Exploring neural network training dynamics through binary node activations

Daniël G. Haasbroek[0000−0002−9974−3626] and
Marelie H. Davel[0000−0003−3103−5858]

Multilingual Speech Technologies (MuST), North-West University, South Africa; and
CAIR, South Africa.

Abstract. Each node in a neural network is trained to activate for a specific region in the input domain. Any training samples that fall within this domain are therefore implicitly clustered together. Recent work has highlighted the importance of these clusters during the training process but has not yet investigated their evolution during training. Towards this goal, we train several ReLU-activated MLPs on a simple classification task (MNIST) and show that a consistent training process emerges: (1) sample clusters initially increase in size and then decrease as training progresses, (2) the size of sample clusters in the first layer decreases more rapidly than in deeper layers, (3) binary node activations, especially of nodes in deeper layers, become more sensitive to class membership as training progresses, (4) individual nodes remain poor predictors of class membership, even if accurate when applied as a group. We report on the detail of these findings and interpret them from the perspective of a high-dimensional clustering process.

Keywords: Neural networks · Generalisation · Clustering

1 Introduction

Deep neural networks (DNNs) have been used to solve increasingly difficult tasks with increasingly high accuracy, and are particularly successful when modelling complex relationships from large quantities of high-dimensional data [8]. While DNN models perform extremely well given sufficient training data, the DNN training process itself is computationally inefficient, with model optimisation requiring expensive searches across a large number of interacting hyperparameters. This search process is mainly guided by heuristics, and by tracking performance on training and held-out validation sets, as no comprehensive theoretical framework yet exists with which to reason about the training process or the expected ability of the optimised models to generalise to out-of-sample data.

The generalisation ability of DNNs has been the topic of much controversy and has been studied from a variety of perspectives. Studies that aim to characterise and predict the generalisation ability of DNNs include approaches that consider the complexity of the hypothesis space, the geometry of the loss surface, characteristics of the classification margins, and statistical measures of uniform

stability and robustness. (See Section 2.1.) While each approach provides additional insight, a general analysis framework of DNN behaviour remains elusive. The 'apparent paradox' that DNNs are able to generalise well despite extremely large capacity remains largely unresolved [14].

With 'DNN behaviour' we refer to the performance of a DNN during and after training, as measured on different subsets of the data, both seen during training and not. Characterising this behaviour should allow us to reason about the training process and differences among networks, and to predict characteristics that lead to better performance.

One approach towards probing DNN behaviour is to consider nodes as individual classifiers, collaborating in solving a network-wide task [4]. A consequence of this analysis is that each individual node is implicitly associated with the specific cluster of samples for which it activates. These sample clusters then become useful elements in analysing network behaviour. Specifically, as any node delineates a region in input space for which it activates, any samples that fall within this region are in effect clustered together. These clusters are then used to refine the weights linked to the specific node, improving cluster boundaries.

While the potential importance of these clusters during the training process has been highlighted [4,27], their evolution during training has not yet been explored. Towards this goal, we train several ReLU-activated MLPs on a simple classification task (MNIST) and track the process whereby these sample sets are formed during training. The **main contributions** of this paper are the following:

1. We provide additional motivation for the potential importance of 'sample sets' (the set of samples that activates an individual node) and their corresponding sample-feature clusters when analysing DNN behaviour.
2. We report on the evolution of these clusters during the training process of different fully-connected feedforward networks, and demonstrate that a consistent training process emerges.
3. We interpret these findings in terms of a high-dimensional clustering process, which we conjecture to be a useful perspective when analysing the generalisation ability of neural networks.

We first present relevant background (Section 2), before discussing our motivation for studying sample clusters (Section 3). A description of the analysis approach follows in Section 4, with Section 5 measuring and reporting on the the process whereby sample sets evolve during training. Findings are discussed in Section 6, and summarised in Section 7.

2 Background

We briefly discuss the generalisation ability of DNNs and approaches towards studying this. We then review earlier work related to the role and analysis of sample sets, specifically focusing on sample sets as an element in probing the generalisation ability of DNNs.

2.1 Generalisation in DNNs

DNNs are well understood from the perspective of expressivity: it is known that even a shallow network with sufficient nodes and non-linear activations is able to approximate any function, given some caveats that typically do not apply to real-world data [24]. Similarly, gradient-based optimisation procedures are theoretically well-grounded, with the conditions for finding minima known. Being expressive and trainable, however, does not imply the ability to generalise well; this requires a model to accurately capture the 'true' underlying data distribution, identifying relevant features and their interaction throughout the input domain.

Statistical learning theory (SLT) suggests that the generalisation error of a trained model is bounded above by the complexity of the hypothesis space of the modelling method [2,11]. This bound explains the generalisation ability of many model types, in the sense that an increase in the complexity of the hypothesis space, beyond the amount necessary to capture important relationships, leads to poorer generalisation of trained models [11]. When the complexity of the hypothesis space is higher than needed, good generalisation can still be obtained by introducing a preference for certain functions in the hypothesis space [10]. The use of such regularisation techniques also explains the generalisation ability of many model types [10,28].

For DNNs, neither the complexity of the hypothesis space of a particular network nor the use of regularisation techniques during training can adequately explain generalisation [28]. This was demonstrated in [28] and [26], among others, where the authors obtained good generalisation with networks that could, without any modification, fit random data easily, regardless of the use of regularisation techniques.

A significant body of work has studied generalisation in DNNs. In addition to a host of empirical studies [1,17,22,23], we highlight four approaches:

- **Geometry of the loss landscape.** It has been argued that the smoothness of the loss landscape and, specifically, the flatness of minima can lead to better generalisation [12,15]. This follows the intuition that, under these conditions, an applicable minimum should be more easily accessible during gradient descent, and small perturbations in either input or parameter space should not influence model behaviour. In high-dimensional space, however, it is extremely difficult to obtain a consistent perspective of the error surface, and it has been shown that this error surface can fairly easily be manipulated with little effect on generalisation [5].
- **Statistical measures.** Both the stability of the training process (stability when trained on different datasets) and its robustness (expected behaviour when trained on all possible datasets) have produced insights into the generalisation behaviour of DNNs [9,25,18]. Kawaguchi et al. [14] argue convincingly that there is no paradox when applying such measures from SLT to DNNs; rather, their direct applicability is restricted.
- **Complexity of the hypothesis space.** Complexity can be reduced through regularisation (as discussed above) or through inducing sparsity. With a

smaller set of trainable parameters, prediction accuracy is expected to be more stable, as justified by SLT [21]. To date, sparsity measures have been more useful in improving the computational cost and interpretability of networks than in predicting generalisation ability [7,19].
– **Margin distributions.** These approaches study the decision margin in order to explain generalisation ability, as has been successfully done with linear models such as support vector machines (SVMs). Results related to DNNs [6,13] are promising, with [13], specifically, demonstrating that a linear model, trained on the margin distributions of numerous DNNs, can predict generalisation error.

These approaches tend to consider the network as a whole, with less attention paid to the role of the individual subcomponents — an approach taken in [4], and discussed in more detail below.

2.2 Sample sets

Simultaneously introduced in [3,4,26], the term 'sample set' refers to the node-specific set of samples that activate a given node. The initial definition arose in the context of a *ReLU-activated*, fully-connected feedforward network applied to a classification task; this is also the context we use for the current discussion.

In [4], a DNN is viewed as consisting of layers of local classifiers, collaborating to solve the overall classification task. During gradient descent, weights linked to a node are only updated based on those samples that activate the node. Gradient descent can then be viewed as a two-step process: (1) during the forward pass, sample sets are created; (2) during the backward pass and parameter-update step, the sample sets are refined [4]. This refinement process is both locally and globally aware: at the local level, only selected samples (those in the sample set) are utilised to update local parameters; globally, the loss value associated with each sample takes all network parameters into account [4].

In a related study [26], extended in [27], sample-set composition is analysed in the presence of different types of noise. Interestingly, if the features of training samples are corrupted (in contrast with corrupting labels) high levels of noise — up to 90% for the tasks studied — can be absorbed by the models, without causing any detrimental effect on classification performance. Probing this behaviour revealed that nodes tend to have sample sets that contain either true or corrupted samples, rather than both. Features attuned to the noisy samples, therefore, have almost no effect on noiseless test results. This provides a simple mechanism for preventing additional parameters from hurting performance: additional parameters create an increased number of sub-components with which to isolate detrimental samples from the rest of the training data, without fundamentally changing the way in which the uncorrupted samples are modelled [27].

In both [26] and [3] it is noted that the sample set of any node is fully described by that node's fan-in weight vector; if the activation vector in the prior layer is aligned with this weight vector (if their dot product is positive), the node will activate, otherwise not. The weight vector, therefore, creates a

boundary that separates samples included in the current sample set from those excluded. Finally, in [4] it is shown that, in deeper layers, sample sets become class-sensitive, containing either almost all or almost none of the samples of any class. This behaviour is consistent across a range of architectures [4].

In summary, the above findings indicate that the training process creates clusters of samples with very specific characteristics. This points towards sample sets as a useful element in understanding network behaviour, a view that we explore further in Section 3.

3 Motivation for exploring sample-feature clusters

As discussed in Section 2.2, sample sets identify regions in the input space where specific features produce coherent behaviour. This can also be understood by viewing the process whereby samples are included or excluded from these sets. As in [4], we focus on ReLU-activated feedforward networks and define[1] the sample set $\hat{S}_{b,l,j}$ at node j of layer l as those samples in batch b that produce a positive activation value at node j. For any sample in the sample cluster, the node j in layer l can be connected to an arbitrary number of active nodes in layer $l+1$. By selecting one active node per layer, the weights connecting these active nodes can be used to define an active path $p = \{w_{p_1}, w_{p_2}, \ldots, w_{p_{N-l}}\}$ associated with a specific sample, starting at layer l and ending at a node in the output layer N.

During standard SGD training, the sample set (and only the sample set) influences the weight update. If we initially limit our analysis to networks with no bias values beyond the first layer, the weight update equation simplifies considerably. As derived in [4], the SGD weight update $\delta w_{l,j,i}$ for the weight connecting node i to node j at layer l is then given by

$$\delta w_{l,j,i} = -\eta \sum_{\mathbf{s} \in \hat{S}_{b,l,j}} \left[z^{\mathbf{s}}_{l-1,i} \sum_{p \in P^{\mathbf{s}}_j} \left(\lambda^{\mathbf{s}}_p \prod_{k=1}^{N-l} w_{p_k} \right) \right] , \tag{1}$$

where the superscript $\mathbf{s}$ indicates sample-specific values, η indicates the learning rate, $z^{\mathbf{s}}_{l-1,i}$ indicates the post-activation value at node i in layer $l-1$, $P^{\mathbf{s}}_j$ indicates the set of all active paths linking node j to the output layer, and $\lambda^{\mathbf{s}}_p$ indicates the derivative of the loss function with respect to the network output.

This sample-specific weight update can be expressed in terms of the *node-supported cost*, a scalar value that represents the portion of the final cost that can be attributed to all active paths emanating from node j, when processing sample $\mathbf{s}$ [3]. Specifically, the sample-specific node-supported cost at layer l, node j can be defined as

$$\phi^{\mathbf{s}}_{l,j} = \sum_{p \in P^{\mathbf{s}}_j} \left(\lambda^{\mathbf{s}}_p \prod_{k=1}^{N-l} w_{p_k} \right) . \tag{2}$$

[1] Note that we follow the derivations from [4] but use a different notation, for better clarity.

The update to the weight vector $\mathbf{w}_{l,j}$ feeding into node j at layer l is then given by

$$\delta \mathbf{w}_{l,j} = -\eta \sum_{\mathbf{s} \in \hat{S}_{b,l,j}} \mathbf{z}^{\mathbf{s}}_{l-1} \phi^{\mathbf{s}}_{l,j} \,. \tag{3}$$

We first consider a setup where all layers prior to l are frozen, i.e. earlier weights and consequently earlier activation values are not allowed to change. Note that this is the true situation at the first hidden layer only. Let all symbols (including sample sets) reflect the values *prior to* the weight update. Then it can be shown that any single sample $\mathbf{t}$ will be included in the sample cluster if

$$\mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{w}_{l,j} > \eta \sum_{\mathbf{s} \in \hat{S}_{b,l,j}} \left(\mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{z}^{\mathbf{s}}_{l-1} \right) \phi^{\mathbf{s}}_{l,j} \tag{4}$$

and removed otherwise. If we now define $\Sigma^{\mathbf{t}}_{l,j}$ as the net cost of the sample set at node j (layer l) as aligned with vector $\mathbf{z}^{\mathbf{t}}_{l-1}$, that is

$$\Sigma^{\mathbf{t}}_{l,j} = \sum_{\mathbf{s} \in \hat{S}_{b,l,j}} \left(\mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{z}^{\mathbf{s}}_{l-1} \right) \phi^{\mathbf{s}}_{l,j} \,, \tag{5}$$

we can derive the precise conditions for which a sample $\mathbf{t}$ not previously in the cluster will be added:

$$\mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{w}_{l,j} \leq 0; \qquad \Sigma^{\mathbf{t}}_{l,j} < 0; \qquad \left| \mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{w}_{l,j} \right| < \eta \left| \Sigma^{\mathbf{t}}_{l,j} \right| \,, \tag{6}$$

or a member sample $\mathbf{t}$ (previously in the cluster) removed:

$$\mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{w}_{l,j} > 0; \qquad \Sigma^{\mathbf{t}}_{l,j} > 0; \qquad \left| \mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{w}_{l,j} \right| \leq \eta \left| \Sigma^{\mathbf{t}}_{l,j} \right| \,. \tag{7}$$

The absolute value signs are used here to emphasise that it is the magnitude of the values that is important. In effect, a margin is created around the decision boundary (where $\mathbf{z}^{\mathbf{t}}_{l-1} \cdot \mathbf{w}_{l,j} = 0$), with the size of this boundary directly specified by the learning rate and the summed loss of all samples that activate the specific node. Only samples falling within this boundary will either be drawn in or excluded from the sample set during the update. (This concept is illustrated in Figure 8 in the Appendix.) Note that this boundary *estimates* the net win of including or excluding additional samples by measuring their alignment with all other loss-generating samples in the set. If the effect is not as anticipated, the boundary will be shifted again in the following update. In this process, the boundary forms by separating samples that do not have coherent loss behaviour. Also note that a larger loss value implies that Equation 4 will be less strict, and, thus, samples that are less aligned with the weight vector might be accepted in cases where this would not have happened with a lower node-specific loss.

This process creates natural boundaries in the input space, separating areas that will benefit from being modelled separately. We illustrate this with an example: synthetic 1-dimensional data is generated, matching an underlying distribution as illustrated in Figure 1. A small network (1 hidden layer of 30 nodes)

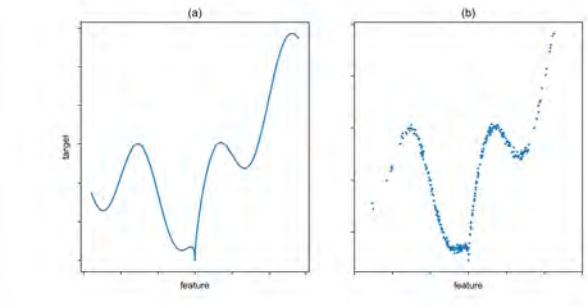

Fig. 1. 1-dim. regression example: (a) underlying distribution, (b) generated data.

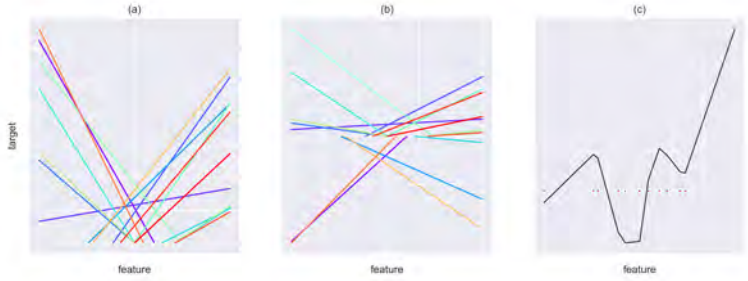

Fig. 2. 1-dim. regression example: (a) fan-in weight vectors, (b) vectors after activation and scaling, (c) weighted sum at target. Red dots indicate sample-set boundaries.

is trained to solve the regression task, and the resulting weight vectors are plotted as shown in Figure 2. Each fan-in vector at a node can be depicted as a line, based on the weight and bias value at that node. After ReLU-activation, all negative values are suppressed, and in the next layer, values are scaled based on fan-out weights. Finally, all contributions are summed in order to estimate the target value.

Once the sample-set boundaries have been drawn, scaling intermediary results to solve the overall task becomes a straightforward process; finding these boundaries is not. We propose that the heart of the training process can be studied through this clustering process of grouping relevant samples in the context of relevant features.

4 Experimental approach

Having provided our motivation for studying sample sets, we now outline our approach to investigating the dynamics of sample sets during training. We are interested in the size of the sample sets, the fraction of samples of a given class that is included in a sample set, how predictable sample-set membership is given

class identity, and how informative sample-set membership is of class identity. These are intuitive concepts that we formalise below.

4.1 Setup

We train several networks, with the number of layers (excluding the input layer) being either 2, 5, or 9, and the number of nodes per hidden layer being either 20, 80, or 320. Bias parameters are added to the first hidden layer of each network. We use ReLU activation functions for the hidden layers. All networks are trained for 200 epochs on MNIST using the Adam optimisation algorithm [16] with a minibatch size of 60. Since we do not aim to maximise the performance of any of the networks, we train all the networks with the default optimiser hyperparameters suggested in [16] ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$). We train the networks with identity activation functions at the output layer and mean-squared-error loss. We repeat the training with cross-entropy loss (softmax activation functions followed by negative log-likelihood loss). The training is performed for 3 different initialisation seeds. For all networks, the training converges.

4.2 Measurements

We measure different characteristics of sample sets during training. The per-class *sample set* of node i in hidden layer l for class c, and the entire sample set of the same node are, respectively,

$$\hat{S}_{l,i}^{(c)} = \left\{ \mathbf{s} : z_{l,i}^{\mathbf{s}} > 0, \mathbf{s} \in S_c \right\} ; \qquad \hat{S}_{l,i} = \bigcup_{c \in C} \hat{S}_{l,i}^{(c)} , \tag{8}$$

where S_c is the set of samples belonging to class c, and $z_{l,i}^{\mathbf{s}}$ is the activation of node i in hidden layer l for sample $\mathbf{s}$ [3,4,26]. At several points in the training process, we calculate the per-class *sample-set size* $|\hat{S}_{l,i}^{(c)}|$ for every node i in every hidden layer l for each class c.

We refer to the fraction of samples of a given class that is included in a sample set as the class-specific *activation fraction*, and define it for node i in hidden layer l for class c as

$$f_{l,i}^{(c)} = \frac{\left| \hat{S}_{l,i}^{(c)} \right|}{|S_c|} . \tag{9}$$

We can view the activation of node i in hidden layer l as a random variable $Z_{l,i}$, defining $\hat{Z}_{l,i} = 0$ if $Z_{l,i} \leq 0$ and $\hat{Z}_{l,i} = 1$ if $Z_{l,i} > 0$. If we also view the class of the input sample as a random variable Y, then we can approximate

$$P\left(\hat{Z}_{l,i} = 0, Y = c\right) \approx e_{l,i}(0,c) = \frac{|S_c| - \left| \hat{S}_{l,i}^{(c)} \right|}{|S|} ; \tag{10}$$

$$P\left(\hat{Z}_{l,i} = 1, Y = c\right) \approx e_{l,i}(1,c) = \frac{\left| \hat{S}_{l,i}^{(c)} \right|}{|S|} , \tag{11}$$

where S is the set of all samples. We approximate the conditional entropy of $\hat{Z}_{l,i}$ given Y and that of Y given $\hat{Z}_{l,i}$, for nodes with $|\hat{S}_{l,i}| \neq 0$ and $|\hat{S}_{l,i}| \neq |S|$, as

$$H\left(\hat{Z}_{l,i} \,|\, Y\right) \approx -\sum_{c \in C} \sum_{\hat{z} \in \{0,1\}} e_{l,i}(\hat{z}, c) \log \left(\frac{e_{l,i}(\hat{z}, c)}{\sum\limits_{x \in \{0,1\}} e_{l,i}(x, c)} \right); \tag{12}$$

$$H\left(Y \,|\, \hat{Z}_{l,i}\right) \approx -\sum_{\hat{z} \in \{0,1\}} \sum_{c \in C} e_{l,i}(\hat{z}, c) \log \left(\frac{e_{l,i}(\hat{z}, c)}{\sum\limits_{x \in C} e_{l,i}(\hat{z}, x)} \right), \tag{13}$$

where we set $0 \log(0) = 0$ [20]. Based on this, we define the *predictability* of node i in hidden layer l as

$$p_{l,i} = 1 + \frac{1}{\log(2)} \sum_{c \in C} \sum_{\hat{z} \in \{0,1\}} e_{l,i}(\hat{z}, c) \log \left(\frac{e_{l,i}(\hat{z}, c)}{\sum\limits_{x \in \{0,1\}} e_{l,i}(x, c)} \right), \tag{14}$$

and the *informativeness* as

$$u_{l,i} = 1 + \frac{1}{\log(|C|)} \sum_{\hat{z} \in \{0,1\}} \sum_{c \in C} e_{l,i}(\hat{z}, c) \log \left(\frac{e_{l,i}(\hat{z}, c)}{\sum\limits_{x \in C} e_{l,i}(\hat{z}, x)} \right), \tag{15}$$

where we again set $0 \log(0) = 0$ [20]. We do not define $p_{l,i}$ or $u_{l,i}$ for nodes with $|\hat{S}_{l,i}| = 0$ or $|\hat{S}_{l,i}| = |S|$, that is, for nodes that are always inactive (dead nodes) or always active (bias nodes). The predictability of a node is the difference between the maximum possible entropy of $\hat{Z}_{l,i}$ (based on the number of possible values of $\hat{Z}_{l,i}$) and the entropy of $\hat{Z}_{l,i}$ given Y, expressed as a fraction of the maximum possible entropy of $\hat{Z}_{l,i}$. Informally, this is proportional to the average amount of information known about $\hat{Z}_{l,i}$ given only an observation of Y. The informativeness indicates similar information about Y given $\hat{Z}_{l,i}$.

We calculate $f_{l,i}^{(c)}$, $p_{l,i}$, and $u_{l,i}$ at several logarithmically spaced points in the training process on a validation set of 12000 samples. We then aggregate these values as follows:

$$f_{l,i} = \frac{1}{|C|} \sum_{c \in C} f_{l,i}^{(c)}; \qquad f_l = \frac{1}{\left|N_a^{(l)}\right|} \sum_{i \in N_a^{(l)}} f_{l,i}; \tag{16}$$

$$p_l = \frac{1}{\left|N_b^{(l)}\right|} \sum_{i \in N_b^{(l)}} p_{l,i}; \qquad u_l = \frac{1}{\left|N_b^{(l)}\right|} \sum_{i \in N_b^{(l)}} u_{l,i}, \tag{17}$$

where $N_a^{(l)}$ is the set of nodes in hidden layer l for which $|\hat{S}_{l,i}| \neq 0$, and $N_b^{(l)}$ is the set of nodes in hidden layer l for which $|\hat{S}_{l,i}| \neq 0$ and $|\hat{S}_{l,i}| \neq |S|$.

5 Results

Here, we highlight interesting patterns observed by presenting results that display typical behaviour. Any other results that do not follow these patterns are pointed out. The performance of all networks during training is similar to that shown in Figure 3. All networks with 80 or more nodes per hidden layer achieve error rates smaller than 5% on the validation set. Those with 20 nodes per hidden layer achieve error rates smaller than 7% on the validation set.

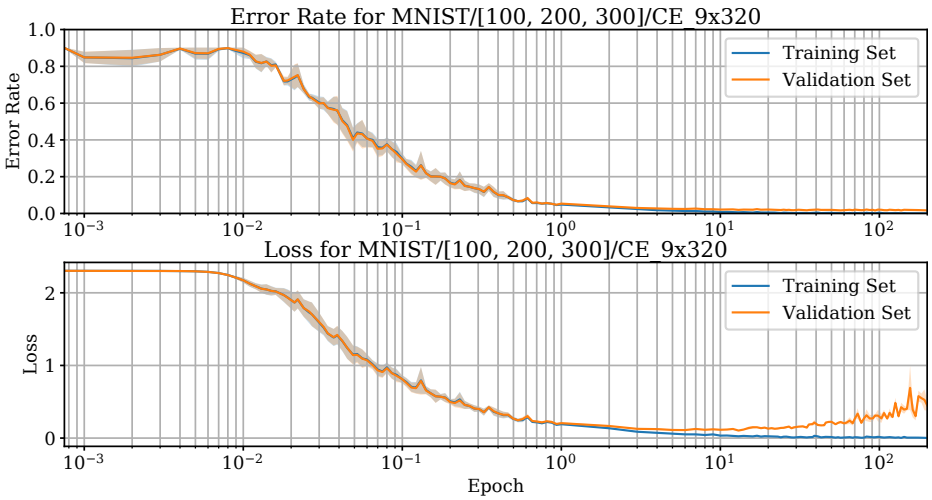

Fig. 3. Performance of 9×320 networks trained with cross-entropy loss, averaged across initialisation seeds. Shaded areas indicate the standard error.

5.1 Activation fraction

The activation fraction $f_{l,i}$ of the hidden nodes of a 9-layer network with 320 nodes per layer is shown in Figure 4. The average activation fraction f_l for the same architecture is shown in Figure 5. It is worth noting that the number of samples per class in the validation set is approximately constant across classes. As a result, $f_{l,i}$ is approximately equal to the fraction of *all* samples for which node i in layer l activates.

For most nodes, an increase in the sample-set size is observed very early in training. This indicates that the training process initially exposes most nodes to most samples. The nodes for which this does not hold activate for almost no samples during the entire training process, and, therefore, these nodes contribute very little to the network. The initial increase in sample-set size is followed by a gradual decrease for the rest of training. For networks trained with cross-entropy loss, this decrease is more rapid in shallower layers than in deeper layers. For

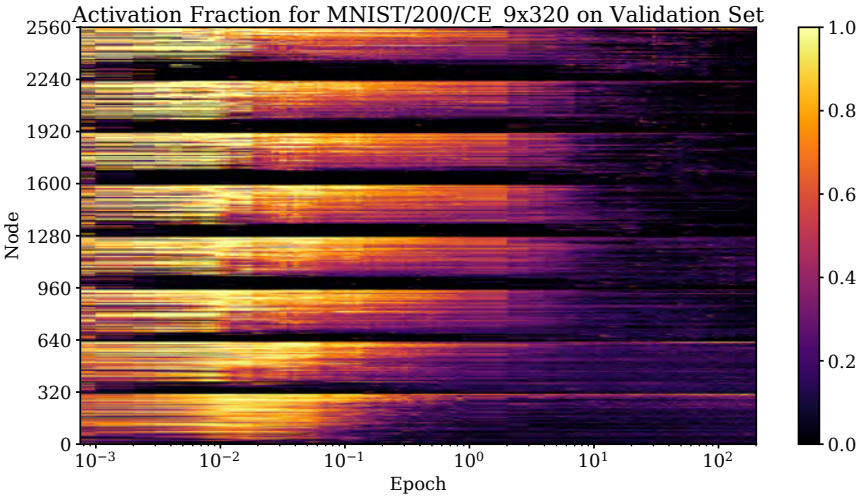

Fig. 4. Activation fraction for all hidden nodes in a 9×320 network trained with cross-entropy loss, calculated on the validation set. Nodes are numbered from shallowest to deepest.

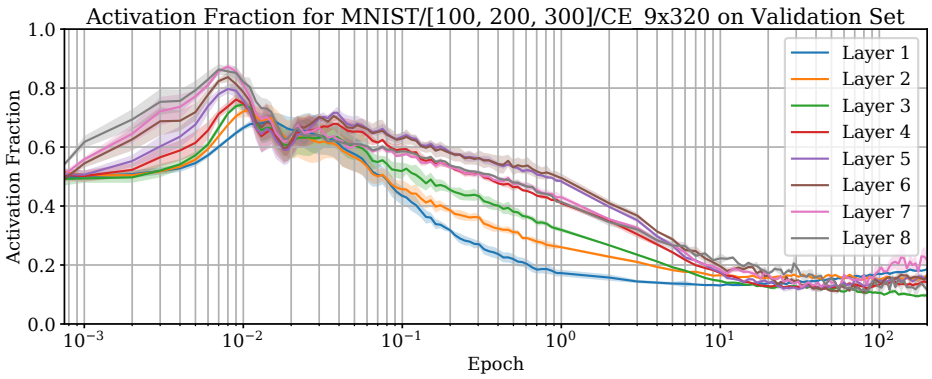

Fig. 5. Activation fraction averaged across nodes and initialisation seeds of 9×320 networks trained with cross-entropy loss, calculated on the validation set. Shaded areas indicate the standard error of the average across seeds. Layers are numbered from shallowest to deepest.

networks trained with MSE loss, this decrease is more rapid in the first hidden layer than in deeper layers.

Some results of individual networks are worth pointing out. For 9-layer networks trained with MSE loss, the sample sets of nodes in deeper layers contain almost all of the samples at the point where the sample sets are largest (see Figure 9 in the Appendix). For the 2-layer networks with 320 nodes per layer trained with MSE loss, the average activation fraction decreases during the entire training process (see Figure 10 in the Appendix). All other networks follow the same trends as in Figures 4 and 5.

5.2 Predictability

The average predictability p_l for a 9-layer network with 320 nodes per layer is shown in Figure 6. The average predictability of all layers increases at the start

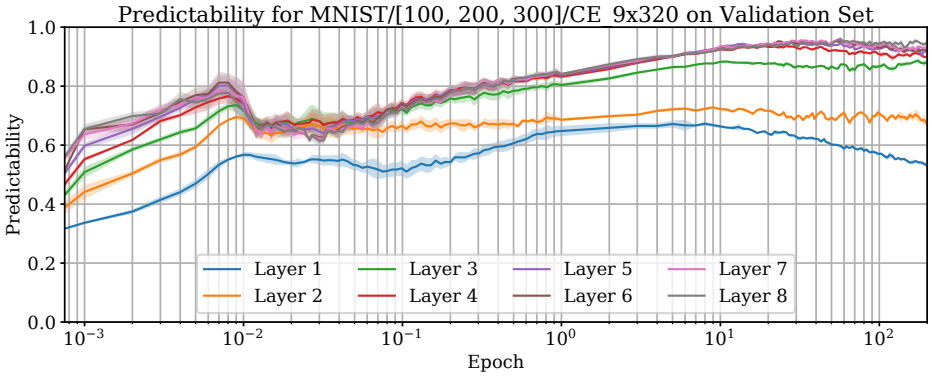

Fig. 6. Predictability averaged across nodes and initialisation seeds of 9×320 networks trained with cross-entropy loss, calculated on the validation set. Shaded areas indicate the standard error of the average across seeds. Layers are numbered from shallowest to deepest.

of training. For a few of the shallower layers, the predictability remains constant or decreases towards the end of training; for the rest of the layers, it continues to increase. The predictability of deeper layers is consistently higher than that of shallower layers. Not all results show the peaks that appear in Figure 6, and, therefore, we refrain from assigning meaning to them. These trends indicate that nodes, especially those in deeper layers, become increasingly more sensitive to class membership as training progresses. It also shows that nodes in deeper layers are more sensitive to class membership — a result that is confirmed by [4].

5.3 Informativeness

The average informativeness u_l for a 9-layer network with 320 nodes per layers is shown in Figure 7. For all layers, the informativeness increases and subsequently

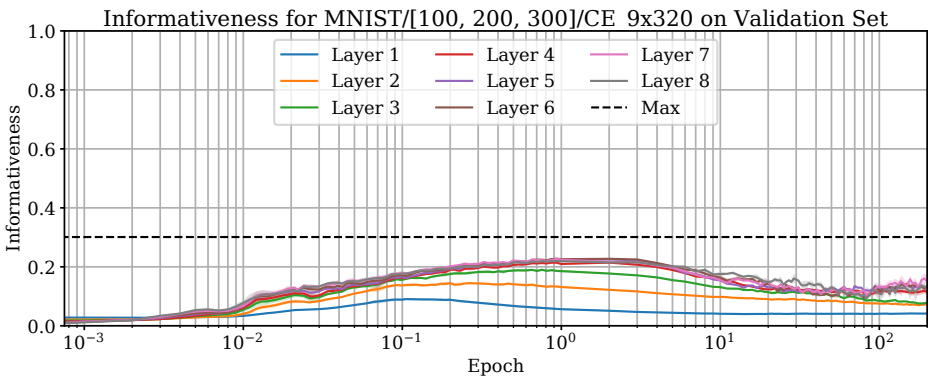

Fig. 7. Informativeness averaged across nodes and initialisation seeds of 9×320 networks trained with cross-entropy loss, calculated on the validation set. Shaded areas indicate the standard error of the average across seeds. Layers are numbered from shallowest to deepest. 'Max' indicates the maximum informativeness that can be achieved.

decreases slightly during training. Assuming that the 10 MNIST classes are perfectly balanced in the validation set, the maximum informativeness that can be achieved is $^{\log(2)}/_{\log(10)} \approx 0.3$, since the binary activation $\hat{Z}_{l,i}$ of a node can only have one of 2 values, but the class Y can have one of 10 values. The increase in informativeness shows that binary node activations become more indicative of class membership as training progresses. However, the amount of information required to establish the class with certainty means that the binary activation of any single node remains a poor predictor of class membership.

6 Discussion

To summarise our empirical findings: (1) Sample clusters initially increase in size and then decrease as training progresses. (2) Most nodes are exposed to most samples very early in the training process. (3) The point where the activation fraction starts decreasing overlaps with the point where network loss starts decreasing. (4) The size of sample clusters in the first layer decreases more rapidly than in deeper layers. (5) Binary node activations, especially of nodes in deeper layers, become more sensitive to class membership as training progresses. (6) Nodes in shallower layers tend to be less sensitive to class membership than nodes in deeper layers. (7) Binary node activations become slightly more indicative of class membership as training progresses, but remain poor predictors of class membership, even if accurate when applied as a group.

How do the above findings shed light on the creation of the sample clusters described in Section 3? When considering the training process, specific phases become evident: Upon initialisation, weights tend to point in arbitrary directions, and nodes in the network activate for samples found somewhere in the vicinity of their fan-in weight vector, creating the initial clusters. Since the loss is initially

very large, additional samples are quickly drawn into the sample set of each of the nodes, according to Equation 4. It is during this time that the activation fraction grows. (See Figure 5.) This process continues up to the point where the activation fraction reaches its maximum. At this point, most nodes are being exposed to most samples, and also to most features.

It is only as the loss starts decreasing (Figure 3) that the activation fraction also starts decreasing, with nodes becoming increasingly specific. As nodes become more specific, weights become attuned to solving smaller subtasks, specifically trying to address any unresolved samples in its own sample set. At this point, nodes actively start selecting features (and directions in feature space) that are the most appropriate for solving its own subtask.

This process continues up to convergence, occasionally displaying a ripple effect where we conjecture that clusters are being re-shuffled. During training, only samples that have not been fully resolved contribute to weight updates. Samples with zero loss are simply ignored, unless a change in the network increases the loss value for such a sample as a side effect, which may cause clusters to break apart or re-combine.

Taken together, this process describes a practical approach to solving a high-dimensional clustering task, and specifically, to finding combinations of samples and features that show coherent behaviour and can therefore be modelled together effectively. It is only in the first layer that the raw input features are used to form clusters; in later layers, this clustering occurs in the transformed space produced by the previous layer.

While the findings presented at the start of this section were empirically confirmed across different architectures, the above interpretation is currently to be considered conjecture, rather than fact. In our current work we are analysing each of these statements in more detail.

7 Conclusion

Motivated by the role that sample sets play in the SGD training process, we studied the evolution of sample sets throughout training for several ReLU-activated networks. Our experiments reveal a consistent training process, as summarised at the start of Section 6. We provide some insight into the SGD training process by interpreting these findings using the conditions under which samples are included or excluded from sample sets (Section 3), and by discussing how this could relate to a high-dimensional clustering process (Section 6).

The current analysis is restricted to ReLU-activated networks. As the analysis in [4], which also studied the binary behaviour of individual nodes, was successfully extended to sigmoid-activated networks, we expect to be able to extend this study to a more diverse set of architectures and datasets as well. Although we do not directly address the apparent generalisation 'paradox' or improve the training process, the presented analyses and interpretations shed light on the training process from an interesting perspective.

References

1. Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A.C., Bengio, Y., Lacoste-Julien, S.: A closer look at memorization in deep networks. In: Proceedings of the 34th International Conference on Machine Learning. pp. 233–242 (2017)
2. Bartlett, P.L., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research **3**, 463–482 (2002)
3. Davel, M.H.: Activation gap generators in neural networks. In: Proc. of the South African Forum for Artificial Intelligence Research FAIR. pp. 64–76. Cape Town, South Africa (12 2019)
4. Davel, M.H., Theunissen, M.W., Pretorius, A.M., Barnard, E.: DNNs as layers of cooperating classifiers. In: Thirty-Fourth AAAI Conference on Artificial Intelligence (2020)
5. Dinh, L., Pascanu, R., Bengio, S., Bengio, Y.: Sharp minima can generalize for deep nets. In: Proceedings of the 34th International Conference on Machine Learning. pp. 1019–1028 (2017)
6. Elsayed, G.F., Krishnan, D., Mobahi, H., Regan, K., Bengio, S.: Large margin deep networks for classification. In: Conference on Neural Information Processing Systems (2018)
7. Gale, T., Elsen, E., Hooker, S.: The state of sparsity in deep neural networks. arXiv Preprint **arXiv:1902.09574** (2019)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, pp. 22–25. MIT Press (2016)
9. Hardt, M., Recht, B., Singer, Y.: Train faster, generalize better: Stability of stochastic gradient descent. In: Proceedings of The 33rd International Conference on Machine Learning. pp. 1225–1234 (2016)
10. Hastie, T., Tibshirani, R., Friedman, J.: Chapter 2: Overview of supervised learning. In: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, pp. 9–41. Springer, 2 edn. (2017)
11. Hastie, T., Tibshirani, R., Friedman, J.: Chapter 7: Model assessment and selection. In: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, pp. 219–259. Springer, 2 edn. (2017)
12. Hochreiter, S., Schmidhuber, J.: Flat minima. Neural Computation **9**(1), 1–42 (1997)
13. Jiang, Y., Krishnan, D., Mobahi, H., Bengio, S.: Predicting the generalization gap in deep networks with margin distributions. In: International Conference on Learning Representations (2019)
14. Kawaguchi, K., Kaelbling, L.P., Bengio, Y.: Generalization in deep learning. In: Mathematics of Deep Learning. Cambridge University Press, to be published
15. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: International Conference on Learning Representations (2017)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
17. Krueger, D., Ballas, N., Jastrzebski, S., Arpit, D., Kanwal, M.S., Maharaj, T., Bengio, E., Fischer, A., Courville, A.C.: Deep nets don't learn via memorization. In: International Conference on Learning Representations (2017)
18. Kuzborskij, I., Lampert, C.H.: Data-dependent stability of stochastic gradient descent. In: Proceedings of the 35th International Conference on Machine Learning. pp. 2815–2824 (2018)

19. Loroch, D.M., Pfreundt, F.J., Wehn, N., Keuper, J.: Sparsity in deep neural networks – An empirical investigation with TensorQuant. In: ECML PKDD 2018 Workshops. pp. 5–20 (2019)
20. MacKay, D.J.: Chapter 2: Probability, entropy, and inference. In: Information Theory, Inference, and Learning Algorithms, pp. 22–46. Cambridge University Press (2005)
21. Maurer, A., Pontil, M.: Structured sparsity and generalization. Journal of Machine Learning Research **13**, 671–690 (2012)
22. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N.: Exploring generalization in deep learning. In: Conference on Neural Information Processing Systems (2017)
23. Novak, R., Bahri, Y., Abolafia, D.A., Pennington, J., Sohl-Dickstein, J.: Sensitivity and generalization in neural networks: an empirical study. In: International Conference on Learning Representations (2018)
24. Pinkus, A.: Approximation theory of the mlp model in neural networks. Acta Numerica **8**, 143–195 (1999)
25. Sokolić, J., Giryes, R., Sapiro, G., Rodrigues, M.R.D.: Robust large margin deep neural networks. IEEE Transactions on Signal Processing **65**(16), 4265–4280 (2017)
26. Theunissen, M.W., Davel, M.H., Barnard, E.: Insights regarding overfitting on noise in deep learning. In: Proc. of the South African Forum for Artificial Intelligence Research FAIR. pp. 49–63. Cape Town, South Africa (12 2019)
27. Theunissen, M.W., Davel, M.H., Barnard, E.: Benign interpolation of noise in deep learning. South African Computer Journal (12 2020)
28. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires re-thinking generalization. In: International Conference on Learning Representations (2017)

A Illustration of Equation 6 and 7

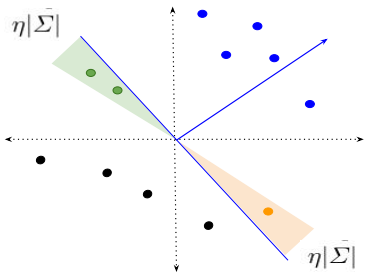

Fig. 8. The size of $\eta|\Sigma|$ determines the area where additional samples will be included in (green) or excluded from (orange) the sample set, after the next weight update. Weight vector is shown in blue, with current boundary line indicated, also in blue. The sample-set membership of other samples already in the sample set (blue) and outside the sample set (black) are not affected.

B Additional results

Figure 9 shows the activation fraction for networks in which the nodes in deeper layers have sample sets that contain almost all of the samples at the point where the sample sets are largest. This pattern holds for all 9-layer networks trained with MSE loss.

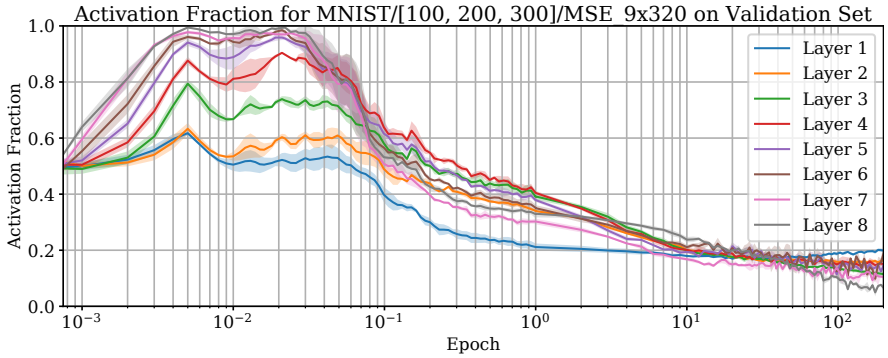

Fig. 9. Activation fraction averaged across nodes and initialisation seeds of 9×320 networks trained with mean-squared-error loss, calculated on the validation set. Shaded areas indicate the standard error of the average across seeds. Layers are numbered from shallowest to deepest.

Figure 10 shows the activation fraction for the networks in which an increase in average activation fraction is not observed. This only holds for 2-layer networks with 320 nodes per layer trained with MSE loss.

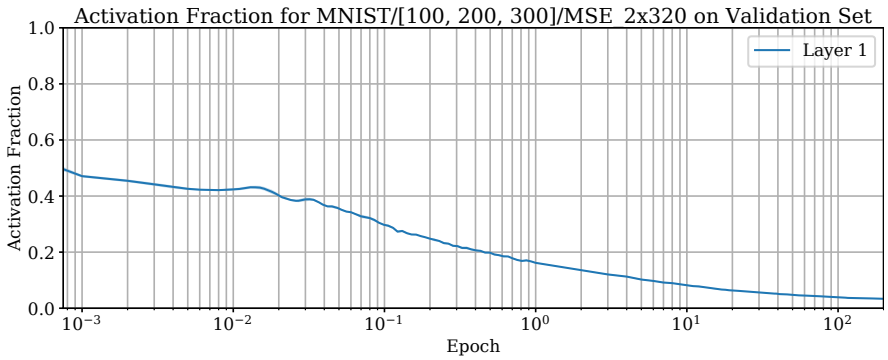

Fig. 10. Activation fraction averaged across nodes and initialisation seeds of 2×320 networks trained with mean-squared-error loss, calculated on the validation set. Shaded areas indicate the standard error of the average across seeds.