Modified Residual Network (ResNet) Model for Image Classification on the CIFAR10 Dataset

Shashank Shekhar, Mohith Jarapala, Ayan Chowdhury

ss16116@nyu.edu, mpj8687@nyu.edu, ac10230@nyu.edu

Abstract

We present a modified ResNet18 model and examine its performance on the CIFAR10 dataset while keeping the number of trainable model parameters below 5 million. We experiment with different optimizers, activation functions, and other hyperparameters to maximize accuracy.

Codebase

The code for the project can be found at this link.

Introduction

There has been substantial research on image classification and the advent of Convolutional Neural Networks (Krizhevsky, Sutskever, and Hinton 2012) has spurred research in the same direction with increasingly complex modifications to the original architecture. CNNs use multiple layers of convolutions and pooling to learn hierarchical representations of the input. Each layer learns more abstract features that are built upon the lower-level features learned in the previous layers. This hierarchy of representations allows the network to learn increasingly complex patterns and relationships between the input and output. It is thus logical to assume that increasing the depth of the CNN would lead to more effective feature discrimination and subsequent classification. However, very deep CNNs suffer from the problem of vanishing gradients since the gradients have to propagate through a large number of layers during backpropagation and can become exponentially small.

ResNet addresses this problem by using skip connections or shortcuts that allow the network to bypass one or more layers. These shortcuts enable the gradients to flow more easily through the network during backpropagation, making it easier to optimize the weights and train the network (He et al. 2016a). This effectively allows the selection of specific layers and bypasses those that are not contributing much.

Overview of Architectural Choices

We have developed a Pre-Activation Residual Network model (He et al. 2016b) to improve the model performance. Pre-activation is a technique used in residual neural networks (ResNets) that involves performing batch normalization and ReLU activation before the convolution operation in a residual block. The advantage of pre-activation is that it enables better gradient propagation through the network, leading to faster and more stable training, as well as improved generalization performance. By performing batch normalization and ReLU activation before the convolution operation, pre-activation can reduce the magnitude of the inputs to subsequent layers, making it easier for gradients to flow through the network. Additionally, pre-activation can help prevent the problem of "dying ReLUs," which can lead to a failure to train the network (Tsang 2018).

We experimented with Adam, SGD, and AdaGrad optimizers but obtained better results with the first two, and tried the ReLU and GeLU activations for both versions. We also experimented with different numbers of epochs and batch sizes, ultimately using early stopping for automatic detection of the optimum number of epochs and a batch size of 64. Batch size was varied from 128 to 64, but the best results were obtained with the smaller batch size, as indicated by previous research too (Masters and Luschi 2018). Out of the combinations of optimizers and activation functions, we obtained the best results with the Adam optimizer and the GeLU activation function. Our experimentation is in line with prior reports on GeLU's superior performance compared to ReLU on several computer vision tasks, including image classification on the CIFAR10 dataset (Hendrycks and Gimpel 2020). Moreover, Adam has been empirically evaluated as a better optimizer for image classification problems, especially using deep architectures in prior research (Dogo, Afolabi, and Twala 2022), and (Hassan et al. 2022), a finding which held for our experiments as well.

Methodology

Dataset

The CIFAR-10 dataset is a collection of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It is commonly used as a benchmark for image classification and machine learning algorithms. We divide the dataset into training, validation, and testing set, with 40,000 training images, and 10,000 validation and test images each.

Data Preprocessing and Augmentation

Data Augmentation involves applying various transformations to the existing data samples to create new images, thereby making the model robust to noise and enhancing its generalizability by exposing it to variations in the input. Augmentation has proven especially successful in deep neural networks, which are more susceptible to overfitting (Takahashi, Matsubara, and Uehara 2020).

We define two types of transformations to be performed in sequence. The first transformation is random cropping, that randomly crops a 32x32 patch from the input image with a padding of 4 pixels on each side. The second transformation is the random horizontal flip that randomly flips the input image horizontally with a probability of 0.5. The third transformation is normalizes the image with the given mean and standard deviation values, which are pre-computed for the CIFAR10 dataset. Normalization helps to ensure that the input data has zero mean and unit variance, which can improve the model's training and convergence. The fourth and final transformation converts the input image to a PyTorch tensor. We then define a similar series of transformations to be applied to the test dataset with the exception of the data augmentation transformations.

Architecture

ResNet (short for Residual Network) is a popular deep learning architecture that introduced residual connections. Residual connections allow for the training of deeper neural networks by mitigating the vanishing gradient problem. Resnet18 has a pre-processing convolution layer, eight residual blocks, and a linear layer. The preprocessing convolution layer extracts the features from 3 channels and gives 64 channels for the subsequent layers. In the next part, the eight residual blocks are divided into four layers, with 2 in each.

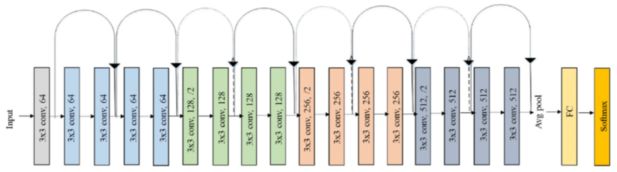

Figure 1: The Original ResNet18 Architecture (Ramzan et al. 2019)

The model we opted for, the Pre-Activation ResNet18 model, is an extension of the original ResNet18 architecture that incorporates a pre-activation residual block instead of the standard residual block. In the original ResNet, the residual connection is added after applying batch normalization and a nonlinearity (ReLU) to the output of a convolutional layer. In contrast, in the pre-activation version, batch normalization and ReLU are applied before the convolutional operation. This has been shown to lead to better performance and faster convergence compared to the standard residual block.

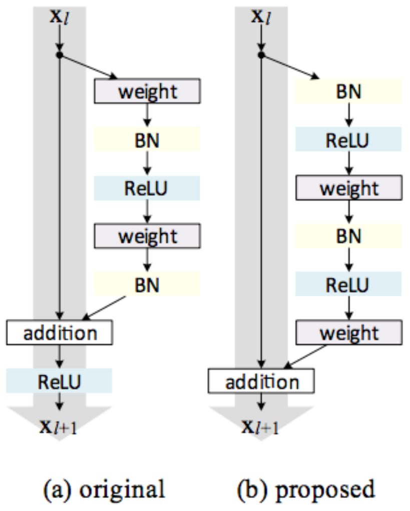

Figure 2: A Comparison of the ResNet Architecture (a) vs the Pre-Activation ResNet Architecture (b) (Wei Yang 2021)

Hyperparameter Tuning

Hyperparameter tuning in deep learning refers to the process of finding the best combination of hyperparameters for a neural network model that can produce the most accurate results for a given problem. Hyperparameters are parameters that cannot be learned directly from the data during training, but are set prior to training and control the behavior of the model. Hyperparameter tuning is an important step in deep learning, as the performance of a model is highly dependent on the choice of hyperparameters. By tuning the hyperparameters, one can improve the accuracy and generalization performance of a deep learning model (Probst, Boulesteix, and Bischl 2019).

We experimented with batch size, number of epochs, activation functions, and optimizers as part of hyperparameter tuning. For batch size, we tried different sizes but obtained the best results with 32. In the case of number of epochs, we utilized early stopping so that the training could automatically stop upon convergence, with the possibility to go on till 100 epochs. We tried out Adam and SGD as our choice of optimizers and obtained the best performance with Adam. ReLU and GeLU were used as activations with GeLU registering better results. These combinations have been plotted below.

Thus, the best performing combination was the Pre-Activation ResNet18 model with Adam as the optimizer, GeLU as the activation function, and 32 as the batch size, giving an accuracy of 90.76% with a total of around 2.8 million trainable parameters. Diagrams for each architecture have been provided in the codebase.

Results

The accuracy score obtained was 90.23% for the Adam+ReLU Pre-Activation model, 88.32% for the

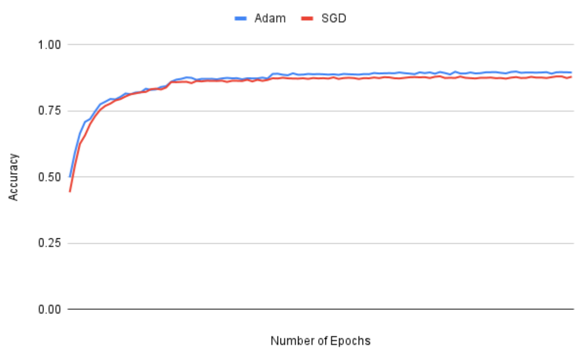

Figure 3: Validation Accuracy for Adam and SGD with ReLU

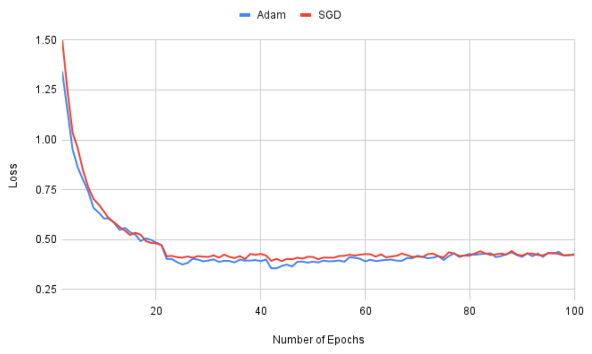

Figure 4: Validation Loss for Adam and SGD with ReLU

SGD+ReLU model, 90.76% for the Adam+GeLU model, and 88.56% for the SGD+GeLU model.

The confusion matrix for the best performing model is shown below

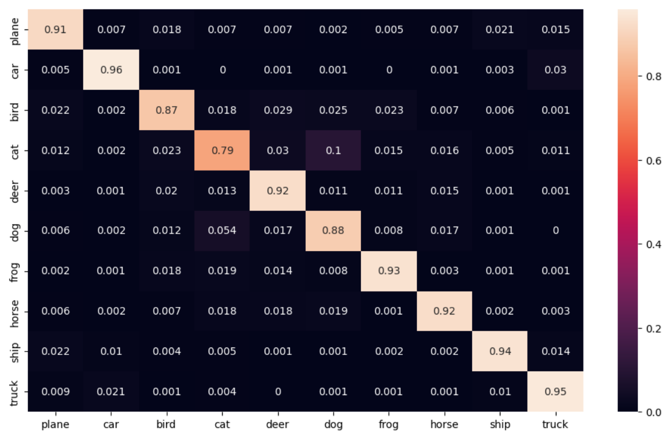

Figure 7: Confusion Matrix for the Pre-Activation ResNet18 Model Using Adam and GeLU

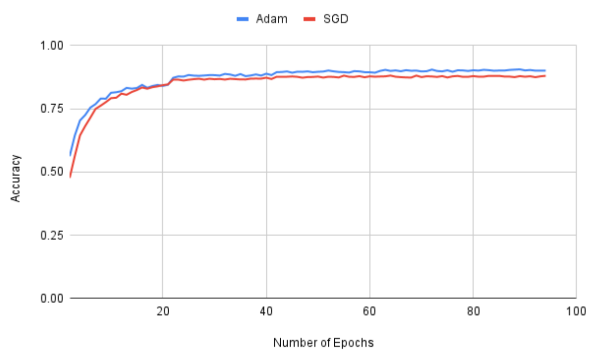

Figure 5: Validation Accuracy for Adam and SGD with GeLU

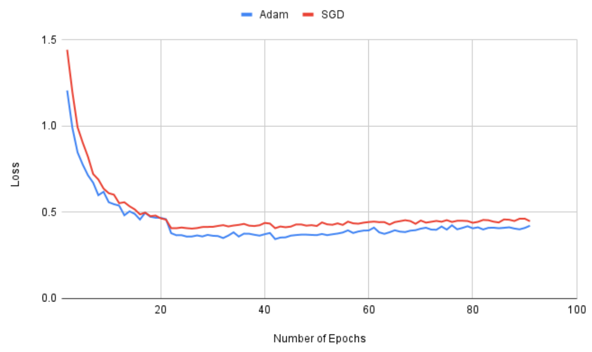

Figure 6: Validation Loss for Adam and SGD with GeLU

References

Dogo, E.; Afolabi, O.; and Twala, B. 2022. On the Relative Impact of Optimizers on Convolutional Neural Networks with Varying Depth and Width for Image Classification. *Applied Sciences*, 12: 11976.

Hassan, E.; Shams, M.; Hikal, N.; and Elmougy, S. 2022. The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study. *Multimedia Tools and Applications*.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity Mappings in Deep Residual Networks. arXiv:1603.05027.

Hendrycks, D.; and Gimpel, K. 2020. Gaussian Error Linear Units (GELUs). arXiv:1606.08415.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Masters, D.; and Luschi, C. 2018. Revisiting Small Batch Training for Deep Neural Networks. arXiv:1804.07612.

Probst, P.; Boulesteix, A.-L.; and Bischl, B. 2019. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *J. Mach. Learn. Res.*, 20(1): 1934–1965.

Ramzan, F.; Khan, M. U.; Rehmat, A.; Iqbal, S.; Saba, T.; Rehman, A.; and Mehmood, Z. 2019. A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks. *Journal of Medical Systems*, 44.

Takahashi, R.; Matsubara, T.; and Uehara, K. 2020. Data Augmentation Using Random Image Cropping and Patching for Deep CNNs. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9): 2917–2931.

Tsang, S.-H. 2018. Review: Pre-Activation ResNet with Identity Mapping. [Online; posted September 22, 2018].

Wei Yang, F. D., Yang Peiwen. 2021. Train CIFAR10 with PyTorch. https://github.com/kuangliu/pytorch-cifar.