



Detecting problematic samples in deep learning

L Freese

 **orcid.org/ 0009-0001-6973-7831**

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor: Dr MW Theunissen

Co-supervisor: Dr R Rabe

Graduation: May 2026

Declaration

I, Leon Freese, hereby declare that the dissertation entitled “Detecting problematic samples in deep learning” is my own original work and has not already been submitted to any other university or institution for examination.

L. Freese

Student number:

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

Acknowledgements

First and foremost, all praise be to God, who gives me strength. Without Him, none of this would have been possible.

I would like to express my sincere gratitude to my supervisors, Tian and Randle, whose advice and insight have been invaluable in shaping this work. Especially Tian, for keeping me focused when there are endless possibilities to explore in our field of research, and for sharing a genuine interest in this work. Any success that comes from this study, we share equally.

A special thanks goes to Gerbrand, a fellow MUST PhD student, whose willingness to help have been a constant source of support. I truly enjoyed our whiteboard sessions.

Pursuing a postgraduate degree is a privilege, and it would not have been possible without the financial and moral support of my family. I am grateful for your continuous support throughout this journey.

A heartfelt thanks to Cara—your relentless encouragement, especially when things went sideways, has been truly special. Thank you for your love and support.

Lastly, I am grateful to the MUST Deep Learning Group for providing the equipment and environment I needed to complete my work. I enjoyed my two years of postgraduate studies, and value the friendships I made along the way.

Abstract

This study investigates problematic samples in deep learning, that is samples that hinder model generalization. Although deep neural networks are typically trained on large datasets under the assumption that more data yields better performance, this work highlights that not all samples contribute positively and equally. To examine this issue, the study proposes a data pruning evaluation framework that systematically evaluates several methods (that we call problematic sample *detectors*) to reveal the presence, nature and impact of problematic samples. Specifically, we evaluate two learning-centric detectors (Sample forgetting and the Error- L_2 -norm) and two data-centric detectors (Confident Learning and k -nearest-neighbours distance), from the fields of data valuation, label noise detection and out-of-distribution detection in deep learning. Using controlled evaluations on synthetic data and experiments involving explicitly added noise on MNIST and CIFAR-10, trained on Multilayer Perceptrons and Convolutional Neural Networks, we analyse the ability of detectors to distinguish various forms of problematic samples.

The results reveal that both input-space and label-space distributional shifts can induce problematic samples and that learning-centric and data-centric methods can be used as problematic sample detectors, although each have their limitations. This prompted us to develop a new problematic sample detector that we call the Variance-Weighted Fitness Gap detector to address the shortcomings of existing learning-centric methods. Our findings also show that problematic-sample detection can improve model generalization even in the absence of explicitly added noise on a real-world dataset. Furthermore, we highlight neglected nuances regarding the detection of problematic samples. These include the meaning of sample contribution in the presence or absence of label noise and the confounding effects of dependencies between samples.

Keywords: *Deep learning, Generalization, Problematic samples, Label noise, Out-of-distribution samples*

Contents

List of Figures	ix
List of Tables	xv
List of Abbreviations	xvi
Notation	xvii
1 Introduction	1
1.1 Understanding generalization in deep learning	1
1.2 Problem statement	4
1.3 Research questions	4
1.4 Objectives	5
1.5 Project scope	5
1.6 Dissertation overview	6
1.7 Publications	7
2 Background	8
2.1 Overview	8
2.2 Data valuation	9
2.3 Label noise detection	12

2.4	Out-of-distribution detection	15
2.5	Influence functions	17
2.6	Discussion	19
3	Evaluation framework for problematic sample detectors	22
3.1	Overview	22
3.2	Problem formulation	23
3.3	Problematic sample detector	24
3.4	Data pruning evaluation framework	26
3.5	Out-of-distribution samples	31
3.6	Discussion	34
4	Controlled evaluation using synthetic data	36
4.1	Overview	36
4.2	Methods	37
4.2.1	Data generation	37
4.2.2	Noise types	38
4.2.3	Experimental setup	42
4.3	Binary classification	44
4.3.1	Label noise	45
4.3.2	Input noise	50
4.4	Multiclass classification	51
4.4.1	Label noise	52
4.4.2	Input noise	55
4.5	Discussion	57
5	On learning-centric detectors	59

5.1	Overview	59
5.2	Data pruning methods	60
5.3	Detectors	61
5.3.1	Sample forgetting	62
5.3.2	Error L2-norm	62
5.4	Experimental setup	64
5.5	Results	65
5.5.1	Redundant samples	65
5.5.2	Problematic samples	68
5.5.3	A deeper look	71
5.6	Discussion	78
6	On data-centric detectors	81
6.1	Overview	81
6.2	A priori OOD detection	82
6.3	Detectors	84
6.3.1	k-nearest-neighbors distance	84
6.3.2	Confident learning	85
6.4	Experimental setup	86
6.5	Results	89
6.5.1	The effect of label noise	89
6.5.2	The effect of input noise	94
6.5.3	Dataset size and model capacity	97
6.6	Discussion	99
7	Possible avenues for improvements	102

7.1	Overview	102
7.2	Variance-Weighted Fitness Gap (VWFG)	103
7.2.1	Intuition	103
7.2.2	Method	105
7.2.3	Experimental setup	107
7.2.4	Results	108
7.3	Real-world analysis	111
7.3.1	Dataset	112
7.3.2	Experimental setup	113
7.3.3	Results	114
7.4	Sample interdependence	116
7.4.1	Intuition	117
7.4.2	Methods	118
7.4.3	Experimental setup	120
7.4.4	Results	122
7.5	Discussion	128
8	Conclusion	131
8.1	Overview	131
8.2	Key findings	131
8.3	Contributions	133
8.3.1	Addressing research questions	133
8.3.2	Implications	134
8.4	Limitations	135
8.5	Future work	136
8.6	Final remark	138

References	139
A Supplemental content	151
A.1 Instance-dependent label noise	151
A.2 k-NN distance to class centroid	152
A.3 Explicit label noise detection of learning-centric detectors	154
A.4 Parameter influence PCA	154

List of Figures

- 3.1 A simple example of possible dependence between samples and the degree to which they are problematic. In all three scenarios we have two dimensional data that can be categorized into one of two classes based on a ground truth boundary across the first feature dimension. (left) The ground truth boundary is accurately approximated by a simple logistic regression solver. (middle) A single problematic sample near the boundary shifts the learned boundary away from the ground truth. (right) The presence of other samples near the boundary results in the performance degradation being significantly reduced. 24
- 3.2 An illustration of ψ -curves for Algorithm 1 when the performance metric $\mathcal{P}$ is accuracy. The **red** dot represents the model performance $\psi_1 = \mathcal{P}(D_{\text{test}}, \hat{f})$ when trained on the full train set. The **green** dot illustrates the maximum model performance $\psi_{\text{max}} = \mathcal{P}(D_{\text{test}}, \hat{f})$ after $3r$ samples of the train set have been removed. 29
- 3.3 Examples of the three categories of distributional shifts for a three-class classification problem with $y \in Y = \{\text{'cat'}, \text{'dog'}, \text{'snake'}\}$. The original sample is shown on the left with two examples of the distributional shift on the right. Note that for covariate- and semantic shifts, the input $\mathbf{x}$ is changed while the label y stays constant. For label corruption, the input $\mathbf{x}$ stays constant while the label y is changed within the label space Y . . . 33
- 4.1 An illustrative visualization of different noise types for a 2D three-class classification problem. Each subplot shows clean samples (translucent) and corrupted samples (opaque with black edges). The noise rate for all plots (except the clean case) is $\eta = 0.2$. The legend (bottom right) applies to all subplots. 41

4.2	Evaluation framework for binary classification with label noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 7$ (blue) and $d = 3$ (orange). The standard deviations for class 1 and 2 are fixed at $\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$, respectively. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	45
4.3	Evaluation framework for binary classification with label noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 3$. The standard deviations are varied: ($\sigma^{(1)} = 3$ and $\sigma^{(2)} = 4$) blue and ($\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$) orange. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	47
4.4	Evaluation framework for binary classification with label noise introduced into the train set, while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 3$ and standard deviations $\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$. The noise rate η is varied from 0.1 to 0.4 in increments of 0.1. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	48
4.5	Evaluation framework for binary classification with three types of input noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 3$ and standard deviations $\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	51
4.6	Evaluation framework for $c = 10$ multiclass classification with three types of label noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 7$ and standard deviations $\sigma^{(j)} \sim \mathcal{U}(1, 5) \in \mathbb{R}$. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	52
4.7	Evaluation framework for $c = 10$ multiclass classification with symmetric label noise introduced into the train set, while the test set remains clean. For (a), we keep the class-related standard deviation identical, while varying the distance d as shown in the legend. For (b), we keep the distance constant and vary the ranges that that standard deviations $\sigma^{(j)}$ are sampled from, as shown in the legend. Lastly, for (c), the distance and standard deviations are kept constant while the noise rate η is varied as shown in the legend. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	54

4.8	Evaluation framework for $c = 10$ multiclass classification with three types of input noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 7$ and standard deviations $\sigma^{(j)} \sim \mathcal{U}(1, 5) \in \mathbb{R}$. Solid lines correspond to <i>oracle</i> detectors, while dashed lines correspond to <i>random</i> detectors.	55
4.9	The columns of this plot correspond to the three types of input noise: Gaussian input noise (left), uniform input noise (middle) and edge input noise (right). The two rows correspond to different visualization techniques: PCA (top) and t-SNE (bottom). The legend in the first plot applies to all plots.	56
5.1	Model performance as a function of pruning an increasing portion of the train set for synthetic (left-top), MNIST (right-top), and CIFAR-10 (bottom) datasets. Samples are pruned in ascending order, with rankings provided by the two detectors in Section 5.3, or by a <i>random</i> baseline. The detectors with a * in the title have their ranking reversed. The legend in the first plot applies to all three plots.	66
5.2	Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ symmetric label noise for synthetic (left-top), MNIST (right-top), and CIFAR-10 (bottom) datasets. Samples are pruned in ascending order, with rankings provided by the two detectors in Section 5.3. The detectors with a * in the title have their ranking reversed. The <i>random</i> and <i>oracle</i> baselines are described in Section 4.2.3. The legend in the first plot applies to all three plots.	69
5.3	Sample ranking for synthetic (top-left), MNIST (top-right), and CIFAR-10 (bottom) before and after label noise is applied. The x -axis shows the samples' ranking before label corruption and the y -axis shows the samples' ranking after label corruption. Red samples have been label corrupted, while black samples were not corrupted. The blue dashed line is a linear best-fit line. The first column, in each plot, considers <i>EL2N</i> scores. The second column considers <i>sample forgetting</i> scores.	70
5.4	Top 20 MNIST train set samples with the highest scores for the two detectors (<i>sample forgetting</i> (a) and <i>EL2N</i> (b)) with 30% symmetric label noise. The images with information shown in red have corrupted labels. . .	71
5.5	Number of MNIST samples from each pre-corruption class having been pruned at each point in order of the ranking obtained by <i>sample forgetting</i> (left) and <i>EL2N</i> (right), in descending order.	72

5.6	Mean <i>sample forgetting</i> score (top) and <i>EL2N</i> score (bottom) for MNIST samples according to their pre- and post-corruption target classes. The smaller plot indicates the standard deviations corresponding to each mean value.	73
5.7	Top 20 CIFAR-10 train set samples with the highest scores for the two detectors (<i>sample forgetting</i> (a) and <i>EL2N</i> (b)) with 30% symmetric label noise. The images with information shown in red have corrupted labels. . .	74
5.8	Number of CIFAR-10 samples from each pre-corruption class having been pruned at each point in order of the ranking obtained by <i>sample forgetting</i> (left) and <i>EL2N</i> (right), in descending order.	75
5.9	Mean <i>sample forgetting</i> score (top) and <i>EL2N</i> score (bottom) for CIFAR-10 samples according to their pre- and post-corruption target classes. The smaller plot indicates the standard deviations corresponding to each mean value.	76
5.10	Sensitivity analysis on the ranking order of samples when the sample forgetting (a) and EL2N (b) scores are calculated across a different number of seeds. The <i>y</i> -axis shows the standard error of the samples' rankings, averaged across all train set samples.	77
6.1	Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ label noise for synthetic (top row), MNIST (middle row), and CIFAR-10 (bottom row) datasets. The results for the three types of label noise are shown in the columns: symmetric (left column), asymmetric (middle column) and instance-dependent label noise (right column). Samples are pruned in descending order, with scores provided by the two detectors in Section 6.3. The <i>random</i> and <i>oracle</i> baselines are described in Section 4.2.3. The legend in the first plot applies to all nine plots.	90
6.2	Confident joint (C matrix) of the train set with $\eta = 0.3$ label noise for synthetic (top row), MNIST (middle row), and CIFAR-10 (bottom row) datasets. The results for the three types of label noise are shown in the columns: symmetric (left column), asymmetric (middle column) and instance-dependent label noise (right column).	92
6.3	Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ input noise for synthetic (a), MNIST nano (b), and CIFAR-10 (c) datasets. Samples are pruned in descending order, with rankings provided by the two detectors in Section 6.3. The <i>random</i> and <i>oracle</i> baselines are described in Section 4.2.3. The legend in the first plot applies to all three plots.	95

6.4	Heatmap showing the difference in evaluation accuracy between models trained on clean ID data and models trained with 30% covariate-shifted (randomly rotated) OOD samples for the MNIST dataset. Rows correspond to different train set sizes, and columns correspond to the hidden layer size of a 2-layer MLP. Each value represents the average result over three random seeds.	98
6.5	An illustration of four types of covariate shifts. The original (ID) sample on the left.	99
6.6	Generalization performance of models trained on MNIST subsets with 30% OOD samples included in each train set. Results are averaged over three random seeds.	100
7.1	Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ label noise for synthetic (top row) and CIFAR-10 (bottom row) datasets. The results for the two types of label noise are shown in the left column (symmetric) and right column (asymmetric) label noise. Samples are pruned in descending order, with scores provided by the three detectors. The <i>random</i> and <i>oracle</i> baselines are described in Section 4.2.3. The legend in the first plot applies to all four plots.	109
7.2	Fitness matrices $\mathbf{F}$ of the train sets with $\eta = 0.3$ label noise for synthetic (top row) and CIFAR-10 (bottom row) datasets. The results for the two types of label noise are shown in the left column (symmetric) and right column (asymmetric) label noise. The rows of each matrix are sorted in ascending order from top to bottom according to the <i>VWFG</i> score, with each row corresponding to a single training sample. The white dotted line shows the 30% pruning threshold, when samples are pruned from bottom to top (descending order of score).	111
7.3	Model performance as a function of pruning an increasing portion of the CheXpert train set, according to the scores of all the problematic sample detectors. The <i>random</i> baseline is also provided. The top plot provides mean values (with standard deviations) across 5 seeds. The bottom plots show the corresponding maximum (left) and minimum (right) values at each point in the curves.	115
7.4	Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ symmetric label noise added to the synthetic (left) and CIFAR-10 (right) datasets. For <i>EL2N*</i> , samples are pruned in descending order of scores provided by the <i>EL2N</i> method, and for <i>EL2N!</i> samples are pruned in descending order with re-scoring at each iteration. The <i>random</i> and <i>oracle</i> baselines are described in Section 4.2.3. The legend in the first plot applies to both plots.	122

7.5	Interdependent training samples identified through influence-based cosine similarity. The leftmost column displays six reference samples (top three clean, bottom three noisy). For each reference sample, the next three columns show the training samples whose parameter influence vectors have the highest cosine similarity with the reference, while the final three columns show the samples with the lowest cosine similarity. The images with information shown in red have corrupted labels.	124
7.6	Collective parameter influence (left) and evaluation accuracy (right) as a function of pruning an increasing portion of the train set with $\eta = 0.3$ symmetric label noise added to the and CIFAR-10 dataset.	127
A.1	k -NN distance score versus the distance to the closest class centroid (excluding the labelled centroid) of the train set with $\eta = 0.3$ label noise for synthetic (top row), MNIST (middle row), and CIFAR-10 (bottom row) datasets. The results for the three types of label noise are shown in the columns: symmetric (left column), asymmetric (middle column) and instance-dependent label noise (right column). Clean samples are shown in blue while label corrupted samples are shown in red.	153
A.2	The cumulative corruption percentage of the learning-centric detectors as a function of pruning an increasing portion of the train set with $\eta = 0.3$ label noise for synthetic (top row) and CIFAR-10 (bottom row) datasets. The results for the two types of label noise are shown in the columns: symmetric (left column) and asymmetric (right column).	155
A.3	All CIFAR-10 samples labelled <i>bird</i> embedded into an estimate of the first and second principle component directions. The smaller heatmap to the top-right is a visualization of a simple Gaussian Kernel Density function fitted to a subset of the scatter-plot to the left. This subset corresponds to the samples that were not label corrupted.	156
A.4	Same as Figure A.3 but for the remaining classes in CIFAR-10.	157

List of Tables

0.1	Distinguishing between types of values.	xvii
5.1	Dataset sizes and model configurations for the three datasets.	64

List of Abbreviations

AI Artificial Intelligence

ML Machine Learning

SLT Statistical Learning Theory

DNN Deep Neural Network

MLP Multilayer Perceptron

CNN Convolutional Neural Network

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

OOD Out-of-distribution

ID In-distribution

EL2N Error L_2 -norm

k -NN k -nearest-neighbour

AUM Area Under the Margin

CL Confident Learning

i.i.d. independent and identically distributed

MSP Maximum Softmax Probability

HVP Hessian-vector product

MDL Minimum Description Length

VWFG Variance-Weighted Fitness Gap

Notation

Unless stated otherwise, we consistently make use of the notation described here.

Table 0.1: Distinguishing between types of values.

Description	Notation	Example
a scalar	a lower-case letter	x
a vector	a bold lower-case letter	$\mathbf{x}$
a matrix	a bold upper-case letter	$\mathbf{X}$
a set	an upper-case letter	X
a number system	an upper-case blackboard letter	$\mathbb{X}$

We use subscripts to denote the index values of vectors or matrices:

- A vector $\mathbf{w}$ can be broken into its components i.e., $\mathbf{w} = (w_1, w_2, \dots, w_m) \in \mathbb{R}^m$
- $w_{i,j} = [\mathbf{W}]_{i,j}$ is a scalar value corresponding to the element in the i -th row and j -th column of matrix $\mathbf{W}$
- $\mathbf{w}^\top = [\mathbf{W}]_{i,*}$ is a vector corresponding to the i -th row of matrix $\mathbf{W}$
- $\mathbf{w} = [\mathbf{W}]_{*,j}$ is a vector corresponding to the j -th column of matrix $\mathbf{W}$

We use superscripts inside round brackets to denote class-specific information:

- $\mathbf{W}^{(j)}$ is a matrix corresponding to class j
- If $D = \{(\mathbf{x}, y)\}_{i=1}^n$ is a dataset, with input feature vectors $\mathbf{x}$ and scalar label y value pairs, then $D^{(j)}$ is a subset of D with class labels j

Chapter 1

Introduction

We introduce the topic of the study along with an overview of our methods.

1.1 Understanding generalization in deep learning

Over the past decade, *deep learning* has emerged as one of the most transformative paradigms in modern science and engineering. By leveraging vast datasets and large-scale optimization, Deep Neural Networks (DNNs) have revolutionized fields including computer vision [1], natural language processing [2], molecular biology [3], and time series forecasting [4]. These successes have not only expanded the capabilities of Artificial Intelligence (AI) but have also reshaped the way we think about representation, abstraction, and learning itself.

The success of deep learning can largely be attributed to the ability of DNNs to learn intricate, high-dimensional mappings between inputs and outputs, that *generalize* to unseen data. Despite this empirical success, the exact mechanisms that underpin generalization in DNNs remain elusive. Nevertheless, the research community has witnessed significant

progress towards uncovering why DNNs generalize so well.

At the heart of this discussion lies the concept of an *implicit bias*. Despite their capacity to represent an enormous number of possible functions, common DNN training procedures do not favour all functions equally [5], [6]. Instead, when trained with stochastic optimization algorithms such as Stochastic Gradient Descent (SGD), these systems exhibit a preference (an implicit bias) for certain types of solutions for reasons that are not yet fully understood. Intriguingly, as model size increases, this bias tends to favour solutions that generalize well, even though the number of possible functions grows [7], [8].

This counterintuitive property manifests clearly in the *double descent* phenomenon [7]. Classical Statistical Learning Theory (SLT) predicts that increasing model complexity beyond a certain point should worsen generalization due to overfitting—a direct consequence of the *bias–variance tradeoff* [9]. However, in the overparameterized regime, DNNs often defy this rule: once the model is large enough to perfectly fit (i.e. interpolate) the training data, further increasing its size can actually improve generalization. In this overparameterized regime, since the training loss remains at zero or near zero, the improvement cannot be explained by fitting more of the training data: it must emerge from the structure of the solution itself.

Empirical studies have revealed that these interpolating solutions often correspond to *flatter minima* in the loss landscape [10], [11]. Due to the high-dimensional geometry of DNN’s, flatness cannot be measured directly but must be estimated through proxy metrics [11]–[14]. Although two models can achieve identical training loss, those converging to flatter minima tend to generalize better, a pattern that is thought to be linked to principles of simplicity such as Occam’s razor. From a model simplicity perspective (related to a concept called *Minimum Description Length (MDL)*), flatter minima correspond to simpler models because they occupy a larger volume in parameter space, meaning fewer “bits” or less information is required to precisely describe a model within that region [10].

This perspective provides a potential explanation for the implicit bias of optimization algorithms as mentioned earlier. When SGD traverse the parameter space with finite

learning rates, it is exponentially more likely to land in large, flat regions, simply because such regions dominate the volume in high dimensions. In this case, the so-called “curse of dimensionality” paradoxically aids generalization: the enormous parameter space makes simpler, flatter solutions more accessible to SGD [15].

However, flatness alone cannot fully explain generalization. It has been shown that simply scaling the weights of a model by a constant (i.e., scaling the learned function) changes the flatness of the loss landscape but not the generalization performance [16]¹. Nonetheless, insights from loss landscape geometry and optimization have collectively deepened our understanding of why certain solutions generalize better than others.

Given that generalization is clearly governed by a combination of the final solution’s properties (e.g., the flatness of the loss landscape) and the characteristics of the optimization algorithm (i.e., implicit bias), a key question emerges: What are some of the main contributors behind both of these aspects? We posit that one critical factor governing both the final solution’s geometry and the optimization path is the *training data*. The present study, therefore, aims to probe neural network generalization from a training data perspective. We argue that a deeper understanding of what drives generalization in deep learning might be achieved by examining which samples *contribute* constructively to learning underlying patterns and which may *hinder* this process by encouraging memorization or reinforcing spurious correlations.

Specifically, we introduce the notion of a *problematic sample*:

Definition: *A problematic sample is any sample in the train set that negatively affects the expected performance of the model on the test set.*

Intuitively, the effect of a single such sample is often negligible in massive datasets. This could be due to the presence of many alternative samples indicating similar solutions to the reference sample (making the absence of the reference sample negligible) or many alternative samples indicating completely different solutions (making the presence of the

¹This invariance holds for classification tasks, where the prediction depends only on the relative order of outputs (e.g., $\arg \max$).

reference sample negligible inherently). However, groups of such samples (problematic sets) may collectively degrade generalization by distorting the learned function toward noise or misleading features.

Throughout this study, we will explore algorithms for identifying and analysing these problematic samples. By developing and evaluating *problematic sample detectors*, we aim to shed new light on the role that training data plays in shaping and understanding generalization.

1.2 Problem statement

Deep neural networks often memorize certain training samples rather than learning generalizable patterns, undermining model performance on unseen data. Despite growing recognition of this issue, existing methods in the literature have not been systematically evaluated in the general context of *problematic samples*—as defined in this study.

1.3 Research questions

To investigate methods for the reliable detection of problematic samples, this study addresses the following research questions:

1. How can we systematically evaluate and compare different methods for detecting problematic samples within a dataset?
2. To what extent are existing methods effective for detecting problematic samples as defined in this study?
3. Can current methods be improved to more accurately identify problematic samples?

1.4 Objectives

Based on the research questions, the study pursues the following objectives:

1. **Framework development** To formalize an evaluation framework that enables 1) comparison of problematic sample detectors, 2) identification of problematic samples, and 3) estimation of the degree to which samples are problematic.
2. **Literature review** To identify relevant existing methods from different fields in literature, that can be adapted for problematic sample detection.
3. **Method evaluation** To empirically evaluate identified methods using the proposed framework and assess their effectiveness for detecting problematic samples as defined in this study.
4. **Method enhancement** To explore and propose improvements to existing methods based on insights gained through evaluation and analysis.

1.5 Project scope

In line with the problem statement and objectives, the scope of this research is defined as follows:

- **Tasks** This study focuses exclusively on classification tasks. This constraint, consistent with related literature, enables focused analysis and direct comparison of methods without the confounding factors introduced by other task types.
- **Datasets** The analysis is limited to synthetically generated datasets and well-established image classification benchmarks, namely MNIST [17] and CIFAR-10 [18]. A smaller subset of experiments will also include CIFAR-100 [18] and CheXpert [19]. Restricting the study to synthetic and benchmark datasets ensures both controlled experimental conditions and comparability with existing research.

-
- **Architectures** The DNN architectures considered are primarily Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs). This limitation ensures consistency with related studies and facilitates fair evaluation across existing methods.
 - **Problematic sample types** Adversarial samples are excluded from the scope of problematic samples in this study. Instead, the focus is on intrinsic occurring problematic instances or those affected by explicit label- or input noise, rather than deliberately manipulated (perturbed) inputs. This constraint allows for a clearer analysis of expected issues in data and model behaviour.

1.6 Dissertation overview

The dissertation is structured to systematically develop an evaluation framework for problematic sample detectors and then use this framework to rigorously test and improve existing methods on both controlled synthetic data and image classification datasets. Specifically, the chapters are structured as follows:

- Chapter 2 presents a *literature review* that outlines potential methods that can serve as problematic sample detectors, addressing **objective 2**.
- Chapter 3 introduces an *evaluation framework* designed to meet the three requirements outlined in **objective 1**.
- In Chapter 4, this framework is tested in a *controlled setting* using synthetically generated data.
- Chapter 5 presents two *learning-centric* methods identified in the literature review, and evaluates them with our evaluation framework as problematic sample detectors using image classification datasets, thereby addressing **objective 3**.
- Chapter 6 presents two *data-centric* methods identified in the literature review, and evaluates them with our evaluation framework as problematic sample detectors

using image classification datasets, thereby addressing **objective 3**.

- The purpose of Chapter 7 is three-fold: 1) develops a *new* problematic sample detector based on the insights from previous chapters, thereby addressing **objective 4**, 2) it conducts a comparison of the problematic sample detectors in Chapters 5 and 6 on a *real-world* image classification dataset and 3) investigates the effect of sample *interdependencies* in the training data.
- Finally, Chapter 8 *concludes* the study with a summary of key findings, avenues for future work and closing remarks.

1.7 Publications

Some of this dissertation’s content has been accepted for presentation at SACAIR 2025, and selected for inclusion in Communications in Computer and Information Science (LNCS sub-series CCIS). Specifically, parts of the content of Chapter 5 are included in “Data Pruning: Redundant, Problematic, and Interdependent Samples” [20].

Chapter 2

Background

We investigate related work in the fields of data valuation, label noise detection, out-of-distribution detection and influence functions in deep learning.

2.1 Overview

This chapter provides a review of the literature relevant to this study, focusing on four interrelated areas that address data quality and sample reliability in machine learning: data valuation, label noise detection, Out-of-distribution (OOD) detection and influence functions. These fields, while originating from different research communities, share a common underlying goal: to understand and improve the relationship between training data and model performance by identifying samples that may hinder generalization or compromise reliability.

Section 2.2 starts by examining *data valuation* literature, which seeks to quantify the “value” of individual training samples in shaping model behaviour and performance. Data valuation offers approaches to understanding which samples contribute most to learning,

and conversely, which may be redundant. This discussion naturally transitions into *label noise detection* in Section 2.3, where the focus shifts from valuing samples to identifying those corrupted by incorrect or inconsistent labels. Here, the tension between sample informativeness and label reliability becomes apparent—some samples may appear valuable for training but are, in fact, mislabelled and harmful to generalization. The chapter then turns to *OOD detection* in Section 2.4, a field concerned with identifying inputs that fall outside the model’s training distribution during inference. OOD detection addresses a complementary challenge: while data valuation and label noise detection focus on the quality of the train set, OOD detection safeguards model predictions by detecting unfamiliar inputs at test time. Finally, Section 2.5 presents a discussion on *influence functions*, a well-known tool in robust statistics that can be used to detect a form of interdependency between training samples. We conclude the chapter with a summarizing discussion in Section 2.6.

2.2 Data valuation

It is widely accepted in the deep learning community that modern model improvements largely stem from increasing both model capacity and train set size—the more parameters and data provided, the better the model tends to perform. However, gains in generalization typically follow a *power-law* relationship with respect to train set size, model capacity, or computational resources. For example, Zhai *et al.* [21] showed that achieving state-of-the-art performance on ImageNet [22] with a vision transformer required expanding the pre-training dataset from 1 billion to 3 billion samples, yielding only marginal gains of a few percentage points in test accuracy. This phenomenon, referred to as *neural scaling laws* [23], [24], raises fundamental questions: How much of the training data is truly necessary for generalization? Could certain samples in the train set contribute more to generalization than others? Conversely, could some samples hinder the generalization process?

Efforts to answer these questions have motivated a growing body of research aimed at as-

signing values to individual training samples: a process known as *data valuation* [25]–[29]. This value (or score) typically quantifies some aspect of the learning dynamics induced by each sample which is expected to correlate with the sample’s effect on the model’s overall performance [24], [29]–[31]. Specifically, studies have generally found that samples that are more “difficult” to fit (corresponding to high scores) contribute substantially to model generalization, while “easier” samples (low scores) constitute the majority of the dataset. Importantly, many of these low-score samples can be removed without significantly degrading performance [26], [29]. This process, known as *data pruning*, has become a central tool for examining model generalization from a training data perspective.

Building on this line of inquiry, Sorscher *et al.* [24] demonstrated that the use of data valuation methods can potentially improve upon the previously mentioned power-law scaling, approximating a faster exponential relationship between dataset size and generalization performance. Through an extensive series of data pruning experiments comprising eight pruning methods and two baselines (no pruning and random pruning), they found that most methods fail to scale effectively to large datasets such as ImageNet [22], and that the most effective methods tend to be computationally expensive. They also introduced a theoretical framework for data pruning under simplified conditions (perceptron learning in a student–teacher setting), which suggests that for large datasets, “easy” samples (those with large margins) should be pruned first, whereas for smaller datasets, “hard” samples (with small margins) should be pruned first.

Three of the pruning approaches (“memorization” and two variants of “influence scores”) analysed by Sorscher *et al.* [24] originate from the influential work of Feldman and Zhang [28], which builds upon Feldman’s [32] earlier theoretical insights. Feldman proposed that most natural data distributions are *long-tailed*, i.e., containing a significant fraction of atypical samples that require memorization to achieve near-optimal generalization. Combining this with Zhang *et al.*’s [33] finding that deep networks can perfectly fit even randomly labelled datasets, Feldman and Zhang [28] provided empirical support for the *long-tail theory*. They introduced the *memorization* score, which measures the extent to which a model’s correctness on a sample depends on that exact sample being present during training. Their experiments showed that samples with higher memorization scores

exhibit greater contribution to generalization.

The findings from Feldman and Zhang [28] align with other notable scoring methods, such as the *sample forgetting* score by Toneva *et al.* [26] and the *Error L_2 -norm (EL2N)* score by Paul *et al.* [29], both examined in the comprehensive analysis by Sorscher *et al.* [24]. Toneva *et al.* [26] applied the concept of *catastrophic forgetting* to individual training samples, measuring how often a model forgets a training sample during the learning process. They found that samples with higher forgetting scores contribute more to generalization. Similarly, Paul *et al.* [29] proposed the EL2N score, which quantifies how incorrect the model’s early predictions were by computing the L_2 -norm between the model’s output and the true label. They showed that this score (after rescaling), upper bounds the number of forgetting events for that sample, arriving at the same conclusion: higher-scoring samples are more informative for generalization. Both studies provided empirical validation through data pruning experiments, which we investigate in detail in Chapter 5.

Beyond these methods compared by Sorscher *et al.* [24], several additional data valuation approaches deserve mention. Ghorbani and Zou [25] employed *Shapley values* to estimate each sample’s contribution to model performance, while Yoon *et al.* [30] used *reinforcement learning* for sample scoring. Ghorbani and Zou [25] showed that pruning samples in descending order of their Shapley scores causes a sharper performance drop than pruning at random or using baseline scores, whereas pruning in ascending order tends to improve performance. This suggests that their scoring function reliably reflects each sample’s contribution to generalization. Kwon *et al.* [34] later extended this approach with similar results. Likewise, Yoon *et al.* [30] demonstrated that their reinforcement learning-based scoring method outperforms Ghorbani and Zou’s [25] approach and other baselines across several benchmarks and label-corrupted datasets.

In addition to advancing our understanding of model generalization, data valuation has practical applications. Ghorbani and Zou [25] suggested that these scores could inform how individuals are compensated for their data. Others have proposed their use in domain adaptation, corrupted-sample detection, and robust learning [30]. Despite the diversity

of approaches, a broad consensus has emerged: high-score samples based on sample difficulty play an important role in improving model generalization. Interestingly, several studies [26], [29], [31] have observed that samples with higher scores often coincide with mislabelled or noisy samples; however, they maintain the argument that samples with higher scores tend to be more important for model performance. This observation highlights an important tension: while high-score samples may be informative, they can also signal potential label noise. This intersection between sample importance and label reliability points to a crucial open question, which we explore next: how can we identify and mitigate label noise to ensure that high-value data truly benefit model generalization?

2.3 Label noise detection

As motivated in the previous section, there is a strong need for high-quality labelled datasets. However, the process of creating such datasets remains both costly and time-consuming. To alleviate these burdens, practitioners often turn to scalable yet lower-cost alternatives such as automated web data collection [35], crowdsourcing [36] or automatic labelling algorithms based on natural language processing [19] (as we discuss in Chapter 7). While these methods greatly expand data availability, they also introduce the pervasive issue of *label noise*—cases where samples are incorrectly labelled or inconsistently annotated across annotators [37].

Despite the presence of label noise in datasets, prior research suggests that its impact on deep learning models is often manageable [38]–[40]. Furthermore, studies indicate that dataset size can affect model generalization under noisy conditions. For instance, Sun *et al.* [41] demonstrated that large-scale datasets can reduce the adverse effects of label noise, while Mahajan *et al.* [42] showed that pretraining on billions of potentially noisy samples, followed by finetuning on clean data, yields robust model performance on target tasks. However, these findings warrant cautious interpretation. In the work of Mahajan *et al.* [42], the extent to which the noisy pretraining data was fitted remains unclear, and the role of finetuning is not fully explored. Similarly, Rolnick *et al.* [43] argued that large

neural networks are resilient to significant label noise when trained with sufficient clean samples, yet their study lacks a detailed experimental setup. A reproducibility report [44] noted that these results were only partially replicated under early stopping conditions, highlighting the need for further scrutiny.

Contrary to studies suggesting that label noise poses minimal concern, it is well established that DNNs exhibit a *memorization* effect [33], [45], [46]. DNNs typically learn simple and generalizable patterns early in training, but as learning progresses, they begin to fit harder or mislabelled samples—eventually overfitting noisy labels. This phenomenon can significantly impair generalization, motivating two main strategies for mitigation: 1) designing *noise-robust* learning methods that can tolerate noisy data, and 2) developing *label noise detection* approaches that identify and filter out corrupted samples [47]. The first strategy modifies training procedures through regularization or adjusted loss functions [48]–[50], while the second relies on explicitly detecting mislabelled samples and either removing them [51] or reweighting their influence during training [52], [53].

Each approach has notable trade-offs. Noise-robust learning methods are often simpler to implement and integrate seamlessly into existing pipelines through standard techniques such as dropout, early stopping, or label smoothing. However, they can still suffer from memorization, especially when clean samples are rare or when noise patterns mimic valid intra-class variations [54], [55]. In contrast, label noise detection methods are typically more computationally demanding, requiring additional preprocessing steps or auxiliary models. Yet, these methods frequently outperform noise-robust learning approaches in improving downstream generalization [54], [56], [57]. Moreover, because they explicitly identify potentially corrupted samples, noise detection methods offer interpretability benefits, i.e., clarifying when poor model behaviour stems from problematic data rather than inherent model flaws.

Recent advances in label noise detection illustrate the evolving sophistication of this field. Zhu *et al.* [58] observed that traditional assumptions—such as the existence of anchor points¹ or well-separated clusters—often fail when feature representations are

¹*Anchor points* refer to samples whose observed (possibly noisy) labels are known with high confidence to be correct.

weak (meaning the learned data embeddings do not effectively separate different classes). They proposed an information-theoretic framework that down-weights less informative features to enhance noise detection. In subsequent work [54], they leveraged the intuition that nearby samples in representation space tend to share correct labels, using a k -nearest-neighbours (k -NNs)-based local voting mechanism to filter likely noisy samples. Chen *et al.* [59] extended this line of research by exploiting the insight that networks trained on noisy datasets often internalize the structure of label corruption; by comparing predictions through cross-validation, they identified clean samples as those whose predicted and observed labels consistently agree across the different cross-validation models.

The connection between *data valuation* and *label noise detection* is less explored than one might expect. In the label noise detection literature, evaluation typically relies on metrics such as the F1-score or precision and recall of detected noisy labels, based on ground truths [47], [54], [60]. Occasionally, performance is assessed by reporting the change in evaluation accuracy after removing samples identified as noisy [51], [56], [59]. To our knowledge, data pruning experiments—originally developed to assess sample importance (value)—have only been explicitly repurposed to evaluate label error detectors in the work of Pleiss *et al.* [61]. They introduced the Area Under the Margin (AUM) statistic and demonstrated that ranking and removing low-AUM samples consistently improved model performance relative to random pruning. Similarly, Northcutt *et al.* [56] proposed Confident Learning (CL), a probabilistic framework that identifies likely label errors based on class probability estimates. Through a variant of data pruning, they showed that progressively removing larger portions of the samples flagged as mislabelled led to steady gains in test accuracy. In both cases [56], [61], removing such samples improved generalization, reinforcing the view that some “hard” samples are not informative but rather detrimental to learning. This raises an intriguing question: if some hard samples are mislabelled and harmful, while others are challenging yet valuable, is the distinction solely due to label correctness? A possible explanation extends beyond label quality—perhaps some of these difficult yet correctly labelled samples exhibit input noise or atypical feature patterns that deviate from the core training distribution.

This observation naturally leads to the next discussion: distinguishing label noise from

OOD phenomena. While label noise pertains to incorrect labels within the label space of the dataset, OOD detection concerns samples whose labels or input features differ fundamentally from those seen during training. Understanding this boundary is critical, not only for improving robustness, but also for ensuring that models trained on “cleaned” data remain reliable when faced with genuinely novel or shifted inputs.

2.4 Out-of-distribution detection

OOD detection is an important task for deploying Machine Learning (ML) models safely and reliably in real-world environments. Its importance lies in the ability to prevent models from making confident yet erroneous predictions when encountering unfamiliar data. For instance, a medical imaging model trained to classify X-ray images might be inadvertently presented with an MRI scan, or a system trained to identify bird species could receive images of mammals or human hikers from camera traps. In such cases, forcing the model to predict regardless of uncertainty can lead to misleading or harmful outcomes. OOD detection provides a critical safeguard by enabling models to recognise and flag uncertain or unfamiliar inputs rather than producing unreliable predictions [62].

Yang *et al.* [62] note that most ML models are trained under the *closed-world assumption* [63], [64], which presumes that training and test samples are drawn independent and identically distributed (i.i.d.) from the same distribution, often referred to as In-distribution (ID). However, real-world scenarios are inherently *open-world*, where test samples may differ significantly from the training data. These OOD instances must therefore be detected and treated with caution to ensure reliable model performance [65], [66].

The term *OOD detection* was first introduced by Hendrycks and Gimpel [67], marking the beginning of an active research area. Although related to anomaly detection, novelty detection, open-set recognition, and outlier detection, OOD detection has distinct goals and assumptions. Yang *et al.* [62] provide one of the few comprehensive clarifications of these differences, categorizing *generalized OOD detection* methods into five overlapping yet distinct paradigms: 1) anomaly detection, which identifies samples that deviate from

a predefined normal pattern; 2) novelty detection, which flags instances that do not belong to any of the known classes; 3) open-set recognition, which jointly classifies known samples and rejects unknown ones; 4) OOD detection, which distinguishes between ID and OOD samples during testing; and 5) outlier detection, which identifies samples that differ substantially from the rest of a dataset due to covariate- or semantic shifts. While the terminology across studies is often used interchangeably [57], [68], [69], these distinctions highlight the breadth of problem formulations under the OOD detection umbrella.

Early methods approached OOD detection by leveraging model confidence. Hendrycks and Gimpel [67] proposed a simple yet influential baseline that uses the *Maximum Softmax Probability (MSP)* of a model output as a confidence score: samples with a lower softmax probability for the labelled class are more likely to be OOD and can be detected with a threshold. Despite its simplicity, MSP-based detection suffers from a critical weakness—modern deep networks are often *overconfident* even on inputs that lie far outside the training distribution [70], [71]. This overconfidence stems from the network’s tendency to overfit the training data, resulting in unreliable uncertainty estimates.

To mitigate this issue, Lee *et al.* [68] introduced a *distance-based* approach that operates in the hidden feature space rather than the output space. They modelled class-conditional embeddings as Gaussian distributions and computed the Mahalanobis distance [72] between a test sample and its nearest class mean to derive an OOD confidence score. Although somewhat effective, this approach assumes that these representations are normally distributed—an assumption that does not always hold, as feature embeddings can fail statistical normality tests [73]. To relax this assumption, non-parametric approaches such as k -NNs have been explored [73]–[75], often outperforming Gaussian-based methods in complex settings. Interestingly, these non-parametric approaches echo techniques used in label noise detection [54], [76], suggesting a conceptual link between detecting mislabelled samples and identifying OOD instances.

Despite the importance of OOD detection, real-world models are often expected to generalize to inputs that have undergone slight distributional shifts: instances that share the same label space as the ID data but differ due to environmental, sensor, or contextual

factors. Training models to generalize to these kinds of shifts is a challenging task and an active area of research. Recent works [77], [78] highlight an inherent trade-off between improving OOD detection and maintaining model generalization: methods that aggressively enforce high uncertainty on auxiliary² OOD samples may inadvertently penalise valid, shifted inputs. Zhang *et al.* [77] termed this tension the *sensitive-robust dilemma*: improving sensitivity to OOD samples can reduce robustness to natural variations within the ID distribution.

This dilemma points to a broader observation: exposing models to OOD samples during training, whether intentionally as a form of regularization or unintentionally through dataset contamination, can degrade downstream generalization to clean ID data. This effect aligns with the prior findings on memorization in neural networks [46], [79], [80] as discussed in the previous section. While the memorization phenomenon has been extensively studied in the contexts of label corruption and adversarial samples [81]–[83], the role of input noise, i.e., OOD samples arising from covariate- or semantic shifts within the train set, remains underexplored. This overlooked aspect forms the basis of the analysis presented in Chapter 6.

2.5 Influence functions

Previous sections on data valuation, label noise detection, and OOD detection address the problem of assigning scores to individual training samples—whether based on their contribution to model performance, label quality, or deviation from the training distribution. Although these approaches analyse model behaviour from the perspective of individual data points, they share a fundamental limitation: they largely ignore the collective effect and interdependencies among samples. This limitation becomes particularly salient in data pruning and subset selection, where assigning independent scalar scores to samples may fail to capture the complex interactions that jointly influence model learning.

Influence functions offer a principled approach to partially address this issue by esti-

²Intentionally introduced during training.

mating how infinitesimal perturbations to individual training samples affect the model’s parameters and predictions. Originating from robust statistics in the 1970s—most notably through the works of Hampel [84] and Jaeckel [85] under the name *infinitesimal jackknife*—influence functions were initially developed for understanding the sensitivity of estimators to data perturbations. Early applications focused on linear models [86], [87] and were later generalized to non-linear settings [88]. Formally, Cook and Weisberg [89] showed that upweighting³ a training sample $(\mathbf{x}, y)$ by infinitesimally small value ϵ changes all of the linear regression model parameters ϕ according to:

$$\mathcal{I}(\mathbf{x}, y) \stackrel{\text{def}}{=} \left. \frac{d\phi_{\epsilon, \mathbf{x}, y}}{d\epsilon} \right|_{\epsilon=0} = -H_{\phi}^{-1} \nabla_{\phi} \ell(\mathbf{x}, y), \quad (2.1)$$

where $H_{\phi} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \nabla_{\phi}^2 \ell(\mathbf{x}_i, y_i)$ is the Hessian of the empirical risk across all n training samples, ℓ denotes the loss function and $\mathcal{I}(\mathbf{x}, y)$ denotes the *parameter influence* of sample $(\mathbf{x}, y)$ on all of the model parameters ϕ (where $\phi_{\epsilon, \mathbf{x}, y}$ simply indicates that ϕ is a function of the upweighting ϵ applied to the specific sample $(\mathbf{x}, y)$). Intuitively, Equation 2.1 asks: “If we slightly increase the *push* or *importance* of a single sample $(\mathbf{x}, y)$ on the training process, how do the final model parameters ϕ change?”

Despite their strong theoretical foundation in statistics, classical influence functions face significant challenges in modern deep learning. They assume that the model is both convex and twice differentiable, conditions rarely satisfied by contemporary neural networks. While first-order derivatives are readily computed via backpropagation, many architectures include non-smooth components such as Rectified Linear Unit (ReLU) activations or max-pooling, which make the loss surface only piecewise differentiable and the Hessian undefined at certain points. Moreover, even when the model is locally differentiable, computing and inverting the Hessian of the empirical risk H_{ϕ} incurs a cost of $O(nq^2 + q^3)$ for n training samples and q parameters, rendering exact computation intractable for large models [90].

Koh and Liang [90] overcame these theoretical and computational obstacles by introducing a tractable approximation of influence functions for DNNs. They showed that influ-

³Upweighting a training sample by a value ϵ means that the sample’s contribution to the overall empirical risk (or loss function) is being incremented by ϵ .

ence can be efficiently estimated using second-order optimization techniques—specifically, Hessian-vector products (HVPs) [91], [92]—which avoid explicit Hessian computation or inversion. Their key insight was that removing a sample $(\mathbf{x}, y)$ from the train set is approximately equivalent to upweighting it by $\epsilon = -\frac{1}{n}$, leading to the linear Taylor approximation:

$$\boldsymbol{\phi}_{-(\mathbf{x}, y)} - \boldsymbol{\phi} \approx -\frac{1}{n} \mathcal{I}(\mathbf{x}, y), \quad (2.2)$$

where $\boldsymbol{\phi}$ is the model parameters trained on all n training samples and $\boldsymbol{\phi}_{-(\mathbf{x}, y)}$ is the model parameters trained on the train set, excluding sample $(\mathbf{x}, y)$. This formulation enables efficient estimation of the parameter change $(\boldsymbol{\phi}_{-(\mathbf{x}, y)} - \boldsymbol{\phi})$ due to sample removal without retraining the model. Koh and Liang [90] demonstrated that, even when the strict assumptions of convexity and smoothness are relaxed, this approximation remains surprisingly accurate in practice, making influence functions applicable to high-dimensional DNNs. Their approach laid the groundwork for subsequent research leveraging influence estimation for interpretability in DNNs [93], [94].

Building upon these foundations, Koh *et al.* [95] extended the framework to study *collective influence*. They observed that, under the linear approximation, the cumulative effect of multiple samples equals the sum of their individual influences. Leveraging this property, Yang *et al.* [96] formulated a discrete optimization problem to identify subsets of training samples with minimal collective influence, which are redundant subsets suitable for pruning. Their approach outperformed alternative scoring-based pruning techniques such as sample forgetting [26] and EL2N [29].

Understanding and estimating group-level influence can provide deeper insight into inter-sample dependencies within datasets. We revisit parameter influence in Chapter 7.

2.6 Discussion

The works reviewed in Sections 2.2–2.5 represent four distinct yet interconnected strands of research within the broader ML literature. Each addresses a different aspect of data

quality and model reliability.

Data valuation methods focus on quantifying the contribution of individual training samples to the learning process and model performance. **Label noise detection** methods, in contrast, are designed to identify and mitigate the impact of mislabelled training samples—either by removing them entirely or by adjusting their weighting during training to reduce their negative impact. **OOD detection** methods operate at inference time, identifying inputs that fall outside the training distribution to prevent the model from producing unreliable predictions. Research on **influence functions** is a sub-branch of data valuation, specifically focused on estimating the impact of training samples on the model’s parameters.

Beyond their conceptual differences, these research areas also employ distinct evaluation metrics. Data valuation studies typically rely on data pruning experiments to assess whether removing low-value samples maintains model generalization. Label noise detection approaches are commonly evaluated using metrics such as the F1-score or precision and recall for detecting mislabelled samples, based on some ground truth. In contrast, OOD detection methods are evaluated based on their ability to distinguish ID from OOD samples, using metrics such as AUROC and FPR95.

From this literature review, a key insight emerges: techniques developed across these four domains can each be reframed as methods for detecting *problematic samples*. Whether the issue arises from the sample’s informational value, incorrect supervision, or distributional mismatch, the central challenge remains the same—determining which samples improve or degrade model performance and understanding why. This observation forms the foundation for the work presented in the following chapters.

Additionally, our review of the data valuation literature (Section 2.2) revealed a notable gap: to the best of our knowledge, no prior work has formally presented a data pruning evaluation framework. The absence of such a framework has led to substantial inconsistencies across studies. For example, researchers often use the same dataset and model yet prune different proportions of the data [26], [31], and others do not explicitly specify how

many samples are removed at each step [29]. Moreover, studies seldom specify if scores should be computed once (before evaluation) or iteratively (during evaluation) [26], how extensively pruning should be performed [26], [29], or whether pruning should proceed in ascending or descending order of the scores if there is label noise in the train set [26], [29], [31]. Finally, based on contradictory approaches [29], [30], it is not clear whether the models used for scoring should also be used for the pruning experiments (along with identical hyperparameters and training criteria). These shortcomings motivated us to consolidate and formalize an evaluation framework, which forms the basis of our analysis in Chapter 3.

Next, in Chapter 3, we formally introduce the data pruning evaluation framework drawn from the data valuation literature and demonstrate how it can be adapted to compare different problematic sample detectors. Chapter 4 validates this framework in a controlled environment using synthetically generated data. In Chapter 5, we evaluate two prominent data valuation methods: *Sample forgetting* by Toneva *et al.* [26] and the *EL2N* metric by Paul *et al.* [29], to test their ability to detect problematic samples rather than merely redundant ones. Chapter 6 extends this investigation by applying representative methods from the label noise detection and OOD detection domains. Specifically, *CL* by Northcutt *et al.* [56] and the *k-NN distance* method by Kuan and Muller [74], to the same evaluation framework. Finally, in Chapter 7, we present a comparative analysis of all four methods and revisit parameter influence to detect interdependencies between samples.

Chapter 3

Evaluation framework for problematic sample detectors

We outline the problem and evaluation framework used to investigate problematic sample detectors throughout this study.

3.1 Overview

In this study, the goal of a problematic sample detector is to assign each training sample a score that reflects the extent to which it hinders model generalization. This naturally leads to several questions: How can we identify a *candidate set* of problematic samples based on these scores? How can we determine whether the identified samples are *truly problematic* for the model or task? And if they are, to *what degree* do they impact performance? Furthermore, how do we *compare* the performance of different problematic sample detectors without having a concrete measure of the ground truth problematic score of samples?

While later chapters consider ways to obtain these scores, this chapter introduces the data pruning evaluation framework, which offers a systematic approach to addressing these questions.

Section 3.2 presents the problem formulation, followed by Section 3.3 that defines a generalized problematic sample detector as a mapping. Section 3.4 formally introduces the proposed data pruning evaluation framework, which serves as the foundation for the analyses presented throughout the study, followed by Section 3.5 that defines three types of OOD samples that may constitute problematic samples. Finally, Section 3.6 concludes the chapter with a discussion and some limitations of the evaluation framework.

3.2 Problem formulation

In a supervised classification problem, let the input be $\mathbf{x} \in X \subseteq \mathbb{R}^p$ and the output label be $y \in Y = \{1, 2, \dots, c\}$. For simplicity, we denote the input $\mathbf{x}$ as a vector, as is typical when building a MLP model. However, in CNN-based architectures for image classification, the input is a multidimensional tensor $\mathbf{X} \in \mathbb{R}^{h \times w \times k}$, where h and w denote the height and width of the image in pixels, respectively, and k represents the number of channels ($k = 1$ for grayscale images and $k = 3$ for RGB images). The label y is a scalar identifying a single class among the c mutually exclusive categories, thereby excluding multilabel classification. This study considers both binary classification, where $c = 2$, and multiclass classification, where $c > 2$. The labelled dataset is defined as $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each pair $(\mathbf{x}_i, y_i)$ is called a *sample*. We partition D into two disjoint subsets: $D_{\text{train}} \cup D_{\text{test}}$.

We define a problematic sample as any sample $(\mathbf{x}, y)_{\text{prob}} \in D_{\text{train}}$ that negatively affects the expected performance of the model on D_{test} . If we assume that the contribution (to performance) of one training sample is independent from any other, then a model trained on $D_{\text{train}} \setminus \{(\mathbf{x}, y)_{\text{prob}}\}$ will outperform a model trained on D_{train} , with the absolute difference between these two levels of performance indicating the degree to which $(\mathbf{x}, y)_{\text{prob}}$ is problematic. Unfortunately, this assumption is often unrealistic. See Fig. 3.1 for an

example of how this assumption can easily be wrong.

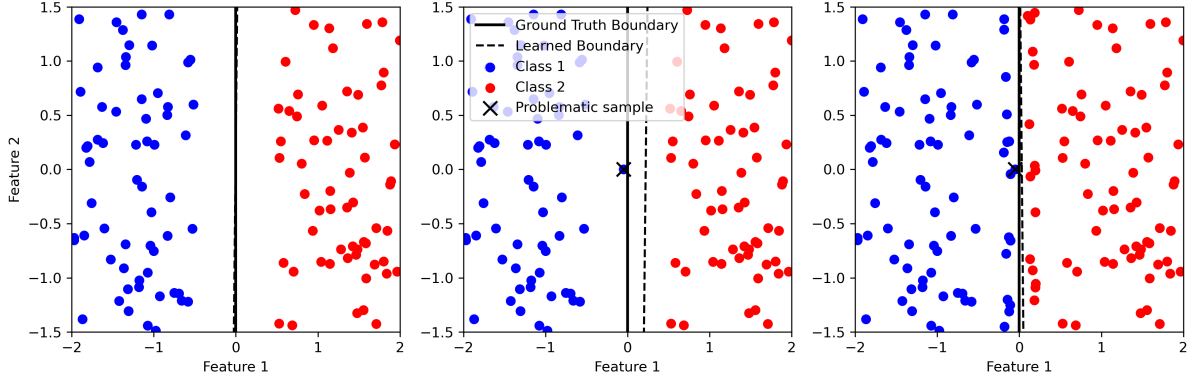


Figure 3.1: A simple example of possible dependence between samples and the degree to which they are problematic. In all three scenarios we have two dimensional data that can be categorized into one of two classes based on a ground truth boundary across the first feature dimension. (left) The ground truth boundary is accurately approximated by a simple logistic regression solver. (middle) A single problematic sample near the boundary shifts the learned boundary away from the ground truth. (right) The presence of other samples near the boundary results in the performance degradation being significantly reduced.

We require a method to estimate the degree to which samples in D_{train} are problematic. In addition, we require a framework to evaluate the accuracy of such an estimator without having access to a ground-truth.

3.3 Problematic sample detector

As is conventional in the literature [9], [97], [98], we consider a set of functions¹ F where each $f \in F$ represents a possible mapping from inputs to outputs $f : X \rightarrow Y$. Since we do not have access to the data generating distribution to minimize the true risk, the *learning process* instead identifies a $\hat{f} \in F$ that best fits the train set D_{train} by minimizing

¹Often referred to as the *hypothesis space*.

the empirical risk R_{emp} :

$$\hat{f} = \arg \min_{f \in F} R_{\text{emp}}(f, D_{\text{train}}), \quad (3.1)$$

where the empirical risk is defined as:

$$R_{\text{emp}}(f, D_{\text{train}}) = \frac{1}{|D_{\text{train}}|} \sum_{(\mathbf{x}_i, y_i) \in D_{\text{train}}} \ell(f(\mathbf{x}_i), y_i), \quad (3.2)$$

where $\ell : Y \times Y \rightarrow \mathbb{R}^+$ is a *loss function* that measures the discrepancy between the model prediction $f(\mathbf{x}_i)$ and the label y_i . In practice, the learning process does not explicitly search over all $f \in F$. Instead, it initializes a parametrized model f (e.g., via a scheme such as Kaiming [64] initialization) and iteratively updates its parameters using a gradient-based optimizer (e.g., SGD [99] or Adam [100]) to minimize the empirical risk on the train set. For simplicity, we denote this *learning process* by the mapping:

$$\mathcal{A} : D \times F \rightarrow F, \quad (D_{\text{train}}, f) \mapsto \mathcal{A}(D_{\text{train}}, f) = \hat{f}, \quad (3.3)$$

where D is the set of possible train sets and F is the set of possible functions (hypothesis space). We typically refer to $\hat{f}$ as a *trained model* and the f that the learning process is initialized with as an *untrained model*.

In this study, we provide the following simple definition for a *problematic sample detector* (sometimes, simply referred to as *detector*) for the context in which we investigate it. A problematic sample detector Λ is defined as a mapping that assigns scores to each sample in the train set. Specifically, for a given train set D_{train} , untrained model f and learning process $\mathcal{A}$, the detector is defined as:

$$\Lambda : D \times F \times \mathcal{A} \rightarrow \mathbb{R}^{|D_{\text{train}}|}, \quad (D_{\text{train}}, f, \mathcal{A}) \mapsto \Lambda(D_{\text{train}}, f, \mathcal{A}) = \mathbf{s}. \quad (3.4)$$

The detector Λ produces a score vector $\mathbf{s} = (s_1, s_2, \dots, s_{|D_{\text{train}}|})^\top$, where $s_i \in \mathbb{R}$ corresponds to the *problematic score* assigned to the i -th sample $(\mathbf{x}_i, y_i) \in D_{\text{train}}$.

Finally, a *performance metric* $\mathcal{P}$ is defined as:

$$\mathcal{P} : D \times F \rightarrow \mathbb{R}, \quad (D_a, \hat{f}) \mapsto \mathcal{P}(D_a, \hat{f}), \quad (3.5)$$

where $\mathcal{P}$ maps the dataset $D_a \in \{D_{\text{train}}, D_{\text{test}}\}$ to a scalar performance real-value (e.g., accuracy or AUROC) of the trained model $\hat{f}$ on dataset D_a .

The goal of the evaluation framework outlined in Section 3.4 is to use the scores $\mathbf{s}$ from the problematic sample detector Λ , to identify a possible subset of D_{train} that hinder generalization to the target distribution $P(\mathbf{x}, y)$, as measured by the test performance $\mathcal{P}(D_{\text{test}}, \hat{f})$. Additionally, it also provides a method of comparing the performance of different detectors.

3.4 Data pruning evaluation framework

Up to this point, this chapter has introduced the problem formulation and defined a problematic sample detector as a mapping that outputs a vector of scores corresponding to the train set. We now turn to the central questions outlined in the chapter overview. Given the score vector $\mathbf{s}$ produced by the problematic sample detector Λ , we seek to answer:

1. How can we identify a *candidate subset* of problematic samples within D_{train} based on the scores $\mathbf{s}$?
2. How can we determine whether this subset is *indeed problematic* for the model or task?
3. If the subset is confirmed to be problematic, to *what degree* does it affect overall model performance?
4. Finally, how can we *compare* the performance of different problematic sample detectors in the absence of ground-truth labels indicating which samples are truly problematic?

Developing an evaluation framework capable of addressing these questions presents several challenges. First, the absence of explicit ground truth makes it difficult to validate whether a detected subset truly corresponds to problematic samples, introducing ambiguity in both qualitative and quantitative analyses. Second, the effect of problematic samples

often arises from complex interactions between data characteristics, model architecture, and learning dynamics—factors that are not easily disentangled.

We turn to prior literature to examine how others have approached similar issues. In particular, we find that the field of *data valuation* frequently employs “data pruning experiments” to identify candidate sets of *redundant samples* [25], [26], [28], [29]. These are experiments where a varying portion of the train set is removed and models are retrained to estimate the impact of the removed samples. This raises the question: can the same framework be adapted to detect problematic samples? We also observe that data pruning experiments are rarely, if ever, formally defined—forcing readers to infer its design choices, inputs, and hyperparameters indirectly from experimental results. To address this gap, we formally present our data pruning evaluation framework in Algorithm 1.

Algorithm 1 produces model performance values (e.g., accuracy) measured on the test set, as an increasing portion of the train set is pruned. Specifically, Algorithm 1 generates a ψ -curve, with $\psi = (\psi_1, \dots, \psi_m)^\top \in \mathbb{R}^m$ (where m is the length of the curve calculated according to Equation 3.6) for every detector Λ that is being compared. In essence, Algorithm 1 iterates over three steps:

1. Reduce the train set by using the scores $\mathbf{s}$ from detector Λ
2. Train a new model f on the reduced train set
3. Test the trained model on the test set

These three steps produce a ψ -curve, whose components $(\psi_1, \dots, \psi_m)$ correspond to the expected model performance, on a held-out test set, as an increasing portion of the train set is pruned.

If the scores $\mathbf{s}$ correlate with how problematic each training sample is in terms of overall model performance, the ψ -curve should either maintain stable performance or have an initial upwards trend as more samples are being pruned. This is because Algorithm 1 ranks the train samples in descending order, meaning the most problematic samples should be pruned first. In general, an unbiased ranking of the train set should correspond to a

gradual downward slope in the ψ -curve since model performance is known to be dependent on the train set size.

By using different approaches to scoring and ranking the train set we can compare different problematic sample detectors. This allows for the investigation of the role of specific samples in the train set and how they contribute to model performance.

The length m of the ψ -curve is given by:

$$m = \frac{\alpha \cdot |D_{\text{train}}|}{r} + 1, \quad (3.6)$$

where $\alpha \in (0, 1]$ is a fraction (rational number) parameter that specifies the maximum proportion of D_{train} to be considered for removal, and $r \in \mathbb{N}$ is a resolution hyperparameter controlling the step size. The hyperparameter r satisfies $1 \leq r \leq \alpha \cdot |D_{\text{train}}|$ and is chosen such that $\alpha \cdot |D_{\text{train}}|$ is divisible by r , ensuring that $m \in \mathbb{N}$. Note that the fraction in Equation 3.6 is incremented by 1, since the first component of the ψ -curve (i.e., ψ_1) contains the test set performance of a model trained on the full train set. We compute m beforehand and pass it as input to Algorithm 1, since every ψ -curve for every detector Λ being compared will have the same length.

Algorithm 1: Generating a ψ -curve

Input: $D_{\text{train}}, D_{\text{test}}, \mathbf{s}, \mathcal{P}, \mathcal{A}, f, r, m$ as defined above.

Output: $\psi \in \mathbb{R}^m$

```

1  $\mathbf{z} = (z_1, z_2, \dots, z_{|D_{\text{train}}|}) \leftarrow \text{argsort}(-\mathbf{s})$ 
2  $\psi \leftarrow (0, 0, \dots, 0) \in \mathbb{R}^m$ 
3  $\psi_1 \leftarrow \mathcal{P}(D_{\text{test}}, \mathcal{A}(D_{\text{train}}, f))$ 
4 for  $i \in \{1, 2, \dots, m - 1\}$  do
5    $I_{\text{remove}} \leftarrow \{z_1, z_2, \dots, z_{(r \times i)}\}$ 
6    $D_{\text{trunc}} \leftarrow D_{\text{train}} \setminus \{(\mathbf{x}_j, y_j) : j \in I_{\text{remove}}\}$ 
7    $\psi_{i+1} \leftarrow \mathcal{P}(D_{\text{test}}, \mathcal{A}(D_{\text{trunc}}, f))$ 
8 return  $\psi$ 
```

Note that if we want to compare different detectors, all inputs in Algorithm 1 are kept identical for every Λ we want to compare. When $\alpha = 1$ in Equation 3.6, the detector Λ

is evaluated across the entire train set, producing a ψ -curve that spans the full range of sample removals. Figure 3.2 provides an illustration of the expected ψ -curves for three different detectors: one effective detector, one ineffective detector, and one that assigns random scores to samples.

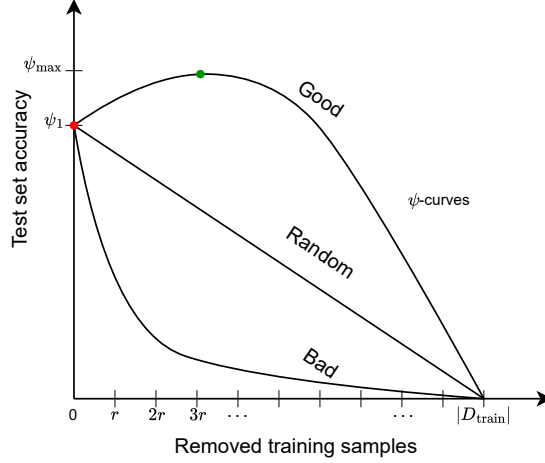


Figure 3.2: An illustration of ψ -curves for Algorithm 1 when the performance metric $\mathcal{P}$ is accuracy. The red dot represents the model performance $\psi_1 = \mathcal{P}(D_{\text{test}}, \hat{f})$ when trained on the full train set. The green dot illustrates the maximum model performance $\psi_{\text{max}} = \mathcal{P}(D_{\text{test}}, \hat{f})$ after $3r$ samples of the train set have been removed.

From the illustration in Figure 3.2, a gradual decrease in test set accuracy is expected when samples are removed at random. This follows from the well-established relationship between model performance and train set size: larger train sets generally enable models to achieve higher test set accuracy by providing more information from which to learn underlying patterns (assuming the model has sufficient capacity to learn the underlying function). In contrast, if problematic samples are present in the train set, an effective detector that assigns high scores to problematic samples would initially yield an increase in test set accuracy when such samples are removed. Conversely, an ineffective detector that assigns higher scores to beneficial samples would result in a ψ -curve exhibiting worse performance than random sample removal.

Considering Algorithm 1 together with the illustration of the ψ -curves in Figure 3.2, the

following questions can be addressed:

1. *How can we determine whether problematic samples are present in the train set?*

The presence of problematic samples in D_{train} is indicated if test set performance, $\mathcal{P}(D_{\text{test}}, \hat{f})$, improves after the removal of $r \times i > 0$ samples.

2. *How can we identify a candidate set of problematic samples based on these scores?*

If the condition in 1) holds, the subset of samples whose removal yields the maximum performance is defined as $D_{\text{prob}} \subseteq D_{\text{train}}$. In Figure 3.2, D_{prob} corresponds to the $3r$ samples removed between the red and green dots.

3. *How can we determine whether the identified samples are truly problematic for the model or task?*

The absence of explicit ground truth makes it difficult to validate whether D_{prob} is truly problematic. This subset represents our best estimate of the problematic portion of the train set using the relevant detector.

4. *To what degree does the problematic subset impact performance?*
- Once D_{prob} is identified, the degree to which these samples are problematic can be quantified by:

$$\Delta\psi = \psi_{\text{max}} - \psi_1, \quad (3.7)$$

where $\psi_{\text{max}} = \max(\psi)$ denotes the component with the maximum value in vector ψ (maximum performance) and ψ_1 represents the baseline performance prior to any sample removal. Throughout this study, we will simply refer to the degree to which the candidate set is problematic as the *impact*.

5. *How do we compare the performance of different problematic sample detectors?*

The ability of a detector to distinguish between problematic and non-problematic samples can be quantified by the *area under the ψ -curve (AUC)*. The *AUC* serves as a comparative measure of detector performance: larger *AUC* values indicate more effective detectors.

The *AUC* is particularly effective as a comparative metric because it takes into account both $\mathcal{P}(D_{\text{test}}, \hat{f})$ and $|D_{\text{prob}}|$, thereby capturing improvements in generalization performance as well as the size of the problematic subset. More specifically, an increase in test

set performance, or an increase in the problematic subset size, will result in an increase in the AUC metric.

In the next section, we outline three types of samples and their distinctions that we might expect to be present in D_{prob} .

3.5 Out-of-distribution samples

In this section, we define three distinct types of *OOD samples* that we might expect to be present in D_{prob} or that can be used to *induce* problematic samples in D_{train} . OOD samples differ from problematic samples: while problematic samples can arise either within the ID data or from outside it (thus being OOD), OOD samples are not inherently problematic by definition (i.e., not all OOD samples are problematic). Nevertheless, deliberately generating OOD samples and incorporating them into the train set is an effective way to induce problematic cases, since OOD data often present challenges for the model fitting the data.

Within D_{train} , we define an OOD sample as one drawn from a distribution $\tilde{P}(\mathbf{x}, y)$ that differs from the target distribution $P(\mathbf{x}, y)$. We assume the existence of an underlying target distribution from which data is generated, such that samples drawn from this distribution are generally non-problematic. This target distribution can be clearly defined in the case of synthetic datasets but is typically unknown in complex tasks such as image classification. The distributional shift can occur in the input space X or the label space Y . In this study, we separate the distributional shifts into three categories:

- **Covariate shift** This type of shift occurs when the marginal input distribution changes while preserving the conditional label distribution:

$$\tilde{P}(\mathbf{x}) \neq P(\mathbf{x}), \quad \text{but} \quad P(y | \mathbf{x}) = \tilde{P}(y | \mathbf{x}).$$

New inputs $\tilde{\mathbf{x}}$ are sampled from the shifted distribution $\tilde{P}(\mathbf{x})$, with labels predicted using the unchanged conditional $\tilde{P}(y | \mathbf{x})$. Examples include perturbed inputs, such

as rotated or blurred images.

- **Semantic shift** For this type of shift, the conditional label distribution becomes undefined due to inputs from novel, unknown classes:

$$\tilde{\mathbf{x}} \sim \tilde{P}(\mathbf{x}), \quad \text{where} \quad P(y \mid \tilde{\mathbf{x}}) \text{ is undefined for } y \in Y.$$

Therefore, inputs originate from distributions associated with classes outside Y , making no existing label y appropriate. For example, in a digit classification task trained on images of numbers, an input image of a *shoe* would fall outside the label space and represent a semantic shift.

- **Label corruption** This type of shift occurs when the input distribution remains unchanged while the observed label is corrupted:

$$P(\mathbf{x}) = \tilde{P}(\mathbf{x}), \quad \text{but} \quad \tilde{y} \neq y \text{ for } y, \tilde{y} \in Y.$$

An example is a mislabelled image of a *dog* annotated as *cat* in a binary cat–dog classification task, where both classes are present in Y .

Figure 3.3 shows examples of these three categories of distributional shifts. These three categories highlight that, although all label-corrupted samples are OOD, not all OOD samples arise from label corruption. For clarity, we will refer to OOD samples that are not label-corrupted—specifically, those affected by a covariate- or semantic shift—as *input noise*. In contrast, we will refer to mislabelled or label-corrupted samples as *label noise*. Throughout this study, we will examine all three categories of distributional shifts and evaluate the extent to which each contributes to problematic samples.

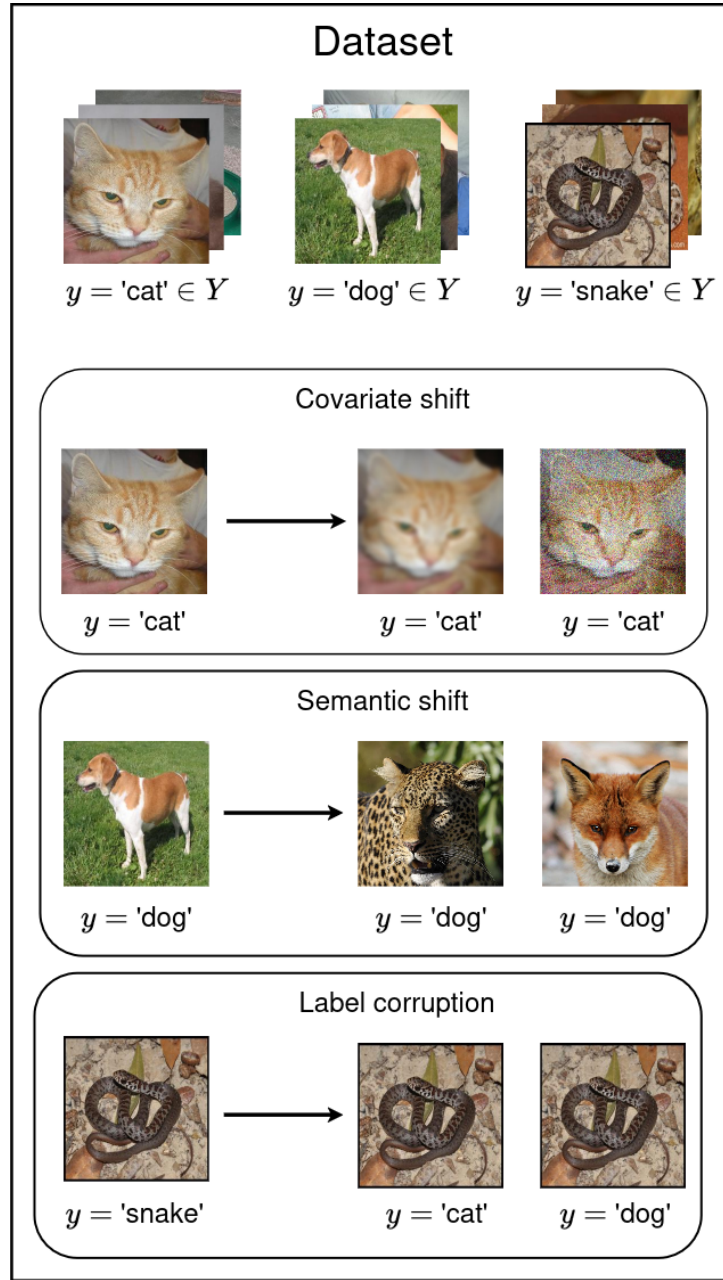


Figure 3.3: Examples of the three categories of distributional shifts for a three-class classification problem with $y \in Y = \{\text{'cat'}, \text{'dog'}, \text{'snake'}\}$. The original sample is shown on the left with two examples of the distributional shift on the right. Note that for covariate- and semantic shifts, the input $\mathbf{x}$ is changed while the label y stays constant. For label corruption, the input $\mathbf{x}$ stays constant while the label y is changed within the label space Y .

3.6 Discussion

In this chapter, the problem formulation was introduced, and three distinct types of OOD samples that may negatively affect the model performance were defined. A problematic sample detector Λ was formally defined as a mapping from the train set, model and learning process, to a vector of real-valued scores, each reflecting the estimated impact of the corresponding training sample. In addition, the evaluation framework was presented in Algorithm 1, which generates a ψ -curve whose components capture the test set performance of models trained on progressively pruned subsets of the train set.

Although the proposed evaluation framework addresses most of the questions outlined in this chapter, it still has several limitations that should be acknowledged:

1. Although the evaluation framework rewards detectors that successfully distinguish problematic from non-problematic samples, it does not necessarily prioritize detectors that maximize generalization performance across any train set size. For example, a detector might score samples such that their ranking and removal produce the largest increase in generalization (compared to other detectors) at a specific point on the ψ -curve. However, if another detector yields consistently better generalization across most pruning levels, the evaluation framework would favour this detector over the first (even if it never achieves the single highest improvement).
2. The total impact of problematic samples may be unevenly distributed: a small number of highly problematic samples may have disproportionately large effects, while the majority of samples remain non-problematic. A purely ranking-based system ignores the magnitude of the impact.
3. Complex interdependencies likely exist between samples. Removing certain samples may alter the problematic nature of others, leading to non-linear and potentially unstable dynamics. Iterative pruning strategies may therefore exhibit unintended behaviours that complicate evaluation.

Nevertheless, the data pruning paradigm has been widely recognized as an effective eval-

uation framework for assessing sample redundancy within train sets [24], [26], [29]. Extending this approach to problematic sample detection provides a principled means for comparing detectors across the literature.

The subsequent chapter applies this evaluation framework to controlled synthetic datasets. This experimental setting enables a systematic investigation of whether the OOD sample types identified in this chapter indeed manifest as problematic under the pruning-based evaluation. Furthermore, it allows examination of whether the empirical ψ -curves align with the expected trends illustrated in Figure 3.2.

Chapter 4

Controlled evaluation using synthetic data

We perform controlled experiments on synthetic data to examine the effect of OOD training samples on the evaluation framework.

4.1 Overview

In previous chapters, problematic samples have been conceptually defined but not explicitly tested. In particular, instances of the three types of OOD samples introduced in Section 3.5 have not yet been empirically tested to determine whether they can indeed be problematic in practice. Similarly, although the data pruning evaluation framework has been employed in prior work to identify redundant samples [24], [26], [29], it remains unclear whether the same framework is effective for detecting problematic samples.

The purpose of this chapter is to develop intuition about problematic samples and to evaluate whether they can be systematically identified using the proposed framework. To

this end, controlled experiments are performed on synthetic datasets, enabling a rigorous assessment of the framework’s behaviour under well-defined conditions.

Section 4.2 describes how the synthetic data is generated, introduces different types of label- and input noise, and outlines the experimental setup. Sections 4.3 and 4.4 present the empirical results for binary- and multiclass classification, respectively. Finally, Section 4.5 concludes the chapter with a discussion of our findings.

4.2 Methods

This section describes our procedure for generating synthetic data by sampling from Gaussian distributions. It also introduces the different types of label- and input noise considered in the experiments and concludes with a detailed description of the experimental setup.

4.2.1 Data generation

Our synthetic data is generated as a simple multidimensional classification problem, designed with variable degrees of spread and overlap between classes. This structure was chosen because it allows precise control over data characteristics, enabling the investigation of how class separation influences the detection of problematic samples.

Specifically, the synthetic dataset is defined as $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each input vector $\mathbf{x}_i \in X \subseteq \mathbb{R}^{100}$ is sampled uniformly from one of c class-specific isotropic Gaussian distributions $\{\mathcal{N}(\boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)})\}_{j=1}^c$. Here, $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{\Sigma}^{(j)}$ denote the *mean vector* and *covariance matrix* of the Gaussian distribution corresponding to class j , respectively.

Since we restrict the setting to isotropic Gaussians, the covariance matrix simplifies to $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, where σ is the standard deviation and $\mathbf{I}$ is the identity matrix. This constraint ensures that each class distribution has a “spherical” rather than “ellipsoidal” shape, allowing us to control class separability through the distance between mean vectors without introducing directional bias. For brevity, we denote each class distribution $\mathcal{N}(\boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)})$

as $\mathcal{N}^{(j)}$.

To control the difficulty of the classification task, all mean vector pairs $\boldsymbol{\mu}^{(j)}$ are separated by a fixed Euclidean distance d . When d falls within a certain range, overlap between the class distributions occurs, thereby increasing task complexity. The distance constraint is enforced by carefully selecting the class mean vectors. Let $\{\hat{\mathbf{e}}_i \mid i = 1, \dots, 100\}$ denote the standard orthonormal basis vectors in $\mathbb{R}^{100}$, where $\hat{\mathbf{e}}_i \cdot \hat{\mathbf{e}}_j = 1$ if $i = j$ and 0 otherwise, and $\|\hat{\mathbf{e}}_i\|_2 = 1$ for all i . The mean vector of class j is then defined as:

$$\boldsymbol{\mu}^{(j)} = \frac{d}{\sqrt{2}} \times \hat{\mathbf{e}}_j. \quad (4.1)$$

Finally, the dataset size is fixed to $n = 15,000$ which is randomly partitioned into two disjoint subsets: $D_{\text{train}} \cup D_{\text{test}}$, such that $|D_{\text{train}}| = 10,000$ and $|D_{\text{test}}| = 5,000$, unless stated otherwise.

4.2.2 Noise types

In the following sections, we modify D_{train} to gain intuition about the nature of problematic samples by introducing one of three types of label noise or one of three types of input noise. No noise is ever added to D_{test} . Throughout these sections, η denotes the *noise rate*, which governs the probability that any training sample is “noisy”. The specific noise types are defined below.

Label noise

- **Symmetric** This type of noise is generated by uniform label flipping, where the true label y is randomly changed to any other class $\tilde{y} \neq y$ with probability η .
- **Asymmetric** This type of noise is generated by pairwise label flipping. Specifically, the true label y is flipped to the subsequent class: $\tilde{y} = (y \bmod c) + 1$, where c is the number of classes.

-
- **Instance-dependent** This type of noise is generated following the approach of Xia *et al.* [60]. Instance-dependent label noise allows each sample to have its own probability of being mislabelled, rather than applying a uniform flip rate across the dataset. In this setting, the class to which a sample is flipped depends on its input features, meaning that samples with similar input features are more likely to be flipped towards specific alternative classes. The steps to generate instance-dependent label noise are outlined in Appendix A.1.

Both symmetric and asymmetric noise models follow the *class-dependent* assumption, where the noisy label distribution depends only on the true label: $P(\tilde{y} \mid \mathbf{x}, y) = P(\tilde{y} \mid y)$. In contrast, the instance-dependent noise model explicitly incorporates feature information, such that the noisy label distribution depends jointly on both $\mathbf{x}$ and y . Symmetric label noise is typically used as the simplest and most benign form of noise, modelling scenarios where annotation errors occur randomly and independently of the input, while asymmetric noise assumes that mislabelling patterns follow semantic structure, i.e., labels are more likely to flip to similar classes [56]. Instance-dependent label noise assumes that the probability of mislabelling depends on the characteristics of the input instance, which suggests that the same true class may be mislabelled differently depending on how ambiguous the sample looks [60].

Input Noise

Semantic- and covariate shifts, as defined in Section 3.5, are introduced in the form of input noise applied to a portion of D_{train} governed by η . Three variants are considered:

1. **Gaussian noise** With probability η , all input features $\mathbf{x}_i \in X \subseteq \mathbb{R}^{100}$ of a sample are replaced with values drawn from a single new Gaussian distribution $\tilde{\mathbf{x}}_i \sim \tilde{\mathcal{N}}$ that differs from the original set of c class-specific Gaussians. The mean vector $\tilde{\boldsymbol{\mu}} \in \mathbb{R}^{100}$ of $\tilde{\mathcal{N}}$ is sampled from a uniform distribution $\tilde{\boldsymbol{\mu}} \sim \mathcal{U}(-5, 5)$ and the standard deviation is fixed to $\tilde{\sigma} = 5$. All corrupted samples, regardless of class, are drawn from $\tilde{\mathcal{N}}$.

-
2. **Uniform noise** With probability η , all input features $\mathbf{x}_i \in X \subseteq \mathbb{R}^{100}$ of a sample are replaced with values drawn independently from a uniform distribution $\tilde{\mathbf{x}}_i \sim \tilde{\mathcal{U}}(-5, 5)$, where $\tilde{\mathbf{x}}_i \in \mathbb{R}^{100}$.
 3. **Edge noise** With probability η , the sample is shifted in the input space X by three standard deviations along the vector pointing from the class mean to the sample. This operation moves points that are close to the class mean towards the edge of the distribution, while points already near the boundary are pushed beyond it. Since approximately 99.7% of the probability mass of a Gaussian distribution lies within ± 3 standard deviations, this perturbation places affected samples near or beyond the outer boundary.

Gaussian- and uniform input noise correspond to *semantic shifts*, as they replace original features with values outside the label space. In contrast, edge noise represents a *covariate shift*, as it modifies the input distribution while preserving the underlying label space.

Noise visualization

To gain a clearer understanding of the various types of label- and input noise, we visualize each noise type using a two-dimensional (2D) toy dataset, as shown in Figure 4.1. We set the number of classes to $c = 3$ to highlight the differences among the label noise types.

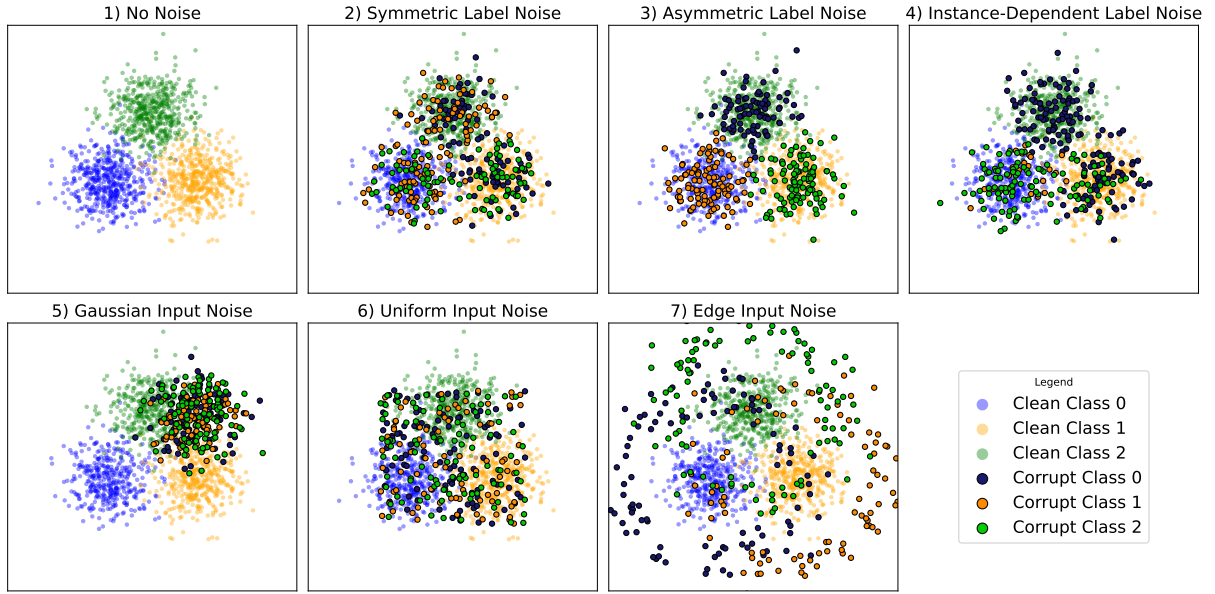


Figure 4.1: An illustrative visualization of different noise types for a 2D three-class classification problem. Each subplot shows clean samples (translucent) and corrupted samples (opaque with black edges). The noise rate for all plots (except the clean case) is $\eta = 0.2$. The legend (bottom right) applies to all subplots.

To isolate the effects of noise, the standard deviations of all class distributions are kept identical. This is done purely for illustration; in the following sections, each class may have a different standard deviation. The class means are equally spaced, forming the vertices of an equilateral triangle in 2D space.

In the case of symmetric label noise, each label of a corrupted sample has an equal probability of being flipped to either of the other two classes, resulting in a uniform mixture of label noise. For asymmetric label noise, the flipping is structured: corrupted class 0 samples are mislabelled as class 2, corrupted class 1 samples as class 2, and corrupted class 2 samples as class 0.

In real-world scenarios, instance-dependent label noise arises when certain samples are intrinsically more confusing than others. For example, an image of a *dog* taken from an odd angle might be mislabelled as a *cat* because it looks more like a *cat* than a typical *dog*. In such cases, the likelihood of mislabelling depends on the specific features of each

instance rather than the overall class.

In our synthetic dataset in Figure 4.1, we simulate this behaviour using randomly generated feature-dependent noise. Under this instance-dependent label noise, we see that most affected class 0 samples are corrupted as class 2 with some corrupted as class 1, most affected class 1 samples are corrupted into class 0 and class 2 in roughly equal parts, and most affected class 2 samples are corrupted as class 0. These proportions are specific to the example in this illustration since they depend on the data distribution, the randomly generated projection matrices $\mathbf{W}^{(j)}$, and the randomly generated per-sample flip rates (as outlined in Appendix A.1). We expect a variety of corruption behaviours across experiments.

For Gaussian input noise, each corrupted sample is drawn from an isotropic (circular) Gaussian distribution in the feature space. In this visualization, the mean is set to $\tilde{\boldsymbol{\mu}} = (2, 2)$ and the standard deviation to $\tilde{\sigma} = 1.5$. In the following sections for the higher-dimensional synthetic datasets, the mean is sampled uniformly $\tilde{\boldsymbol{\mu}} \sim \mathcal{U}(-5, 5)$ and the standard deviation is fixed at $\tilde{\sigma} = 5$ (as mentioned in Section 4.2.2). Uniform input noise produces a square-shaped distribution of noisy samples, reflecting the uniform sampling across each dimension. Finally, edge input noise generates samples concentrated near the outer circular boundaries, as points are displaced away from their respective class means.

4.2.3 Experimental setup

This section outlines the experimental setup, applicable to the binary classification experiments in Section 4.3 and the multiclass classification experiments in Section 4.4.

Detectors

To evaluate whether the noise types introduced in Section 4.2.2 are indeed problematic, two *baseline detectors* are considered and compared with where appropriate throughout the remainder of this study:

-
- **Oracle** This detector has access to ground-truth information about which samples are noisy. It ranks all noisy samples first (in random order), followed by all clean samples (also in random order). Consequently, noisy samples are always removed before clean ones, illustrating the outcome if a detector was able to crudely rank all noisy samples first.
 - **Random** This detector ranks training samples entirely at random. It serves as a baseline, providing a reference point for performance: any detector should ideally outperform this purely random ranking.

It should be noted that these are not detectors in the formal sense as defined in Section 3.3, as they do not necessarily constitute a mapping to scores. Rather, they are intended to illustrate detectors for scoring training samples: either using a naive scoring approach, which effectively orders samples at random, or a more informed approach that leverages access to the ground-truth labels.

In this chapter, the ψ -curves corresponding to the *oracle* detectors are shown as solid lines, while the ψ -curves for the *random* detectors are shown as dashed lines. By comparing the resulting ψ -curves, intuition can be gained about both the noisy samples and the evaluation framework. Specifically, if the *oracle* curves exhibit an initial increase, this indicates that the noisy samples are problematic. Conversely, if the *random* curves show a gradual decrease, this suggests that generalization performance is positively correlated with train set size in the synthetic setting. It is important to note that noise is applied exclusively to the train set, while the test set remains without noise (clean).

Hyperparameters

For the experiments in this chapter, a two-layer MLP with 512 nodes per hidden layer is employed. We use a two-layer MLP to keep the model architecture simple and well-suited to the synthetic datasets, which involve relatively straightforward classification tasks. A width of 512 nodes per hidden layer ensures adequate capacity for the network to fit all training samples, including noisy ones, while keeping the training time computationally

manageable. Training is performed using the Adam optimizer [100] with a batch size of 100 and a learning rate of 0.01. Batch normalization [101] is applied after each hidden layer, followed by a ReLU activation function.

These commonly used hyperparameters were chosen to provide stable and efficient optimization without extensive tuning. Since the goal of this work is not to achieve optimal model performance but to analyse model behaviour under controlled conditions, no hyperparameter search was conducted. All models are trained with cross-entropy loss for 200 epochs or until interpolation, what ever comes first.

Lastly, the training inputs $\mathbf{x}$ were normalized by subtracting the mean and scaling to unit variance, using the mean and variance computed from the train set.

Evaluation framework

In this chapter, all data pruning evaluation results are averaged across five random seeds. The mean performance across these seeds is shown as the ψ -curve, with the corresponding standard deviation represented by shaded regions in the figures. The resolution hyperparameter is set to $r = 100$ with $\alpha = 1$, meaning that subsets of 100 samples are removed at each step until the entire train set has been pruned. Since the train set has 10,000 samples, each ψ -curve contains $m = 101$ components, as calculated in Equation 3.6.

4.3 Binary classification

We begin the evaluation with a simple binary classification problem, corresponding exactly to the synthetic data described in Section 4.2.1 with $c = 2$, to address three key questions:

1. Do we observe problematic samples for a synthetic binary classification problem?
2. Can the proposed evaluation framework reliably indicate the presence of such samples?

-
3. If problematic samples exist, what dataset characteristics influence their impact ($\Delta\psi$) according to Equation 3.7?

4.3.1 Label noise

To investigate these questions, label noise at a noise rate of $\eta = 0.3$ is introduced into the train set. Note that in the binary classification case, the three label noise types defined earlier are equivalent. Regardless of whether the flip probabilities depend on class structure or instance features, there is only one alternative label available, making all three noise types equivalent in effect. The results are presented in Figure 4.2, where each colour corresponds to a different dataset. Specifically, the blue and orange curves represent datasets that are identical in all characteristics except for the distance between the two class mean vectors, which are separated by distances of 7 and 3, respectively.

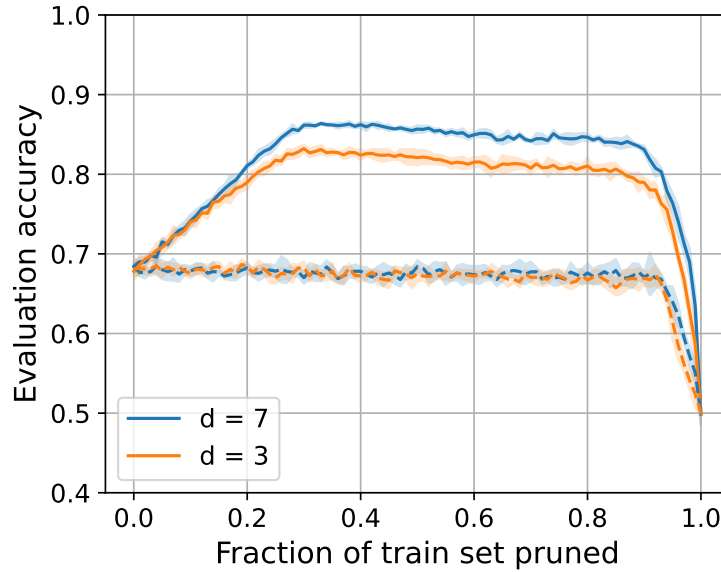


Figure 4.2: Evaluation framework for binary classification with label noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 7$ (blue) and $d = 3$ (orange). The standard deviations for class 1 and 2 are fixed at $\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$, respectively.

Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

Figure 4.2 demonstrates that for this binary classification problem, label noise is indeed problematic. The *oracle* curves (solid lines) exhibit a clear increase in evaluation accuracy as the noisy samples are removed. This increase reaches a peak near the point where all noisy samples are removed, after which there is a gradual decline up to the most extreme pruning, where performance quickly drops to random guessing. Furthermore, the evaluation accuracy reaches a higher overall value for the dataset with the largest distance d between the class means, reflecting the fact that the classification task becomes easier when the classes are more distinct.

A closer inspection, however, reveals an intriguing observation: the ψ -curves for the two datasets start at the same point. This suggests that models trained on the full train set with 30% label noise perform equally well on their respective clean test sets, regardless of whether the class means are separated by 7 or 3. Only after noisy samples are progressively removed do the ψ -curves diverge, with the dataset having a separation of 7 ultimately achieving higher performance. This raises the question: why do the two datasets initially yield the same performance? A plausible explanation is that the relatively high noise rate (30%) overwhelms the underlying distributional differences, effectively masking the advantage of greater class separation and rendering the two tasks equally challenging when training on the full dataset.

From the *random* ψ -curves, it is evident that removing samples at random from this noisy train set does not significantly affect evaluation accuracy until a *critical threshold* is reached: when approximately 90% of the train set has been pruned, leaving around 1,000 samples. Beyond this point, performance declines rapidly. This result indicates that when the train set contains 30% label noise, increasing the dataset size beyond approximately 1,000 samples (while maintaining a noise rate of $\eta = 0.3$) does not yield further improvements in performance. This finding illustrates that, under conditions of substantial label noise, a larger dataset does not necessarily enhance model performance, as 1,000 samples are sufficient for the model to capture a good underlying relationship for a train set with 30% label noise. This suggests that the model effectively hits a *noise-limited ceiling* rather than a data-limited one.

Interestingly, the *oracle* ψ -curves exhibit a similar critical threshold, suggesting that approximately 1,000 samples are sufficient for performing relatively well, regardless of whether the samples are clean or noisy. However, models trained on clean samples achieve consistently higher evaluation accuracy (*oracle* critical thresholds lie above the *random* critical thresholds). Unlike the *random* ψ -curves, the *oracle* ψ -curves demonstrate that adding additional clean data continues to marginally improve evaluation accuracy. This is reflected in the gradual decline of the oracle ψ -curves from the point where all label noise has been removed up to the critical threshold.

Figure 4.3 presents the results of a complementary experiment, in which the class distribution variances are varied while the distance between the class means is fixed.

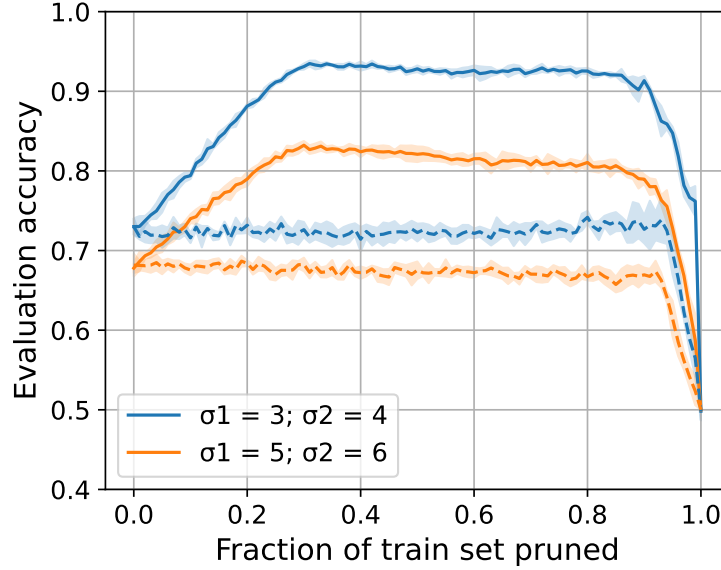


Figure 4.3: Evaluation framework for binary classification with label noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 3$. The standard deviations are varied: ($\sigma^{(1)} = 3$ and $\sigma^{(2)} = 4$) blue and ($\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$) orange. Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

From Figure 4.3, it is evident that smaller standard deviations (represented by the blue curve) result in higher overall evaluation accuracy. Across the entire range of pruning, both the blue *oracle* and blue *random* curves remain above their corresponding orange

curves. This outcome is expected, as smaller standard deviations reduce class overlap, thereby simplifying the classification task. Interestingly, both Figures 4.2 and 4.3 reveal that problematic samples exert a greater negative impact in the easier tasks. Specifically, removing the 30% label noise produces a larger increase in evaluation accuracy for tasks with less overlap between classes.

These findings inform us that when a fixed proportional subset of problematic samples is present in the train set, their impact (degree to which they are problematic) is influenced by dataset characteristics. As shown in Figures 4.2 and 4.3, problematic samples tend to degrade model performance more severely in easier tasks, whereas their impact becomes less pronounced in more complex tasks that exhibit greater inherent class overlap.

Next, the effect of varying the noise rate while keeping all other dataset characteristics constant is examined.

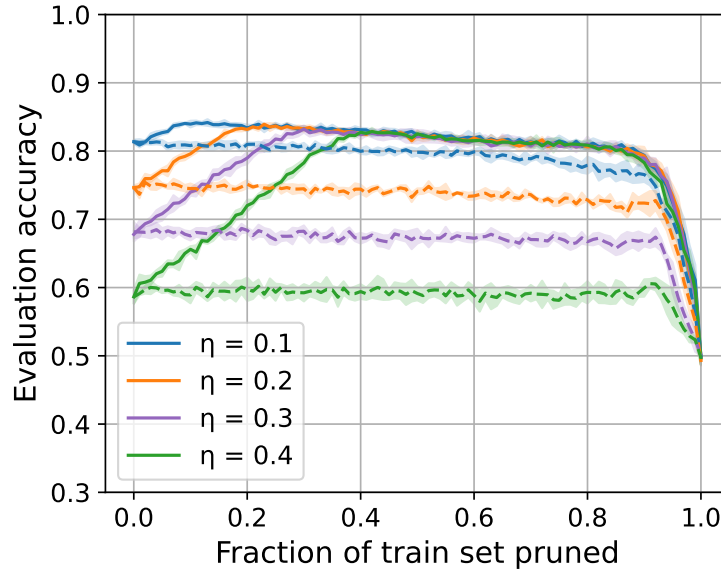


Figure 4.4: Evaluation framework for binary classification with label noise introduced into the train set, while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 3$ and standard deviations $\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$. The noise rate η is varied from 0.1 to 0.4 in increments of 0.1. Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

As expected, Figure 4.4 demonstrates that increasing the noise rate substantially amplifies the impact of problematic samples and directly corresponds to a larger proportion of problematic samples. Furthermore, at lower noise rates the *random* ψ -curves (dashed lines) exhibit a steeper decline in evaluation accuracy. This observation can be attributed to one or both of the following factors: 1) when the noise rate is low, additional training samples contribute more effectively to model generalization, whereas their benefit diminishes as the noise rate increases; or 2) at lower noise levels, a larger fraction of clean samples may be removed at random in the early stages of pruning, leading to the sharper performance drop.

In summary, the main findings from the binary classification experiments on the synthetic dataset with label noise are as follows:

- Label noise is problematic, as it consistently shows a detrimental effect on model performance.
- The classification task becomes easier when there is less inherent overlap between the two class distributions. This reduction in overlap can be achieved either by increasing the distance d between the mean vectors of the class distributions or by decreasing their standard deviations σ .
- A relatively high noise rate (e.g., 30%) can obscure underlying distributional differences between datasets, effectively masking the benefits of greater class separation.
- Regardless of whether label noise is present in the train set, there exists a critical threshold beyond which increasing the number of training samples does not substantially improve evaluation accuracy, provided the noise rate remains constant. In these experiments, this threshold occurred when approximately 90% of the train set was pruned.
- Adding more clean samples to a dataset with no label noise can yield marginal improvements in evaluation accuracy as the classifier’s performance approaches the Bayes error rate.

-
- When a fixed proportional subset of problematic samples is present in the train set, their impact depends on dataset characteristics such as the inherent class overlap. Problematic samples tend to degrade model performance more severely in easier tasks with less inherent class overlap.
 - At lower noise rates, additional training samples contribute more effectively to generalization. However, random removal of clean samples during early stages of pruning can also lead to sharper drops in performance.

4.3.2 Input noise

In this section, the evaluation framework is applied to assess whether input noise can also be problematic in a binary classification setting. Figure 4.5 presents results for the three types of input noise introduced in Section 4.2.2. Unlike the three types of label noise, which are equivalent in the binary case, the three input noise types remain distinct.

Figure 4.5 shows that edge noise (a covariate shift) is problematic, as evidenced by the clear increase in evaluation accuracy along the *oracle* ψ -curve. However, the impact of the problematic samples is lower than that observed for label noise under equivalent class separations, standard deviations, and noise rate (comparing the orange *oracle* ψ -curve in Figure 4.2 with the purple *oracle* ψ -curve in Figure 4.5). More specifically, for $\sigma^{(1)} = 5$, $\sigma^{(2)} = 6$ and $d = 3$, the impact $\Delta\psi = 0.08$ for edge noise and $\Delta\psi = 0.15$ for label noise, calculated according to Equation 3.7.

In contrast to edge noise, Gaussian- and uniform noise (semantic shifts) are not problematic in these experiments, since their *oracle* ψ -curves show no clear increase in evaluation accuracy. A visual inspection of Figure 4.1 suggests that all three types of input noise should be detrimental to model performance. However, this expectation is not supported by the results presented in Figure 4.5. We speculate that in higher-dimensional spaces (e.g., $\mathbb{R}^{100}$), the model can identify subspaces within the input domain where clean and noisy samples are more easily separable, potentially through mechanisms analogous to the “kernel trick” [102]. In contrast, for edge noise, the perturbations are intentionally ap-

plied along directions that likely correspond to “on-manifold” regions of the input space, in this case, from the class centre toward boundary points. Such perturbations may be inherently more difficult for the model to distinguish from legitimate data, thereby making edge noise more problematic.

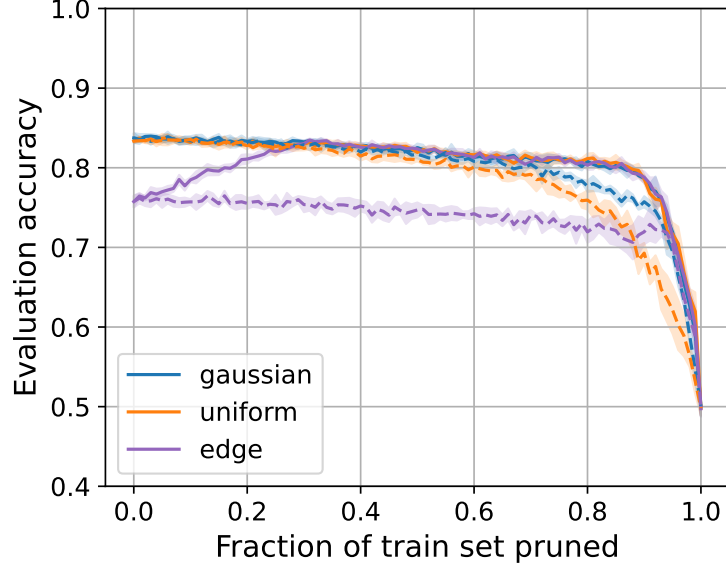


Figure 4.5: Evaluation framework for binary classification with three types of input noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 3$ and standard deviations $\sigma^{(1)} = 5$ and $\sigma^{(2)} = 6$. Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

4.4 Multiclass classification

In this section, the analysis is extended to examine whether the observations from the binary classification setting generalize to multiclass classification. For all experiments, the number of classes is fixed at $c = 10$, while still using the experimental setup as defined in Section 4.2.3.

4.4.1 Label noise

Figure 4.6 compares the effect of the three types of label noise, defined in Section 4.2.2, in order to evaluate their impact and to determine whether their impact is comparable to that observed in the binary classification case.

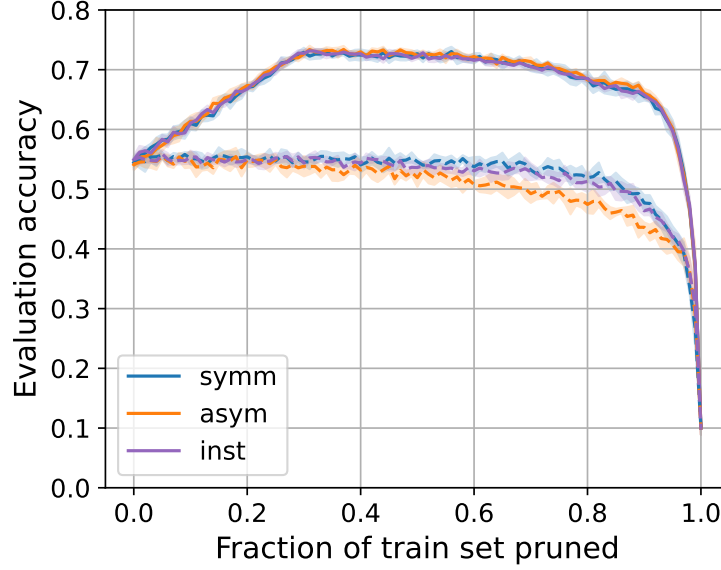


Figure 4.6: Evaluation framework for $c = 10$ multiclass classification with three types of label noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 7$ and standard deviations $\sigma^{(j)} \sim \mathcal{U}(1, 5) \in \mathbb{R}$. Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

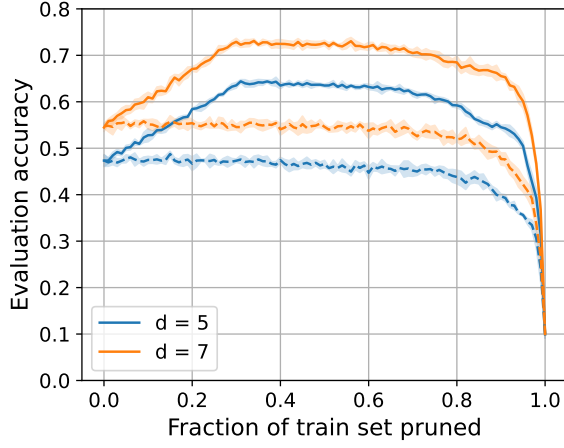
The results indicate that all three types of label noise are indeed problematic, as evidenced by the clear increase in evaluation accuracy along the *oracle* ψ -curves. Using Equation 3.7, the impact is estimated at $\Delta\psi = 0.19$, for all label noise types. This result is very similar to the binary classification case, where $\Delta\psi = 0.18$ for a class separation of $d = 7$ (calculated by considering the blue curve in Figure 4.2). However, it should be noted that the slightly higher value of $\Delta\psi$ in the multiclass setting cannot be attributed solely to the increased number of classes, since the standard deviations in the two settings are not directly comparable. It is hypothesized that, when practical problematic sample

detectors are applied to this problem, performance can vary across the three noise types. This hypothesis is tested and confirmed in Chapter 6.

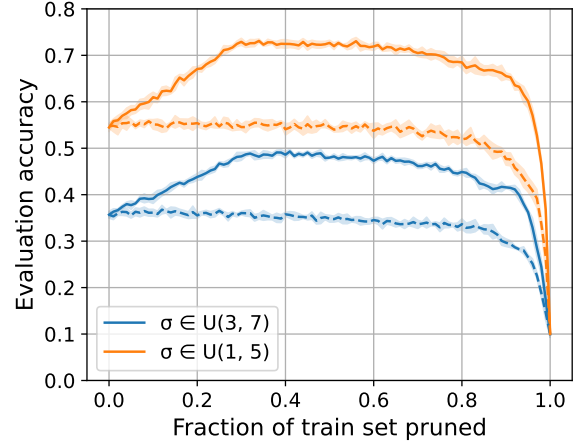
In Figure 4.7, the experiments from the binary classification setting are repeated for multiclass classification to assess whether the previous findings hold for $c > 2$ classes. Figures 4.7a and 4.7b show that increasing the distance between class means or decreasing the standard deviations leads to higher overall evaluation accuracy (orange curves), consistent with the observations from Section 4.3. Interestingly, however, the impact of the problematic samples is less affected by the dataset characteristics than in the binary classification case; specifically, $\Delta\psi$ remains nearly constant across the blue and orange curves in both figures.

If the variation in the impact of problematic samples as observed for binary classification, is indeed driven by the noise rate overwhelming the task as discussed in Section 4.3.1, then the results in Figures 4.7a and 4.7b suggest that the same noise rate does not have an overwhelming effect in a multiclass setting. This observation is consistent with intuition: in the multiclass case, noisy samples are distributed across ten classes rather than two, resulting in less concentrated noise per class and thereby reducing its impact on task difficulty. This is also substantiated by the observation that in Figure 4.2 the ψ_1 performance (full train set performance) is the same regardless of d , while for Figure 4.7a, the ψ_1 performance differs for different d .

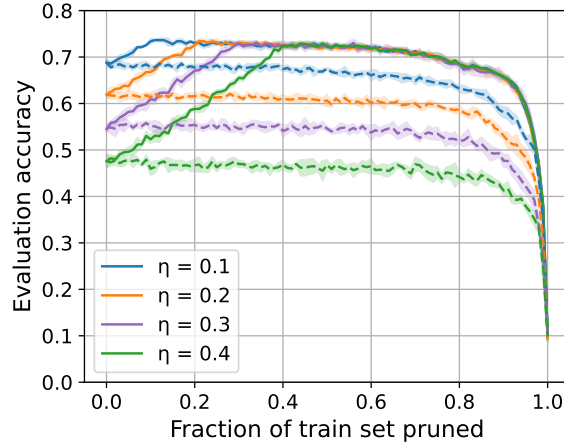
Finally, Figure 4.7c confirms that varying the noise rate in the multiclass setting produces trends consistent with those observed in the binary classification experiments. That is, increasing the noise rate substantially amplifies the impact of problematic samples and directly corresponds to a larger proportion of problematic samples.



(a) Varying d



(b) Varying $\sigma^{(j)}$



(c) Varying η

Figure 4.7: Evaluation framework for $c = 10$ multiclass classification with symmetric label noise introduced into the train set, while the test set remains clean. For (a), we keep the class-related standard deviation identical, while varying the distance d as shown in the legend. For (b), we keep the distance constant and vary the ranges that that standard deviations $\sigma^{(j)}$ are sampled from, as shown in the legend. Lastly, for (c), the distance and standard deviations are kept constant while the noise rate η is varied as shown in the legend. Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

4.4.2 Input noise

In Figure 4.8, the effect of input noise is examined for the multiclass classification task.

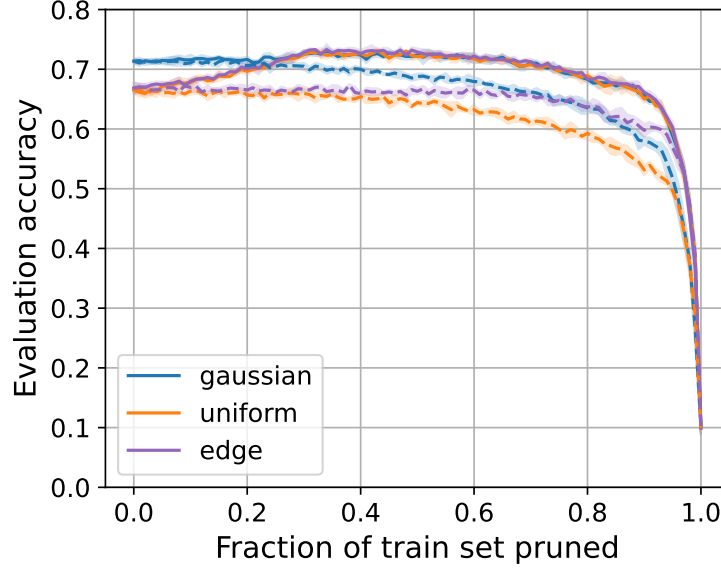


Figure 4.8: Evaluation framework for $c = 10$ multiclass classification with three types of input noise introduced into the train set ($\eta = 0.3$), while the test set remains clean. Each class is sampled from a Gaussian distribution, with class means separated by $d = 7$ and standard deviations $\sigma^{(j)} \sim \mathcal{U}(1, 5) \in \mathbb{R}$. Solid lines correspond to *oracle* detectors, while dashed lines correspond to *random* detectors.

The results in Figure 4.8 indicate that both edge noise (covariate shift) and uniform noise (semantic shift) are problematic, as evidenced by clear improvements in evaluation accuracy observed in the *oracle* ψ -curves when these samples are removed. In contrast, Gaussian input noise does not appear to substantially degrade performance, mirroring the behaviour in the binary classification setting.

This finding highlights an important aspect of noisy samples and the effect they have on model performance: their proximity to the clean samples likely play an important role in the degree to which they are problematic (impact) for the model. This idea has been presented previously [103]. We support this with a visual probing of the distances between clean and input corrupted samples in Fig. 4.9.

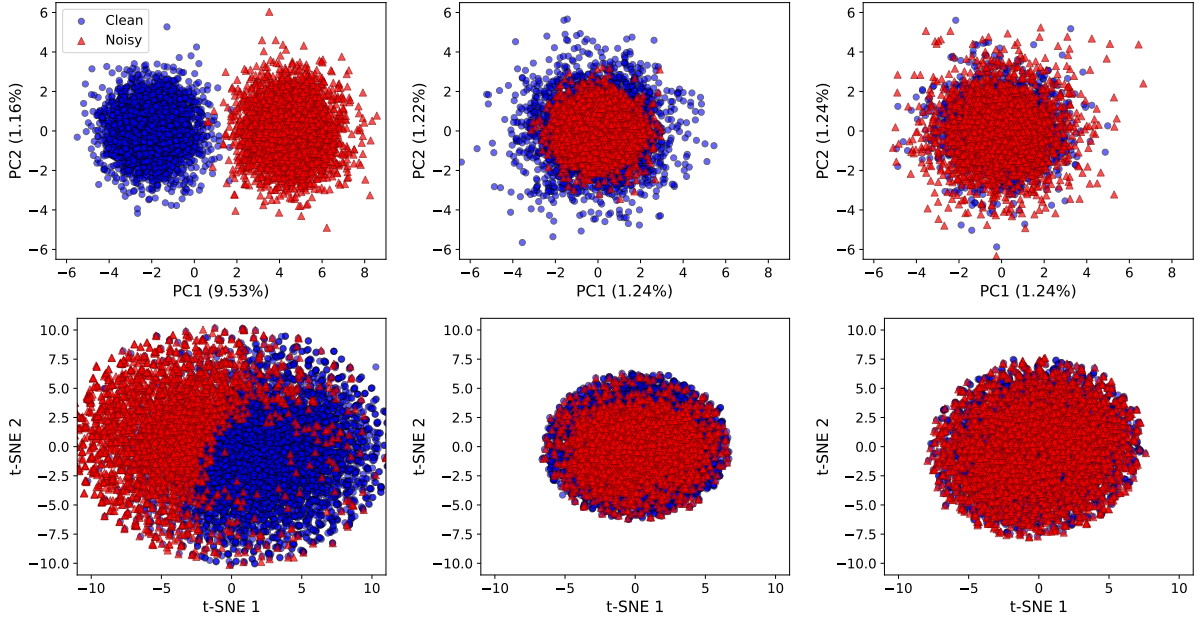


Figure 4.9: The columns of this plot correspond to the three types of input noise: Gaussian input noise (left), uniform input noise (middle) and edge input noise (right). The two rows correspond to different visualization techniques: PCA (top) and t-SNE (bottom). The legend in the first plot applies to all plots.

For edge noise and uniform noise (middle and right columns), the corrupted samples are situated relatively close to the $c = 10$ ID Gaussian class distributions. The noisy and clean distributions cannot be separated in two dimensions; evident by the small explained variance of the first two principal components (as shown on the axis). By contrast, samples affected by Gaussian noise (left column) tend to lie farther from the clean data, as they are drawn from a distribution with a mean displaced well outside the ID regions. The model shows no performance degradation when memorizing the Gaussian input noise because: 1) the evaluation set contains no samples similar to these noisy samples, 2) the regions of feature space occupied by input noise are irrelevant for classifying the evaluation samples, or 3) both factors contribute.

In summary, the results from the multiclass classification experiments demonstrate that several findings from the binary classification setting generalize to scenarios involving multiple classes. Specifically, problematic samples arise in the form of both label noise and input noise, and the degree of separation between class distributions influences the

model’s overall evaluation accuracy in both binary- and multiclass classification settings. However, in multiclass classification, the impact of problematic samples appears to be less sensitive to dataset characteristics than in the binary case. Furthermore, uniform input noise, which was not detrimental in the binary classification task, becomes problematic in the multiclass setting.

4.5 Discussion

The primary objectives of this chapter were twofold: 1) to determine whether the OOD samples defined in Section 3.5 can indeed be problematic, and 2) to evaluate whether the data pruning evaluation framework described in Section 3.4 can systematically detect their presence. The results demonstrated that all three types of OOD samples can negatively affect model performance in a synthetically generated dataset, and that the proposed evaluation framework effectively identifies their presence and estimates their impact. Furthermore, the analyses provided valuable insights into the nature of these samples across both binary- and multiclass classification tasks. The key findings are summarized below:

1. Our evaluation framework, introduced in Chapter 3, effectively identifies problematic samples and estimates their impact. Accordingly, we continue to use this framework in subsequent chapters to compare the performance of different problematic-sample detectors.
2. Label noise is generally more consistently problematic than input noise, and its impact is typically more severe for a given dataset and noise rate. For this reason, the majority of our experiments in the following chapters focus on label noise.
3. In these experiments, all three types of label noise exhibit comparable levels of difficulty. However, the *oracle* and *random* baseline curves do not clearly indicate how the specific type of label noise affects the performance of actual problematic sample detectors. We therefore investigate this further in Chapter 6.

-
4. The impact of input noise varies substantially depending on the noise type and its interaction with dataset characteristics, such as the distance to clean samples. Consequently, the analysis in Chapter 6, evaluating the performance of problematic sample detectors under input noise, uses uniform input noise for the multiclass synthetic dataset. This choice is based on the results from this chapter, which demonstrate that uniform noise is indeed problematic.

Building on these findings, the subsequent chapters will extend this investigation to image classification datasets and dedicated problematic sample detectors. Since both the *oracle* and *random* baselines proved to be effective reference points in this study, they will continue to be employed as baselines for evaluating the performance of actual problematic sample detectors.

Chapter 5

On learning-centric detectors

We implement learning-centric methods to determine whether they can be used as problematic sample detectors.

5.1 Overview

The findings thus far have demonstrated that problematic samples, as defined in this work, do exist and that the proposed evaluation framework is capable of detecting them. However, the analysis up to this point has relied solely on two baseline detectors applied to synthetic datasets.

The objective of this chapter is to investigate whether established data pruning methods from the literature can also be employed to identify problematic samples. In particular, we implement two widely studied approaches: *sample forgetting* [26] and *error L_2 -norm (EL2N)* [29], and conduct an in-depth evaluation of their strengths and limitations. To extend the scope beyond synthetic data, we further apply these methods to image classification tasks and examine whether the trends observed in the synthetic setting persist.

The chapter is structured as follows. Section 5.2 discusses the rationale behind selecting these two methods and highlights the research gap addressed by this chapter. Section 5.3 presents the *sample forgetting* and *EL2N* detectors in detail. The experimental setup is outlined in Section 5.4, and the results are reported in Section 5.5. Finally, Section 5.6 provides a discussion and summary of the key findings.

5.2 Data pruning methods

Foundation and motivation As discussed in Chapter 2.2, a wide range of scoring methods have been developed for data valuation. This raises the question of whether such methods can also be adapted to serve as problematic sample detectors. Among these, the approaches introduced by Toneva *et al.* [26] and Paul *et al.* [29] are particularly influential, having been widely adopted and frequently used as benchmarks in the literature [24], [31], [104]. Moreover, Sorscher *et al.* [24] demonstrated that the scoring distributions produced by these methods are consistent with those of other related approaches. Taken together, these observations motivated us to examine more closely the *sample forgetting* method proposed by Toneva *et al.* [26] and the *EL2N* method introduced by Paul *et al.* [29].

Positioning within prior work The work most closely related to this chapter is that of Sorscher *et al.* [24], who conducted a comprehensive comparison of various data pruning methods and the extent to which they can break power scaling laws. Our investigation provides a targeted extension of this comparison by addressing several gaps that remain underexplored in the literature:

- To the best of our knowledge, no prior pruning studies have pruned the train set nearly to full removal (i.e., $\alpha = 1$ in Equation 3.6) while systematically comparing against a baseline. Most works stop well short of this point [25], [26], [29], [30], [34]. We note that pruning close to full removal somewhat undermines the validity of the pruning experiment: once the train set becomes too small to capture the underlying data distribution, meaningful comparisons between detectors become tenuous. Our

focus here is not on achieving strong performance in the high-pruning regime, but on understanding when the pruning framework itself ceases to be informative. To account for this, we closely monitor the point at which the *random* baseline deteriorates completely, indicating that the remaining data are insufficient for reliable evaluation.

- Only a limited number of studies have explicitly examined data pruning under the presence of label noise—a type of problematic sample. Notable exceptions include the work of Yoon *et al.* [30], as well as the label error detection approaches by Pleiss *et al.* [61] and Northcutt *et al.* [56].
- Prior works, including those of Paul *et al.* [29], Ki *et al.* [31], and Toneva *et al.* [26], consistently note that higher scores are often associated with label noise. Nevertheless, these studies argue that samples with higher scores contribute more to model performance, thereby pruning in ascending order of score.

This chapter builds directly on these insights by addressing the limitations outlined above.

Learning-centric perspective Finally, since both the *sample forgetting* and *EL2N* methods assign scores to samples during the training process (hence the chapter’s title, *On learning-centric detectors*), they also provide a lens through which to study how problematic samples affect model behaviour. In particular, these methods allow us to analyse how such samples contribute to forgetting dynamics and influence gradient updates, thereby offering a deeper understanding of their role in shaping model behaviour throughout training.

5.3 Detectors

In this section we outline the *sample forgetting* and *EL2N* methods.

5.3.1 Sample forgetting

Toneva *et al.* [26] introduce a scoring method grounded in the forgetting behaviour of neural networks. The method draws inspiration from the concept of *catastrophic forgetting* [105], whereby a model loses previously learned information when trained on new tasks. Extending this notion, Toneva *et al.* consider each iteration of SGD as a *mini-task*. A training sample is said to be *forgotten* at iteration t if it was correctly classified at iteration $t - 1$ but misclassified at t , where iterations correspond to consecutive mini-batches containing that sample.

For computational tractability, Toneva *et al.* approximate iterations using epochs, treating them as a lower bound¹. The number of forgetting events for each sample is accumulated over the full training process to yield a cumulative *forgetting score* s_i . Samples that are learned once and never forgotten are termed *unforgettable* and assigned a score of zero, while samples that are never correctly classified receive an *infinite* forgetting score.

Formally, a training sample $(\mathbf{x}_i, y_i) \in D_{\text{train}}$, observed over e iterations during the learning process, is assigned a *forgetting score* defined as:

$$s_i = \mathbb{E} \left[\sum_{t=1}^e \mathbf{1}[f_t(\mathbf{x}_i) \neq y_i \ \& \ f_{t-1}(\mathbf{x}_i) = y_i] \right], \quad (5.1)$$

where $f_t(\mathbf{x}_i)$ denotes the model’s predicted class for input $\mathbf{x}_i$ at iteration t . The function $\mathbf{1}[\cdot]$ is an indicator function that returns 1 if the condition inside the brackets is satisfied and 0 otherwise. The expected value is estimated by averaging across multiple model initializations.

5.3.2 Error L2-norm

Paul *et al.* [29] motivate their approach by analysing the effect of individual training samples on the learning process. They begin with a fundamental question: how much

¹In practice, when the mini-batch size is not a factor of the train set size, the final few samples that do not fit into a complete mini-batch are discarded. Consequently, not all samples are encountered by the model in every epoch.

does removing a single training sample affect the model’s ability to learn from other samples? Through a single-gradient-step analysis, they show that a sample’s contribution to reducing the loss on arbitrary other samples, is bounded by its *loss gradient norm*. Specifically, the change in loss reduction caused by removing sample $(\mathbf{x}_i, y_i)$ from a mini-batch at iteration t , is bounded by:

$$c\|\nabla_{\phi_t}\ell(\mathbf{f}_t(\mathbf{x}_i), \mathbf{y}_i)\|, \quad (5.2)$$

where c is a constant independent of the specific sample, $\nabla_{\phi_t}\ell(\cdot)$ is the gradient of the loss with respect to the set of model parameters ϕ_t at iteration t of SGD². Note that $\ell(\mathbf{f}_t(\mathbf{x}_i), \mathbf{y}_i)$ is the cross-entropy loss between the model prediction $\mathbf{f}_t(\mathbf{x}_i)$ at iteration t as a probability vector and the one-hot-encoded label vector $\mathbf{y}_i$. Intuitively, the bound in Equation 5.2 suggests that samples with larger gradient norms have greater potential effect on the training dynamics.

Building on this insight, Paul *et al.* propose the *Gradient Normed (GraNd) score*, defined as the expected gradient norm across multiple model initializations:

$$s_i = \mathbb{E}\|\nabla_{\phi_t}\ell(\mathbf{f}_t(\mathbf{x}_i), \mathbf{y}_i)\|_2. \quad (5.3)$$

However, they observe empirically that early in training, and when the logit gradients are approximately orthogonal across classes, the gradient norm can be well-approximated by simply measuring the prediction error. This leads to the simpler and more effective *EL2N* metric, that capture sample importance using only the model’s prediction outputs.

The *EL2N* score is simply the Euclidean norm of the error vector of the model’s prediction with respect to the current sample at a relatively early stage in training. More specifically, for a training sample $(\mathbf{x}_i, y_i) \in D_{\text{train}}$, the *EL2N* score is defined as:

$$s_i = \mathbb{E}\|\mathbf{f}_t(\mathbf{x}_i) - \mathbf{y}_i\|_2. \quad (5.4)$$

The expected value is estimated by averaging across multiple model initializations.

²See Paul *et al.* Lemma 2.2 for proof of the bound in 5.2.

5.4 Experimental setup

Datasets We evaluate the *sample forgetting* and *EL2N* detectors on three multiclass classification datasets: the synthetic dataset described in Section 4.2.1, and two widely used benchmarks, MNIST [17] and CIFAR-10 [18]. For the synthetic dataset, we retain the number of classes $c = 10$, the distance constraint of $d = 7$ and sample $\sigma^{(j)} \sim \mathcal{U}(1, 5)$. The sizes of the train- and test sets for all datasets are reported in Table 5.1. For MNIST and CIFAR-10, the train sets are drawn randomly from the full datasets, while the test sets correspond to the predefined evaluation sets. In experiments involving label noise, corruption is applied exclusively to the train sets. Specifically, we employ symmetric label noise, as outlined in Section 4.2.2. The exploration of other noise types is deferred to Chapter 6.

Table 5.1: Dataset sizes and model configurations for the three datasets.

Dataset	$ D_{\text{train}} $	$ D_{\text{test}} $	r	Model	Batch size	Learning rate
Synthetic	10,000	5,000	100	2-layer MLP	100	0.01
MNIST	55,000	10,000	550	2-layer MLP	64	0.01
CIFAR-10	45,000	10,000	2,250	VGG-16	64	0.001

Models For the synthetic and MNIST datasets, we employ MLP models, while for CIFAR-10 we use VGG-16 models. The MLP architecture consists of two hidden layers, each with 512 nodes. For VGG-16, the final three layers consist of fully connected layers with 4,096, 4,096, and 1,000 nodes, respectively, following the original design by Simonyan *et al.* [106]. We use a two-layer MLP to keep the model architecture consistent with the setup in Chapter 4. Additionally, we selected VGG-16 because it serves as a benchmark CNN and is better suited for larger RGB images, such as those in the CIFAR-10 dataset. All models are trained using the Adam optimizer [100], with batch size and learning rate specified in Table 5.1. Batch normalization layers [101] are applied after each hidden layer and before a ReLU activation function. The inputs $\mathbf{x}$ for the image datasets were normalized with min-max normalization, while the inputs for the synthetic dataset were normalized as discussed in Section 4.2.3 (subtracting the mean of each feature across the train set and scaling to unit variance).

Scoring We compute sample scores $\mathbf{s} = (s_1, s_2, \dots, s_{|D_{\text{train}}|})^\top$ using the two detectors defined in Equations 5.1 and 5.4, averaging results over ten randomly initialized models. To measure forgetting events (Equation 5.1), each model is trained for 200 epochs. For the *EL2N* method (Equation 5.4), error vectors are recorded after 20 epochs of training. These design choices are aligned with the experimental setups of Toneva *et al.* [26] and Paul *et al.* [29].

Pruning After assigning a score to each training sample, we apply Algorithm 1 to generate a ψ -curve for every detector Λ . At each point ψ_i along the curve, the model performance is evaluated over three random initializations. These evaluations were performed using the same architecture and learning process used for scoring. The evaluation models are trained for 200 epochs or until interpolation, what ever comes first. The resolution parameter r is set according to Table 5.1, and we keep $\alpha = 1$ to prune across the entire train set. In addition to the *sample forgetting* and *EL2N* detectors, we also conduct pruning experiments using the *random* and *oracle* (where applicable) baselines, as described in Section 4.2.3.

5.5 Results

In this section, we present the results after evaluating the two detectors on unaltered (no added noise) train sets in Section 5.5.1 and on train sets with symmetric label noise in Section 5.5.2. We conclude with a deeper analysis of the identified problematic samples in Section 5.5.3.

5.5.1 Redundant samples

We begin by applying the *sample forgetting* and *EL2N* detectors to the three datasets and corresponding models to verify whether our results align with the findings reported by the original authors. As shown in Figure 5.1, approximately 40%, 90%, and 40% of the training data can be pruned from the synthetic, MNIST, and CIFAR-10 datasets,

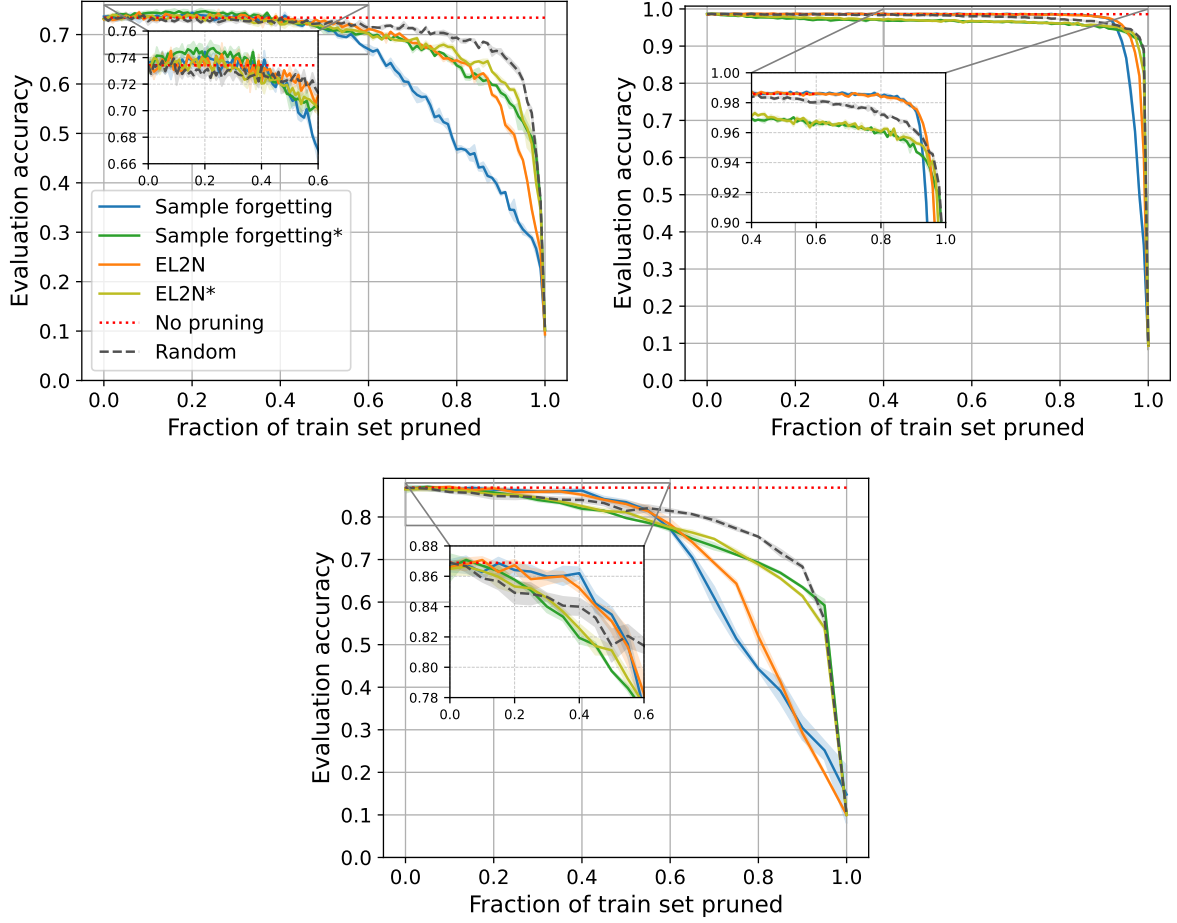


Figure 5.1: Model performance as a function of pruning an increasing portion of the train set for synthetic (left-top), MNIST (right-top), and CIFAR-10 (bottom) datasets. Samples are pruned in ascending order, with rankings provided by the two detectors in Section 5.3, or by a *random* baseline. The detectors with a *** in the title have their ranking reversed. The legend in the first plot applies to all three plots.

respectively, with negligible reductions in model performance (when pruning in ascending order of score). These results are consistent with prior work, where Toneva *et al.* [26] demonstrated that model accuracy can be maintained while pruning up to 80% of the MNIST train set, and both Toneva *et al.* [26] and Paul *et al.* [29] reported similar findings for CIFAR-10, showing that 30% – 50% of the dataset can be removed without significant performance loss. These observations reinforce the conclusion that substantial redundancy exists within these datasets.

However, closer inspection of Figure 5.1 reveals that the apparent advantages of the two original detectors (blue and orange ψ -curves) over the *random* baseline (gray) diminish when considering the entire range of the train set. At the aforementioned thresholds, the performance improvement over the *random* baseline is modest—approximately 2% for CIFAR-10 and even smaller for MNIST and the synthetic dataset.

Interestingly, the *random* baseline maintains higher model performance when the train set size becomes extremely small. Suppose the general intuition is that the samples pruned last are *important* for generalization. Why would the *random* baseline models outperform the two detectors so consistently in the most extreme cases (i.e., when only keeping the most important samples)? We hypothesize that while redundancy can be reduced, it cannot be eliminated. In particular, samples that appear redundant in large datasets may still contribute meaningfully to generalization when the train set is small. Random pruning, being agnostic to sample importance, inadvertently retains a mixture of both *redundant* and *informative samples*. In contrast, the detectors may over-prune redundant samples, thereby removing data that becomes crucial at smaller scales.

To test this hypothesis, we reverse the ranking order of the two detectors and repeat the pruning experiments, as shown by the *-marked ψ -curves (green and olive) in Figure 5.1 (pruning in descending order of score). When only the “least important” samples are retained, model performance unexpectedly improves relative to keeping only the “most important” samples. This finding partially supports the framework proposed by Sorscher *et al.* [24], which suggests that one should prune easy samples when data is abundant, but prune hard samples when data is scarce. Nevertheless, the observation that the *random* baseline outperforms all scoring methods under extreme pruning conditions implies that effective model generalization in such regimes requires a balanced mixture of both easy and hard samples.

In summary, when the train set exhibits high redundancy, a large proportion of samples can be removed (with either informed or random pruning) without significant loss in model performance. However, excessive pruning using sophisticated scoring criteria may lead to disproportionate degradation in accuracy, indicating that redundant samples are

not necessarily unimportant for model generalization.

5.5.2 Problematic samples

As discussed earlier, both Toneva *et al.* [26] and Paul *et al.* [29] advocate pruning samples in ascending order of their respective scores. However, the results presented in the previous section demonstrate that pruning in ascending order yields only marginal improvements over the *random* baseline, whereas descending-order pruning can result in improved performance when large portions of the dataset are removed. In this section, we evaluate the behaviour of the two detectors when the train sets explicitly contain problematic samples, such as those affected by label noise.

From Figure 5.2, it is evident that when using the original *sample forgetting* (blue) and *EL2N* (orange) detectors in the presence of strong label noise, model performance deteriorates catastrophically during pruning. Toneva *et al.* [26] observed that label-corrupted samples tend to receive higher forgetting scores, while Paul *et al.* [29] noted that training exclusively on high-scoring samples in noisy datasets may not be optimal. Building upon these insights, we analyse how the *EL2N* and *sample forgetting* scores rank clean and noisy samples before and after label corruption, as shown in Figure 5.3.

The results in Figure 5.3 indicate that label-corrupted samples generally occupy the highest positions in the ranking (i.e., they are pruned last) under the standard ascending-order scheme. We further observe that uncorrupted samples (black scatter points) largely retain their relative ranking after label corruption (best fit line pointing up), whereas corrupted samples that were previously ranked near the end tend to shift toward earlier positions in the ranking (best fit line pointing downwards). While the clarity of these trends varies across datasets, the overall pattern is consistent.

Based on the findings from Figure 5.3, we test the same simple modification as in Section 5.5.1: reversing the ranking order before pruning. As illustrated by the green and olive curves in Figure 5.2, this adjustment effectively mitigates the catastrophic drop in performance. When pruning in descending order, model performance not only re-

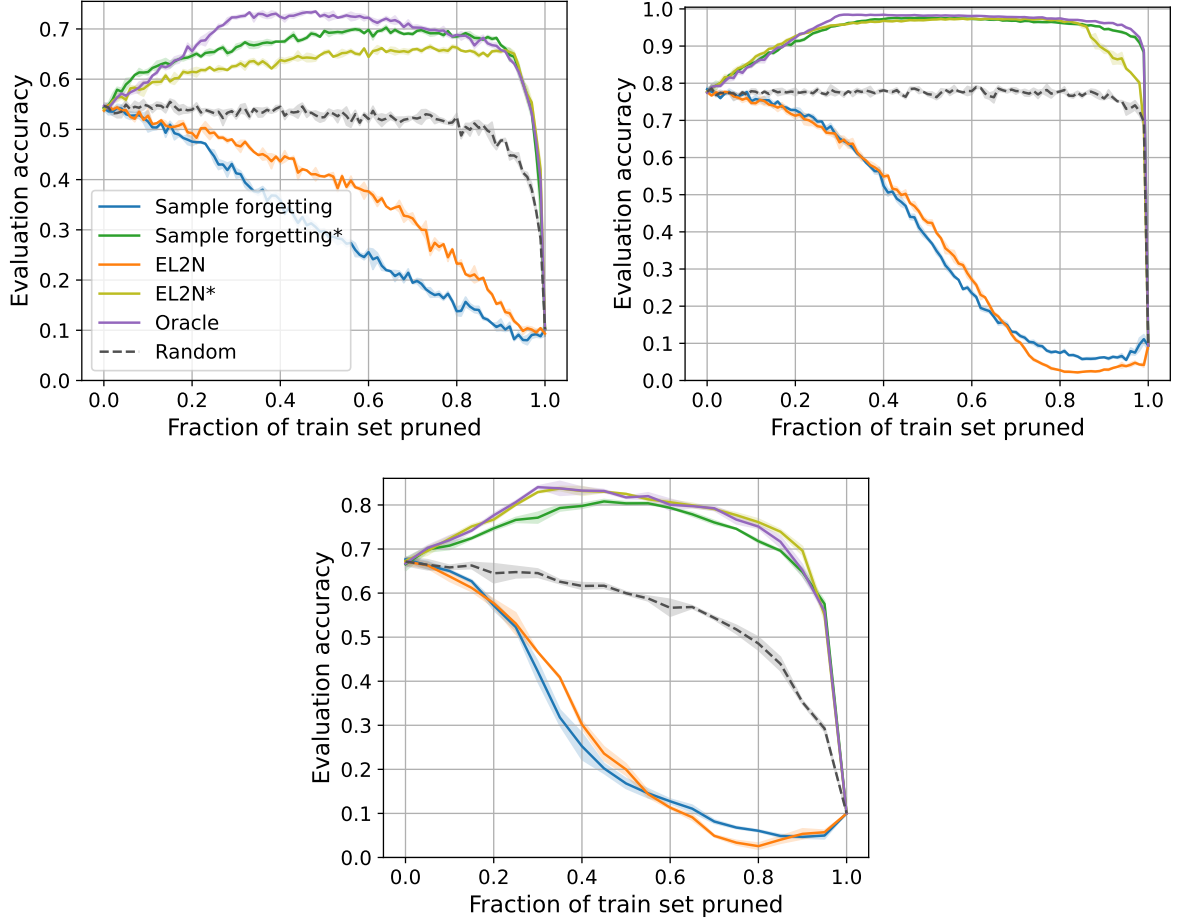


Figure 5.2: Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ symmetric label noise for synthetic (left-top), MNIST (right-top), and CIFAR-10 (bottom) datasets. Samples are pruned in ascending order, with rankings provided by the two detectors in Section 5.3. The detectors with a $*$ in the title have their ranking reversed. The *random* and *oracle* baselines are described in Section 4.2.3.

The legend in the first plot applies to all three plots.

mains stable but initially improves, as the label-corrupted samples (now ranked higher) are removed earlier. Remarkably, for MNIST and CIFAR-10, the *EL2N*-based ranking in descending order achieves performance levels comparable to the *oracle* detector that explicitly removes label noise before any clean samples.

In summary, these results highlight the importance of exercising caution when applying the *sample forgetting* and *EL2N* detectors in the presence of problematic samples. Under

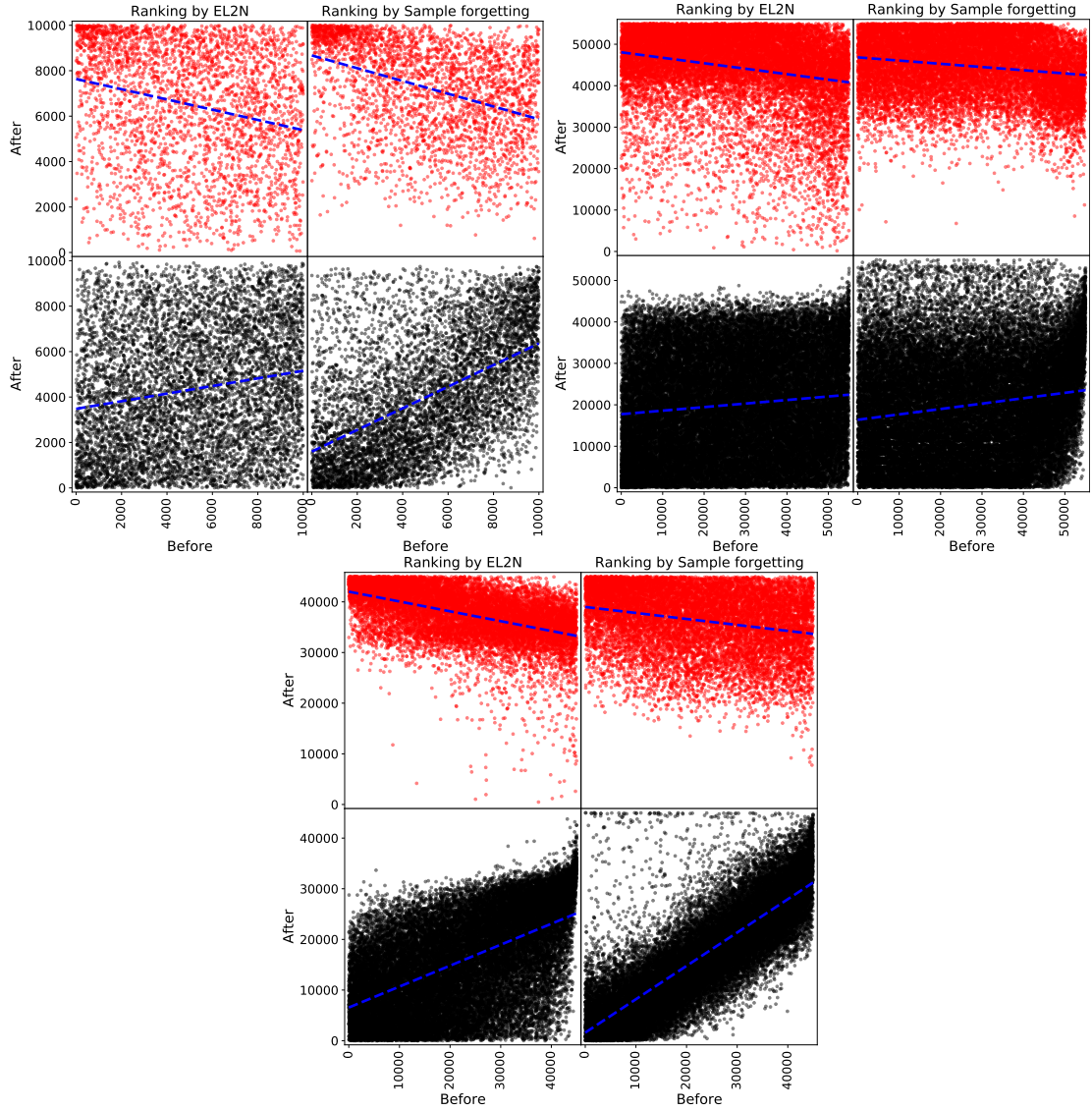


Figure 5.3: Sample ranking for synthetic (top-left), MNIST (top-right), and CIFAR-10 (bottom) before and after label noise is applied. The x -axis shows the samples’ ranking before label corruption and the y -axis shows the samples’ ranking after label corruption. Red samples have been label corrupted, while black samples were not corrupted. The blue dashed line is a linear best-fit line. The first column, in each plot, considers *EL2N* scores. The second column considers *sample forgetting* scores.

such conditions, their default ascending-order pruning strategy can lead to severe degradation (even worse than random pruning). A safer and more effective approach is to reverse the ranking order before pruning.

5.5.3 A deeper look

We conclude this section with an investigation of the characteristics of the problematic samples identified by the two detectors in Figure 5.2.

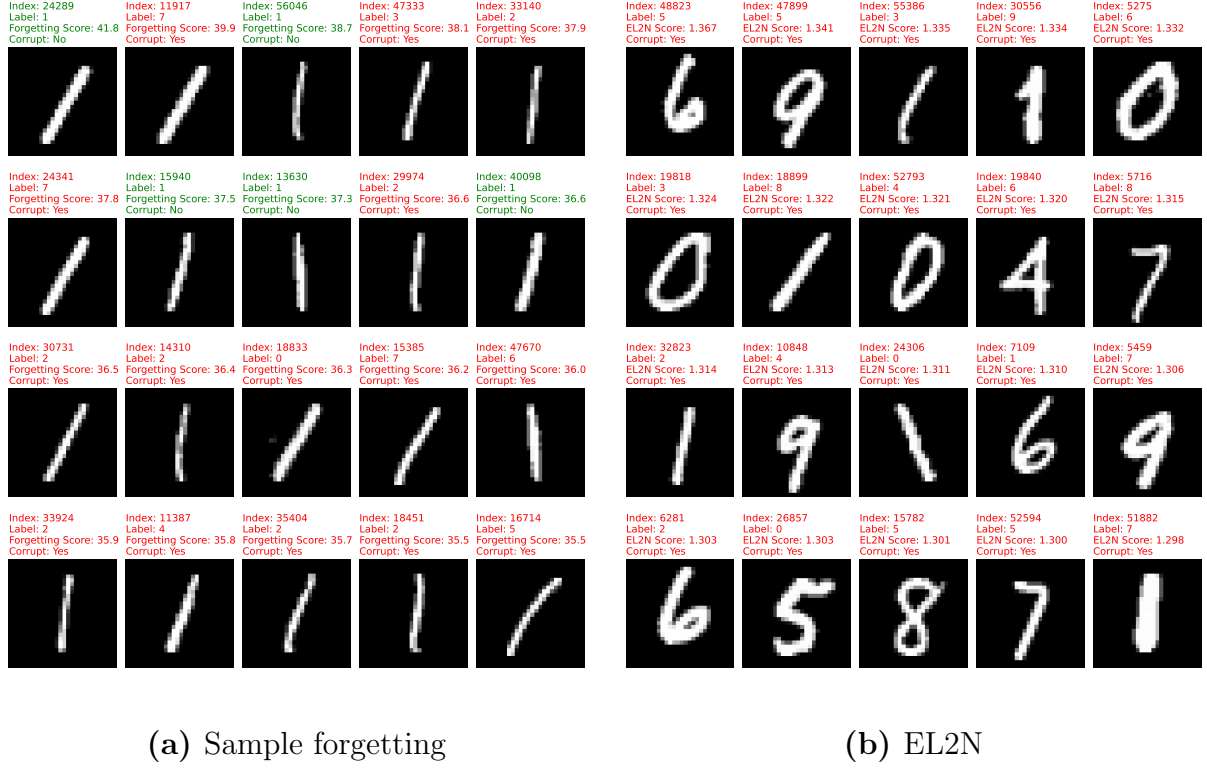


Figure 5.4: Top 20 MNIST train set samples with the highest scores for the two detectors (*sample forgetting* (a) and *EL2N* (b)) with 30% symmetric label noise. The images with information shown in red have corrupted labels.

We begin by looking at the top 20 samples in the MNIST dataset with the highest scores obtained from the two detectors, as shown in Figure 5.4. From Figure 5.4a, we make an immediate observation: all of the top 20 most forgettable samples share input features corresponding to class 1. Likewise, Figure 5.4b shows that class 1 input features accounts for 6 of the top 20 samples, exceeding the representation of any other class. Extending this analysis, we evaluate how many samples from each class are removed as we progressively prune the dataset according to the ranked detector scores in Figure 5.5. We observe that both the *sample forgetting* and *EL2N* detectors have a strong tendency to assign high scores to samples with input features corresponding to the ground truth class 1 (pre-

corrupted class 1), resulting in these samples to be pruned earlier than those from other classes.

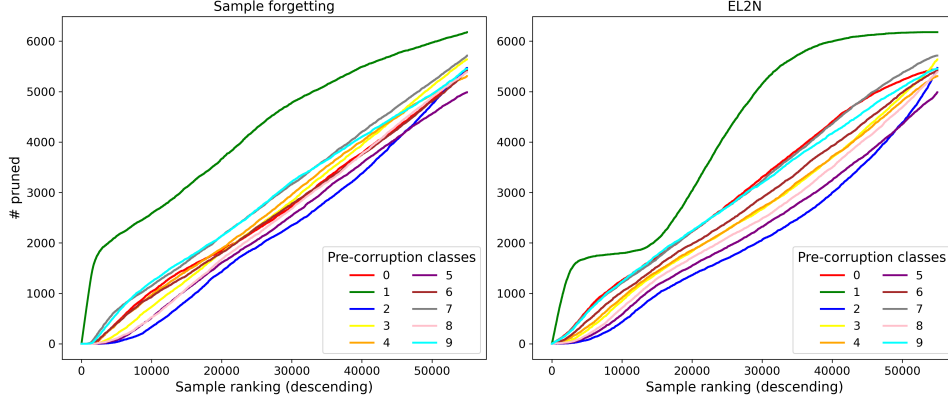


Figure 5.5: Number of MNIST samples from each pre-corruption class having been pruned at each point in order of the ranking obtained by *sample forgetting* (left) and *EL2N* (right), in descending order.

This indicates that, in our MNIST experimental setup, the samples most frequently forgotten, or those that have the largest error norms, are primarily driven by class 1 input features rather than label corruption in general. To further validate this hypothesis, we take a closer look at the type of samples that obtain large scores in Figure 5.6. In order to generate these heatmaps we simply grouped the entire train set into one of 100 possible categories, according to their target classes before and after corruption, and then calculated the mean and standard deviation of each group. The diagonal and off-diagonal entries of Figure 5.6, correspond to the clean- and corrupted samples, respectively.

From Figure 5.6 we make the following observations:

- The vertical patterns forming in the heatmaps indicates that the input structures tend to be stronger drivers for what the model struggles with, as reflected by the high forgetting events and error norms. These vertical patterns show that samples with similar input structures (i.e., belonging to the same ground truth class) tend to receive similarly high scores. This phenomenon is present for both detectors, though it is less pronounced for the *EL2N* detector.

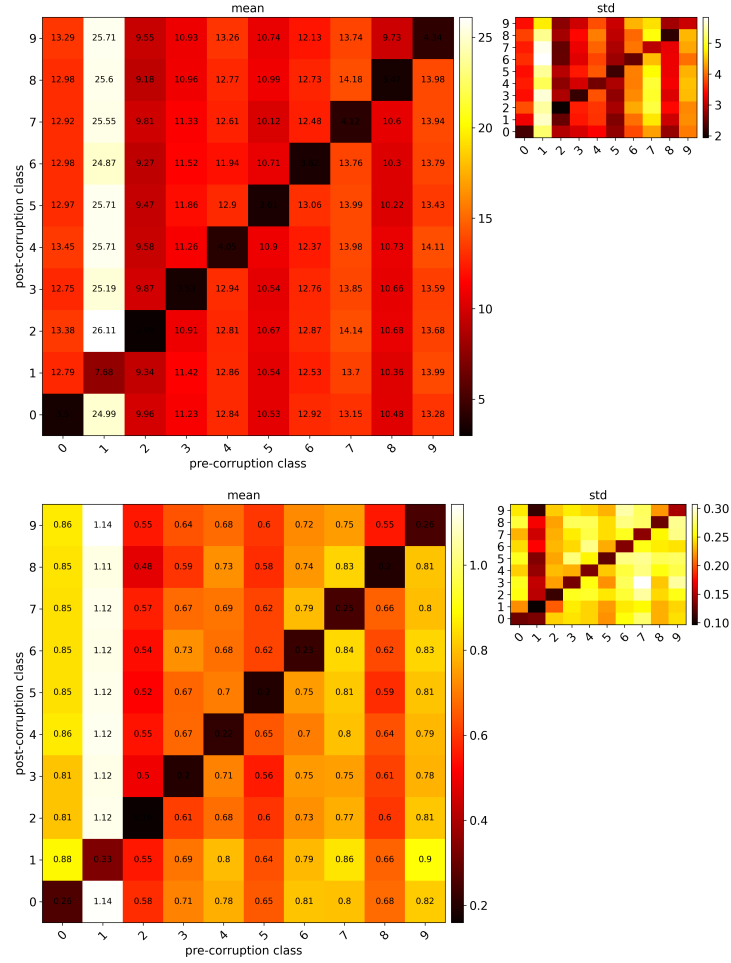
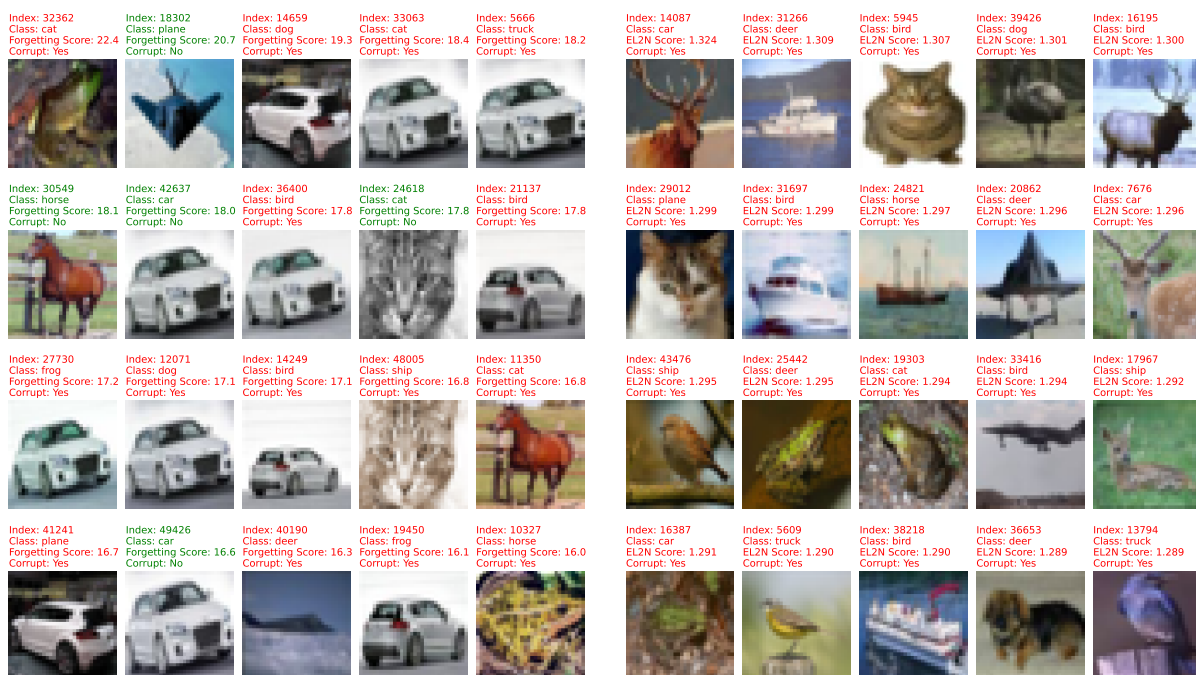


Figure 5.6: Mean *sample forgetting* score (top) and *EL2N* score (bottom) for MNIST samples according to their pre- and post-corruption target classes. The smaller plot indicates the standard deviations corresponding to each mean value.

- *EL2N* scores distinguish better between clean and corrupted samples, than *sample forgetting*, as indicated by the greater contrast between on- and off-diagonal values in the bottom heatmap.
- For the MNIST task setup, class 1 emerges as a problem class: the model struggles disproportionately with mapping class 1 input structures to other classes. This is reflected in the prominent bright vertical band, which shows that the model misclassifies class 1 inputs into many other classes more frequently than it misclassifies inputs from other classes into class 1.



(a) Sample forgetting

(b) EL2N

Figure 5.7: Top 20 CIFAR-10 train set samples with the highest scores for the two detectors (*sample forgetting* (a) and *EL2N* (b)) with 30% symmetric label noise. The images with information shown in red have corrupted labels.

We repeat our analysis on the CIFAR-10 dataset by examining the top 20 samples with the highest *sample forgetting* scores (Figure 5.7a) and *EL2N* scores (Figure 5.7b). In contrast to the MNIST results, the samples with the highest forgetting scores in CIFAR-10 do not originate from a single class, although the “car” class has a disproportionate presence (i.e. 12 out of 20 samples). We note, from Figure 5.7a, that many of the CIFAR-10 samples appear visually similar (almost like duplicates). While these are not exact copies, they are augmented transformations of the same base images, introduced by the dataset creators to increase dataset diversity.

Again, we consider the order of sample pruning in Figure 5.8. For CIFAR-10 we don’t observe a single pre-corruption class that is overwhelmingly prioritized. Instead we see that some classes—such as “car”, “truck”, and “ship”—are only slightly prioritized towards the start of pruning.

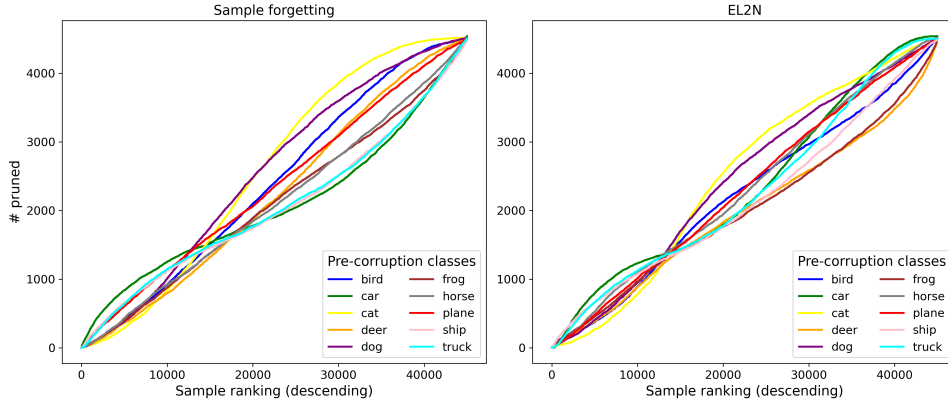


Figure 5.8: Number of CIFAR-10 samples from each pre-corruption class having been pruned at each point in order of the ranking obtained by *sample forgetting* (left) and *EL2N* (right), in descending order.

The CIFAR-10 heatmaps, equivalent to the MNIST heatmaps in Figure 5.6, is shown in Figure 5.9.

As expected, we do not observe the strong vertical pattern for any particular class, such as the pre-corruption class 1 in the MNIST setup. We also do not see the strong presence of vertical patterns (input structure focus) over horizontal patterns (target value focus). We do, however, make two observations with regards to Figure 5.9 specifically:

- The *EL2N* scores continue to provide better separation between clean and corrupt samples, as indicated by the clear contrast between the on- and off-diagonal values in the bottom heatmap.
- The CIFAR-10 model seems to struggle less when mapping animal classes to each other, as indicated by the darker central block for the *sample forgetting* heatmap corresponding to the classes: “horse”, “frog”, “dog”, “deer”, “cat”, and “bird”).

The second point might shed light on why the *sample forgetting* detector underperforms the *EL2N* detector as seen in Figure 5.2. Some classes are inherently easier (i.e. resulting in fewer forgetting events) to map to each other therefore resulting in corrupted samples to appear “clean”. This issue is less present for *EL2N* which is measured relatively early in training where label corruption might not have had a chance to be fitted.

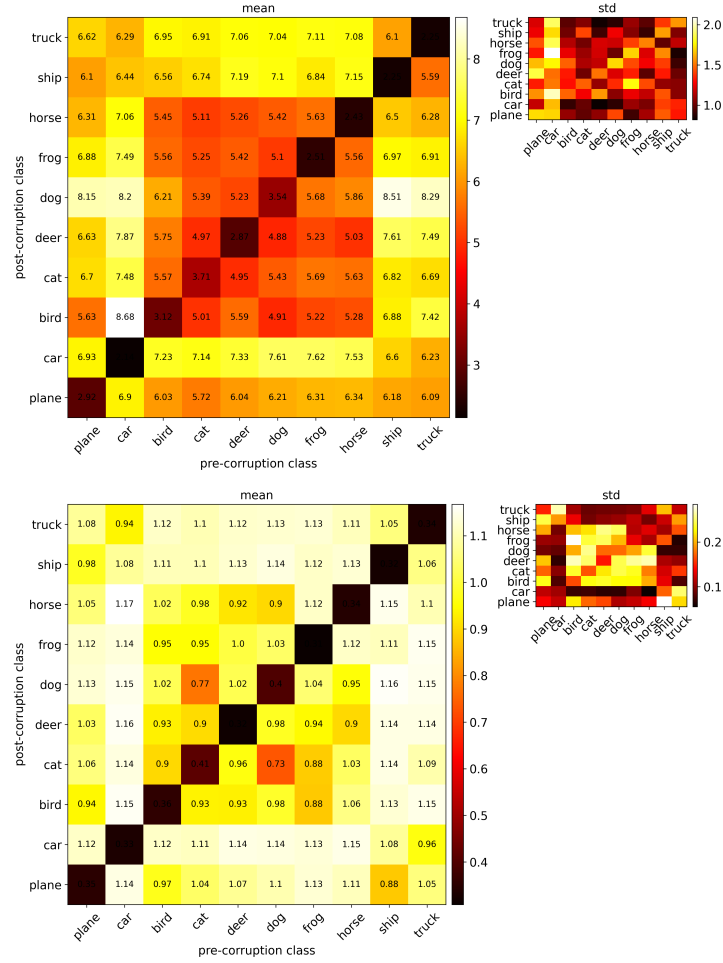


Figure 5.9: Mean *sample forgetting* score (top) and *EL2N* score (bottom) for CIFAR-10 samples according to their pre- and post-corruption target classes. The smaller plot indicates the standard deviations corresponding to each mean value.

Finally, we perform a sensitivity analysis of sample ranking stability—an aspect that neither Toneva *et al.* nor Paul *et al.* explicitly examined. Paul *et al.* [29] report that their scores are averaged over ten independent runs, but do not provide justification for selecting ten random seeds. Toneva *et al.* [26] note that their forgetting scores are stable across seeds, citing high Pearson correlations between scores obtained from different random initializations. However, both studies overlook an important consideration: it is not the absolute score stability that determines the effectiveness of pruning, but rather the stability of the *sample rankings* across seeds. Since the ranking directly determines which samples are pruned, its consistency is far more consequential for reproducibility than the absolute scores themselves. Sample scores may vary only slightly across different

random seeds, but their relative rankings can change significantly.

To investigate this, we compute the standard error of each sample’s ranking when scores are calculated across varying numbers of random seeds. Specifically, we calculate the standard error of the expected values from Equations 5.1 and 5.4, where the expectation is taken over different numbers of independent runs (seeds). We then compute the mean standard error across all samples in the train set to obtain a single mean ranking stability measure per number of seeds.

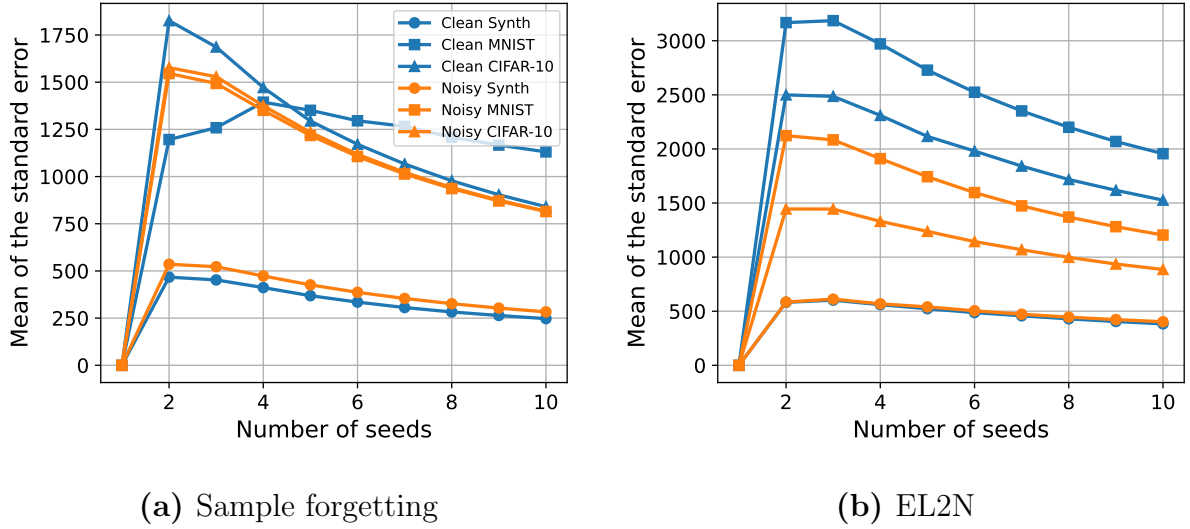


Figure 5.10: Sensitivity analysis on the ranking order of samples when the sample forgetting (a) and EL2N (b) scores are calculated across a different number of seeds. The y -axis shows the standard error of the samples’ rankings, averaged across all train set samples.

As shown in Figure 5.10, the mean standard error is initially high when scores are computed using two seeds, but gradually decreases as the number of seeds increases, for both detectors. Nonetheless, across all datasets, the mean standard error remains relatively high even after averaging over ten seeds, with the trend continuing downward. This suggests that employing more seeds would likely yield more stable and reliable rankings, although at the cost of increased computational overhead. Given practical computational constraints, we limit our analysis to ten seeds; however, it is important to note that further improvements in stability and detector performance may be achievable by averaging

over more seeds.

5.6 Discussion

This chapter evaluated the use of two popular data valuation methods as problematic sample detectors. Specifically, we examined whether the *sample forgetting* method by Toneva *et al.* [26] and the *EL2N* method by Paul *et al.* [29] can also identify problematic samples, rather than merely redundant ones, as they were originally designed to do. A series of data pruning experiments was conducted across the full range of the train set, leading to several noteworthy findings. The main results of this chapter are summarized below:

- Substantial redundancy was observed in all three classification datasets used in this chapter when the scores from both detectors were ranked in ascending order, as proposed by the original authors.
- When pruning to small train set sizes, the benefits of using the two detectors diminished. In these cases, random removal achieved higher evaluation accuracy. We reason that this occurs because samples that appear redundant in large datasets may still contribute meaningfully to model generalization when the train set is small. A balanced mix of “easy” and “hard” samples yields the best generalization in such scenarios.
- In the presence of label noise, pruning in ascending order (as suggested by the original methods) performed poorly, leading to a rapid and severe drop in evaluation accuracy worse than random removal. This happens because label-corrupted samples generally occupy the highest positions in the ranking (i.e., they are pruned last).
- Uncorrupted samples largely maintain their relative ranking after label corruption, whereas corrupted samples that were previously ranked near the end tend to shift toward earlier positions in the ranking.

-
- Based on these observations, we propose a simple modification: reversing the ranking order when label noise is present. This adjustment effectively repurposes these methods into problematic sample detectors. Under this approach, the presence of problematic samples becomes clearly visible in the evaluation framework, with the ψ -curves of both detectors closely following that of the *oracle* detector.
 - A deeper examination of the MNIST dataset revealed that class 1 input features constitute a disproportionately problematic region of the input space. Both *sample forgetting* and *EL2N* consistently assign high scores to class-1 samples, leading them to be pruned earlier than samples from other classes, regardless of whether their labels are corrupted.
 - For the CIFAR-10 dataset, the model exhibits a distinct pattern in which animal classes are more easily mapped to one another. This is particularly evident in the *sample forgetting* heatmap (Figure 5.9 top) which shows a dark central block (indicating fewer forgetting events). A similar, though weaker, pattern is also present in the *EL2N* heatmap.
 - One plausible explanation for the weaker performance of the *sample forgetting* detector relative to the *EL2N* detector on CIFAR-10 is that forgetting events can be suppressed for classes the model intrinsically finds easy to map to each other. As a result, corrupted animal-class samples may appear deceptively “clean” because they exhibit fewer forgetting events. In contrast, the *EL2N* detector is evaluated relatively early in training, when label corruption may not yet have been fully fitted, which can yield better performance on CIFAR-10. However, what counts as “early” depends on the dataset: on the synthetic dataset, epoch 20 may already be too late, as the model likely fits the label noise quickly on such an easy task. This may explain the *EL2N* detector’s suboptimal performance on that dataset. These observations highlight the need for a method that can adjust its measurement dynamically, removing the dependence on a fixed heuristic point as in *EL2N*.
 - Finally, our sensitivity analysis of the sample forgetting and EL2N scores revealed that using 10 random seeds, as proposed by the original authors, may be insufficient,

since a sample’s relative ranking can still vary considerably across runs.

In the next chapter, we continue our investigation into detecting problematic samples in image classification datasets. However, rather than employing data valuation methods, we implement a label noise detection approach and OOD detection method, evaluating their performance under different types of label- and input noise.

Chapter 6

On data-centric detectors

We implement data-centric methods to determine whether they can be used as problematic sample detectors.

6.1 Overview

In the previous chapter, we demonstrated that learning-centric methods can serve as effective problematic sample detectors by simply reversing the ranking order of two widely used data pruning methods¹. However, such learning-centric methods are known to be computationally expensive, as they rely on tracking the learning dynamics throughout the training process to compute sample scores. Moreover, as shown in Chapters 4 and 5, problematic samples predominantly correspond to those affected by noise: either in the label space Y or the input space X . Consequently, it is reasonable to consider more traditional label noise- and OOD detection methods as alternative means of identifying problematic samples, with the added advantage of reducing computational overhead.

¹At least for symmetric label noise as the surrogate problematic sample.

To this end we employ *Confident Learning (CL)* [56], a popular label noise detection method and *k-nearest-neighbours (k-NN) distance* [74], an OOD detection method, and determine whether they can be used as problematic sample detectors.

Section 6.2 introduces the concept of *a priori OOD detection* and outlines how this work addresses an existing gap in the literature. Section 6.3 describes the *CL* and *k-NN* methods in detail and explains how they are adapted for our study. Section 6.4 presents the experimental setup, followed by the results in Section 6.5. Finally, Section 6.6 concludes the chapter with a discussion of the key findings and directions for future work.

6.2 A priori OOD detection

As discussed in Chapter 2, OOD detection is a well-studied problem in the context of inference, where the objective is to distinguish between ID and OOD samples at test time. This distinction serves as an essential safeguard, allowing a model to recognize and abstain from making predictions on unfamiliar inputs, thereby preventing erroneous outcomes.

OOD detection is particularly critical in safety-sensitive applications such as autonomous driving, where incorrect predictions can have severe consequences. Despite the importance of OOD detection, there is often the expectation that models should generalize to inputs that have undergone *slight* distributional shifts. For example, in the case of a self-driving car, we would expect the model to handle covariate shifts, such as changes in input noise caused by environmental conditions like rain or fog.

Balancing the expectation for generalization under covariate shifts with the need to detect semantic shifts represents a complex and ongoing research challenge. A common approach to improving robustness to covariate shifts involves *OOD regularization*: exposing the model to OOD samples during training to encourage generalization. However, this practice introduces what is known as the *sensitive-robust dilemma* [77]: methods that enhance OOD detection often do so at the expense of model generalization to valid

inputs. This trade-off arises because OOD detection typically enforces high predictive uncertainty on auxiliary OOD samples. Unfortunately, this uncertainty can transfer to slightly shifted but valid samples, thereby reducing overall model performance.

As we will demonstrate in this chapter, exposure to OOD samples during training can impair downstream generalization to ID inputs. This exposure can occur intentionally as part of OOD regularization, or unintentionally through dataset contamination.

To this end, this chapter addresses the following:

1. We demonstrate that the presence of OOD samples during training can harm generalization to downstream ID data, even without the explicit uncertainty mechanisms common to OOD regularization. We further show that, under certain conditions, samples containing input noise can constitute a type of problematic sample—an aspect often overlooked in existing literature.
2. While substantial research has explored label noise and adversarial sample detection within the train set, comparatively little attention has been given to detecting OOD input noise (arising from covariate or semantic shifts) within the train set. We refer to this form of detection as *a priori OOD detection*.
3. Finally, we demonstrate that both *CL* and *k-NN* can, to some extent, be used to identify problematic samples exhibiting covariate shifts, semantic shifts, and label noise within our evaluation framework. However, some results also reveal considerable scope for improvement.

Existing research on label noise and OOD detection predominantly evaluates performance using metrics such as F1-score, FPR95, and AUROC. In contrast, by assessing these methods within our evaluation framework (using data pruning), we provide a complementary perspective on their ability to detect problematic samples.

6.3 Detectors

In this section, we present the *k*-NN distance method proposed by Kuan and Mueller [74] and the CL method proposed by Northcutt *et al.* [56].

6.3.1 k-nearest-neighbors distance

Kuan and Mueller [74] propose an OOD detection strategy in which a model is trained on (assumed) ID data, and test samples are subsequently classified as ID or OOD based on a selected metric. Specifically, they compute an OOD score for each test sample as the average cosine distance to its *k*-nearest neighbours in the train set, measured in the embedding space $\mathbf{z}$ of the penultimate hidden layer. A larger average distance indicates a higher likelihood that the sample is drawn from a distribution distinct from that of the train set.

To perform a priori OOD detection, we adapt their methodology by training a surrogate model on our train set, which may contain OOD samples. We then extract penultimate-layer embeddings for all training samples, and assign OOD scores according to Equation 6.1:

$$s_i = \mathbb{E} \left[\frac{1}{k} \sum_{n=1}^k \left(1 - \frac{\mathbf{z}_i \cdot \mathbf{z}_n}{\|\mathbf{z}_i\| \|\mathbf{z}_n\|} \right) \right], \quad (6.1)$$

where s_i and $\mathbf{z}_i$ denote the score and the embedding of sample $(\mathbf{x}_i, y_i)$, respectively, and $\mathbf{z}_n$ represents the embedding of the *n*-th nearest neighbour of $(\mathbf{x}_i, y_i)$. When $n = 0$, the nearest neighbour is the sample itself. The expected value is estimated by averaging across multiple model initialisation seeds.

Note that we apply the existing approach by disregarding the ID/OOD composition in the train set while averaging results across model seeds. The success of this approach therefore depends on the model’s ability to produce distinct embeddings for OOD samples during training, since preliminary experiments showed that training the surrogate model exclusively on ID samples (i.e., with oracle access) leads to improved detection performance. We will demonstrate that such separation can indeed occur.

6.3.2 Confident learning

Northcutt *et al.* [56] present a label noise detection method called *Confident Learning (CL)* which estimates the joint probability distribution between the potentially noisy (observed) labels $y \in Y$ and the true (unknown) labels $y^* \in Y$. Given out-of-sample predicted probabilities $\hat{\mathbf{P}} \in \mathbb{R}^{n \times c}$ for the train set, CL computes the *confident joint* $\mathbf{C} \in \mathbb{Z}_{\geq 0}^{c \times c}$, whose entry $[\mathbf{C}]_{j,l}$ counts the number of samples labelled $y = j$ that are likely to truly belong to $y^* = l$. These are samples for which:

$$[\hat{\mathbf{P}}]_{*j} \geq \frac{1}{|D_{\text{train}}^{(j)}|} \sum_{i \in D_{\text{train}}^{(j)}} [\hat{\mathbf{P}}]_{i,j}, \quad (6.2)$$

where $[\hat{\mathbf{P}}]_{*j}$ denotes the j -th column of $\hat{\mathbf{P}}$ and $D_{\text{train}}^{(j)}$ is the set of indices of training samples with observed label $y = j$. Note that the threshold in Equation 6.2 is simply the average predicted probability for class y , calculated by considering all the samples with label y . If more than one of the elements in a sample's probability vector $[\hat{\mathbf{P}}]_{i*}$ exceeds their class thresholds, the sample is confidently counted to belong to the class with the highest predicted probability. If none of the probabilities in $[\hat{\mathbf{P}}]_{i*}$ exceed their class thresholds, the sample is disregarded from consideration.

The off-diagonal entries of $\mathbf{C}$ correspond to *CL*'s estimate of samples with label errors, and these samples can be scored by their normalised output margin:

$$s_i = \mathbb{E} \left[[\hat{\mathbf{P}}]_{i,\hat{y}_i} - \max_{j \neq \hat{y}_i} [\hat{\mathbf{P}}]_{i,j} \right], \quad (6.3)$$

where $\hat{y}_i = \arg \max_j [\hat{\mathbf{P}}]_{i,j}$. The normalised output margin quantifies how confidently the model prefers one class over all others, by simply calculating the difference between the predicted probability $[\hat{\mathbf{P}}]_{i,\hat{y}_i}$ and the second highest probability $\max_{j \neq \hat{y}_i} [\hat{\mathbf{P}}]_{i,j}$, where the expectation is taken over multiple out-of-sample predictions. This approach corresponds to the default “ $\mathbf{C}_{\tilde{y},y^*}$ ” rank and prune method defined by Northcutt *et al.* [56], which has been shown to produce favourable results empirically.

To adapt the *CL* framework for data pruning, we extend the method by also scoring confidently counted clean samples (diagonal entries of $\mathbf{C}$) and the samples disregarded from consideration, using Equation 6.3. We then create separate rankings for detected

noisy and clean samples, concatenating the rankings such that all confidently counted noisy samples are prioritised for removal before any clean samples. The expected value in Equation 6.3 is estimated by averaging across multiple model initialisation seeds.

Northcutt *et al.* note that out-of-sample predicted probabilities $\hat{\mathbf{P}} \in \mathbb{R}^{n \times c}$ can be obtained either from a pre-trained model or via k -fold cross-validation. However, as discussed in Chapter 3, a problematic sample detector Λ is restricted to the train set D_{train} , the untrained model f and the learning process $\mathcal{A}$. Using a pre-trained model, which may be considerably larger or trained on external data, would violate the constraints of the problematic sample detector and provide an unfair advantage to CL over other detectors. To ensure methodological consistency, we therefore employ k -fold cross-validation to obtain $\hat{\mathbf{P}}$. Specifically, D_{train} is partitioned into k equally sized folds; for each iteration, a model is trained on $k - 1$ folds and used to generate predicted probabilities on the held-out fold. Repeating this process for all folds yields the full matrix of out-of-sample predicted probabilities $\hat{\mathbf{P}}$.

6.4 Experimental setup

Datasets We perform experiments on three multiclass classification datasets: MNIST [17], CIFAR-10 [18] and the synthetic dataset described in Section 4.2.1. The dataset sizes are consistent with those used in Chapter 5 (Table 5.1). All dataset characteristics are kept the same as outlined in Chapter 5 Section 5.4.

For experiments involving MNIST with input noise, we reduce the train set to 2,000 samples before introducing the covariate shift described below. To avoid confusion with the full MNIST dataset, we will refer to this reduced version as *MNIST nano*. This choice is motivated by prior findings showing that covariate- and semantic shifts have a negligible impact on model generalization when applied to the full MNIST train set [107]. A detailed discussion of the effect of dataset- and model size on model generalization performance is provided in Section 6.5.3. To accommodate the reduced train set size, we reduce the resolution hyperparameter to $r = 20$ for MNIST nano, which is in proportion

with the reduced train set size. We keep all other hyperparameters the same as stipulated in Table 5.1.

OOD types We evaluate the two detectors under the three types of distributional shifts: label noise (Section 6.5.1), covariate input noise, and semantic input noise (Section 6.5.2). The objectives are twofold: 1) determine whether the detectors can identify these types of shifts within the train set and 2) analyse how such shifts affect model generalization. For all experiments, 30% of the train set is replaced with OOD samples ($\eta = 0.3$).

Label noise is introduced following the definitions of symmetric, asymmetric, and instance-dependent noise from Section 4.2.2. Input noise is introduced as follows:

- **MNIST nano:** Affected samples are randomly rotated θ degrees around the centre of the image plane, where $\theta \sim \mathcal{U}(1, 360)$. This constitutes a covariate shift as defined in Section 3.5.
- **CIFAR-10:** Affected samples have their input features replaced by those of a random sample from CIFAR-100 [18]. This constitutes a semantic shift as defined in Section 3.5.
- **Synthetic:** Affected samples have their input features replaced by those sampled from a uniform distribution as described in Section 4.2.2. This constitutes a semantic shift as defined in Section 3.5.

As in Chapters 4 and 5, all evaluation accuracies are measured using the predefined, unmodified test sets of the respective datasets.

Models and hyperparameters We use three separate MLP models, one per dataset (synthetic, MNIST, and MNIST nano). Each model contains two hidden layers, and both hidden layers have the same number of nodes. For the synthetic and MNIST datasets, each hidden layer has 512 nodes, whereas for the MNIST nano dataset, each hidden layer has 16 nodes. For CIFAR-10, we use VGG-16 architecture [106] with the final 3 layers of the model being three fully connected layers with 4,096, 4,096, and 1,000 nodes, consistent with the original specification by Simonyan *et al.* [106].

All models are trained using the Adam [100] optimizer with batch sizes and learning rates specified in Table 5.1. Batch normalization [101] layers are applied after each hidden layer, before a ReLU activation function. All models are trained with cross-entropy loss. All models used for extracting penultimate-layer embeddings, performing k -fold cross-validation, and conducting data pruning are trained for a maximum of 200 epochs. Training is stopped earlier if 100% training accuracy is reached. The inputs $\mathbf{x}$ for the image datasets were normalized with min-max normalization, while the inputs for the synthetic dataset were normalized as discussed in Section 4.2.3 (subtracting the mean of each feature across the train set and scaling to unit variance).

Scoring and pruning We compute sample scores $\mathbf{s} = (s_1, s_2, \dots, s_{|D_{\text{train}}|})^\top$ for k -NN *distance* and *CL* detectors by averaging over three random seeds to obtain the expected values in Equations 6.1 and 6.3, respectively. For k -NN *distance*, we set $k = 10$ following the setup from Kuan and Mueller [74]. For *CL*, we use four-fold cross-validation to obtain out-of-sample predicted probabilities $\hat{\mathbf{P}}$, consistent with the setup in Northcutt *et al.* [56]. Both methods are implemented using the open-source `cleanlab` [108] package.

After scoring, we conduct data pruning experiments using Algorithm 1. Pruning performance is evaluated across three random model initializations, using the same architectures and hyperparameters as in the scoring step². The evaluation models are trained for 200 epochs or until interpolation, what ever comes first. The resolution parameter r , which controls the number of samples removed per step, is specified in Table 5.1. To reduce computational overhead, we set $\alpha = 0.6$ in Equation 3.6. This choice not only decreases runtime but also aligns with our focus in this chapter on analysing the initial segment of the ψ -curves, where the most informative trends typically emerge.

Lastly, in addition to the k -NN *distance* and the *CL* detectors, we also conduct pruning experiments using the *random* and *oracle* (where applicable) baselines, as described in Section 4.2.3.

²We do not use four-fold cross-validation when performing the pruning experiments for *CL*.

6.5 Results

In this section, we present the results of the *CL* and *k-NN distance* detectors under label noise (Section 6.5.1) and input noise (Section 6.5.2).

6.5.1 The effect of label noise

Figure 6.1 presents the results of the two detectors in a 3×3 grid of subplots, where each row corresponds to one of the three datasets (Synthetic, MNIST, CIFAR-10) and each column represents a different type of label noise (symmetric, asymmetric, and instance-dependent).

From these results, several key observations emerge. First, the *CL* detector (red ψ -curves) performs consistently well across all datasets and noise types. Its performance closely follows that of the *oracle* baseline and, in most cases, converges to nearly identical evaluation accuracy when approximately 60% of the train set is pruned. This indicates that *CL* is capable of identifying and removing a substantial portion of the label noise while maintaining generalization performance.

However, the *CL* detector shows its greatest weakness under asymmetric label noise (middle column). This can be observed from the larger gap in evaluation accuracy between the *CL* and *oracle* ψ -curves across all datasets. This result is intuitive: under asymmetric label noise, samples from one class are consistently flipped to another specific class rather than randomly across all classes. It is conceivable that in this special case the model will become more confident in predicting label corrupted samples since there will be a large sub-population of each class that is labelled as one specific alternative class. Since *CL* relies on the model’s predicted class probabilities, such confidently incorrect predictions reduce its ability to distinguish mislabelled samples, allowing these asymmetric cases to evade detection.

Although the *CL* detector consistently identifies problematic samples across all three datasets and label-noise types, it still underperforms the *oracle* baseline in every setting.

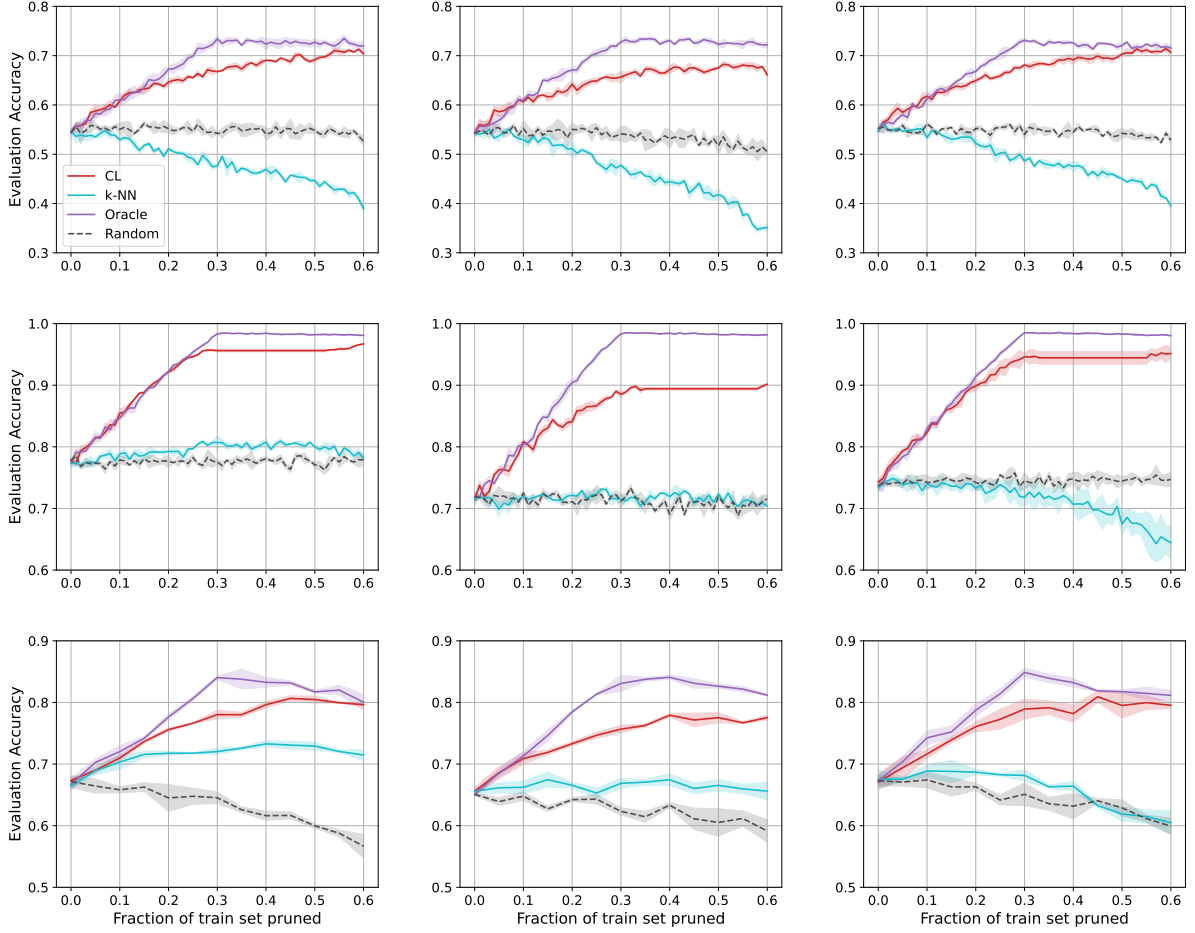


Figure 6.1: Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ label noise for synthetic (top row), MNIST (middle row), and CIFAR-10 (bottom row) datasets. The results for the three types of label noise are shown in the columns: symmetric (left column), asymmetric (middle column) and instance-dependent label noise (right column). Samples are pruned in descending order, with scores provided by the two detectors in Section 6.3. The *random* and *oracle* baselines are described in Section 4.2.3. The legend in the first plot applies to all nine plots.

This suboptimal performance can be attributed to a single factor: the off-diagonal entries of the confident joint matrix $\mathbf{C}$ do not correspond exactly to the samples with explicit label noise (i.e., the samples we intentionally corrupted). If the *CL* framework had accurately and exclusively placed the explicitly corrupted samples into the off-diagonal entries (i.e., counted them as noisy), then the peak values of the *CL* and *oracle* ψ -curves would have

coincided at the 0.3 pruning point. We can assert this despite the fact that the max-margin ranking of the off-diagonal entries might not perfectly match the *oracle* baseline’s initial ranking (corresponding to ground truth noise).

Since the off-diagonal entries of $\mathbf{C}$ do not align perfectly with the explicitly corrupted samples, two possibilities arise:

1. The *CL* framework confidently counts fewer samples as noisy than the actual amount of explicit label noise, or
2. The *CL* framework confidently counts more samples as noisy than the explicit label noise.

The first scenario points toward low recall and the second suggests low precision. To explore this we plot the confident joint for all nine datasets in Figure 6.2.

Figure 6.2 shows, as headings for each heatmap, the total number of off-diagonal entries in the confident joint alongside the total amount of explicit label noise. Across all nine datasets, the *CL* framework consistently identifies more samples as corrupted than the number of explicitly added label noise, indicating low precision. The results also offer insight into why the *CL* detector struggles with asymmetric label noise (middle column): the *CL* framework confidently assigns samples with a given reference label not only to the preceding class (matching actual asymmetric label noise) but also to the subsequent class (which is not introduced with asymmetric label noise). This behaviour is visible as three bright diagonal bands in the middle column heatmaps. This is not entirely surprising, since we know that approximately 30% of all class j samples are relabelled as class $j + 1$ under asymmetric label noise. Assuming ground truth class j samples are fairly well clustered in the input space, we can deduce that the result of the noise would be a dense region containing many correctly labelled class j samples, but also a significant portion incorrectly labelled $j + 1$. The model would likely struggle to distinguish between samples from class j and $j + 1$ in this region, resulting in significant uncertainty in both directions. We expect the *CL* framework to undergo similar difficulties as the noise rate for symmetric label noise increases.

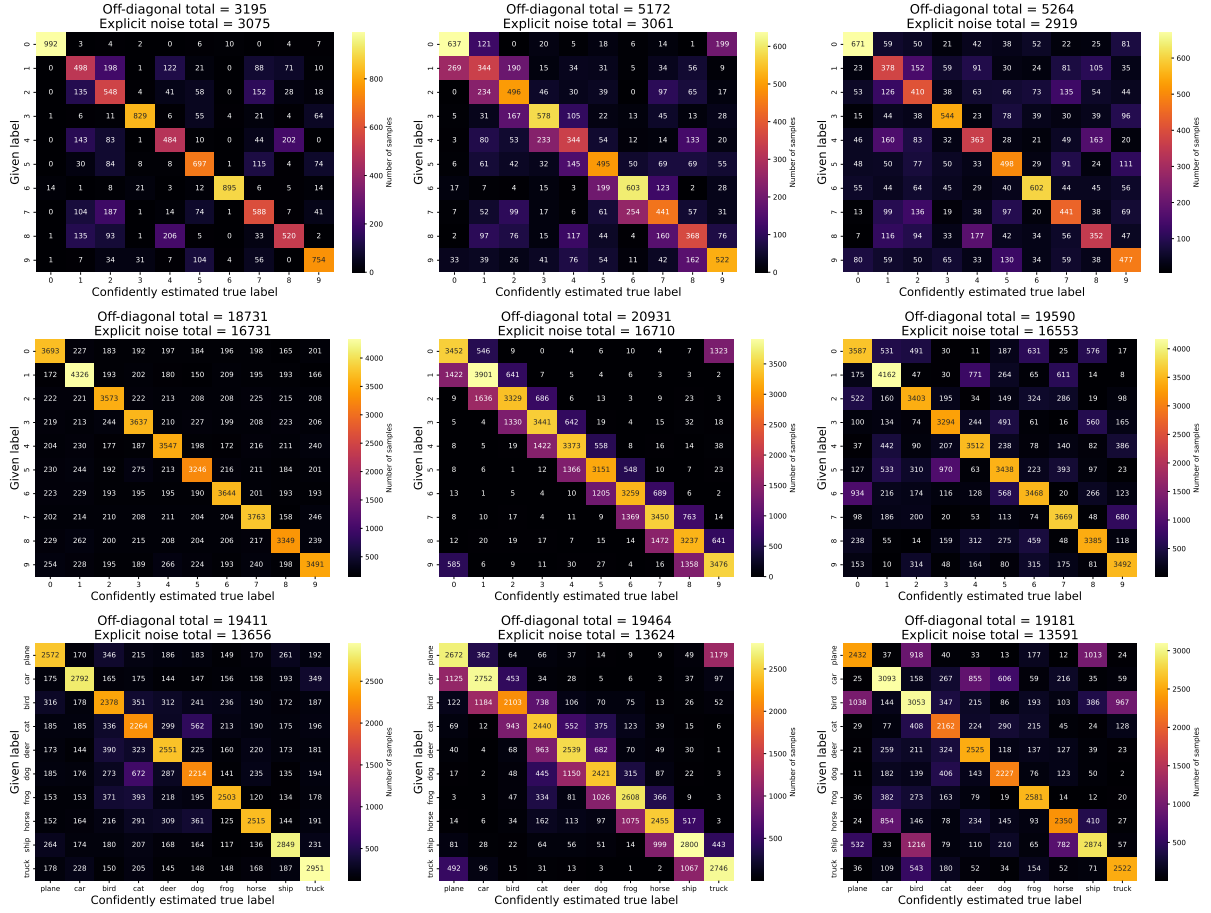


Figure 6.2: Confident joint ($\mathbf{C}$ matrix) of the train set with $\eta = 0.3$ label noise for synthetic (top row), MNIST (middle row), and CIFAR-10 (bottom row) datasets. The results for the three types of label noise are shown in the columns: symmetric (left column), asymmetric (middle column) and instance-dependent label noise (right column).

In contrast to the CL detector, the k - NN distance detector fails to reliably assign corrupted samples with the highest scores as shown in Figure 6.1. This is evident when comparing the cyan ψ -curve with the CL and baseline detectors. This is not a surprise, as the k - NN distance detector does not directly³ utilize class labels; instead, it operates solely on the feature representations extracted from the penultimate hidden layer of the trained model.

The most pronounced failure of the k - NN distance detector is observed on the synthetic

³Although label information may be implicitly encoded in the learned embeddings at this stage of the network.

dataset (top row). We hypothesize that this poor performance arises from this dataset’s characteristics, specifically, the “spherical” (isotropic Gaussian) shape and the overlap between the class distributions. It could be that samples located near the periphery of their respective class distribution (particularly those positioned away from other classes) probably receive the largest distance scores under the k -*NN distance* criterion. Consequently, these peripheral yet valid samples are pruned first, leaving behind samples concentrated in the overlapping regions of the distributions. This residual dataset is inherently more challenging to classify, and because label noise remains present, the model’s performance deteriorates. To test this hypothesis we examine the relationship between a sample’s k -*NN distance* score and its distance to the closest class centroid (i.e., the mean feature vector of all samples belonging to that class), excluding the sample’s labelled centroid in Appendix A.2 for the nine datasets in Figure 6.1. Our results show that the k -*NN distance* detector has a strong tendency to assign high scores to samples that are located far away from the closest class centroid.

Despite this failure, the k -*NN distance* detector demonstrates noteworthy performance on the image datasets, consistently outperforming the *random* baseline (with the exception of the MNIST dataset under instance-dependent label noise). This result is especially surprising, given that the detector has no explicit access to label information. Nonetheless, it appears capable of assigning high scores to some problematic samples. This finding suggests that samples lying far from their nearest neighbours in the learned feature space tend to be inherently problematic for the model, in the presence of label noise. This phenomenon is particularly evident for the CIFAR-10 dataset with symmetric label noise (bottom left), where the k -*NN distance* detector achieves its strongest relative performance.

In summary, the experimental results demonstrate that *CL* is a highly effective problematic sample detector under label noise, performing comparably to the *oracle* baseline across datasets and noise types, with only minor degradation under asymmetric noise. In contrast, the k -*NN distance* detector shows mixed performance: it fails under overlapping synthetic distributions possibly due to its reliance on feature-space distances, yet consistently outperforms *random* pruning on image datasets. These findings suggest that

while the *CL* detector is well-suited for detecting label noise, the *k-NN distance* detector can still capture meaningful feature-space anomalies, highlighting its potential as a label-agnostic problematic sample detector.

6.5.2 The effect of input noise

We evaluate the effect of the three types of input noise described in Section 6.4, with the results of all the pruning experiments presented in Figure 6.3.

From Figure 6.3, we observe that both covariate- and semantic shifts in the train set can substantially degrade a model’s generalization performance. This is evident from the *oracle ψ* -curves, which show that removing the 30% of input noise samples leads to a marked improvement in evaluation accuracy across all three datasets. The results for the MNIST nano dataset (Figure 6.3b) are particularly noteworthy: although covariate shifts are typically not regarded as harmful, this experiment demonstrates that even these shifts can impair model performance when data and model capacity is limited. This finding suggests that practitioners should exercise caution when incorporating covariate-shifted samples during training for OOD regularization, as these samples may inadvertently reduce generalization.

The *k-NN distance* detector performs relatively well on the MNIST nano (Figure 6.3b) and CIFAR-10 (Figure 6.3c) datasets but fails dismally on the synthetic dataset (Figure 6.3a). This outcome indicates that uniform input noise does not produce distinct or separable embeddings in the learned feature space. Moreover, the *k-NN distance* detector underperforms the *random* baseline on the synthetic dataset (consistent with observations in Section 6.5.1), suggesting that samples located near the edges of their class distributions tend to receive the highest distance scores, leaving behind a more ambiguous and challenging subset of samples after pruning as suggested in Appendix A.2.

In contrast, the *CL* detector performs well across all datasets in Figure 6.3. It consistently outperforms the *random* baseline and effectively identifies problematic samples, despite being originally designed for label noise detection. This finding suggests that covariate-

and semantically shifted samples elicit model behaviours similar to those induced by label noise, particularly in terms of output confidence. In essence, the model’s softmax outputs (on which *CL* relies) reflect the degree to which the internal model representations separate classes. We hypothesize that samples that lie outside the main cluster of their respective class (as covariate- and semantically shifted samples probably do) tend to exhibit lower

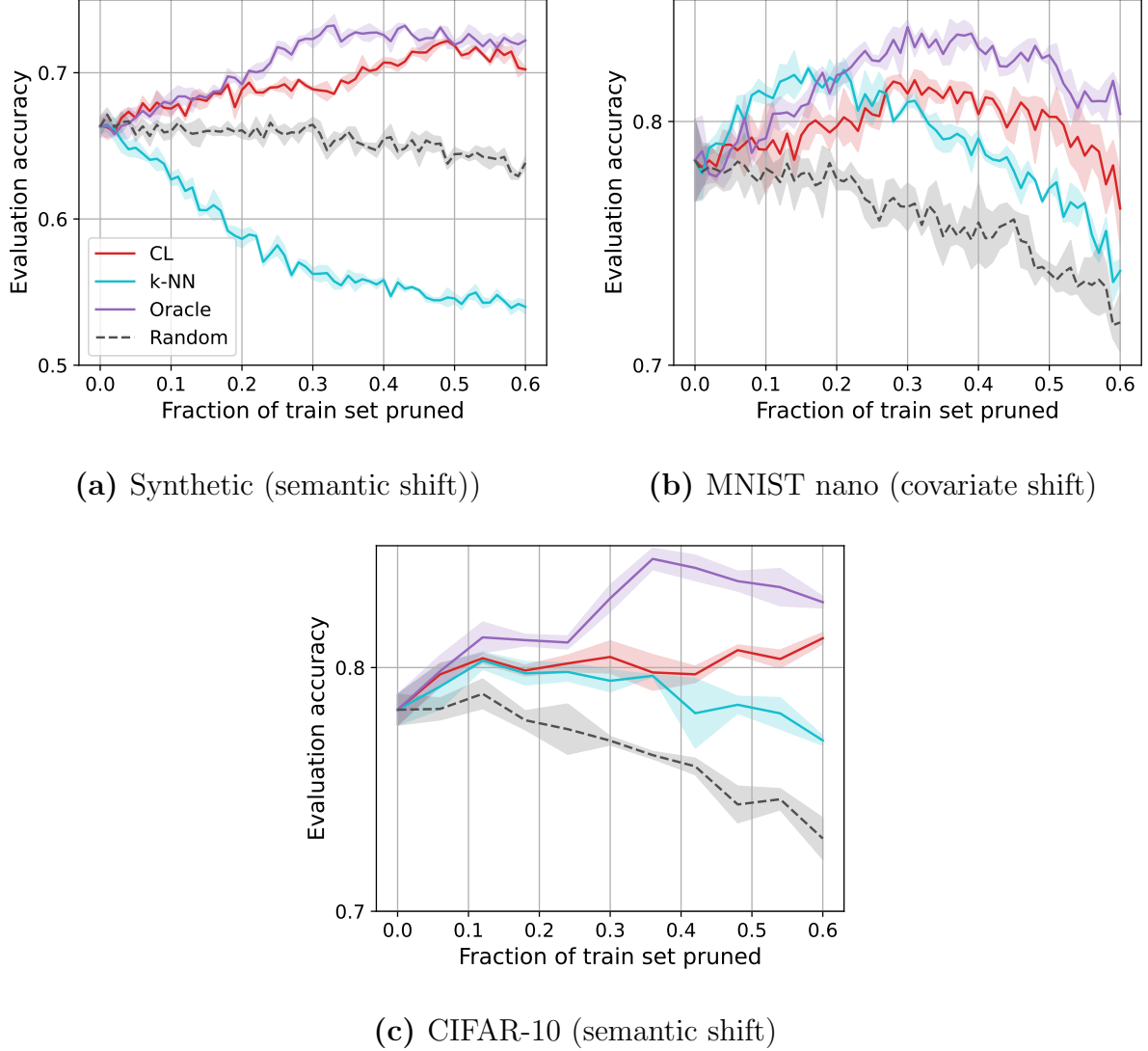


Figure 6.3: Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ input noise for synthetic (a), MNIST nano (b), and CIFAR-10 (c) datasets. Samples are pruned in descending order, with rankings provided by the two detectors in Section 6.3. The *random* and *oracle* baselines are described in Section 4.2.3.

The legend in the first plot applies to all three plots.

maximum predicted probabilities and higher entropy in their predictive distributions. Consequently, the *CL* framework is able to capture this spatial discrepancy indirectly through its dependence on the predicted probabilities. This highlights that *CL* can serve as a versatile problematic sample detector of various types of distributional shifts.

However, when comparing the performance of the *CL* detector on CIFAR-10 under semantic shift (Figure 6.3c) with its performance under the three types of label noise (Figure 6.1 bottom row), it remains evident that the detector performs best when label noise is present. This can be observed from the smaller gap between the *CL* and *oracle ψ* curves in the label noise experiments compared to those involving semantic input noise. We hypothesize that this outcome is related to the effect of semantically shifted samples on the per-class thresholds used by the *CL* framework. If the predicted probability vectors for semantically shifted samples indeed have higher entropy, this would reduce the per-class confidence thresholds (which are computed according to Equation 6.2, as the average predicted probability for that class). As a result, some semantically shifted samples may be incorrectly counted as confidently belonging to their assigned class. Furthermore, suppose a semantically shifted sample’s probability vector has very high entropy and low predicted probability values across the range of classes. In that case, the sample can be disregarded from consideration in the *CL* framework, since none of the probabilities exceed their respective class thresholds. This suggests that the *CL* detector’s strong performance in label noise settings stems, at least in part, from its ability to exploit the model’s confident misclassifications of mislabelled samples, i.e., those with low-entropy output distributions.

In summary, the results demonstrate that both covariate and semantic input noise can substantially impair model generalization, a finding often overlooked in literature. Among the two detectors, *CL* proves effective at identifying such problematic samples despite its original design for label noise, whereas *k*-NN shows limited reliability, particularly on the synthetic dataset.

6.5.3 Dataset size and model capacity

To investigate the effect of dataset size and model capacity on the model’s generalization performance in the presence of covariate-shifted OOD training samples, we train multiple MLP models of varying sizes on differently sized subsets of the MNIST dataset. Each entry in the heatmap in Figure 6.4 shows the difference in evaluation accuracy between two models: 1) a 2-layer MLP with hidden layer size as indicated by the horizontal axis, trained on a clean dataset of size indicated by the vertical axis, and 2) the same model trained where 30% of the samples have been randomly rotated. For example, the value 0.05 (top left) represents the difference in test accuracy between a model trained on 10,000 clean MNIST samples and a model trained on 10,000 MNIST samples with 30% covariate-shifted OOD samples (for an MLP with two hidden layers of 16 nodes each). The batch size and learning rate are kept the same as indicated in Table 5.1 for the MNIST dataset. We use batch normalization layers and ReLU activation functions to keep the models consistent with our setup in Section 6.4. All models were trained until interpolation.

Reading the heatmap from top to bottom, we notice that as the train set size decreases, the effect of OOD training samples becomes more pronounced, peaking around a dataset size of 750. Interestingly, when the dataset size decreases further, the negative effect diminishes. For instance, 30 rotated samples within a train set of 100 have a negligible impact. We hypothesize that this is because at extremely small dataset sizes, the model cannot adequately learn the ID task regardless of OOD contamination, making the effect of OOD samples less distinguishable.

Furthermore, reading the heatmap from left to right, we observe that increasing model size reduces the impact of OOD samples. We attribute this to the larger capacity of wider MLPs, which enables them to effectively learn both the ID and OOD subsets in parallel. As a result, the generalization performance on the ID test set is less affected. This suggests that model capacity can, to some extent, compensate for the presence of OOD samples in the train set.

To conclude this investigation, we also evaluate the impact of four types of covariate shifts

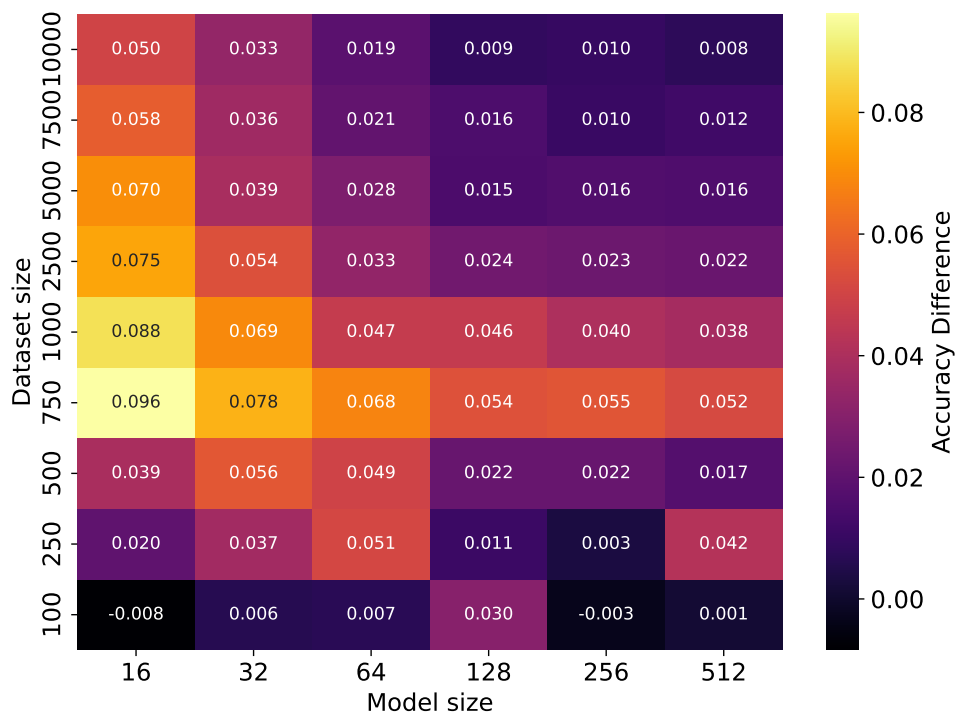


Figure 6.4: Heatmap showing the difference in evaluation accuracy between models trained on clean ID data and models trained with 30% covariate-shifted (randomly rotated) OOD samples for the MNIST dataset. Rows correspond to different train set sizes, and columns correspond to the hidden layer size of a 2-layer MLP. Each value represents the average result over three random seeds.

on the generalization performance of a 2-layer MLP with 16 nodes in each hidden layer. Examples of the four shift types are shown in Figure 6.5.

Figure 6.6 shows that generalization performance steadily improves as the train set size increases, seemingly converging beyond 5,000 samples. Consistent with Figure 6.4, the largest drop in evaluation accuracy compared to models trained solely on clean ID data occurs when the train set size is between 750 and 1,000. Finally, we find that random rotations and vertical flips lead to a greater reduction in evaluation accuracy than salt-and-pepper noise or Gaussian blur. We hypothesize that this is because rotations and flips can alter the class interpretation of some samples; for instance, rotating a ‘6’ may cause it to resemble a ‘9’. Regardless, the trend across dataset sizes remains consistent for the four types of covariate shifts analysed.



Figure 6.5: An illustration of four types of covariate shifts. The original (ID) sample on the left.

6.6 Discussion

This chapter evaluated the performance of the *Confident Learning (CL)* and *k-nearest neighbours (k-NN) distance* detectors under the three types of OOD samples introduced in Section 3.5. The analysis aimed to determine: 1) whether these detectors can be used to identify different forms of distributional shift within the training data, and 2) how such shifts effect model generalization. The main findings are summarized below:

1. Label noise remains a major source of problematic samples across all models and datasets (Figure 6.1), consistent with the findings from Chapters 4 and 5.
2. The *CL* detector reliably identifies all three types of label noise, though it struggles most with asymmetric noise (Figure 6.1). This is because the *CL* framework confidently counts many samples as corrupted in the confident joint, assigning them to both adjacent classes of each true class (Figure 6.2).
3. The suboptimal performance of the *CL* detector may be attributed to its tendency to confidently classify more than the explicit 30% label noise as mislabelled samples (Figure 6.2). This hypothesis aligns with previous findings by Northcutt *et al.* [109], who reported that natural label noise is pervasive in benchmark datasets such as MNIST and CIFAR-10.
4. The *k-NN distance* detector performs inconsistently under label noise, failing on the synthetic dataset due to removing boundary samples that aid class separation (Figure A.2), but still outperforms random pruning on the image datasets.

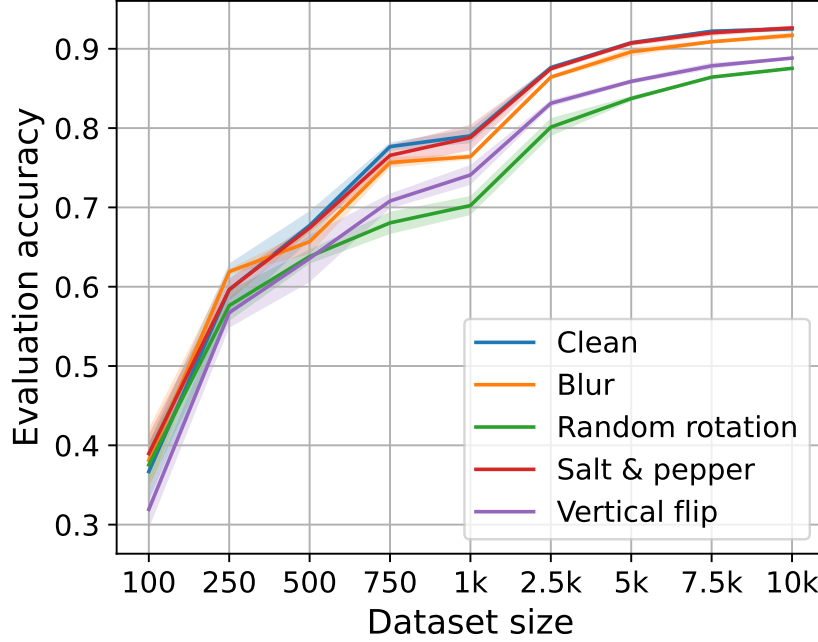


Figure 6.6: Generalization performance of models trained on MNIST subsets with 30% OOD samples included in each train set. Results are averaged over three random seeds.

5. Both covariate- and semantic shifts are shown to introduce problematic samples (Figure 6.3), highlighting an underexplored issue in the literature and cautioning practitioners about the potential drawbacks of OOD regularization.
6. The k -NN *distance* detector partially detects covariate and semantic shifts in image datasets but fails to do so for synthetic data.
7. The *CL* detector generalizes well beyond label noise, effectively identifying both covariate- and semantic-shifted samples, despite not being explicitly designed for this purpose.
8. The negative impact of covariate-shifted OOD samples on model generalization is strongest at moderate train set sizes, but fades when the dataset becomes extremely small. Larger models are less affected overall, as their higher capacity allows them to fit OOD samples without substantially harming ID performance (Figure 6.4).

These results open up several promising directions for future work. Notably, the response of the *CL* detector to input noise (an aspect not explored by Northcutt *et al.*) suggests

that its per-class confidence thresholds could be recalibrated using the entropy of predicted probabilities. Incorporating entropy into threshold calculation (Equation 6.2) may enable more accurate detection of input noise in the train set.

The effectiveness of the k -*NN distance* detector, on the other hand, is inherently tied to the quality of the learned embeddings. While embedding quality can be improved through larger or pre-trained models, our findings indicate that simply averaging distances to a sample’s k nearest neighbours provides only a partial view of problematic samples. Future work could enhance this approach by incorporating class- or label-specific clustering information, enabling the detector to better distinguish between genuinely informative and problematic samples in the embedding space.

In the next chapter, we introduce a new problematic sample detector based on the insights gained from Chapters 5 and 6, evaluate the problematic sample detectors on a real-world dataset, and investigate the effect of interdependencies between training samples.

Chapter 7

Possible avenues for improvements

We present a new problematic sample detector, evaluate detectors on a real-world dataset and investigate the interdependence of training samples.

7.1 Overview

The previous chapters demonstrated that both learning-centric and data-centric approaches can effectively identify problematic samples, though there remains room for improvement. In this chapter, we propose a new method designed to address specific shortcomings of the learning-centric detectors. Furthermore, the problematic sample detectors have only been evaluated on train sets with explicitly added noise (except for the results in Figure 5.1). While injecting noise into small datasets is a common practice (often used to emulate the challenges encountered in larger datasets) it remains important to assess how these methods perform on real-world data. Finally, as discussed in previous chapters, complex interdependencies likely exist among samples, which current problematic sample detectors don't directly take into account.

This chapter concludes the main investigation of this study by addressing these outstanding issues. Specifically, we begin by designing a new problematic sample detection method informed by the insights gained in earlier chapters. We then evaluate all detectors discussed in this work on a real-world dataset that contains no artificially added noise. Finally, we examine the issue of sample interdependence and how it affects current problematic sample detectors and our evaluation framework.

Section 7.2 introduces the *Variance-Weighted Fitness Gap (VWFG)* detector, outlining its derivation from insights obtained through *sample forgetting* and *EL2N*, and comparing its performance against these methods. Section 7.3 presents the evaluation of problematic sample detectors on a chest X-ray dataset used for identifying pulmonary edema. Finally, Section 7.4 investigates the effect of interdependence of training samples, by presenting an iterative evaluation framework and utilizing influence functions (as outline in Section 2.5) to study interdependence from two perspectives.

7.2 Variance-Weighted Fitness Gap (VWFG)

In this section, we present the *VWFG* problematic sample detector and evaluate its performance against the two learning-centric detectors: *sample forgetting* and *EL2N*.

7.2.1 Intuition

The methods presented in Chapter 5 demonstrated that learning-centric approaches are effective problematic sample detectors. The results showed that the *EL2N* detector closely tracks the *oracle* baseline—even surpassing it on CIFAR-10 (Figure 5.2) while the *sample forgetting* detector closely follows. However, on the synthetic and MNIST datasets, the *sample forgetting* detector slightly outperformed *EL2N*. These findings suggest that a detector leveraging the strengths of both methods may offer improved performance.

As noted by others [24], [110], at their core, learning-centric detectors measure how *difficult*

the model finds it to correctly predict each sample during training. *EL2N* does so by approximating the sample gradients early in training, whereas *sample forgetting* quantifies how often a sample is forgotten, i.e., how many times the model fails to maintain a correct and stable prediction. Both methods therefore capture complementary signals about sample difficulty.

We build on this intuition, but several limitations of the existing methods motivate a new approach. First, *EL2N* requires selecting a specific epoch at which to measure the score—a choice that is inherently heuristic and may not generalize across datasets or architectures. Ideally, this choice should be determined dynamically, as mentioned in Section 5.6. Second, by relying on a single early-epoch snapshot, *EL2N* may miss informative dynamics that unfold later in training. Third, while *sample forgetting* incorporates information across the full training trajectory, it treats forgetting events in later epochs as equally important as those in early epochs. Yet, later in training we expect sample fitting dynamics to be more stable and most samples to be fitted, making it harder to distinguish difficult samples. Additionally, it is commonly thought that simpler patterns are learned earlier in training and these patterns are more likely to generalize [46], [79], [80].

In response to these considerations, we propose a detector with the following elements:

- Track the *fit-status* of all samples across the entire training trajectory, as in the *sample forgetting* detector.
- Average these trajectories over multiple random seeds to obtain a more robust estimate, mitigating the effect of stochastic training noise.
- Compute, for each sample, the difference between its fitness curve and the average fitness curve of all samples over epochs.
- Weight this deviation by the spread of the overall fitness curve at each epoch, thereby emphasizing differences during periods when the model struggles most.

This formulation aims to unify the temporal richness of *sample forgetting* (i.e., its use of

the full sequence of fit-status information across training) with the sensitivity to sample difficulty embodied in $EL2N$, producing a detector that is both more dynamic and more informative. In the next section, we formally present this method.

7.2.2 Method

Consider a *fitness matrix* $\mathbf{F} \in \mathbb{R}^{n \times e}$, where rows represent n training samples and columns represent e ordered training epochs. Entries $[\mathbf{F}]_{i,j} \in \mathbf{F}$ indicate how strong the tendency is for the sample at row i to be fitted at the j -th epoch. We generate $\mathbf{F}$, in practice, by tracking per-sample classification errors during training and then averaging over a number of seeds. If $[\mathbf{F}]_{i,j} = 0$ it means that the sample i was never fitted for any seed at epoch j . If $[\mathbf{F}]_{i,j} = 1$ it means that the sample i was always fitted for all seeds at epoch j .

We define the *mean fitness curve* across the entire training dataset as $\boldsymbol{\mu} \in \mathbb{R}^e$ where the j -th entry is simply defined as:

$$\mu_j = \frac{1}{n} \sum_{i=1}^n [\mathbf{F}]_{i,j}. \quad (7.1)$$

This curve represents the average level of fitness across all samples during training.

We define the *fitness variance curve* across the entire training dataset as $\boldsymbol{\sigma} \in \mathbb{R}^e$ where the j -th entry is simply defined as:

$$\sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n ([\mathbf{F}]_{i,j} - \mu_j)^2}. \quad (7.2)$$

This curve represents the spread of fitness levels across all samples during training.

For a training sample, represented by row i , we define the Variance-Weighted Fitness Gap (*VWFG*) score as:

$$s_i = \frac{1}{e} \sum_{j=1}^e (\mu_j - [\mathbf{F}]_{i,j}) \sigma_j. \quad (7.3)$$

Our problematic score for sample i is therefore just a sum¹ of differences between the

¹For data pruning experiments the $\frac{1}{e}$ term in Equation 7.3 is disregarded since we are ranking according to this score.

mean fitness level and the sample specific fitness level, weighted by the relevant level of fitness variance across all samples.

This score has the following advantages over *EL2N* and *sample forgetting*:

- It removes the need for a heuristically chosen epoch at which to measure the *EL2N* score.
- Considers the entire training procedure similar to *sample forgetting* (i.e. it uses more information on training dynamics to calculate the score).
- Dynamically weights the contributions of each epoch based on: 1) how far below the average expected fitness the current sample is, and 2) how much variance across samples exists at the current epoch.

The last advantage is particularly important because of the strong tendency for problematic samples to present as high-error samples early in training, but show similarly low error rates later in training. By making the adjustment dynamic, we ensure that even if the model takes longer to converge on a good solution (or even goes through phases of de-converging) each epoch will be appropriately weighted.

As an aside, note from Equation 7.3 that we actually weight the contribution of each epoch by the *standard deviation* and not the *variance* as the name of the score suggests. If we replaced σ_j with σ_j^2 we would effectively cancel out the root in Equation 7.2 and weight by the variance. Increasing this exponent would place increasing importance on epochs where the model is far away from convergence. Since this importance would likely be specific to the model and dataset dynamics we think of the exponent as a hyperparameter that can be calibrated. In our experiments we use a static exponent of 1, keeping the weighting relatively conservative, and noting the possibility of improvement with fine-tuning.

7.2.3 Experimental setup

To compare the *VWFG* detector to the *sample forgetting* and *EL2N* detectors, we keep most of the experimental setup consistent with the experimental setup in Chapter 5 Section 5.4.

Datasets We evaluate the detectors on two datasets: the multiclass synthetic dataset described in Section 4.2.1, and CIFAR-10 [18]. We decide to keep experimenting on the synthetic dataset due to the controlled nature of the dataset, the reduced computational overhead when compared to image datasets, and most importantly, because this dataset presented more of a challenge to the detectors as shown in Figure 5.2. We keep the dataset sizes the same as reported in Chapter 5 Table 5.1. In experiments involving label noise, corruption is applied exclusively to the train sets. Specifically, for the experiments in this chapter, we employ symmetric label noise and asymmetric label noise, as outlined in Section 4.2.2. We decide to test the detectors’ performance on asymmetric label noise, since it showed to be the most challenging type of label noise for the data-centric detectors in Figure 6.1.

Models For the synthetic- and CIFAR-10 dataset, we employ the same MLP and VGG-16 architectures as in Chapter 5. All models are trained using the Adam optimizer [100], with the same batch size and learning rates as specified in Chapter 5 Table 5.1. Batch normalization layers [101] are applied after each hidden layer and before a ReLU activation function. Inputs are normalized as discussed in Section 5.4.

Scoring We compute sample scores $\mathbf{s} = (s_1, s_2, \dots, s_{|D_{\text{train}}|})^\top$ using the three detectors defined in Equations 5.1 (*sample forgetting**), Equation 5.4 (*EL2N**) and Equation 7.3 (*VWFG*), averaging results over ten randomly initialized models. To measure forgetting events and to obtain the fitness-curve, each model is trained for 200 epochs. For the *EL2N** detector, error vectors are recorded after 20 epochs of training, consistent with the experimental setup in Chapter 5. All detector scores are pruned in descending order.

Pruning After assigning a score to each training sample, we apply Algorithm 1 to generate a ψ -curve for every detector Λ . At each point ψ_i along the curve, the model performance

is evaluated over three random initializations. These evaluations were performed using the same architecture and learning process used for scoring. The evaluation models are trained for 200 epochs or until interpolation, what ever comes first. We keep the resolution parameter r (Table 5.1) and $\alpha = 1$ consistent with the setup in Chapter 5. In addition to the three detectors, we also conduct pruning experiments using the *random* and *oracle* baselines, as described in Section 4.2.3.

7.2.4 Results

The results in Figure 7.1 show that the *VWFG* detector performs exceptionally well at assigning high scores to problematic samples in the presence of label noise. In particular, it achieves near identical evaluation accuracy as the *oracle* baseline when 30% of the samples are removed, with the exception of the synthetic dataset under asymmetric label noise (top-right). When comparing *VWFG* with the other detectors up to the 30% noise threshold, we observe that *VWFG* consistently outperforms them—except on CIFAR-10 with symmetric noise (bottom-left), where it matches the performance of *EL2N**. These findings highlight that the dynamic nature of *VWFG* enables it to reliably distinguish mislabelled samples from clean ones across different datasets and noise types. To confirm this statement, we show the cumulative corruption percentage as the train set is pruned according to the scores of the three detectors in Appendix A.3 Figure A.2. The results show that the majority of corrupted samples (about 90%) are pruned at 30% of the *VWFG* curve.

However, the results in Figure 7.1 also reveal a clear limitation: the performance of the *VWFG* detector declines rapidly once the label noise has been removed—more sharply than the *oracle* baseline (especially noticeable for CIFAR-10). Because *VWFG* scores samples according to the model’s difficulty in fitting them, weighted by the spread of the average fitness curve, it correctly assigns high scores to mislabelled samples. Yet it then tends to score clean but hard-to-fit samples immediately after. As mentioned by Toneva *et al.* [26] and Paul *et al.* [29] (and explored in Section 5.5.1), these difficult (or hard-to-fit) clean samples play an important role in generalization, thus removing them can lead to

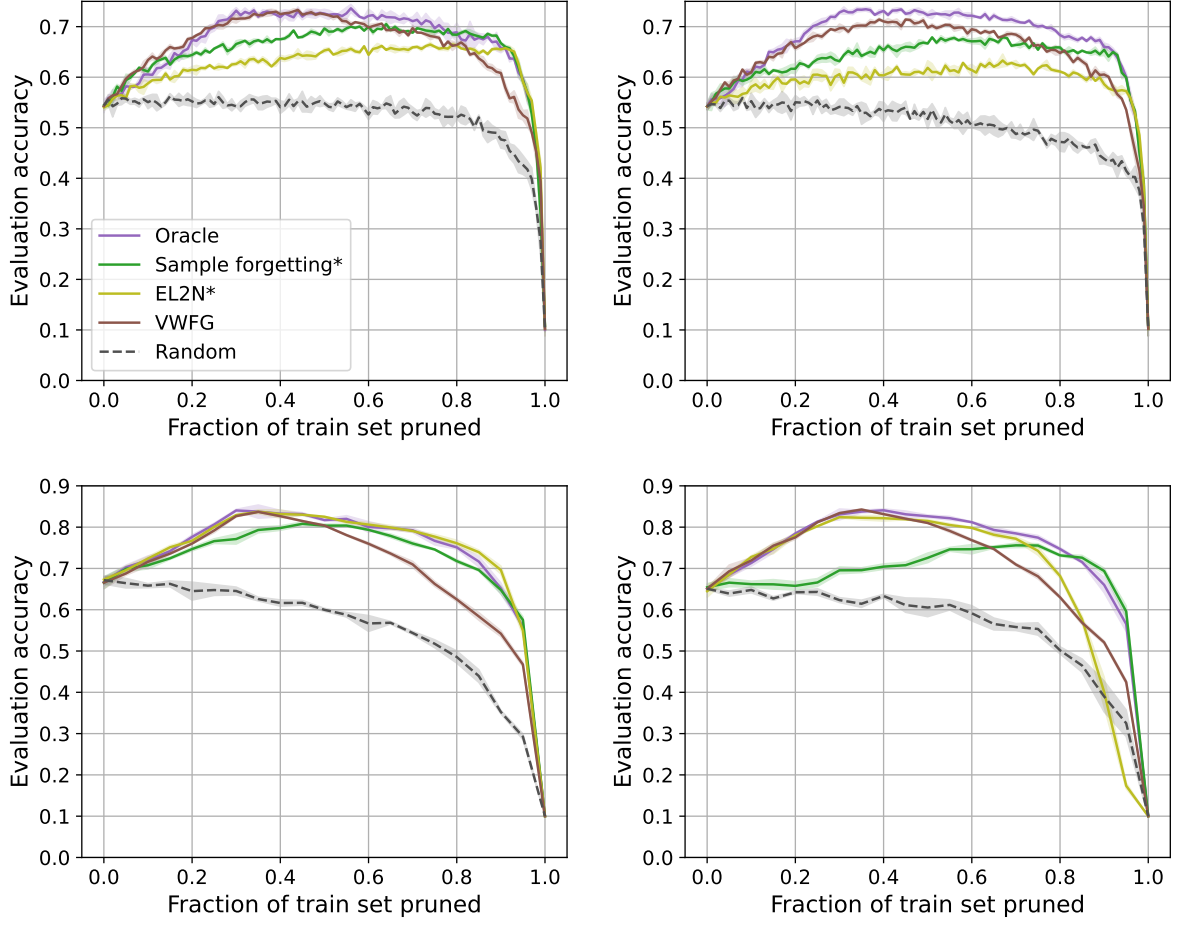


Figure 7.1: Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ label noise for synthetic (top row) and CIFAR-10 (bottom row) datasets. The results for the two types of label noise are shown in the left column (symmetric) and right column (asymmetric) label noise. Samples are pruned in descending order, with scores provided by the three detectors. The *random* and *oracle* baselines are described in Section 4.2.3. The legend in the first plot applies to all four plots.

the observed performance drop.

Figure 7.2 displays the fitness matrices $\mathbf{F}$ for the four datasets. The rows of each matrix are sorted in ascending order according to the *VWFG* score, with each row corresponding to a single training sample. These heatmaps reveal a consistent pattern: noisy samples—the samples *VWFG* ranks highest and therefore appear near the bottom of each matrix—tend

to remain unfit (blue) during the early stages of training.

However, the heatmaps also show that not all early-unfit samples are noisy. Because only 30% label noise was introduced, many of the samples that appear unfit early but are not noisy must be clean, hard-to-fit samples that contribute meaningfully to generalization. Their presence aligns with our earlier observation that *VWFG*, after removing mislabelled samples, begins to assign high scores to clean but hard-to-fit samples.

An especially interesting pattern emerges in CIFAR-10 with asymmetric label noise (Figure 7.2 bottom-right). In this case, some of the easy-to-fit samples in the upper rows of the matrix begin to appear unfit at roughly the same epochs when the mislabelled samples begin to fit. This behaviour could be due to the nature of asymmetric label noise, where labels are flipped to specific incorrect classes. The point at which these patterns emerge likely corresponds to the stage of training where the model starts to form two competing distributions for every class—one clean and one noise-induced—causing certain previously stable samples to become harder to fit.

In summary, the *VWFG* detector excels at assigning the highest scores to label-noise instances (the problematic samples) and is therefore an effective choice when the goal is to isolate the problematic subset from the training data. However, because *VWFG* is inherently dynamic, it assigns higher scores to hard-to-fit yet clean samples that can benefit generalization, than easy-to-fit clean samples. It is worth noting that this clear separation among problematic samples, hard-to-fit clean samples, and easy-to-fit clean samples is a phenomenon we expect to primarily observe when label noise is explicitly added to the train set, and therefore, cannot be certain that this pattern would manifest in datasets where no explicit noise is introduced. In the next section, we evaluate all problematic-sample detectors on a real-world dataset without explicitly adding noise into the train set.

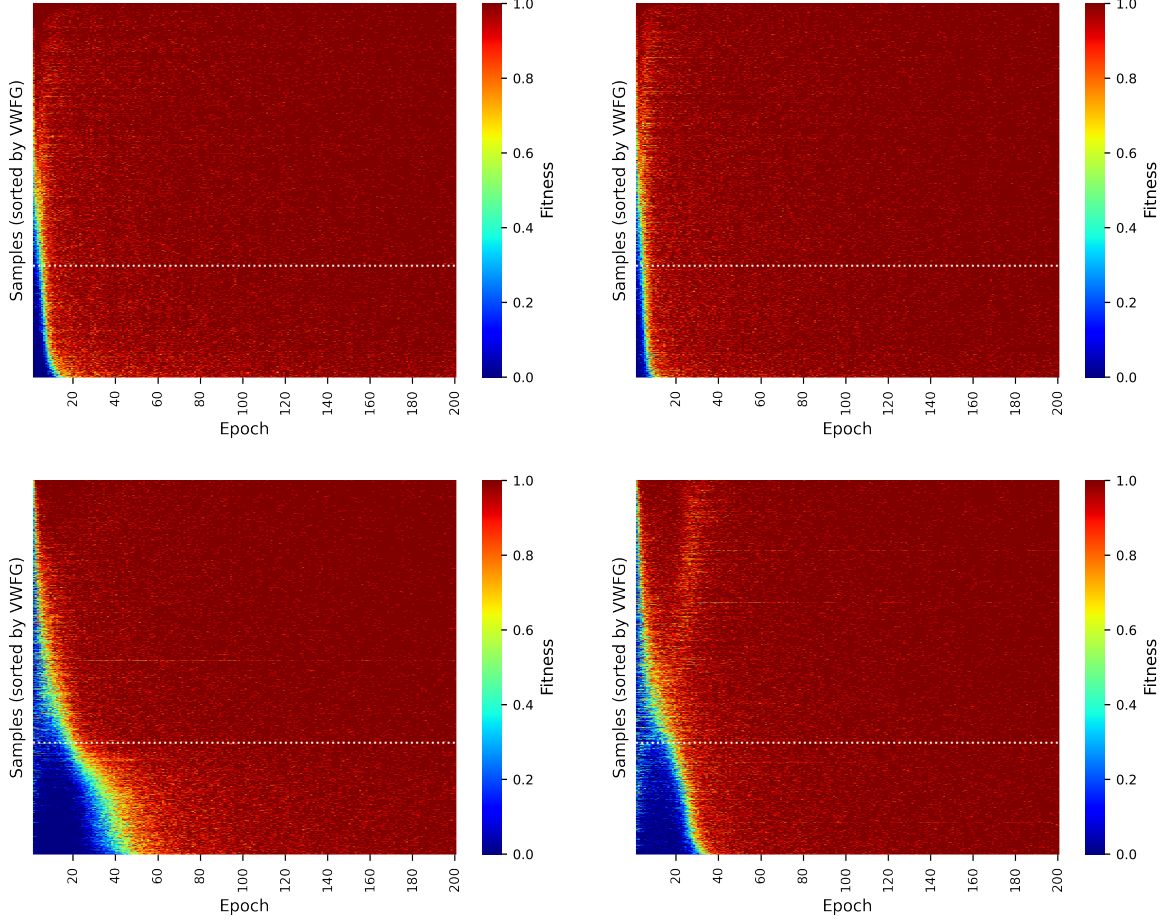


Figure 7.2: Fitness matrices $\mathbf{F}$ of the train sets with $\eta = 0.3$ label noise for synthetic (top row) and CIFAR-10 (bottom row) datasets. The results for the two types of label noise are shown in the left column (symmetric) and right column (asymmetric) label noise. The rows of each matrix are sorted in ascending order from top to bottom according to the *VWFG* score, with each row corresponding to a single training sample. The white dotted line shows the 30% pruning threshold, when samples are pruned from bottom to top (descending order of score).

7.3 Real-world analysis

As previously noted, the detectors evaluated in this study have primarily been assessed on datasets with explicitly added noise. In this section, we extend our investigation to a real-world dataset in order to determine: 1) whether problematic samples arise in a dataset without artificially introduced noise, and 2) whether the detectors can reliably

identify and indicate their presence under our evaluation framework.

7.3.1 Dataset

Our aim is to evaluate the detectors not only on an image classification dataset where we do not explicitly add noise, but also on a dataset that reflects realistic conditions under which problematic samples may occur. *Chest radiography* offers an ideal setting for this purpose. It is the most common imaging examination worldwide and plays a central role in screening and diagnosing numerous diseases [19]. Computer vision applications for automated diagnosis of chest radiographs, at the level of practising radiologists, could substantially improve clinical decision support and reduce workload in medical environments.

The Stanford ML Group’s CheXpert dataset [19] is particularly well suited for our analysis. It contains 224,316 chest radiographs from 65,240 patients, with some patients contributing both frontal and lateral X-ray images. To annotate such a large dataset, the authors developed an automated labeller that identifies the presence of 14 observations that includes 12 pathologies, alongside “support devices” and “no finding”. Mentions of these observations are first extracted from radiology reports, after which natural language processing techniques assign each observation in each image a label of *positive*, *negative*, or *uncertain*. Using these labels, the Stanford group trained a CNN that achieved performance exceeding that of radiologists on a curated test set of 500 studies annotated by five board-certified radiologists, and benchmarked against three additional radiologists.

This dataset is an excellent fit for evaluating problematic sample detectors for two key reasons. First, reliability and accuracy are critical in medical imaging applications, where even seemingly minor improvements (by removing problematic samples) can meaningfully affect diagnostic outcomes. Second, large-scale datasets often rely on automated labelling pipelines, which inevitably introduce mislabelled samples. For these reasons, we evaluate the detectors studied in this work on a subset of the CheXpert dataset, as described in the following section.

7.3.2 Experimental setup

Datasets As is common in multilabel classification settings [111], we frame our task as a binary classification problem (binary relevance) focused on a single pathology. Specifically, we aim to detect the presence of *pulmonary edema* in the chest radiographs. We select pulmonary edema because the original CheXpert authors designated this pathology as one of the official competition tasks, chosen for its clinical relevance and prevalence.

While the authors used images resized to 320×320 pixels, we further reduce the resolution to 128×128 pixels to lower computational costs. The greyscale images are normalized with z-normalization using the mean and standard deviation of the train set.

For our experiments, we sample 20,000 frontal-view samples from the train set by randomly selecting samples from the *positive*, *negative* and *uncertain* classes. This results in a smaller train set than the original, but retains the ratios of *positive*, *negative* and *uncertain* samples, of 26%, 68%, 6%, respectively. The original authors experimented with different ways to use the *uncertain* samples, and found that for pulmonary edema, the best strategy is to label these samples as *positive*. We do the same. In keeping with a realistic real-world evaluation scenario, we do not inspect the test set and therefore do not know how many samples are labelled positive or negative for pulmonary edema by the radiologists.

Models We employ the same VGG-16 architecture as in Chapter 5 and 6 for the classification task. All models are trained with cross-entropy loss, using the Adam optimizer [100] with the same batch size and learning rates as specified in Chapter 5 Table 5.1. Batch normalization layers [101] are applied after each hidden layer and before a ReLU activation function.

Scoring We compute sample scores $\mathbf{s} = (s_1, s_2, \dots, s_{|D_{\text{train}}|})^\top$ using all problematic sample detectors evaluated in this study, i.e., *sample forgetting**, *EL2N**, *CL*, *k-NN* and *VWFG*. To obtain the sample scores, results for all detectors are averaged across three randomly initialized models. All detector scores are pruned in descending order.

Pruning After assigning a score to each training sample, we apply Algorithm 1 to generate a ψ -curve for every detector Λ . At each point ψ_i along the curve, the model performance is evaluated over five random initializations. However, unlike previous results, we do not evaluate the model performance with accuracy since the test set might be unbalanced. Instead, we use the AUROC metric as used by others [19], [112]. These evaluations are performed using the same architecture and learning process used for scoring. The evaluation models are trained for 50 epochs² or until interpolation, what ever comes first. We set the resolution parameter $r = 1,000$ and $\alpha = 0.6$ in Equation 3.6, meaning that we remove 1,000 samples at a time and prune 60% of the train set. In addition to the detectors, we also conduct pruning experiments using the *random* baseline, as described in Section 4.2.3.

7.3.3 Results

Figure 7.3 presents the results of all the problematic sample detectors on the CheXpert dataset. Due to the wide spread of AUROC values across model seeds at each point along the ψ -curves, we also provide the maximum and minimum values, across seeds, in separate sub-plots.

From the results in Figure 7.3, we can make the following observations:

1. The variance across the 5 initialization seeds, when using the AUROC metric, is higher than for the classification accuracy that was used previously in this study. This variance is enough of an issue to cause the evaluation AUROC values for the stronger detectors to drop well below some of the random baseline seeds (see the bottom plots).
2. The *k-NN distance* and *sample forgetting** detectors perform poorly, resulting in approximately the same performance as the *random* baseline.

²Preliminary experiments showed that 50 epochs are more than enough for the models to approach interpolation.

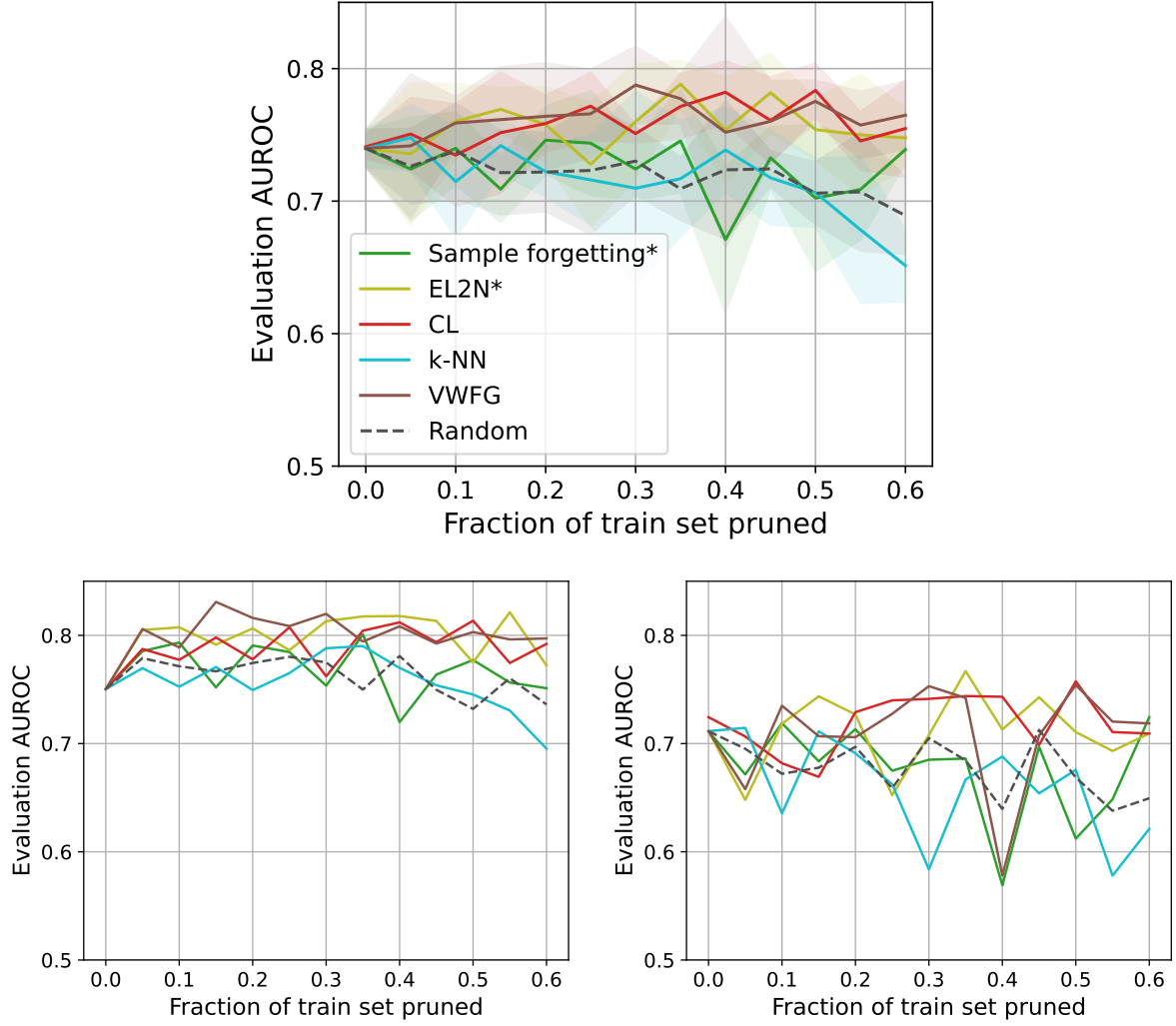


Figure 7.3: Model performance as a function of pruning an increasing portion of the CheXpert train set, according to the scores of all the problematic sample detectors. The *random* baseline is also provided. The top plot provides mean values (with standard deviations) across 5 seeds. The bottom plots show the corresponding maximum (left) and minimum (right) values at each point in the curves.

3. In general, the *EL2N**, *VWFG*, and *CL* detectors perform similarly well. They are clearly able to distinguish some problematic samples in the dataset. They reach a maximum performance increase around the 30% to 50% fraction point.
4. The *VWFG* detector outperforms the other detectors when the goal is to detect few (in number) highly problematic samples. We can see this by noticing that the *VWFG* detector reaches the highest evaluation AUROC performance in the

max evaluation curves (bottom-left) early in the pruning operation. This finding is consistent with the results in Figure 7.1.

5. However, if the goal is to maintain good (or slightly improved) model performance at higher pruning fractions without as much focus on detecting the most problematic samples, then the *EL2N** and *CL* detectors seem to produce more consistent results. Their minimum performance (bottom-right) does not drop as low later in pruning.
6. Due to computational constraints, we are unsure what would happen at even higher pruning fractions.

In summary, problematic samples can arise naturally in datasets, even without explicitly introducing noise. Moreover, most of the problematic sample detectors examined in this study (*EL2N**, *CL* and *VWFG*) are capable of identifying at least a subset of the problematic samples, even though all of these detectors assign scores to samples without directly accounting for the interdependencies that might exist between training samples. In the next section, we examine the issue of interdependent samples and explore whether such dependencies can be detected.

7.4 Sample interdependence

In Chapter 3 (Section 3.2 and 3.6) we mentioned that interdependencies between samples might be an issue when detecting and estimating the degree to which individual samples are problematic. We have indirectly observed the effects of such interdependencies:

- In Chapter 5 we observe that samples that are redundant early in the pruning experiment become important for generalization later (Section 5.5.1), and redundant samples become even more important in the presence of label noise (Section 5.5.2).
- In Chapter 6 we see that the size of the train set (at a constant noise rate) significantly affect the impact of problematic samples (Section 6.5.3).

-
- In this chapter we see that a method that is excellent at distinguishing particular problematic samples can do a poor job of distinguishing moderately problematic from useful samples (Section 7.2.4).

In this section, we examine this issue more systematically. Our goal is to clarify different ways in which sample interdependencies can be conceptualized, to outline methods for detecting them, and to evaluate how such dependencies may affect a problematic sample detector’s performance on our evaluation framework.

7.4.1 Intuition

We consider the interdependencies between training samples from two different perspectives.

Score-level interdependencies One perspective examines how the presence or absence of certain samples in the train set affects the scores assigned by problematic sample detectors to other samples. If a sample’s score changes significantly when specific other samples are removed, this suggests that its apparent “problematic” nature may depend on its interactions with those samples.

This has important implications for our evaluation framework. If a sample receives a high score primarily due to the presence of other samples, then removing those samples (e.g., through pruning) may eliminate the reasons it was problematic in the first place. Thus, score-level interdependencies can obscure a sample’s true contribution to model behaviour.

A brute-force approach to estimating these effects involves comparing detector outputs across many different dataset configurations. For a candidate set of samples, one could train models on the full set, then on modified subsets where one or more samples are removed, and observe how scores shift. This yields coarse estimates of how individual samples affects the group (and how groups affects individual samples) although the combinatorial cost makes this strategy impractical at scale.

Parameter-level interdependence A second way to conceptualize interdependence is to consider how groups of samples jointly affect the model’s parameter updates. Under this perspective, dependencies arise because samples may: 1) cooperate by pushing parameters in similar directions (e.g. through coherent gradients [80]), reinforcing a shared learning objective, or 2) compete by inducing conflicting gradients, thereby weakening or distorting each other’s influence.

This parameter-level notion focuses not on detector scores but on the underlying optimization dynamics. Influence functions as discussed in Section 2.5 provide a principled way to approximate the joint effects of groups of samples, and can help quantify cooperative or antagonistic relationships between them.

In the next section, we show how we can study:

1. Score-level interdependencies by making our existing evaluation framework iterative.
2. Parameter-level interdependencies by calculating the parameter influence of the training samples.

Specifically, we use the *score-level* perspective to evaluate how a problematic sample detector’s performance changes when evaluated on an iterative evaluation framework. Furthermore, we use the *parameter-level* perspective to visualize groups of interdependent samples and present a possible future evaluation framework to evaluate problematic sample detectors.

7.4.2 Methods

Iterative evaluation framework All data valuation methods that we are aware of compute a single set of scores for all training samples using the full train set, and then perform iterative pruning based solely on those initial scores. This procedure implicitly assumes that each sample’s score is relatively stable, i.e., not substantially affected by the presence or absence of other samples in the dataset.

To investigate the impact of this assumption and gage the effect of score-level interdependencies, we modify the standard pruning procedure. After each pruning step, we recompute the scores using the reduced dataset, and prune again according to these updated scores. This alternative pruning procedure is defined in Algorithm 2.

Algorithm 2: Generating a ψ -curve (iterative)

Input: $D_{\text{train}}, D_{\text{test}}, \Lambda, \mathcal{P}, \mathcal{A}, f, r, m$ as defined in Chapter 3.

Output: $\psi \in \mathbb{R}^m$

```

1  $\mathbf{s} \leftarrow \Lambda(D_{\text{train}}, f, \mathcal{A})$ 
2  $\mathbf{z} = (z_1, z_2, \dots, z_{|D_{\text{train}}|}) \leftarrow \text{argsort}(-\mathbf{s})$ 
3  $\psi \leftarrow (0, 0, \dots, 0) \in \mathbb{R}^m$ 
4  $\psi_1 \leftarrow \mathcal{P}(D_{\text{test}}, \mathcal{A}(D_{\text{train}}, f))$ 
5 for  $i \in \{1, 2, \dots, m-1\}$  do
6    $I_{\text{remove}} \leftarrow \{z_1, z_2, \dots, z_{(r \times i)}\}$ 
7    $D_{\text{trunc}} \leftarrow D_{\text{train}} \setminus \{(\mathbf{x}_j, y_j) : j \in I_{\text{remove}}\}$ 
8    $\psi_{i+1} \leftarrow \mathcal{P}(D_{\text{test}}, \mathcal{A}(D_{\text{trunc}}, f))$ 
9    $\mathbf{s} \leftarrow \Lambda(D_{\text{trunc}}, f, \mathcal{A})$ 
10   $\mathbf{z} = (z_1, z_2, \dots, z_{|D_{\text{trunc}}|}) \leftarrow \text{argsort}(-\mathbf{s})$ 
11 return  $\psi$ 

```

If the resulting ψ -curve produced by this iterative re-scoring procedure differs substantially from the original detector's ψ -curve (computed without re-scoring using Algorithm 1), then we conclude that meaningful score-level interdependencies exist. In particular, this indicates that a sample's assigned score is strongly affected by the presence of other samples in the train set.

Parameter influence As discussed in Section 2.5, the joint effect of a group of training samples on the model parameters can be approximated by summing their individual parameter influence vectors. Specifically, let $\mathbf{s}_i = -\frac{1}{n}\mathcal{I}(\mathbf{x}_i, y_i) \in \mathbb{R}^{|\phi|}$ denote the score vector (parameter influence) of sample $(\mathbf{x}_i, y_i)$, where n is the number of training samples, ϕ is the model parameters and the influence $\mathcal{I}(\mathbf{x}_i, y_i)$ is calculated according to Equation 2.1.

Stacking these score vectors $\mathbf{s}^\top$ yields a score matrix $\mathbf{S} \in \mathbb{R}^{n \times |\phi|}$ with each row representing the parameter influence of a single training sample. With the score matrix $\mathbf{S}$, we can do the following:

1. Sum a candidate set of rows in the matrix to obtain the *collective influence* of the training samples in that candidate set on the model parameters ϕ .
2. Calculate the cosine similarity between $\mathbf{s}_i$ and $\mathbf{s}_j$ to determine how the i -th and j -th sample cooperate or compete.

We evaluate both points in Section 7.4.4.

7.4.3 Experimental setup

Datasets We evaluate score-level interdependencies on two datasets: the multiclass synthetic dataset described in Section 4.2.1, and CIFAR-10 [18]. We keep the dataset sizes the same as reported in Chapter 5 Table 5.1. We add symmetric label noise to the train sets as outlined in Section 4.2.2.

Models For the synthetic- and CIFAR-10 dataset, we employ the same MLP and VGG-16 architectures as in Chapter 5. All models are trained using the Adam optimizer [100], with the same batch size and learning rates as specified in Chapter 5 Table 5.1. Batch normalization layers [101] are applied after each hidden layer and before a ReLU activation function.

Scoring Yang *et al.* [96] note if two samples have large gradient norms, they can cancel out if the gradients point in opposite directions. Thus we choose to evaluate the *EL2N* detector on the iterative evaluation framework, since the score approximates a sample’s gradient norm. Specifically, we compute sample scores $\mathbf{s} = (s_1, s_2, \dots, s_{|D_{\text{train}}|})^\top$ using the *EL2N* detector as defined in Equation 5.4, averaging results over ten randomly initialized models. For the *EL2N* detector, error vectors are recorded after 20 epochs of training, consistent with the experimental setup in Chapter 5. Note that every time the itera-

tive evaluation framework (Algorithm 2) removes a subset of r samples, the scores for the remaining samples are re-calculated, averaging results over ten randomly initialized models.

Pruning We apply Algorithm 1 and Algorithm 2 to generate a ψ -curve for the *EL2N* detector. We will denote the ψ -curve from Algorithm 1 as *EL2N** and the ψ -curve from Algorithm 2 as *EL2N!*. Note that in both cases samples are pruned in descending order of score. At each point ψ_i along the curve, the model performance is evaluated over three random initializations. These evaluations were performed using the same architecture and learning process used for scoring. We keep the resolution parameter r (Table 5.1) and $\alpha = 1$ consistent with the setup in Chapter 5. In addition to the two detectors, we also conduct pruning experiments using the *random* and *oracle* baselines, as described in Section 4.2.3.

Calculating parameter influence We study parameter influence by considering the CIFAR-10 dataset on the VGG-16 model as outlined above. Computing full-parameter influence is often computationally prohibitive. Following Yang *et al.* [96], we restrict our analysis to the parameters of the final linear layer—in our case, the weights connecting the last hidden layer to the output layer. For VGG-16 this corresponds to $\phi \in \mathbb{R}^{1,000 \times 10}$ since the last hidden layer has 1,000 nodes and the output layer has 10 nodes. This dramatically reduces computational cost while still capturing important directional influences on the model’s predictions.

As shown in Equation 2.1, a key computational step in influence calculation is evaluating the inverse Hessian–vector product $H_\phi^{-1}v = H_\phi^{-1}\nabla_\phi \ell(\mathbf{x}, y)$. Since we operate only on the last-layer parameters, we can compute this efficiently by solving the linear system $H_\phi x = v$ with respect to x rather than explicitly inverting the Hessian. We measure the parameter influence by considering the Hessian and sample gradients for a model trained on 20 epochs, therefore measuring at the same epoch as *EL2N*.

We implement these computations using PyTorch’s autograd capabilities with the open-source **pyDVL** [113] library, specifically the **DirectInfluence** class, to estimate $H_\phi^{-1}\nabla_\phi \ell(\mathbf{x}, y)$.

A regularization term of $\lambda = 0.1$ is added to the Hessian to ensure numerical stability.

7.4.4 Results

The ψ -curves for the *EL2N* detector evaluated on Algorithm 1 (*EL2N**) and on Algorithm 2 (*EL2N!*) are shown in Figure 7.4.

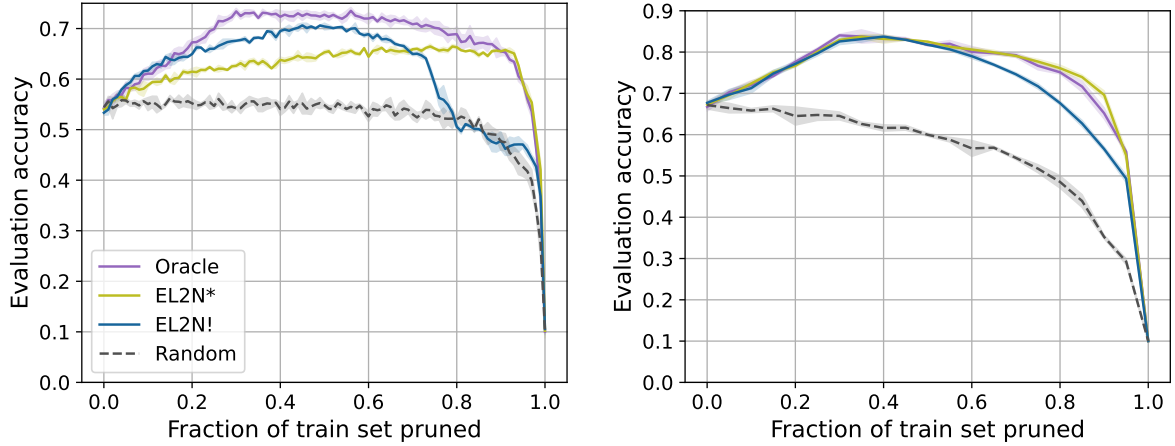


Figure 7.4: Model performance as a function of pruning an increasing portion of the train set with $\eta = 0.3$ symmetric label noise added to the synthetic (left) and CIFAR-10 (right) datasets. For *EL2N**, samples are pruned in descending order of scores provided by the *EL2N* method, and for *EL2N!* samples are pruned in descending order with re-scoring at each iteration. The *random* and *oracle* baselines are described in Section 4.2.3. The legend in the first plot applies to both plots.

For the synthetic dataset, the *EL2N!* ψ -curve begins to diverge from the *EL2N** ψ -curve early in pruning. In particular, *EL2N!* achieves higher evaluation accuracy during the early stages of pruning and tracks the *oracle* curve more closely. It is clear that in this case, there are some strong score-level dependencies between problematic samples in the train set and those that are removed early in the pruning experiment; there are samples for which the degree to which they are problematic was obscured by the presence of others and only when the latter were pruned did it become detectable. However, beyond the 50% pruning point, the *EL2N!* curve declines sharply. Our hypothesis is that by this stage, nearly all noisy samples have been removed; recomputing scores on what is

now predominantly clean data assigns higher scores to some informative clean samples. Pruning these samples disproportionately harms generalization.

Interestingly, the CIFAR-10 results show negligible benefit from iterative re-computation. This suggests an overall lower degree of score-level dependencies in the CIFAR-10 dataset. After the 50% pruning mark, however, the $EL2N!$ and $EL2N^*$ ψ -curves begin to diverge. Because $EL2N^*$ maintains high evaluation accuracy for longer, we infer that clean samples with lower $EL2N$ scores (calculated on the full train set) are more beneficial for generalization. Yet when scores are recomputed only on clean data, these same samples tend to receive higher $EL2N$ scores—leading $EL2N!$ to prune them prematurely and causing accuracy to drop more rapidly. This observation aligns with Paul *et al.* [29], who argue that when scores are computed on clean data, pruning should proceed in ascending order. Taken together, these results provide evidence of some sample interdependencies in CIFAR-10: the presence of problematic samples allows low score $EL2N$ samples to be beneficial for generalization, however, in the absence of problematic samples these low score $EL2N$ samples are less beneficial for generalization.

Having established the presence of sample interdependencies, we aim to visualize these interactions by examining how training samples influence the parameters of the final linear layer of the VGG-16 model trained on CIFAR-10 with label noise. After computing the parameter influence matrix $\mathbf{S}$, we select six reference samples: the first three clean samples and the first three label-corrupted samples in the dataset. For each reference sample, we compute the cosine similarity between its influence vector $\mathbf{s}_i$ and the influence vector of every other sample in the train set. Finally, we plot the samples with the highest and lowest cosine similarity to each reference example. These visualizations are shown in Figure 7.5.

Intuitively, the parameter influence vector captures how a sample “pushes” the model parameters during training. Two samples with high cosine similarity exert parameter influence in similar directions and therefore push the model towards a shared objective. In contrast, samples with very low or negative similarity push the parameters in opposing directions, indicating conflicting objectives.

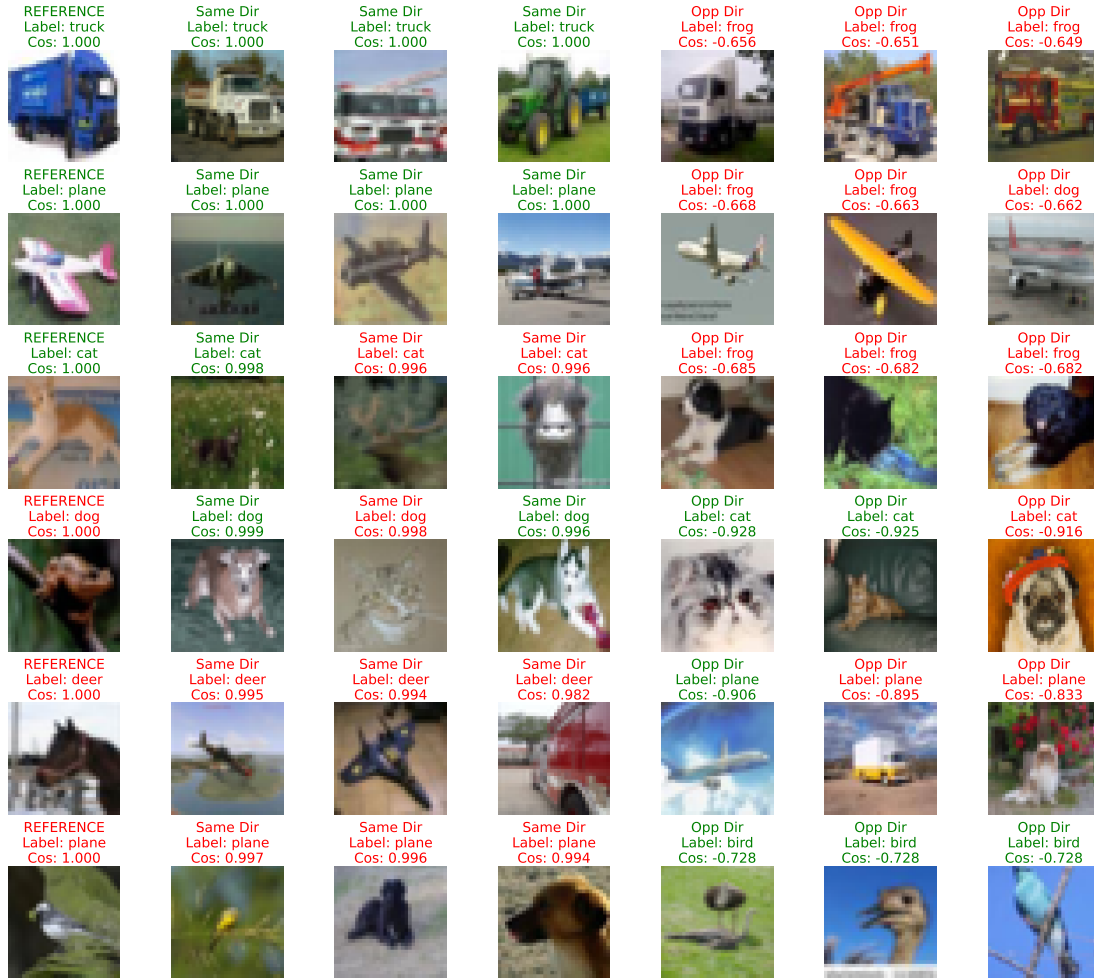


Figure 7.5: Interdependent training samples identified through influence-based cosine similarity. The leftmost column displays six reference samples (top three clean, bottom three noisy). For each reference sample, the next three columns show the training samples whose parameter influence vectors have the highest cosine similarity with the reference, while the final three columns show the samples with the lowest cosine similarity. The images with information shown in red have corrupted labels.

The first three rows of Figure 7.5 correspond to clean reference samples: a *truck*, a *plane*, and a *cat*. For the *truck* and *plane* samples, the images with the highest cosine similarity are also clean examples of *trucks* and *planes*, respectively. This indicates strong cooperative parameter-level interdependencies among samples of the same class. Interestingly, the samples with the lowest cosine similarity to these clean references are also visually similar (*trucks* and *planes*) but are label-corrupted. These samples push the

model parameters in directions that conflict with the clean samples’ objectives, consistent with their incorrect labels. The clean *cat* reference sample behaves slightly different: its highest-similarity samples include two images corrupted to the *cat* label, but these show weaker similarity compared to the clean *truck* and *plane* cases—likely reflecting greater intra-class variability in the *cat* class.

The bottom three rows of Figure 7.5 analyse the corrupted reference samples. For each corrupted reference, the samples with the highest cosine similarity are also corrupted to the same incorrect label. This illustrates how parameter-level interdependencies exists between corrupted labels, since these samples reinforce one another by pushing parameters coherently in the same direction. Conversely, the samples with the lowest cosine similarity to the noisy reference samples, all share the same label, regardless of whether those samples are corrupted or clean. In Appendix A.4 we perform a complementary experiment using principle component analysis to observe these parameter-level interdependencies at a higher level.

These results visually illustrate the parameter-level interdependencies that exists among training samples. Clean samples form cooperative clusters that align strongly in parameter space, whereas corrupted samples form antagonistic clusters that push in opposite directions. Moreover, the clean–corrupted interdependencies reveal that label noise not only misleads the model individually but also distorts the influence structure of the entire dataset. Specifically, a corrupted sample conflicts most strongly with the very samples it should agree with (top three rows of Figure 7.5), and it cooperates most strongly with other corrupted samples (bottom 3 rows of Figure 7.5)—even when they share only a mistaken label.

This sheds light on why we often see sharply declining performance in the ψ -curves once many problematic samples are efficiently pruned (see Figures 4.4, 4.7c, 6.3b, and 7.1.). Samples that are very problematic in the full train set (e.g. label corrupted) inflate the degree to which others appear problematic by counter-acting their parameter influence (recall that most detectors are based on estimating the difficulty with which samples are fitted). Once the highly problematic samples are removed, and no re-scoring is done, the

affected samples are pruned next even though they were far from problematic. We expect a method that is better at distinguishing these highly problematic samples would result in more extreme drops in performance later. From the results in Figure 7.4, we see that simply recalculating does not necessarily prevent this issue.

We conclude this section by examining how the collective influence of the train set evolves as samples are removed during pruning. Following Yang *et al.* [96] and Koh *et al.* [95], we estimate collective parameter influence of removing a subset of training samples as the sum of the individual parameter influence vectors for the samples in that subset. Specifically, building on Equation 2.2, the *collective influence* of a subset D_{trunc} on model parameters ϕ is:

$$\phi_{-D_{\text{trunc}}} - \phi \approx \sum_{(\mathbf{x}_i, y_i) \in D_{\text{trunc}}} \left(-\frac{1}{n} \mathcal{I}(\mathbf{x}_i, y_i) \right) = \sum_{i \in D_{\text{trunc}}} \mathbf{s}_i. \quad (7.4)$$

To obtain a scalar measure of the magnitude of this collective influence, we follow Yang *et al.* and take the L_2 -norm of the resulting vector: $\|\sum_{i \in D_{\text{trunc}}} \mathbf{s}_i\|_2$, where D_{trunc} is the reduced dataset after pruning. Figure 7.6 (left) shows how this quantity changes as larger portions of the dataset are removed. Intuitively, these collective parameter influence curves estimate the extent to which the samples remaining in the train set are coherently pointing in a similar direction in the parameter space at each point along the ψ -curve. For comparison, we show the CIFAR-10 pruning results under symmetric label noise (Figure 7.1, bottom left) alongside it (right).

The collective influence curves in Figure 7.6 exhibit a trend similar to that of the evaluation framework ψ -curves: for the *oracle*, *sample forgetting**, *EL2N**, and *VWFG* detectors, the collective influence initially increases as problematic samples are removed, and then eventually decreases once clean samples begin to be pruned. This parallel suggests that collective influence could serve as an alternative evaluation framework for understanding the impact of pruning strategies in future work. In principle, one could simulate the effect of removing arbitrary subsets of the train set by measuring their collective parameter influence—without the computational cost of retraining evaluation models.

However, a closer comparison of the two plots reveals subtle and informative discrepan-

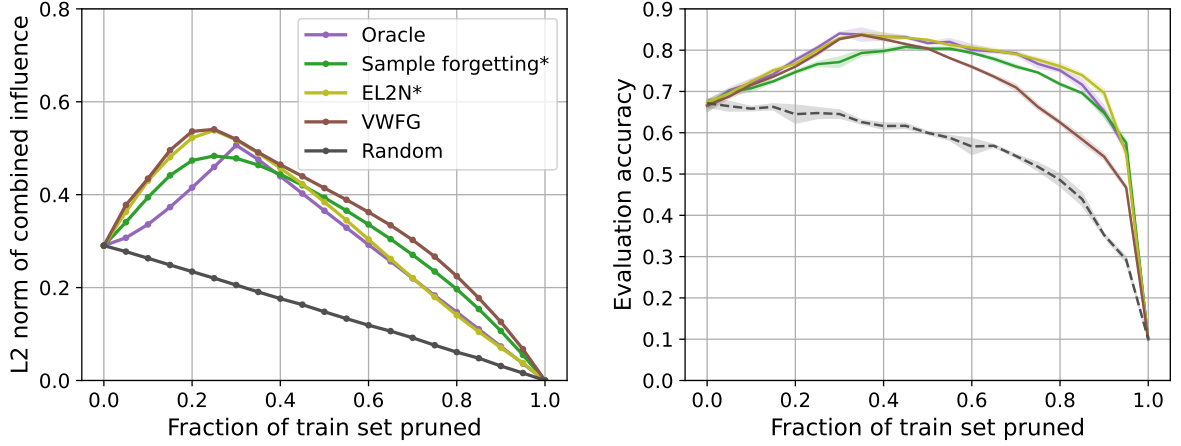


Figure 7.6: Collective parameter influence (left) and evaluation accuracy (right) as a function of pruning an increasing portion of the train set with $\eta = 0.3$ symmetric label noise added to the and CIFAR-10 dataset.

cies. Up to roughly the 30% pruning mark, the *oracle*, *EL2N**, and *VWFG* detectors achieve nearly identical evaluation accuracy, yet their collective influence values differ meaningfully. Specifically, the *EL2N** and *VWFG* detectors produce subsets with higher collective influence than the *oracle* baseline, even though all three attain similar generalization performance. This indicates that, when a detector already effectively separates problematic from clean samples, optimizing for maximum collective influence in the early stages of pruning may not translate into improved evaluation accuracy.

A similar pattern emerges in the later stages of pruning (beyond 30%). The *VWFG* detector maintains the highest collective influence for the longest duration. Nonetheless, the *EL2N** detector and *oracle* baseline preserve evaluation accuracy for longer, even though their retained subsets exhibit lower collective influence. Once again, this suggests that maximizing collective influence does not necessarily correspond to maximizing generalization performance when the detector already distinguishes noisy from clean samples.

Together, these results highlight a nuanced relationship between parameter influence and generalization. While collective influence provides a meaningful global summary of how subsets of data steer model parameters, it does not perfectly align with evaluation accuracy—especially when the detector already identifies problematic samples well. This

indicates that influence-based metrics capture structural properties of the dataset that are not fully reflected in standard pruning curves.

7.5 Discussion

This chapter presented three key aspects of our work that point toward promising avenues for future improvement. Specifically, we 1) examined the limitations of learning-centric methods and introduced a new problematic sample detector that aims to address these shortcomings, 2) evaluated all detectors on a real-world dataset without adding explicit noise, and 3) investigated how interdependencies arise between training samples and how these interdependencies affect our evaluation framework. The main findings are summarized below.

On the *VWFG* detector:

1. The strength of the *VWFG* detector lies in its dynamic scoring mechanism. By integrating information across the entire training trajectory and reweighing it at each epoch according to its importance, *VWFG* effectively identifies label-noise instances in the train set.
2. However, because *VWFG* places greater weight on samples that change early in training, it tends to assign higher scores to hard-to-fit clean samples—those that contribute to generalization—than to redundant clean samples. As a result, its performance declines sharply in our evaluation framework once the problematic subset has been removed.

On the real-world analysis:

1. The performance of the detectors suggests that problematic samples can emerge naturally in real-world datasets, even when no explicit noise is added to the train set.

-
2. The *k*-NN distance and *sample forgetting** detectors performs no better than the *random* baseline on the CheXpert dataset. However, the *EL2N**, *CL* and *VWFG* detectors results in improved evaluation performance at some point during pruning.

On interdependent samples:

1. We conceptualize interdependencies among training samples in two ways: a) the extent to which the presence or absence of certain samples affects the scores produced by a detector (score-level interdependencies), and b) how groups of samples jointly affect the model’s parameters, either by reinforcing one another or by exerting conflicting influences (parameter-level interdependencies).
2. By recomputing detector scores each time a subset of samples is removed, we demonstrated the existence of score-level interdependencies. These dependencies had a substantial impact on performance in the synthetic dataset, though a more moderate effect in CIFAR-10.
3. Using parameter influence calculations for CIFAR-10, we visualized parameter-level interdependencies, showing concrete examples of how groups of samples collaborate or oppose one another in determining the solution found by the optimization process.
4. We further showed that the collective influence of sample subsets reflects, to some extent, the behaviour of our evaluation framework. This finding highlights the potential of parameter-level influence metrics as a future evaluation method that could reduce computational overhead.

The insights gained from this chapter indicate that, if one must choose a single problematic sample detector, *VWFG* is a strong candidate. Not only does it reliably assign the highest scores to noisy samples, but it also achieved the highest evaluation performance value on the real-world dataset—likely due to its dynamic scoring nature. Its behaviour is also partially interpretable: it prioritizes noisy samples first, followed by hard-to-fit clean samples, which explains the rapid performance drop once these groups are removed.

However, if the goal is to maintain stable performance across higher pruning fractions, then the $EL2N^*$ and CL detectors might be a better choice.

The collective parameter influence results also suggest the possibility of developing a new problematic sample detector that explicitly accounts for the collective influence of groups of samples. However, our analysis shows that simply maximizing collective influence does not guarantee improved evaluation accuracy.

In the next chapter, we conclude the study with a summary of the main findings, a discussion of limitations, and potential directions for future research.

Chapter 8

Conclusion

We conclude the study by reviewing the key findings, discussing the limitations, and proposing future work for our research.

8.1 Overview

In this chapter, we summarize our key findings in Section 8.2 and discuss our main contributions in Section 8.3. In Section 8.4 we recapitulate the limitations of our study. Thereafter, we propose possible topics for future work in Section 8.5 and provide final remarks in Section 8.6.

8.2 Key findings

The main goal of the present research is to evaluate problematic sample detectors to gain a better understanding of problematic samples in deep learning and their detection. In pursuit of this goal, many observations were made in the chapters of this dissertation. We

summarize the main take-aways here:

1. There exists various methods across the fields of *data valuation*, *label noise detection* and *out-of-distribution (OOD) detection* that can be adapted as problematic sample detectors as outlined in Chapter 2.
2. The findings from Chapter 2 revealed that there exists numerous inconsistencies across literature when data pruning experiments are performed (as discussed in Section 2.6). We consolidate a formal evaluation framework that can be used to evaluate problematic sample detectors in Chapter 3.
3. With controlled experiments on synthetically generated data, we show that our evaluation framework can be used to indicate the presence of problematic samples in Chapter 4.
4. Additionally, in Chapter 4, we confirm that the three types of OOD samples that we define in Chapter 3, can be used to induce problematic samples in the train set to evaluate their effect on model generalization. Surprisingly, input-noise (samples that underwent a covariate- or semantic shift) can also constitute problematic samples, an aspect often overlooked in literature.
5. We show that the *sample forgetting* and *Error L_2 -norm (EL2N)* methods from the data valuation field can be used as problematic sample detectors on image classification datasets in Chapter 5. However, these methods only work as problematic sample detectors if we reverse the ranking order (opposite to the order proposed by the original authors [26], [29]).
6. We show that the *Confident Learning (CL)* and *k-nearest-neighbours (k-NN) distance* methods from the label noise detection and OOD detection fields, can be adapted for problematic sample detectors in Chapter 6. The results show that the *CL* framework serves as a versatile detector across various types of OOD samples and datasets, while the performance of *k-NN* varies across these settings.

-
7. We show that the strengths of *sample forgetting* and *EL2N* can be combined to develop a new, highly effective problematic sample detector that we call the *Variance-Weighted Fitness Gap (VWFG)* detector.
 8. We show problematic samples can occur naturally in a real-world dataset and that the *VWFG*, *EL2N** and *CL* detectors performed well on this task.
 9. Lastly, we evaluate the effect of sample interdependencies and show that score-level and parameter-level interdependencies exists between training samples. These findings are consolidated in Chapter 7.

We refer the reader to the discussion Sections (2.6, 3.6, 4.5, 5.6, 6.6 and 7.5) of each chapter for a more detailed discussion on the findings of our work.

8.3 Contributions

In this section we clarify the contributions of our study. We do this by: directly addressing the research questions outlined in Section 1.3 and presenting a discussion on the implications of our work.

8.3.1 Addressing research questions

In Chapter 1 we defined three research questions for which answers would contribute to the overall goal of our study. We answer them here, given our findings:

1. *How can we systematically evaluate and compare different methods for detecting problematic samples within a dataset?*
 - We can systematically compare and evaluate problematic sample detectors with our data pruning evaluation framework. Specifically, using Algorithm 1 and the points outlined in Section 3.4.

-
- We find that, in the absence of ground truths, this approach provides a nuanced comparison between problematic sample detectors that provides more information than a point-estimate of detector “goodness”.
2. *To what extent are existing methods effective for detecting problematic samples as defined in this study?*
- The learning-centric detectors, i.e., *Sample forgetting* and *EL2N* are effective problematic sample detectors (if their ranking order is reversed), but leaves room for improvement due to their individual limitations and heuristics as described in Section 7.2.1.
 - The data-centric detectors, i.e., *CL* and *k-NN distance* are effective to some extent. Specifically, the *CL* detector performs well across all settings, however struggles the most with asymmetric label noise, while the *k-NN distance* detector performance is highly dependent on the type of problematic sample and dataset characteristics, and can completely fail in some settings.
3. *Can current methods be improved to more accurately identify problematic samples?*
- Yes, the strengths of the *sample forgetting* and *EL2N* detectors can be combined to form the *VWFG* detector that performs well across both symmetric and asymmetric label noise (where *CL* struggled) and outperform the learning-centric detectors on the synthetically generated dataset where they struggled the most.

8.3.2 Implications

The findings of our work has several implications:

- **Standardized evaluation for data pruning.** By formalizing a data pruning evaluation framework, we aim to provide a standardized methodology that enables more consistent and comparable dataset pruning studies, including those focused on detecting problematic samples.

-
- **Importance of distributional quality, not just quantity.** Our results show that distributional shifts in both the input space X and the label space Y can give rise to problematic samples. This highlights the need for practitioners to attend not only to dataset size but also to dataset *quality* when developing ML systems, and to apply data augmentation cautiously rather than assuming it always improves robustness.
 - **Practical guidance for pruning based on learning-centric scores.** We find that pruning samples in descending order of *sample forgetting* or *EL2N* scores is generally safer and more effective: 1) it introduces only minor performance deviations at low pruning levels; 2) it outperforms ascending-order pruning when pruning aggressively; and 3) it successfully identifies problematic samples and can improve performance even in the absence of explicit added noise.
 - **Broader utility of the CL detector** Our study shows that the *CL* detector can detect semantic- and covariate-shifts—not just label noise, its original target. This suggests that practitioners can use it (and by extension the `cleanlab` package) for dataset cleaning even when label noise is not believed to be present.

8.4 Limitations

Here we delineate the core limitations of our study:

- **Hyperparameter tuning.** Because our goal was to evaluate problematic-sample detectors rather than to achieve optimal model performance, we fixed a reasonable set of hyperparameters and did not explore how different hyperparameter choices might affect our results. Consequently, the robustness of our findings across broader hyperparameter settings remains unexamined. Additionally, hyperparameter tuning is often impracticable for pruning experiment considering the number of models that are trained to generate each ψ -curve. This is a limitation of both our study and the evaluation framework in general.

-
- **Detection and evaluation architectures.** On a related note, throughout the study, we kept the model architectures that are used to generate the scores and those that are evaluated using the pruning framework the same. This was done to ensure a fair comparison between problematic sample detectors, but it might be the case that some detectors perform significantly better with specific model architectures; an option that we did not explore.
 - **Limited model and dataset scale.** Our experiments focused primarily on MNIST and CIFAR-10 using MLPs and VGG-16, respectively. It is therefore unclear how well our observations generalize to larger-scale datasets and modern architectures such as ImageNet and residual networks.
 - **Noise rates.** In almost all experiments involving artificially added noise we chose to use a noise rate of 30%. This is a fairly aggressive noise rate and we expect the observed trends to vary across different noise rates as demonstrated on synthetic data in Chapter 4.
 - **Representational depth.** In Chapter 6, detector scores were computed using embeddings from the output layer and the penultimate hidden layer of VGG-16. While this aligns with common practice in the literature [56], [74], we did not examine how detector behaviour changes when features are extracted from earlier layers with different levels of abstraction. It therefore remains unclear to what extent our findings depend on representational depth. Evaluating detector scores across intermediate layers could help assess the stability of the observed trends across different levels of feature hierarchy.

8.5 Future work

We propose several directions for future research:

- **Dynamic evaluation framework.** In this work, we introduced a dynamic scoring method (VWFG). But what about a *dynamic evaluation framework*? Our eval-

uation framework in Algorithm 1 and other existing data pruning works remove a fixed number of samples at each iteration of pruning. However, the number of samples to remove at each iteration can instead be determined dynamically. As mentioned in Section 3.6, in some settings a small number of highly problematic samples may have disproportionately large effects, while the majority of samples remain non-problematic. A possible solution to overcome this limitation of our evaluation framework is to consider the *scores* of the samples when determining how many samples to prune at each iteration. For example, we can prune a subset of samples with variable size by considering the number of samples that will result in a fixed contribution to the total score. This will result in a ψ -curve that places more emphasis on the problematic samples, (i.e., those with large scores that will be pruned in smaller subsets) while less emphasis will be placed on the large number of non-problematic samples.

- **Distinguishing clean hard-to-fit from problematic samples.** Distinguishing problematic, hard-to-fit but clean and easy-to-fit but clean samples remains an open challenge since the underlying intuition for almost all of the approaches that we considered is based on the difficulty with which the model fits each sample.
- **Recalibrating *CL* thresholds for input-noise detection.** Future work could examine the differences in the out-of-sample predicted probabilities for label noise and input noise samples, to better understand why the *CL* detector is more effective at detecting label noise. If these differences can be characterized, new strategies for calculating the per class thresholds could be developed to improve the *CL* framework’s sensitivity to input noise.
- **Parameter influence as a detector.** Future research may explore how the collective parameter influence of a group of samples can be leveraged to reweight detector scores.
- **Parameter influence as an evaluation framework.** As discussed in Section 7.4, collective parameter influence could also serve as an evaluation framework—simulating how detectors would behave on various sample subsets without requiring

iterative retraining. This would allow for more efficient evaluation of pruning strategies at scale.

- **Stability of problematic sample detection under training stochasticity.**

An important open question is the extent to which problematic sample detection is stable under sources of training stochasticity. For example, when training the same architecture on the same dataset and in the same order, but with different initialisation seeds, do the same samples consistently receive high problematic scores? Alternatively, with a fixed initialisation seed, does variation in the order with which data is presented to the model during training affect which samples are identified as problematic? Investigating the sensitivity of detector outputs to initialisation and data-ordering effects would provide insight into whether problematic samples reflect intrinsic data properties or artifacts of specific training trajectories.

8.6 Final remark

We hope that our findings provide insights about how practitioners should think about training data in deep learning—not as a homogeneous collection, but as a structured collection of samples that can hinder or enhance learning. Only by understanding how problematic samples emerge and influence training dynamics can we equip ourselves with the tools needed to rethink data quality and generalization in modern deep learning.

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Reference Numbers: PSTD23042898868, RCDL240215206999).

Bibliography

- [1] K. He, X. Zhang, S. Ren, *et al.*, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 770–778.
- [2] T. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2020, pp. 1877–1901.
- [3] J. Jumper, R. Evans, A. Pritzel, *et al.*, “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.
- [4] B. Lim, S. Arık, N. Loeff, *et al.*, “Temporal fusion transformers for interpretable multi-horizon time series forecasting,” *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.
- [5] D. Soudry, E. Hoffer, M. S. Nacson, *et al.*, “The implicit bias of gradient descent on separable data,” *Journal of Machine Learning Research*, vol. 19, no. 70, pp. 1–57, 2018.
- [6] A. C. Wilson, R. Roelofs, M. Stern, *et al.*, “The marginal value of adaptive gradient methods in machine learning,” in *Advances in Neural Information Processing Systems*, Long Beach, USA, 2017.
- [7] M. Belkin, D. Hsu, S. Ma, *et al.*, “Reconciling modern machine-learning practice and the classical bias–variance trade-off,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019.

-
- [8] B. Neyshabur, Z. Li, Y. LeCun, *et al.*, “Towards understanding the role of over-parametrization in generalization of neural networks,” *arXiv preprint*, 2018. arXiv: 1805.12076 [cs.LG].
 - [9] V. N. Vapnik, “An overview of statistical learning theory,” *Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.
 - [10] S. Hochreiter and J. Schmidhuber, “Flat minima,” *Neural Computation*, vol. 9, no. 1, pp. 1–42, 1997.
 - [11] N. S. Keskar, D. Mudigere, J. Nocedal, *et al.*, “On large-batch training for deep learning: Generalization gap and sharp minima,” in *International Conference on Learning Representations*, Toulon, France, 2017.
 - [12] L. Sagun, U. Evci, V. U. Guney, *et al.*, “Empirical analysis of the hessian of over-parametrized neural networks,” *arXiv preprint*, 2017. arXiv: 1706.04454 [cs.LG].
 - [13] G. K. Dziugaite and D. M. Roy, “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data,” *arXiv preprint*, 2017. arXiv: 1703.11008 [cs.LG].
 - [14] B. Neyshabur, S. Bhojanapalli, D. McAllester, *et al.*, “Exploring generalization in deep learning,” in *Advances in Neural Information Processing Systems*, Long Beach, USA, 2017.
 - [15] P. Chaudhari, A. Choromanska, Y. LeCun, *et al.*, “Entropy-SGD: Biasing gradient descent into wide valleys,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124018, 2019.
 - [16] L. Dinh, R. Pascanu, S. Bengio, *et al.*, “Sharp minima can generalize for deep nets,” in *International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 1019–1028.
 - [17] Y. LeCun, C. Cortes, and C. J. C. Burges, *MNIST database of handwritten digits*, <http://yann.lecun.com/exdb/mnist/>.
 - [18] A. Krizhevsky, V. Nair, and G. Hinton, *CIFAR10 dataset*, <https://www.cs.toronto.edu/~kriz/cifar.html>.

-
- [19] J. Vin, P. Rajpurkar, M. Ko, *et al.*, “Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison,” in *Association for the Advancement of Artificial Intelligence*, Honolulu, Hawaii, 2019, pp. 590–597.
- [20] L. Freese and M. W. Theunissen, “Data pruning: Redundant, problematic, and interdependent samples,” in *Artificial Intelligence Research*, Cape Town, South Africa, 2025, pp. 193–205.
- [21] X. Zhai, A. Kolesnikov, N. Houlsby, *et al.*, “Scaling vision transformers,” in *IEEE Conference on Computer Vision and Pattern Recognition*, New Orleans, USA, 2022, pp. 12 104–12 113.
- [22] J. Deng, W. Dong, R. Socher, *et al.*, “ImageNet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, USA, 2009, pp. 248–255.
- [23] J. Hestness, S. Narang, N. Ardalani, *et al.*, “Deep learning scaling is predictable, empirically,” *arXiv preprint*, 2017. arXiv: 1712.00409 [cs.LG].
- [24] B. Sorscher, R. Geirhos, S. Shekhar, *et al.*, “Beyond neural scaling laws: Beating power law scaling via data pruning,” in *Advances in Neural Information Processing Systems*, New Orleans, USA, 2022, pp. 19 523–19 536.
- [25] A. Ghorbani and J. Zou, “Data Shapley: Equitable valuation of data for machine learning,” in *International Conference on Machine Learning*, Long Beach, USA, 2019, pp. 2242–2251.
- [26] M. Toneva, A. Sordoni, R. Combes, *et al.*, “An empirical study of example forgetting during deep neural network learning,” in *International Conference on Learning Representations*, New Orleans, USA, 2019, pp. 1768–1785.
- [27] R. Jia, D. Dao, B. Wang, *et al.*, “Towards efficient data valuation based on the shapley value,” in *International Conference on Artificial Intelligence and Statistics*, Naha, Japan, 2019, pp. 1167–1176.
- [28] V. Feldman and C. Zhang, “What neural networks memorize and why: Discovering the long tail via influence estimation,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2020, pp. 2881–2891.
-

-
- [29] M. Paul, S. Ganguli, and G. Dziugaite, “Deep learning on a data diet: Finding important examples early in training,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2021, pp. 20 596–20 607.
- [30] J. Yoon, S. Arik, and T. Pfister, “Data valuation using reinforcement learning,” in *International Conference on Machine Learning*, Vienna, Austria, 2020, pp. 10 842–10 851.
- [31] N. Ki, H. Choi, and C. H. W., “Data valuation without training of a model,” in *International Conference on Learning Representations*, Kigali, Rwanda, 2023.
- [32] V. Feldman, “Does learning require memorization? A short tale about a long tail,” in *ACM SIGACT Symposium on Theory of Computing*, Virtual Conference, 2020, pp. 954–959.
- [33] C. Zhang, S. Bengio, M. Hardt, *et al.*, “Understanding deep learning requires rethinking generalization,” in *International Conference on Learning Representation*, Toulon, France, 2017.
- [34] Y. Kwon and J. Zou, “Beta Shapley: A unified and noise-reduced data valuation framework for machine learning,” in *International Conference on Artificial Intelligence and Statistics*, Virtual Conference, 2022, pp. 8780–8802.
- [35] X. Xie, J. Mao, Y. Liu, *et al.*, “Improving web image search with contextual information,” in *International Conference on Information and Knowledge Management*, Beijing, China, 2019, pp. 1683–1692.
- [36] X. Yu, T. Liu, M. Gong, *et al.*, “Learning with biased complementary labels,” in *European Conference on Computer Vision*, Munich, Germany, 2018, pp. 68–83.
- [37] N. Sambasivan, S. Kapania, H. Highfill, *et al.*, “Everyone wants to do the model work, not the data work: Data cascades in high-stakes AI,” in *Conference on Human Factors in Computing Systems*, Virtual Conference, 2021, pp. 1–15.
- [38] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, “Learning with noisy labels,” in *Advances in Neural Information Processing Systems*, Lake Tahoe, USA, 2013.
-

-
- [39] G. Patrini, A. Rozza, K. M., *et al.*, “Making deep neural networks robust to label noise: A loss correction approach,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, 2017, pp. 1944–1952.
- [40] J. Li, R. Socher, and S. Hoi, “Dividemix: Learning with noisy labels as semi-supervised learning,” in *International Conference on Learning Representations*, Virtual Conference, 2020.
- [41] C. Sun, A. Shrivastava, S. Singh, *et al.*, “Revisiting unreasonable effectiveness of data in deep learning era,” in *IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 843–852.
- [42] D. Mahajan, R. Girshick, V. Ramanathan, *et al.*, “Exploring the limits of weakly supervised pretraining,” in *European Conference on Computer Vision*, Munich, Germany, 2018, pp. 181–196.
- [43] D. Rolnick, A. Veit, S. Belongie, *et al.*, “Deep learning is robust to massive label noise,” *arXiv preprint*, 2017. arXiv: 1705.10694 [cs.LG].
- [44] M. M. Charbonneau, *Peer review of “Deep learning is robust to massive label noise”*, OpenReview, Accessed: 2024-07-22, 2024. [Online]. Available: <https://openreview.net/forum?id=B1p461b0W¬eId=rkfFCmgzz>.
- [45] J. Kremer, F. Sha, and C. Igel, “Robust active label correction,” in *International Conference on Artificial Intelligence and Statistics*, Lanzarote, Canary Islands, 2018, pp. 308–316.
- [46] D. Arpit, S. Jastrzebski, N. Ballas, *et al.*, “A closer look at memorization in deep networks,” in *International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 233–242.
- [47] J. Huang, L. Qu, R. Jia, *et al.*, “O2U-Net: A simple noisy label detection approach for deep neural networks,” in *IEEE/CVF International Conference on Computer Vision*, Seoul, South Korea, 2019, pp. 3326–3334.
- [48] J. Goldberger and E. Ben-Reuven, “Training deep neural-networks using a noise adaptation layer,” in *International Conference on Learning Representations*, Virtual Conference, 2022.
-

-
- [49] S. Reed, H. Lee, D. Anguelov, *et al.*, “Deep learning is robust to massive label noise,” *arXiv preprint*, 2014. arXiv: 1412.6596 [cs.LG].
 - [50] T. Xiao, T. Xia, Y. Yang, *et al.*, “Learning from massive noisy labeled data for image classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, USA, 2015, pp. 2691–2699.
 - [51] B. Han, Q. Yao, X. Yu, *et al.*, “Co-teaching: Robust training of deep neural networks with extremely noisy labels,” in *Advances in Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 8535–8545.
 - [52] L. Jiang, Z. Zhou, T. Leung, *et al.*, “Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels,” in *International Conference on Machine Learning*, Stockholm, Sweden, 2018, pp. 2304–2313.
 - [53] S. Guo, W. Huang, H. Zhang, *et al.*, “Curriculumnet: Weakly supervised learning from large-scale web images,” in *European Conference on Computer Vision*, Munich, Germany, 2018, pp. 135–150.
 - [54] Z. Zhu, Z. Dong, and Y. Liu, “Detecting corrupted labels without training a model to predict,” in *International Conference on Machine Learning*, Baltimore, USA, 2022, pp. 27 412–27 427.
 - [55] Y. Liu, “Understanding instance-level label noise: Disparate impacts and treatments,” in *International Conference on Machine Learning*, Virtual Conference, 2021, pp. 6725–6735.
 - [56] C. G. Northcutt, L. Jiang, and I. L. Chuang, “Confident Learning: Estimating uncertainty in dataset labels,” *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373–1411, 2021.
 - [57] J. Kim, S. Yun, and H. O. Song, “Neural relation graph: A unified framework for identifying label noise and outlier data,” in *Advances in Neural Information Processing Systems*, New Orleans, USA, 2024.
 - [58] Z. Zhu, J. Wang, and Y. Liu, “Beyond images: Label noise transition matrix estimation for tasks with lower-quality features,” in *International Conference on Machine Learning*, Baltimore, USA, 2022, pp. 27 633–27 653.

-
- [59] P. Chen, B. B. Liao, G. Chen, *et al.*, “Understanding and utilizing deep neural networks trained with noisy labels,” in *International Conference on Machine Learning*, Long Beach, USA, 2019, pp. 1062–1070.
- [60] X. Xia, T. Liu, B. Han, *et al.*, “Part-dependent label noise: Towards instance-dependent label noise,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 7597–7610, 2020.
- [61] G. Pleiss, T. Zhang, E. Elenberg, *et al.*, “Identifying mislabeled data using the area under the margin ranking,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2020, pp. 17 044–17 056.
- [62] J. Yang, K. Zhou, Y. Li, *et al.*, “Generalized out-of-distribution detection: A survey,” *International Journal of Computer Vision*, vol. 132, no. 12, pp. 5635–5662, 2024.
- [63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, Lake Tahoe, USA, 2012.
- [64] K. He, X. Zhang, S. Ren, *et al.*, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in *IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 1026–1034.
- [65] I. Evtimov, K. Eykholt, E. Fernandes, *et al.*, “Robust physical-world attacks on machine learning models,” *arXiv preprint*, 2017. arXiv: 1707.08945 [cs.LG].
- [66] M. Sharif, S. Bhagavatula, L. Bauer, *et al.*, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, 2016, pp. 1528–1540.
- [67] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *arXiv preprint*, 2016. arXiv: 1610.02136 [cs.LG].
- [68] K. Lee, K. Lee, H. Lee, *et al.*, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” in *Advances in Neural Information Processing Systems*, Montréal, Canada, 2018.

-
- [69] J. Tack, S. Mo, J. Jeong, *et al.*, “CSI: Novelty detection via contrastive learning on distributionally shifted instances,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2020, pp. 11 839–11 852.
- [70] A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, USA, 2015, pp. 427–436.
- [71] W. Liu, X. Wang, J. Owens, *et al.*, “Energy-based out-of-distribution detection,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2020, pp. 21 464–21 475.
- [72] P. C. Mahalanobis, “On the generalized distance in statistics,” *Journal of The Asiatic Society of Bengal*, vol. 26, pp. 541–580, 1936.
- [73] Y. Sun, Y. Ming, X. Zhu, *et al.*, “Out-of-distribution detection with deep nearest neighbors,” in *International Conference on Machine Learning*, Baltimore, USA, 2022, pp. 20 827–20 840.
- [74] J. Kuan and J. Mueller, “Back to the basics: Revisiting out-of-distribution detection baselines,” in *International Conference on Machine Learning*, Baltimore, USA, 2022.
- [75] J. Park, Y. G. Jung, and A. B. J. Tech, “Nearest neighbor guidance for out-of-distribution detection,” in *IEEE/CVF International Conference on Computer Vision*, Paris, France, 2023, pp. 1686–1695.
- [76] D. Bahri, H. Jiang, and M. Gupta, “Deep k-nn for noisy labels,” in *International Conference on Machine Learning*, Vienna, Austria, 2020, pp. 540–550.
- [77] Q. Zhang, Q. Feng, J. T. Zhou, *et al.*, “The best of both worlds: On the dilemma of out-of-distribution detection,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 69 716–69 746, 2024.
- [78] H. Bai, G. Canal, X. Du, *et al.*, “Feed two birds with one scone: Exploiting wild data for both out-of-distribution generalization and detection,” in *International Conference on Machine Learning*, Honolulu, Hawaii, 2023, pp. 1454–1471.
-

-
- [79] N. Rahaman, A. Baratin, D. Arpit, *et al.*, “On the spectral bias of neural networks,” in *International Conference on Machine Learning*, Long Beach, USA, 2019, pp. 5301–5310.
- [80] S. Chatterjee, “Coherent gradients: An approach to understanding generalization in gradient descent-based optimization,” in *International Conference on Learning Representations*, Virtual Conference, 2020.
- [81] X. Yu, B. Han, J. Yao, *et al.*, “How does disagreement help generalization against label corruption?” In *International Conference on Machine Learning*, Long Beach, USA, 2019, pp. 7164–7173.
- [82] M. Yi, L. Hou, J. Sun, *et al.*, “Improved OOD generalization via adversarial training and pretraining,” in *International Conference on Machine Learning*, Virtual Conference, 2021, pp. 11 987–11 997.
- [83] F. Wenzel, A. Dittadi, P. Gehler, *et al.*, “Assaying out-of-distribution generalization in transfer learning,” in *Advances in Neural Information Processing Systems*, New Orleans, USA, 2022, pp. 7181–7198.
- [84] F. Hampel, “The influence curve and its role in robust estimation,” *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 383–393, 1974.
- [85] L. Jaeckel, “The infinitesimal jackknife,” Unpublished, 1972.
- [86] R. D. Cook, “Assessment of local influence,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 48, no. 2, pp. 133–155, 1986.
- [87] W. Thomas and R. D. Cook, “Assessing influence on predictions from generalized linear models,” *Technometrics*, vol. 32, no. 1, pp. 59–65, 1990.
- [88] B. Wei, Y. Hu, and W. Fung, “Generalized leverage and its applications,” *Scandinavian Journal of Statistics*, vol. 25, no. 1, pp. 25–37, 1998.
- [89] R. D. Cook and S. Weisberg, “Characterizations of an empirical influence function for detecting influential cases in regression,” *Technometrics*, vol. 22, no. 4, pp. 495–508, 1980.

-
- [90] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 1885–1894.
- [91] B. A. Pearlmutter, “Fast exact multiplication by the hessian,” *Neural Computation*, vol. 6, no. 1, pp. 147–160, 1994.
- [92] N. Agarwal, B. Bullins, and E. Hazan, “Second-order stochastic optimization for machine learning in linear time,” *Journal of Machine Learning Research*, vol. 18, no. 116, pp. 1–40, 2017.
- [93] G. Pruthi, F. Liu, S. Kale, *et al.*, “Estimating training data influence by tracing gradient descent,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2020, pp. 19 920–19 930.
- [94] S. Basu, X. You, and S. Feizi, “On second-order group influence functions for black-box predictions,” in *International Conference on Machine Learning*, Virtual Conference, 2020, pp. 715–724.
- [95] P. W. Koh, K. Ang, H. Teo, *et al.*, “On the accuracy of influence functions for measuring group effects,” in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2019.
- [96] S. Yang, Z. Xie, H. Peng, *et al.*, “Dataset pruning: Reducing training data by examining generalization influence,” in *International Conference on Learning Representations*, Kigali, Rwanda, 2023.
- [97] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, 1st. Cambridge, UK: Cambridge University Press, 2014, pp. 36–37.
- [98] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, 2nd. Cambridge, MA: MIT Press, 2018, pp. 10–11.
- [99] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st. Cambridge MA., USA: MIT Press, 2016, pp. 151–153.
- [100] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, San Diego, USA, 2015.
-

-
- [101] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, Lille, France, 2015, pp. 448–456.
- [102] B. Schölkopf, “The kernel trick for distances,” in *Advances in Neural Information Processing Systems*, Denver, United States, 2000.
- [103] M. W. Theunissen, C. Mouton, and M. H. Davel, “The missing margin: How sample corruption affects distance to the boundary in ANNs,” in *Southern African Conference for Artificial Intelligence Research*, Stellenbosch, South Africa, 2022, pp. 78–92.
- [104] Z. Jiang, C. Zhang, K. Talwar, *et al.*, “Characterizing structural regularities of labeled data in overparameterized models,” in *International Conference on Machine Learning*, Virtual Conference, 2021, pp. 5034–5044.
- [105] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” in *Psychology of Learning and Motivation*, London, UK, 1989, pp. 109–165.
- [106] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, San Diego, USA, 2015, pp. 1–14.
- [107] M. W. Theunissen, M. H. Davel, and E. Barnard, “Benign interpolation of noise in deep learning,” *South African Computer Journal*, vol. 32, no. 2, pp. 80–101, 2020.
- [108] Cleanlab Inc., *Cleanlab: Data-centric AI tools for messy, real-world data and labels*, version 2.7.1, Feb. 2025. [Online]. Available: <https://github.com/cleanlab/cleanlab>.
- [109] C. G. Northcutt, A. Athalye, and J. Mueller, “Pervasive label errors in test sets destabilize machine learning benchmarks,” in *Advances in Neural Information Processing Systems*, Virtual Conference, 2021.
- [110] D. Kwok, N. Anand, J. Frankle, *et al.*, “Dataset difficulty and the role of inductive bias,” *arXiv preprint*, 2017. arXiv: 2401.01867 [cs.LG].
-

-
- [111] M. Zhang, Y. Li, X. Liu, *et al.*, “Binary relevance for multi-label learning: An overview,” *Frontiers of Computer Science*, vol. 12, no. 2, pp. 191–202, 2018.
- [112] Y. L. Thian, D. W. Ng, J. T. Hallinan, *et al.*, “Effect of training data volume on performance of convolutional neural network pneumothorax classifiers,” *Journal of Digital Imaging*, vol. 35, no. 4, pp. 881–892, 2022.
- [113] AppliedAI Institute (TransferLab Team), *PyDVL: Python data valuation library*, version 0.10.0, Apr. 2025. [Online]. Available: <https://github.com/aai-institute/pyDVL>.

Appendix A

Supplemental content

We provide supplementary material related to Chapter 4, 6 and 7.

A.1 Instance-dependent label noise

As mentioned in Section 4.2.2, instance-dependent label noise is generated following the approach of Xia *et al.* [60].

We start by sampling instance-dependent (sample-dependent) flip rates $\mathbf{q} \in \mathbb{R}^{|D_{train}|}$ from a truncated normal distribution where each entry $q_i \sim \mathcal{N}(\eta, 0.1^2, [0, 1])$ and $[0, 1]$ indicates the range of the distribution. This means that not all training samples have the same probability of being flipped. We also sample c class-specific matrices $\mathbf{W}^{(j)} \in \mathbb{R}^{p \times c}$ from the standard normal distribution where each entry $w_{i,j} \sim \mathcal{N}(0, 1^2)$ and p denotes the dimensionality of the input features.

Consider a sample $(\mathbf{x}_i, y_i)$ belonging to class j i.e., $y_i = j$. We generate an *instance-dependent flip score* vector $\mathbf{v} \in \mathbb{R}^c$ for the sample by multiplying the input feature vector

$\mathbf{x}_i$ with its corresponding class-specific matrix $\mathbf{W}^{(j)}$:

$$\mathbf{v} = \mathbf{x}_i \times \mathbf{W}^{(j)}.$$

Next, we make $\mathbf{v}$ a probability vector by doing the following. We explicitly set $v_j = -\infty$ to ensure that the j -th class entry of the vector is zero after normalization. We then normalize the remaining entries using the softmax function and scale them by the instance-specific flip rate q_i :

$$\tilde{\mathbf{v}} = q_i \times \frac{e^{v_i}}{\sum_{j=1}^c e^{v_j}}.$$

Finally, we assign the class entry (correct-label) to be the remaining probability mass:

$$v_j = 1 - q_i.$$

This yields a valid instance-dependent transition distribution $\tilde{\mathbf{v}}$ for sample $(\mathbf{x}_i, y_i)$, whose non-class entries sum to q_i and whose class entry is $1 - q_i$. The noisy label $\tilde{y}_i$ is then sampled from this categorical distribution:

$$\tilde{y}_i \sim \text{Cat}(\tilde{\mathbf{v}}).$$

A.2 k-NN distance to class centroid

Figure A.1 shows the *k-NN distance* detector scores against the minimum distance to a class centroid (i.e., the mean feature vector of all samples belonging to that class), excluding the distance to the sample’s labelled centroid. The results are presented in a 3×3 grid of subplots, where each row corresponds to one of the three datasets (Synthetic, MNIST, CIFAR-10) and each column represents a different type of label noise (symmetric, asymmetric, and instance-dependent).

Firstly, the results in Figure A.1 for the synthetic datasets (top row) show that the *k-NN distance* detector tends to assign the highest scores to samples that lie far from their closest class centroid (excluding their labelled centroid). This indicates that points situated near the periphery of their class distribution—particularly those isolated from other classes—receive the largest distance scores. Because such samples are pruned first, the remaining

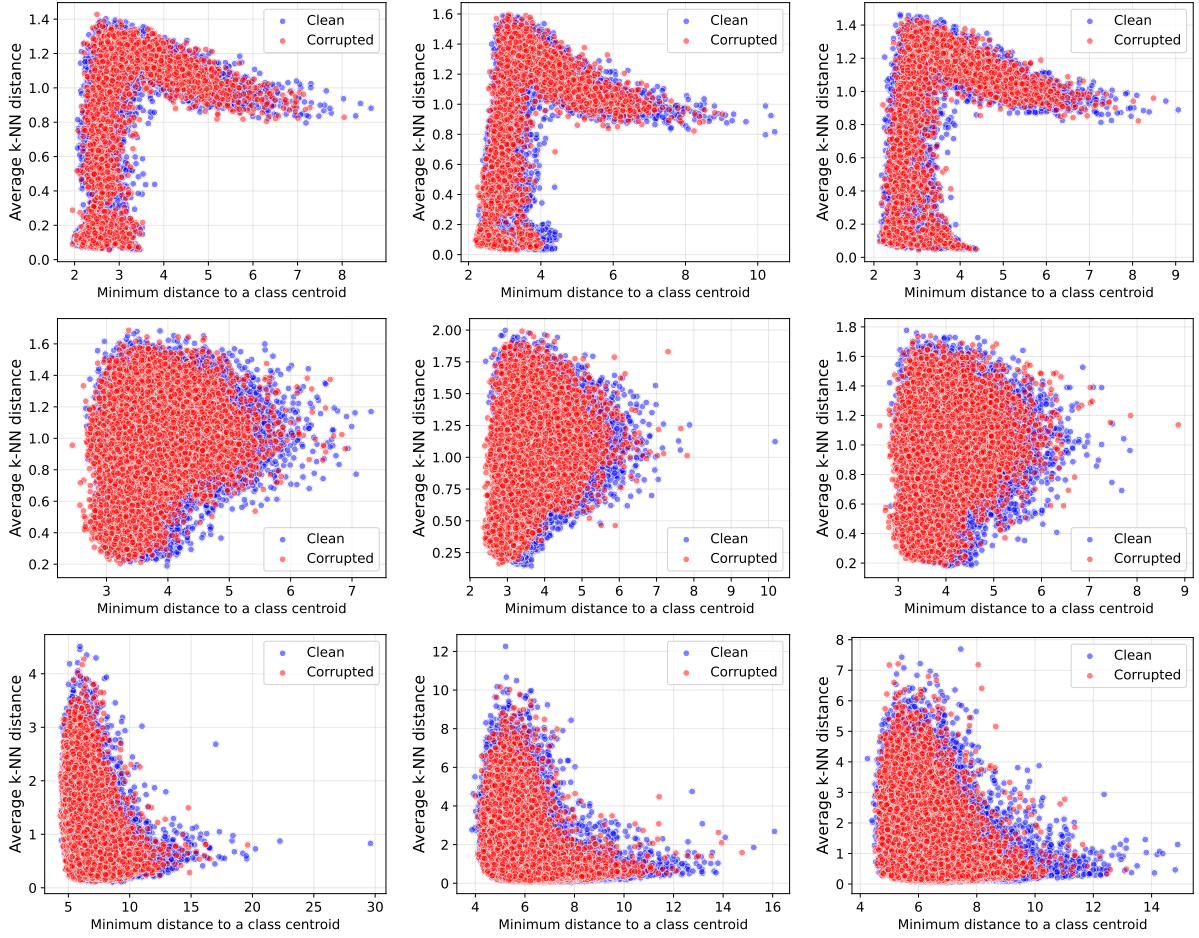


Figure A.1: k -NN distance score versus the distance to the closest class centroid (excluding the labelled centroid) of the train set with $\eta = 0.3$ label noise for synthetic (top row), MNIST (middle row), and CIFAR-10 (bottom row) datasets. The results for the three types of label noise are shown in the columns: symmetric (left column), asymmetric (middle column) and instance-dependent label noise (right column). Clean samples are shown in blue while label corrupted samples are shown in red.

data become increasingly concentrated in the regions where class distributions overlap, thereby making the classification task more difficult.

The phenomenon described above is less pronounced for MNIST (middle row) and even less so for CIFAR-10 (bottom row). This is consistent with the performance of the k -NN distance detector observed in Figure 6.1.

Secondly, Figure A.1 also reveals that the k -NN distance detector fails to distinguish

between label corrupted and clean samples across all tasks. This implies that the increase observed in the k -NN distance ψ -curve on CIFAR-10 with symmetric label noise (Figure 6.1, bottom left) is driven by samples that are problematic due to their large average k -NN distance from other samples, and not necessarily because they are label corrupted.

A.3 Explicit label noise detection of learning-centric detectors

To assess whether the *VWFG* detector assigns high scores to corrupted samples (and therefore prunes them early) we plot the cumulative corruption percentage as a function of the fraction of the training set pruned, and compare it with the *EL2N** and *sample forgetting** detectors. The cumulative corruption percentage is computed by counting the number of corrupted samples removed at any point during pruning and dividing by the total number of corrupted samples in the train set, then multiplying by 100 to express it as a percentage. The results for the four datasets shown in Figure 7.1 are presented in Figure A.2.

From Figure A.2, we observe that the *VWFG* detector consistently scores corrupted samples higher than the other two detectors (and therefore these samples are removed earlier), except on CIFAR-10 with symmetric label noise (bottom left), where its performance matches that of the *EL2N** detector. We also find that by the 30% pruning point, a substantial portion of the label noise has been removed across all datasets.

A.4 Parameter influence PCA

In Section 7.4.4 (Figure 7.5) we visualize several noisy CIFAR-10 samples based on the cosine similarity between their parameter influence vectors as captured in the $\mathbf{S}$ matrix defined in Section 7.4.2.

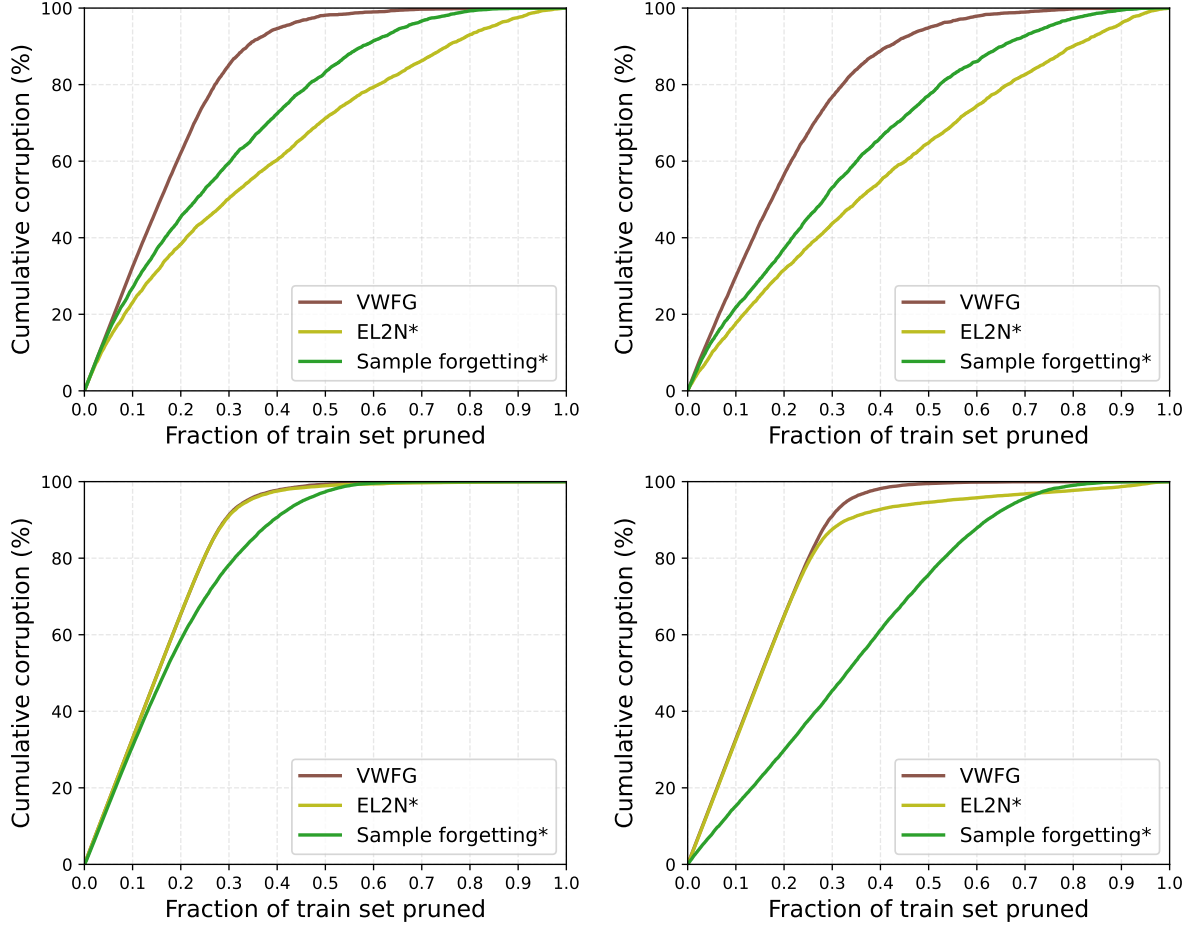


Figure A.2: The cumulative corruption percentage of the learning-centric detectors as a function of pruning an increasing portion of the train set with $\eta = 0.3$ label noise for synthetic (top row) and CIFAR-10 (bottom row) datasets. The results for the two types of label noise are shown in the columns: symmetric (left column) and asymmetric (right column).

Here, using the exact same experimental setup, we embed these parameter influence vectors into a lower-dimensional space to visualize the similarities between them. Specifically, we estimate the first two principle component directions (directions of highest explained variance in the native parameter space) using 10,000 randomly sampled training samples. Then we plot samples with the same target label in a single two-dimensional plot. We make use of an out-of-the-box principle component analysis algorithm for this.

The resulting scatter plot, for the *bird* class, can be seen in Figure A.3. The smaller

heatmap in the figure is only meant to illustrate the general location of the uncorrupted samples (which would be more numerous than the corrupted samples).

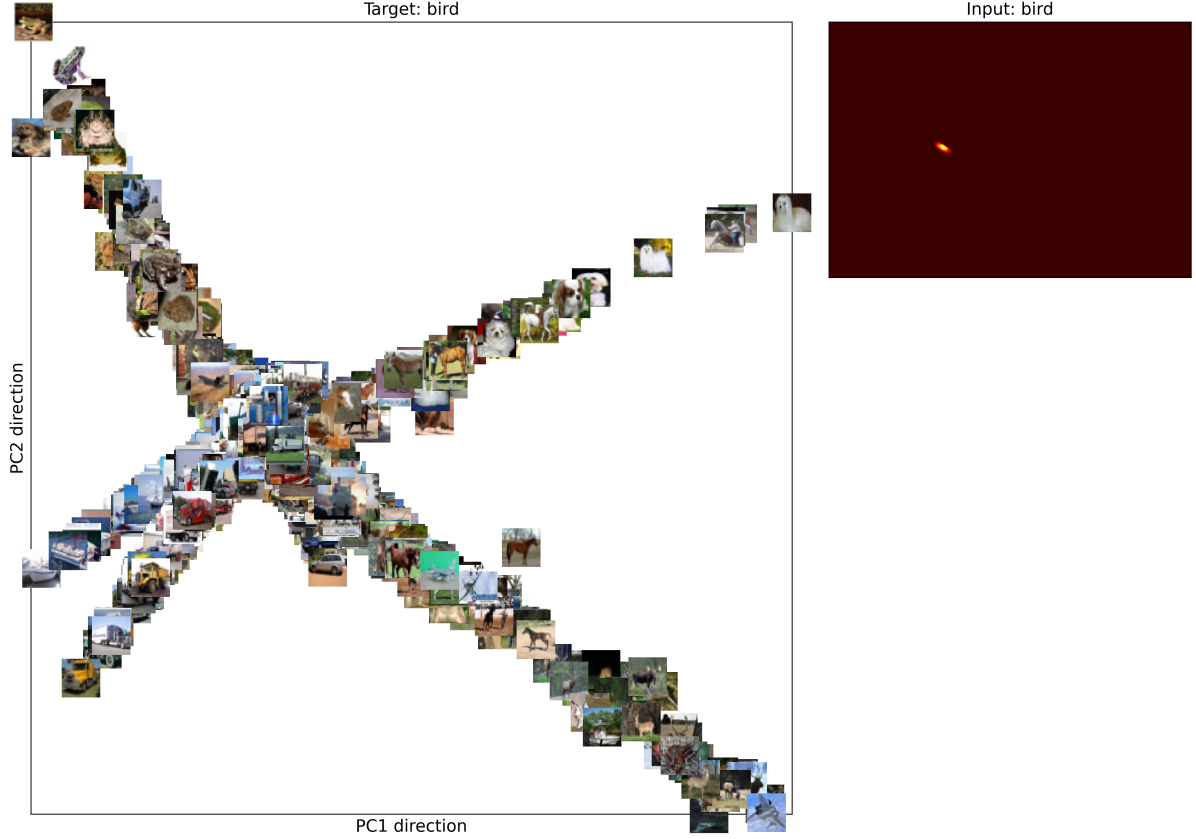


Figure A.3: All CIFAR-10 samples labelled *bird* embedded into an estimate of the first and second principle component directions. The smaller heatmap to the top-right is a visualization of a simple Gaussian Kernel Density function fitted to a subset of the scatter-plot to the left. This subset corresponds to the samples that were not label corrupted.

From Figure A.3 we observe that most samples cluster around a central point, indicating corresponding parameter influence vectors pointing into similar directions. We also observe multiple off-shoot clusters which seem to mostly correspond to input structures from “frog”, “dog”, “horse”, “truck”, and “ship” in clock-wise order.

What this represents is that, for a given target class, there are many samples that bias the parameter updates into a similar direction. However, there is a noticeable number of them that point in very different directions in parameter space. This means that

the solution that is converged on (which determines model performance) is intimately affected by what combination of samples are present in the train set. Some combinations of samples bias the optimizer in one direction and others do so into a different direction. These influences can cancel each other out or strengthen each other. It is then clear that the degree to which a sample is problematic should be interdependent on others based on their corresponding influences on the overall optimization process.

For completeness we include the corresponding PCA plots for the other classes without discussion.

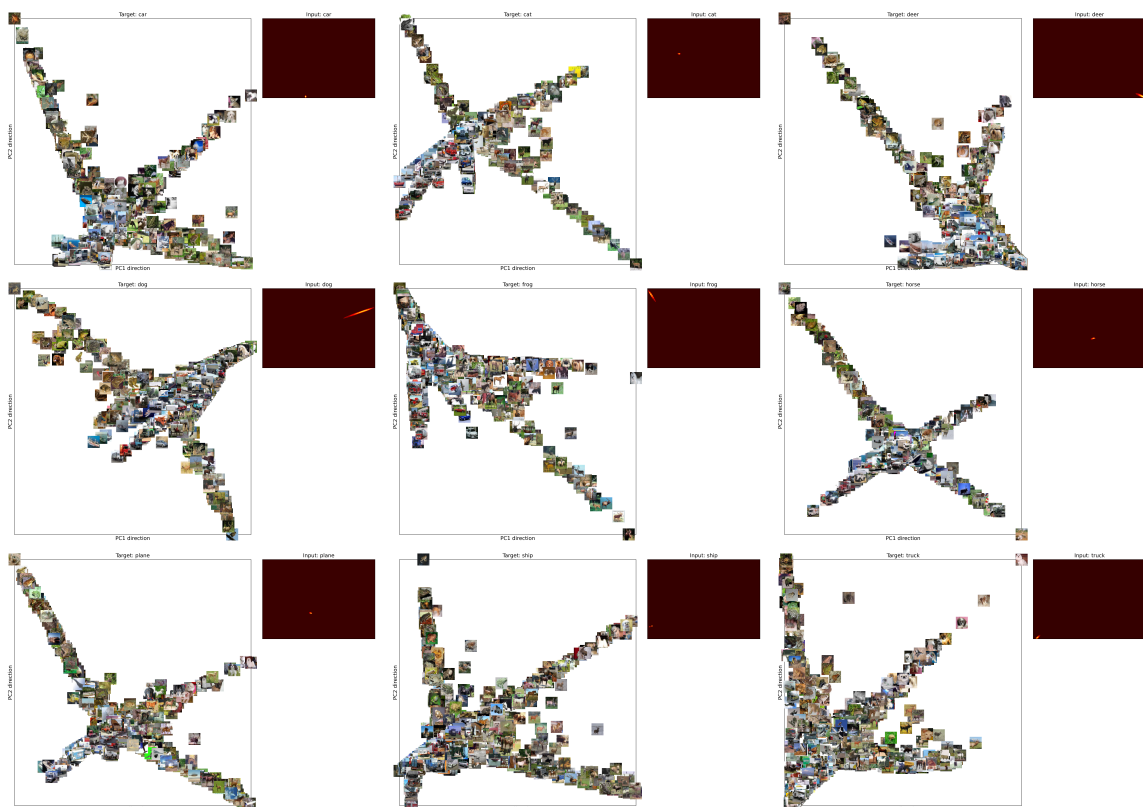


Figure A.4: Same as Figure A.3 but for the remaining classes in CIFAR-10.