**Improvements on Image Classification with Convolutional Neural Network Architecture ResNet-50D**

CSC767 Project 2

Shokhina Badrieva and Anastasia Golev

College of Staten Island, Master of Computer Science Department

1. **Title and Authors:**

“Bag of Tricks for Image Classification with Convolutional Neural Networks” (He et al., 2018) [1]

“Deep Residual Learning for Image Recognition” (He et al., 2015) [2]

1. **Abstract:**

Deep neural networks have long been used for the problem of classification, but various issues such as ease of training and accuracy improvement are still being investigated. Our paper aims to research, run experiments on, and improve the deep network model titled “ResNet50-D”. The ResNet classification model is first introduced in the paper titled, “Deep Residual Learning for Image Recognition” (He et al., 2015) [1]. The model is then improved and expanded upon in the paper, “Bags of Tricks for Image Classification with Convolutional Neural Networks” (He, et al., 2018) [2]. The paper, “Deep Residual Learning for Image Recognition”, introduces the model architecture ResNet. The purpose of this model is to ease training of networks that are considerably deeper than those used prior to the publishing of the paper. The second paper, “Bags of Tricks for Image Classification with Convolutional Neural Networks”, investigates the problem of improving CNN model accuracy through methods other than improved model architecture: rather they use techniques to improve model accuracy using a collection of training procedures and model architecture refinements. Our paper will propose possible improvements to the ResNet50D architecture when tested on Imagenette dataset [3]. We will describe the suggested improvements, run experiments, and draw conclusions on the efficacy of our experiments in regards to model accuracy.

1. **Introduction:**
2. **Proposed architectures to solve problem of degradation**

The paper “Deep Residual Learning for Image Recognition” introduces the model architecture ResNet. As mentioned, the purpose of this model is to ease the training of networks that are considerably deeper than those used prior to the publishing of the paper. Research showed at the time of this paper that network depth was of crucial importance when it came to the results of the models [4]. As a result, the question of whether more layers mean better networks came under investigation. The first issue that emerged was one of vanishing gradients that stop networks from converging. This was addressed by normalized initialization and normalization layers. Then arose the issue of degradation, which this paper addressed. What was discovered is that the more layers a model had, the more saturated accuracy became: leading to the degradation of the model’s accuracy. The introduction of deep residual learning framework by He et al. looked to investigate this problem. We will discuss the method [1] used to address this problem.

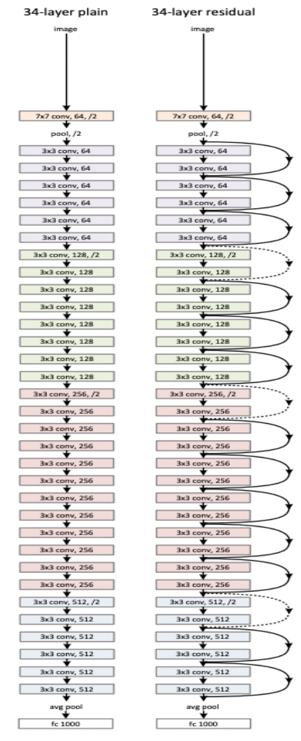
Introduced in [1] is the original ResNet architecture, the Resnet-50 variation, and finally the Resnet50-D architecture which we will be expanding upon. Each architecture builds on the previous one.  
The idea of the original ResNet is to let a few stacked layers fit a residual mapping. The underlying mappings are denoted as H(x) and the stacked nonlinear layers fit another mapping F(x)=H(x)-x where x denotes the inputs to the first of these layers. The formula of H(x) is then F(x)+x. The reasoning for using residual mappings is it is easier to optimize the residual mappings. The formula F(x)+x is achieved by feedforward neural networks with shortcut connections. Shortcut connections skip one or more layers. In the case of the ResNet architecture, identity shortcut connections are used in which the shortcut connections perform identity mappings, and their outputs are added to the outputs of the stacked layers. Residual learning is applied to every few stacked layers. A building block can be seen below in Figure 1. In Figure 1, there are two layers, F = W2σ(W1x) in which σ denotes ReLU and the biases are omitted for simplifying notations. The operation F + x is performed by a shortcut connection and element-wise addition.

Diagram

Description automatically generated

Figure 1. Residual Learning: a building block

A plain network is the first network used as a base for the ResNet. In it, the convolution layers mostly have 3 × 3 filters and follow 2 rules. The first rule is that if the output feature map size is the same for layers, the number of filters is also the same. The second rule is that if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. Downsampling is also performed by convolution layers that have a stride of 2. There is a global average pooling layer at the end and a 1000 way fully connected layer with Softmax. In total, the plain baseline has 34 weighted layers. The ResNet is built off this original baseline. In it, shortcut connections are added. The identity shortcuts can be directly used when the input and output are of the same dimensions. The two architectures can be seen in the figure below.



The next step of the architecture is to use the 34 layer ResNet and create deeper nets for the ImageNet dataset. For the sake of training time, the building blocks are modified as bottlenecks. In this modification, for each residual function F, a stack of three layers are used instead of two. The layers in each block are a follows: 1 × 1, 3 × 3, and 1 × 1 convolutions, where the 1 × 1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3 × 3 layer a bottleneck with smaller input/output dimensions. This can be seen in Figure 2, where the image on the right represents this new bottleneck block for ResNet-50. In ResNet-50, the 34 layer ResNet is used, but each 2 layer block is replaced with a 3 layer bottleneck block which gives us the 50 layer ResNet50.

Diagram

Description automatically generated

Figure 2. Image on right represents this new bottleneck block for ResNet50 where three layers are used for a residual function.

The second paper, “Bags of Tricks for Image Classification with Convolutional Neural Networks”, investigates the problem of enhancing CNN model accuracy through methods other than improved model architecture: instead, the authors experimented with varying techniques to improve model accuracy using a collection of training procedures and model architecture refinements. The paper introduces a new architecture called ResNet-50-D and implementations in training onto the already improved ResNet architecture. The aim of the paper is to improve training, increase accuracy, match the computational complexity, and to generalize their approach to other networks. The paper explores the ResNet-50 architecture along with 2 popular tweaks called ResNet-50-B and ResNet-50-C, ultimately proposing a new model tweak called ResNet-50-D in which a 2 × 2 average pooling layer with a stride of 2 before the convolution is added. In addition to this, they explore 4 training refinements such as Cosine Learning Rate Decay, Label Smoothing, Knowledge Distillation, and Mixup training to further improve model accuracy.

The architecture involves an input stem, 4 subsequent stages, and a final output layer. At stage 1, the input stem consists of a 7 × 7 convolution with an output channel of 64 and a stride of 2. This is followed by a 3 × 3 max- pooling layer. The input stem then reduces the width and height of the input by 4 and increases the channel size to 64. Beginning with stage 2 of the network, each stage is passed through a down-sampling block followed by several residual blocks. Within the downsampling block, there are 2 paths: path A and path B, each of which performs different convolutions themselves. Path A consists of 3 convolutions of varying kernel sizes, strides, and output sizes. The first convolution within part A is a 1 × 1 convolution with a stride of 2 which halves the width and height of the input, and output size of 512. The second is a 3 × 3convolution with an output size of 512. The final convolution in path A is a 1 × 1 convolution whose output size is 2048, 4 times larger than the others, resulting in a bottleneck structure. Path B contains only 1 convolution: a 1 × 1 convolution with a stride of 2 and an output size of 2048. Being that the output shapes of the two paths match after their respective convolutions, they are summed to obtain the output of the downsampling block. It is clear that to obtain different ResNet models, the number of residual blocks in each stage can be diversified. The authors propose an adaption to the ResNet-50 architecture based on this, called ResNet-50-D. Introduced as tweaks to the original ResNet-50 architecture are the ResNet-50-B and ResNet-50-C architectures. The ResNet-50-B architecture exchanges the stride sizes of the first 2 convolutions in path A: allowing for more information to be passed through the initial convolution. ResNet-50-C is a more cost-efficient architecture, replacing the 7 × 7 convolution within the input stem and instead performing three 3 × 3 convolutions: the first 2 use output sizes of 32 and strides of 2, and the last uses an output size of 64. Reducing the size of the kernel not only is a cost benefit but also improves the efficiency of the model. Because there are now 3 convolutions within the input stem, the number of parameters necessary has also decreased.

Finally, proposed in this paper is the ResNet-50-D architecture whose goal is to have minimal to no information ignored in the process of convolution. This architecture is built off the ResNet-50-B architecture, in which path A is modified. ResNet-50-D in-turn modifies path B. The authors discovered that the addition of a 2 × 2 average pooling layer with a stride of 2 before the convolution prevents the loss of 25% of information, as was seen in the ResNet-50-B architecture. This new addition has shown improved results in practice and does not impact the computational cost in a significant manner.

Diagram

Description automatically generated

1. **Results of authors’ experiments**

The data used in the original papers is the ImageNet Large Scale Visual Recognition Challenge (ISLVRC2012) dataset [5]. This dataset includes 1.3 million images for training and 1000 classes. The images in the dataset are manually annotated training images.

The results of the original ResNet architecture can be seen below. The results show that ResNet produced less error on ImageNet when compared to state-of-the-art algorithms such as VGG and GoogleNet.

Table

Description automatically generated

Next, the results of ResNet50D show improved results in practice and there is no impact the computational cost in a significant manner. Results can be seen in Figure 4 below. The ResNet50D tweak has a higher top 1 and top 5 accuracy than its predecessors.

**Table

Description automatically generated**

1. **Improving the ResNet50D Architecture**
2. **Our Initial Experiments on imagenette using ResNet50D**

For the experiments, we used a subset of the ImageNet dataset called Imagenette due to constraints on training time. The following parameters were used for training based on information given in the papers: image\_size=(224, 224), lr= 5e-3, smoothing=0.1, mixup=0.2, sutmix=1.0, batch\_size=32, and num\_epochs=50 in which the first epoch is epoch 0 and the last is epoch 49.

Our results show that a top-1 accuracy of 83% and a top 5 accuracy of 98% were achieved in epoch 35. Figure 4 shows the results the authors got when training the ResNet50D model on Imagenet, which is considerably larger than the subset we used. The results the authors achieved were 77% accuracy for top-1 and 93% accuracy for top-5. The top-1 percent accuracy is the accuracy of the actual expected label when compared to the predicted label. Our experiment results were notably higher than the results of the authors’ but this is reasonable in that the dataset used with our experiment is substantially smaller, raising the percent accuracy for correct predictions. The table below represents the Top-1, Top-1 error, Top-5, and Top-5 error metrics of the top 10 epochs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Top-1** | **Top-1 error** | **Top-5** | **Top-5 error** |
| **35th** | 83.0064 | 16.9936 | 98.1146 | 1.8854 |
| **38th** | 82.7006 | 17.2994 | 98.0382 | 1.9618 |
| **39th** | 82.7516 | 17.2484 | 98.1911 | 1.8089 |
| **40th** | 82.6752 | 17.3248 | 98.0637 | 1.9363 |
| **44th** | 82.6752 | 17.3248 | 98.2166 | 1.7834 |
| **45th** | 82.5987 | 17.4013 | 98.2675 | 1.7325 |
| **46th** | 82.6497 | 17.3503 | 98.1911 | 1.8089 |
| **48th** | 82.7006 | 17.2994 | 98.0892 | 1.9108 |
| **49th** | 82.6497 | 17.3503 | 97.9108 | 2.0892 |
| **50th** | 82.9045 | 17.0955 | 98.3185 | 1.6815 |

After completion, the best metric was the 35th epoch: with a loss value of 0.4507, accuracy at Top-1 and Top-5 being 83.0064% and 98.1146 respectively. The mean is (0.485, 0.456, 0.406) and the standard deviation is (0.229, 0.224, 0.225). The graphs for training and validation loss can be seen below. In addition, the graphs for top 1 percent accuracy (the predicted class matched the class label) and top 5 accuracy. The graphs for loss and accuracy can be seen below.

**Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated**

1. **Proposed Improvements**

Based on the experimental results, we noted some room for improvement. When thinking about possible improvements, we considered various types of adjustments to raise accuracy. Some possibilities of improving model accuracy include changes with data (data augmentation, more images, etc), changes in architecture, changes in parameters, and various other methods. We will discuss the adjustments we tested and their results on model accuracy.

The first adjustment we considered has to do with a pooling layer added to the architecture. One issue with feature maps outputted by layers in neural networks is that they often can be sensitive to the location of the features in regard to the input. Pooling layers provide a solution to down sampling feature maps. They look at patches of the feature map when summarizing presence of features in down sampling. The architecture presented by the authors, called ResNet-50-D, focuses on path A: an additional 2 × 2 average pooling layer with a stride of 2 before the convolution. One method to consider for our possible improvement is max pooling rather than average pooling within path A [6]. An average pooling layer results in a smoother image while maintaining the features which explains the additional 25% of information that the authors were able to extract through their experiment. A max-pooling layer, as the name suggests, will focus on the maximum values, bringing out more intensity from an image [7]. We hypothesize that with the right kernel size, this will further improve our accuracy.

The second improvement we will test is changing the optimizer function to Adagrad [8]. An optimizer function is used to modify parameters of a model such as learning rate. Including an optimal optimizer function raises accuracy and reduces overall loss. Our experimental results were run on a Stochastic Gradient Descent (SGD) optimizer which is a method to optimize a function with smoothness properties, resulting in a more generalized output. One suitable substitution may be the Adagrad optimizer. Adagrad is an optimizer than uses per parameter learning rate method. Specifically, it performs larger updates if the parameters are sparse and smaller updates if the parameters are less sparse. Adagrad does well if there is sparse data, which is part of the reason we chose to test it. Our dataset is smaller compared to the authors, so Adagrad seems like a suitable option for improvement.

The third optimizer we wanted to test out in our experiments for model improvement is RMSProp [9]. RMSProp is also an adaptive learning rate algorithm which shares some similarities with Adagrad. Adagrad keeps a sum of squared gradients and adapts the learning rate by diving it by the sum. Due to this, as training continues, steps get smaller because the squared gradients grow as training continues. RMSProp is very similar to this, but instead of letting the squared gradients accumulate over training, RMSPRop keeps a moving average of squared gradients. Due to this main difference, we want to run experiments to see whether RMSProp gives us better results than Adagrad.

The fourth improvement we will test is changing the optimizer function to Adam [10]. Adam combines the best properties of the AdaGrad and RMSProp algorithms, so we hypothesize that Adam will raise accuracy compared to the second improvement. Furthermore, the Adam optimizer converges faster than SGD; combining the Root Mean Square Propagation and Adaptive Gradient Algorithm to compute the learning rate.

The last improvement we will consider is to test out a different activation function with the next best optimizer from the previous examples. The activation function we will consider is “Elu” [11]. The activation function titled ‘ELU” stands for exponential linear unit. This function fixes some issues associated with Relu [12], which was used by the authors from the original paper. Elu has pros over Relu such as avoiding the dead Relu problem. The dead Relu problem occurs when components of a network are mostly never updated to a new value. For some models, this can be an issue. Testing out the Elu activation function will give us a better idea of which function will work best with our dataset and the new optimizer. In this experiment we will also change the stride to 4. Stride is how many pixels shift over the input in our input matrix. The authors used a stride of 2. Increasing stride has the effect of down-sampling an image so lower details are not as important. We theorize that implementing these improvements will raise the accuracy of the model.

1. **Max Pooling Results**

Note: The experiment for Max Pooling improvement couldn’t fully run by the submission time since we ran out of free GPU, but we will include it in the final presentation as well as resubmit paper with results

1. **Adagrad Results**

For the Adagrad improvement, we were able to achieve a top 1 accuracy of 87 percent. Training ran for 50 epochs and the best results were achieved in epoch 43. The top 5 percent accuracy for this improved model was 99 percent. We achieved a top 1 error of 12 and a top 5 error of 0.9. The results for training/validation accuracy and training/validation loss can be seen below.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

To further analyze the results, we looked at the top ids produced by the model for each image in the validation dataset. When looking at top id’s, the first column presents the top 1 prediction for the specific image. The other four columns give the other predictions. We will present some correct predictions as well as incorrect predictions for reference.

New stuff starts here

Table

Description automatically generated

Figure. Two correctly classified images and two incorrectly classified images using Adagrad. The first numerical column shows the top 1 prediction. Correct prediction is ID 0.

Correct Prediction Examples:

A person holding a large fish

Description automatically generated with medium confidence Correctly predicted as class 0: tench

A person holding a fish

Description automatically generated Correctly predicted as class 0: tench

Incorrect Prediction Examples:

**A picture containing grass, outdoor

Description automatically generated** Correctly predicted as class 8: golf ball

 Correctly predicted as class 5: French horn

New stuff ends here

1. **RMSProp Results**

For the RMSProp experiment, we were able to achieve a top 1 accuracy of 75 percent. Training ran for 50 epochs and the best results were achieved in epoch 37. The top 5 percent accuracy for this model was 97 percent. We achieved a top 1 error of 25 and a top 5 error of 2.9. The results for training/validation accuracy and training/validation loss can be seen below.

Graphical user interface, chart, line chart

Description automatically generated**Chart, line chart

Description automatically generated**

We will present some correct predictions as well as incorrect predictions for reference.

New stuff starts here

Graphical user interface, text, application, table

Description automatically generated

Figure. Two correctly classified images and two incorrectly classified images using Adagrad. The first numerical column shows the top 1 prediction. Correct prediction is ID 1.

Correct Prediction Examples:

A dog lying on the ground

Description automatically generated with medium confidence Correctly predicted as class 1: English Springer

A brown and white dog lying in grass

Description automatically generated with low confidence Correctly predicted as class 1: English Springer

Incorrect Prediction Examples:

A picture containing tree, outdoor, mammal

Description automatically generated Correctly predicted as class 0: tench

A dog lying on a bed

Description automatically generated Correctly predicted as class 4: church

New stuff ends here

1. **Adam Results**

Note: The experiment for Adam improvement couldn’t fully run by the submission time since we ran out of free GPU, but we will include it in the final presentation as well as resubmit paper with results

1. **ELU Results**

For the ELU improvement with stride of 4 and using the Adam optimizer, we were able to achieve a top 1 accuracy of 89 percent. Training ran for 50 epochs and the best results were achieved in epoch 37. The top 5 percent accuracy for this model was 99 percent. We achieved a top 1 error of 11 and a top 5 error of 1. The results for training/validation accuracy and training/validation loss can be seen below.

Chart, line chart

Description automatically generated**Chart, line chart

Description automatically generated**

We will present some correct predictions as well as incorrect predictions for reference.

New stuff starts here

Table

Description automatically generated

Figure. Two correctly classified images and two incorrectly classified images using Adagrad. The first numerical column shows the top 1 prediction. Correct prediction is ID 2.

Correct Prediction Examples:

A picture containing indoor, blue, silver

Description automatically generatedCorrectly predicted as class 2: cassette player

A picture containing car

Description automatically generated Correctly predicted as class 2: cassette player

Incorrect Prediction Examples:

**** Correctly predicted as class 5: French Horn

A picture containing text, electronics

Description automatically generated Correctly predicted as class 5: French Horn

New stuff starts here

1. **Comparison With Original Architecture**

For our analysis, we will compare the experiments to the results we received using the original architecture presented by the authors on the smaller Imagenette dataset we tested out. As mentioned before, our original results show that a top-1 accuracy of 83% and a top 5 accuracy of 98% were achieved in epoch 35. Changing the optimizer to SGD to Adagrad increased the accuracy by 3 percent and reduced the top 1 error from 16 to 12. We were expecting accuracy to increase since Adagrad is an adaptive optimizer. Surprisingly, the results for RMSProp decreased accuracy to 75 top 1 percent accuracy and increased the top 1 percent error from 16 to 25. These results were slightly surprising since RMSProp is also an adaptive optimizer, similar in implementation to Adagrad. The final experiment of changing the activation function to ELU, using optimizer Adam and stride of 4 gave the best results for our experiments. We were able to achieve a top 1 accuracy of 89 percent and error of 11. We were successfully ale to improve model accuracy and reduce loss by implementing the changes we mentioned before. We still must test out two more experiments fully. These include testing Adam alone and changing the global pooling layer from average to max pool.

1. **Conclusions**

Our project’s goal was to study and run experiments on a recent classification model architecture. We chose the ResNet50D architecture because it was interesting to see how it evolved over time and how the authors of [2] were able to improve upon the architecture mentioned in [1]. We were also interested in the fact that ResNet50D is a deep network that keeps computational cost down. Our goal was to run various experiments and attempt to improve the model on the smaller Imagenette dataset. We chose the improvements based on our best knowledge of various parameters used to tune a model. After trying out different parameter changes, we were able to increase the model accuracy from our initial experiments from 83 to 89. We were also able to lower the loss from 16 to 11.

**REFERENCES**

[1] Kaiming He et al. Deep Residual Learning fro Image Recognition. (arXiv:1512.03385, 2015)

[2] Tong He et al. Bag of Tricks for Image Classification with Convolutional Neural Networks. (arXiv:1812.01187, 2018)

[3] H. Husain. Fastai/ Imagenette. 2021. Github repository at https://github.com/fastai/imagenette

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.)

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Computer Vision and Pattern Recognition (CVPR), 2009.*

**//need to add**

**6. average pooling**

**7.max pooling**

**8. adagrad**

**9.rmsprop**

**10.adam**

**11. elu**

**12. relu**