Increasing the complexity of "SGD on neural networks learns functions of increasing complexity"

Coen Schouten
c.j.schouten@student.tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

Kelvin Toonen
k.a.b.toonen@student.tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

ABSTRACT

Stochastic gradient descent (SGD) is a well known method to train neural networks, however, much of its workings are unknown. In the paper by Nakkiran et al. [2] it is investigated whether SGD learns functions of increasing complexity. This paper aims to reproduce and extend their work to see if this is the case for multi-class scenarios and for different model architectures, namely a vision transformer model. We show that these results hold not only for the binary scenario, but also for the multi-class case. The transformer model seems to learn the different classes in the same order as the non-linear convolutional neural network from the paper, although it does so much sooner.

KEYWORDS

SGD, CNN, Transformer, Mutual Information

ACM Reference Format:
Coen Schouten and Kelvin Toonen. 2023. Increasing the complexity of "SGD on neural networks learns functions of increasing complexity". In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

This paper discusses a reproduction study and two extensions of experiments about learning dynamics of SGD on neural networks as described in the paper "SGD on neural networks learns functions of increasing complexity" by Nakkiran et al. [2]. In that paper the authors state the hypothesis that the initial increase in performance of a non-linear convolutional neural network can be fully attributed to a simple fully connected linear network. From now on we will refer to the non-linear convolutional neural network as just CNN. In this paper we reproduce two of the experiments presented in section 3 of the paper of Nakkiran et al. Furthermore, we present an extension of the experiments from binary classification scenarios to multi-class classification scenarios to investigate whether the same behaviour still occurs. Finally, we replace the CNN by a simple transformer model and we compare the CNN and the transformer model to investigate whether or not the SGD training shows similar results in different settings.

2 THE PAPER

The paper by Nakkiran et al. provides the hypothesis that SGD learns functions of increasing complexity. They support this hypothesis by providing experiments that show that in the initial epochs of learning SGD seems to have a bias towards simple classifiers, and that in later epochs SGD is relatively stable and retains the information from the simple classifier it obtained in the initial epochs. [2] To measure the performance correlation between different classifiers the authors define the notion of performance correlation μ_Y on data Y between two classifiers F and L as in Equation 1.

$$\mu_Y(F;L) := I(F;Y) - I(F;Y|L) = I(L;Y) - I(L;Y|F) \tag{1}$$

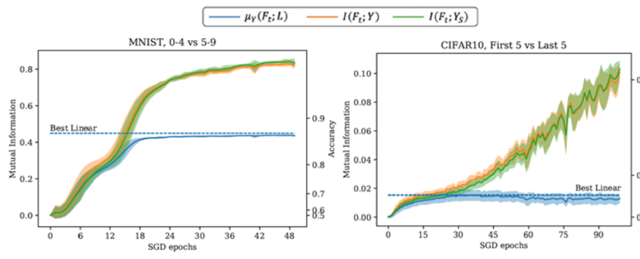
The authors propose to use this metric over the mutual information between the two classifiers because this metric captures the degree to which the information learned by F about Y is explained by L, whereas the mutual information between the two classifiers only captures their correlation, regardless of the data Y. In this paper we are only interested in reproducing some of the results of section 3 of the original paper, which are shown in Figure 1a. The original paper also contains experiments going beyond the linear setting, with CNNs of increasing complexity, but we will not attempt to reproduce these.

3 EXPERIMENTAL SETUP

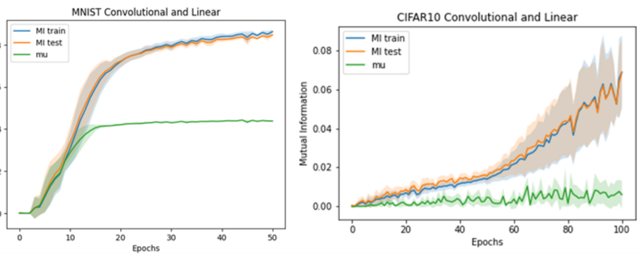
For all of the experiments in this paper two different datasets were used, namely MNIST and CIFAR10.

3.1 Reproduction

For the experiments regarding reproducibility of the original paper we chose to only reproduce two of the experiments that were presented in section 3, as when these are reproducible, we can expect that all the others are as well. We chose to reproduce the experiments of the binary MNIST (0-4 vs 5-9) and the CIFAR10 first 5 vs last 5 datasets, as these showed the most interesting behaviour. The original paper does not clearly describe how the linear model is obtained, so for the reproduction we used the simplest fully connected linear model possible, connecting input to output, which we train simultaneously with the CNN. For the non-linear CNN this reproduction uses the same architecture as the original paper, furthermore all of the learning parameters are the same. We repeated our experiments 5 times, the number of repetitions of the orignal paper is 10, we were only able to do 5 because of limited availability of compute.



(a) The results of the original paper



(b) The reproduction results

Figure 1: Experiments in a binary classification setting

3.2 Multi-class

The work of Nakkiran et al. only considers cases of binary classification, for which they give 2 reasons: "(1) there is a natural choice for the simplest model class (i.e, linear models) and (2) our mutual-information based metrics can be more accurately estimated from samples" [2]. The authors also state that they have preliminary work extending the results to the multi-class setting, but do not provide these results, which prompted us to investigate how we could extend these experiments to multi-class settings. The mutual information estimation approach described in the original paper makes use of the empirical distributions $\hat{p}$ over the test set, as shown in Equation 2, where C denotes the set of classes, f and g are classifiers and y are the true labels.

$$I(F;Y|G) = \sum_{(f,y,g)\in C^3} \hat{p}(f,y,g) \log\left(\frac{\hat{p}(f,y|g)}{\hat{p}(f|g)\hat{p}(y|g)}\right) \tag{2}$$

It is straightforward to condition this mutual information on the true classes of the data, which allows us to investigate the behaviour for each true class separately. We show how this can be done in Equations 3 and 4.

$$I(F;Y|G) = \sum_{y\in C} \sum_{(f,g)\in C^2} \hat{p}(f,y,g) \log\left(\frac{\hat{p}(f,y|g)}{\hat{p}(f|g)\hat{p}(y|g)}\right) \tag{3}$$

$$I(F;Y=y|G) = \sum_{(f,g)\in C^2} \hat{p}(f,y,g) \log\left(\frac{\hat{p}(f,y|g)}{\hat{p}(f|g)\hat{p}(y|g)}\right) \tag{4}$$

This result allows for using the same approach as in the earlier experiments, using the same network architectures, on the regular multi-class MNIST and CIFAR10 datasets.

3.3 Transformer

In the paper of Nakkiran et al. there was only a comparison between a CNN and a linear model. We wondered if similar results would hold for transformers, as they are much more different than these models. For this paper the vision transformer architecture from the `torchvision` [1] library is used. The parameters were adjusted such that the parameters of the vision transformer were in the same order as the parameters of the CNN. The transformer was first compared to the linear model in the same way as had been done for the CNN in the multi-class setting, then it was compared in a similar way to the CNN.

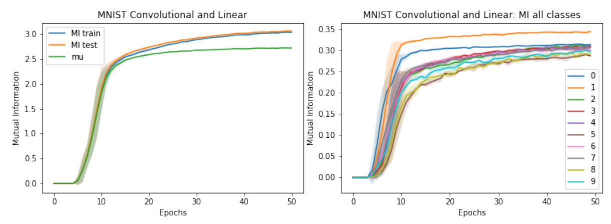


Figure 2: Overview of all classes and MI per class for MNIST data and between CNN and linear model

4 RESULTS

4.1 Reproduction

The results for the reproduced experiments of the original paper are shown in Figure 1a, the results of our reproduction are shown in Figure 1b. When comparing the figures we can make some interesting observations. First of all, we can see that the general evolution seems to be the same for the original results and the reproduction on the same data. Furthermore we can see that our standard deviation seems to be a bit larger, which is likely due to the fact that we perform less repetitions. Furthermore we can observe that our linear model can explain less of the performance on CIFAR10, which might simply be due to the fact that our linear model is worse, as it is very simple and the original paper does not describe how they obtain their model.

4.2 Multi-class

The results of the multi-class experiments can be seen in Figures 2 and 3. In Figure 2 it can be seen that the CNN has a very clear warm-up phase, where it does not seem to learn much. However, after this phase the performance of the CNN increases for many of the classes. We can see, however, that there seems to be an order in which classes are learned. In particular classes 0 and 1 seem to be learned first and classes 5 and 8 last. In Figure 3 this seems to correspond to how well μ_Y approaches the MI, as the distance between them is small for classes 0 and 1 and gets bigger until it has reached class 5, the last class learned.

In Figures 4 and 5 the multi-class results on CIFAR10 can be seen. In Figure 4 we can see that in the first plot there is a steady increase, but in the second plot we can see that there are several large increases for a few classes at a time. Frog, ship and plane are the first to have such a jump in performance and when we look at Figure 5 we can see that μ_Y also increases much more quickly than for the bird

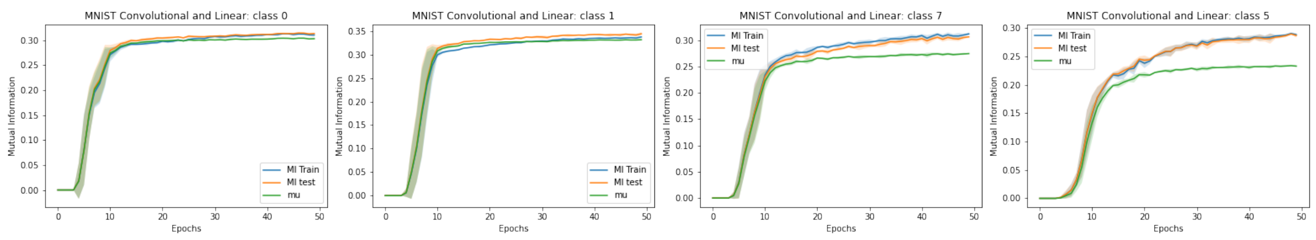


Figure 3: MI and μ_Y of different classes for MNIST data and between CNN and linear model

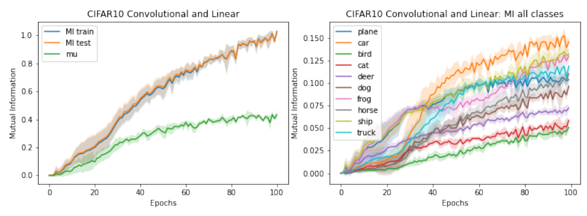


Figure 4: Overview of all classes and MI per class for CIFAR10 data and between CNN and linear model

class, where the MI also increases much more slowly at the start. In Figure 4 in the second plot it can be clearly seen that around epoch 27 the difference between a car and a truck is learned, after which the performance on each increases heavily.

4.3 Transformer

In Figure 6 the results of the experiments between the transformer and linear model can be seen on both datasets. In Figure 6a we can see that unlike the CNN the transformer does not have a warm-up phase and has a great increase in performance at each step in the beginning. The MI of class 1 in Figure 6b seems to be part of the reason for this performance, as it already starts at a high MI before training. However, it seems that the linear model also has a large starting performance on class 1, as the performance correlation at the start is very close to the MI of the transformer in Figure 6a.
In Figure 6d it is clear that the difference between MI an μ_Y is much larger much sooner than for the CNN in Figure 4. Also, the evolution of the performance correlation is also different in these figures, implying that there is something learned by the linear model that is not learned by the other models. From Figure 6d we can see that the order in which the model has a large increase in performance for certain classes is very similar to the CNN in Figure 5. However, these jumps in performance are much sooner than in the case of the CNN.
The experiments between the transformer and the CNN on MNIST can be found in Figure 7. In Figure 7a it can be seen that the evolution of the MI and of the μ_Y are very similar, where the μ_Y is delayed by a warm-up phase. It does look like the μ_Y approaches the MI, meaning that the CNN eventually can explain at least as much as the transformer model, albeit much later. This is especially interesting to see in Figure 7b for class 1, as there is a big starting difference between the MI and the μ_Y, but this gap starts to close very quickly. What is interesting in Figure 7c is that this plot is very similar to the right plot in Figure 2. If we look at equation 1 this seems to be the case where the transformer model can explain everything that the CNN can explain, thus $I(L;Y|F) = 0$ where F is the transformer and L is the CNN.
The results of the experiments between the transformer and the CNN on CIFAR10 can be found in Figure 8. In Figure 8a we can see that there is a large difference between MI and μ_Y, although it seems that also for this dataset the μ_Y approaches the MI. In Figure 8b we can see that the μ_Y of many classes is still increasing. However, it seems that for car and plane the μ_Y has already stagnated, as can be seen in Figures 8c and 8d respectively. Therefore, it may not converge to the MI.

5 DISCUSSION AND CONCLUSIONS

We looked at many different experiments to see if and in what manner SGD on neural networks learns functions of increasing complexity. We found very similar results to that in the paper by Nakkiran et al. [2]. These results were extended beyond the binary case to the multi-class case, where very similar results were found. In all of the experiments it seems as if the order in which the classes are learned is related to their complexity. An interesting result was that is seems as if the linear model learned more at the start of learning than the CNN. The addition of transformers to the experiments showed that the transformer learned much faster than the CNN and do not have a warm-up phase. However, the CNN did seem to approach the level of the transformer, at least in the part of the data that the transformer performed well on.
Possible future works could take a look at reversing the way we looked at the models, meaning that for example the MI of the linear model is shown together with the performance correlation to the CNN. Another next step could be to combine these experiments with the experiments from Saxe et al. [4] and Pinson et al. [3]. These papers look at singular values to determine the order in which classes are learned. It would be interesting to see if that method gives a similar ordering as were found in our experiments. A final suggestion for extending the experiments from this paper is by looking at the effect that initialization has on the results. Especially because of the fact that the transformer was able to get good performance on class 1 from MNIST so quickly.

ACKNOWLEDGMENTS

We would like to thank Dr. H.E.A. Pinson for her supervision in this project. Our discussions gave rise to the very interesting extension experiments, which would not have happened without her input.

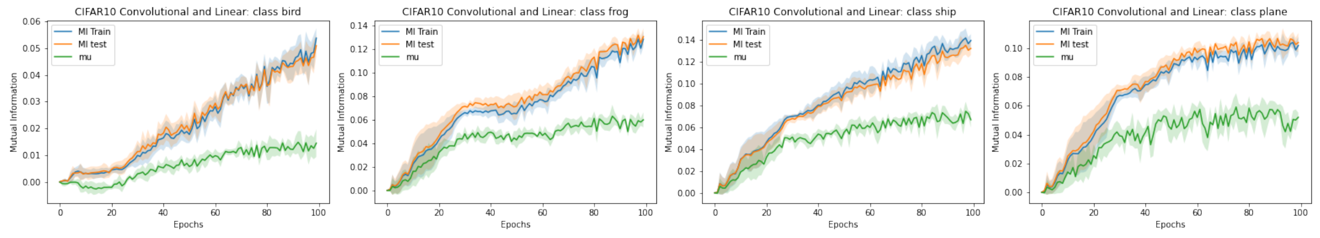
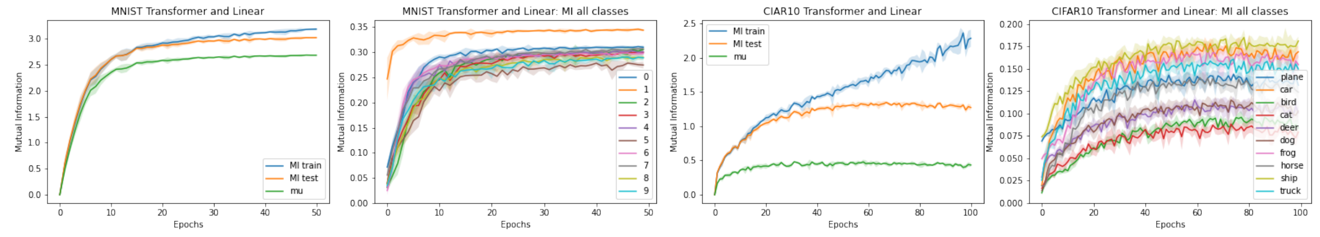
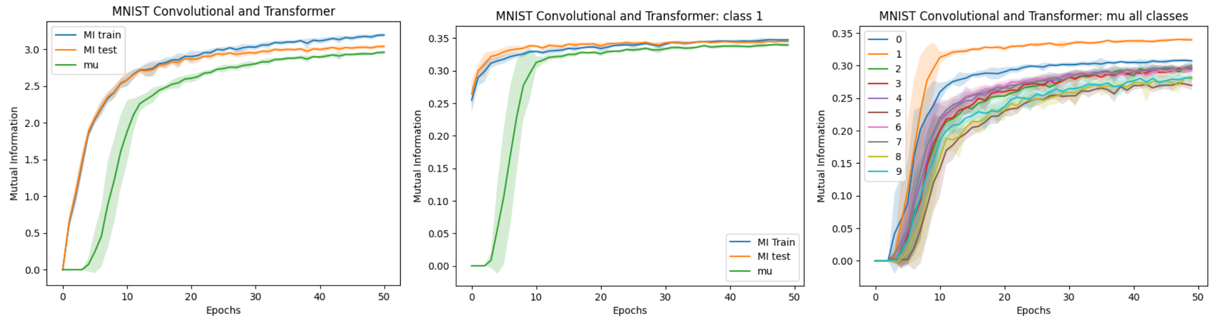


Figure 5: MI and μ_Y of different classes for CIFAR10 data and between CNN and linear model



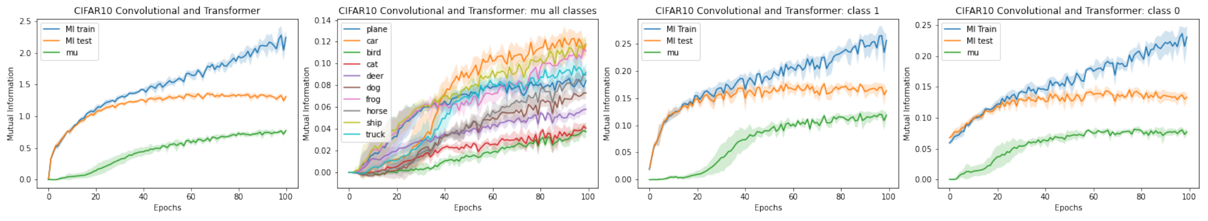
(a) Overview of all classes on MNIST **(b) The MI per class on MNIST** **(c) Overview of all classes on CIFAR10** **(d) The MI per class on CIFAR10**

Figure 6: Results of the analysis between the transformer and the linear model



(a) Overview of all classes **(b) MI and μ_Y of class 1** **(c) The μ_Y per class**

Figure 7: Results of the analysis between the transformer and the CNN on MNIST



(a) Overview of all classes **(b) The μ_Y per class** **(c) MI and μ_Y of class car** **(d) MI and μ_Y of class plane**

Figure 8: Results of the analysis between the transformer and the CNN on CIFAR10

REFERENCES

[1] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the Machine-Vision Package of Torch. In *Proceedings of the 18th ACM International Conference on Multimedia* (Firenze, Italy) *(MM '10)*. Association for Computing Machinery, New York, NY, USA, 1485–1488. https://doi.org/10.1145/1873951.1874254

[2] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. 2019. SGD on Neural Networks Learns Functions of Increasing Complexity. arXiv:1905.11604 [cs.LG]

[3] Hannah Pinson, Joeri Lenaerts, and Vincent Ginis. 2023. Linear CNNs Discover the Statistical Structure of the Dataset Using Only the Most Dominant Frequencies. *arXiv preprint arXiv:2303.02034* (2023).

[4] Andrew M Saxe, James L McClelland, and Surya Ganguli. 2019. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences* 116, 23 (2019), 11537–11546.

A AVAILABILITY OF CODE AND COMPLETE RESULTS

The source code of the project and all the plots of the experimental results are public available on github via the following url: https://github.com/CoenSchouten01/ATIAI

B TRANSFORMER SPECIFICS

For our transformer model we used the `VisionTransformer` class from the `torchvision` [1] library. We gave it the following parameters:

- `image_size` = 28,
- `patch_size` = 4,
- `num_layers` = 4,
- `num_heads` = 4,
- `hidden_dim` = 64,
- `mlp_dim` = 128,
- `num_classes` = `n_classes`

Where `n_classes` is either 10 or 2 depending on whether the global variable `MULTICLASS` is True or False, respectively. We also replaced the first layer of this model such that it aligns with the shape of the images of MNIST, as it only has one channel.

C WORK DISTRIBUTION

Both researchers worked together on all parts of the research, hence the work was distributed equally.