Deep Learning Mini-Project

Omar Rayyan, Rameen Mahmood, Mahmoud Hafez

NYU Tandon School of Engineering
Github repository: https://github.com/omarrayyann/Deep-Learning-Competition

Abstract

In pursuit of advancing the field of image recognition, this project presents a modified Residual Network (ResNet) architecture, specifically tailored to maximize classification performance on the CIFAR-10 dataset within the parameter constraint of fewer than 5 million trainable parameters. Through a process of architectural refinement and hyperparameter optimization, the proposed model demonstrates significant improvements in accuracy while adhering to the stipulated parameter budget. This report details the modifications implemented, evaluates the impact of various architectural choices, and discusses the performance of the proposed model. Despite the complexity inherent to CIFAR-10's richly varied images, our model achieved a remarkable training accuracy of 100% and test accuracy of 96.5%. More impressively, when confronted with a set of unseen images from a Kaggle competition, the model demonstrated robust generalizability, attaining an accuracy of 87.4%.

Introduction

The CIFAR-10 dataset presents an array of challenges for image classification models, primarily due to its diverse array of small, colored images. Residual Networks (ResNets) have risen to the challenge with their skip connections and profound depth. This project seeks to push the boundaries of ResNet's capabilities by introducing a modified architecture tailored for superior performance within the bounds of a 5 million parameter limit.

Our architecture incorporates Squeeze-and-Excitation (SE) blocks to harness channel-wise feature recalibration, thus fostering nuanced learning of discriminative features. With a series of carefully calibrated residual blocks, our network embraces depth and width optimization to enhance learning capacity while preventing overfitting. The result is a model that not only masters the CIFAR-10 training and test datasets, achieving accuracies of 100% and 97.4%, respectively, but also maintains a good generalization performance, as evidenced by an 87.4% accuracy on previously unseen images on a Kaggle competition.

This report provides a granular look at our ResNet variant, detailing the rationale behind each architectural choice and demonstrating how these decisions collectively contribute to the model's efficacy. Subsequent sections will unpack the intricacies of each component, from the initial convolutional layer to the final output, and analyze how these contribute to the model's impressive performance on both familiar and unseen datasets.

Methodology

We present a detailed framework for the training and evaluation of a deep learning model, specifically designed for the CIFAR image classification task. The model is based on a modified Residual Network (ResNet) architecture, enhanced with Squeeze-and-Excitation (SE) blocks to recalibrate channel-wise feature responses adaptively.

Data Preprocessing

Our dataset comprises images from the CIFAR dataset, specifically CIFAR-10. Each subset contains images of 32x32 pixels, divided into training and testing batches. We preprocess this data by reshaping and normalizing it to fit the input requirements of our neural network. To enhance the model's ability to generalize and prevent overfitting, we apply data augmentation techniques to the train and test data. This includes converting images to tensors, randomly cropping and padding the images to introduce variability, randomly flipping images horizontally to simulate different orientations, and normalizing the pixel values to ensure uniformity across the dataset. The normalization parameters are specifically chosen based on the dataset's characteristics to facilitate faster convergence during training. In contrast, the test images undergo a more straightforward preprocessing pipeline, where they are converted to tensors and normalized using the same parameters as the training images.

Modifications to ResNet Architecture

Initial Convolutional Layer The model begins with an initial convolutional layer (LeCun, Bengio, and Hinton 2015), which acts as the first point of contact with the input image. This layer scans the image and starts to recognize basic patterns, such as edges, colors, and textures. The convolutional layer sets the stage for deeper analysis by trans-

forming the raw pixel data into a form that the subsequent layers can work with more effectively.

Squeeze-and-Excitation (SE) Block Following the initial convolution, the model integrates an Squeeze-and-Excitation block (Hu, Shen, and Sun 2018), focusing on channel-wise recalibration of feature maps. This mechanism allows the model to dynamically adjust the importance of each channel based on the global information, thereby emphasizing relevant features and suppressing less useful ones. The SE block's structure comprises an adaptive average pooling layer to squeeze global spatial information into a channel descriptor, followed by a two-layer excitation mechanism involving reduction and expansion through convolutional layers. This improves the representational power of the network without significant increases in parameter count or computational complexity.

Residual Blocks

This configuration of residual blocks is inspired by the deep residual learning frameworks (He et al. 2016), which demonstrated significant improvements in training deep networks by using shortcut connections and identity mappings. The structure of a Residual Block in our model begins with a 2D convolutional layer with a kernel size of 3x3, stride of 1, and padding to maintain the spatial dimensions of the input. This is followed by batch normalization and a ReLU activation function to introduce non-linearity. A second convolutional layer with the same specifications follows to further process the features. If there is a mismatch in dimensions between the input and output of the block, a separate convolutional layer with a 1x1 kernel is applied to the block's input to align its dimensions with the block's output, ensuring the additivity of the shortcut connection.

Layer Configuration Our model includes three sets of residual blocks configured as follows:

- The first layer contains 4 residual blocks. All blocks in this layer maintain the number of channels at 64, but utilize padding and stride adjustments to ensure spatial dimensions are preserved after each convolution.
- The second layer comprises 4 residual blocks with an increase in channels from 64 to 128. The first block of this layer uses a stride of 2 to reduce the spatial dimensions by half, effectively pooling the features spatially while increasing the depth.
- The third layer consists of 3 residual blocks, further increasing the channels to 256. Similar to the second layer, a stride of 2 is used in the first block to continue reducing the spatial dimensions while increasing the feature depth.

Each layer's transition is marked by an increase in the number of channels, allowing for deeper feature extraction, and includes a stride change in the first block to reduce dimensionality and increase receptive field size. These layers collectively improve the network's ability to learn more complex features without excessive computational overhead, adhering to the model's design constraints of fewer than 5 million trainable parameters.

Average Pooling Layer In our ResNet model, the progression from deep convolutional layers to the final classification decision is facilitated through a sequence of Average Pooling, Flattening, and a Linear Fully Connected Layer. Initially, the Average Pooling layer reduces the spatial dimensions of the feature maps from 4x4 to 1x1 across 256 channels, summarizing the extracted features while maintaining channel depth. This is followed by the Flattening operation, which transforms the pooled output into a one-dimensional vector of 256 elements, preparing it for final processing. The culmination of this sequence is the Linear Fully Connected Layer, which maps these 256 features directly to the 10 output classes of the CIFAR dataset. This reduces the computational complexity but also extracts the high-level features necessary for accurate image classification.

Output Layer The output layer produces a one-dimensional vector of 256 features. This vector encapsulates the high-level abstractions learned by the network from the input image. The Output Layer consists of a Linear Fully Connected Layer which maps the 256-element vector to the 10 output classes corresponding to the CIFAR-10 dataset. This dense layer utilizes a linear transformation, characterized by a set of learned weights and biases, to project the extracted features into a space where each dimension represents one of the ten possible categories. The use of a softmax function is implied in the subsequent loss calculation phase, which interprets the linear outputs as logits representing the model's confidence in each class. This enables the network to produce a probability distribution over the classes.

Training procedure

Training is performed using Stochastic Gradient Descent (SGD) with momentum and weight decay to optimize the network's parameters. We employ a CrossEntropyLoss function to compute the loss, and gradient clipping is applied to stabilize the training process. The learning rate is initially set high and gradually decreased using a Cosine Annealing schedule across epochs to fine-tune the model's weights effectively.

Hyperparameter	Value
Loss Function	Cross Entropy
Weight Decay	0.0005
Learning Rate	0.1, 0.01, 0.001
Momentum	0.9
Optimizer	SGD
Scheduler	CosineAnnealingLR
$T_{\max}$ for Scheduler	200

Table 1: Hyperparameters for modified ResNet Training

Mixup We employed the mixup data augmentation technique (Zhang et al. 2017) to enhance the generalization capabilities of our image classification model. Mixup operates by generating virtual training examples through the linear interpolation of pairs of training examples and their corresponding labels. Specifically, each new training input is created by combining two randomly chosen images, weighted

by a parameter λ, which is sampled from a Beta distribution (figure 1). This approach not only diversifies the training data but also encourages the model to learn more robust features by smoothing the decision boundaries between classes. In our implementation, the mixup algorithm was integrated into the training loop, where each batch of images underwent the mixup transformation before being fed into the neural network. This strategy was particularly effective in preventing overfitting, as evidenced by the high accuracy observed on the test set, which consistently exceeded the training accuracy by a significant margin. After 250 epochs, our model achieved a test accuracy of up to 96.5%, demonstrating the potent impact of the mixup on enhancing model performance in image classification tasks.

The high test accuracy result was achieved despite a very low training accuracy when using mixup (50-60%) even after 200 epochs. This is because, when using mixup during training, each input and label is a combination of two different examples. This means the model is trained not on the original clean examples but on blended ones. This can make the task of precisely predicting the blended label more challenging, potentially lowering training accuracy. The accuracy calculation during training might not fully reflect the model's ability to classify unblended, clean test data. The training accuracy is computed against the mixed labels, which are harder to predict accurately compared to the original labels used in the test set.

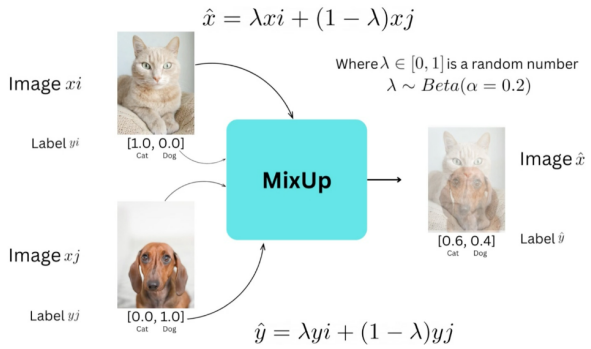

Figure 1: Example of mixup

Our final model was trained with mixup with 0.9 probability for 400 epochs and then mixup was removed for a few more epochs to get the best accuracy until it started overfitting. Right after its removal, there was a significant jump in both the training and test accuracy in the first epoch. Eventually, the training accuracy reached almost 100% and the training accuracy 97.0%

Results

We experimented with various settings, adjusting both the number and size of the network layers. The most effective configuration we discovered features 4, 3, 2, and 3 residual blocks in each layer, with the channels increasing from 64 to 64, 64 to 128, 128 to 128, and then 128 256 across these layers. Mixup was also introduced in this model and showed a significant difference in accuracy when compared to models that did not utilize mixup.

The model is well-suited to handle complex and diverse image data within the constraints of fewer than 5 million parameters, as it contains 4,993,166 parameters exactly. This specific model setup achieved an almost perfect accuracy on the training data, and a very high test accuracy of 97.0% on the CIFAR-10 dataset (figure 2 and 3).

Figure 2: Training and Testing accuracy across epochs

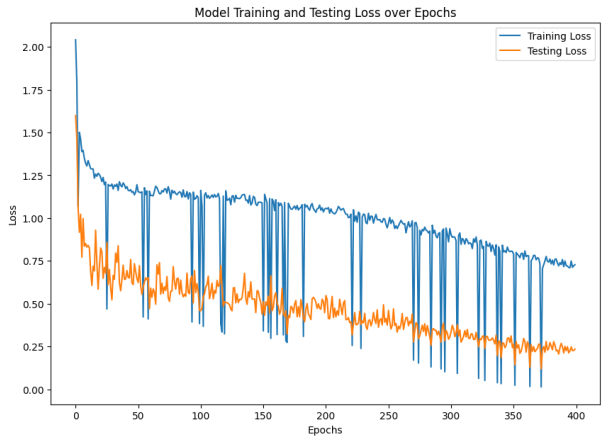

Figure 3: Training and Testing loss across epochs

When we tested our best model on unseen images in a Kaggle competition, it still performed very well, achieving an accuracy of 87.9%. This drop in performance compared to our CIFAR-10 results likely reflects the new challenges and differences in the Kaggle images, but the decrease was modest. This indicates that our model, with its strategic design and the addition of Mixup, can adapt well to new environments and did not overfit on our training data.

Conclusion

This project has successfully demonstrated the effectiveness of our modified ResNet architecture tailored for the CIFAR-10 dataset, while following the parameter constraint of fewer than 5 million trainable parameters. The architectural changes, including the integration of Squeeze-and-Excitation (SE) blocks and optimized residual blocks, have significantly enhanced the model's ability to learn discriminative features effectively and efficiently. The final model not only achieved high accuracy levels—100% on the training set and 97.0% on the test set—but also showed robust generalization capabilities as shown by the 87.9% accuracy in the secondary test set of unseen images. These results underscore the model's strengths in handling diverse and complex image data without resulting to overfitting. The combination of depth and width in the network layers, along with advanced training techniques and data preprocessing methods, has enabled the model to perform exceptionally well within the given parameter constraints. The inclusion of Mixup during the training phase also introduced additional robustness, improving the model's ability to generalize and perform well on real-world datasets that differ from the training data. This balance of complexity and efficiency serves as a foundation for future improvements in image classification tasks. Overall, this project contribute serves as a robust framework for developing high-performance, parameter-efficient models in the field of deep learning for image recognition. This sets a precedent for future research aimed at optimizing deep neural networks for similar tasks with limited computational resources. Future work may explore further enhancements in model architecture, such as integrating more advanced forms of regularization and exploring alternative training methodologies to push the boundaries of model performance and efficiency even further.

References

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature*, 521(7553): 436–444.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.