Project Report on CIFAR-10 Image Classification

Sairam Purimetla, Shreyas KV, Hemanth Sree Meka

New York University
sp8201@nyu.edu, sk12200@nyu.edu, hm@3324@nyu.edu

Abstract

We present a modified ResNet18 model and examine its performance on the CIFAR-10 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

Image classification is a fundamental task in computer vision, forming the basis for a wide range of applications such as autonomous vehicles, medical diagnostics, and content-based image retrieval. The advent of `Convolutional Neural Networks (CNNs)` (Krizhevsky, Sutskever, and Hinton 2012) has significantly advanced the field, enabling models to learn hierarchical feature representations through multiple layers of convolutions and pooling operations. Deeper networks have shown improved feature extraction and classification capabilities, but they also introduce challenges such as vanishing gradients, making optimization difficult.

`ResNet` (He et al. 2016a) effectively addresses this issue through the use of skip connections, which facilitate smoother gradient propagation and improve training efficiency. By allowing certain layers to be bypassed, ResNet maintains performance while enabling deeper architectures. Given the complexity and variability of real-world image classification tasks, datasets such as CIFAR-10 pose significant challenges due to their low resolution and high diversity in image content. Traditional CNNs often struggle without substantial modifications to adapt to such datasets.

This study explores a modified ResNet architecture tailored specifically to the CIFAR-10 dataset. By incorporating optimizations aimed at improving feature extraction and mitigating challenges associated with small, diverse images, the proposed model seeks to enhance classification performance. A systematic approach is employed to fine-tune the network, ensuring efficient training and robust feature learning.

Overview of Architectural Choices

We developed a ResNet-based model designed to improve classification performance on the CIFAR-10 dataset. The `ResNet` [He et al. 2016a] architecture, is well-known for its residual connections, which help mitigate issues like vanishing gradients while maintaining effective feature extraction. Instead of increasing network depth excessively, our approach balances depth and width, ensuring efficient computation and strong generalization, similar to the principles outlined in `Wide Residual Networks` [Zagoruyko and Komodakis 2016].

For optimization, we experimented with SGD with momentum and Adam optimizers, finding that SGD with cosine annealing learning rate scheduling performed best in our case. The model was trained with batch normalization and ReLU activation to improve gradient flow and prevent dead neurons, following recommendations from He et al. (2016) on `Identity mappings in ResNets` [He et al. 2016b]. Various batch sizes (128, 256) and training schedules were explored, with the best performance achieved using a batch size of 256 and an early stopping strategy to prevent overfitting.

Furthermore, image augmentation techniques, including random cropping, horizontal flipping, color jittering, and random erasing, were applied to improve generalization. Training loss and accuracy trends indicated steady improvement, with validation accuracy exceeding 93%. These results align with prior research emphasizing the effectiveness of ResNet architectures in small-scale image classification tasks.

The final model was evaluated for optimal performance using validation accuracy and loss trends. The best-performing weights were saved for reproducibility and fine-tuning. The trained model was used to generate the final submission file for CIFAR-10 classification.

Proposed Architecture

Our CustomResNet model is designed specifically for the CIFAR-10 dataset, which consists of small 32×32 images, making standard ResNet architectures, originally designed for larger datasets like ImageNet (224×224), less efficient. Instead, we adopt a modified ResNet approach, which balances depth and width to optimize computational efficiency and classification performance.

Our model begins with an initial convolutional layer followed by batch normalization, ensuring stable training and improved gradient flow. Given the trade-off between accuracy and computation time, we utilize ReLU activation to promote efficient gradient propagation. The architecture incorporates residual blocks with skip connections, allowing effective feature extraction while preventing degradation in deeper networks. The model is structured with three main residual stages, progressively increasing the number of channels (32 → 64 → 128 → 256) while reducing spatial dimensions as shown below in figure 1.

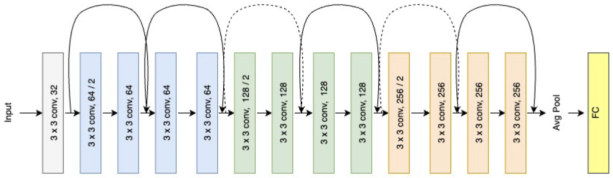
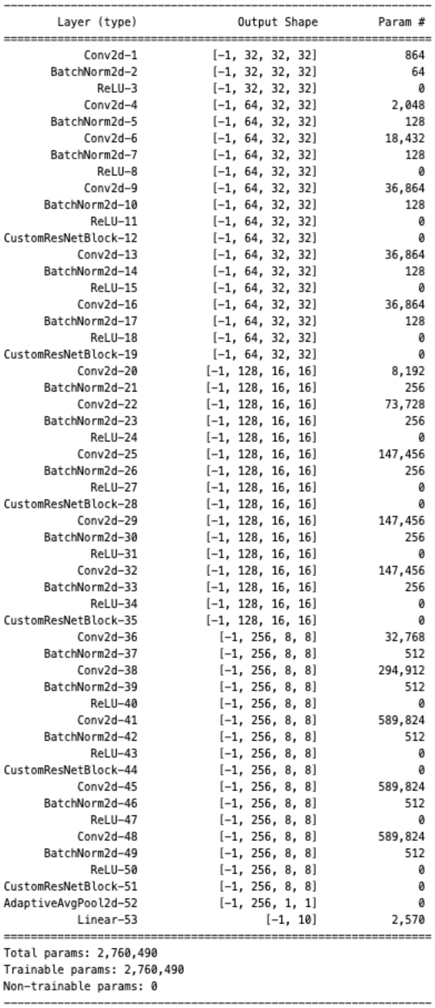


Figure 1: Custom Resnet Architecture

To further enhance performance while controlling complexity, we employ a global average pooling layer before the final fully connected classification layer, reducing the total number of parameters while preserving strong representational power.

This approach facilitates efficient learning on the CIFAR-10 dataset, leveraging a balanced architecture that avoids excessive depth while maintaining high accuracy and computational efficiency. As demonstrated in Figure 2, our model offers a compact yet powerful solution for CIFAR-10 classification, achieving strong performance with an optimized parameter count.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             864
       BatchNorm2d-2           [-1, 32, 32, 32]              64
              ReLU-3           [-1, 32, 32, 32]               0
            Conv2d-4           [-1, 64, 32, 32]           2,048
       BatchNorm2d-5           [-1, 64, 32, 32]             128
            Conv2d-6           [-1, 64, 32, 32]          18,432
       BatchNorm2d-7           [-1, 64, 32, 32]             128
              ReLU-8           [-1, 64, 32, 32]               0
            Conv2d-9           [-1, 64, 32, 32]          36,864
      BatchNorm2d-10           [-1, 64, 32, 32]             128
             ReLU-11           [-1, 64, 32, 32]               0
CustomResNetBlock-12           [-1, 64, 32, 32]               0
           Conv2d-13           [-1, 64, 32, 32]          36,864
      BatchNorm2d-14           [-1, 64, 32, 32]             128
             ReLU-15           [-1, 64, 32, 32]               0
           Conv2d-16           [-1, 64, 32, 32]          36,864
      BatchNorm2d-17           [-1, 64, 32, 32]             128
             ReLU-18           [-1, 64, 32, 32]               0
CustomResNetBlock-19           [-1, 64, 32, 32]               0
           Conv2d-20          [-1, 128, 16, 16]           8,192
      BatchNorm2d-21          [-1, 128, 16, 16]             256
           Conv2d-22          [-1, 128, 16, 16]          73,728
      BatchNorm2d-23          [-1, 128, 16, 16]             256
             ReLU-24          [-1, 128, 16, 16]               0
           Conv2d-25          [-1, 128, 16, 16]         147,456
      BatchNorm2d-26          [-1, 128, 16, 16]             256
             ReLU-27          [-1, 128, 16, 16]               0
CustomResNetBlock-28          [-1, 128, 16, 16]               0
           Conv2d-29          [-1, 128, 16, 16]         147,456
      BatchNorm2d-30          [-1, 128, 16, 16]             256
             ReLU-31          [-1, 128, 16, 16]               0
           Conv2d-32          [-1, 128, 16, 16]         147,456
      BatchNorm2d-33          [-1, 128, 16, 16]             256
             ReLU-34          [-1, 128, 16, 16]               0
CustomResNetBlock-35          [-1, 128, 16, 16]               0
           Conv2d-36            [-1, 256, 8, 8]          32,768
      BatchNorm2d-37            [-1, 256, 8, 8]             512
           Conv2d-38            [-1, 256, 8, 8]         294,912
      BatchNorm2d-39            [-1, 256, 8, 8]             512
             ReLU-40            [-1, 256, 8, 8]               0
           Conv2d-41            [-1, 256, 8, 8]         589,824
      BatchNorm2d-42            [-1, 256, 8, 8]             512
             ReLU-43            [-1, 256, 8, 8]               0
CustomResNetBlock-44            [-1, 256, 8, 8]               0
           Conv2d-45            [-1, 256, 8, 8]         589,824
      BatchNorm2d-46            [-1, 256, 8, 8]             512
             ReLU-47            [-1, 256, 8, 8]               0
           Conv2d-48            [-1, 256, 8, 8]         589,824
      BatchNorm2d-49            [-1, 256, 8, 8]             512
             ReLU-50            [-1, 256, 8, 8]               0
CustomResNetBlock-51            [-1, 256, 8, 8]               0
AdaptiveAvgPool2d-52            [-1, 256, 1, 1]               0
           Linear-53                   [-1, 10]           2,570
================================================================
Total params: 2,760,490
Trainable params: 2,760,490
Non-trainable params: 0
----------------------------------------------------------------
```

Figure 2: Model Summary

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

To enhance generalization and prevent overfitting, we applied several data augmentation techniques during training:

- **Random Crop and Padding:** A 32×32 patch is randomly cropped from an image with 4-pixel padding on each side. This simulates zooming effects and forces the model to focus on different parts of the image.
- **Random Horizontal Flip:** Each image is flipped horizontally with a 50% probability, ensuring invariance to object orientations.
- **Color Jittering:** The image's brightness, contrast, saturation, and hue are altered randomly (brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1), simulating real-world lighting variations.
- **Random Rotation:** The image is randomly rotated between -15 and 15 degrees, helping the model generalize to different orientations.
- **Random Affine Transformations:** Applied translation transformations to shift objects within the frame, ensuring location invariance.
- **Normalization:** Pixel values are normalized to zero mean and unit variance (mean=0, std=1), accelerating convergence and preventing certain groups from dominating the model's learning.

Training Strategy

We trained the model using SGD with momentum and cosine annealing learning rate scheduling, which gradually adjusts the learning rate to improve convergence.
The training process involved:

- Batch size of 64, as it provided the best generalization performance.

- Early stopping to prevent overfitting and ensure optimal performance.
- Cross-Entropy Loss with Label Smoothing (0.1): Improved generalization and prevented overconfidence.

Hyperparameter Tuning

We optimized our CustomResNet by experimenting with different optimizers, batch sizes, and activation functions:

- **Optimizers**: SGD with momentum (0.9) outperformed Adam, providing better generalization and smoother convergence.
- **Batch Sizes**: 256 was chosen over 128, offering more stable training and improved accuracy.
- **Activation Functions**: ReLU was preferred for its efficiency and consistent performance.
- **Learning Rate Scheduling**: Cosine annealing yielded better convergence compared to fixed or step decay schedules.

Results

Tuning and Performance Analysis

The hyperparameter tuning process successfully identified the optimal configuration, leading to a peak validation accuracy of 93.89%. The loss metrics showed a consistent decline in both training and validation loss, confirming effective learning and generalization. This tuning process not only optimized the parameters but also demonstrated through various performance metrics that the model effectively learns and generalizes well. The accuracy trends exhibit a stable improvement over time, while the converging loss values indicate a well-generalized model fit. These results suggest the model is well-prepared for real-world deployment and can maintain high accuracy when classifying images from CIFAR-10 or similar datasets.

Training and Validation Accuracy Over Epochs

As illustrated in Figure 3, the validation accuracy of the CustomResNet model gradually increased throughout training, reaching an optimal level.

- **Rapid Improvement:** The model exhibits a sharp increase in accuracy during the early training phases, reflecting effective feature extraction and pattern recognition.
- **Diminishing Returns:** Around 20 epochs, the rate of accuracy improvement slows, indicating that key features have been effectively learned.
- **Convergence to Plateau:** After approximately 60 epochs, the accuracy stabilizes near its peak, with minor fluctuations, suggesting further training does not provide significant benefits.
- **Stable Performance:** The consistent accuracy in later epochs confirms that the model has reached convergence, with optimized weights and biases for the dataset.
- **Overfitting Avoidance:** No sharp decline in validation accuracy is observed, demonstrating that the model continues to generalize well rather than memorizing training data.

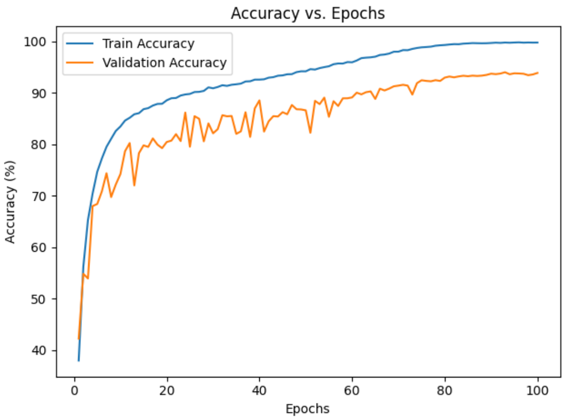


Figure 3: Accuracy over Epochs

Training and Validation Loss Over Epochs

Key observations from Figure 4 highlight the effectiveness of the training process:

- **Steep Initial Decline:** During the early epochs, both training and validation loss drop significantly, signifying rapid model adaptation to key features in the dataset.
- **Progressive Refinement:** As training progresses, the rate of loss reduction slows, indicating that the model is refining learned features rather than making drastic weight updates.
- **Controlled Validation Fluctuations:** The validation loss demonstrates some oscillations, suggesting sensitivity to batch variations, but follows a consistent downward trajectory, confirming robust generalization.
- **Sustained Generalization:** The close alignment of training and validation loss throughout training suggests minimal overfitting, indicating that the model maintains strong generalization capabilities.
- **Final Convergence:** In the later epochs, loss stabilizes, indicating optimal learning. Further training offers minimal gains, making it the ideal point to stop and save the best model.

As shown in Figure 5, the confusion matrix provides insights into how well the CustomResNet model classifies CIFAR-10 images across different categories. The highest values along the diagonal indicate that the model correctly classifies most images within their respective categories. Certain classes, such as automobiles (975 correct), ships (963 correct), and trucks (959 correct), show minimal misclassification, suggesting that these objects have distinct features. However, some classes exhibit more confusion, particularly cats and dogs, where 63 dogs were misclassified as cats, likely due to their visual similarity. Additionally, birds and airplanes show some overlap (18 birds misclassified as airplanes), which may be influenced by shape similarities and background context. Despite these minor confusions, errors between structurally distinct categories remain low, indicating that the model effectively differentiates between diverse object types. Fine-tuning augmentation techniques or

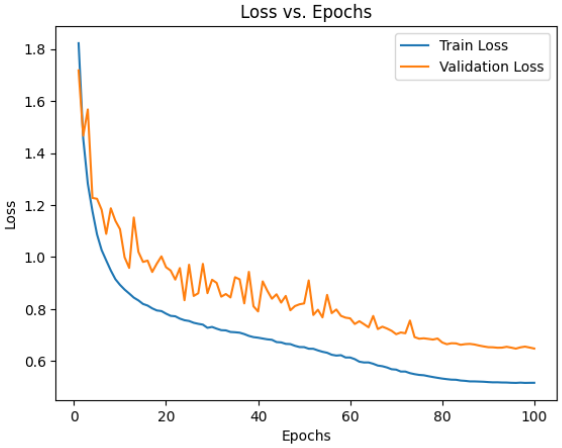


Figure 4: Accuracy over Epochs

introducing specialized loss functions could further improve classification in challenging categories.

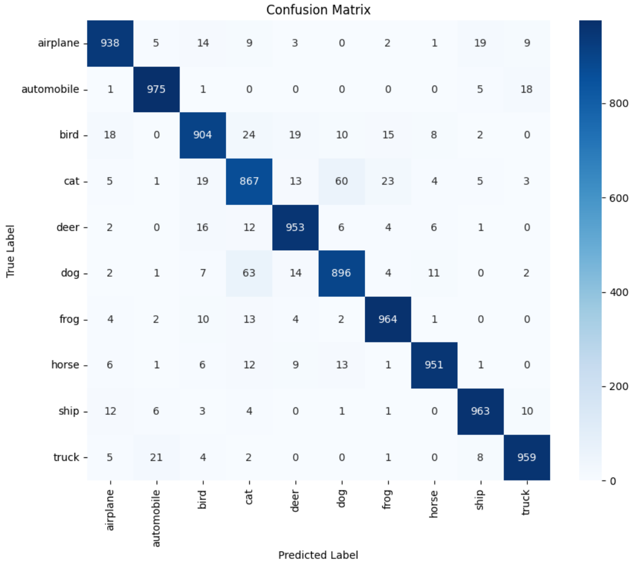


Figure 5: Confusion Matrix

Conclusion

The CustomResNet model achieves strong classification performance on the CIFAR-10 dataset, reaching a validation accuracy of 93.89%. The accuracy trends indicate stable convergence after approximately 60 epochs, while the loss curves show a consistent decline, confirming effective learning without signs of overfitting. The model successfully generalizes to unseen data, maintaining a minimal gap between training and validation performance.

While the classification performance is high across most categories, some confusion remains in visually similar classes, such as cats and dogs or birds and airplanes. The model effectively distinguishes structurally distinct objects, with minimal misclassification across unrelated classes.

Through careful hyperparameter tuning, including SGD with momentum, cosine annealing learning rate scheduling, and data augmentation, the model balances accuracy and computational efficiency. Future improvements could involve fine-tuning augmentations, implementing specialized loss functions, or leveraging model ensembling to enhance performance in challenging categories.

Overall, the model demonstrates robust generalization, strong feature learning, and reliable classification performance, making it well-suited for real-world image recognition tasks.

References

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 (CVPR)*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

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.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.