Rethinking of Deep Models Parameters with Respect to Data Distribution

Shitala Prasad, Dongyun Lin, Yiqun Li, Dong Sheng, and Oo Zaw Min
Visual Intelligence, Institute for Infocomm Research (I^2R), A^*STAR, Singapore
Email: {shitalap, lin_dongyun, yqlin, dong_sheng, ozmin}@i2r.a-star.edu.sg

***Abstract*—The performance of deep learning models are driven by various parameters but to tune all of them every time, for every given dataset, is a heuristic practice. In this paper, unlike the common practice of decaying the learning rate, we propose a step-wise training strategy where the learning rate and the batch size are tuned based on the dataset size. Here, the given dataset size is progressively increased during the training to boost the network performance without saturating the learning curve, which is seen after certain epochs. We conducted extensive experiments on multiple networks and datasets to validate the proposed training strategy. The experimental results proves our hypothesis that the learning rate, the batch size and the data size are interrelated and can improve the network accuracy if an optimal progressive step-wise training strategy is applied. The proposed strategy also reduces the overall training cost compared to the baseline approach.**

I. Introduction

The revolutionization of deep learning (DL) [1] in field of computer vision (CV) has changed the way the visual learning takes place. Because of which today transfer learning is very much feasible even with small datasets, by simply sharing the useful knowledge from one domain to another. A well-established pre-trained model helps to achieve decent performance by fine-tuning to the target training dataset. The success of AlexNet [1] led to a rapid development of large convolutional neural networks (CNN) to achieve state-of-the-arts in several vision tasks such as image classification [2], [3], object detection [4], [5], [6], image segmentation [7], action recognition [8] and zero-shot learning [9].

In DL-based image classification, huge amount of data are involved like ImageNet [10], which requires manual labeling. This data annotation is a very tedious and costly job. As one of the solution, Yalniz *et al.* [11] proposed a teacher-student model where a strong teacher model is trained to label an unlabeled billion-scale image dataset as a training dataset for student network. Unlike the teacher model, the student model trained on these pseudo labeled data, now has a better weight initialization that helps network to fine-tune on the original labeled data. This provides a significant improvement in the student model performance compared to the teacher. However in a real-world scenario, collecting such a huge dataset and then use it for training is infeasible with a given limited computing resources. In DL, decaying of learning rate and adjusting batch size is a common practice [12], [13], [14], [15]. Touvron *et at.* [16] proposed a simple training-testing strategy to optimize the classifier performance by employing different resolutions to train and test. Such strategy boosts the *top-1* classification accuracy of ImageNet to 86.4% (till now the best, as claimed).

Inspired by the aforementioned ideas, in this paper we first show that adjusting data distribution while training can lead to a significant performance improvement without any additional data contribution. Then, we show that this strategy actually works for various different state-of-the-art image classification architectures such as ResNet [17], ResNeXt [18], Res2Net [19] and Res2NeXt [19]. In general, the proposed training strategy randomly splits the data into small subsets and incrementally train the classifier by updating the dataset with other subsets in a progressive step-wise fashion.

The proposed approach is based on a rigorous experimental analysis of the effect of data distribution to optimize the deep learning models. The experiments are performed on two different types of datasets: interclass and intraclass; and their results can answer questions like Can the learning performance be improved without additional data?, Can accuracy be increased for the same architecture with the same dataset?, Can data distribution strategy boosts the accuracy and reduces the training cost? and Can it reduce network over-fitting?

The main contributions of this paper are summarized as:

- We propose a simple step-wise hyper-parameter tuning strategy to boost the network classification performance without using any additional data.
- We evaluated the proposed strategy to be consistently valued for several state-of-the-art image classification network architectures.
- Compared with the baseline training, the proposed strategy significantly achieves uprise in *top-1* and *top-5* classification accuracies on CIFAR-100 [20], Birdsnap [21], Food-101 [22] and COVID-19_mask-nomask datasets for different networks.
- The proposed step-wise training reduces the overall training cost by $\approx 40\%$.

The rest of the paper is summarized as follows. Section II briefly introduces the related works done in this field. Section III describes the proposed approach followed by Section IV that provides an extensive experimental analysis for the classification task. Finally, Section V concludes the paper.

II. Related Work

In computer vision (CV), one of the core challenging problem is image classification which is used to understand the

natural scene images. Since 2012, starting from AlexNet [1] to GoogleNet [23] in 2014 to SENet [24] and GPipe [25] in 2018, the network keeps on growing their parameters to train for image classification task. Although these models are mainly designed for ImageNet dataset but they are also proved to be the better pre-trained model for variety of transfer learning datasets such as CIFAR-100.

In recent CV research on image classification, the trend is now more shifting towards larger networks and higher resolution such as ResNeXt-101 32x48d [26] and EfficientNet [27]. ResNeXt-101 32x48d is a huge network with roughly 829 million parameters for 224×224 input image when used for image classification task. Whereas, EfficientNet-b7 with only 66 million parameters for 600×600 training image, achieves state-of-the-art in ImageNet classification. In evaluation, it is seen that the *top-5* metric is more robust and tends to saturate with the modern architectures while the *top-1* accuracy is more sensitive in network improvements.

On the other side, efficient models are also explored that can suit devices like smartphones which is becoming ubiquitous. Researchers proposed SqueezeNets, MobileNets, ShuffleNets, Onces-for-all [28] and EfficientNet-B0 as an alternative solution to the above heavy networks with a trade-off between accuracy and efficiency [27]. Yet it is unsure how to design efficient mobile-size CNN for larger models that have much more expensive tuning cost but acquires higher accuracy. Moreover, these CNN models requires more data to acquire better training and here the question still remains unanswered, *i.e.*, how much data is enough to train a deep model.

In this paper, we aim to study different training strategies for efficient learning using the same data that surpass the traditional or baseline training approach in terms of accuracy. We also study the data-accuracy dependency in CNN models and to achieve this, we propose a simple progressive step-wise training strategy to apply on state-of-the-art CNN models.

III. Our Training Strategy

In this section, we describe and derive the data distribution problem in image classification, then study different training approaches and finally propose our new training strategy to optimize the learning.

A. Problem Definition

Let's say $\mathscr{N}$ be a CNN model where it's i^{th} layer is defined as $Y_i = \mathscr{F}_i(X_i)$ such that Y_i is the output, $\mathscr{F}_i$ is the operator and X_i is the input to the i^{th} layer with tensor shape $(\boldsymbol{b}, h_i, w_i, C_i)$. Here, (h_i, w_i) is the spatial dimension, C_i is the channel number and $\boldsymbol{b}$ is the batch size. Then, according to [27], $\mathscr{N}$ can be mathematically represented as:

$$\mathscr{N} = \bigodot_{\substack{j=1,...s \\ i=1,...k}} \mathscr{F}_i^{l_j}(X_{(\boldsymbol{b},h_i,w_i,C_i)}) \tag{1}$$

where $\mathscr{F}_i^{l_j}$ denotes a repeated $\mathscr{F}_i$ operator for l_j times in j^{th} stage with an input tensor X of i^{th} layer. Here, j represents the number of convolutional layers stages used in CNN model, just like ResNet has five stages. Now, if model $\mathscr{N}$ has α parameters then to minimize the loss, $\mathscr{L}(\alpha)$ is defined as:

$$\mathscr{L}(\alpha) = \frac{1}{M} \sum_{\mathbf{m}=1}^{M} \mathscr{L}_{\mathbf{m}}(\alpha) \tag{2}$$

where M is the training set size and $\mathscr{L}_{\mathbf{m}}(\alpha)$ denotes the loss for the $\mathbf{m}^{th}$ training sample.

Currently in CNN, stochastic gradient optimization algorithms are used to update the parameters which is based on the gradient averages over a mini-batch size of the complete training set M. This mini-batches, also known as batch size $\boldsymbol{b}$, is commonly used in network parameter tuning. According to Stochastic Gradient Descent (SGD) optimization, the weight learning can be formated as:

$$\alpha_{m+1} = \alpha_m + \lambda \Delta \alpha_m \tag{3}$$

where

$$\Delta \alpha_m = -\frac{1}{m} \sum_{i=1}^{m} \bigtriangledown_{\alpha} \mathscr{L}_i(\alpha_m) \tag{4}$$

where λ is network learning rate and m is the training example in a batch. In general, according to Smith and Li [29], the scale of fluctuation in SGD is defined as:

$$\mathscr{G} = \lambda(\frac{M}{\boldsymbol{b}} - 1) \tag{5}$$

In [12], author proved empirically that $\boldsymbol{b} \propto \lambda$, *i.e.*, decaying λ can be directly proportional to increasing of $\boldsymbol{b}$.

Now, unlike the regular CNN training that uses loss Eq. 2, if the training set M is split into several small subsets, *i.e.*, $M = \{m_1, m_2, ...m_n\}$ and they are gradually added for training the network $\mathscr{N}$, then the new updated loss function $\mathscr{L}'(\alpha)$ can be defined as:

$$\mathscr{L}'(\alpha) = \frac{1}{\sum_{i=1}^{d} M_i} \sum_{m=1}^{\sum_{i=1}^{d} M_i} \mathscr{L}_m(\alpha) \tag{6}$$

where M_i is the subset of M *s.t.* $M_i = (m_1 + m_2 + ...m_i)$. This approach will not add on the computation overhead but will gradually increase the learning capacity of $\mathscr{N}$. In each training subset implementation, we typically use a weight correction based on the average of their local gradients according to Eq. 3. Therefore, since $\boldsymbol{b} \propto \lambda \mathscr{M}_i$, the noise scale can be approximated as $\mathscr{G}_i \approx \frac{\lambda M_i}{\boldsymbol{b}}$. In this paper, to avoid the data subset division argument, we simply take a random equal splitting to simplify our focus on the proposed hypothesis.

Another important factor to regulate the CNN learning is mini-batch $\boldsymbol{b}$, as discussed in Eq. 1. So using the sum of the gradients, the SGD parameter update rule for α_m can be re-written as:

$$\alpha_{m+1} = \alpha_m - \lambda' \sum_{i=1}^{\boldsymbol{b}} \bigtriangledown_{\alpha} \mathscr{L}_{\boldsymbol{b}}(\alpha_m) \tag{7}$$

where $\lambda' = \frac{\lambda}{\boldsymbol{b}}$. This way, the network weights generated by M_i will be used as an initializer for network with M_{i+1} data.

Our ultimate target is to maximize the model accuracy $\mathscr{A}$ for a given dataset, say $\mathscr{D}$, by optimizing the data size, batch size and learning rate and thus, can be formulated as:

$$\underset{(M,\boldsymbol{b},\lambda)}{maximize} \quad \mathscr{A}(\mathscr{N}(\alpha_{(M,\boldsymbol{b},\lambda)})) \tag{8}$$

B. Training Approaches

To optimize the data usage and implement problem 8, there are some issues which needs to be discussed first.

1) Dataset Size: According to [11], DL-based models are data hunger networks and the more data we provide, the better $\mathscr{N}$ performance is. Similarly, in case of transfer learning, the better the weight initialization is, the better the performance is. Yalniz *et al.* used a semi-supervised billion-scale training set for a student model to train first before it is fine-tuned on the original labeled dataset [11]. This introduces a better initializer and thus acquire a higher accuracy compared to its teacher. Figure 1a shows the testing accuracy on CIFAR-100 dataset. According to this accuracy curve, we observe that the larger M acquires higher accuracy for a given network $\mathscr{N}(\alpha)$ and also the learning process is faster.

2) Batch Size: In general, DL model uses batch size $\boldsymbol{b}$ to control the weights update, as explained previously. By word batch size, we actually mean that a random set of $\boldsymbol{b}$ images from the training dataset, say $\mathscr{D}$, are taken to estimate the error gradient to update weights. Figure 1b shows an accuracy performance comparison on 20% of CIFAR-100 dataset which is again trained from scratch for 20 epochs with a constant λ but varying $\boldsymbol{b}$. Different $\boldsymbol{b}$ achieves different accuracy performance for the same dataset (see Figure 1b). According to Eq. 7, small $\boldsymbol{b}$ results in rapid learning while the large ones leads to a slow learning process but the convergence is more stable, which is not in the prior case [30]. Small $\boldsymbol{b}$ are usually noisy with lower generalization error. Figure 1b validates these observations and encourage to choose a proper $\boldsymbol{b}$ for the best $\mathscr{N}(\alpha)$ output.

3) Learning Rate: In neural networks, weights cannot be calculated by using any analytical method, instead are discovered using SGD optimizer and the amount of change in weight during this search is called the learning rate. Its a configurable hyper-parameter and CNN models can potentially update their weights more optimally [31]. Generally, it is observed that larger λ allows $\mathscr{N}(\alpha)$ to learn faster by arriving on a sub-optimal final set of weights [32]. Whereas, small λ allows $\mathscr{N}(\alpha)$ to learn slowly but a more optimal or globally optimal set of weights. Figure 1c experimentally represents the above concept by keeping $\boldsymbol{b}$ constant. When λ is too large ($\lambda = 0.1$), the weight update is also too large (refer Eq. 7) and thus, the model $\mathscr{N}(\alpha)$ performance oscillates over the training epochs. On the other hand, when λ is too small ($\lambda = 0.0001$), the model never converge or stuck on sub-optimal minima and requires more training epochs. Therefore, its important to get an optimal λ with learning decay to achieve higher $\mathscr{A}$.

C. Training Strategy

Based on the above experiments, we observed that scaling up the dataset with an optimal batch size and learning decay gives better performance. Hence, it is important to obtain an optimal set of $(M,\boldsymbol{b},\lambda)$.

We pragmatically observe that fine-tuning only one of these three hyper-parameters will not give the best solution as they are interrelated and needs to be tuned together. In this paper, we introduce a new step-wise training strategy where with the same computation cost, $\mathscr{N}(\alpha)$ achieves higher performance compared to old strategy. The new learning rate λ' for given dataset size M_i is calculated as

$$\lambda' = \begin{cases} \frac{\lambda}{b} & \text{for } M_i \text{ where } i = 1 \text{ or } n \\ \frac{(\lambda/10)}{n \times b} & \text{for } M_i \text{ where } 1 < i < n \end{cases} \tag{9}$$

where n is the total number of subsets of M dataset used in training process and $\lambda = \lambda'$ for next i. Intuitively, $\boldsymbol{b}$ is a user-defined hyper-parameter which is set according to the available resources while λ regulates the learning speed of $\mathscr{N}(\alpha)$. With gradual increase in M_i, we increase the batch size such that $M_i/\boldsymbol{b}$ is constant. This controls the learning of $\mathscr{N}(\alpha)$ and when $M_i = M$, the model is fine-tuned with $\lambda' = \lambda/b$ to regulate the learning curve. For simplicity, let λ' be λ_i. Therefore, the noise scale can be equated as:

$$\mathscr{G}_i \approx \frac{\lambda' M_i}{\boldsymbol{b}_i} \approx \lambda_i \times M_i \tag{10}$$

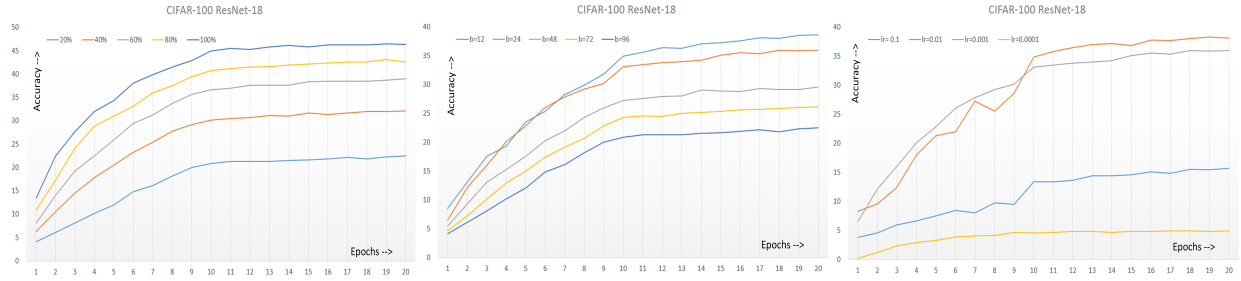
where i represents the progressive update during the training. In the next section, we experimentally proves the proposed hypothesis.

IV. Image Classification Experiments and Analysis

In this section, we firstly describe the dataset and the implementation details used in this paper followed by various sets of experiments and there analysis to uphold our hypothesis.

A. Dataset and Implementation

All experiments in this paper are based on SGD optimization. For training and testing, we used two different types of datasets: interclass CIFAR-100 dataset and intraclass Birdsnap and Food-101 datasets. CIFAR is a natural object dataset of 100 classes with 50k training samples and 10k testing samples. Each class has 500 images for training and 100 for testing. The image resolution is 32 $\times$ 32 which is scaled to 224 $\times$ 224 for our experiments. Birdsnap is the second dataset used in this paper which includes 500 North American bird species collected from Flicker. It has total 49,829 images out of which 2443 images are used for testing. The third dataset is the Food-101 which consists of 101 food categories with total 101k training and testing images. Among 1000 images per food category, 750 are involved in training and 250 is used for testing. This dataset includes 512×512 images with some wrong labels and high color intensity. This adds a different level of challenge to validate the proposed training strategy. In all datasets, the testing set is not disturbed and the original test evaluation set are used. In addition, for further experiments and ablation studies, we introduce a new COVID-19_mask-nomask dataset with 2,509 train images and 539 test images which is



(a) (b) (c)

Fig. 1: Training approaches with different settings: (a) dataset size M_i is $\{20\%, 40\%, 60\%, 80\%, 100\%\}$, (b) batch size $\boldsymbol{b} = \{12, 24, 48, 72, 96\}$ and (c) learning rate $\lambda = \{0.1, 0.01, 0.001, 0.0001\}$. The baseline network used for these graphs is ResNet-18 with an input tensor (3, 224, 224) from CIFAR-100 dataset. Note, all the trainings are from the scratch and are trained for 20 epochs with a milestone at 10. The X-axis and Y-axis are the epochs and the testing *top-1* accuracy, respectively.

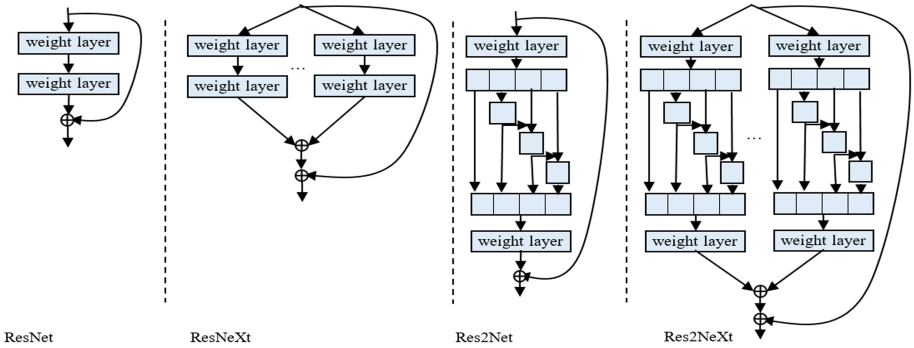


Fig. 2: Architectural network comparison of: ResNet, ResNeXt, Res2Net (*scale=4*) and Res2NeXt (*scale=4*).

a combination of three smaller datasets[1] for masked and non-masked face classification. The purpose of introducing such a small datasets is to test the robustness of proposed hypothesis on dataset size. The sample images are shown in Figure 3.

Fig. 3: COVID-19_mask-nomask dataset. Details: train (1050 mask, 1459 no-mask) and test (264 mask, 375 no-mask) images.

Recently, numerous new backbone networks are introduced which achieves state-of-the-art performance in image classification [27], [19], [17]. These four datasets are investigated on different versions of widely used ResNet architecture, *i.e.*, ResNet-50 and ResNeXt-50. The hypothesis is also tested on different version of recently proposed Res2Net[2] architecture, *i.e.*, Res2Net-50, Res2Net-50_v1b_26w_4s and Res2NeXt-50. Note that the networks chosen here are almost equivalent in terms of depth and performs best in image classification task.

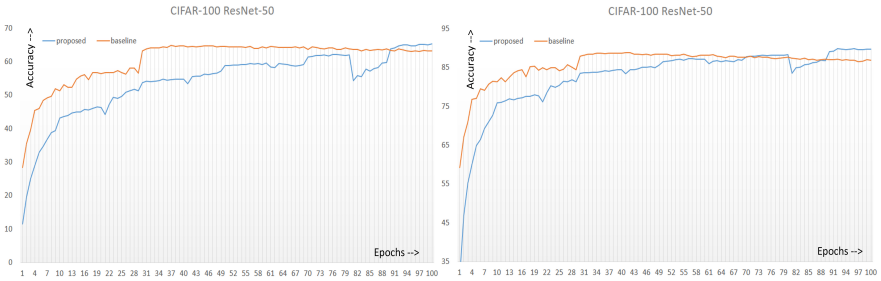
For the implementation of ResNet and Res2Net, we used Pytorch[3]. We used a set of step-wise batch sizes $\beta = \{\boldsymbol{b}_1, ...\boldsymbol{b}_i\}$ and initial learning rate $\lambda = 0.001$ (the best and stable case in Figure 1c). The momentum ϕ is set to 0.9, gamma γ is 0.1, weight decay ω is fixed to $1 \times e^{-4}$. In this paper, all the network models $\mathcal{N}$ are trained from scratch (except Section IV-C) to count the real contribution of the proposed step-wise training strategy. Because if we use the pre-trained weights for initialization, the model learning task will be highly influenced by transfer learning. Also note that there are no data augmentation algorithms involved in any of the experiments shown in this paper.

The comparison matrices used in this paper are *top-1* and *top-5*, as they are the most used matrices by other image classification researchers [27], [11], [19]. The experiments was performed on a Titan X GPU cards with 12GB RAM in Linux environment.

[1] 1. **covid_mask_images:** https://www.kaggle.com/danielferrazcampos/face-mask-images
2. **mask_dataset:** https://www.kaggle.com/ahmetfurkandemr/mask-datasets-v1
3. **COVID19 mask image dataset:** https://github.com/UniversalDataTool/coronavirus-mask-image-dataset

[2] Res2Net source code: https://github.com/Res2Net/Res2Net-PretrainedModels

[3] Pytorch: https://pytorch.org/



(a) (b)

Fig. 4: *top-1* (**left**) and *top-5* (**right**) accuracy comparison used in this paper: a) ResNet-50 *with* the traditional training strategy on set O and b) ResNet-50 *with* the proposed training strategy with A-B-C-D-O sets.

B. Experiments

We first provide meticulous experimental results on CIFAR-100 dataset and then validate the hypothesis on Birdsnap dataset followed by ablation study on Food-101 and COVID-19_mask-nomask datasets. This subsection is segmented into various deep learning models, $\mathscr{N}$ with different number of parameters α. The basic block diagram difference of the networks used in this paper are shown in Figure 2. The network size in terms of parameters are 25.56M, 25.02M, 25.70M and 24.67M respectively for ResNet-50, ResNeXt-50, Res2Net-50 and Res2NeXt-50.

1) ResNet: We exploited the proposed step-wise training strategy on various different networks and achieved significant improvements for all. In ResNet family [17], we adopted ResNet-50 due to its wide popularity and are used by most of the researchers. Table I shows *top-1* and *top-5* accuracy measures for CIFAR-100 dataset using ResNet-50. The baseline experiment, *i.e.*, when ResNet-50 is trained on the complete training set O for 100 epochs (refer Eq. 5) acquires 64.94% *top-1* accuracy and 88.62% *top-5* accuracy. While when it is trained on sets A, B, C, D and O for 20 epochs, the same model with the same number of parameters α, managed to secure 35.79%, 46.73%, 52.52%, 55.29% and 57.28% *top-1* accuracy, respectively without sharing the weights. In the last row, Table I represents the accuracy achieved by the proposed step-wise training strategy where it achieves 66.34% (*top-1*) and 90.72% (*top-5*) accuracy when trained using different batch sizes (batch sizes are mentioned along the arrows) with increase in data size (following Eq. 7). It shows an uprise of 1.4% (*top-1*) and 2.1% (*top-5*) for the same dataset with the less computation cost. A step-wise learning rate decay is used in all training process, refer Eq. 10. For the baseline performance computation, the learning rate decay is used at every 30^{th} epochs while for all other experiments the decay is at every 10^{th} epochs.

Figure 4 shows the accuracy curve in terms of *top-1* and *top-5* accuracy for CIFAR-100 dataset, comparing the baseline and the proposed step-wise training strategy progress. From this graph, we observe that the basic training strategy gets saturated after few epochs, in this case, it's after 30 epochs. While in the proposed strategy, the dataset images are progressively added with different learning rate and batch size which generates an impulse fluctuation to the network's learning capability to further drive the accuracy curve with updated noise scale $\mathscr{G}$ (Figure 4).

TABLE I: ResNet-50 performance analysis on CIFAR-100 dataset. Letters A, B, C, D, and O respectively represents 20%, 40%, 60%, 80% and 100% training sets of CIFAR-100, as discussed in Section III-A. Note: all the trainings are from the scratch and the **bold** is the best results.

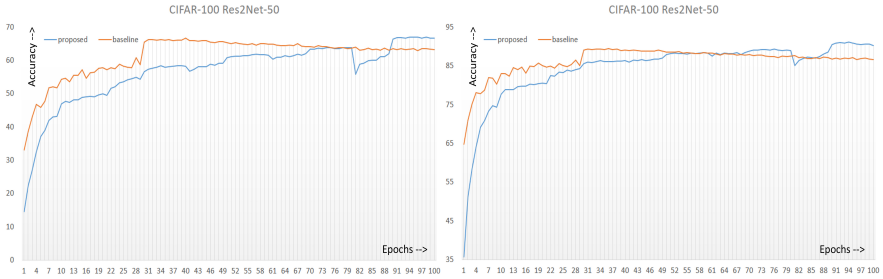
CIFAR-100 Dataset	**ResNet-50**		
	top-1	*top-5*	**Epochs**
with set O (baseline)	64.9400	88.6200	100
with set A	35.7900	66.3300	20
with set B	46.7300	77.4700	20
with set C	52.5200	81.9900	20
with set D	55.2900	84.3500	20
with set O	57.2800	85.8100	20
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	**66.3400**	**90.7200**	20 *each*

Next, the computation cost involved in step-wise training is 40% less[4] than the traditional training approach where complete data is used throughout the training.

2) Res2Net: We perform similar experiments with other networks to validate the proposed hypothesis functioning that is not biased with the network architecture. Recently, Gao *et al.* [19] proposed Res2Net constructed by adding a hierarchical residual-like connections with in a single residual block, as shown in image of Figure 2. Using Res2Net-50, we plot the accuracy graph (Figure 5) and observed that the proposed training strategy continuously boosts the network performance while the baseline curve gets saturated after 31 epochs. The proposed approach achieves 67.04% (*top-1*) and 90.65% (*top-5*) accuracy which is 0.39% (*top-1*) and 1.50% (*top-5*) higher than the baseline training.

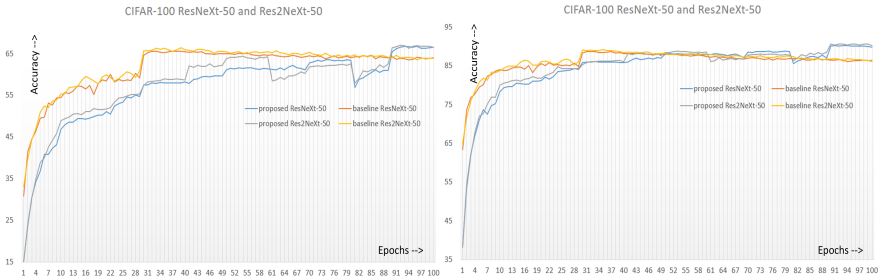
In [19], another version of Res2Net-50 with filter width of 26, scale size of 4 and trainable parameters of 25.72M is also trained and tested. By following the same training strategy, it achieves 66.11% *top-1* accuracy while the baseline saturates at 64.79%.

[4]Training cost (epoch$\times M$): 1. Traditional approach $\approx 100 \times M$
2. Proposal $\approx 20 \times (0.2M + 0.4M + 0.6M + 0.8M + M) \approx 60 \times M$



(a) (b)

Fig. 5: *top-1* (**left**) and *top-5* (**right**) accuracy comparison of: a) Res2Net-50 *with* the traditional training strategy on set O and b) *with* the proposed training strategy, *i.e.*, A-B-C-D-O sets.



(a) (b)

Fig. 6: *top-1* (**left**) and *top-5* (**right**) accuracy comparison of ResNeXt-50 and Res2NeXt-50 *with* O set and *with* the proposed training strategy, *i.e.*, A-B-C-D-O sets.

3) ResNeXt and Res2NeXt: Xie *et al.* [18] and Gao *et al.* [19] proposed a finer version of ResNet and Res2Net, respectively. The difference between ResNeXt and Res2NeXt models are respectively shown in Figure 2b and Figure 2d. These upgraded models are bit lighter than their original versions (Section IV-B). The performance of Res2NeXt-50 *vs* ResNeXt-50 is shown in Figure 6 and also quantified in Table II. ResNeXt-50 and Res2NeXt-50 using the proposed training strategy achieved 1.22% and 0.57% rise in the *top-1* accuracy.

4) Birdsnap: After we tested the proposed training hypothesis on CIFAR-100 dataset, we examined it on a totally different dataset, *i.e.*, Birdsnap dataset in which all the classes belong to a single 'Bird' category. Therefore, the challenge of image classification here is more harder. Following the similar splitting as in CIFAR-100, we used the recent Res2Net-50 and Res2NeXt-50 networks to train and test with 500 different bird species. Compared to the baseline training, the proposed training strategy achieved 0.35% and 1.77% improvement in *top-1* accuracy and 0.47% and 0.98% in *top-5* accuracy for Res2Net-50 and Res2Next-50, respectively. This validate that the hypothesis is also true for the intraclass datasets too.

C. Ablation Study

The proposed step-wise training strategy is further examined for a detailed ablation studies on Food-101 dataset with different combinations of M and $\boldsymbol{b}$, keeping λ constant, see Tables IV-VI. In Table IV, unlike CIFAR-100 and Birdsnap datasets, Food-101 dataset compares the proposed step-wise training strategy with the baseline training strategy using with and without pre-trained weights. In this ablation study, it is observed that the proposed training strategy shows significant improvement in both the cases. The *top-1* accuracy is raised by 0.97% and 1.43% for with and without pre-trained weights, respectively. Table IV also details the intermediate accuracies with and without weight sharing between transit from one data set to another and noticeably found that sharing weight boosts the network learning more efficiently compared to without sharing. The proposed step-wise training strategy achieves 86.71% and 97.26% in *top-1* and *top-5* accuracy, respectively when weights are shared while without weight sharing it's limited to 54.99% and 81.50%. Note that the settings are all kept same in with and without sharing and no data augmentation algorithms are used. Also, the pre-trained weights used in this paper are from ImageNet.

In Table V, similar experiments are conducted by varying the batch size and keeping other settings same as in Table IV. It shows a noteworthy improvement even with different batch sizes when compared to the baseline approach. We also tested the proposed strategy for k-fold by keeping $\boldsymbol{b}$=24 and varying M splits, *i.e.*, splitting the given dataset into three subsets of 20%, 60% and 100%, as shown in Table VI. Compared to Table IV, we perceive that the larger the dataset split the better the performance is. When the dataset splits into three subsets, the *top-1* and *top-5* accuracy reached to 86.37% and

TABLE II: Performance analysis of ResNeXt-50 and Res2NeXt-50 on CIFAR-100 dataset.

CIFAR-100 Dataset	ResNeXt-50		
	top-1	*top-5*	**Epochs**
with set O (baseline)	65.5800	88.6700	100
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	**66.8000**	**90.1900**	20 *each*

CIFAR-100 Dataset	Res2NeXt-50		
	top-1	*top-5*	**Epochs**
with set O (baseline)	66.4100	88.7400	100
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	**66.9800**	**90.5900**	20 *each*

TABLE III: Performance analysis of Res2Net-50 and Res2NeXt-50 on Birdsnap dataset.

Birdsnap Dataset	Res2Net-50		
	top-1	*top-5*	**Epochs**
with set O (baseline)	61.7901	83.3790	100
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	**62.1391**	**85.150**	20 *each*

Birdsnap Dataset	Res2NeXt-50		
	top-1	*top-5*	**Epochs**
with set O (baseline)	61.9900	84.0025	100
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	**62.4611**	**84.9875**	20 *each*

97.13%, respectively. Whereas, when its split into five subsets, the accuracy outstretched to 86.71% *top-1* and 97.26% *top-5*. These experimental analysis proves that if the batch size $\boldsymbol{b}$ successively keeps changing along with the data size M, the gradient optimization is more efficient without letting the learning curve get saturated.

This gives a second thought, what if the dataset is too small? Will this noise scaling be still valid? To exploratory examine the above questions, we further test the hypothesis on COVID-19_mask-nomask dataset where the experiments are split into four possibilities of Eq. 10, *i.e.*, $\mathscr{G} \approx \frac{\lambda_i M_i}{\boldsymbol{b}_i}$, $\mathscr{G} \approx \frac{\lambda_i M_i}{\boldsymbol{b}}$, $\mathscr{G} \approx \frac{\lambda M_i}{\boldsymbol{b}_i}$ and $\mathscr{G} \approx \frac{\lambda M_i}{\boldsymbol{b}}$. Table VII shows the experimental results on various methods and evident that due to $\mathscr{G} \approx \frac{\lambda_i M_i}{\boldsymbol{b}_i}$ the proposed step-wise training strategy performs better. The first two set of experiments where two different $\boldsymbol{b} = \{16, 24\}$ are chosen as initial batch size, validates that the hypothesis work well with different batch size settings too. From the table, it is also clear that if only λ and M are updated according to Eq. 9 and 10, the noise scale is disturbed which results in poor network performance. As from the previous Table VI, it is observed that the higher n results better and so we tested it on COVID-19_mask-nomask dataset too. For k-fold test, we tested Res2Net-50 on set A $\overset{2b}{\rightarrow}$ C $\overset{b}{\rightarrow}$ O and obtained an accuracy of 99.5305%, whereas for A $\overset{2b}{\rightarrow}$ O the accuracy is limited to 98.4351% (detailed in supplementary section).

D. Discussion

We summarize the major observations from these experiments, which are as follows:

- In the proposed step-wise training strategy, similar to [11], the M_i trained network weights are uses as the new initializer for M_{i+1} subset training that boost the learning curve without saturating.
- This study analyzes a close interrelation between M, $\boldsymbol{b}$ and λ and propose a step-wise training to uprise the performance instead of traditional baseline training.
- The proposal proves that the learning curve can still be improved by using an optimal step-wise training without performing any change in the network architecture. It also validates that the higher M, the better $\mathscr{A}$ of $\mathscr{N}(\alpha)$ is.
- The experiments also confirms that the proposed step-wise training reduces the risk of over-fitting by adopting different $\boldsymbol{b}$ and also reduces the training cost by 40%.
- Lastly, the experiments endorse the hypothesis to be true for different architectures as well as different data types.

The experiments conducted in this paper, constructs a re-thinking of data distribution while training the deep models to obtain much better performance with the given dataset.

V. Conclusion

We presented simple yet effective step-wise training strategy to further improve the learning curve with the same given dataset. The strategy proposed is a combination of data size, batch size and learning rate in such a way that it generates impulse to update the weights. This adds a significant improvement to ResNet, Res2Net, ResNeXt and Res2NeXt architectures. The evaluation of the proposed hypothesis is performed on multiple types of datasets: CIFAR-100, Birdsnap, Food-101 and COVID-19_mask-nomask.

Although the proposed training strategy significantly works for image classification tasks, in future we would like to explore other aspects of CV such as object detection and segmentation where annotation is the biggest challenge. There can be another dimension of data distribution for such processes.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[2] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. on EC*, 2019.

[3] W. Zheng and Z. Zhang, "Accelerating the classification of very deep convolutional network by a cascading approach," in *ICPR*. IEEE, 2018, pp. 355–360.

[4] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *arXiv preprint arXiv:1911.09070*, 2019.

[5] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. on NNLS*, vol. 30, no. 11, pp. 3212–3232, 2019.

[6] S. Cao, Y. Liu, C. Zhou, Q. Sun, L. Pongsak, and S. M. Shen, "Thinnet: An efficient convolutional neural network for object detection," in *ICPR*. IEEE, 2018, pp. 836–841.

[7] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, 2019, pp. 82–92.

[8] H. Yang, C. Yuan, B. Li, Y. Du, J. Xing, W. Hu, and S. J. Maybank, "Asymmetric 3d convolutional neural networks for action recognition," *PR*, vol. 85, pp. 1–12, 2019.

[9] R. Gao, X. Hou, J. Qin, J. Chen, L. Liu, F. Zhu, Z. Zhang, and L. Shao, "Zero-vae-gan: Generating unseen features for generalized and transductive zero-shot learning," *IEEE Trans. on IP*, vol. 29, pp. 3665–3680, 2020.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. Ieee, 2009, pp. 248–255.

TABLE IV: Performance analysis of Res2Net-50 on Food-101 dataset. Rest same as previous.

Food-101 Dataset	**Res2Net-50**					
	Pre-trained=False		**Pre-trained=True**		**Epochs**	**Weight Shared**
	top-1	*top-5*	*top-1*	*top-5*		
with set O (baseline)	53.5584	81.0297	85.7347	96.8832	100	✓
with set A	12.0832	32.2733	75.5881	93.2198	20	
with set B	15.7109	38.2733	80.2851	95.1723	20	
with set C	17.0257	40.8119	82.0371	95.8020	20	×
with set D	18.2574	42.7287	83.9247	96.6059	20	
with set O	37.8614	67.5485	86.2574	97.1406	20	
with set A	13.0218	33.4020	75.5881	93.2198		
$\overset{2b}{\rightarrow}$ B	21.8970	48.4634	78.7247	94.3842	20 each	✓
$\overset{3b}{\rightarrow}$ C	30.2851	59.0416	80.6337	95.0416		
$\overset{4b}{\rightarrow}$ D	38.2772	66.9782	81.8614	95.5762		
$\overset{b}{\rightarrow}$ O	**54.9901**	**81.5010**	**86.7089**	**97.2554**		

TABLE V: Performance analysis of Res2Net-50 on Food-101 dataset with varying $\boldsymbol{b}$.

Food-101 Dataset	**Res2Net-50**					
	Pre-trained=True				**Epochs**	**Weight Shared**
	b=12		***b=16***			
	top-1	*top-5*	*top-1*	*top-5*		
with set O (baseline)	84.6931	96.5901	85.2356	96.7050	100	✓
with set A	75.2832	92.9386	76.0119	93.1248		
$\overset{2b}{\rightarrow}$ B	81.5089	95.5129	81.5327	95.2752	20 each	✓
$\overset{3b}{\rightarrow}$ C	83.3307	96.2059	83.7109	9.61069		
$\overset{4b}{\rightarrow}$ D	84.1624	96.4792	84.4554	96.4356		
$\overset{b}{\rightarrow}$ O	**85.9168**	**97.1129**	**86.2653**	**96.9822**		

TABLE VI: Res2Net-50 performance analysis on Food-101 dataset.

Food-101 Dataset	**Res2Net-50**			
	Pre-trained=True		**Epochs**	**Weight Shared**
	top-1	*top-5*		
with set O (baseline)	85.7347	96.2198	60	✓
with set A	75.5881	93.2198		
$\overset{2b}{\rightarrow}$ C	83.5287	96.1505	20 each	✓
$\overset{b}{\rightarrow}$ O	**86.3723**	**97.1327**		

TABLE VII: Res2Net-50 performance analysis on COVID-19_mask-nomask dataset. Training settings as previous.

COVID-19_mask-nomask Dataset	**Res2Net-50**	
	top-1	**Method**
with set O (baseline) (***b=16***)	98.90	$\mathscr{G} \approx \frac{\lambda_i M_i}{b_i}$
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	97.50 $\overset{2b}{\rightarrow}$ 98.75 $\overset{3b}{\rightarrow}$ 99.06 $\overset{4b}{\rightarrow}$ 98.90 $\overset{5b}{\rightarrow}$ 99.67	
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	97.50 $\overset{2b}{\rightarrow}$ 98.75 $\overset{3b}{\rightarrow}$ 99.06 $\overset{4b}{\rightarrow}$ 98.90 $\overset{b}{\rightarrow}$ **99.84**	
with set O (baseline) (***b=24***)	99.53	
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	97.65 $\overset{2b}{\rightarrow}$ 97.18 $\overset{3b}{\rightarrow}$ 97.34 $\overset{4b}{\rightarrow}$ 97.50 $\overset{b}{\rightarrow}$ **100**	
with set O (baseline) (***b=24***)	**99.53**	$\mathscr{G} \approx \frac{\lambda_i M_i}{b}$
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	97.65 $\overset{2b}{\rightarrow}$ 97.65 $\overset{3b}{\rightarrow}$ 97.81 $\overset{4b}{\rightarrow}$ 98.28 $\overset{b}{\rightarrow}$ 97.97	
with set O (baseline) (***b=24***)	99.53	$\mathscr{G} \approx \frac{\lambda M_i}{b_i}$
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	97.65 $\overset{2b}{\rightarrow}$ 97.50 $\overset{3b}{\rightarrow}$ 98.12 $\overset{4b}{\rightarrow}$ 98.75 $\overset{b}{\rightarrow}$ **99.69**	
with set O (baseline) (***b=24***)	99.53	$\mathscr{G} \approx \frac{\lambda M_i}{b}$
with set A $\overset{2b}{\rightarrow}$ B $\overset{3b}{\rightarrow}$ C $\overset{4b}{\rightarrow}$ D $\overset{b}{\rightarrow}$ O	97.66 $\overset{2b}{\rightarrow}$ 98.44 $\overset{3b}{\rightarrow}$ 99.22 $\overset{4b}{\rightarrow}$ 99.37 $\overset{b}{\rightarrow}$ **99.69**	

[11] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv preprint arXiv:1905.00546*, 2019.

[12] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489*, 2017.

[13] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[14] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[15] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[16] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," in *NIPS*, 2019, pp. 8250–8260.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[18] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017, pp. 1492–1500.

[19] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE Trans. on PAMI*, 2020.

[20] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[21] T. Berg, J. Liu, S. Woo Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, "Birdsnap: Large-scale fine-grained visual categorization of birds," in *CVPR*, 2014, pp. 2011–2018.

[22] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *ECCV*, 2014.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018, pp. 7132–7141.

[25] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in *NIPS*, 2019, pp. 103–112.

[26] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *ECCV*, 2018, pp. 181–196.

[27] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[28] H. Cai, C. Gan, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[29] S. L. Smith and Q. V. Le, "A bayesian perspective on generalization and stochastic gradient descent," *arXiv preprint arXiv:1710.06451*, 2017.

[30] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[32] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.