WHEN DO CONVOLUTIONAL NEURAL NETWORKS STOP LEARNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNNs) have demonstrated outstanding performance in computer vision tasks such as image classification, detection, segmentation, and medical image analysis. 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. In practice, validation error with training loss is used to estimate the neural network's generalization, which indicates the optimal learning capacity of the network. Current practice is to stop training when the training loss decreases and the gap between training and validation error increases (i.e., the generalization gap) to avoid overfitting. However, this is a trial-and-error-based approach which raises a critical question: Is it possible to estimate when neural networks stop learning based on training data? This research work introduces a hypothesis that analyzes the data variation across all the layers of a CNN variant to anticipate its near-optimal learning capacity. In the training phase, we use our hypothesis to anticipate the near-optimal learning capacity of a CNN variant without using any validation data. Our hypothesis can be deployed as a plug-and-play to any existing CNN variant without introducing additional trainable parameters to the network. We test our hypothesis on six different CNN variants and three different datasets (CIFAR10, CIFAR100, and SVHN). The result based on these CNN variants and datasets shows that our hypothesis saves 58.49% of computational time (on average) in training. 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., 2016; Szegedy et al., 2015; Simonyan & Zisserman, 2014). Deep neural networks behave "double-descent" curve while traditional machine learning models are stuck to the "bell-shaped" curve as deep neural networks have larger model complexity (Belkin et al., 2019). Deep neural networks require a large amount of data to be trained. The data interpolation is reduced in the deep neural network as the data are fed into the deeper layers of the network. However, a core question remains: 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 tasks (Sinha et al., 2020). Specifically, deeper layer-based CNN tends to achieve higher accuracy on vision tasks such as image classification (Sinha et al., 2020), image segmentation (Long et al., 2015), object detection (Redmon et al., 2016). Light-weighted CNN variants are introduced for computational time saving, with a trade-off between speed and accuracy. However, it remains unclear when a CNN variant reaches its near-optimal learning capacity and stops significant learning from training data.

In general, all training data are fed into a deep neural model as an epoch in the training phase. The current practice uses many epochs (e.g., 200∼500) to train a deep neural model. The selection of optimal epoch

numbers to train a deep neural model 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 to speed the training of transformer-based language models, Piergiovanni & Ryoo (2020) use 256 epochs for their public video dataset to action recognition , Peng et al. (2020) use 360 epochs for programming AutoML based on symbolic programming, Khalifa & Islam (2022) use 150 epochs for pretrained sentence embeddings along with various readability scores for book success prediction. Another trend is to pick the same epoch number for a specific dataset or deep neural model. For example, Li et al. (2020) and Reddy et al. (2020) use 200 epochs for CIFAR10 and CIFAR100 dataset. Sinha et al. (2020) and Kim et al. (2020) use 200 epochs for ResNet and VGG architecture. Dong et al. (2020) and Liu et al. (2020) also use 200 epochs for their two simple global hyperparameters that efficiently trade off between latency and accuracy experiment. Huang et al. (2020) use 50∼500 epochs as a range for their synthetic image experiments. Curry et al. (2020) use 1000 epochs for their custom dataset. In short, most deep neural models adapt a safe epoch for their training.

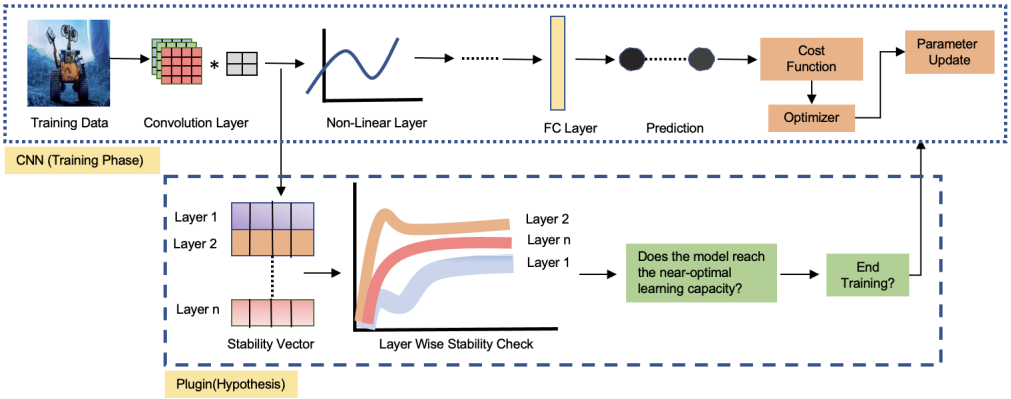


Figure 1: Top doted box represents traditional steps of training a CNN variant. At each epoch, our plugin (bottom doted box) measures data variation after convolution operation. Based on all the layers data variation, the plugin decides the continuity of training.

Validation data are used alongside training data to estimate the generalization error during or after training (Goodfellow et al., 2017). Traditionally, training of the model is stopped when the validation error or generalization gap starts to increase (Goodfellow et al., 2017). The generalization gap indicates the model's capability to predict unseen data. However, the current approach of early stopping is based on trial-and-error. This is because it monitors the average loss function on the validation set and continues training until it falls below the value of the training set objective, at which the early stopping procedure is halted (Goodfellow et al., 2017). This strategy avoids the high cost of retraining the model from scratch, but it is not as well behaved (Goodfellow et al., 2017). For example, the objective on the validation set may ever not reach the target value, so this strategy is not even guaranteed to terminate (Goodfellow et al., 2017). Our research objective is to replace this trial-and-error-based approach with an algorithmic approach to anticipate the near-optimal learning capacity while training a deep learning model. To narrow down the scope of this work, we choose CNN as a member of broader deep learning models.

Generally, a CNN has some basic functions to conduct the training phase, as illustrated in the top doted box in Figure 1. In practice, a dataset is divided into three parts: Training, validation, and testing. A CNN variant can have convolution, non-linear, and fully connected (FC) layers, and the order of these layers can vary based on the variant. The cost function is the technique of evaluating the performance of a CNN variant, and the optimizer modifies the attributes such as weights and learning rate of a CNN variant to reduce the overall loss and improve accuracy.

We hypothesize (illustrated by bottom dotted box in Figure 1) that a layer after convolution operation reaches its near-optimal learning capacity if the produced data have significantly less variation. We use this hypoth-

esis to identify the epoch where all the layers reach their near-optimal learning capacity, representing the model's near-optimal learning capacity. Thus, at each epoch, the proposed hypothesis verifies if the CNN variant reaches its near-optimal learning capacity without using validation data. Our hypothesis terminates the CNN's training when it reaches its near-optimal learning capacity. The hypothesis does not change the learning dynamics of the existing CNN variants or the design of a CNN architecture, cost function, or optimizer. As a result, the hypothesis can be applied to any CNN variant as a plug-and-play after the convolution operation. In summary, any CNN variant that uses training data and/or validation data by multiple epochs can utilize our hypothesis.

It is worth mentioning that our hypothesis does not introduce any trainable parameter to the network. As a result, our hypothesis can be deployed on any wide and deep or compact CNN variant. The main contributions of this paper can be summarized as:

- We introduce a hypothesis regarding near optimal learning capacity of a CNN variant without using any validation data.
- We examine the data variation across all the layers of a CNN variant and correlate it to the model's near-optimal learning capacity.
- The implementation of the proposed hypothesis can be embodied as a plug-and-play to any CNN variant and does not introduce any additional trainable parameter to the network.
- To test our hypothesis, we conduct image classification experiments on six CNN variants and three datasets. Embodying the hypothesis to train the existing CNN variants saves 32% to 79% of the computational time.
- Finally, we provide a detailed analysis of how the proposed hypothesis verifies the CNN variants' optimal learning capacity.

2 Related Work

Modern neural networks have more complexity than classical machine learning methods. In terms of bias-variance 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).

In deep neural networks, validation data are used alongside training data to identify the generalization gap (Goodfellow et al., 2017). Generalization refers to the model's capability to predict unseen data. The 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 widely used in the current training strategy. In order to use this strategy, a validation dataset is required.

Duvenaud et al. (2016); Mahsereci et al. (2017); Bonet et al. (2021) proposed an early stopping method without a validation dataset. However, Duvenaud et al. (2016); Mahsereci et al. (2017) rely on gradient-related statistics and fail to generalize to more advanced optimizers such as those based on momentum. Both of the works require hyperparameter tuning as well. Bonet et al. (2021) designed early stopping method specifically for one particular framework, not a generalized solution.

There are some CNN architectures that aim to obtain the 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). These works 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, we consider epoch number as a metric to analyze the computational time of training a CNN variant.

The usual practice is to adopt a safe epoch number for a specific dataset and a CNN variant. However, the epoch number selection is random, and an arbitrary safe number is picked for most of the experiments. This inspires us to investigate when a CNN variant almost stops learning significantly from the training data.

3 Training Behavior of Convolutional Neural Network

3.1 Convolutional neural network (CNN)

To denote the convolutional operation of some kernel θ_k on some input X, we use $\theta_k \circledast X$. In deep learning, a typical CNN is composed of stacked trainable convolutional layers (LeCun et al., 1998), pooling layers (Boureau et al., 2010), and non-linearities (Nair & Hinton, 2010).

In a single epoch (e), the entire training data (i.e., the number of training samples) is sent by multiple iteration (t) with batch size (N). Thus the number of training samples (D_{train}) sent in a single epoch is expressed by the following equation:

$$D_{\text{train}} = Nt \tag{1}$$

The input tensor X is organised by batch size N, channel number c, height h, and width w as $X(N, c, h, w)$. A typical CNN convolution operation at n-th layer and at t-th iteration can be mathematically represented by Equation 2, where θ_k are the learned weights of the kernel.

$$X_n^t = (\theta_k \circledast X_{n-1}^t) \tag{2}$$

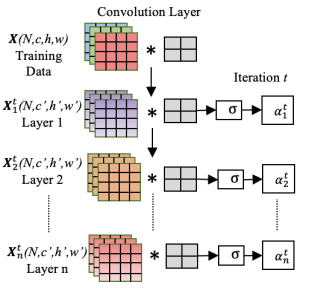


Figure 2: At t-th iteration, the process of computing stability values $\alpha_1^t, \alpha_2^t, \ldots, \alpha_n^t$ for 1 to n layers.

3.2 Stability vector

In the training phase, we examine whether the CNN model keeps learning or not by measuring data variation after convolution operation. To do that we introduce the concept of stability value and stability vector. After the convolution operation at t-th iteration and n-th layer, we measure the stability value (element of a stability vector) α_n^t by computing the standard deviation value of X_n^t as $\alpha_n^t = \sigma(X_n^t)$. The process of constructing stability values is shown in Figure 2.

At e-th epoch and at n-th layer, we construct stability vector $S_n^e = [\alpha_n^1, \alpha_n^2, \ldots, \alpha_n^t]$ by computing the stability values for all the iterations (t). At e-th epoch, the process of constructing stability vectors for all the layers (i.e., layers 1 to n) after t number of iterations is shown in Figure 3. Thus at each epoch, we have n number of stability vectors (i.e., based on the number of layers) with size t (i.e., the number of iterations).

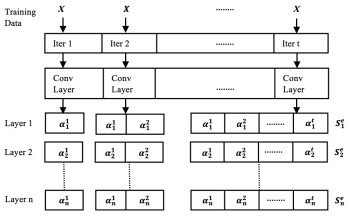


Figure 3: At e-th epoch, the process of constructing stability vectors S_1^e, S_2^e, …, S_n^e for 1 to n layers.

3.3 Layer and model stability

Significantly less data variation of a particular layer's stability vectors for consecutive epochs indicates that that layer of the CNN gets stable (i.e., fails to learn significant information from training data). When all the layers of the model get stable, it implies the possibility that the model reaches its near-optimal learning capacity.

In order to measure the data variation of a layer for two consecutive epochs, at first we compute the mean of stability vector, μ_n^e, at e-th epoch and n-th layer by the following equation:

$$\mu_n^e = \frac{1}{t} \sum_{i=1}^{t} \alpha_n^i \tag{3}$$

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

$$\delta_n^e = p^r(\mu_n^e) - p^r(\mu_n^{e-1}) \tag{4}$$

At e-th epoch and n-th layer, if δ_n^e equals zero, we consider that the n-th layer is stable at e-th epoch. If all the layers show the stability by using $\sum_{i=1}^{n} \delta_i^e = 0$ indicates the possibility that the CNN model gets stable (i.e., reaches its near-optimal learning capacity) at e-th epoch. It also means that the model does not extract any more significant information from the training data. To make sure that the CNN model reaches its near-optimal learning capacity, we repeat using $\sum_{i=1}^{n} \delta_i^e = 0$ for two more epochs (i.e., epochs $e+1$, and $e+2$). If the result remains the same, we conclude that the model reaches its near-optimal learning capacity and we terminate the training phase. The trained model is now ready for the 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 A WALK-THROUGH EXAMPLE OF MODEL STABILITY ON RESNET18 ARCHITECTURE (USING CIFAR100 DATASET)

In CIFAR100 dataset, the total number of training sample is 50000. We consider 64 as the batch size for training (i.e., $N = 64$). So, in each epoch (e), the iteration number is $\frac{50000}{64} = 782$ (i.e., $t = 782$).

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 7b)

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

At e-th epoch and n-th layer, the first iteration constructs the first element (i.e., α_n^1) of stable vector S_n^e. 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., $S_1^e, S_2^e, \ldots, S_{18}^e$). 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 74 to 76. As the δ_n^e is 0 for three consecutive epochs, our hypothesis terminates the ResNet18 training on CIFAR100 dataset at epoch 76.

4 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of our hypothesis on six different CNN variants such as ResNet18 (He et al., 2016), ResNet18+CBS (Sinha et al., 2020), CNN (LeCun et al., 1998), CNN+CBS (Sinha et al., 2020), VGG16 (Simonyan & Zisserman, 2014), and VGG16+CBS (Sinha et al., 2020). We test these CNN variants on three different datasets (i.e., CIFAR10, CIFAR100 (Krizhevsky et al., 2009), and SVHN (Netzer et al., 2011)) and analyze the computational time saving (CTS) and Top-1 classification accuracy by embodying our hypothesis. We further provide an ablation study to analyze the influence of our strategy.

Table 2: Dataset

Dataset	Batch Size (N)	Train. Data (D_{train})	Train. Iter. (t_{train})	Valid. Data (D_{val})	Valid. Iter. (t_{val})
CIFAR10	64	50000	78210000		157
CIFAR100	64	50000	78210000		157
SVHN	64	73257	114526032		407

4.1 CNN VARIANTS, DATASETS, AND TASKS

To evaluate our hypothesis, we perform the image classification task on two standard vision datasets, CIFAR10 and CIFAR100, containing images for 10 and 100 classes, respectively.

SVHN, the other dataset, is a digit recognition dataset that 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 tasks. Both ResNet (He et al., 2016) and VGG (Simonyan & Zisserman, 2014) are based on CNN architecture and have different variations based on the number of layers. We consider ResNet18 (He et al., 2016) and VGG16 (Simonyan & Zisserman, 2014) variations in our experiment. Curriculum by Smoothing (Sinha et al., 2020)(CBS) is a general method for training CNNs, which can be applied to any CNN variant. CBS controls the amount of high-frequency information during the training phase. It augments the training scheme and increases the amount of information in the feature maps so that the network can progressively learn a better representation of the data. CBS applied to CNN variants, such as ResNet18+CBS and VGG16+CBS, improve the accuracy of image classification tasks. We also embody our hypothesis with ResNet18+CBS and VGG16+CBS variants in the experiment.

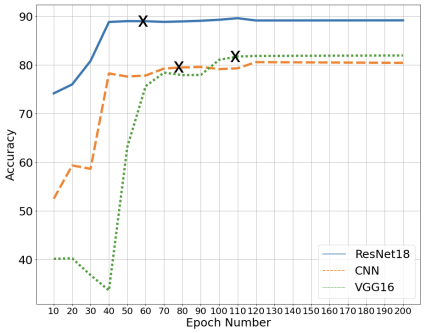


Figure 4: The horizontal axis shows the epoch number (ranging from 10–200) used to train the ResNet18, CNN, and VGG16 on the CIFAR10 dataset. The vertical axis shows the testing accuracy of those models. The X mark shows the testing accuracy and the epoch number to train a CNN variant based on the near-optimal learning capacity anticipated by our hypothesis (best viewed in color).

4.2 Computational time saving (CTS)

Let the total number of epochs required to train a CNN variant be E. Then, based on Equation 1, we compute the total iteration (training iteration and validation iteration) needed in E epochs to train a CNN variant by the following equation[1]:

$$t_{total} = E(\frac{D_{train}}{N} + \frac{D_{val}}{N}) \tag{5}$$

Computational time saving (CTS) between model m_1 and m_2 defines how much less time (i.e, percentage decrease) in terms of total iteration (i.e., t_{total}) required by m_1 to complete the training than m_2. For an example, in order to train ResNet18 architecture on CIFAR100 dataset, the total number of iteration required based on Equation 5 is $200((50000/64) + (10000/64) = 187800$. At 76 epoch, our hypothesis anticipates that ResNet18 reaches its near-optimal learning capacity and terminates the training. By embodying our hypothesis in ResNet18 on the CIFAR100 dataset requires $76(50000/64) = 59432$ iterations to train which saves $\frac{(187800-59432)}{187800}$ = 68.35% computation and gains $\pm$ 0.30 top-1 classification accuracy.

We consider 200 epochs as the benchmark epoch number. CBS (Sinha et al., 2020) use 200 epochs in their experiments. Li et al. (2020) use 200 epochs on CIFAR10 and CIFAR100 datasets. Kim et al. (2020) use 200 epochs on VGG and ResNet variants. We use CBS, VGG, and ResNet architectures on CIFAR10, CIFAR100 datasets and compare the CTS based on 200 epochs for all of our experiments (i.e., six CNN architectures and three datasets). We keep the batch size constant (i.e., 64) for all the datasets. That is, in one iteration, the model uses 64 samples.

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. 200 epoch number is considered safe by the respective researchers on these three datasets and across the six CNN variants. For all 18 experiments,

[1]Symbols are defined in Table 2.

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	CIFAR10				CIFAR100				SVHN			
Model	Train. Epoch	Total Iter.	CTS (in %)	Acc.	Train. Epoch	Total Iter.	CTS (in %)	Acc.	Train. Epoch	Total Iter.	CTS (in %)	Acc.
CNN	200	187800	0	**80.4**±0.2	200	187800	0	48.2 ± 0.2	200	310400	0	89.6±0.2
CNN+Our	**78**	**60996**	**67.52**	79.5±0.2	**123**	**96186**	**48.78**	**49.2**±0.2	**99**	**113355**	**63.48**	**89.8**±0.2
CNN+CBS	200	187800	0	**77.3**±0.2	200	187800	0	**46.5**±0.2	200	310400	0	**89.4**±0.2
CNN+CBS+Our	**128**	**100096**	**46.70**	77.2±0.2	**139**	**108698**	**42.12**	46.4±0.2	**134**	**153430**	**50.57**	89.2±0.2
ResNet18	200	187800	0	**89.3**±0.3	200	187800	0	64.3±0.3	200	310400	0	**95.0**±0.2
ResNet18+Our	**59**	**46138**	**75.32**	89.0±0.3	**76**	**59432**	**68.35**	**64.6**±0.3	**56**	**64120**	**79.34**	94.3±0.2
ResNet18+CBS	200	187800	0	**89.3**±0.3	200	187800	0	**65.8**±0.3	200	310400	0	**96.1**±0.2
ResNet18+CBS+Our	**65**	**50830**	**72.82**	89.0±0.3	**74**	**57868**	**69.18**	65.3±0.3	**63**	**72135**	**76.76**	95.9±0.2
VGG16	200	187800	0	**82.0**±0.2	200	187800	0	**48.8**±0.3	200	310400	0	**93.8**±0.2
VGG16+Our	**109**	**85238**	**54.61**	81.7±0.2	**163**	**127466**	**32.12**	48.0±0.3	**113**	**129385**	**58.31**	93.6±0.2
VGG16+CBS	200	187800	0	**83.6**±0.3	200	187800	0	49.1±0.3	200	310400	0	94.2±0.2
VGG16+CBS+Our	**109**	**85238**	**54.61**	83.5±0.3	**148**	**115736**	**38.37**	**50.4**±0.3	**125**	**143125**	**53.89**	**94.5**±0.2

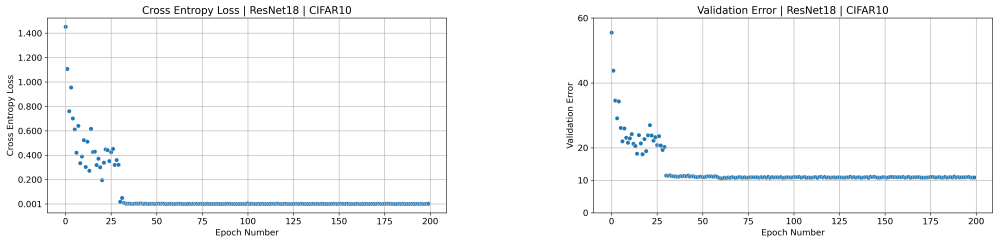


Figure 5: The cross entropy loss (top) and the validation error (bottom) are shown up to 200 epochs for ResNet18 on the CIFAR10 dataset.

our hypothesis anticipates the near-optimal learning capacity of CNN variants which require significantly less than 200 epochs to train. 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 18 experiments. We report the mean accuracy over five different seeds.

4.4 Generalization and near-optimal learning capacity

In our experiments, we work with six different CNN variants. For optimization, we use stochastic gradient descent (SGD) with the same learning rate scheduling, momentum and weight decay as stated in the original papers (Sinha et al., 2020; He et al., 2016), without hyper-parameter tuning. The task objective for all image classification experiments is a standard unweighted multi-class cross-entropy loss (Sinha et al., 2020).

Novak et al. (2018) conducts an empirical study about generalization by using thousands of models with various fully-connected architectures, optimizers, and other hyper-parameter on image classification datasets. For the image classification task, based on the wide range of experiments on the CIFAR10 dataset, Novak et al. (2018) comprehended that train loss does not correlate well with generalization. In the 18 experiments without our hypothesis (i.e., using validation data), we observe similar behavior in the training phase. As an example, Figure 5 shows the cross-entropy (CE) loss and validation error on the CIFAR10 dataset for ResNet18 architecture. The CE loss and validation error reduce in the beginning phase of training. However,

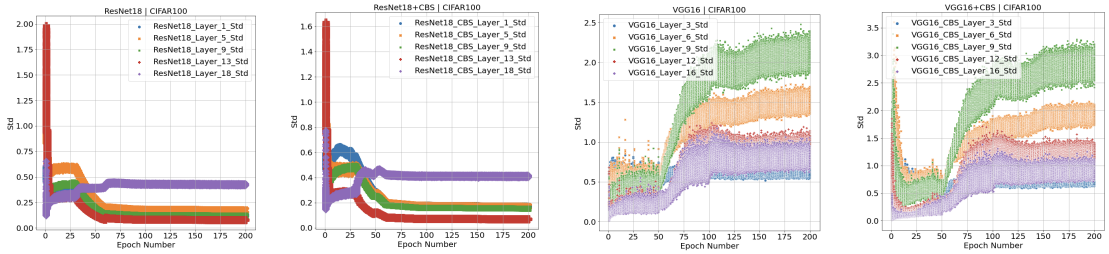
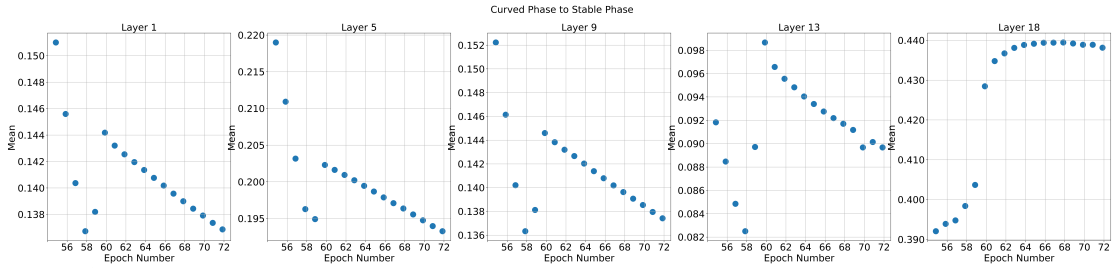
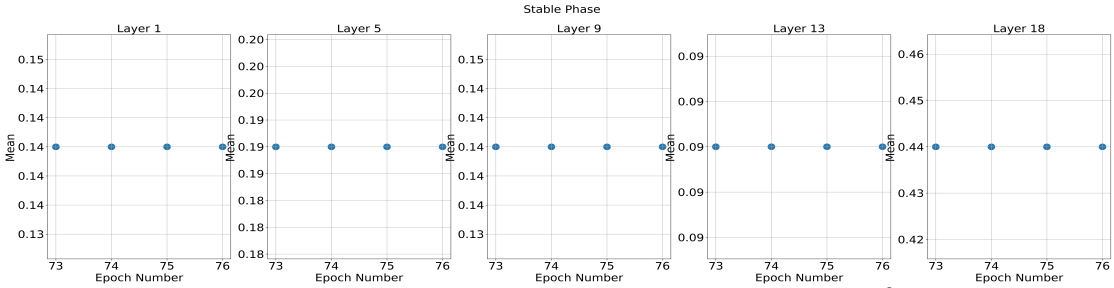


Figure 6: Data variation after convolution operation for different layers of ResNet18, VGG16 and their CBS variants on the CIFAR100 dataset for 200 epochs.



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



(b) 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^e=0 indicates that the CNN reaches its near optimal learning capacity at epoch 76 and terminates the training.

Figure 7: Mean of stability values (μ_n^e) at ResNet18 on the CIFAR100 dataset. Figures 7a show μ_n^e values on curved phase to stable phase. Figure 7b shows $p^2(\mu_n^e)$ values on stable phase.

after that, the generalization gap (i.e., the increase of validation error with CE loss) does not significantly increase. Thus, it is not guaranteed to early stop the training by using validation data.

We further analyze the generalization ability of CNN variants across a wide range of epoch numbers to train. Figure 4 shows the top-1 classification accuracy of ResNet18, VGG16, and CNN on the CIFAR10 dataset where 10 to 200 epochs are used for training. Figure 4 shows that all the model's testing accuracy reaches a stable stage after a certain number of training epochs. Our hypothesis anticipates that ResNet18, VGG16, and CNN reach the near-optimal learning capacity at epochs 59, 109, and 78, respectively (marked by X). In summary, Figure 4 shows that the CNN variants generalization ability (i.e., the ability to predict on unseen data) does not significantly improve after the near-optimal learning capacity anticipated by our hypothesis.

5 Training Behavior Analysis

This section provides a detailed analysis of the pattern we observed in the training phase of CNN variants. Top-left of Figure 6 shows the S_n^e values (for layers 1, 5, 9, 13, and 18) for ResNet18 on CIFAR100 datasets

across 200 epochs. Each epoch contains 782 data points (i.e., t_{train} in Table 2) for each layer. To explain the S_n^e values behavior, we divide the training phase into the following four different phases[2] to anticipate near-optimal learning capacity of CNN variants: Initial phase, curved phase, curved phase to stable phase, and stable phase. It is noteworthy that the range of all the phases can vary based on CNN variants and dataset.

At e-th epoch and n-th layer, each data point of Figure 8 shows the μ_n^e value which measure the mean of stability vector (S_n^e). We also show the behavior of μ_n^e approaching the stable phase. Our goal is to identify the 'stable phase' to anticipate the near-optimal learning capacity of CNN variants. We use subplots for different layers in Figure 8 to provide better understanding.

Curved phase to stable phase refers to the indication that CNN gets closer to its near-optimal learning capacity. For ResNet18 on the CIFAR100 dataset, we consider the curved phase to stable phase's approximate range from epoch 56 to epoch 72. At the start of this phase, the μ_n^e values fluctuate, but as the training goes on, the fluctuation gradually increase or decrease with epochs. Figure 7a 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 the mean of stability vector with its previous epoch by rounding to decimal places r using Equation 4 to compute δ_n^e. If there is no significant difference between two epochs' mean of stability vectors for all the layers, in that case, it indicates the possibility that the CNN variant is close to its near-optimal learning capacity. To make sure that the CNN variant reaches its near-optimal learning capacity, we verify the $\sum_{i=1}^{n} \delta_i^e = 0$ for two more epochs. Figure 7b shows the stable region for ResNet18 on the CIFAR100 dataset. In Figure 7b, we can observe that after two decimal points, there are no changes in μ_n^e values from epoch 73 to 76 for all n layers. Thus, our hypothesis terminates the training of ResNet18 on the CIFAR100 dataset at epoch 76.

It is noteworthy that in the stable phase, we compute δ_n^e by using the function p^r, and we choose the value of $r=2$. Choosing the value of $r=1$ causes a very early stop of the training, while $r=3$ does not guarantee stopping training at the near-optimal learning capacity. Choosing $r>3$ does not stop training even if the epoch number is large enough[3]. We observe a similar pattern of data variation after convolution operation (S_n^e) for all the six CNN variants on CIFAR10, CIFAR100, and SVHN datasets. Figure 6 shows S_n^e values of these six CNN variants during the training phase on the CIFAR100 dataset.

6 Conclusion

In this paper, we analyze the data variation of a CNN variant by introducing the concept of stability vector to anticipate the near-optimal learning capacity of the variant. Current practices select arbitrary safe epoch numbers to run the experiments. Traditionally, for early stopping, validation error with train loss is used to identify the generalization gap. However, it is a trial-and-error-based approach, and recent studies suggest that train loss does not correlate well with generalization. We propose a hypothesis that anticipates the near-optimal learning capacity of a CNN variant during the training and thus saves computational time. The proposed hypothesis does not require a validation dataset and does not introduce any trainable parameter to the network. The implementation of the hypothesis can easily be embodied to any existing CNN variant as a plug-and-play. We also provide an ablation study that shows the effectiveness of our hypothesis by saving 58.49% computation time (on average) across six CNN variants and three datasets. We expect to further investigate the data behavior based on different statistical properties for other deep neural networks.

[2]Initial and curved phases are described in appendix section.

[3]We checked with $r=4$ for ResNet18 on the CIFAR100 dataset and VGG16 architecture on the SVHN dataset, and the models do not stop training even after the 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.

David Bonet, Antonio Ortega, Javier Ruiz-Hidalgo, and Sarath Shekkizhar. Channel-wise early stopping without a validation set via nnk polytope interpolation. *arXiv preprint arXiv:2107.12972*, 2021.

Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

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.

David Duvenaud, Dougal Maclaurin, and Ryan Adams. Early stopping as nonparametric variational inference. In *Artificial Intelligence and Statistics*, pp. 1070–1077. PMLR, 2016.

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, 2016.

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. In *55th Hawaii International Conference on System ScienceKarevs (HICSS 2022)*, 2022.

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, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

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.

Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*, 2017.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

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

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJC2SzZCW`.

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.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

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

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.

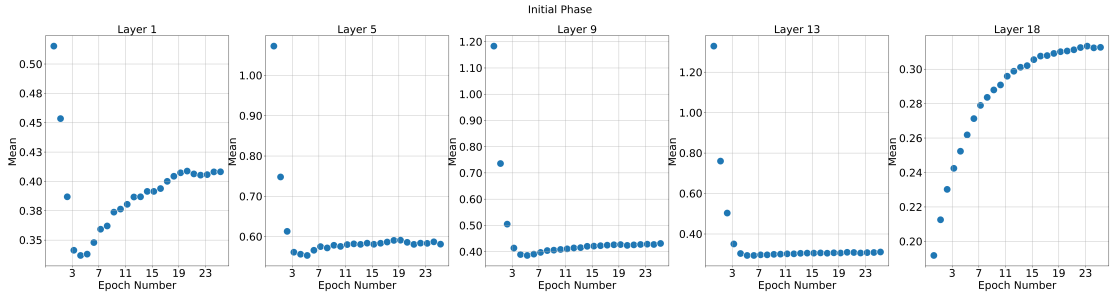
Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. In *NeurIPS*, 2020.

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.

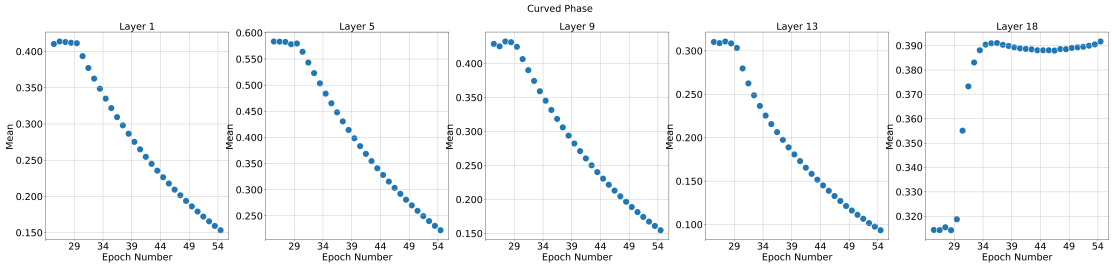
A APPENDIX

Initial phase refers to the early stage of training. For ResNet18 on the CIFAR100 dataset, we consider the approximate range of the initial phase from epoch 1 to epoch 25. In this phase, S_n^e values are unstable across all the layers (top-left one of Figure 6). We also observe a sharp drop or rise of μ_n^e values in most of the layers (Figure 8a).

Curved phase refers to the smooth changes of S_n^e values in the training phase. For ResNet18 on the CIFAR100 dataset, we consider the curved phase's approximate range from epoch 26 to epoch 55. We observe S_n^e values gradually increase or decrease (Figure 6, top-left) in curved phase. Figure 8b also shows that μ_n^e values across all the layers create a smooth-shaped curve.



(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



(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.

Figure 8: Mean of stability values (μ_n^e) at ResNet18 on the CIFAR100 dataset. Figures 8a, 8b, and 7a show μ_n^e values on initial phase, curved phase, and curved phase to stable phase. Figure 7b shows $p^2(\mu_n^e)$ values on stable phase.