HiLD 2024: 2nd Workshop on High-dimensional Learning Dynamics

Exploring the development of complexity over depth and time in deep neural networks

Hannah Pinson H.PINSON@TUE.NL
Data and AI cluster & EAISI, Eindhoven University of Technology, The Netherlands
Data Analytics Laboratory, Vrije Universiteit Brussel, Belgium

Aurélien Boland
Data and AI Cluster & EAISI, Eindhoven University of Technology, The Netherlands

Vincent Ginis
Data Analytics Laboratory & Applied Physics Group, Vrije Universiteit Brussel, Belgium
School of Engineering and Applied Sciences, Harvard University, USA

Mykola Pechenizkiy
Data and AI Cluster & EAISI, Eindhoven University of Technology, The Netherlands

Abstract

Neural networks obtain their expressivity from nonlinear activation functions. While it is often assumed that the implemented transformations are effectively nonlinear at every layer, recent studies indicate that the overall function implemented by the network is close to linear at the start of training, only becoming more complex as training progresses. It is unclear how the evolution of the overall function during training can be related to changes in the effective (non)linearity of the individual network layers. In this study, we investigate these changes in effective (non)linearity over time *and* depth, i.e., over updates during training and per layer. We present a straightforward way to asses the effective linearity of layers through the use of partly linear models; in the case of an 18-layer nonlinear convolutional neural network (CNN) trained on the Imagenet dataset, we find that a large part of the layers start out in the effective linear regime, and that layers become effectively nonlinear in the direction from deep to shallow layers. The evolution over depth and time thus follows a distinct, wave-like pattern. We also propose an alternative method to reveal this evolution in a computationally efficient way, and we extend our experimental results to the Resnet50 architecture. The simple techniques we propose could help gain valuable insights in the relationship between depth, training time, and the complexity of the function a neural network implements.

1. Introduction

Recent studies suggest that the *effective* complexity of a given, fixed network changes during training: it has been shown that the functions the networks implement start out close to linear and gradually become more complex, at least in simple settings and when trained with gradient descent [7, 8, 11]. However, it is unclear how this development takes place over the depth of the network. In this workshop paper, we present our first experimental results showing how the complexity of the function implemented by a deep neural network develops over depth and time:

- using an 18 layer CNN with non-linear activations and trained on Imagenet, and associated partly linear models, we show that most of the layers start in a linear-like regime and move out of this regime during training,

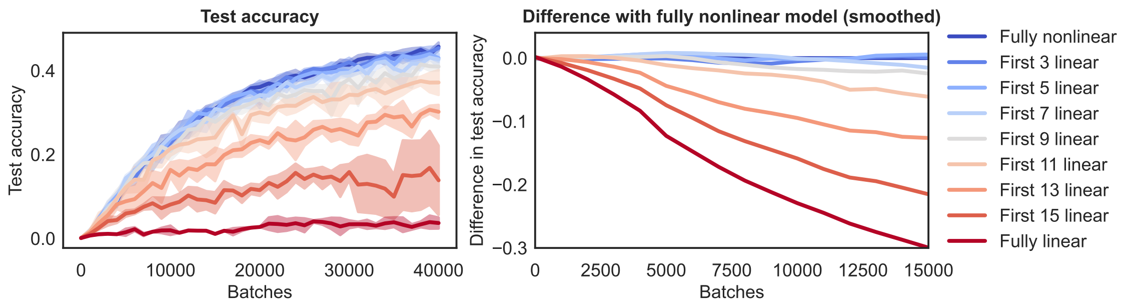

Figure 1: Evolution of the test accuracy of the nonlinear BaseNet18 model and its partly linear counterparts (left), and difference of all models with the nonlinear model (right). Left: solid curves are averages over runs, shaded areas indicate standard deviations. Right: the averaged curves are additionally smoothed to visualize the evolution more clearly (see appendix A).

- we show that this evolution takes place at different speeds at different layers, resulting in a complexity ‘wave’ over depth and time,
- we introduce a simple metric that can be used to study this evolution over depth and time using limited computational resources,
- and we also extend our results to a 50 layer CNN with residual connections, ResNet50 [6], to show our proposed tools could be used for studying the evolution in more complicated architectures.

Our work is related to studies on learning dynamics and simplicity bias, i.e., the tendency of neural networks trained with (stochastic) gradient descent (GD) to implement, out of all possible functions, ‘simpler’ ones that also generalize well [12, 13]. This simplicity bias is often linked to experiments showing that the functions implemented by nonlinear networks start out close to linear and gradually become more complex over time [7, 8, 11, 17]. It has also been theoretically explained how different types of linear networks learn to separate classes from broader (simpler) to finer (more complex) distinctions [10, 14, 16] and this phenomenon can still be observed for nonlinear networks [4, 14]. Moreover, neural networks trained with SGD learn dataset distributions of increasing complexity [15] and dataset distillation has to yield samples of increasing complexity to match different phases of learning [2]. Our results are most closely related to the work of Kalimeris et al. [11], where the authors use an information-theoretic metric to show that in the initial epochs of training, almost all of the performance improvement of a nonlinear network can be explained by a linear classifier. They furthermore provide some evidence that the complexity of the network further increases as training progresses, through the training of different models of different depth on simple datasets. In this work, we use a far more complex dataset (imagenet vs simple binary classification from MNIST and CIFAR-10 datasets), but the main difference is that we show how this increasing complexity can be studied *within* the same network.

2. Methods and Experiments

Partly linear models For each fully nonlinear model, we introduce a set of partly linear models: these are models with the same architecture, trained under the same conditions, but with all activation functions at specific layers set to linear. If the accuracy and predictions of the original model and the partly linear model are indistinguishable up to stochastic effects, we say the selected layers in the original model are effectively operating in a linear-like regime. To estimate when the behavior of a partial linear model diverges from the main model, we compare their accuracy curves (averaged over different runs): after an additional smoothing step to reduce stochastic effects (details, see appendix A), we compute the timepoint where the accuracy of the partly linear model drops below 95% of that of the main model. Given that models with similar performances could potentially have different predictions (even when taking stochastic effects from different initializations into account), a comparison based on performance might not be enough. We therefore perform a more fine-grained analysis using an adaptation of the information-theoretic metric proposed in [11], and show that our conclusions still hold, in appendix B.

Mean of preactivations $\overline{\mathbf{p}}^{\mathbf{l}}$ We denote the distribution of inputs z to a nonlinear activation function $f(z)$ as the *preactivations*. For each activation function/node, we compute the mean of the preactivations, and then we compute another mean of these values per layer l: $\overline{p}^l = \frac{1}{M}\sum_i^M(\frac{1}{N}\sum_s^N z_{s,i}^l)$, with M the number of nodes in layer l and N the number of samples, and $z_{s,i}^l$ the preactivation value for sample s at node i at layer l. We compute this value through randomly selecting 500 samples of the input data instead of the whole dataset, which significantly reduces the computational cost. We argue that we can use this simple metric to indicate the effective (non-)linearity of a layer: when the value is positive or close to zero, (most of) the implemented ReLU transformations at that layer are not very different from applying linear transformations. This is further explored and confirmed through our different experiments; however, this is a very simple metric, and we do believe more suitable and/or more precise metrics exist. Apart from this, there are some important remarks to be made concerning the network architecture: batch normalization (BN) [9] is added right after each convolution and before the activation function, therefore the preactivations are the outcome of a batch normalization operation; but note that there are in fact two trainable parameters in a BN layer that allow for the resulting distribution to have a mean different from zero. In the case of residual connections, the preactivations at certain layers are the result of summing the output of two previous layers, see He et al. [6].

Experiments The first architecture we consider is in fact the ResNet18 version 1 architecture [6] but without residual connections -as residual connections complicate the notion of 'depth', we start our exploration with a model where they are removed. We call his model BaseNet18. We furthermore used 7 corresponding partial linear models with respectively the first 3, 5, 7, 9, 11, 13, and 15 convolutional ReLU [1] layers set to linear, and also used fully linear variants (however, the max-pooling and softmax operations remain). All these models were trained on the Imagenet [3] ILSVRC 2012 dataset. We use the same training settings as [6], but without the additional regularization. Each type of model was trained 5 times, and reported results are the average values. A detailed overview of the architecture and training details can be found in the appendix A. The second set of experiments is conducted on the ResNet50 version 1 architecture, which has more layers, has a more intricate structure in the blocks of convolutional layers, and which features residual connections [6]. We here consider 4 different kinds of partly linear models: the first 20 conv. layers linear, the first 40 conv. layers linear, the last 20 conv. layers linear, and a selected subset of conv. layers linear.

This subset consists of layers of the original model with a positive mean of preactivations at the end of training, which in this case were the layers with indices 0, 2, 3, 6, 8, 9, 10, 11, 12, 15, 18, 20, 21, 22, 24, 27, 30, 33, 36 and 39. We added this last experiment to further illustrate the relationship between a positive or close to zero mean of preactivations and the effective linearity of layers.

3. Results and Discussion

In Fig. 1, we compare the performance of the fully nonlinear BaseNet18 model with the performance of its different partial linear counterparts. Fig.1(left) shows the evolution of the test accuracy up to 40000 batches, in fig.1(right) we show the difference in accuracy of the partial linear models w.r.t. the fully nonlinear models, using smoothed curves to enhance the visualization, and showing the evolution up to 15000 batches. We find that most partial linear models exhibit a behavior similar to the main model in the beginning of training; however, the more layers of a model are set to linear, the earlier its performance starts to diverge from the main model. Using the 95% threshold, we found the divergence points to be 0 (fully linear), 0 (15 linear), 3000 (13 layer), 9000 (11 linear), 10000 (9 linear) and 13000 (7 linear) batches (the resolution is 1000 batches), but we found no divergence point for the models with the first 5 and 3 layers linear. In short, the main model starts out with at least the first 13 layers in the linear-like regime, and gradually more and more layers move to the effective nonlinear regime. This evolution happens from deeper to more shallow layers. However, the first 5 layers seem to remain in the linear-like regime throughout training: there is no effective difference in loss (and in the information-theoretic metric based on the predictions, see Appendix B) between the main model and the partial linear models with the 3 and 5 first layers kept linear.

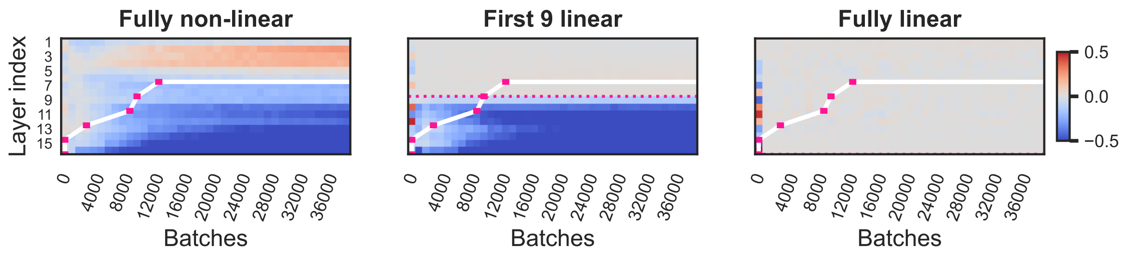

Figure 2: Evolution of the mean of preactivations over depth and time for the fully nonlinear model, the model with first 9 layers linear, and the fully linear model (average of training runs).

In fig. 2, we show the evolution over depth and time of the mean of preactivations of the BaseNet18 model, its partial linear counterpart with the first 9 layers set to linear, and the fully linear counterpart (for a visualization for all partial linear models, see Appendix C). In the plot for the fully nonlinear model (left), we note that in the beginning of training most layers exhibit patterns values close to zero. From looking at the model with the first 9 layers linear (fig. 2, middle) and the fully linear model (fig. 2, right), we can note that truly linear layers can be recognized from having a mean preactivation value that is very close to zero. For the fully nonlinear model (left), there is an evolution over time in the values from the deepest to the shallowest layers: the deepest layers exhibit negative values first, and over time the values change from negative in the deep layers to positive in the shallow layers. On top of the plots in this fig.2 we have added the divergence points

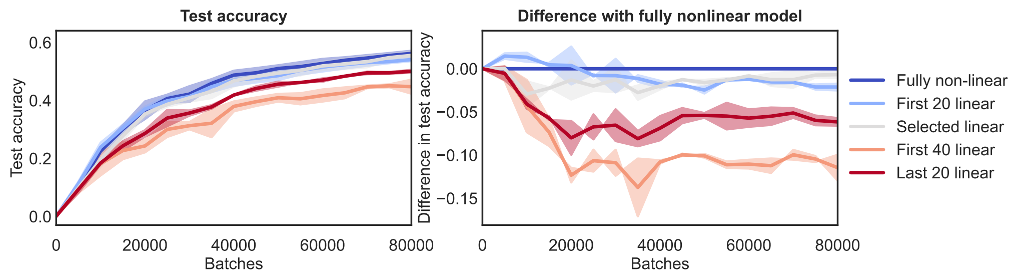

Figure 3: Evolution of the test accuracy of the nonlinear ResNet50 model and its partly linear counterparts (left), and difference of all models with the nonlinear model (right). Solid curves are averages over runs, shaded areas indicate standard deviations.

obtained from the analysis of the partly linear models as pink squares: a pink square at layer index l and timepoint t means a partial linear model with the first l layers linear will diverge from the main model at timepoint t. We connected the divergence points with a white line, indicating the "wavefront" of developing complexity: at any timepoint in any of the plots, the line indicates how far the complexity has developed in the fully *nonlinear* model, i.e., how many of its (deeper) layers are in the effective non-linear regime. We extended this white line horizontally beyond the last divergence point to indicate that there is no further progression of effective non-linearity in the fully non-linear model beyond the last divergence (time)point. From comparing this wavefront line with evolution of the mean of preactivations, we can conclude that the mean of preactivations gives a good indication of the evolution of complexity over depth and time in the network. Nonlinear layers in the effective linear regime can be recognized from a positive value or a value close to zero.

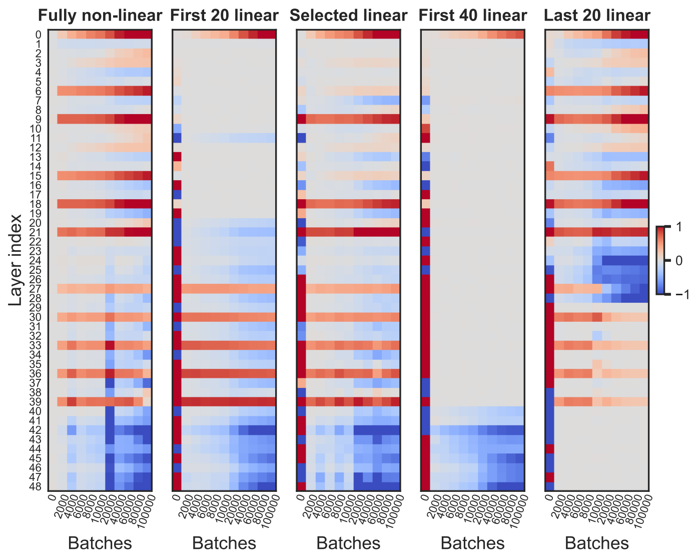

Figure 4: Mean of preactivations for the ResNet50 model and corresponding partly linear models.

In fig. 3 we show the test accuracy for the different Resnet50 models (left) and the difference compared to the fully nonlinear model (right). From looking at fig. 3 we can see that neither setting the first 20 layers to linear, nor setting the selected subset to linear, changes the performance considerably. Making the first 40 layers or last 20 layers linear, however, results in a more clear decrease in the accuracy. In fig. 4 we show the corresponding patterns of the means of preactivations over depth and time. The layers with larger positive values (i.e., the red lines) are at the end of a residual connection; their preactivations are the sum of two terms, of which one is the result of a ReLU function [6], making the result mostly positive. From the different plots, we can still see that deeper layers move towards the effective non-linear regime faster than shallower layers. We can further compare the results shown in fig. 3 to these different patterns of the means of preactivations. The first 20 linear and selected linear models retain a relatively large number of layers up to index 40 that develop negative values and are thus, based on our previous observations, in an effective non-linear regime; the profile of these models in the last 10 layers is also similar to that of the nonlinear model, and the performance of these models is comparable. The model with the first 40 layers linear cannot develop complexity in those first 40 layers, and lacks expressivity. This model also exhibits the lowest performance overall. Interestingly, for the model with the last 20 layers linear, the negative values for the means of preactivations develop instead over the first 30 layers. This model exhibits an intermediate performance.

Discussion Together, our results show that the development of effective nonlinearity over depth and time happens in a distinct, wave-like pattern. We find that deeper layers move towards the effective non-linear regime faster than shallower layers, and this pattern can still be noted in complex architectures with residual connections. This provides further evidence for the claim that neural networks trained with gradient descent gradually become more complex during training. Unlike previous studies, we find that the network is not necessarily completely 'linear-like' at the start of training, as some layers start already outside this regime. Increasing the resolution in time (i.e., including more timepoints before the first 1000 batches) might better reveal the evolution in the very first steps of training. The techniques we propose to study this phenomenon are easy to implement, and especially obtaining the means before preactivations profile is computationally inexpensive. However, more precise metrics probably exist, and we leave this to future work. Finally, we find that the specific evolution depends on the network architecture, and we hypothesize that the complexity of the dataset could influence this pattern as well. In future work, we will examine whether different architectures or datasets still yield the same evolution of complexity over depth and time.

Acknowledgements V.G. acknowledges support from Research Foundation Flanders under Grant No.G032822N and G0K9322N.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL http://arxiv.org/abs/1803.08375.

[2] Xuxi Chen, Yu Yang, Zhangyang Wang, and Baharan Mirzasoleiman. Data Distillation Can Be Like Vodka: Distilling More Times For Better Quality, October 2023. URL http://arxiv.org/abs/2310.06982. arXiv:2310.06982 [cs].

[3] 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 Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

[4] Guy Hacohen and Daphna Weinshall. Principal components bias in over-parameterized linear models, and its manifestation in deep neural networks. *J. Mach. Learn. Res.*, 23:155:1–155:46, 2022. URL http://jmlr.org/papers/v23/21-0991.html.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

[6] 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*, pages 770–778, 2016.

[7] Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *Advances in Neural Information Processing Systems*, 33:17116–17128, 2020.

[8] Xia Hu, Weiqing Liu, Jiang Bian, and Jian Pei. Measuring model complexity of neural networks with curve activation functions. In *Proceedings of the 26th ACM SIGKDD International Conference on knowledge discovery & data mining*, pages 1521–1531, 2020.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

[10] Clémentine C J Dominé, Lukas Braun, James E Fitzgerald, and Andrew M Saxe. Exact learning dynamics of deep linear networks with prior knowledge *. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(11):114004, November 2023. ISSN 1742-5468. doi: 10.1088/1742-5468/ad01b8. URL https://iopscience.iop.org/article/10.1088/1742-5468/ad01b8.

[11] Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

[12] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6614.

[13] Guillermo Valle Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *7th International Conference on Learning Representations, ICLR*, 2019.

[14] Hannah Pinson, Joeri Lenaerts, and Vincent Ginis. Linear cnns discover the statistical structure of the dataset using only the most dominant frequencies. In *International Conference on Machine Learning, ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 27876–27906. PMLR, 2023. URL https://proceedings.mlr.press/v202/pinson23a.html.

[15] Maria Refinetti, Alessandro Ingrosso, and Sebastian Goldt. Neural networks trained with sgd learn distributions of increasing complexity. In *International Conference on Machine Learning*, pages 28843–28863. PMLR, 2023.

[16] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, June 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1820226116. URL https://pnas.org/doi/full/10.1073/pnas.1820226116.

[17] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6cfe0e6127fa25df2a0ef2ae1067d915-Abstract.html.

Appendix A. Architectures and experimental details

In fig. 5, we schematically show the BaseNet18 architecture and two of its example partly linear counterparts. Note that each convolutional 'layer' (indicated in blue) in itself consists of convolutional operations, subsequent batch normalization, and ReLU activation functions. In the figure, we indicate the filter size (always 3x3) and the number of filters at each layer. When a layer has the indication '/2', this means we use a stride of two to decrease the dimensions of the representations.

All networks are initialized with He uniform initialization [5]. We use SGD with momentum 0.9 and starting learning rate = 0.1; this learning rate is decreased by a factor 10 when a validation loss plateau occurs. The dataset we use is the Imagenet [3] ILSVRC2012 dataset, which has 1000 classes, 1.281.167 training images and 50.000 validation images (which we use as the 'test' set). The batch size is 256. Images are augmented and cropped to dimension 224x224 before training, following the procedure outlined in [6]. No additional regularization is added.

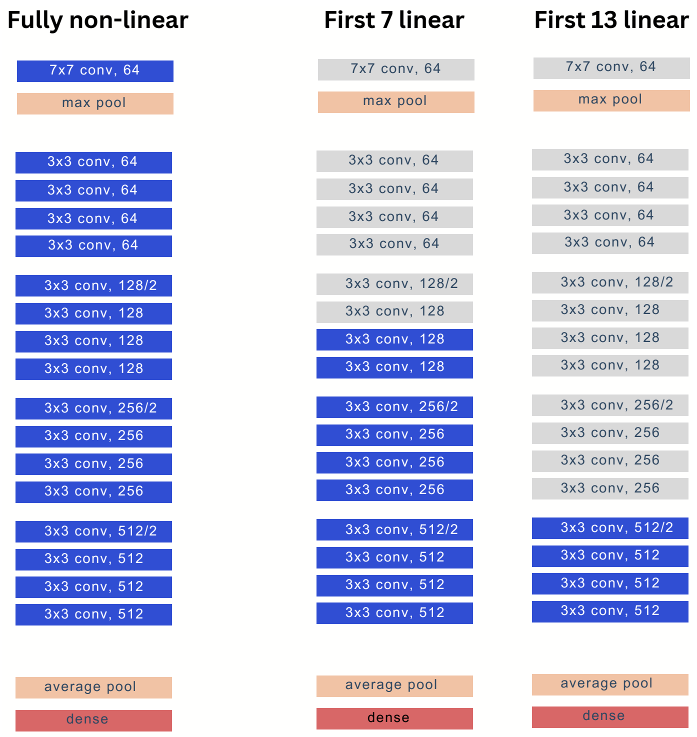

Figure 5: Schematic overview of BaseNet 18 architecture and example partly linear counterparts. Blue indicates the presence of a ReLU activation function; when the layer is colored gray the activations functions are removed.

To determine the divergence points of the different models w.r.t to the fully linear models, we first smooth the curves with a Savitzky–Golay filter of size 5 and polynomial order 2, and then compute where the relative performance drops below 95%.

Appendix B. Information-theoretic metric

Two different models could have the same loss at a given timepoint during training, but still their predictions could be very different in nature. To compare the predictions of models on a more fine-grained level, we adapt the information-theoretic metric originally proposed in [11].
Consider the ground truth labels $\boldsymbol{Y}$, the predictions of a fully non-linear model $\boldsymbol{M}^t$ and the predictions of a corresponding partly linear model $\boldsymbol{P}^t$ at the same timepoint t during their training. We can calculate how much of the performance of the main model can be explained from the predictions of the partly linear model:

$$\mu(\boldsymbol{M}^t; \boldsymbol{P}^t) = I(\boldsymbol{M}^t; \boldsymbol{Y}) - I(\boldsymbol{M}^t; \boldsymbol{Y}|\boldsymbol{P}^t) \tag{1}$$

with $I(\boldsymbol{X}_1, \boldsymbol{X}_2)$ the mutual information between two random variables $\boldsymbol{X}_1, \boldsymbol{X}_2$. Here the mutual information $I(\boldsymbol{M}^t; \boldsymbol{Y})$ monotonically increases with increasing performance (decreasing loss). When $I(\boldsymbol{M}^t; \boldsymbol{Y}|\boldsymbol{P}^t) = 0$, this means knowing the predictions of the main model $\boldsymbol{M}^t$ does not help in predicting $\boldsymbol{Y}$ if we already know the predictions of the partly linear model $\boldsymbol{P}^t$. Therefore, if $\mu(\boldsymbol{M}^t; \boldsymbol{P}^t) = I(\boldsymbol{M}^t; \boldsymbol{Y})$, the performance of the main model can be fully explained from the partly linear model; but when $\mu(\boldsymbol{M}^t; \boldsymbol{P}^t) < I(\boldsymbol{M}^t; \boldsymbol{Y})$, there is information in the predictions $\boldsymbol{M}^t$ about $\boldsymbol{Y}$ not present in the predictions $\boldsymbol{P}^t$. In the original work by Kalimeris et al. [11], the used predictions for the partial models $\boldsymbol{P}$ are not those at the same timepoint in training t, but rather the predictions when convergence is reached. We here choose to compare with partial models at the same timepoint during their training, such that we can really compare the evolution of the different models over time.

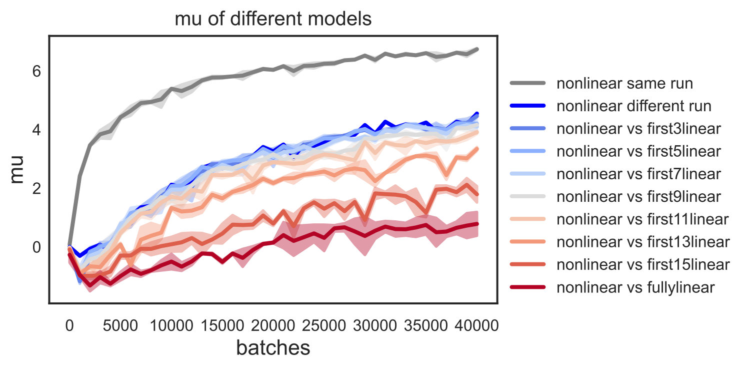

Figure 6: Evolution of the metric $\mu(\boldsymbol{M}_i^t; \boldsymbol{M}_j^t)$ during training for the different combinations i, j of nonlinear and (partly) linear models.

To compare each partly linear model with the main, fully nonlinear model up to stochastic effects (arising from different initial conditions), we consider the following variations of the metric μ:

$$\mu(\boldsymbol{M}_i^t; \boldsymbol{M}_i^t) = I(\boldsymbol{M}_i^t; \boldsymbol{Y}) - I(\boldsymbol{M}_i^t; \boldsymbol{Y}|\boldsymbol{M}_i^t) = I(\boldsymbol{M}_i^t; \boldsymbol{Y}) \tag{2}$$

where the repeated symbol i denotes the fact that we compare the main model with itself, i.e., from the same run. We also consider the case where we compare one run of the main model with a different run of the same main model, denoted by different indices i, j:

$$\mu(\boldsymbol{M}_i^t; \boldsymbol{M}_j^t) = I(\boldsymbol{M}_i^t; \boldsymbol{Y}) - I(\boldsymbol{M}_i^t; \boldsymbol{Y}|\boldsymbol{M}_j^t) \tag{3}$$

$\mu(\boldsymbol{M}_i^t; \boldsymbol{M}_j^t)$ thus captures how much of the performance of the model in run i could be equivalently explained by the same model in a different run j, both taken at timepoint t. Whenever $\mu(\boldsymbol{M}^t; \boldsymbol{P}^t) = \mu(\boldsymbol{M}_i^t; \boldsymbol{M}_j^t)$, we can interpret this as the partially linear model being equally predictive w.r.t to the main model as a different run of that main model. In fig. 6, we show the evolution of the metric $\mu(\boldsymbol{M}_i^t; \boldsymbol{M}_j^t)$ during training for the different combinations i, j of nonlinear and (partly) linear models. We can see that the partially linear models with the 3 and 5 first layers set to linear are equally predictive of the nonlinear model as a different run of that same nonlinear model. We can also see the same ordering of the models as we found before, both in terms of predictivity as in moment of divergence from the main model. However, the divergence points are harder to determine exactly with this metric, which is why we resorted to the use of the accuracy in the main paper. This is partly due to the fact that this metric can take on negative values, which is a reflection of the fact that the predictions from the different models are still very different in the beginning of training [11] (i.e., when still transitioning from random to more structured weights).

Appendix C. Mean of preactivations for all Basenet models

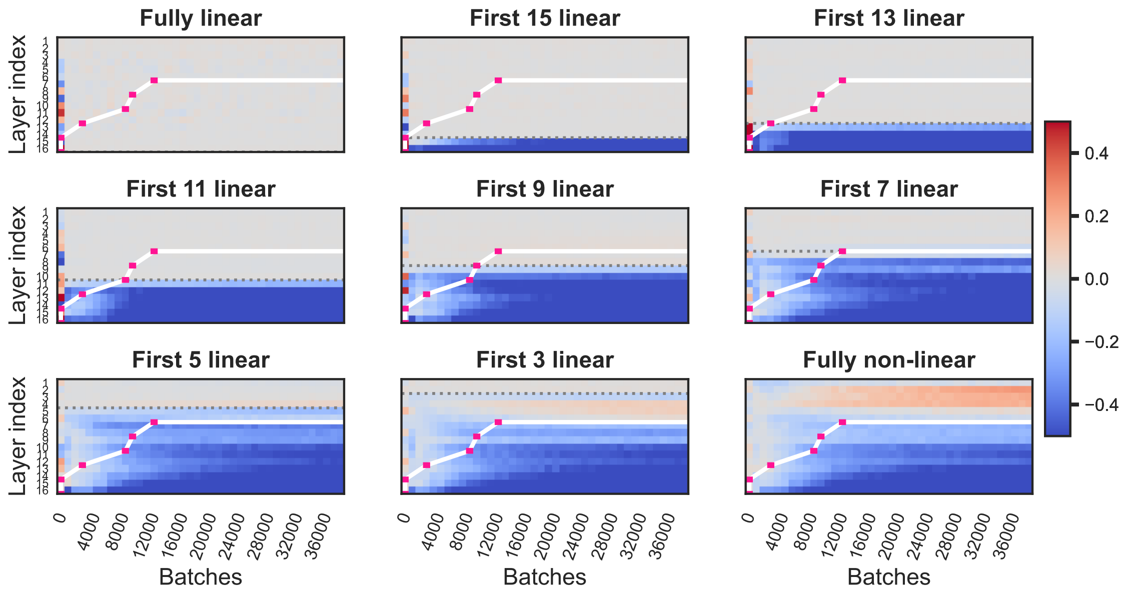

Figure 7: Evolution of the mean of preactivations over depth and time for the Basenet18 model and all partial linear models. White line and pink squares indicate the divergence wavefront, as described in the main text.