CIFAR 10 image classification using a modified Residual Network architecture

Abhignya Bhat, Anshika Gupta , Mudassir Hussain
ayb5037 , ag8800, sh7420

Github:
https://github.com/csendranshi/DL-mini-project

Abstract

This study examines the application of residual networks in image classification tasks. Various modifications to the ResNet18 model architecture, including different optimizers, learning rates, and the number of residual blocks, will be compared to identify the optimal model. The constraint imposed on the model is that it must have fewer than 5 million parameters, and the objective is to achieve the highest possible accuracy.

Literature Review

Residual networks (ResNets) are a popular type of deep neural network architecture used in image classification tasks. They are highly effective in improving the accuracy of image classification models, especially for deep networks with many layers. ResNets innovate by incorporating residual connections, which allow for the direct skipping of certain layers in the network to mitigate the issue of vanishing gradients. Vanishing gradients occur when gradients become too small during training, making it difficult to train deep networks. Residual connections help overcome this problem, enabling effective training of deeper networks. ResNets also utilize skip connections that preserve information from earlier layers, preventing the loss of important features during training and leading to more accurate and robust feature representation. This results in improved classification performance. In addition, ResNets are known for their ability to achieve state-of-the-art accuracy with fewer parameters compared to other deep neural network architectures. This makes them computationally efficient and suitable for applications with constraints on model size or computational resources. Overall, ResNets are widely used in image classification tasks due to their ability to enhance accuracy, enable training of deeper networks, preserve important features, and achieve good performance with fewer parameters.

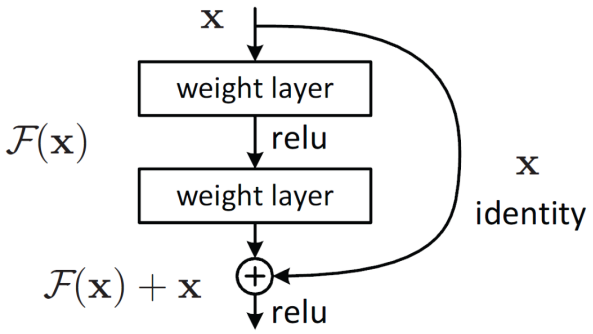

Fig1. Basic Logic of ResNet

We investigated various architectural choices such as different optimizers, learning rates, and the number of residual blocks to analyze their impact on model performance. Most of the models showed a saturation point around epoch 80, with test accuracy ranging from 89% to 91%.

Dataset

The CIFAR-10 dataset is a widely used benchmark dataset for image classification tasks in machine learning and computer vision. It consists of 60,000 color images of size 32x32 pixels, divided into 10 classes, with 6,000 images per class. The dataset is further split into a training set of 50,000 images and a test set of 10,000 images. The 10 classes in the CIFAR-10 dataset are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The images in the dataset are relatively low-resolution and represent a wide range of objects and animals commonly found in daily life.

Data Augmentation

Data augmentation is used to address the issue of limited training data, which is common in many real-world applications. By augmenting the dataset, data augmentation helps in increasing the variability and diversity of the training data, which in turn helps the model to learn more robust and generalizable features. This can improve the

model's ability to generalize well to unseen data and can reduce overfitting, leading to better model performance and higher accuracy.

In this study we have used the following data augmentation techniques:

1. Random Crop:
   Applies a random crop transformation to the input image with a size of 32x32 pixels. The input image is padded with 4 pixels on each side before cropping, to avoid information loss at the edges of the image.
2. Random Horizontal Flip
   It randomly flips the image horizontally with a probability of 0.5
3. Normalization
   Normalization helps to scale the input data to a common range and can improve the convergence and performance of the model during training. Subtracting the mean values (0.4914, 0.4822, 0.4465) from each channel (red, green, and blue) of the input image, and then dividing by the standard deviation values (0.2023, 0.1994, 0.2010)

Architecture

Resnet18 is a popular deep learning architecture that uses residual blocks, which were introduced by a research team from Microsoft. It consists of several layers, including a preprocessing convolution layer, eight residual blocks, and a linear layer. The preprocessing convolution layer extracts features from the input with 3 channels and produces 64 channels for the subsequent layers. The residual blocks are divided into four layers, with 2 blocks in each layer. The first layer acts as an identity layer to enhance the extracted features. The subsequent three layers further extract features from the outputs of the preceding layer using a stride of 2 in a single convolution layer of each block, which reduces the image size by 2x2. The final output of the residual layer is a tensor of size 512x4x4, which is then reduced to 512 features per image using 2D average pooling with a kernel size of 4. These 512 features are then fed into a linear layer to map them to 10 classes.

We made modifications to the original Resnet18 architecture for our base model to meet our constraint of having fewer than 5 million parameters, compared to the original architecture's 11 million parameters. Instead of 4 layers, we used 3 layers, with 3 residual blocks in layers 1 and 3, and 2 blocks in layer 2. Similar to the original architecture, the first layer acts as an identity layer, while the subsequent two layers use a stride of 2 in a single convolution layer of each block to reduce image size by 2x2 and extract features from the preceding layer's outputs. The final output of the residual layer is a tensor of size 256x8x8. These features are then fed into a linear layer to map them to the 10 output classes.

Methodology

The study aimed to investigate the effects of different architectural choices on model performance. Three factors were considered: choice of optimizer (SGD vs Adam), learning rate (0.005 vs 0.01), and number of residual blocks (8 blocks vs 10 blocks).

For the first set of experiments, two models were trained using the same base architecture. Model 1 used SGD as its optimizer with a learning rate of 0.005, while Model 2 used Adam as its optimizer with the same learning rate.

- Model 1:
  Architecture: Base model
  Learning Rate: 0.005
  Optimizer: SGD

- Model 2:
  Architecture: Base model
  Learning Rate: 0.005
  Optimizer: Adam

For the second set of experiments, two models were trained with the same base architecture and optimizer (SGD), but with different learning rates. Model 1 used a learning rate of 0.005, while Model 2 used a higher learning rate of 0.01.

- Model 1:
  Architecture: Base model
  Learning Rate: 0.005
  Optimizer: SGD

- Model 2:
  Architecture: Base model
  Learning Rate: 0.01
  Optimizer: SGD

For the third set of experiments, two models were trained with different numbers of residual blocks in different layers of the architecture. Model 1 used 3 blocks in layer 1, 2 blocks in layer 2, and 3 blocks in layer 3, while Model 2 used 4 blocks in layer 1, and 3 blocks in layers 2 and 3. Both models used SGD as their optimizer with a learning rate of 0.005.

- Model 1:
  Architecture: Base model (Blocks: [3,2,3])
  Learning Rate: 0.005
  Optimizer: SGD

- Model 2:
  Architecture: Base model(Blocks [4,3,3])
  Learning Rate: 0.005
  Optimizer: SGD

The effects of these different architectural choices on model performance were studied and analyzed to understand their impact on the overall performance and behavior of the models.

Results

In the first experiment where we compared different optimizers, we obtained the following results:

	Training accuracy	Training Loss	Testing Accuracy	Testing Loss
Model1 (SGD)	99.046%	0.029	90.254%	0.396
Model2 (Adam)	99.061%	0.027	91.602%	0.481

Table 1. Performance Metrics for different choice of optimizers

The accuracy and loss curves are given below

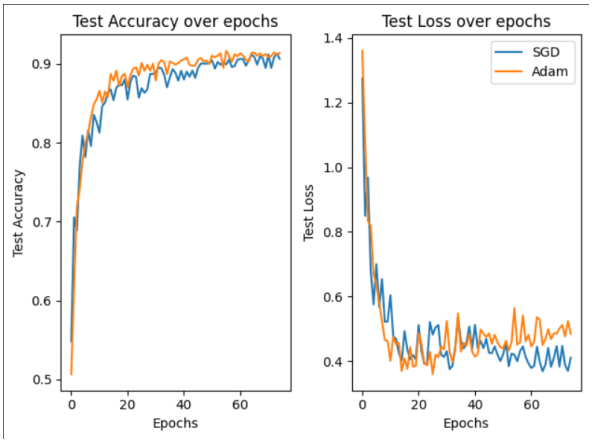

Fig2. Test accuracy and Loss curves for Model 1(SGD) Vs Model 2 (Adam)

Based on the accuracy and loss curves, it is evident that Adam slightly outperformed SGD. Although Adam had a higher testing loss, it achieved better accuracy and had a smoother accuracy curve compared to SGD. This result can be attributed to the fact that Adam optimizer performs well on larger datasets due to its adaptive learning rates and momentum properties, which result in faster convergence.

For the second experiment where we compared different learning rates, we obtained the following results
The accuracy and loss curves are given below:

	Training accuracy	Training Loss	Testing Accuracy	Testing Loss
Model1 (0.01)	98.389%	0.048	90.742%	0.336
Model2 (0.005)	99.051%	0.030	91.758%	0.362

Table 2. Performance Metrics for different choice of learning rates

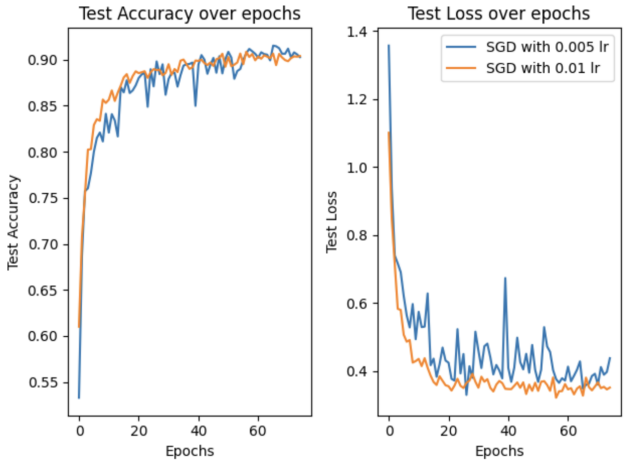

Fig3. Test accuracy and Loss curves for Model 1(lr=0.005) Vs Model 2 (lr=0.1)

The given graph shows that using a lower learning rate of 0.005 has more pronounced spikes compared to a learning rate of 0.01, which could be due to the model getting stuck in local optima or saddle points, where the loss function remains relatively flat, leading to slower convergence and more fluctuations in the loss curve. Additionally, the presence of noise in the data could make the model more sensitive to variations, resulting in more fluctuations in the loss curve.

For the third experiment where we compared different number of residual blocks, we obtained the following results

	Training accuracy	Training Loss	Testing Accuracy	Testing Loss
Model1 ([3,2,3])	99.132%	0.027	90.918%	0.418
Model2 ([4,3,3])	99.008%	0.030	91.426%	0.361

Table 3. Performance Metrics for different number of residual blocks

The accuracy and loss curves are given below,

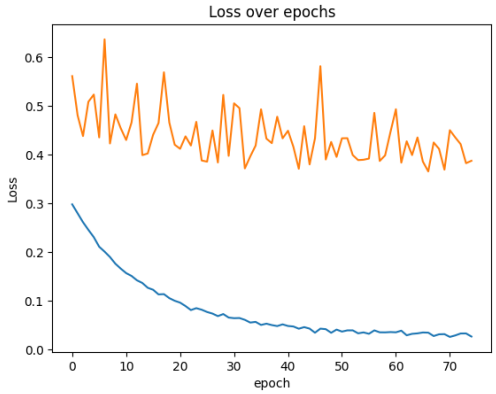

Fig4. Train and Test losses for Model1

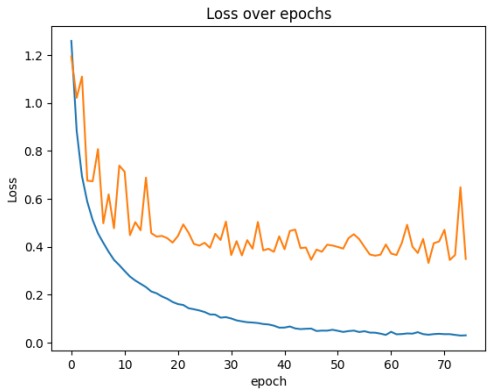

Fig5. Train and Test losses for Model2

From the above graphs we can conclude that the train accuracy decreases slightly from Model1 to Model2 whereas the test accuracy increases. The loss for Model2 takes a steep descent for initial epochs and there's greater variation.

For Model 1 the loss fluctuates in a small range and we can observe a gradual decrease in the pattern. Hence, we can say that increasing the number of residual blocks increases the testing accuracy and we achieve better results.

From the experiments conducted in this study we can conclude the following:

1. For bigger datasets Adam Optimizer seems to perform better than SGD because of its adaptive learning rates and momentum properties, which result in faster convergence.
2. It's worth noting that when the model encounters a local optima during training, the gradient of the loss function becomes close to zero, making it difficult for the optimization algorithm to escape from that point and continue moving towards the global optima.
3. Increasing the number of residual blocks improves testing accuracy and leads to better results, despite a slight decrease in train accuracy and greater loss variation.

References

Website or online resource
Geeks for Geeks. 2023. Residual Networks (ResNet) – Deep Learning
https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

Website or online resource
Towards Data Science. 2018. ResNets for CIFAR-10
https://towardsdatascience.com/resnets-for-cifar-10-e63e900524e0

Website or online resource
Medium. 2021. A 2021 Guide to improving CNNs-Optimizers: Adam vs SGD
https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008

Website or online resource
Github. 2017. Pytorch-cifar. Train CIFAR10 with PyTorch
https://github.com/kuangliu/pytorch-cifar

Website or online resource
Neurohive .2019. ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks
https://neurohive.io/en/popular-networks/resnet/

Website or online resource

ChatGPT. (2021). from OpenAI. OpenAI.