Mini-Project 1: Residual Network Design

Shubham Shandilya New York University ss15590@nyu.edu
Tanisha Madhusudhan New York University tm3805@nyu.edu
Bhargav Makwana New York University bm3125@nyu.edu

Abstract

The main aim of the project is to achieve best possible test accuracy on CIFAR-10 data set using ResNet model. There is a constraint on trainable parameters, restricted up to $\approx 5M$ in total. Thus hyper-parameter tuning of the parameters like number of residual layers N, number of residual blocks in given residual layer i B_i, number of channels in residual layer i C_i, convolution kernel size in residual layer i F_i, skip connection kernel size in residual layer i K_i and average pool kernel size P has been performed keeping in mind the limitations of the constraint. The later part of the report describes the process and steps of how we are able to achieve more than 95% accuracy with the help of fine tuning the hyper-parameters along with well-judged use of Optimizer and Regulators.

1 Introduction

A residual neural network(ResNet) [2] is an artificial neural network(ANN) [7] most commonly used for image classification that can utilize skip connections, to jump over some layers in a stride. Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds up learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through initially. The network then gradually restores the skipped layers as it learns the feature space. ResNets are the backbone of many computer vision tasks. Advantage of ResNets is that they can help maintain a low error rate much deeper in the network. Residual network architecture has the capability to stack up to 1000 Deep layers and still be able to train unlike normal stacking of the DL layers. This makes it not only special but also allows one to train the network faster and achieve reasonable accuracy with less number of layers. The project here is a genuine attempt to showcase the use of Residual Networks and its potential to for further developments and applications.

2 Methodology

For CIFAR-10 data-set building and training your network from scratch. We first define a skeleton class in Pytorch as given in the python file so that it can be used as a template to add layers seamlessly. Next, we developed a custom ResNet class with all the hyper parameters added as an argument as a way of tuning them. Functional solutions such as dropout regularization, batch normalization, transfer learning, and pretraining have been developed to try to extend Deep Learning for application on smaller datasets like CIFAR-10.

2.1 The Approach:

The approach to train the Residual Network is outlined below:

1. The 5 versions of resnet models [2], which contains 18, 34, 50, 101, 152 layers respectively were inspected. We started with a ResNet-18 model, so-called because it has 18 layers as suggested by the project description.

2. The ResNet consists of N Residual Layers (N = 4 for ResNet-18). Each residual layer contains one or more residual blocks. Layer i has B_i blocks (for ResNet-18, B_i = 4 for i = 1, 2, 3, 4). Residual layers are preceded by a fixed convolutional block and followed by an average pooling and fully connected layer.
3. Since ResNet-18 has around 11M (more than 5M) parameters we modified the network in 4 different ways. This was intended for two main reasons, one being to satisfy the constraints and the other to get to the best possible accuracy with minimal configurations.
4. The other element we tested was the optimizer. We have trained our network on SGD, Adam and Momentum SGD. Although theoretically Adam is considered more stable and known to converge faster, studies have shown that SGD and momentum [6] SGD give better results after fine-tuning of hyper parameter. The reason behind this is that Adam helps to achieve sharp local minima but SGD and Momentum SGD although bit slower to train, SGD generalizes better than Adam and thus results in improved final performance.
5. The CONV layers in any residual block of residual layer i have C_i channels. As per the general convention, $C_{i+1} = 2C_i$, C_1 was selected. The CONV layers and SKIP connections in any residual block of residual layer i have filters of size K_i x K_i and $K_i \times K_i$, respectively.
6. The first CONV in the first residual block of residual layer $i \geq 2$ is special because it has a stride of 2. For this reason, the skip connection of the first residual block of residual layer i $\geq$ 2 also has a stride of 2. In all other cases the stride is 1.
7. The 32 × 32 × 3 input image is processed by conv layer $F_1 \times F_1$ sized filters and C_1 output channels. This is followed bn and relu.
8. The last residual layer's output feeds into a $P \times P$ average pooling layer, followed by an appropriately sized fully connected layer. Finally, the output of dimension 10 is passed through a softmax layer.

2.1.1 Data Augmentation

To build useful Deep Learning models, the validation error must continue to decrease with the training error. Data Augmentation is a very powerful method of achieving this. The augmented data will represent a more comprehensive set of possible data points, thus minimizing the distance between the training and validation set, as well as any future testing sets. We conducted experimentation on the CIFAR-10 data-set, which consists of 50k training images and 10k testing images in 10 classes to employ best data augmentation techniques as discussed here [5] employing Rotations, Blurring, Cropping, Flipping and Normalization. We saw that among these **Normalization, RandomCrop and RandomHorizontalFlip** served more useful than others.

2.1.2 Regularization

Regularization is a set of techniques which can help avoid over-fitting in neural networks, thereby improving the accuracy of deep learning models when it is fed entirely new data from the problem domain. It works on the premise that smaller weights lead to simpler models which in results helps in avoiding over-fitting. So to obtain a smaller weight matrix, these techniques add a 'regularization term' along with the loss to obtain the cost function.

$$Cost\ function\ =\ Loss\ +\ Regularization\ term$$

The difference between L1 and L2 regularization techniques [4] lies in the nature of this regularization term. In general, the addition of this regularization term causes the values of the weight matrices to reduce, leading simpler models.

L2 technique forces the weight to reduce but never makes them zero. Also referred to as *ridge regularization*, this technique performs best when all the input features influence the output, and all the weights are of almost equal size. We employed L2 because of the architecture's constraint using the TORCH.OPTIM package alongwith SGD optimizer, taking the **weight decay of 5e-4**.

Dropout is a regularization technique [3] that zeros out the activation values of randomly chosen neurons during training. This constraint forces the network to learn more robust features rather than relying on the predictive capability of a small subset of neurons in the network. This has helped to decrease train and test loss, thus reflecting the in the model's ability to generalize better. Inplace Dropout2d layer with **rate of 0.25** is appended after every *BatchNorm2d* layer which helped us generate reasonable results with other configurations.

Batch normalization is another regularization technique that normalizes the set of activations in a layer. Normalization works by subtracting the batch mean from each activation and dividing by the batch standard deviation. This normalization technique, along with standardization, is a standard technique in the pre-processing of pixel values.

2.1.3 Choosing size of mini-batches:

The batch size defines the number of samples to work before updating the internal model parameters. Smaller batch sizes are used for three main reasons:

- Smaller batch sizes are noisy, offering a regularizing effect and lower generalization error.
- Smaller batch sizes make it easier to fit one batch worth of training data in memory (i.e. when using a GPU).
- A third reason is that the batch size is often set at something small, such as 32 examples, and is not tuned by the practitioner. Small batch sizes such as 32 do work well generally.
- We experimented 4 different values of batch sizes with our model: $32, 64, 128, 256$.

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches.

The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset.

The number of epochs can be set to an integer value between one and infinity. The algorithm can be run for as long as required and even stopped using other criteria besides a fixed number of epochs, such as a change (or lack of change) in model error over time.

They are both integer values and they are both hyperparameters for the learning algorithm, e.g. parameters for the learning process, not internal model parameters found by the learning process.

In order to keep the computing cost at bay, we used only 250 epochs as training further did not make much difference in the accuracy.

3 Results

3.1 Model Architecture

To achieve the goal to *maximize accuracy* on the CIFAR-10 benchmark while keeping the *size* of your ResNet model under some specified fix budget, we experimented with 4 Architecture using the residual connections.

3.1.1 ResNet-18 but with only 3 Layers.

As can be seen in summary information 5, the architecture resembles the one for the ResNet-18 model with one residual layer less. This allowed us to reduce the number of trainable parameters from $\approx 11M$ to **2,777,674**.

Following are the Hyper-parameters used:

- N: Residual Layers: 3
- B_i: Residual blocks in Residual Layer i: $[2, 2, 2, 2]$
- C_i: channels in Residual Layer i: $[64, 128, 256]$

- F_i: Conv. kernel size in Residual Layer i: $[3, 3, 3]$
- K_i: Skip connection kernel size in Residual Layer i: $[3, 3, 3]$
- P: Average pool kernel size: 4

The Test accuracy after 200 epochs for different Batch size and Optimizer is shown in figure 6 wherein Optimizer = *SGD + Momentum* with batch size = 256 helped model to optain Best accuracy of **93.060%**.

3.1.2 ResNet-18 with C_1 = 32.

As can be seen in summary information 7, the architecture resembles the one for the ResNet-18 model with C_1 = **32**. This allowed us to reduce the number of trainable parameters from $\approx 11M$ to **2,797,610**.

Following are the Hyper-parameters used:

- N: Residual Layers: 4
- B_i: Residual blocks in Residual Layer i: $[2, 2, 2, 2]$
- C_i: channels in Residual Layer i: $[32, 64, 128, 256]$
- F_i: Conv. kernel size in Residual Layer i: $[3, 3, 3, 3]$
- K_i: Skip connection kernel size in Residual Layer i: $[3, 3, 3, 3]$
- P: Average pool kernel size: 4

The Test accuracy after 200 epochs for different Batch size and Optimizer is shown in figure 8 wherein Optimizer = *SGD + Momentum* with batch size = 32 helped model to obtain Best accuracy of **92.370%**.

3.1.3 ResNet-18 with C_1 = 32 and 3 blocks per layer

As can be seen in summary information 9, the architecture resembles the one for the ResNet-18 model with C_1 = **32** and each Residual layer containing 3 blocks instead of 2. This allowed us to exploit the budget and get closer to $5M$ mark so as to allow maximum number of trainable parameters reaching to **4,366,250**.

Following are the Hyper-parameters used:

- N: Residual Layers: 4
- B_i: Residual blocks in Residual Layer i: $[3, 3, 3, 3]$
- C_i: channels in Residual Layer i: $[32, 64, 128, 256]$
- F_i: Conv. kernel size in Residual Layer i: $[3, 3, 3, 3]$
- K_i: Skip connection kernel size in Residual Layer i: $[3, 3, 3, 3]$
- P: Average pool kernel size: 4

The Test accuracy after 200 epochs for different Batch size and Optimizer is shown in figure 8 wherein Optimizer = *SGD + Momentum* with batch size = 32 helped model to obtain Best accuracy of **92.800%**.

3.1.4 ResNet-18 with only 1 block for last 3 layer

As can be seen in summary information 11, the architecture resembles the one for the ResNet-18 model with 2^{nd}, 3^{rd} and 4^{th} Residual layer containing only 1 block instead of 2. This allowed us to reduce the number of trainable parameters from $\approx 11M$ to **4,977,226**.

Following are the Hyper-parameters used:

- N: Residual Layers: 4
- B_i: Residual blocks in Residual Layer i: $[2, 1, 1, 1]$

- C_i: channels in Residual Layer i: $[32, 64, 128, 256]$
- F_i: Conv. kernel size in Residual Layer i: $[3, 3, 3, 3]$
- K_i: Skip connection kernel size in Residual Layer i: $[3, 3, 3, 3]$
- P: Average pool kernel size: 4

The Test accuracy after 200 epochs for different Batch size and Optimizer is shown in figure 8 wherein Optimizer = *SGD + Momentum* with batch size = 128 helped model to obtain Best accuracy of **94.980%**.

Therefore, out of 6 iterations of each of the 4 models, we can clearly conclude Model 4: *ResNet-18 with only 1 block for last 3 layer* when used with Optimizer = *SGD + Momentum* with batch size = 128 yields the best accuracy of **94.980%** on the CIFAR-10 benchmark while keeping the size of ResNet model under the specified fix budget of $5M$ with only **4,977,226** trainable parameters.

3.1.5 Impact of Regularization

Further experimentation entails analysis of the impact caused after using different Regularization techniques, namely, *Dropout* and */L2 Regularization*.

Following shows the plots for Accuracy and Loss against the number of epochs for different configuration:

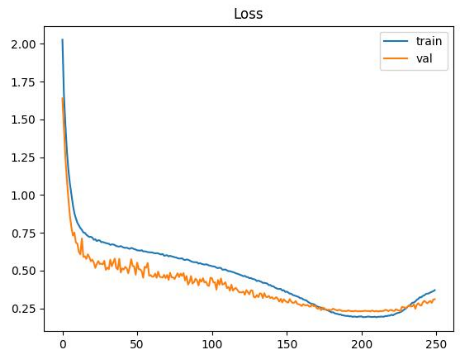


Figure 1: Dropout + L2 Reg
Test Loss: 0.229 :: Acc: 93.150%

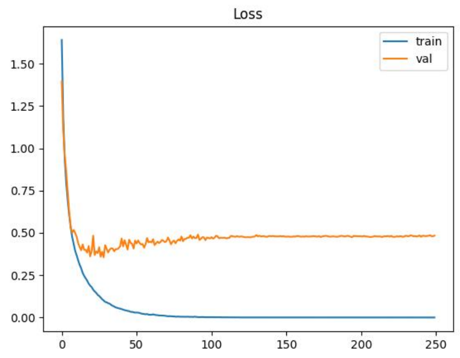


Figure 2: without Regularization
Test Loss: 0.478 :: Acc: 92.950%

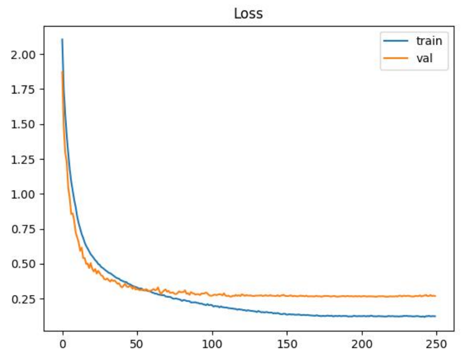


Figure 3: only Dropout
Test Loss: 0.264 :: Acc: 93.020%

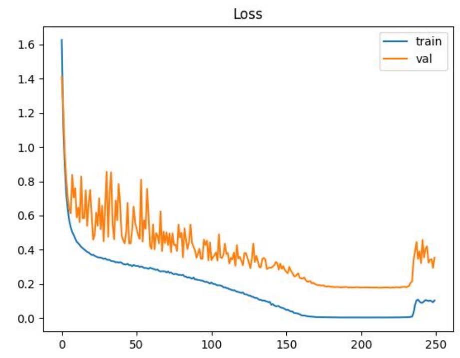


Figure 4: only L2 Regularization
Test Loss: 0.181 :: Acc: **95.140**%

4 Conclusion

Inferring through the extensive experimentation, the goal to maximize accuracy on the CIFAR-10 benchmark while keeping the size of your ResNet model under some specified fix budget can only be met when employing following configurations:

Hyper-Parameters

Following are the Hyper-parameters used:

- N: Residual Layers: 4
- B_i: Residual blocks in Residual Layer i: $[2, 1, 1, 1]$
- C_i: channels in Residual Layer i: $[32, 64, 128, 256]$
- F_i: Conv. kernel size in Residual Layer i: $[3, 3, 3, 3]$
- K_i: Skip connection kernel size in Residual Layer i: $[3, 3, 3, 3]$
- P: Average pool kernel size: 4

Data Augmentation

Chaining together the common but, fine-grained transformations:

- RandomCrop: Crop the given image at a random location.
- RandomHorizontalFlip: Horizontally flip the given image randomly with a given probability.
- Normalize: Normalize a tensor image with mean and standard deviation.

Optimizer

Stochastic Gradient Descent along with Momentum factor of 0.9 and scheduled the setting of the learning rate of each parameter group using a *cosine annealing schedule*, where η_{max} is set to the initial lr and T_{cur} is the number of epochs since the last restart in SGDR.

Regularization

L2 - Regularization using the TORCH.OPTIM package alongwith SGD optimizer, taking the **weight decay of 5e-4**.

Final Accuracy and Loss

As can be seen in the 4, the final Accuracy after epochs: 197 is 95.140% and final loss encountered is: 0.181

Available code

Please find the complete code: *DL_MiniProject_1.py* along with saved trainable parameters in *checkpoint.pt* in our GitHub Repository.

Please Note

The inspiration for the methodology of the experimentation was taken from [1].

References

[1] Akiba, T., Suzuki, S., Fukuda, K.: Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes. arXiv preprint arXiv:1711.04325 (2017)

[2] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[3] Jindal, I., Nokleby, M., Chen, X.: Learning deep networks from noisy labels with dropout regularization. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 967–972. IEEE (2016)

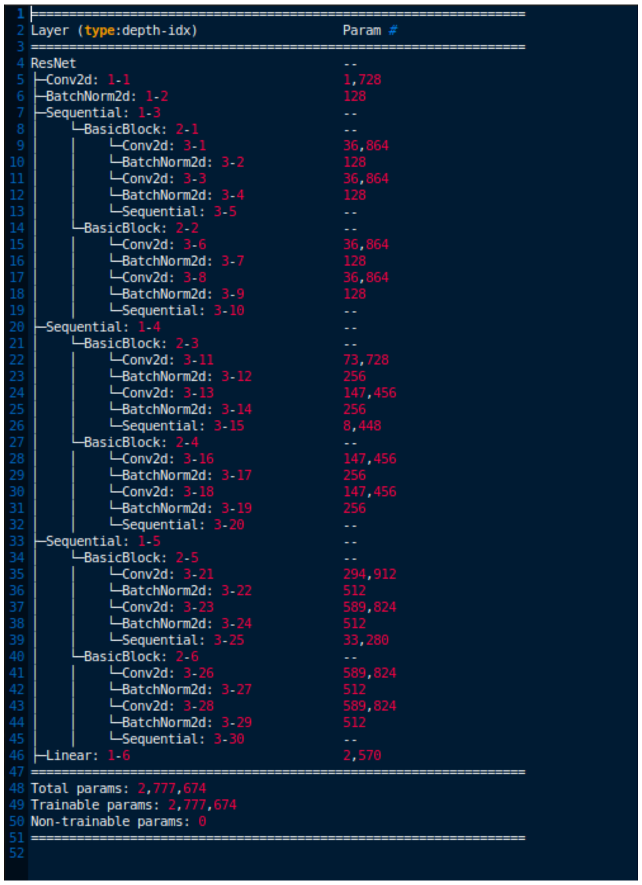
[4] Ng, A.Y.: Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In: Proceedings of the twenty-first international conference on Machine learning. p. 78 (2004)

[5] Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)

[6] Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International conference on machine learning. pp. 1139–1147. PMLR (2013)

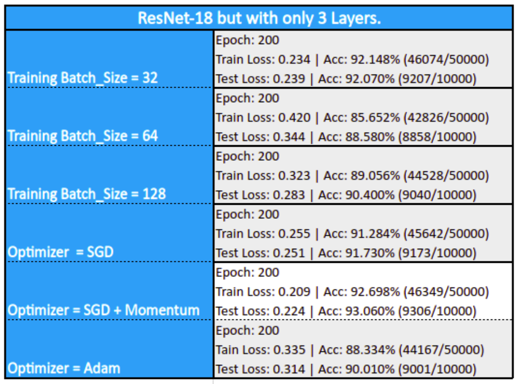
[7] Wang, S.C.: Artificial neural network. In: Interdisciplinary computing in java programming, pp. 81–100. Springer (2003)

Figures



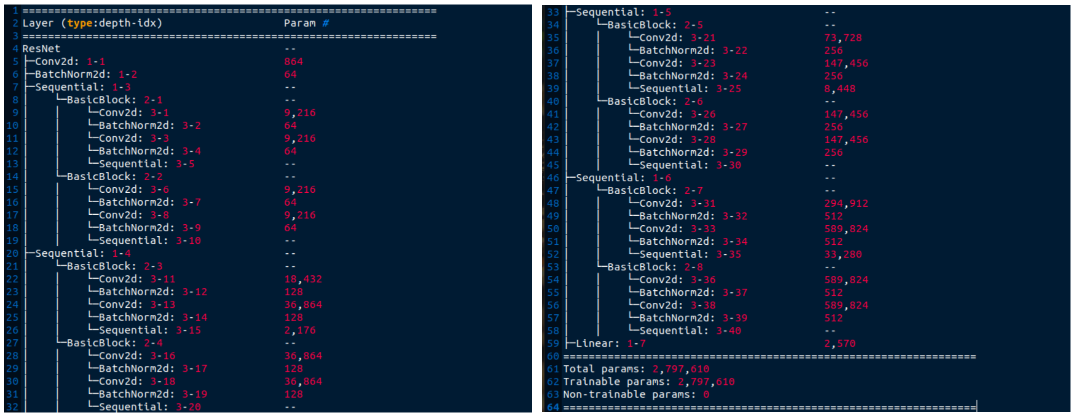
```
=================================================================
Layer (type:depth-idx)                   Param #
=================================================================
ResNet                                   --
├─Conv2d: 1-1                            1,728
├─BatchNorm2d: 1-2                       128
├─Sequential: 1-3                        --
│    └─BasicBlock: 2-1                   --
│    │    └─Conv2d: 3-1                  36,864
│    │    └─BatchNorm2d: 3-2             128
│    │    └─Conv2d: 3-3                  36,864
│    │    └─BatchNorm2d: 3-4             128
│    │    └─Sequential: 3-5              --
│    └─BasicBlock: 2-2                   --
│    │    └─Conv2d: 3-6                  36,864
│    │    └─BatchNorm2d: 3-7             128
│    │    └─Conv2d: 3-8                  36,864
│    │    └─BatchNorm2d: 3-9             128
│    │    └─Sequential: 3-10             --
├─Sequential: 1-4                        --
│    └─BasicBlock: 2-3                   --
│    │    └─Conv2d: 3-11                 73,728
│    │    └─BatchNorm2d: 3-12            256
│    │    └─Conv2d: 3-13                 147,456
│    │    └─BatchNorm2d: 3-14            256
│    │    └─Sequential: 3-15             8,448
│    └─BasicBlock: 2-4                   --
│    │    └─Conv2d: 3-16                 147,456
│    │    └─BatchNorm2d: 3-17            256
│    │    └─Conv2d: 3-18                 147,456
│    │    └─BatchNorm2d: 3-19            256
│    │    └─Sequential: 3-20             --
├─Sequential: 1-5                        --
│    └─BasicBlock: 2-5                   --
│    │    └─Conv2d: 3-21                 294,912
│    │    └─BatchNorm2d: 3-22            512
│    │    └─Conv2d: 3-23                 589,824
│    │    └─BatchNorm2d: 3-24            512
│    │    └─Sequential: 3-25             33,280
│    └─BasicBlock: 2-6                   --
│    │    └─Conv2d: 3-26                 589,824
│    │    └─BatchNorm2d: 3-27            512
│    │    └─Conv2d: 3-28                 589,824
│    │    └─BatchNorm2d: 3-29            512
│    │    └─Sequential: 3-30             --
├─Linear: 1-6                            2,570
=================================================================
Total params: 2,777,674
Trainable params: 2,777,674
Non-trainable params: 0
=================================================================
```

Figure 5: ResNet with 3 Layers and trainable parameters = 2, 777, 674



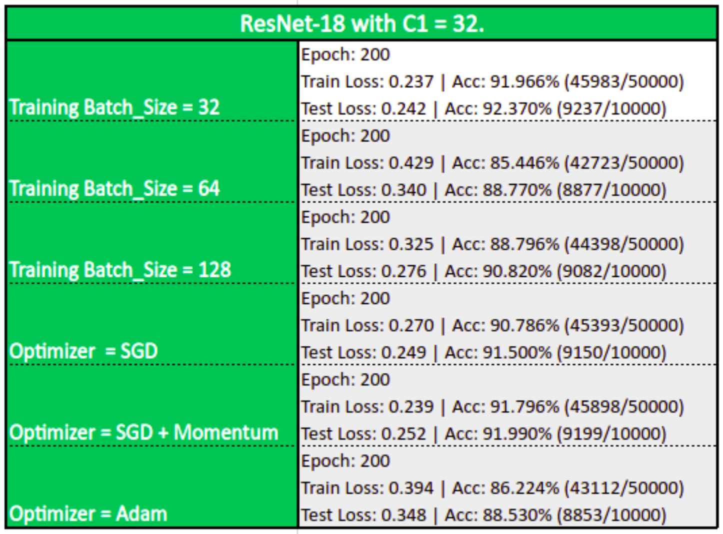
ResNet-18 but with only 3 Layers.	
Training Batch_Size = 32	Epoch: 200 Train Loss: 0.234 | Acc: 92.148% (46074/50000) Test Loss: 0.239 | Acc: 92.070% (9207/10000)
Training Batch_Size = 64	Epoch: 200 Train Loss: 0.420 | Acc: 85.652% (42826/50000) Test Loss: 0.344 | Acc: 88.580% (8858/10000)
Training Batch_Size = 128	Epoch: 200 Train Loss: 0.323 | Acc: 89.056% (44528/50000) Test Loss: 0.283 | Acc: 90.400% (9040/10000)
Optimizer = SGD	Epoch: 200 Train Loss: 0.255 | Acc: 91.284% (45642/50000) Test Loss: 0.251 | Acc: 91.730% (9173/10000)
Optimizer = SGD + Momentum	Epoch: 200 Train Loss: 0.209 | Acc: 92.698% (46349/50000) Test Loss: 0.224 | Acc: 93.060% (9306/10000)
Optimizer = Adam	Epoch: 200 Tain Loss: 0.335 | Acc: 88.334% (44167/50000) Test Loss: 0.314 | Acc: 90.010% (9001/10000)

Figure 6: Best accuracy of **93.060%** is for Optimizer = SGD + Momentum with batch size = 128.



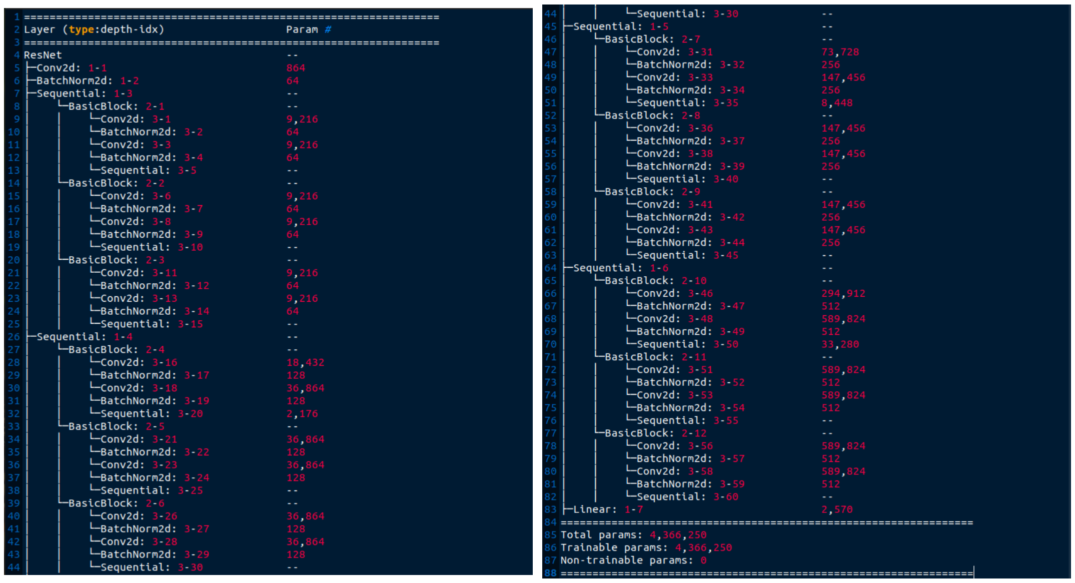
```
=================================================================
Layer (type:depth-idx)                   Param #
=================================================================
ResNet                                   --
├─Conv2d: 1-1                            864
├─BatchNorm2d: 1-2                       64
├─Sequential: 1-3                        --
│    └─BasicBlock: 2-1                   --
│    │    └─Conv2d: 3-1                  9,216
│    │    └─BatchNorm2d: 3-2             64
│    │    └─Conv2d: 3-3                  9,216
│    │    └─BatchNorm2d: 3-4             64
│    │    └─Sequential: 3-5              --
│    └─BasicBlock: 2-2                   --
│    │    └─Conv2d: 3-6                  9,216
│    │    └─BatchNorm2d: 3-7             64
│    │    └─Conv2d: 3-8                  9,216
│    │    └─BatchNorm2d: 3-9             64
│    │    └─Sequential: 3-10             --
├─Sequential: 1-4                        --
│    └─BasicBlock: 2-3                   --
│    │    └─Conv2d: 3-11                 18,432
│    │    └─BatchNorm2d: 3-12            128
│    │    └─Conv2d: 3-13                 36,864
│    │    └─BatchNorm2d: 3-14            128
│    │    └─Sequential: 3-15             2,176
│    └─BasicBlock: 2-4                   --
│    │    └─Conv2d: 3-16                 36,864
│    │    └─BatchNorm2d: 3-17            128
│    │    └─Conv2d: 3-18                 36,864
│    │    └─BatchNorm2d: 3-19            128
│    │    └─Sequential: 3-20             --
├─Sequential: 1-5                        --
│    └─BasicBlock: 2-5                   --
│    │    └─Conv2d: 3-21                 73,728
│    │    └─BatchNorm2d: 3-22            256
│    │    └─Conv2d: 3-23                 147,456
│    │    └─BatchNorm2d: 3-24            256
│    │    └─Sequential: 3-25             8,448
│    └─BasicBlock: 2-6                   --
│    │    └─Conv2d: 3-26                 147,456
│    │    └─BatchNorm2d: 3-27            256
│    │    └─Conv2d: 3-28                 147,456
│    │    └─BatchNorm2d: 3-29            256
│    │    └─Sequential: 3-30             --
├─Sequential: 1-6                        --
│    └─BasicBlock: 2-7                   --
│    │    └─Conv2d: 3-31                 294,912
│    │    └─BatchNorm2d: 3-32            512
│    │    └─Conv2d: 3-33                 589,824
│    │    └─BatchNorm2d: 3-34            512
│    │    └─Sequential: 3-35             33,280
│    └─BasicBlock: 2-8                   --
│    │    └─Conv2d: 3-36                 589,824
│    │    └─BatchNorm2d: 3-37            512
│    │    └─Conv2d: 3-38                 589,824
│    │    └─BatchNorm2d: 3-39            512
│    │    └─Sequential: 3-40             --
├─Linear: 1-7                            2,570
=================================================================
Total params: 2,797,610
Trainable params: 2,797,610
Non-trainable params: 0
=================================================================
```

Figure 7: ResNet with $C_1 = 32$ and trainable parameters = $2,797,610$



ResNet-18 with C1 = 32.	
Training Batch_Size = 32	Epoch: 200 Train Loss: 0.237 | Acc: 91.966% (45983/50000) Test Loss: 0.242 | Acc: 92.370% (9237/10000)
Training Batch_Size = 64	Epoch: 200 Train Loss: 0.429 | Acc: 85.446% (42723/50000) Test Loss: 0.340 | Acc: 88.770% (8877/10000)
Training Batch_Size = 128	Epoch: 200 Train Loss: 0.325 | Acc: 88.796% (44398/50000) Test Loss: 0.276 | Acc: 90.820% (9082/10000)
Optimizer = SGD	Epoch: 200 Train Loss: 0.270 | Acc: 90.786% (45393/50000) Test Loss: 0.249 | Acc: 91.500% (9150/10000)
Optimizer = SGD + Momentum	Epoch: 200 Train Loss: 0.239 | Acc: 91.796% (45898/50000) Test Loss: 0.252 | Acc: 91.990% (9199/10000)
Optimizer = Adam	Epoch: 200 Train Loss: 0.394 | Acc: 86.224% (43112/50000) Test Loss: 0.348 | Acc: 88.530% (8853/10000)

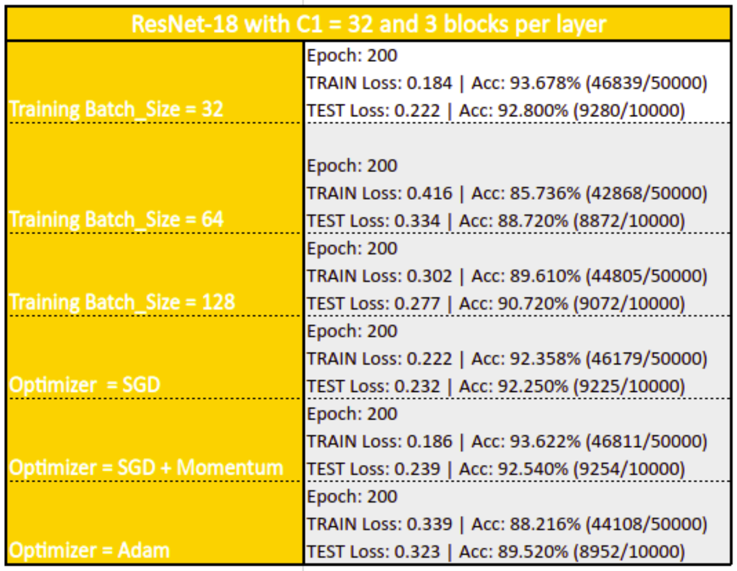
Figure 8: Best accuracy of **92.370%** is for Optimizer = SGD + Momentum with batch size = 32.



```
==================================================================
Layer (type:depth-idx)                            Param #
==================================================================
ResNet                                            --
├─Conv2d: 1-1                                     864
├─BatchNorm2d: 1-2                                64
├─Sequential: 1-3                                 --
│    └─BasicBlock: 2-1                            --
│    │    └─Conv2d: 3-1                           9,216
│    │    └─BatchNorm2d: 3-2                      64
│    │    └─Conv2d: 3-3                           9,216
│    │    └─BatchNorm2d: 3-4                      64
│    │    └─Sequential: 3-5                       --
│    └─BasicBlock: 2-2                            --
│    │    └─Conv2d: 3-6                           9,216
│    │    └─BatchNorm2d: 3-7                      64
│    │    └─Conv2d: 3-8                           9,216
│    │    └─BatchNorm2d: 3-9                      64
│    │    └─Sequential: 3-10                      --
│    └─BasicBlock: 2-3                            --
│    │    └─Conv2d: 3-11                          9,216
│    │    └─BatchNorm2d: 3-12                     64
│    │    └─Conv2d: 3-13                          9,216
│    │    └─BatchNorm2d: 3-14                     64
│    │    └─Sequential: 3-15                      --
├─Sequential: 1-4                                 --
│    └─BasicBlock: 2-4                            --
│    │    └─Conv2d: 3-16                          18,432
│    │    └─BatchNorm2d: 3-17                     128
│    │    └─Conv2d: 3-18                          36,864
│    │    └─BatchNorm2d: 3-19                     128
│    │    └─Sequential: 3-20                      2,176
│    └─BasicBlock: 2-5                            --
│    │    └─Conv2d: 3-21                          36,864
│    │    └─BatchNorm2d: 3-22                     128
│    │    └─Conv2d: 3-23                          36,864
│    │    └─BatchNorm2d: 3-24                     128
│    │    └─Sequential: 3-25                      --
│    └─BasicBlock: 2-6                            --
│    │    └─Conv2d: 3-26                          36,864
│    │    └─BatchNorm2d: 3-27                     128
│    │    └─Conv2d: 3-28                          36,864
│    │    └─BatchNorm2d: 3-29                     128
│    │    └─Sequential: 3-30                      --
```

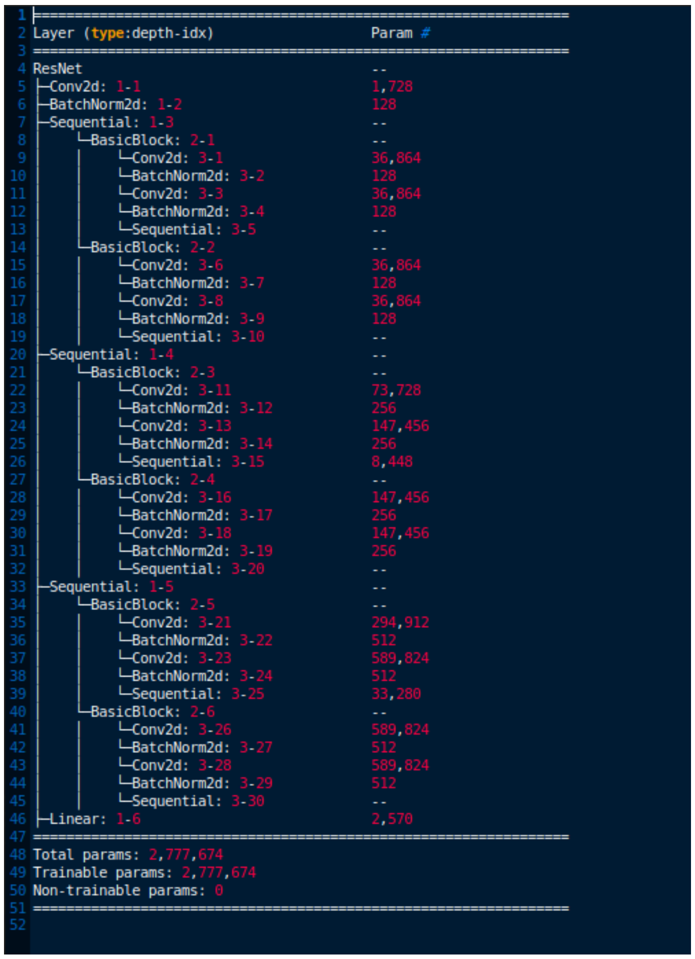
```
│    │    └─Sequential: 3-30                      --
├─Sequential: 1-5                                 --
│    └─BasicBlock: 2-7                            --
│    │    └─Conv2d: 3-31                          73,728
│    │    └─BatchNorm2d: 3-32                     256
│    │    └─Conv2d: 3-33                          147,456
│    │    └─BatchNorm2d: 3-34                     256
│    │    └─Sequential: 3-35                      8,448
│    └─BasicBlock: 2-8                            --
│    │    └─Conv2d: 3-36                          147,456
│    │    └─BatchNorm2d: 3-37                     256
│    │    └─Conv2d: 3-38                          147,456
│    │    └─BatchNorm2d: 3-39                     256
│    │    └─Sequential: 3-40                      --
│    └─BasicBlock: 2-9                            --
│    │    └─Conv2d: 3-41                          147,456
│    │    └─BatchNorm2d: 3-42                     256
│    │    └─Conv2d: 3-43                          147,456
│    │    └─BatchNorm2d: 3-44                     256
│    │    └─Sequential: 3-45                      --
├─Sequential: 1-6                                 --
│    └─BasicBlock: 2-10                           --
│    │    └─Conv2d: 3-46                          294,912
│    │    └─BatchNorm2d: 3-47                     512
│    │    └─Conv2d: 3-48                          589,824
│    │    └─BatchNorm2d: 3-49                     512
│    │    └─Sequential: 3-50                      33,280
│    └─BasicBlock: 2-11                           --
│    │    └─Conv2d: 3-51                          589,824
│    │    └─BatchNorm2d: 3-52                     512
│    │    └─Conv2d: 3-53                          589,824
│    │    └─BatchNorm2d: 3-54                     512
│    │    └─Sequential: 3-55                      --
│    └─BasicBlock: 2-12                           --
│    │    └─Conv2d: 3-56                          589,824
│    │    └─BatchNorm2d: 3-57                     512
│    │    └─Conv2d: 3-58                          589,824
│    │    └─BatchNorm2d: 3-59                     512
│    │    └─Sequential: 3-60                      --
├─Linear: 1-7                                     2,570
==================================================================
Total params: 4,366,250
Trainable params: 4,366,250
Non-trainable params: 0
==================================================================
```

Figure 9: ResNet with $C_1 = 32$ and 3 block per Residual layer and trainable parameters = $4,366,250$



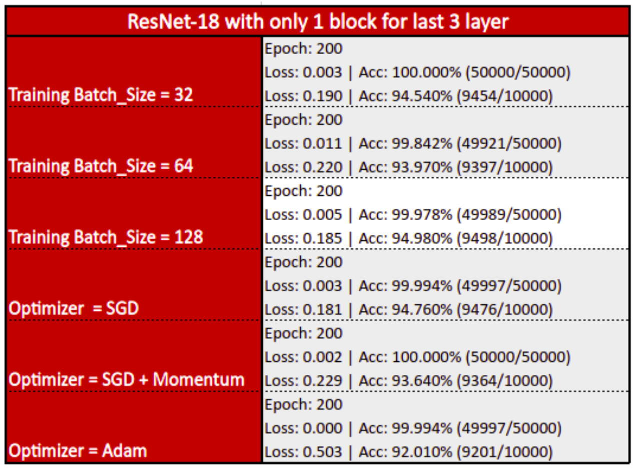
ResNet-18 with C1 = 32 and 3 blocks per layer	
Training Batch_Size = 32	Epoch: 200 TRAIN Loss: 0.184 | Acc: 93.678% (46839/50000) TEST Loss: 0.222 | Acc: 92.800% (9280/10000)
Training Batch_Size = 64	Epoch: 200 TRAIN Loss: 0.416 | Acc: 85.736% (42868/50000) TEST Loss: 0.334 | Acc: 88.720% (8872/10000)
Training Batch_Size = 128	Epoch: 200 TRAIN Loss: 0.302 | Acc: 89.610% (44805/50000) TEST Loss: 0.277 | Acc: 90.720% (9072/10000)
Optimizer = SGD	Epoch: 200 TRAIN Loss: 0.222 | Acc: 92.358% (46179/50000) TEST Loss: 0.232 | Acc: 92.250% (9225/10000)
Optimizer = SGD + Momentum	Epoch: 200 TRAIN Loss: 0.186 | Acc: 93.622% (46811/50000) TEST Loss: 0.239 | Acc: 92.540% (9254/10000)
Optimizer = Adam	Epoch: 200 TRAIN Loss: 0.339 | Acc: 88.216% (44108/50000) TEST Loss: 0.323 | Acc: 89.520% (8952/10000)

Figure 10: Best accuracy of **92.800%** is for Optimizer = SGD + Momentum with batch size = 32.



```
================================================================
Layer (type:depth-idx)                   Param #
================================================================
ResNet                                   --
├─Conv2d: 1-1                            1,728
├─BatchNorm2d: 1-2                       128
├─Sequential: 1-3                        --
│    └─BasicBlock: 2-1                   --
│    │    └─Conv2d: 3-1                  36,864
│    │    └─BatchNorm2d: 3-2             128
│    │    └─Conv2d: 3-3                  36,864
│    │    └─BatchNorm2d: 3-4             128
│    │    └─Sequential: 3-5              --
│    └─BasicBlock: 2-2                   --
│    │    └─Conv2d: 3-6                  36,864
│    │    └─BatchNorm2d: 3-7             128
│    │    └─Conv2d: 3-8                  36,864
│    │    └─BatchNorm2d: 3-9             128
│    │    └─Sequential: 3-10             --
├─Sequential: 1-4                        --
│    └─BasicBlock: 2-3                   --
│    │    └─Conv2d: 3-11                 73,728
│    │    └─BatchNorm2d: 3-12            256
│    │    └─Conv2d: 3-13                 147,456
│    │    └─BatchNorm2d: 3-14            256
│    │    └─Sequential: 3-15             8,448
│    └─BasicBlock: 2-4                   --
│    │    └─Conv2d: 3-16                 147,456
│    │    └─BatchNorm2d: 3-17            256
│    │    └─Conv2d: 3-18                 147,456
│    │    └─BatchNorm2d: 3-19            256
│    │    └─Sequential: 3-20             --
├─Sequential: 1-5                        --
│    └─BasicBlock: 2-5                   --
│    │    └─Conv2d: 3-21                 294,912
│    │    └─BatchNorm2d: 3-22            512
│    │    └─Conv2d: 3-23                 589,824
│    │    └─BatchNorm2d: 3-24            512
│    │    └─Sequential: 3-25             33,280
│    └─BasicBlock: 2-6                   --
│    │    └─Conv2d: 3-26                 589,824
│    │    └─BatchNorm2d: 3-27            512
│    │    └─Conv2d: 3-28                 589,824
│    │    └─BatchNorm2d: 3-29            512
│    │    └─Sequential: 3-30             --
├─Linear: 1-6                            2,570
================================================================
Total params: 2,777,674
Trainable params: 2,777,674
Non-trainable params: 0
================================================================
```

Figure 11: ResNet with only 1 block for last 3 Residual layer and trainable parameters = $4,977,226$



ResNet-18 with only 1 block for last 3 layer	
Training Batch_Size = 32	Epoch: 200 Loss: 0.003 | Acc: 100.000% (50000/50000) Loss: 0.190 | Acc: 94.540% (9454/10000)
Training Batch_Size = 64	Epoch: 200 Loss: 0.011 | Acc: 99.842% (49921/50000) Loss: 0.220 | Acc: 93.970% (9397/10000)
Training Batch_Size = 128	Epoch: 200 Loss: 0.005 | Acc: 99.978% (49989/50000) Loss: 0.185 | Acc: 94.980% (9498/10000)
Optimizer = SGD	Epoch: 200 Loss: 0.003 | Acc: 99.994% (49997/50000) Loss: 0.181 | Acc: 94.760% (9476/10000)
Optimizer = SGD + Momentum	Epoch: 200 Loss: 0.002 | Acc: 100.000% (50000/50000) Loss: 0.229 | Acc: 93.640% (9364/10000)
Optimizer = Adam	Epoch: 200 Loss: 0.000 | Acc: 99.994% (49997/50000) Loss: 0.503 | Acc: 92.010% (9201/10000)

Figure 12: Best accuracy of **94.980%** is for Optimizer = SGD + Momentum with batch size = 128.