Improving CNN Model Generalization with Progressive Resizing and Randomized Progressive Deblurring

Dylan Khor

December 2023

1 Abstract

As Computer Vision and Image Recognition tasks become more demanding this days to keep up with this digitalized world, CNNs become more and more popular as a result of it. However, image data gets larger and larger (e.g., ImageNet), CNN keeps evolving to become larger and larger as well in terms on the number of parameters. With the increase in parameters to be trained, accuracy and speed now becomes a concern. With more classes to classify in ImageNet, model generalization becomes harder as well. Here, I propose a possible training algorithm for improving both the accuracy and generalization of general CNN models by coupling the adaptive progressive resizing method proposed in [1] and randomized progressive deblurring. Progressive deblurring is a method that starts training on images with a high blurring intensity, and as the training proceeds, the images to be trained on starts getting less blurrier until they reach their original resolution. In a sense, the images to be trained on "deblurs" as training proceeds. On the other hand, adaptive progressive resizing is shown to be able to speed up training while simultaneously improving model accuracy in general. By randomly blurring the batch of images during each step, variation are introduced in the training set, which can make the model more robust to changing images and generalize better with less degree of overfitting. By adding the random blurring mechanism to adaptive progressive resizing and without too much loss in accuracy, I was able to improve model generalization with an average generalization gap of 4.37% as opposed to the average generalization gap of 5.82% with just adaptive progressive resizing.

Code is available in: ***ProgressiveResizing_With_Deblurring.ipynb***

2 Keywords

Convolutional Neural Networks (CNN), Progressive Resizing, Progressive Deblurring, Gaussian Blur

3 Introduction and Background

Improving the training speed and model generalization of CNN's has been a hot research topic for the past few years, especially since the release of ImageNet, which is a large image database with over 12 million images, 1000 classes, and a average image resolution of 469×387 pixels. The increase in size of image and the amount of images to be trained on leads to the huge increase in parameters and training time, which makes it unrealistic to be trained for real-time object detection in the computer vision field. Several advancements have been made and lots of efforts have been put into improving the performance of CNNs. For instance, the YOLO9000 CNN model proposed in [2] was able to predict detections for more than 9000 different object categories with high accuracy while still running in real time, which is a huge contribution to the image recognition and computer vision field as it made real time object

detection possible and efficient. However, more improvements can still be made, such as the problem of overfitting and generalization during training phase of a CNN.

Random Resizing & Progressive Resizing

There are multiple factors that contributed to YOLO9000's efficiency and accuracy. One of the factors is a technique called random resizing, which is first introduced in [3] as Mix&Match, in which the model trains on different image sizes in every step. Traditional training of CNNs involved using fixed-size images. However, since CNNs are usually evaluated using images with multiple sizes, [4] showed that the traditional training approach of CNNs results in a drop in accuracy for image sizes below or greater a certain size threshold as shown in figure 1. By training the CNN model with images of random sizes at each step, the Mix&Match method of training CNNs showed an improved robustness to changing image sizes as well as an improved generalization.

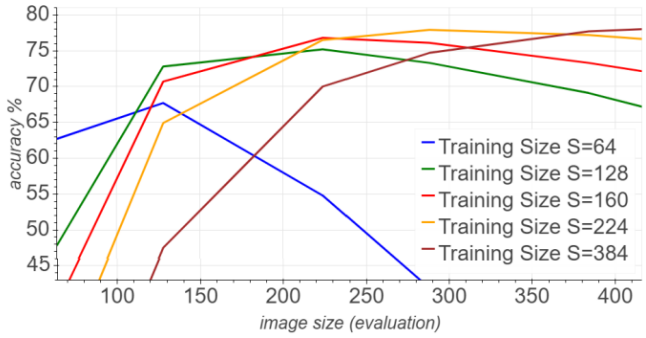

Figure 1: Test accuracy per test image size (Source: [3])

Another technique for improving performance of CNNs is called progressive resizing, which is introduced in [5]. The motivation is similar to Mix&Match, however, instead of randomly choosing size of image to trained on in each step, progressive resizing starts by training on smaller images then gradually increases the image size to train on. The idea is to slowly increase the size of the image being trained on while training until the image size is the same as the original image size in the dataset. In other words, this training procedure pretrains the CNN model on lower resolution images (less pixels), and fine tunes the model on the original image that has a higher resolution, as shown in figure 2. This method reduces the amount of computational cost needed to train the model at early stages while learning the coarse-grained features of the original image on the reduced-version (lower resolution due to lost of pixels during resizing), whereas in later stages, when the image size to be trained on grows, the model can learn the fine-grained features of the original image. Similar to Mix&Match, this approach improves generalization of CNNs and training time.

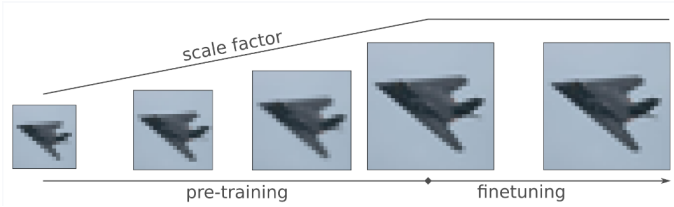

Figure 2: Progressive Resizing Workflow (Source: [6])

Both Mix&Match and progressive resizing showed that image size plays an important part in training efficiency,

however in some cases, dynamically changing image sizes during training can lead to a drop in model accuracy due to overfitting. Tan & Le [1] argued that this is due to the same regularization strength applied to both small and large images during training, and proposed a new adaptive progressive resizing method that adaptively changes the regularization strength according to the image size being trained on, where smaller images require weaker regularization, vice versa. This approach is shown to be able to achieve better model accuracy than Mix&Match and the original progressive resizing.

Gaussian Blur [7]

Gaussian blur, also known as Gaussian smoothing, is an image processing technique used to blur an image using a Gaussian function in order to reduce detail and resolution of the image. It is also a widely used technique in computer vision algorithms in order to enhance image structures at different scales.

Applying Gaussian blur to a 2D image is the same as convolving the image with a 2D Gaussian function, which is shown in figure 3.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 3: 2D Gaussian Function (Source: [7])

The σ in the function is tuned to adjust the intensity of blurring. The higher the value of σ, the blurrier the transformed image, vice versa. Figure 4 shows the effect of applying Gaussian blur on CIFAR-10 images with different σ values.

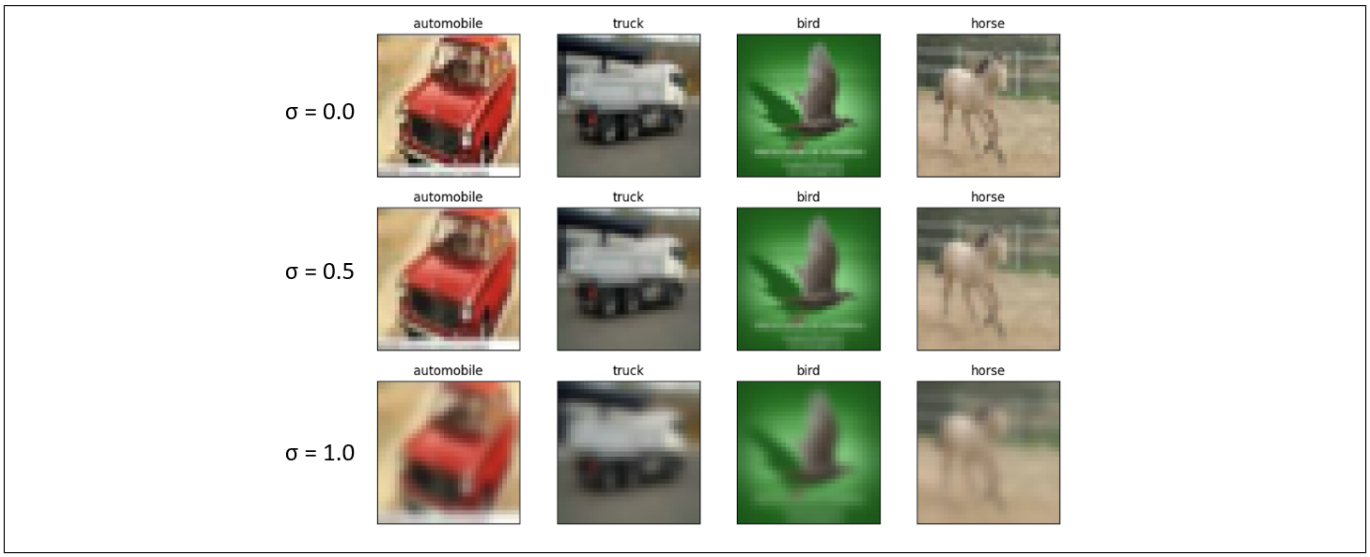

Figure 4: Effect of Gaussian Blur with different σ (Source: [7])

My contribution is as such:

- I propose an improved method of adaptive progressive resizing training approach, which, with a probability of 40%, applies Gaussian blurring on the batch of images with a σ value that scales with image size. I show

that this approach is able to reduce the generalization gap, which improves model generalization and degree of overfitting, without too much loss in accuracy.

4 Methodology

Theory and Approach

To further enhance CNN's ability to generalize, in other words, its ability to analyze and understand patterns in any dataset, we need to constrict the generalization gap between the training and validation accuracy during the learning phase of a CNN. By doing so, we can effectively reduce the model's susceptibility to overfitting, such that the model is able to adapt well to any unseen data.

To do so, I propose adding more variation to the training set by some form of data augmentation while incorporating the adaptive progressive resizing method to reduce training time and increase model accuracy. The introduced variability in the training data should be able to reduce generalization error by learning the features in different representations of the same images.

To match with how adaptive progressive resizing works, I decided to use Gaussian blurring as the other form of data augmentation since both resizing and blurring have one thing in common: lowering the resolution of images. The idea of progressive resizing is to first train on lower resolution images and gradually increase the image size to be trained on until the image reaches its original resolution or size. By applying Gaussian blurring with higher values of σ in early training stages, and progressively decrease the value of σ until it reaches 0 (no blurring), we can achieve the same concept as progressive resizing. By combining both resizing and blurring transformations on the training set images, we can achieve a transformed image with a much lower resolution and sharpness. However, if both resizing and blurring are to be applied to all images in the training set, there will be a lot of information lost since resizing reduces the number of pixels in the image to be learned whereas blurring distorts the features of the image, which can make learning inefficient. Therefore, instead of just resizing all images and blindly blurring all of them, I proposed to apply Gaussian blurring on a batch of images based on a probability threshold $0 < \alpha < 1$, such that not all batches of images in the training dataset will be blurred. Doing so will not only retain most information from the original image, but also introduce more variation in the training set since the training set is now a mix of batches with resized images and batches with resized and blurred images, which helps improve model generalization.

Proposed training approach

My proposed training approach, illustrated in Algorithm 1, builds upon the adaptive progressive resizing method proposed in [1], with the addition of the random application of Gaussian Blurring to each batch of images based on a blurring intensity σ. In early training epochs, this approach trains the model with smaller images that are randomly blurred with a stronger blurring intensity and weak regularization. Then, the image size to be trained on is gradually increased followed by a stronger regularization to make learning more difficult. In addition, the blurring intensity σ also progressively decrease down to 0 as training proceeds in order to simulate the scenario where the model first train on images with blurry edges to first learn the general outlines and structure, then proceeds to training on images with a sharper edge and vision to learn the more finer features, as in the case in progressive resizing.

Algorithm 1 Adaptive Progressive Resizing with Randomized Progressive Deblurring (built upon [1])

Require: Initial image size S_0, regularization R_0, and blurring intensity σ_0
Require: Final image size S_e and regularization R_e
Require: Blurring probability α
Require: Number of epochs N to train and number of batches M in dataset
  for $i = 0$ to $N - 1$ **do**
    Image Size $S_i \leftarrow S_0 + (S_e - S_0) \cdot \frac{i}{N-1}$
    Regularization $R_i \leftarrow R_0 + (R_e - R_0) \cdot \frac{i}{N-1}$
    Blurring Intensity $\sigma_i \leftarrow \sigma_0 - \sigma_0 \cdot \frac{i}{N-1}$
    for $j = 0$ to $M - 1$ **do**
      Generate a random real number β between 0 and 1
      if $\beta \leq \alpha$ **then**
        Train model with image batch j that with S_i, R_i, and σ_i
      else
        Train model with image batch j that with S_i and R_i
      end if
    end for
  end for

For simplicity, the types of regularization that are implemented are:

- **L2 Regularization**, which is fixed throughout the entire training.
- **Dropout**, in which the dropout rate γ increases as training progresses for stronger regularization on larger images.

Network Structure

Layer	Type	Filters	Filter Size	Filter Stride	Filter Padding
1	Convolutional	16	3 x 3	1	1
2	MaxPool	-	2 x 2	2	-
3	Convolutional	32	3 x 3	1	1
4	Convolutional	16	1 x 1	1	0
5	Convolutional	32	3 x 3	1	1
6	MaxPool	-	2 x 2	2	-
7	Convolutional	64	3 x 3	1	1
8	Convolutional	32	1 x 1	1	0
9	Convolutional	64	3 x 3	1	1
10	Convolutional	10	1 x 1	1	0
11	Dropout	-	-	-	-
12	AvgPool	-	Global	Global	-
13	Softmax	-	-	-	-

Figure 5: Network Structure for Classifying CIFAR-10

To test the performance of adaptive progressive resizing and my proposed training approach on CIFAR-10 dataset, I used the network structure shown in figure 5. This simple model is inspired by the Darknet-19 model in [2] that uses the concept in VGG models [8] and Network in Networks (NiN) [9], which uses global average pooling to make predictions as well as 1×1 filters to compress feature representations between 3×3 convolutions. In my case, every convolutional layers are followed with a batch normalization and ReLU operation except for the convolutional layers using 1×1 filters, which are only followed by a ReLU operation. Since the model only contains convolutional and pooling layers, the model can learn the weights of the filters using any size images as an input, which allowed the model to accommodate the resized images during progressive resizing.

5 Results

This section presents my experimental setups, and the results of training the implemented model on CIFAR-10 using three training approaches:

1. **Normal training procedure**, where the image size and regularization strength are kept constant throughout the whole training.
2. **Adaptive progressive resizing**, where image size progressively increase back up to their original size and regularization progressively become stronger as image size grows during training.
3. **My proposed training procedure**, which is built upon the adaptive progressive resizing method. In addition to progressively up-scaling the image size and strengthening the regularization, a blurring factor σ is introduced to blur the training images and is progressively reduced during training until it reaches 0. The probability of blurring α I used in my implementation is 0.4, which means that 40% of the batches in the dataset will be blurred using the current σ.

Experimental Setup

Dataset: In this report, I will only evaluate the performance of each training approaches on CIFAR-10 [10] dataset, which is an image dataset containing 60,000 $32 \times 32 \times 3$ color images over 10 different classes, with each class having 6,000 images. The reason I chose this dataset to work with is because the image dimensions are small, which allowed for a faster training time to see how each training approach compares. Ideally, I want to test the performances on ImageNet21k, which has a larger image dimension, more classes (> 20000 classes), and more data (about 14 million images), to better understand how the proposed training approach compared to previous works. However, due to time constraints, evaluating the methods on CIFAR-10 seems more realistic. The statistics of the dataset are as follows:

- **Training set**: 40000 images
- **Validation set**: 10000 images
- **Testing set**: 10000 images

For this experiment, I will be using three CIFAR-10 datasets (training, validation, and testing), each using a batch size of 128. They will all contain the same data, the only difference is the order of batches in which the model will train on. I will train the model using all three training approaches on the three dataset, and the performance of each approach will be the average over the three datasets.

Setup: The training settings used are: SGD optimizer with L2 norm regularization of 0.001, and 0.9 momentum; initial learning rate of 0.01 with a decay factor of 0.9 every 2 epochs; cross entropy loss as loss function. I trained the model for 15 epochs using each approach on each dataset. For both adaptive progressive resizing and my proposed training approach, the model starts by training on $24 \times 24 \times 3$ images with dropout rate $\gamma = 0.2$, then the image size increases by 2 and dropout rate γ increases by 0.05 after every 2 epochs where the image size is capped at its original size (32) and γ is capped at 0.4, which means that the model will:

- Train on $24 \times 24 \times 3$ images on epochs 1 and 2 with $\gamma = 0.2$
- Train on $26 \times 26 \times 3$ images on epochs 3 and 4 with $\gamma = 0.25$
- Train on $28 \times 28 \times 3$ images on epochs 5 and 6 with $\gamma = 0.3$
- Train on $30 \times 30 \times 3$ images on epochs 7 and 8 with $\gamma = 0.35$
- Fine tune on $32 \times 32 \times 3$ images on epochs $9 - 15$ with $\gamma = 0.4$

On the other hand, in addition to the parameters defined above, my proposed training approach also has two additional hyperparameters: blurring intensity σ and probability of blurring α. The model starts with a blurring intensity $\sigma = 2$ and decreases by 0.5 after every 2 epochs until it reaches 0 where it no longer decreases, whereas α stays fixed at 0.4, meaning a 40% chance the current batch of images will be blurred with the current σ. Lastly, for the normal training approach, the model is trained on fixed $32 \times 32 \times 3$ images with a dropout rate γ of 0.4.

Performance

Figure 6 shows the training and validation accuracy curve for each training approach on the three datasets over the 15 training epochs, where red curves represents the normal training approach, green curves represents the adaptive progressive resizing approach, blue curves represents my proposed training approach, solid lines represents training accuracy, and dashed lines represents validation accuracy.

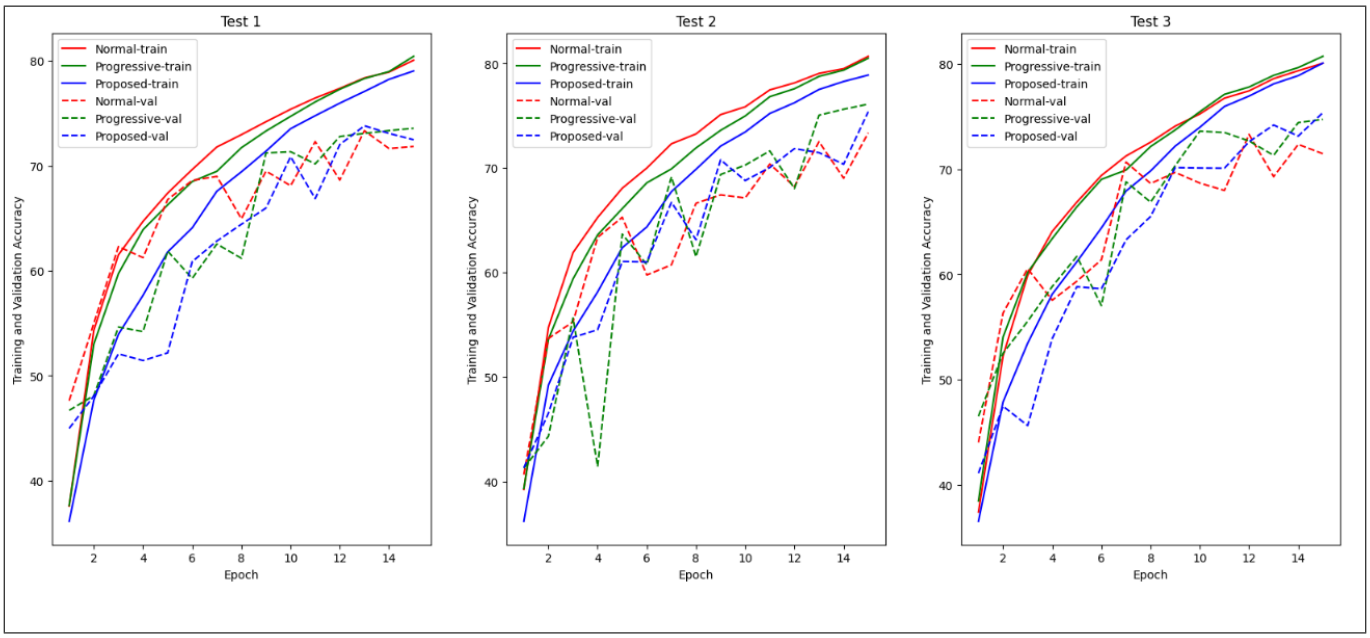

Figure 6: Training Accuracy vs Validation Accuracy of each Training Approach

As shown in figure 6, we can see that the proposed training approach has less degree of overfitting during training as the generalization gap (difference between training and validation accuracy) remains close to each other as training proceeds, whereas the other two approaches tend to have a small generalization gap at the start but the gap tends to fluctuate or increase as training proceeds, in other words, overfitting starts to occur. This means that my proposed training approach is less prone to overfitting and much more stable as the prediction curve (validation curve) does not change much when the training data in modified slightly (order of image batches are changed in this case).

Here, figure 7 shows how the testing accuracy of each approach evolves over each epoch of training. As we can see, the training accuracy of my proposed approach doesn't fluctuate down as intense as the other two approaches, which shows the stability of training using this approach. Even though, the final testing accuracy isn't as high as the one using just adaptive progressive resizing, the drop in accuracy isn't too much, and most importantly, it still has a higher accuracy than the normal training approach.

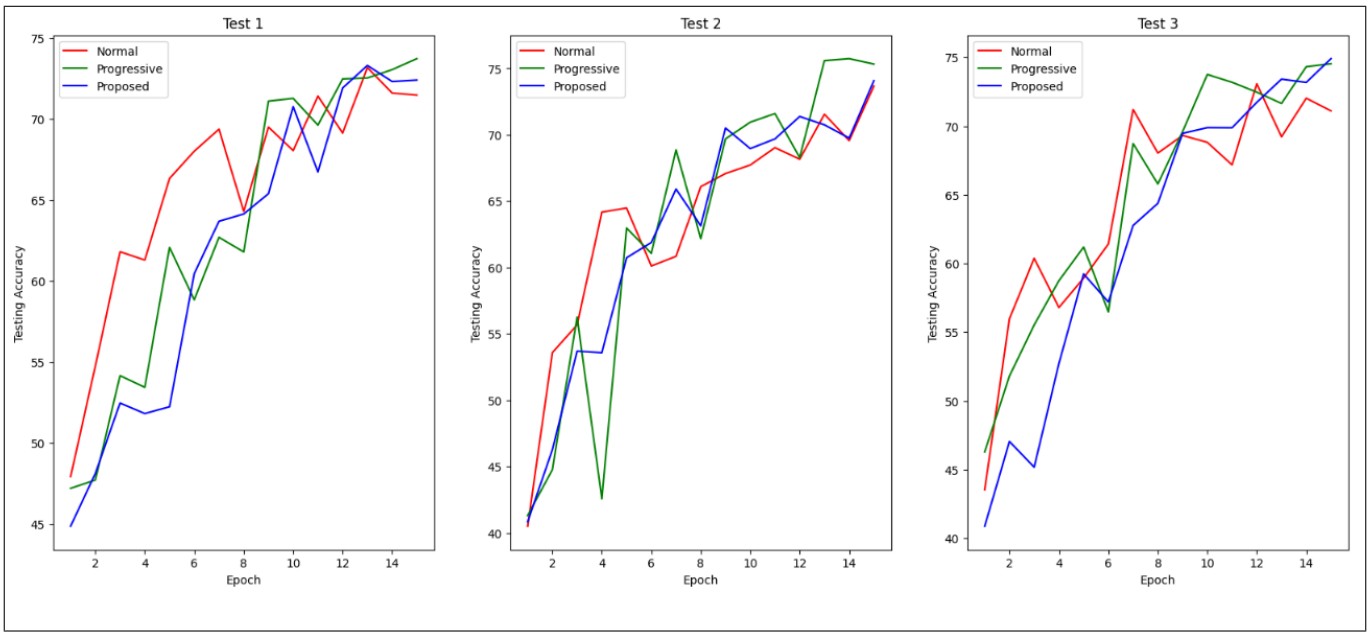

Figure 7: Testing Accuracy of each Training Approach

Figure 8 summarizes the averaged statistics obtained for each training approach across all three datasets. My proposed training approach is able to achieve an average generalization gap of 4.37%, which is almost a 1.5% improvement from the generalization gap of adaptive progressive resizing and the normal training method. There is a drop in 0.7% accuracy as compared to the adaptive progressive resizing approach, but it is still able to be 1.7% more accurate than the model that uses the normal training procedure. Even though the adaptive progressive resizing approach has been showed to reduce training time because of the training on lower dimension images, my experiment showed that it actually takes longer to train (6 seconds longer) than the normal standard approach. However, these results are obtained by utilizing GPU. When I switch over to using CPU, the training time obviously takes way longer, but the training time for both adaptive progressive resizing and my proposed approach is indeed shorter than the normal approach by about 200 seconds at least. Therefore, it still remains a mystery to me as for why the resizing approach took a longer training time than the normal training approach on GPU but not on CPU.

Approach	Average Generalization Gap	Average Test Accuracy	Average Time Taken (s)
Normal	5.76%	72.08%	190.74
Progressive	5.82%	74.53%	196.43
Proposed	4.37%	73.79%	200.41

Figure 8: Average Statistics of Each Training Approach

6 Conclusion

In this paper. I present a new training approach, which is built upon the adaptive progressive resizing approach in [1], that randomly blurs the training images with a progressively decreasing blurring intensity in addition to just progressively resizing the training images back to their original resolution. My proposed approach is able to achieve a 1.45% better generalization gap than the adaptive progressive resizing approach, meaning it is less susceptible to overfitting. However, the downside is that there is a 0.74% drop in accuracy with a slightly longer training time. As a result, my proposed training approach is an improvement of the adaptive progressive resizing approach in terms of generalization without too much loss of model accuracy and is much suited for training a model with high number of epochs as it has better stability.

If I have more time in the future, I will defintely try to implement a larger model (or just use the already implemented state-of-the-art CNN models, such as AlexNet or ResNet) to train on a much bigger image dataset with larger image dimensions, for instance, ImageNet, to further verify the performance of my proposed training procedure. This is because resizing typically works best when the original image dimensions are large and the resolution is good. In my experiment, I used CIFAR-10 as a test dataset, and resizing an already low resolution image to a lower dimension doesn't really help much since they are both still low quality images. In addition, I will also try to train with more epochs to see how the training and validation curves evolve to further grasp the abilities of my proposed approach as compared to previous works.

References

[1] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[2] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

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

[4] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," *Advances in neural information processing systems*, vol. 32, 2019.

[5] J. Howard, "Training imagenet in 3 hours for usd 25; and cifar10 for usd 0.26," *Training imagenet in 3 hours for usd 25; and cifar10 for usd*, 2018.

[6] MosaicML, "Progressive image resizing," Available at https://docs.mosaicml.com/projects/composer/en/latest/method_cards/progressive_resizing.html#technical-details (2023/12/09).

[7] Wikipedia, "Gaussian blur," Available at https://en.wikipedia.org/wiki/Gaussian_blur (2023/12/10).

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

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