Progressive 3-Layered Block Architecture for Image Classification

Munmi Gogoi, Shahin Ara Begum
Department of Computer Science
Assam University Silchar (A Central University)
Silchar Assam, India

***Abstract*—Convolutional Neural Networks (CNNs) have been used to handle a wide range of computer vision problems, including image classification and object detection. Image classification refers to automatically classifying a huge number of images and various techniques have been developed for accomplishing this goal. The focus of this article is to enhance image classification accuracy implemented on CNN models by using the concept of transfer learning and progressive resizing with split and train strategy. Furthermore, the Parametric Rectified Linear Unit (PReLU) activation function, which generalizes the standard traditional rectified unit, has also been applied on dense layers of the model. PReLU enhances model fitting with almost little significant computational cost and low over-fitting hazard. A "Progressive 3-Layered Block Architecture" model is proposed in this paper which considers the fine-tuning of hyperparameters and optimizers of the Deep network to achieve state-of-the-art accuracy on benchmark datasets with fewer parameters.**

Keywords—CNN; transfer learning; progressive resizing; PReLU; deep network

I. Introduction

Image classification methods using convolution neural networks (CNNs) have recently achieved remarkable success in the field of computer vision, compared to other classic machine learning techniques [1-3]. Many Computer Vision tasks, such as image segmentation and object identification, can be simplified to image classification, thereby enhancing accuracy of classification can have a broad impact across a variety of application domains. The automatic feature extraction capability of the CNN network replaces the conventional feature extraction methods (e.g., SIFT, HOG, GIST), etc., as the deep learning network does not require hand-engineered feature design [4].

We have seen remarkable advancements in the image classification domain in the last several years, owing primarily to advances in two technical directions: creating more sophisticated network architectures and designing efficient techniques to handle overfitting. As neural networks become more complex (e.g., increased depth [1, 2], increased width [5, 6], as well as the utilization of shorter strides [5, 6, 2], new non-linear activations emerge [7-12], and as more complex layer designs emerge [1, 13]), the ability of neural networks to fit training data is improving. On the other hand, effective regularization approaches [12,14-16], active data augmentation [1, 2, 17, 18], and large-scale data [19, 20] lead to greater generalization. Considering the factors which affect the performance of deep models, this paper proposes a "Progressive 3-Layered Block architecture" for image classification by implementing transfer learning and progressive resizing concept. Transfer learning technique not only reduces the problem of network overfitting but also reduces the training time and addresses the issue of insufficient training data in deep models [21]. Progressive Resizing is a technique for resizing all of the images in a sequential manner while training CNN models on lower to larger image sizes, which results in fine-tuning the final model as well as increasing the accuracy score. Furthermore, on the dense layers of the proposed model, we have employed the Parametric Rectified Linear Unit (PReLU) activation function [22], which generalizes the standard rectified unit. PReLU enhances model fitting at a low computational cost with little possibility of overfitting. The developed model is trained under optimized hyper-parameters and experimental evaluation is carried out on benchmark datasets. The proposed model achieves higher accuracy and leads to better performance with fewer parameters as compared to previously developed models.

The rest of the paper is organized as follows: Section II describes the theoretical background of the concepts used in developing the model. Section III presents the related research on image classification using deep learning models, focusing on three aspects *viz*. model development based on parameter efficiency, progressive training and improvement on datasets, optimization strategies and developmental platforms. Section IV deals with the proposed architecture and implementation setup. Section V reports the results obtained on benchmark dataset and Section VI presents concluding remarks.

II. Theoretical Background

The theoretical background used in the proposed "Progressive 3-Layered Block and developmental platforms Architecture for image classification" is described in this section

A. Transfer Learning

Deep learning suffers from the problem of data dependence as it requires a massive amount of training data to understand the latent patterns in the data. Deep learning has a linear relationship between the model and the size of the data set. It is not feasible in deep learning to train an entire Convolutional Neural Network (CNN) from the beginning as it is very challenging to obtain a large enough dataset.

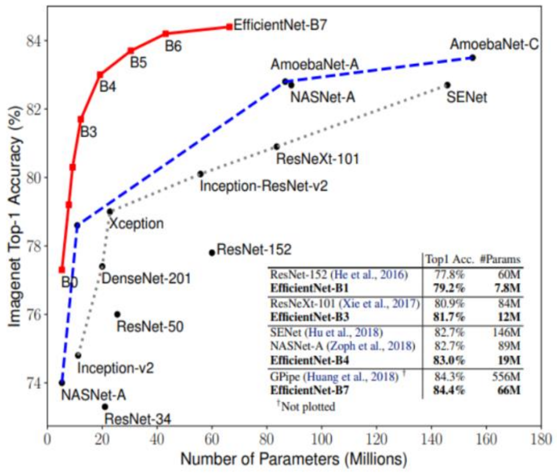
Therefore, it is common to pre-train a CNN on a very large dataset like ImageNet and then reuse the CNN either as a starting point or as a feature extractor for the second target task. This technique has gained huge success in particularly the computer vision field because of its extraordinary setting [46]. Transfer learning addresses the problem of insufficient training data and training time but also it reduces the problem of network overfitting [21]. The proposed work implements the EfficientnetB5 as a pretrained model which has been trained on the popular imageNet dataset.

EfficientNet is a recent CNN architecture developed by Google. EfficientNet sets new records for both accuracy and computational efficiency in image classification and it outperforms the present state of the art. Mingxing Tan and Quoc V. Le of the Google Research Brain team introduced the EfficientNet model in [54]. According to the paper, optimizing the depth, width, and resolution of networks helps to improve classification performance.The family of EfficientNet is scaled up in multiple block layers (from B0 to B7 through compound scaling formula i.e., all three dimensions such as depth, width and resolution are scaled up together to make it more accurate and effective, and there is an optimal balance between all the dimensions. Compared to other previously developed pretrained networks EfficientNet is more effective because it follows the compound scaling formula. Fig. 1 depicts a visualization of the compound scaling method of EfficientNet [54].

A comparison of EfficientNet's performance on the ImageNet dataset with other sophisticated transfer learning models is also documented in the literature. The most recent version of EfficientNet, EfficientNet-B7, has been shown to have the highest accuracy of all with the fewest parameters as depicted in Fig. 1.

B. Progressive Re-Scaling

The notion of progressively re-scaling image datasets has been introduced into Deep Learning Networks to improve accuracy [33]. Super-resolution [47] and GAN training [48] have both leveraged progressive re-scaling approaches.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Fig. 1. Recent Version of EfficientNet and their Accuracy and Parameters with respect to other Networks [54].

The progressive training of image data begins with low-resolution images and incrementally changes the image resolution as training continues. In general Progressive Image resizing is a strategy for resizing the image dataset successively while the CNN models are trained on lower to larger image sizes as shown in Fig. 2.

One way to apply this technique is to train a model on smaller image sizes, such as 128 by 128 pixels, and then use the weights of this model to train another model on larger images, and so forth. Larger models use layers and weights from earlier smaller models in their architecture, which allows them to fine-tune their models and improve their accuracy scores. To the human eye, resizing images from (64 x 64) to (128 x 128) is an insignificant change. However, to CNN models, it provides a whole new dataset to train on.

Image size is vital in improving model accuracy, and several studies have been published in which researchers dynamically modify image sizes throughout training [34]. Three image sizes have been input into the system in the proposed "Progressive 3-Layered block Architecture", while the regularization parameter has been considered to combat over-fitting of the deep model during training (Algorithm 2). The pipeline of progressive resizing of images in all three blocks of images is shown in Fig. 3.

C. Parametric ReLU (PReLU) Approach

The activation function of a neural network can be defined as in equation 1.

$$f(y_i) = \begin{cases} y_i \quad if \ y_i > 0 \\ a_i y_i, if \ y_i \le 0 \end{cases} \tag{1}$$

Where y_i, denotes the input to the nonlinear activation function f on the i^{th} channel, and a_i is a coefficient which governs the negative part's slope. The subscript i in a_i denotes that we enable nonlinear activation to vary across channels. When the value of the coefficient ($a_i = 0$), the activation function is denoted as ReLU; and when a_i is a learnable parameter, the (1) is referred to as Parametric ReLU (PReLU) [13]. A Parametric Rectified Linear Unit, or PReLU can be defined as an activation function that generalizes the traditional rectified unit by adding a slope for negative values as shown in Fig. 4.

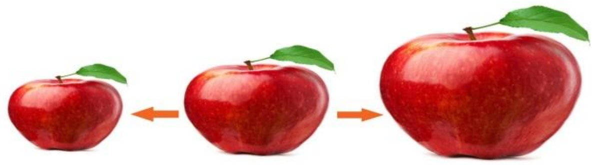


Fig. 2. Progressive resizing of Image.

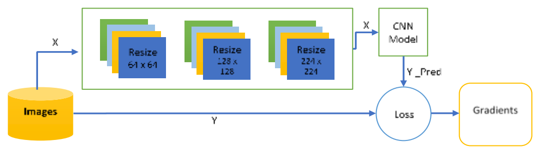


Fig. 3. Progressive resizing Pipeline.

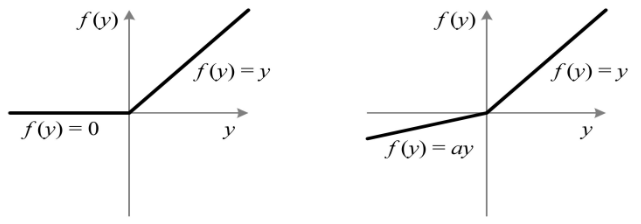


Fig. 4. ReLU Vs PReLU((In PReLU the Coefficient is Not Constant in the Negative Part and is Adaptively Learned).

Different layers may have different forms of nonlinearities. According to the literature, the PReLUs for the initial layers have more positive slopes while investigates with convolutional neural networks (CNNs), i.e., closer to linear. Because the initial layers' filters are Gabor-like filters like edge or texture detectors, this demonstrates a situation in which positive and negative filter responses are respected.

In contrast, the authors find deeper layers have smaller coefficients, suggesting the model becomes more discriminative at later layers (while it wants to retain more information in earlier layers).

III. Related Work

There has been a lot of research work in the literature to improve deep learning models for image classification. In recent years, with the growth of deep learning, in the realm of image classification, various deep architectures, including CNNs, R-CNN, Caps Net, ResNet, etc., have been introduced through which deep-level features can be obtained [23,24]. This section presents the related work on image classification using deep learning into three aspects viz. model development based on parameter efficiency, progressive training and improvement on data-set, optimization techniques and developmental platforms.

A. Model Development based on Parameter Efficiency

In recent years, Convolutional Neural Networks (CNNs) have considerably improved performance on a variety of computer vision applications. [1-3]. Many studies, including DenseNet [25] and EfficientNet [26], concentrate on parameter efficiency, with the goal of achieving higher accuracy with fewer parameters. In the present day, the variations of ResNet such as EffNet-L2 (SAM (Sharpness Aware Minimization)), PyramidNet (SAM) [27] BiT-L (ResNet), BiT-M (ResNet) [28], TResNet-L-V2 [29], etc. with their improvements have gained huge success in the image classification domain over various benchmark datasets.

B. Progressive Training and Improvement on Dataset

Progressive training relies on dynamically changing the setting of the network during training. Some new techniques such as transfer learning [30], adversarial learning [31], and language models [32] have recently gained popularity. Introducing Progressive rescaling of image data in a deep learning network is one of the major factors which can improve accuracy of the model [33]. Image size is a vital component for CNN models accuracy, and several studies have been published in which throughout training, researchers dynamically change the image size to improve model performance. However, accuracy and training speed both are influenced by progressive rescaling of image data and related works such as Mix & Match [34] has been found in the literature on resizing of image data where similar regularization is applied to all image sizes resulting in a decrease in model accuracy. There is another work on regularization where both training speed and accuracy can be improved by adjusting regularization in an adaptive manner [26].

C. Optimization Techniques and Developmental Platforms

Increasing the number of layers in a network raises the network's complexity, necessitating the use of optimization techniques. SGD, Adam [35], AdaGard [36] and AdaDelta [37] are some of the different optimization strategies implemented in the deep architectures such as CNN model along with hyperparameter optimization [38], of the deep network which is vital for better performance as well as network optimization. The advancement of deep learning techniques with GPU processing in combination with a vast dataset enables researchers to solve research issues across different application domains. Many popular frameworks for deep learning applications viz. Tensorflow, Caffe, Torch, Theano, CNTK, and libraries like Pydrive, Cuda, OpenCL, OpenCV, OpenMP, Keras, etc. allow the development of deep learning applications rapidly [39] [40] [41]. Image categorization [42] and object detection [43] are examples of applications where neural architecture search (NAS) has been applied in network architecture optimization. NAS initiatives in the past have primarily focused on increasing the efficiency of FLOPs [44, 45].

This paper aims to improve model accuracy, training speed, and parameter efficiency significantly over the state of the art by taking into account the aforementioned factors.

IV. Proposed Architecture

The proposed “Progressive 3-Layered Block Architecture” model for image classification is presented here. The model has three phases as shown in Fig. 5, the model uses the pre trained architecture of EfficientNet5 as a transfer learning model and constructs the model by adding layers with the concept of compound scaling. The proposed architecture applies the PReLU activation in the dense layers which adaptively learns the parameter from the data. As in (1), the controlling coefficient ai controls the slope of the negative part by adaptively learning the parameters. The total amount of additional parameters introduced by PReLU is negligible while considering the total weights. In the proposed “Progressive 3-Layered Block Architecture”, progressive rescaling technique on images has been implemented to improve the accuracy while considering the regularization parameter along with image size. As an improvement on the progressive learning, the split and train strategy has been implemented which speeds up training along with improved resizing dataset.

The “Progressive 3-Layered Block Architecture” contains three phases:

Phase I: In the phase I, we build a base model for 64X64 image size:

Step 1: Load the pre-trained model

Step 2: Load image dataset for training

Step 3: Set the parameters and add layers implementing compound scaling

Step 4: Set PReLU activation for model layers

Algorithm 1(PReLU activation on model layers)

Step i: Initialize the value of PReLU *(a_i)* for equation 1, where a_i is the controlling coefficient.

Step ii: Update the value of a_i for one layer as in equation 2,

$$\frac{\partial \varepsilon}{\partial a_i} = \sum_{y_i} \frac{\partial \varepsilon}{\partial f(y_i)} \frac{\partial f(y_i)}{\partial a_i} \qquad (2)$$

Where, ε is the objective function and $\frac{\partial \varepsilon}{\partial f(y_i)}$ is the gradient that has been propagated from a deeper layer.

Step iii: The summation $\sum_{y_i}$ runs all the positions of the feature map to update the value of all the layers.

Step iv: Adopt momentum as in equation 3 while updating a_i

$$\Delta a_i = \mu \Delta a_i + \in \frac{\partial \varepsilon}{\partial a_i} \qquad (3)$$

Where μ denotes momentum and $\in$ denotes learning rate.

Step v: Repeat step (1-4) for all the layers.

Step 5: The network takes a set of training images as input, performs feed forward propagation (convolution, PReLU, and pooling operations, as well as forward propagation in the Fully Connected layer), and calculates the output probabilities for each class..

Step 6: Calculate the output layer's total error by, Total Error= ∑ ½ (target probability- output probability) ^{2}

Step 7: The CNN model goes through several convolutions and pooling phases during training and updates the weights with a backpropagation algorithm to minimize the output error.

Finally, save the model weights of the first phase.

Phase II: Build a model for 128x128 image size where the output of the first block is the input of the second block. The progressive resizing on images is performed considering adaptive regularization.

Algorithm 2 (Progressive rescaling on images considering regularization)

```
Step i: Initialize image size $S_0$, set regularization $\{\phi_0^k\}$,
    where $k$ is dropout.
Step ii: Set target image size $S_t$, set regularization $\{\phi_t^k\}$.
Step iii: Total model training has $N$ steps and $P$ stages
    Where, for every stage $1 \le i \le P$,
    Now, for $i=0$ to $P-1$ do
    Size of the input image:
    $S_i \leftarrow S_0 + (S_t - S_0 . \frac{i}{P-1}$
    Regularization parameter:
    $R_i \leftarrow \{\phi_i^k\} = \{\phi_0^k\} + (\{\phi_t^k\} - \{\phi_0^k\}) \cdot \frac{i}{P-1}$
    Train the model for $\frac{N}{P}$ steps with $S_i$ and $R_i$.
    End for
```

Now, train the model by following the steps of **Phase I** and save the model weight of the **Phase II**.

Phase III: The third block loads the weights of block 2 as an input and builds a model for image size 224X224 by repeating all the steps of Phase I and Phase II

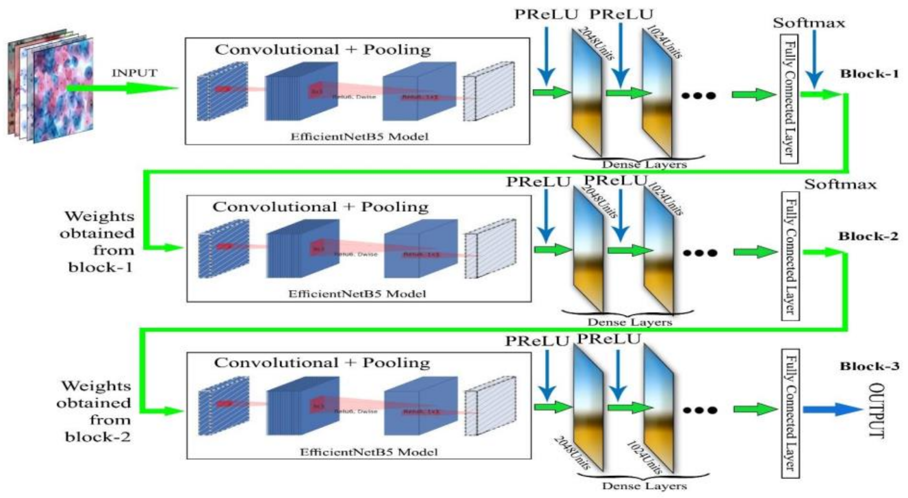


Fig. 5. The "Progressive 3-Layered Block Architecture" for Image Classification.

V. Dataset and Experimental Setup

A. Dataset

To ensure the robustness of the proposed "Progressive 3-Layered Block Architecture", the network is trained and evaluated on three publicly available datasets: CIFAR-10, CIFAR100, and CALTECH 101. The datasets were randomly split into 90:10 (CIFAR-10), (CIFAR100) and 80:10:10 (CALTECH101) proportions for training and validation respectively. The CIFAR-10 dataset comprises 60,000 color images from diverse objects with and categorized into 10 classes (airplane, bird, dog, frog, deer, dog, horse, ship, and truck, automobile) for a total of 6000 images per class [49] Fig. 6. During the training of the proposed method, the datasets were automatically split into 50000 training images and 10000 test images. CIFAR100 is similar to CIFAR10, with the exception that it has 100 classes, each with 600 images. Each class has 500 training images and 100 testing images. The CIFAR-100's 100 categories are divided into 20 super classes. Fei-Fei Li et. al gathered the CALTECH101 dataset in September 2003.

It is made up of images of objects from 101 different classes, as well as one backdrop clutter class. Each class has approximately 40 to 800 images, for a total of approximately 9000 photographs.Images come in a variety of sizes, with common edge lengths ranging from 200 to 300 pixels. The detail of datasets used for the experiment is presented in Table I.

Fig. 6. Data Set of CIFAR10 and CALTECH1.

TABLE I. Details of Datasets for Experiment

Name of the Dataset	Training Images	Validation Images	Classes
CIFAR-10 [49]	50,000	10,000	10
CIFAR-100 [49]	50,000	10,000	100
CALTECH101	60,000	15,000	101

B. Implementation Setup

The network architecture is implemented in Python using Keras [50], a deep learning framework, with TensorFlow [51] as the backend. The experiments for image classification are conducted using model and data-parallelism. The setup includes a GPU environment running the Linux operating system which comprises over 2,000 CPU cores, 1.5TB memory, and GPU accelerators (NVIDIA Tesla V100 32GB) using Google collab pro version. By using the GPU configuration platform, a pretrained network has been implemented on ImageNet and the network has been re-trained on the CIFAR10, CIFAR100, and CALTECH101 dataset using fine-tuning approaches. Moreover, an adaptive learning schedule has been considered where each iteration of the learning process uses 150 epochs with a decreasing learning rate schedule of 5% for every 10 epochs.

VI. Result and Discussion

The experimental setup and outcomes of the proposed network model on benchmark data sets are presented in this section. This section compares and contrasts recent deep model progress with the proposed model of "Progressive 3-Layered Block Architecture" for Image Classification.

Accuracy and error rate is calculated to compare various models [55]. Models attaining the lowest error rate and the highest possible accuracy are usually the most desirable.

The accuracy and the error rate is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Error = \frac{FP + FN}{TP + TN + FP + FN}$$

TN, FN, TP, FP are the number of true negatives and false negatives, true positives and false positives respectively.

A. The PReLU Setup and Comparison Experiment

An improved accuracy of PReLU over ReLU is observed in the "Progressive 3-Layered Block Architecture" for Image Classification over benchmark datasets. The training implementation gained 96.15% accuracy and 96.47% accuracy on CIFAR10 and CIFAR100 using 10 view testing respectively. The model has been trained with ReLUs in all layers by implementing progressive training on different blocks without loading the weight of the previous block which gains average 3% accuracy in all three blocks. Later the model is trained by replacing all ReLUs with PReLUs and also performs progressive training by loading weights from the previous block. Table II details the result obtained where PReLU gains 3.2% accuracy over ReLU in CIFAR10, 3.73% accuracy gain over ReLU in CIFAR100 and 2.67% accuracy gain over ReLU in CALTECH101. The better value for each dataset is denoted in boldface.

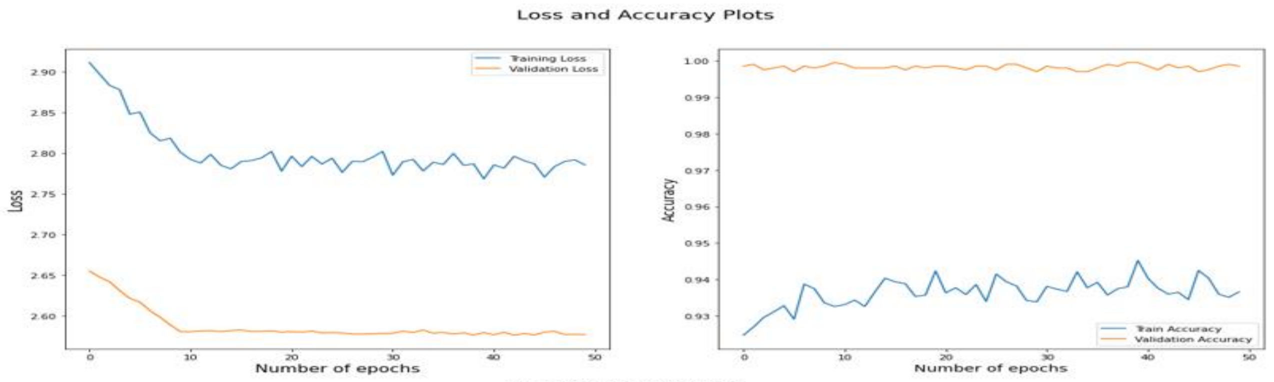
TABLE II. THE PERFORMANCE COMPARISON OF PRELU AND RELU ON THREE BENCHMARK DATASETS

Datasets	**Activation Function**	**Image size(Progressive scaling**	**Performance Metric (Accuracy)**
CIFAR 10	ReLU	64 X 64	79.70
		128 X 128	83.96
		224 X 224	93.66
	PReLU	64 X 64	81.26
		128 X 128	87.26
		224 X 224	96.15
CIFAR 100	ReLU	64 X 64	54.90
		128 X 128	86.69
		224 X 224	92.99
	PReLU	64 X 64	59.78
		128 X 128	88.44
		224 X 224	96.47
CALTECH101	ReLU	64 X64	63.31
		128 X 128	85.91
		224 X 224	93.95
	PReLU	64 X 64	64.88
		128 X 128	86.94
		224 X 224	95.39

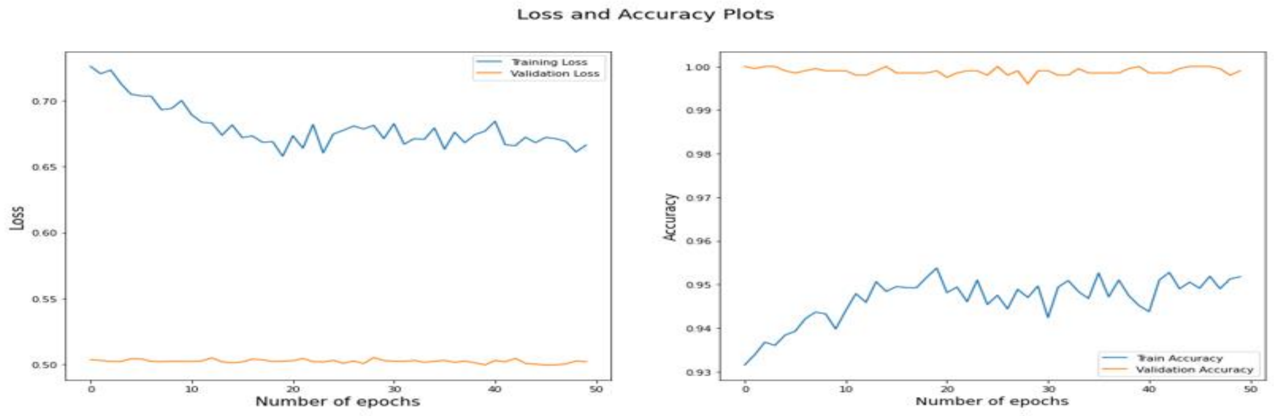
Table II presents the performance comparison of PReLU and ReLU on three benchmark datasets where experimentation is carried out considering the three image sizes *viz.*, 64x64, 128x128, and 224x224. Since implementation of Parametric Rectified Linear Unit (PReLU) activation function on model layers generalizes the classic rectified unit and enhances model fitting with almost no additional computing cost and no risk of overfitting, it shows better performance on all the datasets with rescaled images. In CIFAR10 the model obtained 96.15% accuracy compared to ReLU (93.66%) in 224x224 image data. In CIFAR100 and CALTECH101, the model obtained 96.47% and 95.39% accuracy respectively on PReLU which was higher than ReLU at 92.99% and 93.95 % in both the cases. Fig. 7. further compares the training and validation curve of this approach where ReLU is 93.66% and PReLU is 95.17% on 50 epochs on CIFAR10 dataset. These results suggest that the "Progressive 3-Layered Block Architecture" for Image Classification model generalized well in all three datasets while implementing PReLU with a progressive approach.

B. Progressive Learning Setup and Experiment

The size of the image has a significant impact on the effectiveness of training and accuracy improvement of Deep Neural Networks (DNN). This experimentation considers three parameters while training the network with resized images as presented in Table III.



(a) ReLU on CIFAR10



(b) PReLU on CIFAR10

Fig. 7. Accuracy Plot of PReLU over ReLU on CIFAR 10 (a) Loss and Accuracy Plots of ReLU on CIFAR10 where ReLU is 93.66 (b) Loss and Accuracy Plots of PReLU on CIFAR10 where PReLU is 95.17 on 50 Epoch.

TABLE III. PROGRESSIVE RESCALING SETUP PARAMETERS

Parameters	First training stage		Second training stage		Third training stage	
	Min	Max	Min	Max	Min	Max
Image Size	64	244	64	244	64	244
RandAugm-ent	5	10	5	15	5	20
Dropout rate	0.1	0.3	0.2	0.5	0.1	0.5

In literature, it is found that the accuracy of deep models depends on the regularization parameters of the model while implementing progressive rescaling on image data, it is recommended to adjust the regularization parameters for better accuracy instead of keeping fixed regularizations. In [43] it is mentioned that, to combat overfitting in large models, stronger regularization is required: EfficientNet-B7, for example, employs larger dropout rates and stronger data augmentations than EfficientNet-B0. In the present experimentation, Dropout [15] and RandAugment [52] regularization has been considered with progressive training of images for three training stages for different image sizes.

This experimentation presents the performance over three benchmark datasets with the experimental setup presented as in Table III where for each stage 100 epoch is set.

Through Table IV, we observe that the accuracy is 79.28% in CIFAR10 while the image size is 64X64 with weaker regularization (RandAugment =5, Dropout 0.1) and on the other hand the accuracy is increased to 93.17 after the third training stage with bigger image size and stronger regularization. For CIFAR100 and CALTECH101 datasets, we observe that the accuracy increases as well to 93.99 and 92.72% respectively with a larger image size and stronger regularization parameters. The progressive rescaling of the dataset improves the model's accuracy as well as its training time.

From the experiments conducted it is observed that, in a high-accuracy regime, scaling up data size is more effective than merely scaling up a model size, where it can be concluded that, for instance, when the accuracy of CIFAR100 is beyond 96.47, it is quite challenging to further increase its accuracy by simply increasing model size and so in other image datasets used in this experiment. Fig. 8, further illustrates the training and validation accuracy of CIFAR10 and CIFAR100 on progressive rescaling setting where we observed that the model obtained 96.15% accuracy on CIFAR10 and 96.47% accuracy on CIAFR100 after certain set of training on each stage while implementing progressive rescaling setting.

TABLE IV. PROGRESSIVE LEARNING FOR DIFFERENT IMAGE SIZES AND THEIR ACCURACY WHILE CONSIDERING REGULARIZATION PARAMETERS IN CIFAR10, CIFAR100 AND CALTECH101

Dataset	Image size	RandAug-ment	Dropout rate	Accuracy
CIFAR10	64	5	0.1	79.28
	224	20	0.5	93.17
CIFAR100	64	5	0.1	57.89
	224	20	0.5	93.99
CALTECH 101	64	5	0.1	62.19
	224	20	0.5	92.72

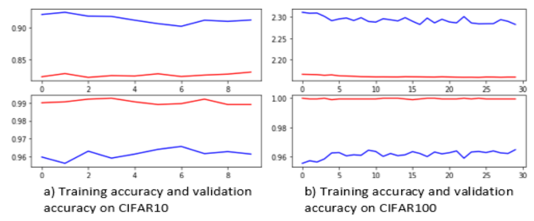


Fig. 8. Plotting the Training and Validation Accuracy of CIFAR10 and CIFAR100.

C. Performance Comparison

This section presents the comparative results on ImageNet, and the performance comparison of the proposed "Progressive 3-Layered Block Architecture" with other models on CIFAR-10, CIFAR-100, and CALTECH 101.

The proposed model uses the pre-trained transfer learning model Efficientnet-B5 trained on ImageNet, ILSVRC2012 and fine-tuned with the same ImageNet setting as in [53]. A smaller batch size of 512, and small initial learning rate of 0.001 with cosine decay has been used. Table V demonstrates the performance comparison of the proposed "Progressive 3-Layered Block architecture" with other models where the top-1 Accuracy is 86.52 %. 2.49%, 2.52 % and 3.51% improvement compared to EfficientNet (2019), ResNet-RS (2021), and DeiT/ViT (2021) respectively. At the same time, the proposed model has fewer FLOPs and fewer parameters compared to other models, which implies that scaling up image size seems more effective than increasing model size by adding layers over the considered datasets.

Table VI presents the performance comparison of the proposed model with other models on all three datasets.

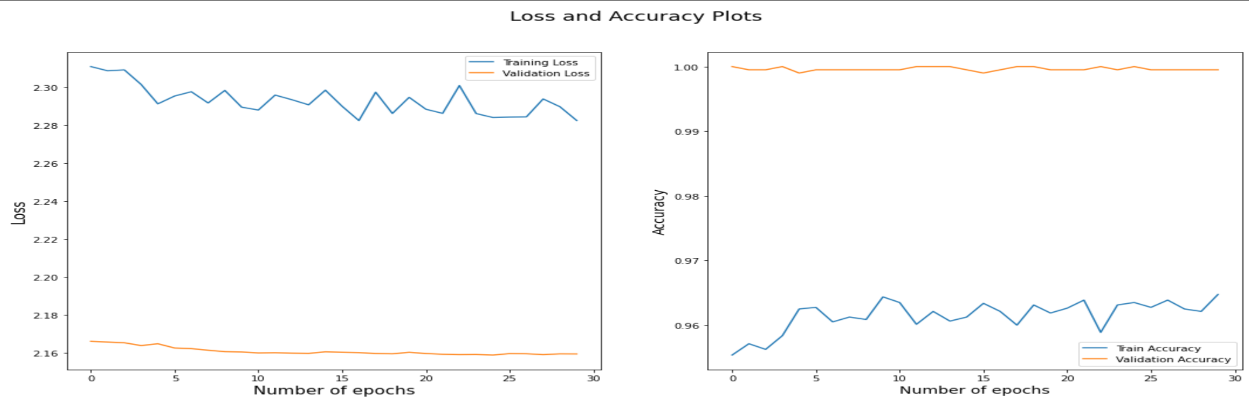
The "Progressive 3-Layard Block architecture" model for image classification combines the progressive rescaling setting with transfer learning technique and PReLU activation where better optimizers and fine-tuning of hyperparameters improve the model accuracy. The model is trained for 65 hrs.' 70 hrs.', and 71 hrs.' respectively for all three datasets which are comparatively less than other benchmark models in the literature. The accuracy on CIFAR10 is 98.79%, CIFAR100 is 96.47% and CALTECH101 is 95.39% for 224X224 images while loading the previous weight of images during training and follows the split and training strategy while training. Fig. 9. further compares the loss and accuracy of CIFAR100 and CALTECH101 where the accuracy is 96.47 % and 95.39% respectively on 224x224 images where the proposed model outperforms the accuracy of CIFAR100 and CALTECH101 with other benchmark models.

TABLE V. PERFORMANCE COMPARISON AND RESULTS OF THE PROPOSED MODEL IN TERMS OF ACCURACY, PARAMETERS, AND FLOPS ON IMAGENET

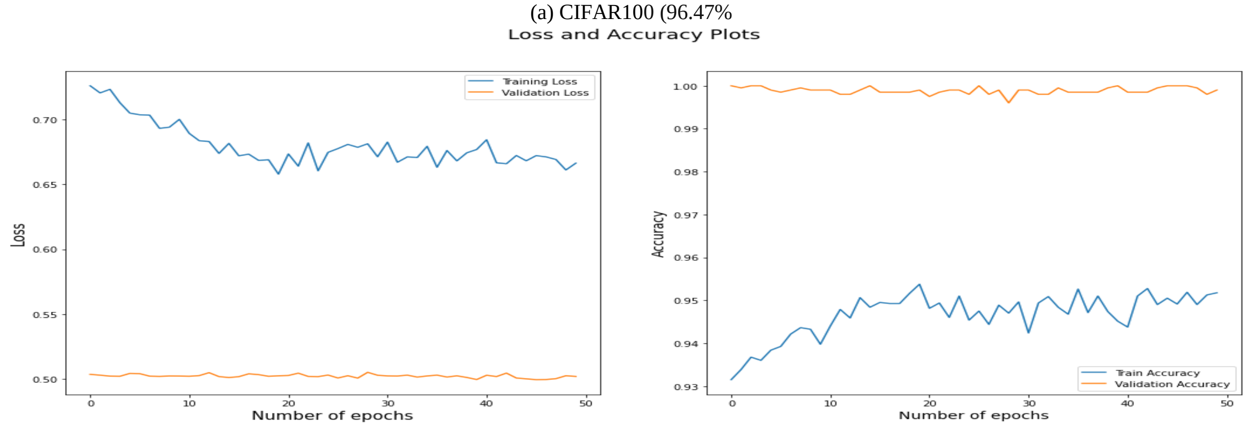
Name of the Network	EfficientNet 2019	ResNet-RS 2021	DeiT/ViT (2021)	Progressive 3-Layered Block Architecture
Top 1 Accuracy	84.3%	84.0 %	83.1%	86.52%
Parameters	43 M	164 M	86 M	36 M
FLOPs	19 B	64 B	56 B	20 B

TABLE VI. PERFORMANCE COMPARISON AND RESULTS ON CIFAR10, CIFARR100 AND CALTECH101

Datasets	Network Models	Accuracy	PARAM	Flops	Training time(Hours)
CIFAR 10	AlexNet (Doan Cong Danh 2019)	89.67%	27.31M	---	144
	BiT-L(ResNet)Kolesnikov, A et al., 2019	98.91	928	---	----
	EfficientNet-B7 **(**Tan & Le 2019)	98.9	64 M	38B	139
	EffNet-L2(SAM)	99.70	0.575114	--	
	Progressive 3-Layered Block Architecture	98.79	36 M	20 B	65
CIFAR100	AlexNet(KP) Akrout, M et.,al 2019	66.78	61 M	-----	146
	BiT-L(ResNet)Kolesnikov, A et al., 2019	93.51		------	
	EfficientNet-B7 **(**Tan & Le 2019)	91.7	64M	38B	138
	EffNet-L2(SAM) Foret, P et al., 2020	96.08	56 M		
	Progressive 3-Layered Block Architecture	96.47	36 M	20 B	70
CALTECH101	Alexnet	-----		------	
	ResNet34	95	60.5 M	----	
	EfficientNet-B7 **(**Tan & Le 2019)	93.0	64 M	38 B	139
	EffNet-L2 (SAM) Foret, P et al., 2020	94.32	56 M		
	Progressive 3-Layered Block Architecture	95.39	36 M	20 B	71



(a) CIFAR100 (96.47%



(b) CALTECH101 (95.39)

Fig. 9. Plotting the Loss and Accuracy Plot of CIFAR100 and CALTECH101 on 224x224 Images.

VII. Conclusion and Future Work

Accuracy improvement in image classification has a wide ranging impact on various application domains and deep convolution neural networks (CNNs) have gained remarkable success in this area. This paper proposed a "Progressive 3-Layered Block Architecture" for image classification by implementing transfer learning and progressive resizing concept. The proposed architecture applies the PReLU activation in the dense layers which adaptively learns the parameters from the data. The efficiency of the proposed architecture is established by its superior performance, low execution time and fewer parameters compared to other models in literature. We evaluate the model in tensorflow environment using three benchmark datasets viz.CIFAR10, CIFAR100 and CALTECH101. The evaluation of proposed "Progressive 3- Layered Block Architecture" is carried out in three different experimental setup viz.PReLU setup, progressive rescaling setup and performance comparison with other CNN models. At the same time, our proposed model has fewer FLOPs and fewer parameters compared to other models. From the results obtained it may be concluded that enhancement in learning strategies, adopting standard activation functions, fine-tuning of hyper-parameters and optimizers and scaling up image size is more effective than increasing model size by adding layers in high accuracy regime. With increased performance, less execution time, and fewer parameters, the suggested architecture outperforms the competition in all three benchmark datasets. However, more testing on multiple datasets with different image sizes and training samples is required for general applicability.

References

[1] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, .., and A (2015) Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

[2] Simonyan, K., Zisserman, A. et al. (2014) Very deep convolutional networks for large-scale image recognition.

[3] Girshick, R. (2015) Fast r-cnn. Proceedings of the IEEE international conference on computer vision, pp. 1440–1448.

[4] Liu, X., Zhang, R., Meng, Z., Hong, R., and Liu, G. (2019) On fusing the latent deep CNN feature for image classification. World Wide Web, 22 (2), 423–436.

[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2014.

[6] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.

[7] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, pages 807–814, 2010.

[8] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In ICML, 2013.

[9] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton. On rectified linear units for speech processing. In ICASSP, 2013.

[10] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv:1312.4400, 2013.

[11] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber. Compete to compute. In NIPS, pages 2310– 2318, 2013.

[12] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. arXiv:1302.4389, 2013.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv:1406.4729v2, 2014.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, pages 1929–1958, 2014.

[16] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In ICML, pages 1058–1066, 2013.

[17] A. G. Howard. Some improvements on deep convolutional neural network based image classification. arXiv:1312.5402, 2013.

[18] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[21] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018, October). A survey on deep transfer learning. In International conference on artificial neural networks (pp. 270-279). Springer, Cham.

[22] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

[23] Wang, Y. and Wang, Z. (2019) A survey of recent work on fine-grained image classification techniques. Journal of Visual Communication and Image Representation,59, 210–214.

[24] Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2017) ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60 (6), 84–90 .

[25] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. CVPR, 2017.

[26] Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. ICML, 2019a.

[27] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020), Sharpness-Aware Minimization for Efficiently Improving Generalization.

[28] Kolesnikov, A., Zhai, X., and Beyer, L. (2019) Revisiting self-supervised visual representation learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1920–1929.

[29] Ridnik, T., Lawen, H., Noy, A., Baruch, E.B., Sharir, G., and Friedman, I. (2021) Tresnet: High performance gpu-dedicated architecture. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1400–1409.

[30] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improve quality, stability, and variation. ICLR, 2018.

[31] Yu, H., Liu, A., Liu, X., Li, G., Luo, P., Cheng, R., Yang, J., and Zhang, C. Pda: Progressive data augmentation for general robustness of deep neural networks. arXiv preprint arXiv:1909.04839, 2019.

[32] Press, O., Smith, N. A., and Lewis, M. Shortformer: Better language modeling using shorter inputs. arXiv preprint arXiv:2012.15832, 2021.

[33] Howard, J. Training imagenet in 3 hours for 25 minutes https://www.fast.ai/2018/04/30/dawnbench-fastai/, 2018.

[34] Hoffer, E., Weinstein, B., Hubara, I., Ben-Nun, T., Hoefler, T., and Soudry, D. Mix & match: training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. arXiv preprint arXiv:1908.08986, 2019.

[35] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[36] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of Machine Learning Research 12.Jul (2011): 2121-2159.

[37] Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

[38] Dong, X., Tan, M., Yu, A. W., Peng, D., Gabrys, B., and Le, Q. V. Autohas: Efficient hyperparameter and architecture search. arXiv preprint arXiv:2006.03656, 2020.

[39] Bahrampour, S., Ramakrishnan, N., Schott, L., Shah, M. et al. (2015) Comparative study of deep learning software frameworks. Comparative study of deep learning software frameworks.

[40] Akiba, T. (2018), Performance of distributed deep learning using ChainerMN.

[41] Karmanov, I., Salvaris, M., Fierro, M., and Dean, D. (2018) Comparing deep learning frameworks: a Rosetta stone approach. Machine Learning Blog.

[42] Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. CVPR, 2018.

[43] Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. CVPR, 2020.

[44] Tan, M. and Le, Q. V. Mixconv: Mixed depthwise convolutional kernels. BMVC, 2019b.

[45] Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. CVPR, 2019.

[46] Huh, M., Agrawal, P., & Efros, A. A. (2016). What makes ImageNet good for transfer learning? arXiv preprint arXiv:1608.08614.

[47] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017.

[48] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.

[49] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[50] Keras (2018), Keras-high-level neural networks API. URL https://keras.io/.

[51] Tensorflow (2018). URL https://www.tensorflow.org/.Accessed20.

[52] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. ECCV, 2020.

[53] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR, 2021.

[54] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

[55] T. Pan-Ning, M. Steinbach, and V. Kumar, “Classification: Basic Concepts, Decision Trees, and Model Evaluation,” Introduction to data mining 1 (2006): 145-205.