, Eds. Cham: Springer Nature Switzerland, 2025, pp. 192–206.

[10] S. A. Ngorima, A. Helberg, and M. H. Davel, "Sequence Based Deep Neural Networks for Channel Estimation in Vehicular Communication Systems," in *Artificial Intelligence Research*, A. Pillay, E. Jembere, and A. J. Gerber, Eds. Cham: Springer Nature Switzerland, 2023, pp. 172–186.

[11] G. Acosta-Marum and M. A. Ingram, "Six Time- and Frequency-Selective Empirical Channel Models for Vehicular Wireless LANs," in *2007 IEEE 66th Vehicular Technology Conference*, 2007, pp. 2134–2138.

[12] S. A. Ngorima, A. Helberg, and M. H. Davel, "A Data Pilot-Aided Temporal Convolutional Network for Channel Estimation in IEEE 802.11p Vehicle-to-Vehicle Communications," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 2024.

[13] Y. LeCun, Y. Bengio *et al.*, "Convolutional Networks for Images, Speech, and Time-Series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[14] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[15] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data mining*, 2019, pp. 2623–2631.