WHEN DO CONVOLUTIONAL NEURAL NETWORKS STOP LEARNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNNs) is one of the most essential architectures that has shown impressive performance in computer vision tasks such as image classification, detection, and segmentation. In general, an arbitrary number of epochs is used to train such neural networks. In a single epoch, the entire training data—divided by batch size—are fed to the network. However, the optimal number of epochs required to train a neural network is not well established. In practice, validation data is used to identify the generalization gap. To avoid overfitting, it is recommended to stop training when the generalization gap increases. However, this is a trial and error based approach. This raises a critical question: Is it possible to estimate when neural networks stop learning based on only the training data? In this research work, we introduce a hypothesis that analyze the data variation in a layer to predict its near optimal learning capacity. Based on this hypothesis, we predict the near optimal epoch number to train a CNN without using any validation data. We experiment our hypothesis on six different CNN models and on three different datasets (CIFIR 10, CIFIR 100, and SVHN). We save on average 58.49% computational time to train a CNN model. Our code is available at https://github.com/PaperUnderReviewDeepLearning/Optimization.

1 INTRODUCTION

"Wider and deeper are better" has become the rule of thumb to design deep neural network architecture (Guo et al., 2020; Huang et al., 2017; He et al., 2016b; Szegedy et al., 2015; Simonyan & Zisserman, 2014b). Deep neural network requires large amount of data to be trained but how much data we should feed to a deep neural network to gain optimum performance is not well established. Deep neural networks behave "double-descent" curve while traditional machine learning models stuck to the "bell-shaped" curve as deep neural networks have larger model complexity (Belkin et al., 2019). However, in deep neural network the data interpolation is reduced as the data are fed into the deeper layers of the network. This raises a critical question: Can we predict whether the deep neural network keeps learning or not based on the training data behavior?

Convolutional Neural Network (CNN) gains impressive performance on computer vision task (He et al., 2016a). Deeper layer based CNN tends to achieve higher accuracy on vision task (e.g., image classification) (Sinha et al., 2020). In terms of computational time saving, light-weighted CNN architectures are introduced which have a trade-off between speed and accuracy. However, when a CNN architecture reaches its' model capacity and stops significant learning from the training data remains unclear.

In training phase, all training data are fed into the CNN as an epoch. Current practice is to use many epochs (e.g., 200$\sim$500) to train a CNN model. The optimum number of epochs required to train a CNN model is not well researched. To infer whether the model keeps learning or not in each epoch, validation data are used alongside training data. Traditionally, training of the model is stopped when the validation error or generalization gap starts to increase (Goodfellow et al., 2017). Generalization gap indicates model's capability to predict unseen data. However, this current approach is based on trial-and-error. Our research objective is to replace this trial-and-error based approach by algorithmic approach.

We hypothesize that a layer after convolution reaches its near optimal learning capacity if the produced data have significantly less variation. We use our hypothesis to identify the epoch where all

the layers reach their near optimal learning capacity which also represents the model's near optimal learning capacity. Thus, our hypothesis predicts the near optimal epoch number to train a CNN model without using any validation data set.

The selection of optimal epoch number to train a deep neural network is not well established. Followings are some of the recent works that use different epoch number for their experiments: Zhang & He (2020) use 186 epochs, Piergiovanni & Ryoo (2020) use 256 epochs, Peng et al. (2020) use 360 epochs, Khalifa & Islam (2020) use 150 epochs. For CIFIR10 and CIFIR100 dataset Li et al. (2020) and Reddy et al. (2020) use 200 epochs. For ResNet and VGG architecture, Kim et al. (2020) use 200 epochs. Dong et al. (2020) and Liu et al. (2020) also use 200 epochs for their experiment. Huang et al. (2020) use 50$\sim$500 epochs as a range. Curry et al. (2020) use 1000 epochs for their custom dataset. In short, most deep neural models adapt a safe epoch number.

As illustrated by Figure 1, our hypothesis can be deployed as a plug-and-play to any CNN architecture. Our hypothesis does not introduce any additional trainable parameter to the network. At the end of each epoch, our hypothesis verifies the models' learning capacity. The training is terminated when the model reaches its near optimal learning capacity.

The main contributions of this paper can be summarized as:

- We introduce a hypothesis regarding near optimal learning capacity of CNN architecture.
- We examine the data variation across all the layers of a CNN architecture and correlate to its near optimal learning capacity.
- The proposed hypothesis can predict the near optimal epoch number to train a CNN model without using any validation dataset.
- The implementation of the proposed hypothesis can be deployed as plug-and-play to any CNN architecture and does not introduce any additional trainable parameter to the network.
- To test our hypothesis, we conduct image classification experiments on six CNN architectures and three datasets. Adding the hypothesis to the existing CNN architectures save 32% to 78% of the computational time.
- Finally, we provide detail analysis of our hypothesis on different phases of training and how we obtain the near optimal epoch number.

2 Related Work

Validation data are used along side training data to identify generalization gap (Goodfellow et al., 2017). Generalization refers the model's capability to predict unseen data. Increasing generalization gap indicates that the model is going to overfit. It is recommended to stop training the model at that point. However, this is a trial and error based approach that has been widely used in current training strategy. In order to use this strategy, validation dataset is required.

In terms of model complexity, modern neural networks have more complexity compared to the classical machine learning methods. In terms of bias-varience trade off for generalization of neural networks, traditional machine learning methods behave the "bell-shaped", and modern neural networks behave the "double descent curve" (Belkin et al., 2019).

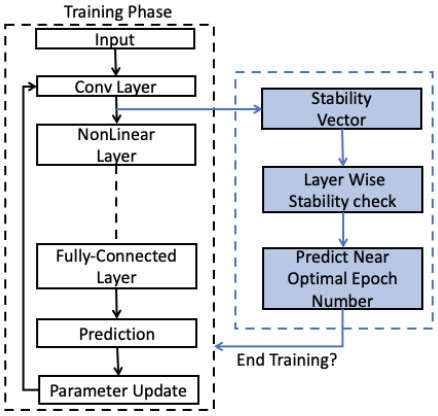


Figure 1: Black doted box represents traditional steps of training a CNN architecture. At each epoch, our plugin (blue doted box) measures data variation after convolution operation. Based on all layers data variation, the plugin decides the continuity of training.

To the best of our knowledge, there is no previous work that investigates at what optimal epoch a CNN model stops learning. However, there are CNN architectures that aim at obtaining best possible accuracy under a limited computational budget based on different hardware and/or applications. This results a series of works towards light-weight

CNN architectures and have speed-accuracy trade-off, including Xception (Chollet, 2017), MobileNet (Howard et al., 2017), ShuffleNet (Zhang et al., 2018), and CondenseNet (Huang et al., 2018). They use FLOP as an indirect metric to compare computational complexity. ShuffleNetV2 (Ma et al., 2018) uses speed as a direct metric while considering memory access cost, and platform characteristics. However, considering epoch number as metric to analyze computation on CNN or at what optimal epoch CNN reaches optimal learning capacity is not well researched.

For a specific dataset and CNN architecture, the usual practice is to adapt a safe epoch number. However, the epoch number selection is random and an arbitrary safe number is picked for most of the experiments. This inspires us to conduct our research work to find out when CNN almost stops learning and to predict near optimal epoch number to train any CNN architecture regardless of dataset.

3 Training Behavior on Deep Neural Network

3.1 Convolutional neural network (CNN)

In deep learning, a typical CNN is composed of stacked trainable convectional layers (LeCun et al., 1998). In one epoch (e), the entire training data is sent by multiple iteration (t) in batch size (N). The entire training data sent in one epoch is expressed as $e = Nt$. The input tensor $\mathbf{X}$ is organised by batch size N, channel number c, height h, and width w as $\mathbf{X}(N, c, h, w)$. A typical CNN convolution operation on n-th layer can be mathematically represented by Equation 1, where θ_w are the learned weights of the convolutional kernel.

$$\mathbf{X}_n = (\theta_w \circledast \mathbf{X}_{n-1}) \tag{1}$$

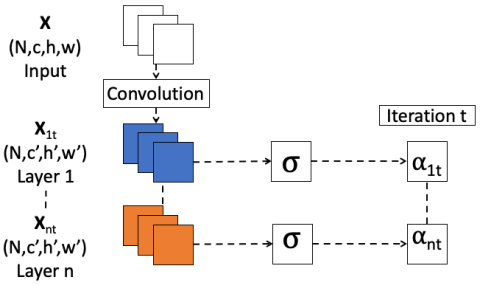


Figure 2: Computing stability vector elements $\alpha_{1t}, \alpha_{2t}, \ldots, \alpha_{nt}$ for n layers at t-th iteration.

3.2 Stability vector

During training phase, we examine whether the deep leaning model is learning or not by measuring data variation after convolution operation. We introduce stability vector S to measure data variation. In every epoch, we construct stability vector for each layer of the deep neural network. The stability vector for e-th epoch and n-th layer is denoted by S_{ne}. At each epoch and each layer, we construct stability vector S_{ne} by computing stability value for all the iterations (t) of an epoch. At t-th iteration, after the convolution of n-th layer, we measure stability value (element of a stability vector) α_{nt} by computing the standard deviation value of $\mathbf{X}_{nt}$ as $\alpha_{nt} = \sigma(\mathbf{X}_{nt})$. The process of constructing the elements of stability vector is shown in Figure 2.

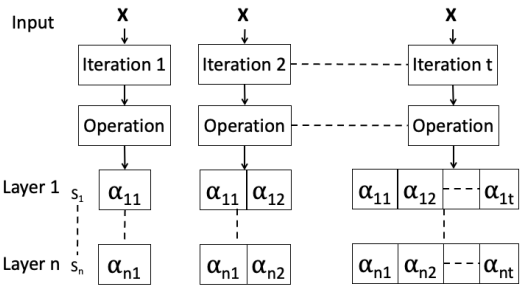


Figure 3: At e-th epoch, the process of constructing stability vectors $S_{1e}, S_{2e}, \ldots, S_{ne}$ for n layers.

After t iterations at n-th layer and e-th epoch, we have the stability vector $S_{ne} = [\alpha_{n1}, \alpha_{n2}, \ldots, \alpha_{nt}]$. The process of constructing stability vectors for all the layers (i.e., layers 1 to n) after t number of iterations at epoch e is shown in Figure 3. At every epoch, we have n number of stability vectors (i.e., based on number of layers) where each vector has size t.

3.3 LAYER AND MODEL STABILITY

We compute the mean of stability vector, μ_n^e, at e-th epoch and n-th layer by the following equation:

$$\mu_n^e = \frac{1}{t}\sum_{i=1}^{t}\alpha_{ni} \tag{2}$$

We define a function p^r that rounds a number to decimal places r. Thus, if $\mu_n^e = 1.23456$, $p^2(\mu_n^e)$ will return 1.23. For each layer n, we compare the mean of stability vector with previous epoch by rounding to decimal places r by using the following equation:

$$\delta_n = p^r(\mu_n^e) - p^r(\mu_n^{e-1}) \tag{3}$$

At n-th layer and e-th epoch, if δ_n equals zero, we consider that the n-th layer is stable on e-th epoch. If all layers shows the stability by $\sum_{i=1}^{n}\delta_i = 0$, then it indicates the possibility that the CNN model becomes stable (i.e., reaches its near optimal learning capacity) and it cannot extract significant information from the training data. To make sure that the CNN model reaches its near optimal learning capacity, we verify the $\sum_{i=1}^{n}\delta_i = 0$ for two more epochs and if the result remains same, we reach to the conclusion that the model reaches near optimal learning capacity and we terminate the training phase. The trained model is now ready for testing environment.

All the variables we use in our hypothesis are not trained via back-propagation, and do not introduce any trainable parameter to the network.

3.3.1 RESNET18 STABILITY ON CIFAR100 DATASET

On CIFAR100 dataset, the total number of training sample is 50000. We consider 64 as the batch size for training (i.e., $N = 64$). So, we need $\frac{50000}{64} = 782$ iterations (i.e., $t = 782$) in an epoch (e) to use entire training data.

At epoch e and layer n, the first iteration constructs the first element (i.e., α_{n1}) of stable vector S_{ne}. In ResNet18 architecture, at epoch e, there are 18 layers and for each layer we construct one stability vector, so we have in total 18 stability vectors (i.e., $S_{1e}, S_{2e}, \ldots, S_{18e}$). The length of each stability vector is 782 because each epoch consists of 782 iterations (Figure 3). Table 1 shows the $p^2(\mu_n^e)$ values for epoch 73 to 76. As the δ_n is 0 for four consecutive epochs, our hypothesis terminates the ResNet18 training on CIFAR100 dataset at epoch 76.

Table 1: $p^2(\mu_n^e)$ values across epoch 73 to 76 for ResNet18 on CIFAR100 dataset ($p^2(\mu_n^e)$) values are from Figure 5d)

Layer	$p^2(\mu_n^{e=73})$	$p^2(\mu_n^{e=74})$	δ_n	$p^2(\mu_n^{e=75})$	δ_n	$p^2(\mu_n^{e=76})$	δ_n
Layer 1	0.14	0.14	0	0.14	0	0.14	0
Layer 5	0.19	0.19	0	0.19	0	0.19	0
Layer 9	0.14	0.14	0	0.14	0	0.14	0
Layer 13	0.09	0.09	0	0.09	0	0.09	0
Layer 18	0.44	0.44	0	0.44	0	0.44	0

4 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of our hypothesis on six different CNN architectures such as ResNet18 (He et al., 2016a), ResNet18+CBS (Sinha et al., 2020), CNN (LeCun et al., 1998), CNN+CBS (Sinha et al., 2020), VGG (Simonyan & Zisserman, 2014a), and VGG+CBS (Sinha et al., 2020). We test these CNN architectures on three different datasets (CIFIR10, CIFIR100 (Krizhevsky et al.), and SVHN (Netzer et al., 2011)) and analyze the Computational Time Saving (CTS) and Top-1 classification accuracy by using our hypothesis. We further provide an ablation study to analyze the influence of our strategy.

4.1 DATASETS, TASKS, AND CNN ARCHITECTURES

To evaluate our hypothesis, we perform image classification task on two standard vision datasets namely CIFAR10 and CIFAR100 which contain images for 10 and 100 classes, respectively. SVHN, the other dataset, is a digit recognition dataset which consists of natural images of the 10 digits collected from the street view. Table 2 shows more details about these datasets.

CNN demonstrated remarkable performance in computer vision task. Both ResNet and VGG are based on CNN architecture and have different variations based on number of layers. We consider ResNet18 and VGG16 variations in our experiment to compare our hypothesis with Sinha et al. (2020). Curriculum by Smoothing (CBS) gains performance on the top of these architectures. CBS controls the amount of high-frequency information during the training phase. CBS augments the training scheme and increases the amount of information in the feature maps so that the network can learn progressively better representation of the data.

Table 2: Dataset

Dataset	**Batch Size N**	**Training Data**	**Training Iteration t**	**Validation Data**	**Validation Iteration**
CIFIR10	64	50000	782	10000	157
CIFIR100	64	50000	782	10000	157
SVHN	64	73257	1145	26032	407

4.2 COMPUTATIONAL TIME SAVING (CTS)

We report the Computational Time Saving (CTS) based on epoch and iteration numbers. We consider 200 epochs as the benchmark number to compare with Sinha et al. (2020) use 200 epochs as default parameter in CBS (Curriculum by Smoothing). Li et al. (2020) use 200 epochs on CIFAR10 and CIFAR100 datasets. Kim et al. (2020) use 200 epochs on VGG and ResNet architecture. As we use CBS, VGG, and ResNet architectures on CIFAR10, CIFAR100 datasets, we compare the complexity based on 200 epochs for all of our experiments (i.e., six CNN architectures, and three datasets). The iteration number is different based on the dataset size but we use fixed batch size number to obtain iteration number for a dataset (Table 2). In Table 2, we show the batch size (N), iteration number (t), training, and validation data. We keep the batch size constant (i.e., 64) for all datasets. That is, in one iteration, the model uses 64 samples.

As an example, the training and validation iteration for CIFAR100 is 782 and 157, respectively (Table 2). In order to run ResNet18 architecture on CIFIR100 dataset, it requires $200 \times 782 + 200 \times 157 = 187800$ (Table 3) iterations. Our hypothesis predicts 76 as the near optimal epoch number to train ResNet18 on CIFAR10. The updated architecture based on our hypothesis requires $76 \times 782 = 59432$ iterations which saves 68.35% computation and gains 0.29 top-1 classification accuracy.

Table 3: Computational Time Saving (CTS) in percentage and Top-1 classification accuracy (Acc.) on CIFAR10, CIFAR100, SVHN datasets. The **bold** numbers represent better scores.

DataSet	CIFIR10				CIFIR100				SVHN			
Model	Training Epoch	Total Iter.	CTS (in %)	Acc.	Training Epoch	Total Iter.	CTS (in %)	Acc.	Training Epoch	Total Iter.	CTS (in %)	Acc.
CNN	200	187800	0	**80.45**	200	187800	0	48.27	200	310400	0	89.68
CNN+Our	**78**	**60996**	**67.52**	79.51	**123**	**96186**	**48.78**	**49.20**	**99**	**113355**	**63.48**	**89.83**
CNN+CBS	200	187800	0	**77.38**	200	187800	0	**46.51**	200	310400	0	**89.48**
CNN+CBS+Our	**128**	**100096**	**46.70**	77.25	**139**	**108698**	**42.12**	46.43	**134**	**153430**	**50.57**	89.25
ResNet18	200	187800	0	**89.35**	200	187800	0	64.34	200	310400	0	**95.05**
ResNet18+Our	**59**	**46138**	**75.32**	89.01	**76**	**59432**	**68.35**	**64.63**	**56**	**64120**	**79.34**	94.34
ResNet18+CBS	200	187800	0	**89.39**	200	187800	0	**65.81**	200	310400	0	**96.17**
ResNet18+CBS+Our	**65**	**50830**	**72.82**	89.06	**74**	**57868**	**69.18**	65.32	**63**	**72135**	**76.76**	95.96
VGG16	200	187800	0	**82.06**	200	187800	0	**48.80**	200	310400	0	**93.87**
VGG16+Our	**109**	**85238**	**54.61**	81.75	**163**	**127466**	**32.12**	48.04	**113**	**129385**	**58.31**	93.63
VGG16+CBS	200	187800	0	**83.68**	200	187800	0	49.11	200	310400	0	94.25
VGG16+CBS+Our	**109**	**85238**	**54.61**	83.55	**148**	**115736**	**38.37**	**50.41**	**125**	**143125**	**53.89**	**94.57**

4.3 ABLATION STUDY

The ablation study results are summarized in Table 3. To evaluate the Computational Time Saving (CTS) and Top-1 classification accuracy (Acc.), we run 36 experiments, 18 of them are conducted without our hypothesis and the rest 18 are conducted with our hypothesis. Our hypothesis predicts the near optimal epoch numbers, which are significantly less than 200, for all the 18 experiments. On these three datasets and across the six CNN architectures, the 200 epoch number is considered a safe one by the respective researchers.

By using our hypothesis, computational time saving ranges from 32.12% to 79.34%. On average, we save 58.49% computational time based on the 36 experiments. In terms of top-1 classification accuracy, the result varies. Compared to the state-of-the-art frameworks, the maximum accuracy we gain for VGG16+CBS architecture on CIFAR100 dataset is 1.3 and the maximum accuracy drop for CNN on CIFAR10 dataset is 0.94. On average, the accuracy drops 0.106, based on the 36 experiments.

5 ANALYSIS OF OUR HYPOTHESIS

To analyze our hypothesis, we study the data behavior at all the layers of CNN architectures during the training phase. We introduce data stability concept in layers of CNN which identifies a CNN model's ability to learn from training data. Our hypothesis predicts the near optimal epoch number required to train a CNN model. We perform ablation study which supports our hypothesis.

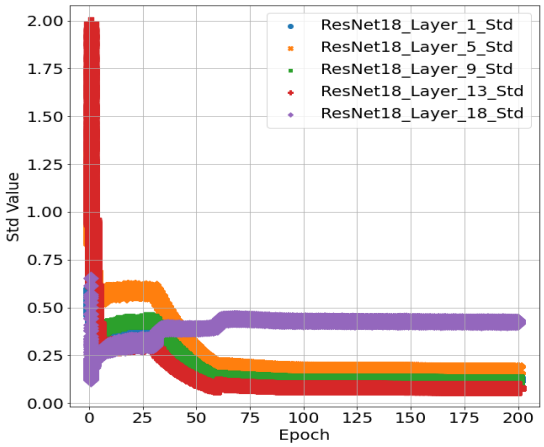


Figure 4: Data stability for five different layers of ResNet18 on CIFAR100 dataset.

5.1 TRAINING BEHAVIOR ANALYSIS

Figure 4 shows the S_{ne} values (for layers 1, 5, 9, 13, and 18) for ResNet18 on CIFAR100 datasets across 200 epochs. Each epoch contains 782 data points for each layer (Table 2). To explain the S_{ne} values behavior, we divide the training phase into the following four different phases to identify near optimal epoch number where a CNN model gets stable and stops learning: initial phase, curved phase, curved phase to stable phase, and stable phase.

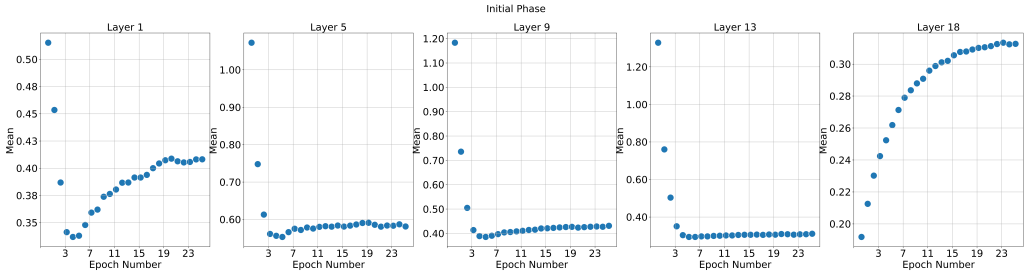
In Figure 5, we show the μ_n^e value which is the mean of S_{ne} values for an epoch e and layer n. We also show the μ_n^e behavior in these phases. Our goal is to identify the 'stable phase' to predict the near optimal epoch number required to train a CNN architecture.

Figure 5 contains more detail explanation of our analysis during these four phases. Each data point represents the mean of stability vector (μ_n^e). We use subplots for different layers to provide better understanding.

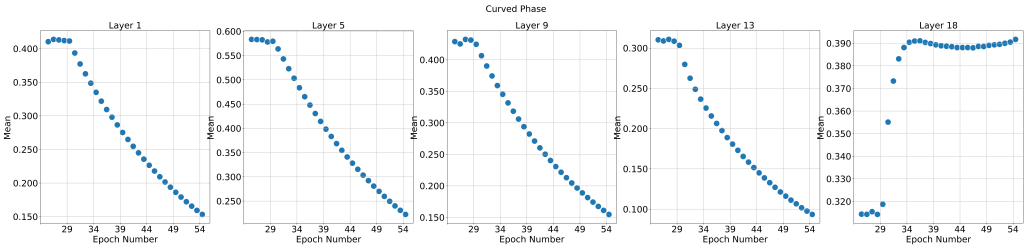
Initial phase refers to the early stage of training. In this phase, S_{ne} values are unstable across all the layers (Figure 4). We also observe a sharp drop of μ_n^e values in most of the layers (Figure 5a). For ResNet18 architecture on CIFAR100 dataset, the approximate range of initial phase is from epoch 1 to epoch 25. The range of all the phases can vary based on CNN architecture and dataset.

It is noteworthy that at layer 1, 5, 9, and 13 the μ_n^e values decrease but at layer 18 the μ_n^e values increase with epochs (Figure 5a, 5b, 5c). The reason behind this is that there is an average pool function used in the last layer of ResNet18 architecture.

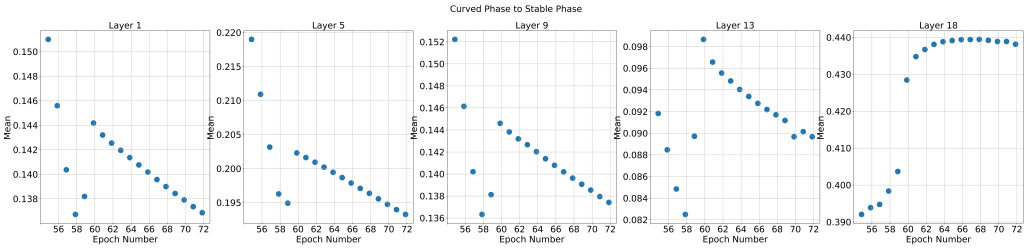
Curved phase refers to the smooth changes of S_{ne} values in training phase. We observe S_{ne} values gradually increase or decrease (Figure 4) in curved phase. For ResNet18 on CIFAR100 dataset, the



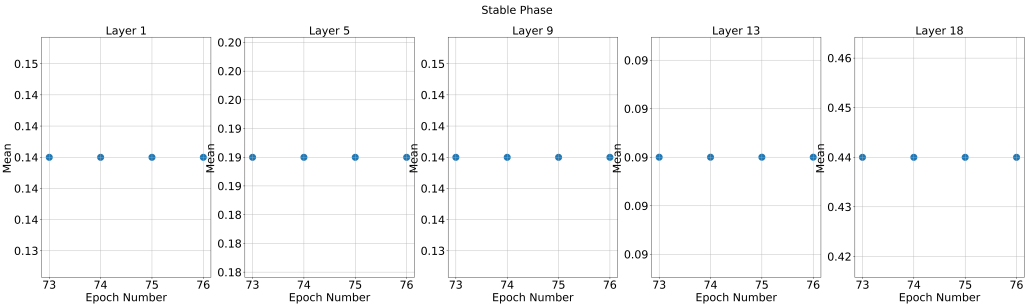
(a) μ_n^e values show the instability during the initial phase of training from epoch 1 to 25 for ResNet18 on CIFAR100 dataset. The instability shows for layer 1, 5, 9, 13, and 18. The sharp drop of μ_n^e values can be observed in the initial phase.



(b) μ_n^e values show the gradual increase or decrease from epoch 26 to 55 for ResNet18 on CIFAR100 dataset. This smooth transition of μ_n^e values creates a curved shape across all layers.



(c) μ_n^e values show significantly low fluctuation as the model gets closer to its near optimal learning capacity.



(d) As the rate of change of μ_n^e values gets significantly low, the probability of getting stable $p^2(\mu_n^e)$ values for consecutive epochs gets higher. At stable phase, δ_n=0 indicates that the CNN reaches its near optimal learning capacity and terminates the training. Our hypothesis predicts 76 as the near optimal epoch number for ResNet18 on CIFAR100 dataset.

Figure 5: ResNet18's data stability (μ_n^e) on CIFIR100 dataset. Figures 5a, 5b, and 5c show μ_n^e values on initial phase, curved phase, and curved phase to stable phase. Figure 5d shows $p^2(\mu_n^e)$ values on stable phase.

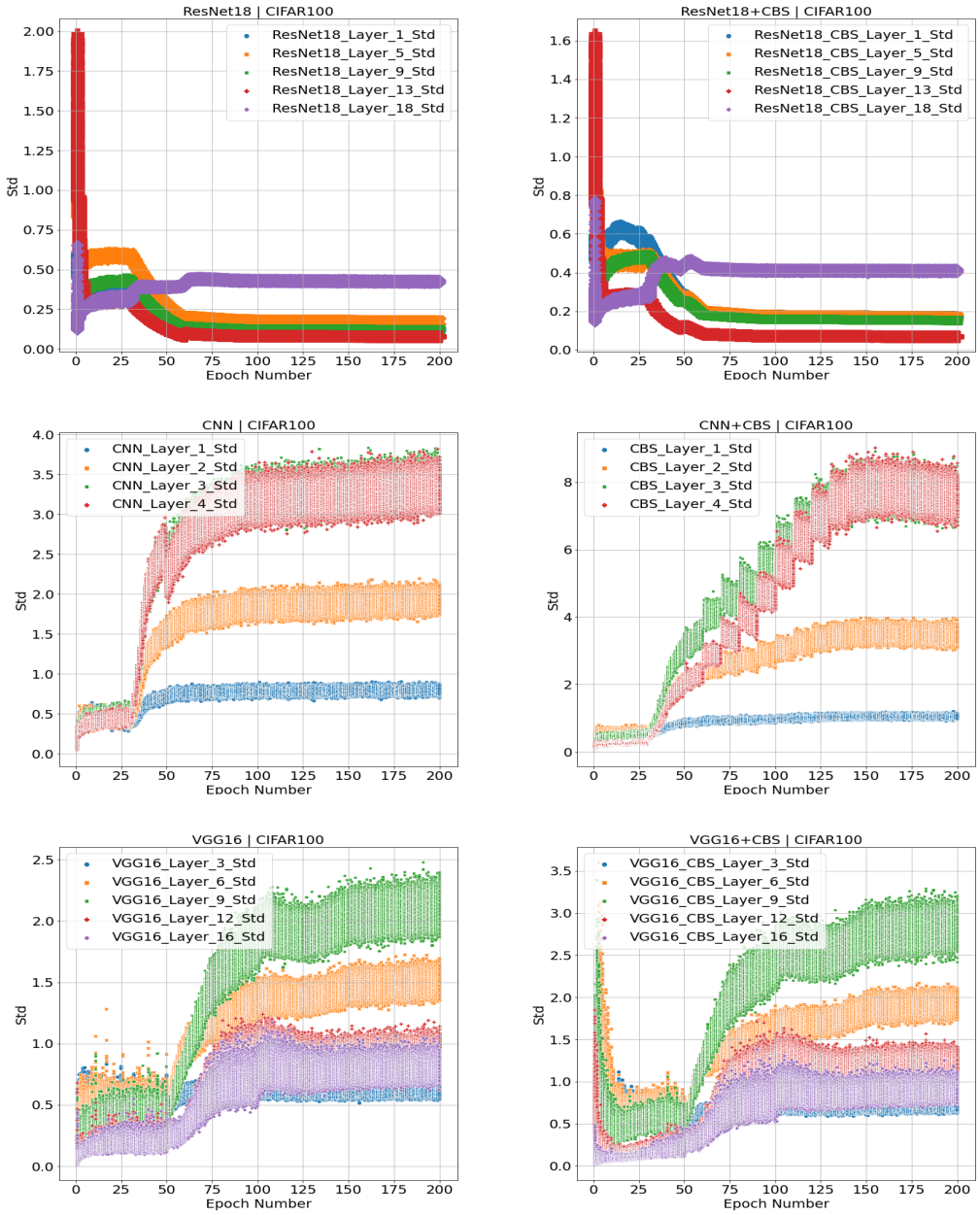


Figure 6: Data stability for different layers of ResNet18, CNN, and VGG16 on CIFAR100 dataset.

approximate range of curved phase is from epoch 26 to epoch 55 (Figure 4). Figure 5b also shows that μ_n^e values across all layers create a smooth shaped curve.

Curved phase to stable phase in the start of this phase, the μ_n^e values fluctuate but as training goes on the fluctuation gradually decreases with epochs. Figure 5c shows the μ_n^e values for ResNet18 on CIFAR100 dataset ranging epoch 56 to 72.

Stable phase refers to the range of epochs where the change of μ_n^e values are almost insignificant across all the layers. For each layer n, we compare mean of stability vector with previous epoch by rounding to decimal places r using Equation 3 to compute δ_n. If there is no significant difference between two epochs' mean of stability vectors for all the layers, it indi-

cates the possibility that the CNN model may get stable. To make sure that the CNN model reaches its near optimal learning capacity, we verify the $\sum_{i=1}^{n} \delta_i = 0$ for two more epochs.

Figure 5c shows stable region for ResNet18 architecture on CIFAR100 dataset. In the figure, we can observe that after two decimal points there is no change in μ_n^e values from epoch 73 to 76 for all n layers. So, for ResNet18 architecture and on CIFAR100 dataset, the training is terminated at epoch 76.

It is noteworthy that in the stable phase, we compute δ_n by using function p^r and we choose the value of $r = 2$. Choosing the value of $r = 1$ causes very early stop of the training, while $r = 3$ does not guarantee to stop training at the earliest optimal epoch. Choosing $r > 3$ does not stop training even if the epoch number is large enough[1].

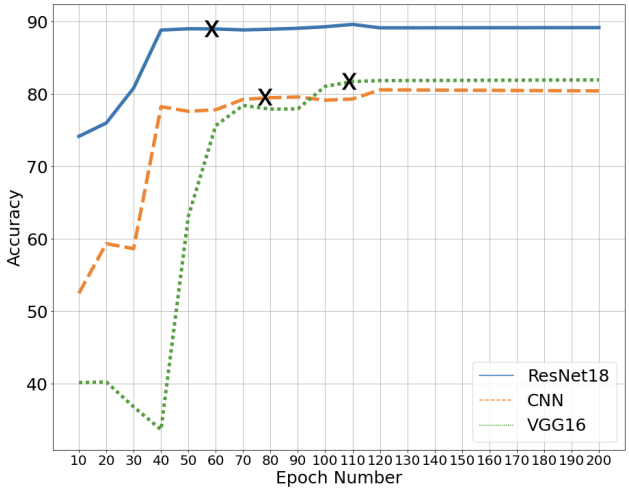


Figure 7: ResNet18, CNN, and VGG16's testing accuracy on CIFIR10 dataset based on training the models ranging 10–200 epochs. It also shows the testing accuracy based on the near optimal epoch number predicted by our hypothesis to train the models, marked by X (best viewed in color).

We observe data stability (S_{ne}) for six CNN architectures on CIFAR10, CIFAR100, and SVHN datasets. Figure 6 shows these six CNN architectures' S_{ne} values during training phase on CIFAR100 dataset.

5.2 Generalization Ability

We also analyze the generalization ability of CNN architectures across wide range of epoch numbers in training. Figure 7 shows the testing accuracy (top-1 classification) of ResNet18, VGG16, and CNN on CIFIR10 dataset where 10 to 200 epochs are used for training. It also includes the testing accuracy where the models are trained by the epoch number predicted by our hypothesis (marked by X). From Figure 7, it is clear that all the models testing accuracy reach to a stable stage after certain number of training epoch. Our hypothesis predicts that ResNet18, VGG16, and CNN reach the near optimal learning capacity at epochs 59, 109, and 78, respectively. Figure 7 also supports our hypothesis because all these models' generalization ability (i.e., the ability to predict on unseen data) does not significantly improve after the near optimal epoch number predicted by the hypothesis.

6 Conclusion

In this work, we introduce and analyze the data stability of CNN architectures in layers. We discover similar data stability pattern in six CNN architectures on three datasets. We propose a hypothesis that can identify near optimal epoch number where CNN architecture stops learning. We also show that current practices select random and safe epoch number to conduct experiment, whereas our hypothesis predicts the near optimal epoch number which is significantly less than the safer option. The proposed hypothesis does not require validation dataset as well as does not introduce any trainable parameter to the network. The implementation of the hypothesis can easily be attached to any existing CNN architecture. We also provide ablation study that shows the computational time saving across six CNN architectures on three datasets. Identifying near optimal epoch number as well as computational time saving support the effectiveness of our hypothesis. Future work is to identify other statistical properties of the data in deeper layers of deep learning model. It will help us to better understand the insight about how data change its' behavior in deeper layers.

[1]We checked with $r = 4$ for ResNet18 architecture on CIFAR 100 dataset and for VGG16 architecture on SVHN dataser and the models do not stop training even after 350 epoch

REFERENCES

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Michael Curry, Ping-Yeh Chiang, Tom Goldstein, and John Dickerson. Certifying strategyproof auction networks. *Advances in Neural Information Processing Systems*, 33, 2020.

Jiangxin Dong, Stefan Roth, and Bernt Schiele. Deep wiener deconvolution: Wiener meets deep learning for image deblurring. In *34th Conference on Neural Information Processing Systems*, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2017.

Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks. *Advances in Neural Information Processing Systems*, 33, 2020.

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, 2016a.

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, 2016b.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

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.

Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2752–2761, 2018.

Qian Huang, Horace He, Abhay Singh, Yan Zhang, Ser-Nam Lim, and Austin Benson. Better set representations for relational reasoning. *Advances in Neural Information Processing Systems*, 2020.

Muhammad Khalifa and Aminul Islam. Will your forthcoming book be successful? predicting book success with cnn and readability scores. *arXiv preprint arXiv:2007.11073*, 2020.

Woojeong Kim, Suhyun Kim, Mincheol Park, and Geunseok Jeon. Neuron merging: Compensating for pruned neurons. *Advances in Neural Information Processing Systems*, 33, 2020.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Guilin Li, Junlei Zhang, Yunhe Wang, Chuanjian Liu, Matthias Tan, Yunfeng Lin, Wei Zhang, Jiashi Feng, and Tong Zhang. Residual distillation: Towards portable deep neural networks without shortcuts. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pp. 8935–8946, 2020.

Rui Liu, Tianyi Wu, and Barzan Mozafari. Adam with bandit sampling for deep learning. *Advances in Neural Information Processing Systems*, 2020.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Daiyi Peng, Xuanyi Dong, Esteban Real, Mingxing Tan, Yifeng Lu, Gabriel Bender, Hanxiao Liu, Adam Kraft, Chen Liang, and Quoc Le. Pyglove: Symbolic programming for automated machine learning. *Advances in Neural Information Processing Systems*, 33, 2020.

AJ Piergiovanni and Michael S Ryoo. Avid dataset: Anonymized videos from diverse countries. *Advances in Neural Information Processing Systems*, 2020.

Manish Vuyyuru Reddy, Andrzej Banburski, Nishka Pant, and Tomaso Poggio. Biologically inspired mechanisms for adversarial robustness. *Advances in Neural Information Processing Systems*, 33, 2020.

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

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

Samarth Sinha, Animesh Garg, and Hugo Larochelle. Curriculum by smoothing. *Advances in Neural Information Processing Systems*, 33, 2020.

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.

Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. In *NeurIPS*, 2020.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.