Activation gap generators in neural networks

Marelie H. Davel[0000−0003−3103−5858]

Multilingual Speech Technologies, North-West University, South Africa;
and CAIR, South Africa
marelie.davel@nwu.ac.za

Abstract. No framework exists that can explain and predict the generalisation ability of DNNs in general circumstances. In fact, this question has not been addressed for some of the least complicated of neural network architectures: fully-connected feedforward networks with ReLU activations and a limited number of hidden layers. Building on recent work [2] that demonstrates the ability of individual nodes in a hidden layer to draw class-specific activation distributions apart, we show how a simplified network architecture can be analysed in terms of these activation distributions, and more specifically, the sample distances or *activation gaps* each node produces. We provide a theoretical perspective on the utility of viewing nodes as activation gap generators, and define the gap conditions that are guaranteed to result in perfect classification of a set of samples. We support these conclusions with empirical results.

Keywords: Generalisation · fully-connected feedforward networks · activation distributions · MLP.

1 Introduction

Deep Neural Networks (DNNs) have been used to achieve excellent performance on many traditionally difficult machine learning tasks, especially high-dimensional classification tasks such as computer vision, speech recognition and machine translation [4]. DNNs generalise well: trained on a limited data set, they are able to transfer this learning to unseen inputs in a demonstrably effective way. Despite various approaches to studying this process [1, 3, 6–8, 11–14], no framework yet exists that can explain and predict this generalisation ability of DNNs in general circumstances.

Specifically, one of the central tenets of statistical learning theory links model capacity (the complexity of the hypothesis space the model represents) with expected generalisation performance [15]. However, a sufficiently large DNN represents an extremely large hypothesis space, specified by hundreds of thousands of trainable parameters. While any architecture that has a hypothesis space that is sufficiently large to be able to memorise random noise is *not* expected to generalise well, this is not the case for DNNs. In a paper that caused much controversy, Zhang et al. [17] demonstrated how both Convolutional Neural Networks (CNNs) and standard Multilayer Perceptrons (MLPs) are able to memorise noise perfectly, while extracting the signal buried within the noise with the same efficiency

as if the noise was not present. Even more pointedly, this was shown to occur with or without adding regularisation [17].

In this work we take a step back, and analyse the classification ability of a simplified neural network architecture, since even for a minimal network (as soon as it has multiple layers and non-linear activation functions) generalisation behaviour has not been fully characterised. In recent work [2], we showed how the individual nodes of a standard fully-connected feedforward network can draw class-specific activation distributions apart, to the extent that these distributions can be used to train individual likelihood estimators and produce accurate classifications at each layer of the network. Here we show that the ability of a fully-connected feedforward network to generalise can be analysed in terms of these activation distributions, and more specifically the distances or 'activation gaps' each node produces: the difference in activation value of any two samples at a node.

The main contribution of this paper is the development of a conceptual tool that can be used to probe the generalisation ability of a DNN, and the empirical confirmation of the soundness of the approach when applied to a simplified MLP architecture. We start by reviewing the concept of node-specific sample sets and nodes as likelihood estimators (Section 2) before introducing activation gaps (Section 3) and exploring the role of activation gaps in achieving perfect classification from a theoretical perspective. Expected relationships are empirically confirmed in Section 4, and possible applications of these ideas as well as future work discussed in Section 5.

2 Nodes as network elements

While a network functions as a single construct, each layer also acts as a functional element, and within a layer, each node has both a local and global function. Nodes are *locally* attuned to extracting information from a very specific part of the input space, while collaborating *globally* to solve the overall classification or regression task [2]. In this section we review the local nature of nodes: their role as individual likelihood estimators, their relationship with the samples that activate them (their 'sample sets'), and the node-specific interaction between sample sets and the weights that feed into a specific node.

From this point onwards, we restrict our discussion to fully-connected feedforward architectures with ReLU activations and unrestricted breadth (number of nodes in a hidden layer) and depth (number of hidden layers), applied to a classification task.

2.1 Nodes as estimators

The network behaviour for an architecture as described above was analysed in [2]. Generalisation ability was interpreted as arising from two collaborative information processing systems, one continuous and one discrete. Specifically, it was shown how the network both represents a discrete system that only utilises

information with regard to whether a node is active or not, and a continuous system that utilises the actual pre-activation value at each node. In both systems, each node implicitly represents the likelihood of an observation of each class: in the discrete case, this likelihood can be estimated by counting the number of times a node activates for a given class; in the continuous case, by fitting a density estimator to the pre-activation distribution of each class. In both cases, the posterior probability (given the observation) can be calculated using Bayes rule, and the probabilities multiplied across all nodes of any layer to produce a layer-specific class prediction for each sample.

An example of this is depicted in Figure 1, where an MLP trained and tested on FMNIST [16] is analysed at each layer, in the manner described above. During classification, each discrete system *only* uses information with regard to whether a node is activated or not; each continuous system *only* uses pre-activation values; and the combined system uses the true information available to the ReLU-activated network (either the continuous or discrete estimate). For this network, as for all other networks of sufficient size analysed, it was observed that at some layer, the network is able to achieve similar classification accuracy as the actual network, irrespective of the system used to perform classification [2].

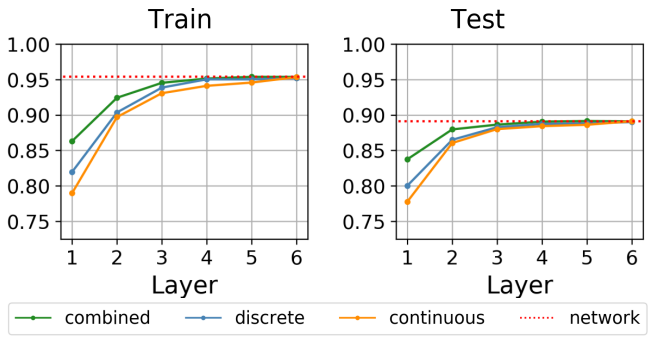


Fig. 1: Comparing continuous, discrete and combined system accuracy at each layer for a depth 6 width 100 network trained and tested on FMNIST. The red dotted line indicates network performance if trained and tested in the normal manner. Reproduced from [2].

This is quite surprising. In both systems, each node therefore uses locally available information to develop its own probability estimate in isolation, but then collaborates with other nodes to solve the classification task. Using this perspective, the set of samples that activates any node (its sample set) becomes very significant. During gradient descent, the forward process applies weights to create sample sets; the backward process uses sample sets to update weights: each weight attuned to only its specific sample set. Sample sets can be very large

(consisting of almost all samples) to very specialised, describing a few isolated samples.

2.2 Node-supported cost

The interplay between sample sets and weight updates was further investigated in [2]: rewriting the weight update $\Delta w^s_{i,j,k}$ associated with a single sample s at layer i from node k to node j in an iterative form, produces a surprisingly simple structure. Specifically, noting that

$$Relu(x) = xT(x) \tag{1}$$

$$\text{where } T(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases} \tag{2}$$

and with η the (potentially adaptive) learning rate, $a_{i-1,k}$ the activation result at layer $i-1$ for node k, λ^s_m the cost function at output node m, $z_{i,j}$ the sum of the input to node j in layer i, and $I(i,j)$ an indexing function (see [2]), the weight update associated with a single sample can be written as:

$$\Delta w^s_{i,j,k} = -\eta\, a_{i-1,k} \sum_{b=0}^{B_i-1} \lambda^s_{I(N,b)} \prod_{g=i}^{N-1} T(z_{g,I(g,b)}) \prod_{r=i+1}^{N} w_{r,I(r,b),I(r-1,b)} \tag{3}$$

with B_i a product of the the number of nodes in all layers following after layer i. If we now define the sample-specific *node-supported cost* at layer i, node j as:

$$\phi^s_{i,j} = \sum_{b=0}^{B_i-1} \lambda^s_{I(N,b)} \prod_{g=i}^{N-1} T(z^s_{g,I(g,b)}) \prod_{r=i+1}^{N} w_{r,I(r,b),I(r-1,b)} \tag{4}$$

then the weight update by a single sample can be written as

$$\delta w^s_{i,j,k} = -\eta a^s_{i-1,j,k} \phi^s_{i,j} \tag{5}$$

and over all samples (in the mini batch) used when computing the update:

$$\delta w_{i,j,k} = -\eta \sum_{s \in \{.\}_{i,j}} a^s_{i-1,j,k} \phi^s_{i,j} \tag{6}$$

This sum can either be calculated over $\{.\}_{i,j}$, or over $\{.\}_{i,j} \cap \{.\}_{i-1,k}$ as only samples that are active at node k will contribute to the sum, either way.

The node-supported cost is a scalar value that represents the portion of the final cost that can be attributed to all active paths initiated from this node, when processing sample s. Note that ϕ^s_a does not differentiate between a node that creates paths with large positive and negative costs that balance out, and one that produces a cost close to zero. Also, a positive cost implies too much activation, a negative cost, too little.

2.3 Weights as directions

From the above discussion, each node vector is updated in response to the errors remaining in its sample set. For some samples, activation values would be too high, for others too low. Per node, the process of updating the node vector can be viewed as one of finding a direction in its input space (the output of the previous layer), such that samples producing different errors are separated when calculating the projection of the sample onto the weight.

With sample sets in mind, training can then be viewed as a process of finding important directions in the layer-specific input space, projecting the original features in each of these directions to create a transformed input space, and repeating the process layer for layer. Important directions are those useful for classification: class-distinctive in the initial layers, class-specific in the later layers. It is important to note that this optimisation process is both local and global: the direction of this node vector is optimised specifically for the sample set concerned, and only the sample set concerned, resulting in a local process, but the errors used to guide the process (the node-supported cost) is calculated globally.

3 Gaps and ratios

In order to develop a theoretical perspective on the interplay between the weight vectors, node-supported cost and sample sets, we constrain our initial analysis to a much restricted architecture. This allows us to determine the exact theoretical conditions for perfect classification of a set of samples.

3.1 A simplified architecture

While an MLP is typically viewed as an input and output layer flanking any number of hidden layers, each hidden layer can also be seen as a small 3-layer

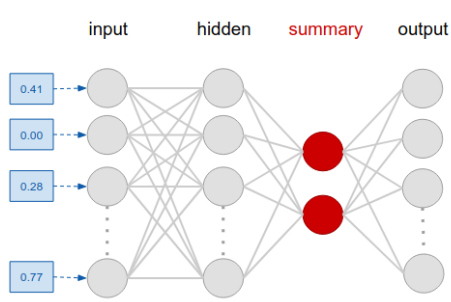


Fig. 2: In the simplified architecture, it is only the summary layer that is restricted to two nodes; all other layers are allowed an unlimited number of nodes.

subnetwork in its own right: utilising the output from the prior layer as input, and trying to address the loss (the node-supported cost of Section 2.2) passed along from its immediate output layer, the next hidden layer. As a starting point for our analysis, we therefore restrict ourselves to the setup of such a 3-layer subnetwork: considering only a single functional hidden layer in addition to an input and output layer. The term 'functional' is used to emphasise that only in this single hidden layer are nodes trained (in the standard manner) to act as generalising elements.

An additional hidden layer is added between the functional hidden layer and the output as a summarising element: this layer contains only two nodes, in order to summarise the activations produced by the functional hidden layer for analysis purposes. Note that if there was a single node in the summary layer, classification would not have been possible, but with only two nodes, it is possible to distinguish among an unlimited number of classes. Also see Figure 2.

3.2 Theoretical requirements for perfect classification

Consider a network that only has two nodes in its last hidden layer. At least two nodes are required to be able to differentiate among classes, but two nodes are sufficient to differentiate among an unlimited number of classes. Consider two samples c_s and c'_t from two different classes c and c', respectively. Limit the nodes in the last hidden layer to j and k, let a_{jc_s} be the activation value of the c_s sample from the jth node and w_{jc} the weight from node j to the output node associated with class c.

We can now define two useful values: the *activation gap* α is defined as the difference between activation values at a node:

$$\begin{aligned} \alpha_{jcc'} &= a_{jc_s} - a_{jc'_t} \\ \alpha_{kcc'} &= a_{kc_s} - a_{kc'_t} \end{aligned} \tag{7}$$

We therefore use $\alpha_{jcc'}$ as a shorthand for $\alpha_{jc_sc'_t}$, remembering that all α values are specific to a sample pair, and not a single value per class. The *weight gap* ϕ is defined as the difference between weight values anchored at the same node, and terminating in the outer layer:

$$\begin{aligned} \phi_{jcc'} &= w_{jc} - w_{jc'} \\ \phi_{kcc'} &= w_{kc} - w_{kc'} \end{aligned} \tag{8}$$

The weight gaps are not sample-specific, and truly has a single value for each node and pair of classes. These definitions are illustrated in Figure 3.

For any two samples from any two different classes c and c' to be classified correctly, the following must hold:

$$a_{jc}w_{jc} + a_{kc}w_{kc} > a_{jc}w_{jc'} + a_{kc}w_{kc'} \tag{9}$$

$$a_{jc'}w_{jc'} + a_{kc'}w_{kc'} > a_{jc'}w_{jc} + a_{kc'}w_{kc} \tag{10}$$

$$a_{jc}w_{jc} + a_{kc}w_{kc} > 0 \tag{11}$$

$$a_{jc'}w_{jc'} + a_{kc'}w_{kc'} > 0 \tag{12}$$

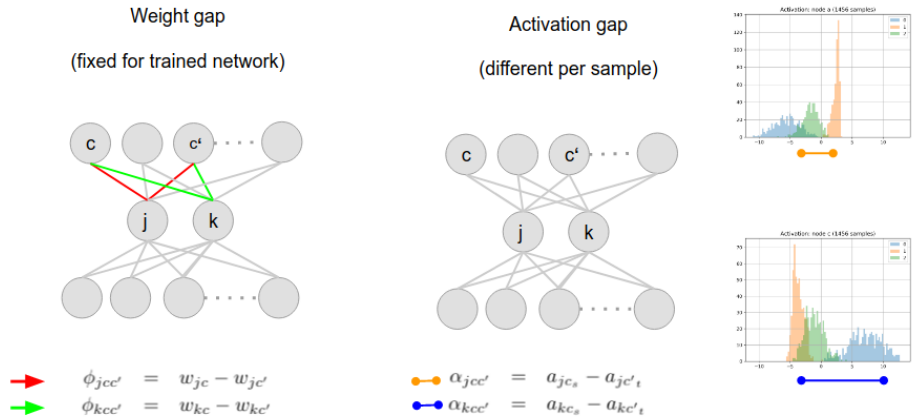


Fig. 3: Calculating the activation and weight gaps.

These equations can be combined:

$$
\begin{aligned}
a_{jc}w_{jc} + a_{kc}w_{kc} &> a_{jc}w_{jc'} + a_{kc}w_{kc'} \\
-a_{jc'}w_{jc} - a_{kc'}w_{kc} &> -a_{jc'}w_{jc'} - a_{kc'}w_{kc'} \\
\implies a_{jc}w_{jc} + a_{kc}w_{kc} - a_{jc'}w_{jc} - a_{kc'}w_{kc} &> a_{jc}w_{jc'} + a_{kc}w_{kc'} - a_{jc'}w_{jc'} - a_{kc'}w_{kc'} \\
\implies \alpha_{jcc'}w_{jc} + \alpha_{kcc'}w_{kc} &> \alpha_{jcc'}w_{jc'} + \alpha_{kcc'}w_{kc'} \\
\implies \alpha_{jcc'}w_{jc} - \alpha_{jcc'}w_{jc'} &> \alpha_{kcc'}w_{kc'} - \alpha_{kcc'}w_{kc} \\
\implies \alpha_{jcc'}\phi_{jcc'} &> \alpha_{kcc'}\phi_{kc'c}
\end{aligned} \tag{13}
$$

This does not mean that there is any asymmetry to the roles of j and k. The ratio can be rewritten in different ways, as the following are all equivalent:

$$
\begin{aligned}
\alpha_{jcc'}\phi_{jcc'} &> \alpha_{kcc'}\phi_{kc'c} \\
\alpha_{jcc'}\phi_{jcc'} &> -\alpha_{kcc'}\phi_{kcc'} \\
\alpha_{kcc'}\phi_{kcc'} &> \alpha_{jcc'}\phi_{jc'c}
\end{aligned}
$$

Using the definition of Equation 8 in Equations 9 and 10, it follows that

$$
\begin{aligned}
a_{jc}\phi_{jcc'} + a_{kc}\phi_{kcc'} &> 0 \\
a_{jc'}\phi_{jcc'} + a_{kc'}\phi_{kcc'} &< 0
\end{aligned} \tag{14}
$$

which means that, since the activation values ≥ 0, one of $\phi_{jcc'}$ or $\phi_{kcc'}$ must always be negative, and the other positive; which is similar to requiring that $\phi_{jcc'}$ or $\phi_{kc'c}$ must be positive (since $\phi_{kcc'} = -\phi_{kc'c}$). Combining this information with Eq. 13 it will always hold that for correctly classified samples, that:

$$\phi_{jcc'}\phi_{kc'c} > 0 \tag{15}$$

$$\alpha_{jcc'}\phi_{jcc'} > \alpha_{kcc'}\phi_{kc'c} \tag{16}$$

Following the reverse process (not shown here), Equations 15 and 16 then become the requirements for correct classification. Taking signs into account, Equation 16 can be restated as a ratio, with the required inequality depending on the signs of the separate elements. Correct classification then requires that in addition to Equation 15 either

$$\alpha_{jcc'}\alpha_{kcc'} < 0 \tag{17}$$

or, if $\alpha_{jcc'}\alpha_{kcc'} > 0$, then:

$$\frac{|\alpha_{jcc'}|}{|\alpha_{kcc'}|} \begin{cases} > \dfrac{\phi_{kc'c}}{\phi_{jcc'}} \text{ if } \phi_{jcc'}\alpha_{kcc'} > 0 \\ < \dfrac{\phi_{kc'c}}{\phi_{jcc'}} \text{ if } \phi_{jcc'}\alpha_{kcc'} < 0 \end{cases} \tag{18}$$

which needs to hold for all samples of all classes classified correctly, for every single c and c' pair. We define the *activation ratio (α-ratio)* as the term on the left, and the *weight ratio (ϕ-ratio)* as the term on the right.

Note that the weight ratio will always be positive, and is fixed for all samples, once the network has been trained. Since these values are set in the output layer, the role of the nodes up to the last hidden layer is to create activation gaps between samples, consistent with the established weight ratios. Since each gap is created simply by summing over all active nodes in the previous layer, nodes that are able to separate classes well are re-used, and their ability to separate classes, can be analysed by analysing the α values.

3.3 Network normalisation

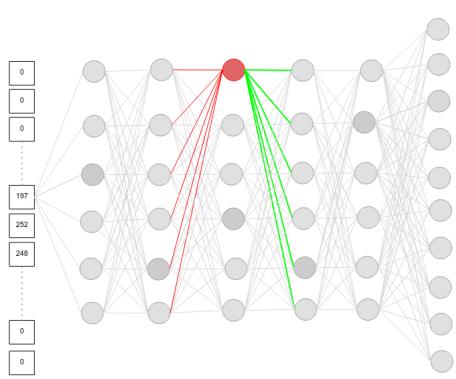


Fig. 4: If all in-fan weights at a node (red) are divided with the same value used to multiply all out-fan weights (green) with, network behaviour remains unchanged, but per-layer analysis would produce very different results.

After training and prior to any analysis, the network is normalised to remove potential cross-layer artefacts. Figure 4 demonstrates how it is possible to introduce artefacts that would invalidate any layer-specific analysis, without changing overall network behaviour. We therefore perform weight normalisation one layer at a time; normalising the in-fan weight vector per node, and passing this norm along to the out-fan weight vector of the same node at the next layer. Specifically, we calculate the node-specific norm of the in-fan weight vector at that node; and use this value to both divide the in-fan weight vector with, and multiply the out-fan weight vector with. This has the added benefit that all weights now have a norm of 1, which means that the activation values at any node are actually simply the projections of all the sample values onto the weight vector.

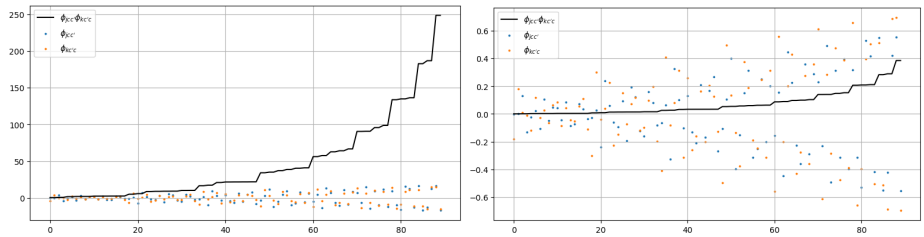
4 Empirical confirmation

In order to demonstrate the concept of generation gaps, we train ReLU-activated networks with the architecture of Figure 2 using the MNIST [10] dataset. We use a fairly standard training setup: initialising weights and biases with He initialisation [5]; optimising the Cross-Entropy loss with Adam as optimiser [9]; and performing a grid search of learning rates across different training seeds. No additional regularisation apart from early stopping is used. All hyper-parameters are optimised on a 5,000 sample validation set. We use the same protocol to train networks with hidden layers of 100, 300 and 600 hidden nodes, and select the sample networks listed in Table 1 for analysis. Results across the three architectures are very similar (and per architecture, identical before and after weight normalisation).

model	**training accuracy**	**test accuracy**
600-node	0.965	0.937
300-node	0.984	0.933
100-node	0.969	0.936

Table 1: Training and test accuracy of models with different number of hidden nodes. All models include a 2-node summary layer.

We confirm that the expected ratios hold in practice, by analysing the weight and gap ratios for correctly classified samples. We find that these ratios do indeed hold, as illustrated in Figures 5 to 7. Ratios were confirmed for all samples and networks, but in these figures we extract the weight and activation gaps for 300 random samples correctly classified by the 300-node model.



(a) unnormalised (b) normalised

Fig. 5: As anticipated from Equation 15, the value of $\phi_{jcc'}\phi_{kc'c}$ is always positive (black line) even though the individual weights can be positive or negative. Weight gaps are shown both before (left) and after weight normalisation (right).

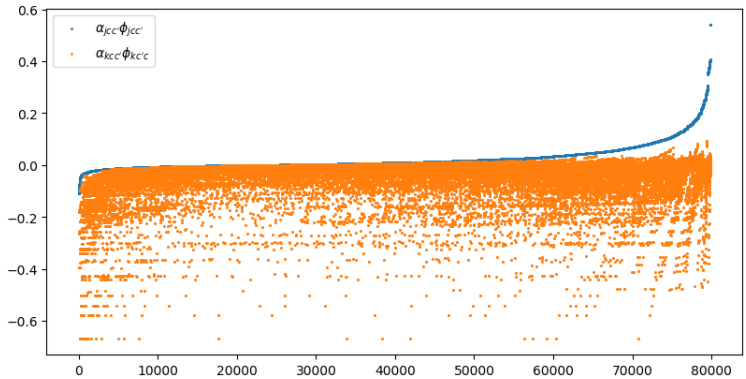


Fig. 6: For correctly classified samples, it always holds that the $\alpha_{jcc'}\phi_{jcc'}$ values (blue) are larger than the matching $\alpha_{kcc'}\phi_{kc'c}$ (orange), as expected from Equation 13. Weight-normalised values shown here.

5 Conclusion

We build on results from [2] which views DNNs as layers of collaborating classifiers, and probe the interplay between the local and global nature of nodes in a hidden layer of an MLP. We ask what the theoretical conditions are for perfect classification of a set of samples from different classes, and answer this question for a simplified architecture.

While the architecture is simplified, it is not trivial: a summary layer can be added to a pre-trained MLP, which makes the 2-node summary layer less of a restriction than it initially seems. Also, MLPs with multiple hidden layers can be considered as consisting of multiple 3-layer sub-networks stacked on top of each other; it is not immediately clear to what extent the same factors that are important for a 3-layer MLP are important for a sub-network within a larger structure. Recurrence and residual connections are problematic for theoretical

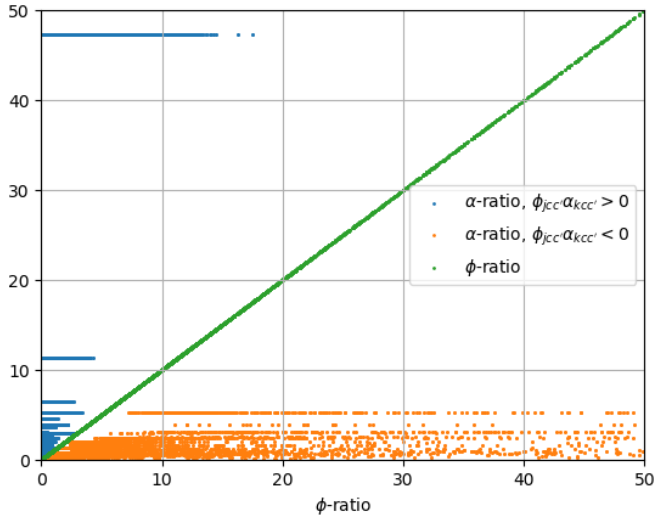


Fig. 7: Weight ratios (green) plotted with matching activation ratios under the two conditions specified by Equation 18. The activation ratio is larger when $\alpha_{jcc'}\alpha_{kcc'} > 0$ (blue) and smaller when $\phi_{jcc'}\alpha_{kcc'} < 0$ (orange). All values after weight normalisation; both axes limited to show detail.

analysis, but convolutional layers can to a large extent be analysed as sparse, connected-weight MLPs. All in all, while it should be possible to extend the ideas of this paper to more complex architectures, our first goal is to fully understand the generalisation ability of straightforward ReLU-activated MLPs,

For a simplified MLP, we have shown how activation gaps are formed at node-level, and how the consistency of these gaps gives rise to the classification ability of a layer. Specifically, nodes act as 'gap generators' between samples. These gaps are re-used across multiple samples and their manner of use sheds light on the characteristics of nodes that we expect to better support generalisation. In this paper, we do not yet use either the gaps or the ratios to probe the networks themselves: How are gaps distributed across nodes? How are nodes (and the gaps they create) re-used? From the interaction between sample sets and gaps, are some nodes more general and others more specific? What does this say about the generalisation ability of the network? While we have not answered any of these questions, we have developed a conceptual tool that can be used to probe networks for answers to questions such as these.

6 Acknowledgements

This work was partially supported by the National Research Foundation (NRF, Grant Number 109243). Any opinions, findings, conclusions or recommendations expressed in this material are those of the author and the NRF does not accept any liability in this regard.

References

1. Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. In: Advances in Neural Information Processing Systems 30. pp. 6240–6249 (2017)
2. Davel, M.H., Theunissen, M.W., Pretorius, A.M., Barnard, E.: DNNs as layers of cooperating classifiers. In: AAAI Conference on Artificial Intelligence, *accepted for publication* (2020)
3. Dinh, L., Pascanu, R., Bengio, S., Bengio, Y.: Sharp minima can generalize for deep nets. arXiv preprint (also ICML 2017) **arXiv:1703.04933v2** (2017)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
5. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. arXiv preprint (also ICCV 2015) **arXiv:1502.01852** (2015)
6. Jiang, Y., Krishnan, D., Mobahi, H., Bengio, S.: Predicting the generalization gap in deep networks with margin distributions. arXiv preprint (also ICLR 2019) **arXiv:1810.00113v2** (2019)
7. Kawaguchi, K., Pack Kaelbling, L., Bengio, Y.: Generalization in deep learning. arXiv preprint **arXiv:1710.05468v5** (2019)
8. 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. arXiv preprint (also ICLR 2017) **arXiv:1609.04836** (2016)
9. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR) (2014)
10. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
11. Montavon, G., Braun, M.L., Müller, K.R.: Kernel analysis of deep networks. Journal of Machine Learning Research **12**, 2563–2581 (2011)
12. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N.: Exploring generalization in deep learning. In: Advances in Neural Information Processing Systems 30. pp. 5947–5956 (2017)
13. Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., Sohl-Dickstein, J.: On the expressive power of deep neural networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML). vol. 70, pp. 2847–2854 (2017)
14. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv preprint **arXiv:1703.00810** (2017)
15. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (1989)
16. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint **arXiv:1708.07747v2** (2017)
17. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. arXiv preprint (also ICLR 2017) **arXiv:1611.03530** (2016)

A Ratios for unnormalised networks

Figures 6 and 7 were produced after weight normalisation. Here we include the pre-normalisation results: patterns are similar, while scale is different.

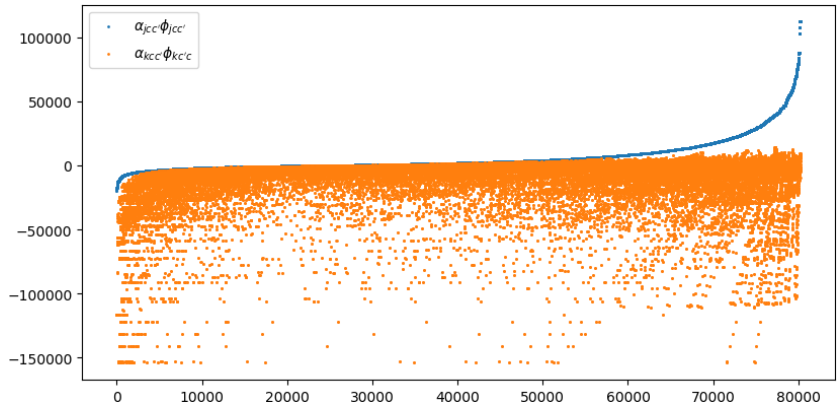
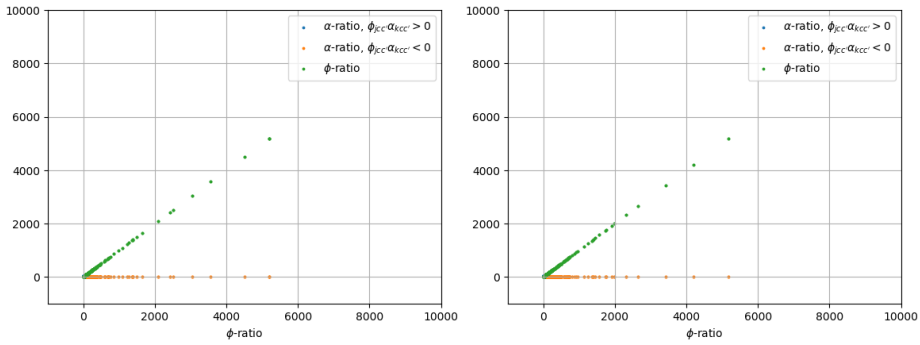


Fig. 8: The same analysis as in Figure 6 but repeated before weight normalisation.



(a) unnormalised (b) normalised

Fig. 9: A more complete version of Figure 7, before and after weight normalisation. Note that the α-ratios tend to be very small, when compared with with the more evenly distributed ϕ-ratios.