Counting the Paths in Deep Neural Networks as a Performance Predictor

Anonymous authors
Paper under double-blind review

Abstract

We propose a novel quantitative measure to predict the performance of a deep neural network classifier, where the measure is derived exclusively from the graph structure of the network. We expect that this measure will become a fundamental first step in developing a method to evaluate new network architectures and reduce the reliance on the computationally expensive trial and error or "brute force" optimisation processes involved in model selection. The measure is derived in the context of multi-layer perceptrons (MLPs), but the definitions are shown to be useful also in the context of deep convolutional neural networks (CNN), where it is able to estimate and compare the relative performance of different types of neural networks, such as VGG, ResNet, and DenseNet. Our measure is also used to study the effects of some important "hidden" hyper-parameters of the DenseNet architecture, such as number of layers, growth rate and the dimension of 1×1 convolutions in DenseNet-BC. Ultimately, our measure facilitates the optimisation of the DenseNet design, which shows improved results compared to the baseline.

1 Introduction

Deep neural networks (DNN) have achieved outstanding results in several classification tasks (Huang et al., 2017; He et al., 2016). There is some theoretical understanding of the workings of individual elements, such as convolutional filters, activation funtions, and normalisation (Goodfellow et al., 2016; LeCun et al., 2015; Schmidhuber, 2015). However, current ideas behind the DNN graph design are still based on ad-hoc principles (Mishkin et al., 2017). These principles are largely qualitative and tend to improve classification accuracy – examples of these principles include: an increase of the network depth (Szegedy et al., 2015), and an increase of the representation dimensionality (by, for example, expanding the number of channels in deeper parts of the DNN) (Huang et al., 2017; He et al., 2016). We notice that an effective DNN graph design is largely independent of the data set, as long as the type of data (e.g., images) and task (e.g., classification) are similar. Hence, we argue that good design principles can be encoded in a quantitative measure of the graph and should be justified by a quantitative assessment of the DNN architecture performance. An alternative way of designing a DNN graph structure is based on (meta-)optimisation methods (Jenatton et al., 2017; Kandasamy et al., 2019; Mendoza et al., 2016; Snoek et al., 2012). Although useful in practice, such optimisation methods add little to our understanding of the design principles of new DNN graphs and are computationally challenging to execute.

DNNs form a hierarchical structure of filters that can be seen as a directed graph. The first layers of this graph contain neurons that are active for low level patterns, such as edges and patches (in the case of images) (Zeiler & Fergus, 2014), while deeper layer neurons are active for more complex visual patterns, such as faces or cars, formed by a hierarchical combination of a large number of simpler filters (Zeiler & Fergus, 2014). In such representation, each neuron behaves like a binary classifier of the visual pattern learned by the neuron. Also, the strength of the activation is related to how well the pattern is matched.

We argue that this linear separability promoted by the neurons is the key ingredient behind an effective quantitative measure of model performance. In this paper, we introduce a new measure that can be a proxy for DNN model performance. This proposed measure is first formulated in the context of Multi Layer Perceptrons (MLPs) to quantitatively predict the classification accuracy of the model. Then, we extend the applicability of the measure to predict the classification accuracy of the following CNNs: VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016) and DenseNet

(Huang et al., 2017). The experiments demonstrate how this quantity can be used to predict the "correct" depth of a simple feed forward DNN with constraints on the parameter budget. The experiments also show how the proposed quantity can be used to improve the design of DenseNet (Huang et al., 2017) and in the study of the effects of some important "hidden" hyper-parameters such as the dimension of 1×1 convolutions in the bottlenecks of DenseNet-BC, the number of layers, and the growth rate.

2 METHOD

We first define the proposed measure using discrete calculations, then we relax the computation with continuous mathematics to formulate an optimisation problem for the proposed quantity. We then make some assumptions that facilitate the calculation and optimisation of the measure. We close the section showing how we can relax these assumptions without loosing predictive power.

2.1 DISCRETE PATH COUNTING

The goal of DNN models is to maximise the chances to have a large number of useful patterns recognised by each layer. In each layer, new low level patterns must be recognised without losing the ability to separate patterns that were separable in previous layers. One possible way to realise such goal is by increasing the dimension of the representation (Vapnik, 2013). This argument can be quantified by computing the number of $paths$ that affect a neuron in layer k as $\prod_{i<k} N_i$, where N_i is the number of nodes or channels in layer i, i.e., nodes for multi-layer perceptrons (MLP) and channels for convolutional neural networks (CNN). Such definition resembles a concept called forward path entropy that studies the hierarchy of complex networks (Corominas-Murtra et al., 2013). We argue that modern CNNs are man-made complex networks developed incrementally by a large number of researchers, and so comply with many of the assumptions behind complex networks. A measure of number of paths in DNNs was proposed by Veit et al. (2016) to explain the effectiveness of residual networks (ResNet) (He et al., 2016), their work counts the paths of the residual connections, which is different from our approach.

The maximisation of the number of $paths$ defined above, given a fixed number of parameters, implies that the optimal design would be of a very deep network, where each channel (or node) has size 2. Unfortunately, this clearly does not match the collective wisdom of the field regarding the design of deep learning models. In this paper, we aim to propose an alternative measure that can explain more effectively the superior performance of recently proposed DNNs.

State-of-the-art DNNs can be represented by a graph that takes in data with a small number of channels for CNNs (e.g., 3 channels for colour images) or a small number of input neurons for MLPs, and after each layer, it expands the dimension of this representation by increasing the number of channels or neurons. Also when the number of neurons does not increase, the effectively used dimensions of the representation is likely to increase nonetheless. Increasing the dimension of the representation helps in classification tasks because it makes samples belonging to different classes "more" linearly separable – this is analogous to the kernel trick argument (Vapnik, 2013). Consequently, we redefine $paths$ to be the *number of additional channels or neurons* between two consecutive layers, as in $path_i = N_i - N_{i-1}$. We justify this definition by arguing that for each additional layer i, only the "extra" dimensions ($N_i - N_{i-1}$) are beneficial to enable new linearly separable patterns without losing the separability of patterns already learned by earlier layers. These extra dimensions allow to embed the data encoded in the previous layer in a higher dimensional space, where classes become more linearly separable (Cover, 1965; Budinich, 1991; Fink et al., 2017).

For MLPs, we propose the following objective function to maximise:

$$paths = \prod_i (N_{i+1} - N_i), \tag{1}$$

subject to a fixed number of parameters $P = \sum_i N_{i+1}N_i$, disregarding the biases for simplicity. This measure can be extended for CNNs by replacing N_i with $N_i \times$ filter width $\times$ filter height. To make the calculation of the paths manageable, following the same approach taken in the definition of entropy, we take the $\log(.)$ of the objective in equation 1:

$$Z = \log(paths) = \sum_i \log(N_{i+1} - N_i). \tag{2}$$

Finally, for the definition to work in cases where the number of channels does not change (or decreases) for a few layers, we consider the upper limit of the paths calculated under the constrain of the available neurons:

$$Z = \sup_{\{D_i | D_i \leq N_i\}} \sum_i \log\left(D_{i+1} - D_i\right), \tag{3}$$

where the optimal values for D_i can be trivially found. For instance, for a block of layers that have the same number of channels N_i, the values of D_i optimising Z in equation 3 grow linearly between D_0 and D_{last}, with D_0 typically representing the number of input channels.

2.2 Continuous Path Counting

In the limit of large number of layers, it is easier to optimise Z in equation 2 as a continuous function of the layer depth. Then, i becomes the continuous variable λ (with Λ denoting the total number of layers), where the discrete N_i is approximated by the continuous variable $n(\lambda)$. Replacing the summations with integrals, the total number of parameters is defined by:

$$P(\Lambda) = \int_0^{\Lambda} n^2(\lambda) d\lambda, \tag{4}$$

and the $\log$ of the number of paths is

$$Z(\Lambda) = \int_0^{\Lambda} \log\left(\frac{dn(\lambda)}{d\lambda}\right) d\lambda. \tag{5}$$

From equation 4, we have $d\lambda = dp/n^2$, so equation 5 can be re-written as

$$Z(P) = \int_0^{P} \frac{\log\left(n^2 \frac{dn}{dp}\right)}{n^2} dp. \tag{6}$$

In the next section we show that, by making weak assumptions about the network structure, we can obtain a closed form solution for Z from equation 6.

Assuming that each layer has $(1+\alpha)$ times the number of channels of the previous layer, i.e., $N_{i+1} = N_i \times (1+\alpha)$, means that for the continuous case we have $n(\lambda) = n_0 e^{\alpha\lambda}$, where n_0 is the number of channels in the first layer. This allows us to find a closed-form solution for equation 6 (explicit calculations are presented in Appendix A.1),

$$Z(\alpha, n_0, P) = \frac{\log(\alpha n_0)\log\left(\frac{2\alpha P}{n_0^2} + 1\right)}{2\alpha} + \frac{\log\left(\frac{2\alpha P}{n_0^2} + 1\right)^2}{8\alpha}. \tag{7}$$

If we set the number of parameters to $P = 35 \times 10^6$ as in ResNet101 (He et al., 2016), the maximum value for Z is at $\alpha = 0.0178$ with the number of channels in the first layer $n_0 = 56.5$. This means that the total number of layers is $\Lambda = 167.5$ and the optimal size of the last layer is $n(\Lambda) = 1113$ (Fig. 1). These numbers are remarkably close to the actual hyper-parameter values for large CNN architectures, such as ResNet101 (He et al., 2016), trained on large image classification tasks (Deng et al., 2009). This suggests that large image classification tasks (Deng et al., 2009) may not benefit from larger and deeper neural networks with a simply-connected feeedfoward architecture. This also suggests that to really exploit models bigger than the current state-of-the-art CNNs, we will need problems more complex than the 1000-class ImageNet (Deng et al., 2009). For the case above with a fixed P, the maximum value for Z is 251. This number corresponds to 10^{109} paths, which represents an extremely large number of ways to activate the output neurons – this can qualitatively explain how CNNs can learn complex tasks.

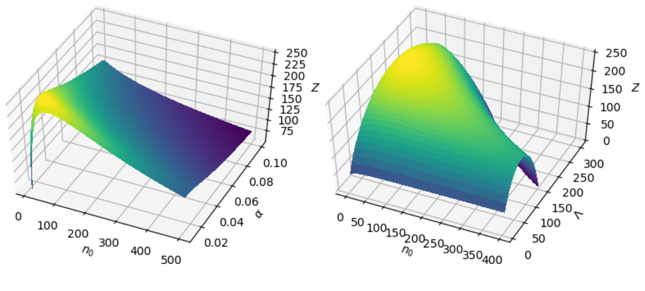


Figure 1: Z as a function of α and n_0 (left – computed from equation 7) and as a function of Λ and n_0 (right – computed from equation 7 and inverting equation 10 from the Appendix A.1).

By relaxing the constrain on the functional form of $n(\lambda)$, defined above, we can rewrite equation 6 as: $Z(n) + \delta Z(n, \delta n) = Z(n + \delta n)$. A necessary condition for Z being an optimum over the space on functions n is $\delta Z = 0$. This imposes a set of conditions on $n(\lambda)$ which is equivalent to a differential equation and boundary conditions at $\lambda = 0$ and $\lambda = \Lambda$. Solving numerically this differential equation, as shown in details in Appendix A.2, results in a solution qualitatively very similar to an exponential, confirming the validity of the analytical results obtained above.

2.3 How to Calculate Z for Convolutions of Mixed Sizes

The contribution to the paths for convolutions are proportional to the filter width and height, as explained in Sec. 2.1. For example, if the filter has size 3×3, the contribution to Z is 9 times what it would be for a filter of size 1×1. In the case of the DenseNet-BC bottleneck block, we have first a layer of 1×1 and then a layer of 3×3 convolutions (Fig. 5a, middle). In the original DenseNet paper (Huang et al., 2017), the number of channels in the 1×1 convolutions is $r = 4$ times the number of channels in the 3×3 convolutions. Therefore, this particular block has $Z = \log((rk)(9k - rk)/9)$, where the 1×1 layers contributes rk, because its input has a skip connection, and the 3×3 layer contributes $(9k - rk)/9$ because each convolution has 9 inputs and its input has no skip connection. If we optimise r for the highest Z, we find $r \gtrsim 4.5$, with small r favored by economy of parameters. This explains well the value of 4 used in the original DenseNet-BC (Huang et al., 2017). Increasing r further is expected to bring little return in performances, as Z does not increase and the network ends up having unnecessary channels in the 1×1 filters.

3 Literature Review

There is a vast collection of papers that describe the ”tricks of the trade”, consisting of suggestions about network structures, activation functions, different types of network layers, etc. (Goodfellow et al., 2016; LeCun et al., 2015; Schmidhuber, 2015). Even though such suggestions are useful from a practical point of view, they are fundamentally qualitative descriptions that lack rigour and offer little insight into the design of new classification models. Some other papers have aimed to characterise the performance of deep learning model in a post-hoc manner, and consequently offered little help in the design of new deep learning models (Keskar et al., 2016; Lee et al., 2016; Liao et al., 2018; Littwin & Wolf, 2016; Sagun et al., 2016; Soudry & Carmon, 2016).

An alternative to a proper understanding of the DNN graph structure is to turn the problem of designing new models into an optimisation task. Unfortunately, this task often depends on what is infamously known as the grad student optimisation: a gruesome trial and error of innumerous DNN models (Kandasamy et al., 2019). Thankfully, smarter approaches have been developed, such as Bayesian optimisation (Jenatton et al., 2017; Kandasamy et al., 2019; Mendoza et al., 2016; Snoek et al., 2012), reinforcement learning (Zoph & Le, 2016), evolutionary methods (Liu et al., 2017), and random optimisation (Li & Talwalkar, 2019). These optimisation methods are useful, but still suffer from scalability and robustness issues to a certain extent. Also, when the network is finally optimised, there is hardly any insight on why the selected model was chosen.

What we propose in our paper addresses these two issues: 1) we propose a method that increases our current understanding of how deep learning methods work, and 2) our method does not rely on computationally expensive optimisation processes to enable the design of a potentially interesting model. Therefore, such increasing of our understanding can potentially improve our ability to propose new structures that have higher chances of succeeding at producing more effective DNNs. Also, the low run-time and memory costs involved in the calculation of our proposed measure makes it significantly more efficient in terms of computational costs than the optimisation approaches mentioned above. The main challenge involved in our approach is that the computation of Z requires a non-trivial understanding of the model to design the formula to compute Z. We provide general guidelines on how to formulate Z in Sec. 5

4 Experiments

We first run experiments using simple setups, in terms of data set and models, to show the effectiveness of our proposed measure. Then, we run an experiment that compares the proposed measure Z for several DNN models: VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016) and DenseNet (Huang et al., 2017). We then optimise the DenseNet architecture to maximise our measure Z. All experiments are run on two data sets. Fashion-MNIST (Xiao et al., 2017) is a 10-class data set that has a training set of 60,000 grey scale images and a testing set of 10,000 images, where each image has 28×28 pixels. CIFAR-10 (Krizhevsky & Hinton, 2009) has 10 classes and a total of 50,000 32×32 training colour images and 10,000 testing images – this data set is extended to CIFAR-10+, which relies on standard data augmentation (mirroring and shifting) to build the training set (He et al., 2016). All data sets are balanced, i.e., they have the same number of training images per class. We train the models for the FashionMNIST experiments with sgd+momentum for 30 epochs, learning rate 0.001, momentum 0.9, and mini-batch size of 100. We follow the experimental setup suggested by Huang et al. (2017) for our dense-net experiments on CIFAR-10+: 300 epochs, initial learning rate of 0.1 divided by 10 at epoch 150 and again at epoch 225.

4.1 Experiments with MLP on CIFAR-10

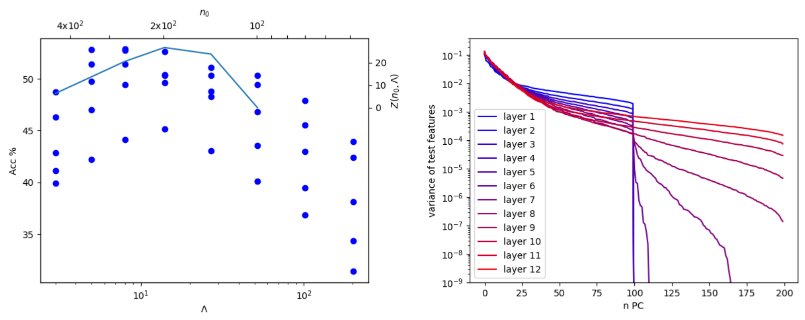


Figure 2: (Left) Accuracy and Z as a function of number of layers for a fixed parameter budget using a simple MLP on PCA whitened CIFAR10. The top axis shows the corresponding number of neurons of the first layer (n_0). The multiple points for each model are the accuracy values at epochs 5, 10, 15, 30 and 50 for the optimal choice of initialisation and learning rate. The solid line represents Z calculated assuming linear growth of effective neurons (equation 3). (Right) Variance of PCA decomposition of the test set features at different layers depth of the best performing MLP (largest Z on the left panel). The effective dimensionality of the representation grows from 100 (input) to 200 (output) gradually using all the available neurons.

We start with simple experiments involving MLPs on CIFAR-10. The goal of this experiment is to show an intuitive result to most deep learning practitioners: assuming a fixed parameter budget, the optimal network structure shows a compromise between depth and width. We follow closely the architectural choices suggested by Pennington et al. (2017). Our networks are MLPs containing 500K parameters with a depth in a range from few layers up to 800 layers. As shown by Pennington et al. (2017), the training of such architectures involves an optimal choice of initialisation method and learning rate. In particular, we will use the empirical relation for the dependency of the optimal learning rate with depth ($1/L$) and the theoretical result for the optimal variance of the weight initialisation ($1 - \sigma_w^2 \simeq 1/L$). We explore a range of values around the ones suggested by such relations to make sure that we pick the optimal values for the initialisation method and learning rate. Also, we train all models with weight decay of 10^{-5}. To simplify the problem, we pre-process CIFAR-10 by running PCA on the training set and keeping the first 100 whitened PCs. Fig. 2(left) shows that for a given budget of parameters, the maximum trainable depth is obtainable by optimising Z (equation 3). The best models train to an accuracy comparable to what is obtained by Pennington et al. (2017). Fig. 2(right) shows the growth of dimensionality (illustrative of D_i in equation 3) of the feature representation in a trained network, starting with 100 dimensions up to the size of the last layer.

4.2 Estimation of Z on a Simple Feed-forward CNN on FashionMNIST (Xiao et al., 2017)

In this section, we try our proposed measure Z using a simple CNN, consisting of a variable number of layers Λ, a variable number of channels in the first layer n_0, fixed filter sizes of 3×3 and final fully connected layer before the softmax activation, and a number of channels that grows exponentially with the layer index according to the function $n_i = n_0 e^{\alpha i}$. Figure 3

Table 1: Z and P values, and top-1 error results for the deep learning models VGG16 (Simonyan & Zisserman, 2014), ResNet34Plain (He et al., 2016), ResNet34 (He et al., 2016), WRN-18-1.5 and WRN-34-1.5 (Zagoruyko & Komodakis, 2016), and DenseNet121-BC (Huang et al., 2017) as reported from the original papers on ImageNet (Deng et al., 2009).

	VGG16	ResNet34Plain	ResNet34	WRN-18-1.5	WRN-34-1.5	DenseNet121-BC
Z	77	82	120	73	133	448
P	$138M$	$21M$	$21M$	$26M$	$47M$	$8M$
Top-1 1crop	–	–	26.77	27.06	24.5	–
Top-1 10crops	28.7	28.54	25.03	–	–	23.61

shows the accuracy of the models (multiple accuracy results at each value of Λ and n_0 are obtained from results of epochs between 10 and 30) and the value of Z as a function of depth Λ. In all experiments, the number of parameters of all networks is constrained to be $P = 5 \times 10^5$. For most models, the computed Z from equation 2 is shown to be a good predictor of the accuracy of the models, supporting the idea that optimising the network architecture to have a large Z is beneficial to get an optimal design.

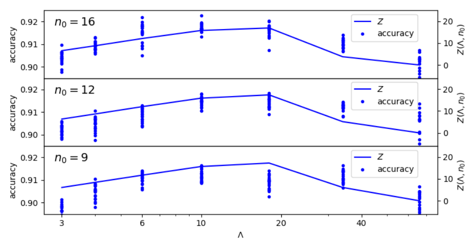


Figure 3: Accuracy and Z as a function of depth Λ using a simple CNN on FashionMNIST (Xiao et al., 2017). The multiple points represent the results from epochs between 10 and 30. The solid line is Z calculated with equation 3.

4.3 Estimation of Z on Standard CNNs: VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016) and DenseNet (Huang et al., 2017)

We estimate Z on several "real world" CNN architectures trained and tested on ImageNet (Deng et al., 2009). For VGG (Simonyan & Zisserman, 2014) and ResNet (He et al., 2016), the number of layers does not grow within each block, so we rely on equation 3 and the results observed in Sec. 4.1 to estimate value of Z. Recall from Sec. 4.1 that the number of channels effectively used by the data in the network grows gradually between the input dimension of the block N_{in} and the output N_{out}. A linear growth in the number of channels used (D_i) within each block gives the optimal value for Z. For a block with m layers in a standard feed forward CNN, such as VGG16, we have $Z = m \log \Delta N$, where $\Delta N = \frac{N_{out} - N_{in}}{m}$ with the number of parameters defined by $P = m N_{out}^2$. For a block of the ResNet, we have one skip connection every two layers, so the contribution to the number of paths is $\sim \log N$ for the layer with skip connection, and $\sim \log \Delta N$ for the layer without skip connection, where we again use that that the optimal configuration for D_i grows linearly between N_{in} and N_{out}. For ResNet, the number of parameters is $P = m N_{out}^2$.

In a DenseNet block, the number of channels per layer is equal to the growth-rate k. The dimensionality of the data is equal to the growth rate plus the dimensions of the previous layer. In each layer $\Delta N = N_i - N_{i-1} = k$, so $Z = m \log k$. For the DenseNet block, the number of parameters is defined by: $P = (N_{in} + mk)^2 + \sum_{0 \leq i < m} (N_{in} + ik)k = (1+2)mkN_{in} + (1+1/2)m^2k^2 + N_{in}^2$. This means that if $N_{in} < mk$, the number of parameters grows quadratically with m and k. Also, Z grows linearly in depth and logarithmically in growth-rate. Qualitatively, this means that the optimal design has a small growth-rate and a depth that is not too high (to avoid a quadratic growth of the number of parameters). This imposes a limit for the maximum depth of each block ($m_{max} \sim N_{in}/k$), which is, empirically, close to how the DenseNet architectures have been designed. This principle can be used to explain the effectiveness of DenseNet(-BC) (Huang et al., 2017) blocks and to optimise the design of new DenseNet(-BC) variants.

For all models above, we show how to compute the Z value per block. Therefore, we also need to take all the blocks into account when computing the final values for Z. Table 1 shows the Z values computed for various networks recently proposed in the field, along with their number of parameters P and classification error results on ImageNet (Deng et al., 2009). The quantity Z is a good predictor for the accuracy of the networks, for a given parameter budget P.

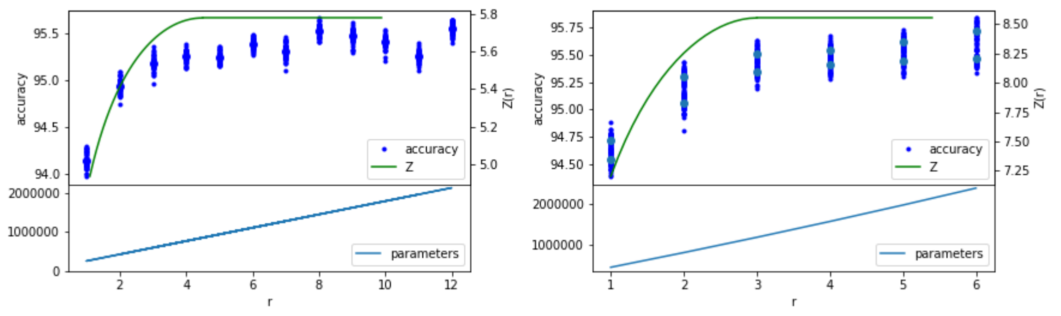
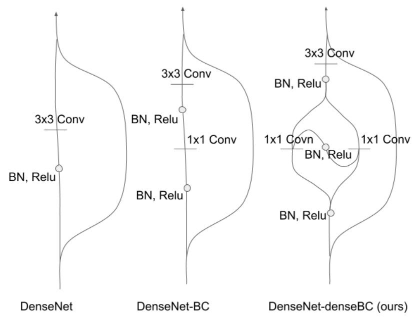


Figure 4: (left) Accuracy and Z for different values of the r ratio on a DenseNet-BC bottleneck block on CIFAR-10+ (Krizhevsky & Hinton, 2009). The horizontal axis r denotes the ratio of number of layers between the 1×1 convolution and the 3×3 convolution. The graph at the bottom shows the number of model parameters as a function of r. The green line shows the value for Z as a function of r, where the peak is reached for $r \approx 4.5$, as predicted in Sec. 2.3 (3 for DenseNet-Dense_BC, right). For larger values of r, adding channels to the 1×1 convolution does not contribute to the value of Z (solid horizontal line). The larger blue points in the graph on the top shows the accuracy of trained model, where the several smaller points represent the last 50 epochs of the training. (right) Study of accuracy of our proposed DenseNet-dense_BC bottleneck block (analogous to the left panel).



(a) On the left, we show a DenseNet block; the middle graph displays the bottleneck block of DenseNet-BC. On the right, our proposed redesigned bottleneck block with the additional dense connection to improve the value for Z, as explained in Sec. 4.4.

Architecture	P(mil.)	Top-1
”Small” models		
DenseNet-40-12	1.0	5.24
DenseNet-BC-100-12	0.8	4.51
DenseNet-148-3 (ours)	1.17	4.87
DenseNet-denseBC-100-12 (ours)	1.17	**4.28**
”Large” models		
DenseNet-BC-250-24	15.3	3.62
DenseNet-BC-190-40	25.6	3.46
DenseNet-denseBC-250-24 (ours)	22.6	**3.32**

(b) P and top-1 error results of DenseNet architectures on CIFAR-10+ data set (Krizhevsky & Hinton, 2009).

Figure 5: The DenseNet-DenseBC architecture and experimental results.

We also compare accuracy and Z in Fig. 4 on CIFAR-10+ (Krizhevsky & Hinton, 2009), which shows the accuracy of models trained with different values of the ratio r for a DenseNet-BC bottleneck block. In general, it is observed that the accuracy of the model saturates for a ratio larger than ≈ 4.5. Recall from Sec. 2.3, that Z is maximised for $r \geq 4.5$, and the accuracy remains relatively high for $r > 4.5$. We conjecture that regularisation techniques applied during training reduces the number of “effective” channels, corresponding to the set of D_i values that maximise Z. Therefore, Z explains when increasing further the number of channels (and parameters) of a 1×1 layer of a bottleneck layer will not yield improved performances.

4.4 Optimise the Design of a DenseNet block

To test the usefulness of our measure for the design of a new DenseNet variant, we optimise the block of a DenseNet-40-12 to increase Z (initially for this model, $Z = 89$) – this is the smallest of the DenseNet architectures proposed by Huang et al. (Huang et al., 2017). The optimisation is based on the reduction of the growth rate and increase of the number of layers by the same factor. This approach is chosen because it is a simple way to increase Z without changing the number of input and output channels (boundary conditions of the blocks). The first architecture that we propose is a DenseNet-148-3 with 148 layers and growth rate of 3, which produces $Z = 148$ that compares favourably with the original value. The number of parameters and classification results for the new and original models are shown in the table of Figure 5b.

We propose another optimisation of the structure of the bottleneck blocks of the DenseNet-BC architecture. Like before, we focus on optimising one of the smallest elements of the graph, which, we argue, should be largely independent from the data set. In the original architecture, the BC block is built with a 1×1 convolutional layer followed by a 3×3 convolutional layer, giving $Z = (rk)((9k - rk)/9)$. The 1×1 convolutional layer has 4 times more channels than the corresponding 3×3 convolutional layer. We replace the single 1×1 layer with two 1×1 layers that are densely connected, as shown in Fig. 5a on the rightmost panel. With the new design, the number of paths becomes larger, with $Z = \log\left((rk)(rk)(9k - 2rk)/9\right)$, where rk is the number of channels of each 1×1 layer. The optimum Z for the minimum number of parameters is achieved with $r = 3$ – results are displayed in the table of Figure 5b, where we show two different proposed versions of the DenseNet-BC architecture. In Figure 4 (right), we show the accuracy of different versions of this architecture as a function of r. Similarly to what happens to the simpler DenseNet-BC, the accuracy of the model increases significantly up to the optimal r, and then remains stable. This behaviour strengthens our conjecture that regularisation techniques effectively reduce the number of channels and parameters of the network. We finally performed a single experiment with the DenseNet-denseBC-250-24, $r = 3$ (last line in the table of Fig. 5b) improving over the baseline and DenseNet-BC-190-40, which represents the most accurate model from (Huang et al., 2017).

5 Discussion and Conclusions

Our measure (Z) is calculated based only on the graph structure of the model and assumes that the initialisation and training strategies are close to optimal. We argue that the choice of these strategies is important because it guarantees the realisation of the model potential for a particular classification problem. However, we see these strategies as somewhat orthogonal to network design. We show in Fig. 3 that the value of Z is a good predictor of the optimal depth Λ for a simple feed forward CNN. We also show that the value of Z is a good predictor of accuracy for small values of r (channels of bottleneck layer), and for larger values of r, while Z "saturates", the accuracy tends to remain stable. We conjecture that this happens because regularisation techniques are likely to reduce the effective number of channels when this allows an increase in the number of paths. We find indicative support of this mechanism from the study of the dimensionality of the PCA decomposition of MLP classifiers (Fig. 2(right)). We will leave the proof for this conjecture for a future work, note that this behaviour does not undermine the method – the implementation of models with optimal Z appears to offer best performance with the minimum number of parameters.

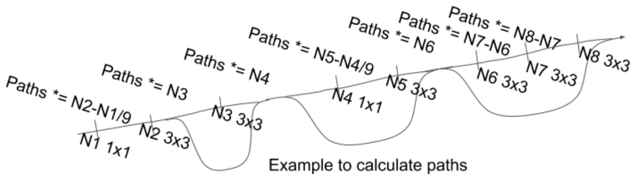


Figure 6: This example shows the rules to calculate the contribution to the total number of paths given by each layer in a NN. Each N_i can be $\leq$ than the corresponding channels in the NN.

We now provide guidelines on how to formulate Z – see Fig. 6. When there is a skip connection around a layer, the number of channels in such layer can contribute fully to the paths (N_i). When two layers are in sequence, the contribution to the paths is the difference of the channels after the reduction due to regularisation ($N_i - N_{i-1}$). When the filter size changes (e.g., 1×1 followed by 3×3 convolution), the channels of the first layer need to be divided by the ratio of the dimension of the filters ($\frac{3\times3}{1\times1}$). In the common case of convolution blocks with constant number of channels, the supremum of eq 3 gives that the optimal D_i's grow linearly between N_{in} and N_{out}.

In conclusion, in this manuscript we propose a quantitative principle for predicting the accuracy of a deep learning classifier that takes into account the structure of the DNN graph. We hope that this will be the basis for more general principles for network design and it will lead to a deeper understanding of DNNs. As a practical application of this work, we also propose a new DenseNet-denseBC architecture that improves over baseline results.

References

Marco Budinich. On linear separability of random subsets of hypercube vertices. *Journal of Physics A: Mathematical and General*, 24(4):L211, 1991.

Bernat Corominas-Murtra, Joaquín Goñi, Ricard V Solé, and Carlos Rodríguez-Caso. On the origins of hierarchy in complex networks. *Proceedings of the National Academy of Sciences*, 110(33): 13316–13321, 2013.

Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, pp. 326–334, 1965.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Martin Fink, John Hershberger, Nirman Kumar, and Subhash Suri. Hyperplane separability and convexity of probabilistic point sets. *Journal of Computational Geometry*, 8(2):32–57, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1655–1664. JMLR. org, 2017.

Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. *arXiv preprint arXiv:1903.06694*, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pp. 1246–1257, 2016.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

Zhibin Liao, Tom Drummond, Ian Reid, and Gustavo Carneiro. Approximate fisher information matrix to characterise the training of deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Etai Littwin and Lior Wolf. The loss surface of residual networks: Ensembles and the role of batch normalization. *arXiv preprint arXiv:1611.02525*, 2016.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards automatically-tuned neural networks. In *Workshop on Automatic Machine Learning*, pp. 58–65, 2016.

Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*, 161:11–19, 2017.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pp. 4785–4795, 2017.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pp. 550–558, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A Appendix

A.1 Simplified solution of continuous counting

If we assume that each layer has $(1+\alpha)$ the number of channels of the previous one:

$$N_{(i+1)} = N_i(1+\alpha) \tag{8}$$

the continuous case becomes:

$$n(\lambda) = n_0 e^{\alpha\lambda}. \tag{9}$$

Integrating Equations 4 and 6 we obtain:

$$\Lambda = \frac{\log\left(\frac{2\alpha P}{n_0^2}+1\right)}{2\alpha} \tag{10}$$

and

$$Z(\alpha, n_0, \Lambda) = \Lambda \log \alpha n_0 + \frac{\alpha \Lambda^2}{2} \tag{11}$$

Finally, substituting:

$$Z(\alpha, n_0, P) = \frac{\log(\alpha n_0) \log\left(\frac{2\alpha P}{n_0^2} + 1\right)}{2\alpha} + \frac{\log\left(\frac{2\alpha P}{n_0^2} + 1\right)^2}{8\alpha}. \tag{12}$$

A.2 Maximising Z as a Functional to Solve Continuous Counting

A more effective way to maximise Z over all functions $n(\lambda)$ under the constraint of a fixed number of parameters P relies on the use of calculus of variations. More specifically, from equation 6, we have

$$Z + \delta Z = \int_0^P \frac{\log\left((n+\delta n)^2 \times \frac{d(n+\delta n)}{dp}\right)}{(n+\delta n)^2} dp, \tag{13}$$

where a maximum point of the function Z implies $\delta Z = 0$ for any (small) δn. By expanding $Z + \delta Z$ in the first order in δn, discarding higher order terms, and relying on integration by parts, we obtain:

$$\begin{aligned} Z + \delta Z &\simeq \int_0^P \frac{\log\left((n+\delta n)^2 \frac{d(n+\delta n)}{dp}\right)(1 - 2\delta n / n)}{n^2} dp \\ &\simeq \int_0^P \left(\frac{\log(n^2 \frac{dn}{dp})}{n^2} - \frac{2\delta n \log(n^2 \frac{dn}{dp})}{n^3} + \frac{2\delta n}{n^3} + \frac{\frac{d\delta n}{dp}}{n^2 \frac{dn}{dp}} \right) dp \\ &\simeq \int_0^P \left(\frac{\log(n^2 \frac{dn}{dp})}{n^2} + \left(-\frac{2\log(n^2 \frac{dn}{dp})}{n^3} + \frac{2}{n^3} - \frac{d\left(\frac{1}{n^2 \frac{dn}{dp}}\right)}{dp} \right) \delta n \right) dp. \end{aligned} \tag{14}$$

In the last step we integrate by parts and set the boundary conditions for n. To maximise the functional in equation 14, we set boundary conditions with $\delta n(P) = \delta n(0) = 0$. Note that the first term of the integral in equation 14 is just Z, as defined in equation 6. We search for an optimal n such that δZ does not grow (at first order), no matter the choice of δn. This is equivalent to have $\delta Z = 0$ for any $\delta n \ll n(p)$, which means that for any p the factor inside the integral is zero – that is:

$$-\frac{2\log(n^2 \frac{dn}{dp})}{n^3} + \frac{2}{n^3} - \frac{d\frac{1}{n^2 \frac{dn}{dp}}}{dp} = 0. \tag{15}$$

In other words, the optimal $n(p)$ has to respect this differential equation. By defining $v = \frac{dn}{dp}$ we can re-write equation 15 as: $-\frac{2\log(n^2 v)}{n^3} + \frac{2}{n^3} + \frac{2}{n^3} + \frac{\frac{dv}{dp}}{n^2 v^2} = 0$. Since $\frac{dv}{dp} = \frac{dv}{dn}\frac{dn}{dp} = \frac{dv}{dn} v$, we can remove all the explicit occurrences of p and obtain:

$$\frac{dv}{dn} = \frac{2v}{n}\left(\log(n^2 v) - 2\right). \tag{16}$$

This differential equation is a first order ordinary differential equation and, given the initial condition for v at $n(0)$, can be integrated numerically to obtain a family of solutions for v as a function of n. Then, $n(p)$ and $n(\lambda)$ can be integrated numerically. Fig. 7 shows a family of solutions for $n_0 = 56.5$. The thick line is the solution with the largest $Z \simeq 272$, which is larger than 251, obtained in Sec. 2.2 with the exponential assumption (i.e., $n(\lambda) = n_0 e^{\alpha\lambda}$). For comparison, we show this exponential function from Sec. 2.2 as a thin black line. The system predicts a network depth $\Lambda \approx 174$ (the highest point in Fig. 7 right panel), which is comparable with the value of 167.5 found in Sec. 2.2.

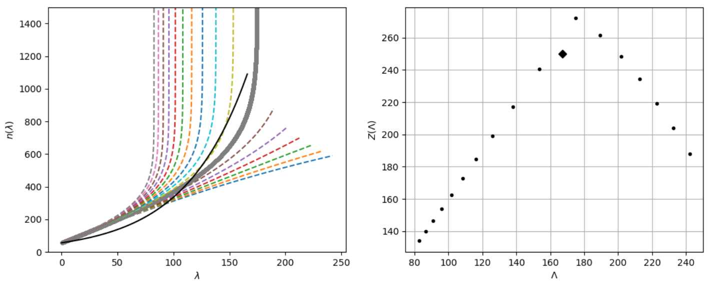


Figure 7: In the left panel, the dashed lines represent different solutions of the differential equation obtained by maximising Z locally, with the same $n(0)$ and different $v(0)$. The solution for optimal global Z is the thick grey line. The exponential approximation from Sec. 2.2 (i.e., $n(\lambda) = n_0 e^{\alpha\lambda}$) is the solid black line. In the right panel, the dots represent the depths Λ and the corresponding values of Z of the solutions of the left panel. The diamond represents the exponential approximation solution.