Optimizing CIFAR10 Image Classification with Modified ResNet Architecture under Parameter Constraints

Arushi Arora[1*], **Saaketh Koundinya**[1*], **Shreya Agarwal**[1*]

[1]Department of Electrical and Computer Engineering, New York University
aa10350@nyu.edu, sg7729@nyu.edu, sa6981@nyu.edu

Abstract

We present a modified ResNet architecture which achieves an accuracy of 93.5% on the CIFAR10 image classification dataset while constraining the model to have no more than 5 million parameters. Our approach involved using $2, 1, 1, 1$ ResNet basic blocks instead of the standard $2, 2, 2, 2$ blocks used in ResNet18 (He et al. 2016a). Additionally, convolution filters of sizes $64, 128, 256, 512$ were used for each basic block respectively.

The ResNet architecture has shown exceptional performance on various image classification tasks, and the proposed modifications aim to further improve its performance on the CIFAR10 dataset. By constraining the number of parameters to 5 million, we aimed to develop a model that is efficient in terms of both memory and computation requirements.

The code is available at the following link.

Introduction

Convolutional Neural Networks (CNNs) have shown remarkable performance in image classification tasks. Among them, Residual Networks (ResNets) have achieved state-of-the-art performance in various image classification benchmarks, including CIFAR10 (a widely used benchmark dataset for image classification tasks, consisting of 50,000 training images and 10,000 test images). However, increasing the depth and width of ResNets results in a large number of model parameters, which increases the computational cost and memory requirements. Therefore, it is essential to explore methods to improve the performance of ResNets while keeping the model size low.

We came up with a modified ResNet architecture that has comparable test accuracy on the CIFAR10 dataset while having no more than 5 million parameters. Our solution proposes using $2, 1, 1, 1$ ResNet basic blocks instead of the standard $2, 2, 2, 2$ configuration and using $64, 128, 256, 512$ convolutional filters for each basic block, respectively.

Our aim is to demonstrate that a modified ResNet architecture with fewer parameters can achieve competitive performance on the CIFAR10 dataset. We evaluate our proposed architecture and compare it to the original ResNet architecture with 11 million parameters.

[*]These authors contributed equally.

Our Contributions

- Proposed a modified ResNet architecture with improved performance on the CIFAR10 dataset while adhering to a parameter constraint of no more than 5 million parameters.
- Investigated the effects of different learning rate and optimizers, such as Adam, AdaDelta and SGD, on the performance of ResNet.

Related Work

The ResNet architecture was first introduced in Deep Residual Learning for Image Recognition (He et al. 2016a), which showed significant improvement in image classification tasks compared to previous deep neural network architectures. Subsequent (He et al. 2016b), (Huang et al. 2018), (Loshchilov and Hutter 2017), and (Zhang et al. 2019) have explored modifications to the ResNet architecture, including varying the depth and width of the network, introducing additional shortcut connections, and using different activation functions. Several studies have explored different deep learning architectures for CIFAR-10 classification, such as VGG16 introduced in (Simonyan and Zisserman 2015), and DenseNet introduced in (Huang et al. 2017) .

Hyperparameter optimization is an important aspect of deep learning, as it can significantly impact the performance of the model. In the field of deep learning, optimizing the performance of neural networks is an active area of research. Various techniques such as learning rate scheduling, early stopping, use of different optimizers, and weight initialization have been commonly used to improve the performance of deep neural networks, as discussed in (Bergstra and Bengio 2012), (Bergstra et al. 2011) and, (Smith 2017).

Experiments

We conducted experiments on various ResNet model architectures and explored the effects of different learning rates, optimizers, and schedulers.

Our experimentation involved five distinct models, namely ResNet-10, ResNet-12, ResNet-14-4, ResNet-14-5, and ResNet-18. ResNet-10 comprises 8 BasicBlocks with a configuration of $2, 1, 1, 1$ with $64, 128, 256, 512$ channels respectively, ResNet12 utilizes 6 BasicBlocks with a configuration of $2, 2, 2$ with $16, 32, 64$ channels respectively,

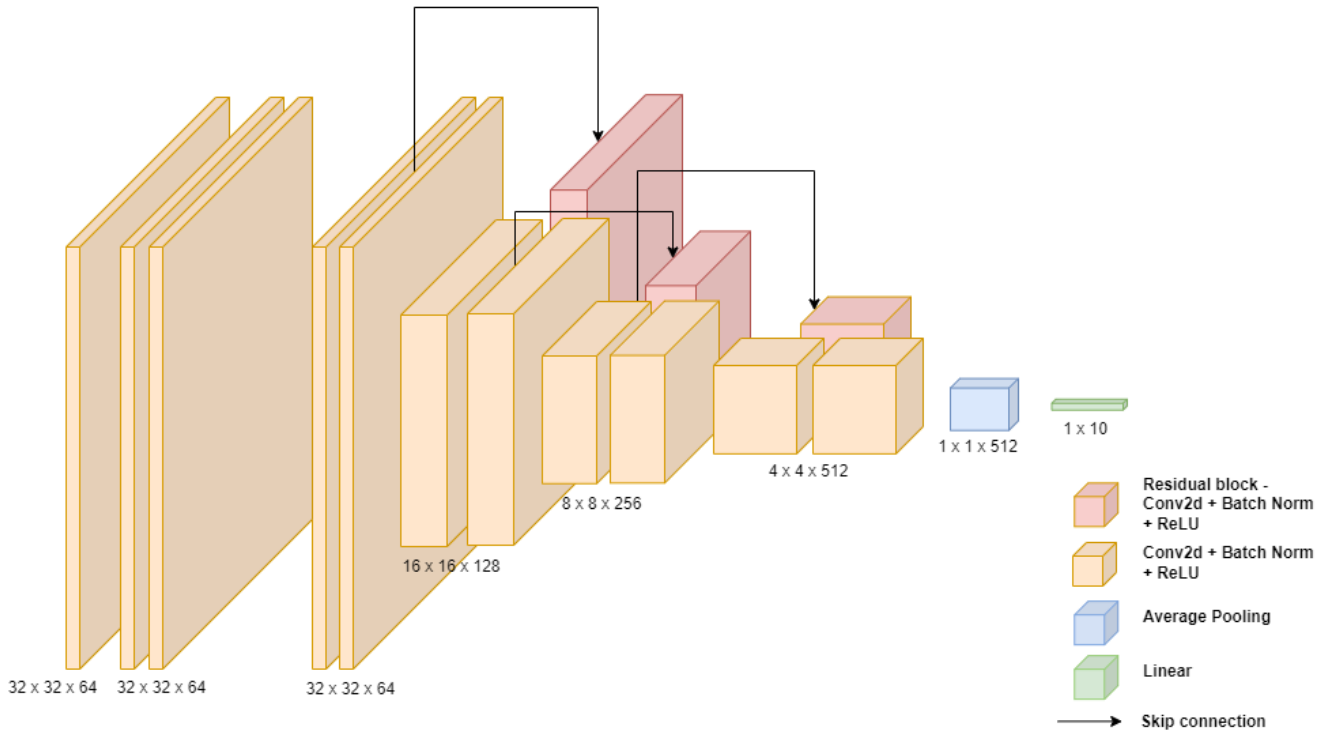

Figure 1: Model Architecture

ResNet14-4 incorporates 7 BasicBlocks with a configuration of $2, 2, 2, 1$ with $16, 32, 64, 512$ channels respectively, ResNet14-5 employs 10 BasicBlock with a configuration of $2, 3, 3, 1, 1$ with $16, 32, 64, 128, 512$ channels respectively, and the standard ResNet18 architecture uses 8 BasicBlock with a configuration of $2, 2, 2, 2$ with $64, 128, 256, 512$ channels respectively.

Using grid search, as introduced in (Bergstra and Bengio 2012), hyperparameters were tuned, and configurations shown in Table 1 were employed in the grid search.

Parameter	**Values**
Learning rate	$0.1, 0.01, 0.001$
Optimizers	Adam, SGD, AdaDelta

Table 1: Grid search configuration

Furthermore, we experimented with two different filter sizes of 3 and 5 and implemented Kaimings Normal initialization as shown in (He et al. 2015) for convolutional layers and Xavier Normal, as introducted in (Glorot and Bengio 2010), for linear layers. The Table 2 summarizes the model architectures used for the experiments.

Methodology

Data Pre-Processing

We utilized the CIFAR-10 dataset comprising 60000 color images of size 32 x 32 pixels distributed over 10 classes, each class consisting of 6000 images.

Model Architecture	**# of Blocks**	**# of Out Channels**
ResNet-10	$[2, 1, 1, 1]$	$[64, 128, 256, 512]$
ResNet-12	$[2, 2, 2]$	$[16, 32, 64]$
ResNet-14-4	$[2, 2, 2, 1]$	$[16, 32, 64, 512]$
ResNet-14-5	$[2, 3, 3, 1, 1]$	$[16, 32, 64, 128, 512]$
ResNet-18	$[2, 2, 2, 2]$	$[64, 128, 256, 512]$

Table 2: Experiments with different Model Architectures

The dataset is split into 50000 training images and 10000 test images. Prior to training the model, we normalized both the training and test images.

To increase the diversity of images seen by the model during training and improve its generalization ability, we applied several image augmentations to the training data, including random rotations between -5 and 5 degrees, random horizontal flipping with a 50% probability, and random cropping of 32 x 32 patches from the image with a 2-pixel padding on each side.

During training, we split the training data into a $90 : 10$ train-validation ratio and saved the model at each epoch when the validation accuracy improved.

Design Approach

At the outset, we utilized the ResNet-18 model architecture which comprises of four layers, with each layer consisting of two blocks. The channel sizes of the blocks were set at 64, 128, 256, and 512, respectively, leading to a total of 11.17M parameters.

Model Architecture	**Number of Parameters**	**Optimizer**	**Learning Rate**	**Scheduler**	**Test Accuracy**
ResNet-10	$4.98M$	SGD	0.1	LR decay	91.45%
ResNet-10	**4.98M**	**SGD**	**0.001**	**OneCycleLR**	$\mathbf{93.54}\%$
ResNet-12	$0.19M$	SGD	0.1	LR decay	89.03%
ResNet-14-4	$3.15M$	Adam	0.01	LR decay	89.74%
ResNet-14-5	$4.13M$	SGD	0.1	LR decay	89.45%
ResNet-18	$11.17M$	Adam	0.001	LR decay	92.98%

Table 3: Model Comparison

To enhance the performance of the model, we made alterations to the architecture by adjusting the number of residual blocks and channels in each layer. Specifically, we sought to reduce the number of parameters in the model by modifying the architecture.

To estimate the parameter count of the modified architecture, we utilized the formula given in Equation 1

$$p = ((m * n * d) + 1) * k) \tag{1}$$

where m=width of the filter, n=height of the filter, d=number of filters in the previous layer, k=number of filters

Overall, our modifications aimed to reduce the number of parameters while maintaining high model performance. This would enable the model to be trained more efficiently and achieve better accuracy.

Implementation

The proposed model architecture is a variant of ResNet-18 that consists of several blocks, each block containing one or more layers of convolutional neural network (CNN) operations. The architecture utilizes BasicBlock as the building block of the network.

The BasicBlock contains two convolutional layers, each followed by a batch normalization operation and a rectified linear unit (ReLU) activation function. The BasicBlock also utilizes skip connections to improve the flow of gradients during backpropagation. If the input and output of the block are of different sizes, then the skip connection utilizes a 1x1 convolution followed by batch normalization to adjust the dimensionality of the input to match the output.

ResNet-10 has 10 convolutional layers, with 4 blocks having feature maps of sizes $64, 128, 256, 512$ respectively. The first layer consists of 2 blocks and the subsequent layers contains 1 block each. ResNet-12 has 12 convolutional layers, with 3 blocks having feature maps of sizes $16, 32, 64$ respectively. Each layer contains 2 blocks. ResNet14-4 has 14 convolutional layers, with 4 blocks having feature maps of sizes $16, 32, 64, 512$ respectively. The first 3 layer contains 2 blocks and the last layer contains 1 block. ResNet14-5 has 14 convolutional layers, with 5 blocks having feature maps of sizes $16, 32, 64, 128, 512$ respectively. The first layer contains 2 blocks, the next 2 layers contain 3 blocks each and the last 2 layers contain 1 block each.

The input to the network is a 3-channel image with dimensions 32 x 32. The first layer of the network is a convolutional layer with 64 output channels, kernel size 3x3, and padding of 1. The output of the first layer is passed through a batch normalization operation and a ReLU activation function before being passed through the ResNet models.

The output of the last layer of blocks is passed through an average pooling layer with a kernel size of 4 x 4 to reduce the feature map size to 1 x 1. The resulting feature map is flattened and passed through a linear layer with 10 output units, corresponding to the number of classes in the dataset.

The weights of the convolutional and linear layers are initialized using Xavier and Kaiming initialization, respectively, while the biases are initialized to zero. The activation function used throughout the network is ReLU.

Analysis of Model architectures

ResNet-10: This ResNet architecture has shown highest accuracy of 93.5% on CIFAR-10 dataset, which is close to the state-of-the-art performance on this dataset. This architecture with 5M parameter constraints performs better on this specific task compared to other architectures with more parameters. What makes it better is the fact that it has a simpler structure compared to deeper architectures, making it easier to implement and train. However, this architecture requires additional tuning and experimentation to find the optimal hyperparameters and training strategy.

ResNet-12: Low number of parameters (1M) makes the model lightweight and efficient. Simple architecture with only 3 blocks of BasicBlocks, makes it easy to understand and modify. The demerit of this architecture is that accuracy of 88% is not state-of-the-art, and other models may achieve higher accuracy.

ResNet-14-4: This architecture has a moderate number of parameters (3.1M), which strikes a balance between model complexity and accuracy. However, the Adam optimizer has been observed to sometimes overshoot the minima, leading to poor generalization and overfitting in this case.

ResNet-14-5: The inclusion of more BasicBlocks with increasing number of channels allows the model to learn more complex representations of the data.

ResNet-18: While ResNet-18 is a highly effective architecture for image classification, and it has achieved state-of-the-art performance on various benchmarks, it has relatively large number of parameters (11.7M), which makes it less suitable for deployment on resource-constrained devices.

Results and discussion

The ResNet-10 architecture comprising a BasicBlock of [2,1,1,1] with 64, 128, 256, and 512 channels with 4.9M parameters and SGD optimizer with OnecycleLR achieves the highest test accuracy of 93.54%, despite having fewer parameters than ResNet-18. ResNet-12 had the fewest parameters, but a lower test accuracy of 89.03%.

Overall, the choice of architecture, number of parameters, optimizer, and learning rate all play important roles in the performance of deep learning models. In this study, ResNet-10 outperformed the other models on the task at hand. However, the choice of architecture ultimately depends on the specific requirements of the task, and other architectures may be more suitable for different tasks or datasets.

Conclusion

In conclusion, the goal of this study was to create a ResNet model with less than 5 million parameters and achieve the highest accuracy. We experimented with different model architectures, number of parameters, optimizers, learning rate schedulers, and evaluated their performance based on test accuracy. Our results showed that it is indeed possible to create compact ResNet models that outperform larger ones in terms of accuracy. Among the models we tested, the ResNet10 with a learning rate of 0.001 and OneCycleLR scheduler achieved the highest test accuracy of 93.54%, while only using 4.98 million parameters. We also observed that the choice of optimizer and learning rate scheduler can have a significant impact on model performance.

Our findings demonstrate the effectiveness of our approach to parameter reduction in deep learning models, which can have important implications for real-world applications where resource constraints are a significant concern. By reducing the number of parameters while maintaining high accuracy, we can create more efficient models that require less computation and memory resources. Overall, our study provides insights into the design of compact ResNet models and highlights the importance of careful selection of hyperparameters for achieving optimal performance.

References

Bergstra, J.; Bardenet, R.; Bengio, Y.; and Kégl, B. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2546–2554.

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb): 281–305.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 630–645. Springer.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2018. Condensed nearest neighbor based on a deep residual network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2486–2494. IEEE.

Loshchilov, I.; and Hutter, F. 2017. SGDR: stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 3rd International Conference on Learning Representations*.

Smith, L. N. 2017. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and López-Paz, D. 2019. Mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Appendix

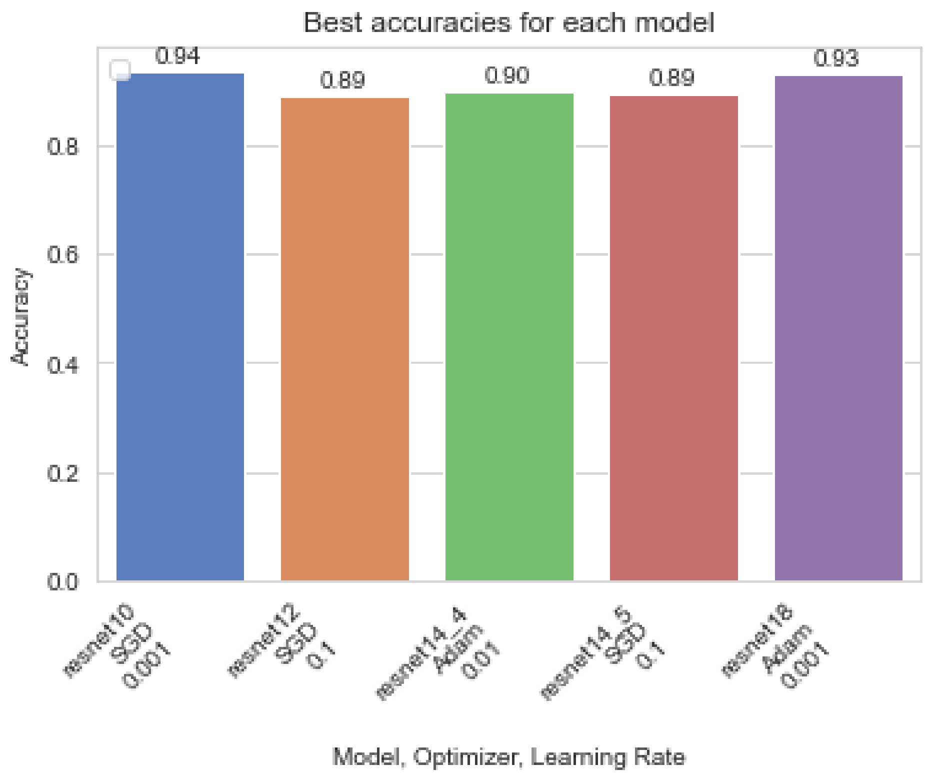

Figure 2: Comparison of model architectures on tuned hyper-parameters

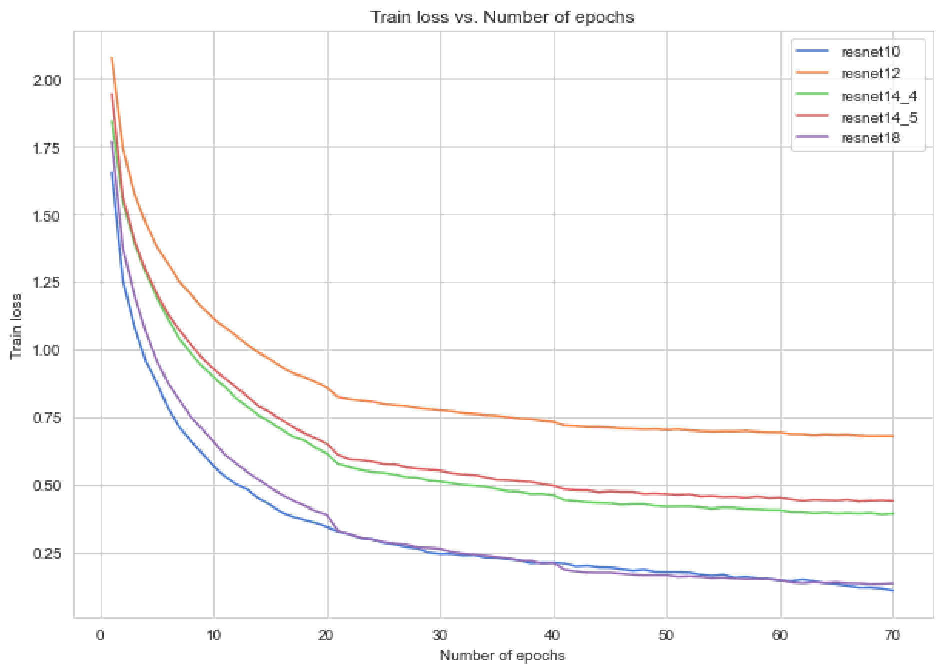

Figure 3: Comparison of model architectures: train loss vs epochs with optimal parameters

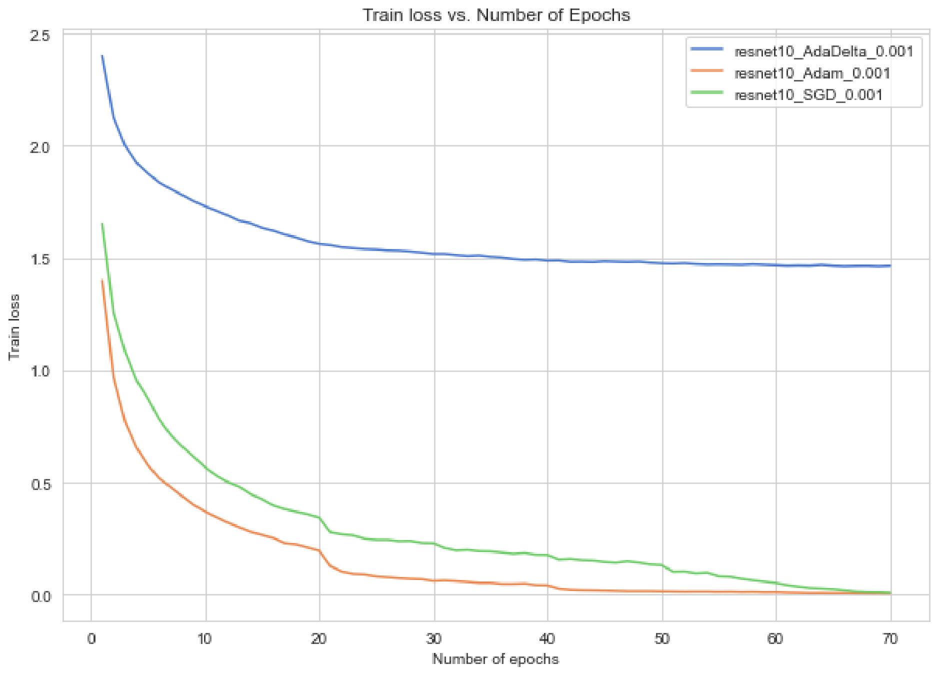

Figure 4: Training progress for best model with optimal learning rate: train loss vs epochs

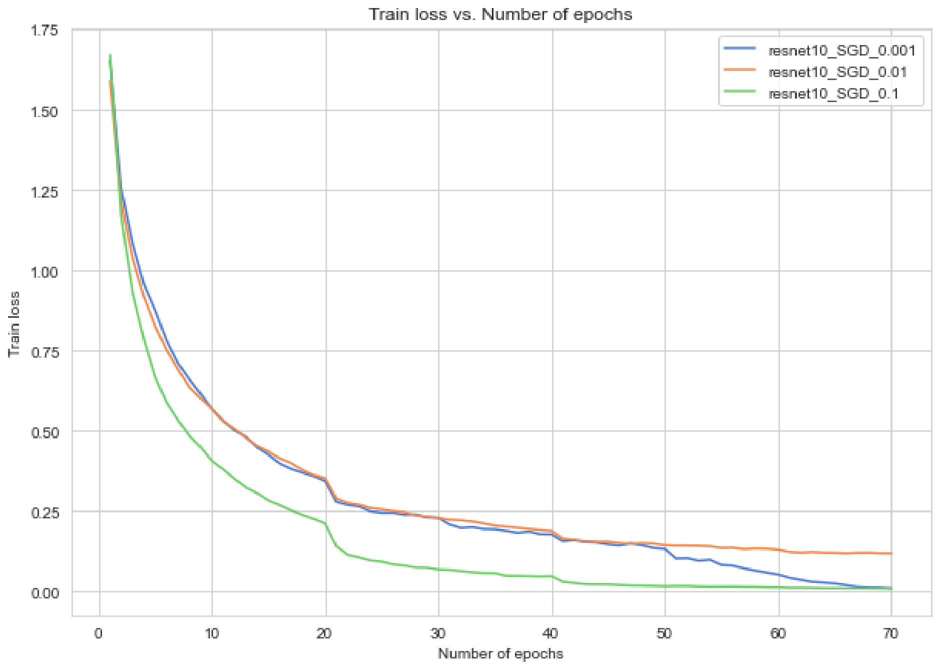

Figure 5: Training progress for best model with optimal optimizer: train loss vs epochs

Model architecture	Optimizer	Learning rate	Accuracy
resnet10	AdaDelta	0.001	0.467187
resnet10	AdaDelta	0.01	0.690234
resnet10	**SGD**	**0.001**	**0.935489**
resnet10	AdaDelta	0.1	0.865917
resnet10	SGD	0.01	0.876367
resnet10	Adam	0.1	0.890625
resnet10	Adam	0.001	0.742968
resnet10	Adam	0.01	0.914355
resnet10	SGD	0.1	0.914550
resnet12	AdaDelta	0.001	0.379785
resnet12	AdaDelta	0.01	0.607519
resnet12	SGD	0.001	0.743945
resnet12	Adam	0.1	0.829785
resnet12	AdaDelta	0.1	0.835156
resnet12	SGD	0.01	0.870898
resnet12	Adam	0.001	0.877636
resnet12	Adam	0.01	0.888476
resnet12	SGD	0.1	0.890332
resnet14_4	AdaDelta	0.001	0.463378
resnet14_4	AdaDelta	0.01	0.718066
resnet14_4	SGD	0.001	0.826855
resnet14_4	AdaDelta	0.1	0.869335
resnet14_4	Adam	0.1	0.871582
resnet14_4	Adam	0.001	0.889257
resnet14_4	SGD	0.01	0.893751
resnet14_4	SGD	0.1	0.895605
resnet14_4	Adam	0.01	0.897363
resnet14_5	AdaDelta	0.001	0.440234
resnet14_5	AdaDelta	0.01	0.679785
resnet14_5	SGD	0.001	0.815722
resnet14_5	AdaDelta	0.1	0.865429
resnet14_5	Adam	0.1	0.868066
resnet14_5	SGD	0.01	0.887207
resnet14_5	Adam	0.01	0.891503
resnet14_5	Adam	0.001	0.893652
resnet14_5	SGD	0.1	0.894531
resnet18	AdaDelta	0.001	0.574511
resnet18	AdaDelta	0.01	0.814843
resnet18	SGD	0.001	0.859375
resnet18	Adam	0.1	0.897074
resnet18	SGD	0.1	0.902832
resnet18	AdaDelta	0.1	0.908984
resnet18	SGD	0.01	0.914355
resnet18	Adam	0.01	0.919726
resnet18	Adam	0.001	0.929785

Table 4: Results of grid search for finding optimal parameters