Optimizing ResNet Architecture for CIFAR-10 Classification

Runqi Chen, Luyi Ge
New York University
rc5235@nyu.edu, lg3846@nyu.edu

Introduction

Deep learning has transformed the field of computer vision, enabling remarkable advancements in image classification tasks. Among the myriad of deep neural network architectures, ResNet (Residual Network) stands out as a pivotal innovation that has revolutionized the training and performance of very deep networks. Introduced by Kaiming He et al. (2015) in their seminal paper "Deep Residual Learning for Image Recognition", ResNet addresses critical challenges associated with training deep networks, particularly in the context of image classification.

The appeal of ResNet lies in its ability to effectively train neural networks with hundreds of layers without encountering issues like vanishing gradients or performance degradation. This is achieved through the ingenious use of residual blocks, which incorporate skip connections or identity mappings. These skip connections allow gradients to flow directly through the network, facilitating smoother training and faster convergence.

In traditional deep neural networks, as more layers are added, the performance often saturates or even degrades due to difficulties in optimization. ResNet's architecture circumvents this problem by adopting a residual learning framework, where each residual block learns a residual mapping relative to the input. This residual learning approach simplifies the optimization process, enabling the network to capture intricate patterns and nuances in data more effectively.

In this paper, we present a novel adaptation of the ResNet-68 architecture, achieving a remarkable test accuracy of 90.14% on the CIFAR-10 image classification dataset while adhering to a constrained parameter budget of around 1 million parameters. To attain this level of accuracy, we conducted a comprehensive exploration of design parameters within the ResNet framework, including variations in channel configurations, filter sizes, and skip connection kernels. Furthermore, we investigated the impact of different optimization strategies (such as SGD and ADAM), regularization techniques, learning rates, batch sizes, epochs, and data augmentation methods on model performance.

Methodology

The building and selection process of our model started with the model structure proposed with the ResNet structure proposed by the paper by Kaiming He et al. (2005). Our exploration and experiments mainly focused on the pros and cons of different model depths. After experimented with models with different number of layers, we chose the ResNet-68 model structure i.e. a 68-layer ResNet model with 3 residual layers and 11 residual blocks within each layer. Our main reason for choosing this structure is that this model depth provides a good balance between model performance, training stability and convergence difficulty, and computational complexity:

- Model performance: we learned that the ResNets structure successfully overcome the optimization difficulty and demonstrate accuracy gains when the depth increases (Kaiming He et al. 2015) and the paper directly showed that the error rates for the ResNets models decreases as the layers increase. In our model selection process, to get better performance, we would like to fully utilize this accuracy gain brought by increased model depth. However, as explained below, increasing model depths bring other problems such as harder training parameter selection process and increased computational complexity. Therefore, we experimented with different model depths to find the model with high performance and acceptable level of difficulty of the training process and computational complexity.
- Training stability and convergence difficulty: as the depth of ResNet model increases, the model requires carefully tuned hyperparameters such as learning rate, momentum, etc. The 68-layer structure is considered as a model with relatively deep model but with careful selection of learning rate and other parameters, together with scheduler, it is still trainable. We have also experimented with models with more depths and found that the training process is harder in the sense that even with other techniques such as warm-up learning rate, the training loss curve tends to be quite unstable.
- Computational complexity: another problem brought by increased model depths and increased number of trainable parameters is the computational complexity, as more

https://github.com/LobbyBear/DL_spring24_mini_project computation is required, both in terms of memory usage and processing power. This can make the model slower to train and may require more powerful hardware. In our experiments with increased model layers, a direct observation is the training time significantly increased, but the computational complexity of the 68-layer model is within our acceptable range.

Combined the reasons stated above, we decided to use 68 layers in our ResNet model. Although we could potentially achieve better results with more layers, considered the complexity of the training process and computational resources, we believe that 68 layers is a reasonable selection for still reaching a good performance and putting the disadvantages of model depths under control. In the following part of this section, we will give a detailed description of our model structure, including its basic residual block design and the parameters of the whole model architecture.

Basic Residual Block

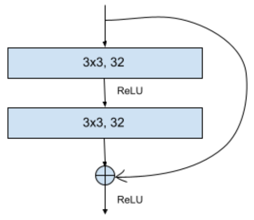

Figure 1: The Example of a Basic Residual Block.

The basic block employed in ResNet68 architecture serves as a fundamental building block for constructing deep convolutional neural networks. Specifically, within residual layer 2 each basic block comprises two consecutive 3x3 convolutional layers followed by batch normalization and ReLU activation functions. The first convolutional layer (Conv1) takes an input feature map with a specified number of input channels and performs a convolutional operation to transform it into a feature map with a desired number of output channels. Batch normalization is then applied to the output of Conv1 to stabilize and accelerate training. Subsequently, the second convolutional layer (Conv2) processes the normalized feature map, again using a 3x3 kernel, to further extract and encode spatial information. Another batch normalization step follows Conv2 to standardize the feature map's distribution. Concurrently, a skip connection (or shortcut) is introduced to facilitate the learning of residual mappings. This skip connection involves a 1x1 convolutional layer (Shortcut Conv) and batch normalization, ensuring that the input tensor dimensions match the output tensor dimensions if necessary. Finally, the output of the Conv2 layer is added elementwise to the output of the shortcut connection, and the combined result is passed through a ReLU activation function to produce the final output of the basic block. Through these sequential operations and the incorporation of residual connections, the basic block enables effective feature extraction, gradient propagation, and deeper network training while mitigating issues associated with vanishing gradients in deep neural networks.

ResNet-68 Model Architecture

The ResNet model with 68 layers is shown in Fig. 2.

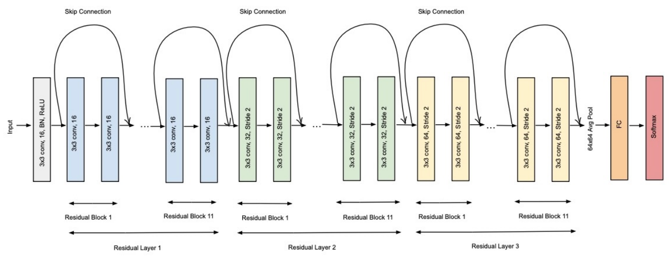

Figure 2: ResNet-68 Model Architecture.

Model parameters:

- Number of channels in the i-th layer: C1 = 16, C2 = 32, C3 = 64
- Filter size in the i-th layer: F1 = F2 = F3 = F4 = 3x3
- Kernel size in the i-th skip connection: K1 = K2 = K3 = 1x1
- Pool size in the average layer: 64x64

In the ResNet-68 architecture, the number of channels in each layer progressively increases to capture complex features: starting with 16 channels in the initial convolutional layer (C1), followed by 32 channels (C2) in subsequent layers, and culminating with 64 channels (C3) in deeper layers. The uniform filter size of 3x3 (F1, F2, F3) across all convolutional layers promotes effective feature extraction and preserves spatial information throughout the network. Notably, the use of 1x1 kernel size (K1, K2, K3) in the skip connections allows for dimensionality adjustments while maintaining computational efficiency. This design choice enables seamless gradient propagation during training, alleviating issues related to vanishing gradients in deep networks. Additionally, the average pooling layer utilizes a sizable pool size of 64x64, facilitating spatial down sampling and feature aggregation before the final classification layer. By integrating these design variables, the ResNet-68 architecture achieves a balance between model complexity and computational feasibility, showcasing its efficacy in image classification tasks and underscoring its relevance in advancing the field of deep learning for computer vision.

Implementation on CIFAR-10

We evaluated our model performance using the CIFAR-10 dataset, which is a widely recognized collection of images used for machine learning and computer vision research. It consists of 50k training images and 10k test images, each of which is 32x32 pixels in resolution, divided into 10 classes. We used data augmentation techniques and carefully selected the methods and parameters in our training process.

Data Augmentation

On our training data, we applied random crop with a padding of 4 pixels and random horizontal flip, which artificially enlarges the image and then randomly crops them back to their original sizes and randomly flips the images along the horizonal axis, which together help the model to enhance its robustness to variations in object positioning and to learn object-invariant features. The normalization process is also applied to enhance learning stability and the effectiveness of our network.

Selection of Training Parameters

We use SGD as our optimizer over Adam, particularly due to its better generalization performance on the CIFAR-10 dataset. As indicated by Zhou et al. (2020), SGD is more locally unstable than Adam at sharp minima and can thus better escape from them to flatter ones which leads to better model generalization. Additionally, SGD offers a simpler hyperparameter tuning process and is less computationally expensive, which simplified our hyperparameter selection process.

After experimenting with different hyperparameters, we selected a combination of an initial learning rate of 0.01, a batch size of 64, along with the CosineAnnealingLR rate scheduler to optimize our model's training process. We have observed that this combination provides a good balance between stability and convergence speed.

The use of CosineAnnealingLR, which cyclically tunes the learning, makes the initial learning rate choice relatively flexible since the scheduler can adjust the learning rate as the training progresses and can thus mitigate the risks associated with setting the learning rate too high or too low. We set an initial learning rate of 0.01 as it is within a reasonable range. We use a relatively small batch size of 64 since it updates to the weights more frequently and can lead to faster convergence. The small batch size also introduces more noises during the training process and leads to better generalization. Although smaller batch sizes could potentially extend the training time per epoch and introduce instability, our use of a learning rate scheduler can help counterbalance this issue effectively. As shown in the training and test metrics in the results section, our choice of parameters led to a smooth training process.

Finally, through experimenting with different epochs, we found that at around 50 epochs, the training and validation losses have stabilized and the validation accuracy plateaus. Therefore, we use epochs of 50 as a good stop point to avoid overfitting.

Results

As described in the previous section, our model is a 68-layer residual network. The first layer is 3x3 convolutions. Then it is followed by a stack of 3 residual layers, with 11 residual blocks in each layer, and within each residual block, the model performs 2 3x3 convolutions on the feature map sizes of {32, 16, 8} respectively. The skipped connection is added back within each residual block. The number of filters within each residual layer is {16, 32, 64} respectively. The network ends with a global average pooling and softmax layer and has 68 layers in total.

This modified ResNet model has a total of 1,050,202 trainable parameters. The detailed model summary is shown in Table x.

```
================================================================
Total params: 1,050,202
Trainable params: 1,050,202
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 12.47
Params size (MB): 4.01
Estimated Total Size (MB): 16.49
----------------------------------------------------------------
```

Table 1: Modified ResNet Model Summary

Upon the completion of the training process, our model achieved a final accuracy of 90.14% on the test dataset, which indicates an adept model design and parameter selection. Fig. x and Fig. y below show the train and validation losses and accuracies across epochs.

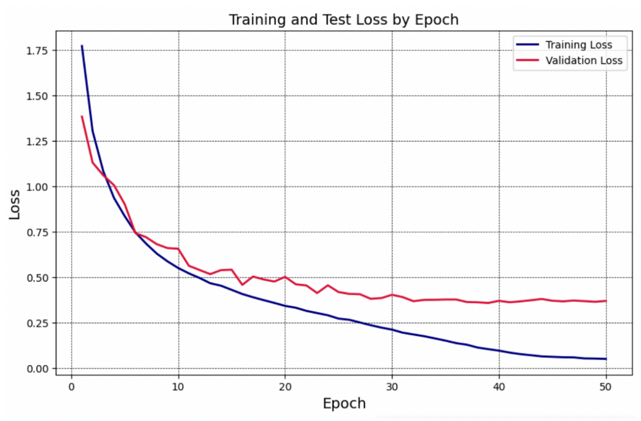

Figure 3: Model Train and Test Loss by Epoch

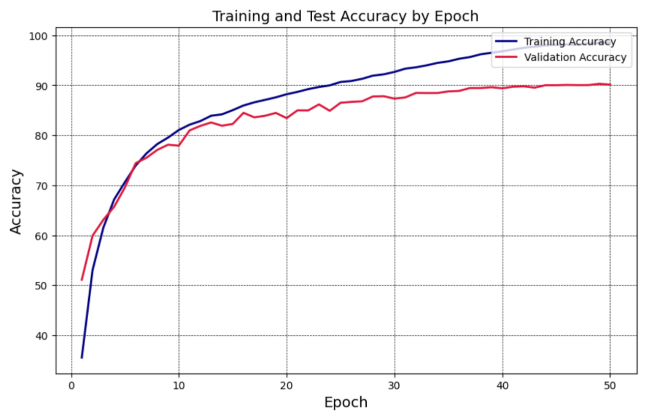

Figure 4: Model Train and Test Accuracy by Epoch

From these figures, we can see that the training loss steadily decreases, and the training accuracy steadily increases over epochs, indicating that the model is effectively learning from the training data and converging towards an optimal solution. On the other hand, the test loss curve and test accuracy curve indicate that the model has a healthy degree of generalization capacity without substantial overfitting.

In conclusion, the observed training and testing metrics suggest that the modified ResNet model has been effectively trained, demonstrated stable convergence, robust generalization, and reliable performance on both the training and unseen test data.

Conclusion

In our project, we delved into the architecture of residual networks as presented in the seminal paper "Deep Residual Learning for Image Recognition" by Kaiming He et al., published in 2015. Our exploration primarily centered on analyzing the performance of models with varying depths, alongside meticulous hyperparameter tuning. We modified the ResNets models and built a ResNet68 model, which demonstrated a compelling test accuracy of 90.14%.

Throughout this research endeavor, we also gained insights into the remarkable capability of ResNets to address optimization difficulty and degradation problem from increased network depths.

Should our project continue, we would further explore how other model attributes, such as the structure within each basic block, the strides and padding sizes etc., could further enhance the model performance.

References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[2] Zhou, P., Feng, J., Ma, C., Xiong, C., Hoi, S., & E, W. (2020). Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.