MiniProject 4: Report
Comp 551 - Group 49

Benoit Kuzak (260744679)
Miranda LaBrash (260631836)
Sitao Lu (260866413)

December 19, 2020

Abstract

This work explores the original ResNet neural network model (https://github.com/KaimingHe/deep-residual-networks) proposed by Kaiming He et al. in the groundbreaking paper "Deep Residual Learning for Image Recognition". This exploration was conducted through systematic ablation testing performed on the ResNet18 and ResNet34 models featured in the paper, altering the batch and kernel size, stride, and filter architecture. Their performance was tested on image recognition tasks using the CIFAR-10 image dataset, as used in the original paper. Additionally, this report outlines the performance of numerous alterations made to the model in an attempt to increase performance, including the use of different bottleneck and dropout block configurations compared to basic block architecture. Ultimately we find that our strongest performing ResNet re-implementation achieves a 85.03% accuracy, compared to the original Keras model's performance of 91.0%.

1 Introduction

The ResNet deep residual network model developed by Kaiming He et al. of Microsoft Research Asia represents a formidable contribution to the newest generation of neural networks. The model features a ground-breaking residual feed-forward structure that enables the construction of extremely deep networks. The goal of this work was to analyze the originally proposed ResNet18 and ResNet34 neural networks in detail through ablation studies, and explore new improvements that could be made.

To begin, we reproduced the results of the paper using the canonical 3rd party Keras ResNet implementation and tested the model on the image recognition task employed by Kaiming He et al.: categorizing the 50 000 images of the CIFAR-10 dataset. We found that our model performed consistently to the results of the original paper and other 3rd-party implementations. Once a baseline implementation was achieved, we conducted extensive ablation tests, altering the batch size, stride, kernel, and filter hyperparameters of the model, finding significant variations in performance accuracy.

Additionally, we experimented with enhancing the basic block system of the original ResNet model by implementing a bottleneck block architecture, expecting to decrease time complexity while maintaining model accuracy. Our ResNet18 results showed a slight decrease in accuracy (from 86.03% to 84.71% accuracy), and a decrease in time per epoch by almost half (from 67s to 32s). ResNet34 demonstrated a smaller decrease in accuracy (from 85.94% to 85.13% with bottleneck), but resulted in a significant increase in time per epoch (from 67s to 105s). Overall, our ResNet18 test with a batch size of 100 performed the strongest out of all ResNet variations, achieving an accuracy of 86.03%.

2 Original Paper

In "Deep Residual Learning for Image Recognition", Kaiming He et al. proposed a new residual learning framework, ResNet, which enabled the training of significantly deeper neural networks than what had been successfully trained previously. Not only were ResNet implementations demonstrated to be empirically easier to optimize, but they also gained accuracy with considerably increased depth. When deep neural networks start converging, a degradation problem exists. As the network depth increases, accuracy saturates and then degrades rapidly. This degradation is empirically not caused by overfitting, and adding more layers to a suitably deep model leads to higher training error. The ResNet model describes a system of stacked non-linear network layers. Given an underlying mapping H(x), the stacked nonlinear layers are fit to another mapping of F(x) := H(x) - x. The original mapping is then recast into F(x) + x. F(x) + x is formulated by "shortcut" connections that skip over one or multiple layers in the feed-forward model. He et al. 2016a

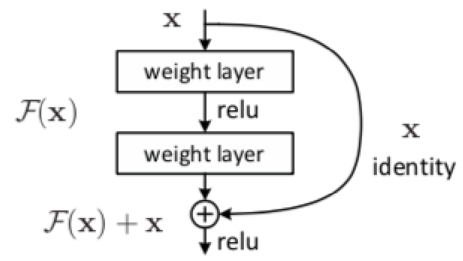
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1: Example Residual Learning Layer

Kaiming He et al. trained and tested ResNet image recognition performance on both the CIFAR-10 and ImageNet databases (the latter for which the team famously won the ILSVRC 2015 classification challenge). We chose to test exclusively on CIFAR-10, as Kaiming He et al. had reported very similar test results between the two datasets and our team was faced with significant computational constraints. To train the CIFAR-10 dataset, Kaiming He et al. used a weight decay of 0.0001, momentum of 0.9, and mini batch size of 128. The learning rate was initialized at 0.1, and divided by 10 at 32 000 and 48 000 iterations. Training was terminated at 64k iterations. For training, 4 pixels are padded on each side of each example image.

3 Dataset and Setup

The dataset used for analysis was the CIFAR-10 dataset, which contains 60 000 examples of coloured images, each represented by a 32x32 pixel matrix contained in a numpy array of uint8s. The dataset is evenly divided into 10 different image classes: airplanes, cars, deer, cats, birds, frogs, horses, ships, and trucks. The dataset contains 6 000 examples of each class. There are 50 000 training images and 10 000 test images. The dataset has been divided into five training sets of 10 000 images, and one test set of 1000 randomized examples from each class.

Figure 2: Example CIFAR-10 images and their corresponding labels

Manual feature extraction was not employed, as feature extraction takes place automatically within the ResNet framework. Accuracy and loss were used as evaluation criteria in training and validation.

4 Proposed Approach

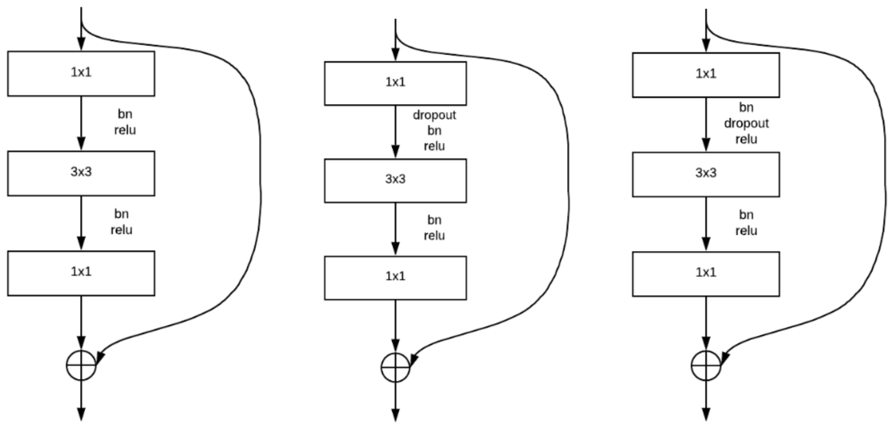

Figure 3: Original Bottleneck Block vs Dropout First Block vs Proposed Block

In our experiment, we tested and contrasted ResNet18 and ResNet34 with numerous hyperparameters, beginning with the original parameters used in the ResNet paper. We systematically tested the impact of different parameters one by one: batch size, stride and kernel size. Based on the result of our prior test, we selected the best parameter for batch size, stride and kernel size and proceeded to test the next parameter.

Kaiming He et al. replaced a stack of 2 layers of 3x3 convolutions (i.e. basic blocks) by 3 layers of 1x1, 3x3, 1x1 convolutions in a technique called bottleneck design. Kaiming He et al. indicates that bottleneck design will have the same time complexity as the replaced basic block but leads to deeper layers. We applied the bottleneck design to ResNet18 and compared the runtime and accuracy with the original design.

Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang (2018) tested the combination of dropout and batch normalization, both strong techniques in deep learning to improve accuracy. They showed that applying dropout before batch normalization almost always leads to worse results but the reverse order works better. We adopted their findings and combined dropout with batch normalization into the bottleneck design. We specifically tested two designs, applying dropout before batch normalization (dropout first block) and reverse order (proposed block), as is shown in Figure 3.

5 Results

Figure 4 describes the results of a series of tests on ResNet18 (basic block version) where the default stride is set to (1,1). Note that for the initial layer of a block, the stride is always (2, 2). A test is then conducted where the stride is changed to (2,2) at all times, unchanged for first layer of block. The default kernel size for these blocks is (3,3) and these will be changed in 2 different tests: one test with a kernel size of (2,2) and the other with a kernel size of (4,4). The last test changes the initial filters from 64 to 32, in order to see its impact.

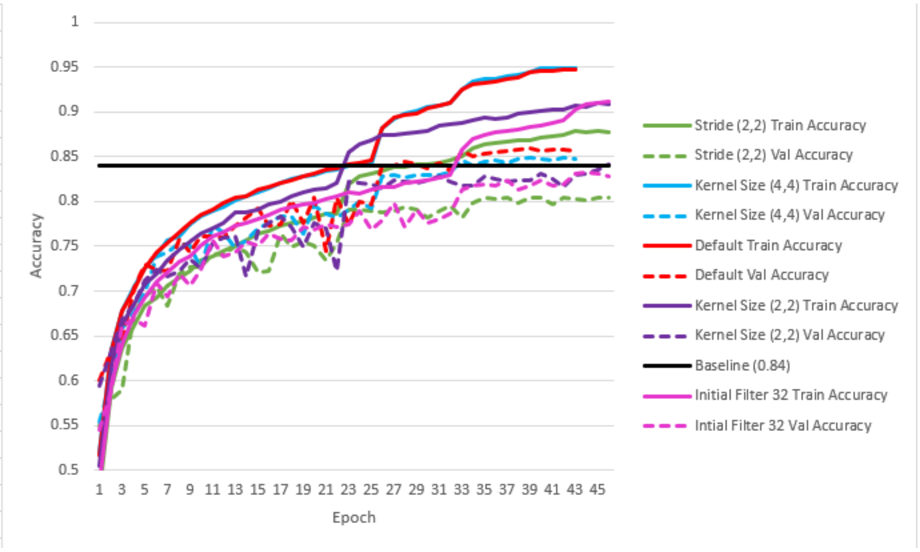

Figure 4: Accuracy of Model over Epochs with different parameter changes.

We see that the Kernel Size (3,3) and Kernel Size (4,4) have an identical training accuracy over the epoch, however Kernel Size (4,4) is slower per epoch, and also leads to a slightly lower final validation accuracy. Every other parameter change does not affect the epoch time. However they all performed worse, from 1.5% to 5% worse in accuracy by the final epoch.

Parameter	Validation Accuracy	Time per Epoch (s)
Default Parameters	0.8568	30
Stride (2,2)	0.8042	30
Kernel Size (4,4)	0.8469	37
Kernel Size (3,3)	0.8416	29
Filters at 32	0.8283	28

Table 1: Parameter changes and their accuracy (ResNet18 - Basic Block)

We ran this test with all default settings in accordance to the original paper, with the exception of altering the batch size. We tested larger batch sizes vs small batch sizes. Since the code is tested based on the CIFAR-10 dataset, which has 50,000 training images(divisible by 10), we included fully divisible batch sizes 200, 100 and 50 as options. From Table 2, we noticed that fully divisible batch sizes tend to have higher training time per epoch values but lower epoch to finish training. In addition to the training time, lower batch size empirically works better on the CIFAR-10 dataset. Ultimately, when batch size is 100, ResNet is able to reach 86.03% accuracy, which is higher than any other batch size by 1% - 2%.

Batch Size	Validation Accuracy	Time per Epoch (s)	Total Training Time (s)
100 (Default)	0.8603	67s	3350
32	0.8409	155	5776
50	0.8517	105	4387
64	0.8593	92	4743
128	0.8482	55	2035
200	0.8480	41	1845
256	0.8478	41	1804

Table 2: ResNet18 - Basic Block with different batch size

After replacing ResNet18's basic block with bottleneck architecture, we tested and recorded the accuracy and time per epoch in Table 3. We noticed that both accuracy and time per epoch is inferior compared to the original ResNet18. We speculate that the model with 26 layers is still too simple for the dataset and the 1 second difference is negligible due to the machine variance.

Model	Accuracy	Time per Epoch (s)
ResNet18 - Basic Block	0.8603	30
ResNet18 - Bottleneck	0.8471	31

Table 3: Basic Block vs Bottleneck Block accuracy and speed.

We tested the modified bottleneck design and recorded the result in Table 4. We verified that applying dropout after batch normalization leads to better accuracy than the original bottleneck block with the same training time per epoch. Additionally we have seen that the dropout first block has much worse accuracy and with longer time per epoch.

Model	Accuracy	Time per Epoch (s)
Bottleneck	0.8471	32
Dropout First Block	0.7812	36
Proposed Block	0.8505	32

Table 4: Bottleneck vs. Dropout vs. Proposed accuracy and speed on ResNet18.

6 Discussion and Conclusion

Ultimately we find that our strongest performing ResNet re-implementation achieves a 85.03% accuracy, compared to the original Keras model's performance of 91.0%. A possible option for further testing to improve our accuracy further would be to reduce or increase the number of blocks. As every ResNet model uses four blocks with differing numbers of layers/repetitions, we could change the repetition counts per block and change the amount of blocks to see what the effects are. This would affect the amount of filters since each consecutive block has twice the amount of filters as the previous. Additionally, since each block's first layer differs from the other layers within the block, these could have more or less of an effect depending on how we change the block count. It is also worth noting that having only a single block makes this model no longer a ResNet model, since the blocks are what define the model, with the initial input being used in each block. We attempted to test increasing the amount of blocks, but were stymied by severely increased runtime, which we could not afford with our resources. This, however, this does not mean that the final results could not show improvement in the model,therefore it proves worthwhile to test.

7 Statement of Contributions

Sitao tests batch size, new bottleneck design, Benoit tests kernel size, stride, and tried ImageNet, Miranda writes most of the report.

References

[1] Francois Chollet et al. *Keras*. `https://github.com/fchollet/keras`. 2015.

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016, pp. 770–778.

[3] Kaiming He et al. "Identity Mappings in Deep Residual Networks". In: *CoRR* abs/1603.05027 (2016). arXiv: `1603.05027`. URL: `http://arxiv.org/abs/1603.05027`.

[4] Raghavendra Kotikalapudi. *Keras-Resnet*. `https://github.com/raghakot/keras-resnet`. 2016.

[5] A. Krizhevsky, I. Sutskever, and G.E. Hinto. "ImageNet Classification with Deep Convolutional Neural Networks". In: *ACM* (2012), pp. 84–90.

[6] H. Li et al. "CIFAR10-DVS: An Event-Stream Dataset for Object Classification". In: 2017, pp. 11–309. DOI: `0.3389/fnins.2017.00309`.

[7] Xiang Li et al. "Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift". In: *CoRR* abs/1801.05134 (2018). arXiv: `1801.05134`. URL: `http://arxiv.org/abs/1801.05134`.

[8] Travis E. Oliphant. *A guide to NumPy*. Trelgol Publishing, USA, 2006.