Model Accuracy Exploration with CIFAR-10

CSE455 Final Project

Jack Chuang, Warren Shen, Thomas Lin, John Cheng
University of Washington
March 2022

Abstract

This project contains the training implementation with CIFAR-10 based on EfficientNet and Cutout. The goal is to participate in the CIFAR-10 competition hosted by Kaggle to accomplish a new integrational model to improve the performance of the training process. Source code is at https://github.com/Jack-Chuang/UW-CSE-455/tree/main/final%20project.

1. Introduction

The CIFAR-10 dataset consists of 60000 images in 10 classes, and the training target is to differentiate each image into classes without overlapping or misplaced results. There are eight or more notable results since 2015 with an average accuracy of 96%. We aim to integrate the EfficientNet with augmentation to reinforce the accuracy which exceeds the result of our previous coursework.
In addition, since the current models often require support of high-performance hardware and take time to train and debug, another goal of the project is to build a flexible model that is fast and runnable on personal devices.

2. Related Resources

2.1 Previous Work

8 notable proposals on the ML Compiled website [1] are surveyed during the early stage. Two papers were aimed for integration: EfficientNet [2] and Cutout [3]. The source code from GitHub was used for testing, and later a new training model is built from our own. Libraries from Tensorflow and Keras are implemented in the model.

2.2 Dataset

CIFAR-10 from official website [4].

3. Approach

3.1. Scaling

Unlike conventional practice, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. For example, we can simply increase the network depth by α with power of ϕ, width by β with power of ϕ , and image size by γ with power of ϕ, where α,β,γ are constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way.

3.2 Our Own Model

In our cases, we design an experiment to see if the scaling method works. We started from the last homework, tried to do the image recognition based on cifar-10 using only one fully connected layer. While applying the scaling method, as we didn't use the same baseline of the EfficientNet, we didn't use the exact same value of α,β,γ suggested by the paper. Instead, we increase α and β by a constant rate, just to validate whether the performance would improve as the model was enlarged.
The first model is only based on the fully connected layer like what we did in our last homework. The baseline included one dense layer with 16 filters. As we enlarge the scale of the baseline, we would add one more layer and 16 more filters respectively from version 1 to version 6. The baseline of which is as the following:

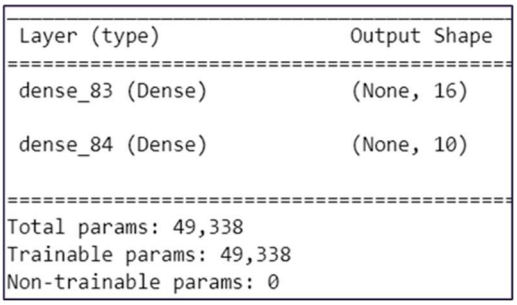


Figure 1. Baseline using only fully connected layer

The second model is based on the set of layers, including a convolutional layer and a max pooling layer, which is taught later in the class. As we enlarge the scale of the baseline, we would add one more set of the layers mentioned above and 16 more nodes respectively from version 1 to version 6.

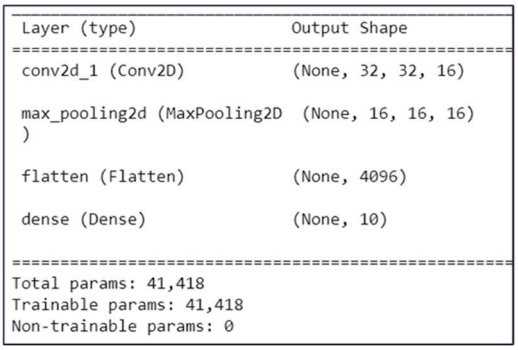


Figure 2. Baseline using convolutional layer

4. Training Results

Finally, we plot the graph with accuracy with respect to the number of tunable parameters as we enlarge the scale of the baseline. We found that as the paper suggested, the performance will improve as the scaling is larger. Yet, the accuracy seems to have an upper limit for about 50 percent using only fully connected layers, and 75 percent using the convolutional layer and max pooling layer. The fact is that the accuracy is still not good enough as shown in the paper, so we move our experiment forward with focus on finding an efficient baseline.

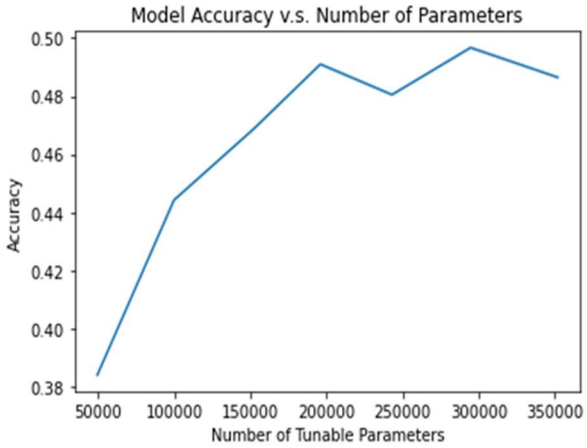


Figure 3. Accuracy to parameters used of Baseline using fully connected layers

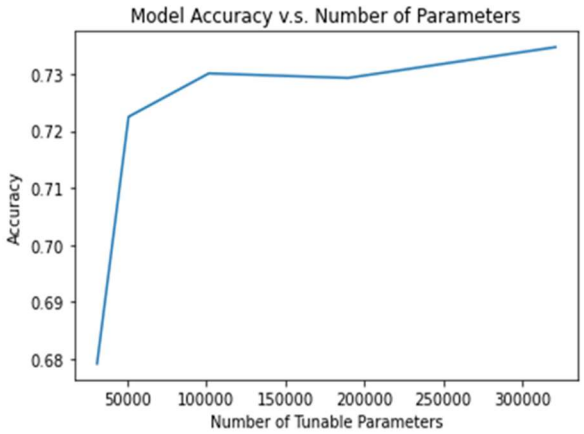


Figure 4. Accuracy to parameters used of Baseline using convolutional layers

5. Experiments for Baseline Reinforcement

5.1 Baseline Network (Jack)

To further increase our model accuracy on the CIFAR-10 data set, we have selected three network models as the candidates of our model baseline. The first one is MnasNet[5], which uses a good amount of depthwise convolution neural network as a part of its network architecture. It uses a reinforcement learning algorithm called PPO to search for its best hyperparameters. Due to our limited knowledge on PPO, we implemented another way of hyperparameter tuning — Bayesian Optimization. We define the search space on our own and try the best parameters and number of layers provided by this method as our MnasNet baseline. The accuracy of this new model on the CIFAR-10 dataset reaches 80%.

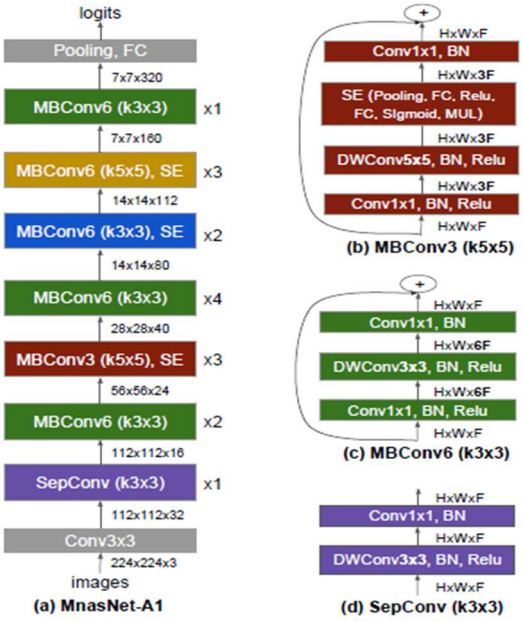


Figure 5. MnasNet architecture

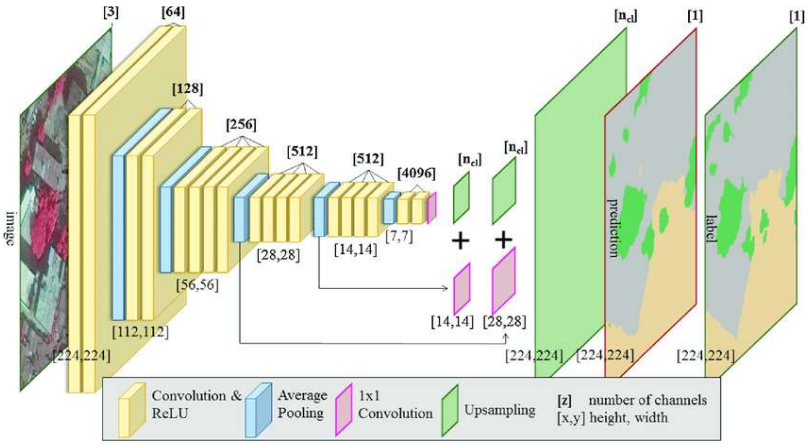


Figure 6. VGG19 architecture

We found that the MnasNet model that we build performs significantly better than our naive implementation but still not good enough. We suspect that the reason the accuracy does not reach our expectation is that we are not familiar with the math behind Bayesian Optimization and we do not have enough knowledge and experience to come up with a set of hyperparameters that will improve the accuracy of our model. Therefore, we stick to the idea of scaling up our model by simply using 2D convolution layers and fully connected layers. We stumbled upon VGG19[6] while walking through the lecture notes and found that VGG19 is a great fit for this idea. We've tried the model that we built from scratch using the concept introduced in VGG19, and the performance has another significant rise to 84%.

We are still not satisfied with the result and decide to look into the problem of VGG19. We found that our VGG19 model converges as early as the first ten epochs. We suspect that it might be the result of gradient vanishing. The most common solution for this problem is to add an identity mapping into the network, and the most famous existing network of this kind is ResNet. We then decided to try ResNet on the CIFAR-10 dataset. There are several versions of ResNet and we choose ResNet34 as our candidate simply because it is the simplest one. Note that we are trying to find the baseline so the simpler the better. Just as we expected, the accuracy of this model beats the two previous models and rose to 87%.

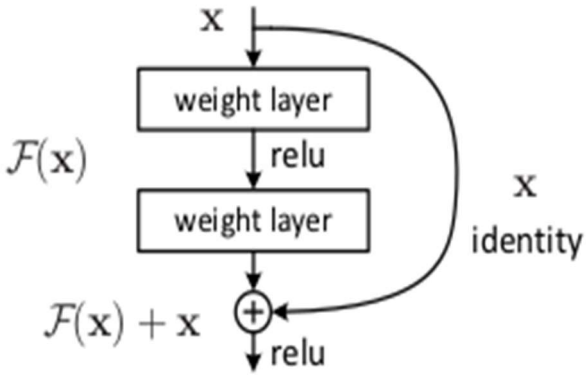


Figure 7. ResNet identity mapping

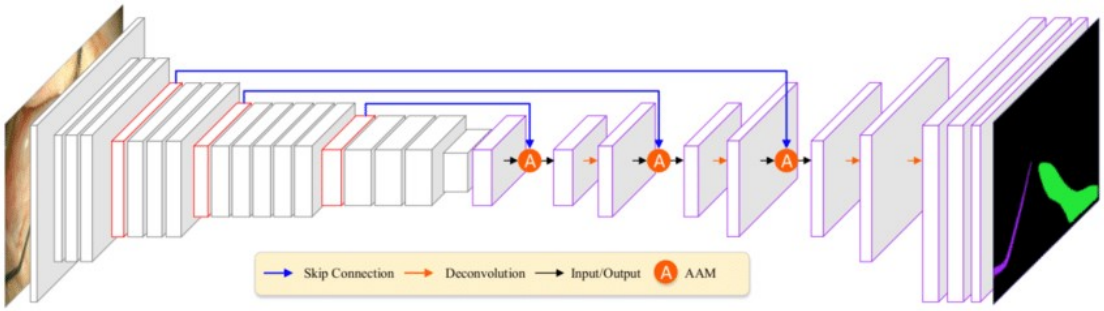


Figure 8. ResNet architecture

The performance and the scale of the three baseline we've tried is listed below:

	MnasNet	VGG19	ResNet34
Accuracy	80%	84%	87%
Total Number of parameters	21,530,851	20,051,530	41,782,666

Figure 9. Accuracy and scale of the three model we've tried

5.2 Augmentation

We have several ways to improve the model with regularization methods. There are data augmentation, dropout and cutout. These methods are used on top of the hyperparameter tuning and the scaling of the model layers.

Affine transformation, being one of the most common techniques in the data augmentation methods, is widely used to expand the original dataset, it could also be used if there is a large imbalance in the different amounts of the classes. There are translation, euclidean, projection, etc.

Dropout is another useful technique when it comes to regularization, it is implemented by setting hidden unit activations to zero with some probability during training. And appears to be very effective at regularizing fully-connected layers. Nevertheless, it is shown to be less powerful when used with convolutional layers. There are two reasons, first is that the convolutional layers have fewer connections and thus fewer parameters compared to the fully-connected layers. Secondly, the neighboring pixels already share similar characteristics, so that there is not much effect if we zero out the output.

Finally, there is the cutout method, which is also what we are focusing on in this project. Cutout is a regularization method for convolutional neural networks, it removes contiguous sections of input images, which augments the dataset with new, occluded variants.

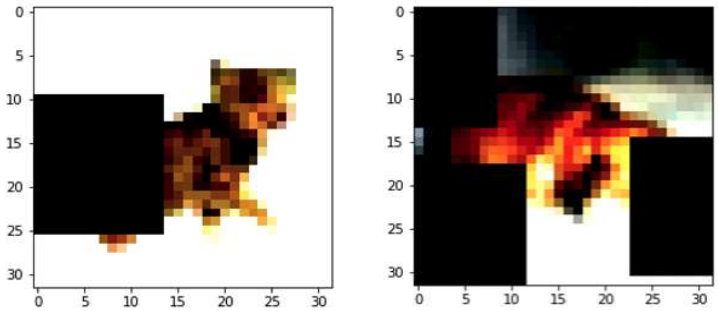


Figure 10-11. Dog class with cutout

For our cutout input, we have a parameter called hole number, for every hole, we generate an arbitrary size of the hole to be cutout. As shown in figure 8, we have samples of the dog class with one and three cutout holes as our parameter. A sample of the whole dataset with cutout applied is shown in figure 9, all having a hole of one.

Here is the result referenced from the paper[7] shown in figure 10, we see a 2 to 4 percent of improvement margin in accuracy when trained with CIFAR-10, CIFAR-100, SVHN, and STL-10 , using three models which are Deep residual network d = 18(ResNet18).

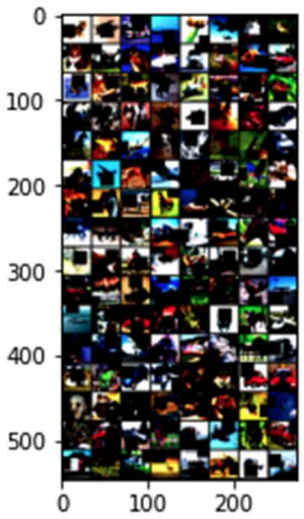
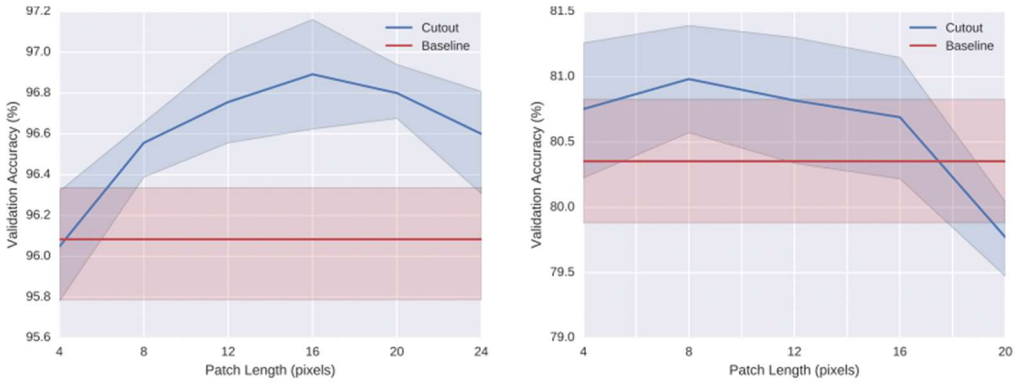


Figure 12. Cutout all

Wide residual network d = 28 (WRN-28-10), and shake-shake. The optimized performance is with a shake shake regularization model with a cutout of square shaped, performing a grid search.



(a) CIFAR-10 (b) CIFAR-100

Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	18.41 ± 0.27	$\mathbf{1.30 \pm 0.03}$
Shake-shake regularization [4]	-	2.86	-	15.85	-
Shake-shake regularization + cutout	-	$\mathbf{2.56 \pm 0.07}$	-	$\mathbf{15.20 \pm 0.21}$	-

Figure 13-15. Cutout performance

6. Discussion

6.1. Problem encountered

The three neural network architectures we've tried performed better than the naive implementation of our plain convolution layers and dense layers. However, we didn't try the scaling methods mentioned in EfficientNet simply because we don't have enough time and training resource, e.g., GPU.

Another is the compatibility with PyTorch and TensorFlow. One of our teammates has a MacBook, which does not have an intel chip, thus it was impossible for him to install a Cuda driver. Consequently, he ended up with TensorFlow that was the only one of the two that supported his hardware. Converting and rewriting existing code is also a very time consuming and resource intensive process.

6.2. Next Steps

The next step is to accomplish the integration among scaling, augmentation and network, comprehensively. We anticipate to achieve an accuracy of 92 percent or higher. Another step is to execute the transformative learning on other datasets, such as CIFAR-100 and SVHN. In addition, more layers can be designed and applied like how the EfficientNet does, to further reinforce the accuracy of the training model.

6.3. How does the approach differ from others? Was that beneficial?

The approach of the project is to integrate the ideas from multiple existing papers, while building up a new model on our own. Even though a fine integration result has not been accomplished yet, the training result might be a valuable reference to other later designs. Another feature of our model is to support hardware accessibility. Since the model does not require support of high-performance hardware, the training process can be done on personal devices with great efficiency.

7. Conclusion

In this project, we comprehensively study EfficietNet and Cutout to integrate the idea of augmentation, network and scaling. By applying the new training model, the overall training accuracy with CIFER-10 is accelerated from 50 percent to 87 percent. With a hardware friendly feature of the model, we anticipate more implementation and integration be done to the model, and a higher accuracy can be achieved.

Reference

[1] Chris Ratcliff. ML Compiled: Computer vision, 2021, from https://ml-compiled.readthedocs.io/en/latest/computer_vision.html

[2] Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2020.

[3] Terrance DeVries, Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout, 2017.

[4] Alex Krizhevsky, The CIFAR-10 dataset, 2015, from https://www.cs.toronto.edu/~kriz/cifar.html

[5] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile, 2019.

[6] Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014.