Comparative Analysis of Deep Learning Techniques for Image Classification on CIFAR-10

Rajatsubhra Chakraborty*

Introduction

This report presents a comparative analysis of three deep learning experiments aimed at improving image classification accuracy on the CIFAR-10 dataset. The CIFAR-10 dataset comprises 60,000 32x32 color images in 10 classes, with 6,000 images per class. Our experiments focused on evaluating the performance of different convolutional neural network (CNN) architectures and augmentation techniques, including the use of standard convolutions, depthwise separable convolutions, and mixup augmentation.

Methodology

Data Preparation

All experiments utilized the same data preprocessing steps. The CIFAR-10 dataset was divided into training, validation, and test sets. The training data underwent transformations such as random horizontal flips and random cropping to augment the dataset. Both training and test datasets were normalized.

Experiment 1: Baseline Model

The baseline model comprised a simple CNN architecture with three convolutional layers followed by max pooling, batch normalization, and dropout layers to reduce overfitting. The model was trained using the Adam optimizer, and performance metrics including accuracy, precision, recall, and F1-score were recorded.

Experiment 2: Depthwise Separable Convolutions

In the second experiment, we modified the baseline model by replacing standard convolutions with depthwise separable convolutions. This approach aimed

*Univeristy of North Carolina Charlotte, Email: rchakra6@uncc.edu

to reduce the number of parameters and computational complexity while maintaining model performance.

Experiment 3: Mixup Augmentation

The third experiment employed mixup augmentation, a technique that generates virtual training examples by linearly combining pairs of examples and their labels. This method was applied to the baseline model to investigate its impact on model robustness and performance.

Model Architectures

Baseline Model Architecture

- **Input Layer**: Accepts 32x32 color images from the CIFAR-10 dataset.
- **Convolutional Layer 1**: Convolution with 32 filters, kernel size 3x3, padding, followed by ReLU activation.
- **Batch Normalization Layer 1**: Normalizes the output of the first convolutional layer.
- **Max Pooling Layer 1**: Pooling with a 2x2 filter and stride.
- **Convolutional Layer 2**: Convolution with 64 filters, kernel size 3x3, padding, followed by ReLU activation.
- **Batch Normalization Layer 2**: Normalizes the output of the second convolutional layer.
- **Max Pooling Layer 2**: Pooling with a 2x2 filter and stride.
- **Convolutional Layer 3**: Convolution with 128 filters, kernel size 3x3, padding, followed by ReLU activation.
- **Batch Normalization Layer 3**: Normalizes the output of the third convolutional layer.
- **Max Pooling Layer 3**: Pooling with a 2x2 filter and stride.
- **Dropout Layer**: Applied to reduce overfitting.
- **Fully Connected Layer 1**: Dense layer with 512 units followed by ReLU activation.
- **Dropout Layer**: Applied again before the final output layer.
- **Output Layer**: A fully connected layer with 10 units and softmax activation.

Depthwise Separable Convolution Model Architecture

This model modifies the Baseline Model by incorporating depthwise separable convolutions:

- **Input Layer**: Same as the baseline model.
- **Depthwise Separable Convolutional Layers**: Replace standard convolutional layers.
- **Batch Normalization and Max Pooling**: Same structure as the baseline model.
- **Dropout and Fully Connected Layers**: Adjustments in layer sizes and dropout rates.

Mixup Augmentation Model Architecture

This model uses the Baseline Model architecture with mixup augmentation during training:

- **Input Layer through Dropout Layer**: Identical structure as the baseline model.
- **Mixup Augmentation**: Applied during training to generate virtual training examples.
- **Fully Connected and Output Layers**: Same as the baseline model.

Results

Performance Metrics

The performance of each model was evaluated based on training and validation loss, accuracy, precision, recall, and F1-score. The baseline model achieved an accuracy of 76.51%, precision of 0.767, recall of 0.765, and F1-score of 0.763. The depthwise separable convolution model showed a slight decrease in performance, with an accuracy of 71.42%, precision of 0.727, recall of 0.714, and F1-score of 0.711. The mixup augmentation model demonstrated an accuracy of 71.25%, precision comparable to the baseline model, and similar recall and F1-scores.

Graphical Analysis

The training and validation losses and accuracies over epochs for all three experiments were plotted to visualize the learning process. Figure 1 shows the training loss and validation accuracy over 10 epochs for the three experiments: the baseline model, the depthwise separable convolution model, and the mixup augmentation model.

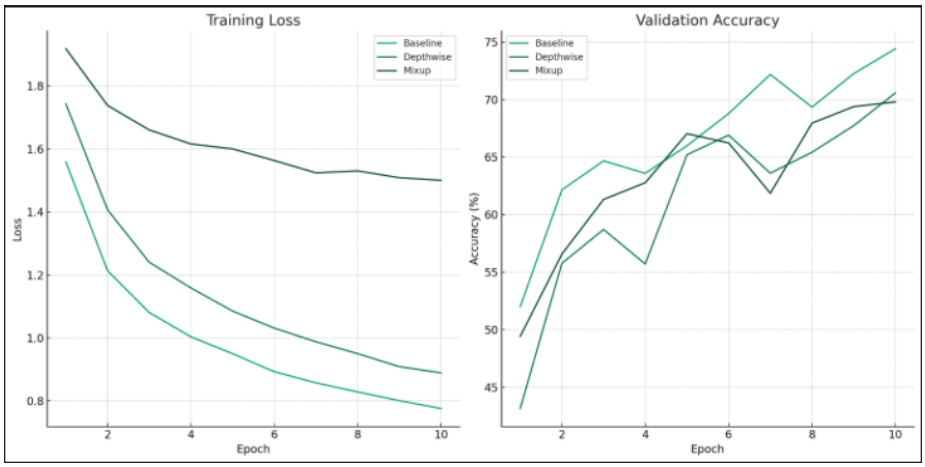

Figure 1: Training loss and validation accuracy of baseline, depthwise separable convolution, and mixup augmentation models over epochs.

The left graph shows the training loss for each model. It's noticeable that the mixup model starts with the highest loss but gradually improves, indicating the effect of mixup augmentation on model training. The depthwise model, despite its higher complexity and parameter count, does not significantly outperform the baseline in terms of loss reduction over epochs.

The right graph displays the validation accuracy of each model across epochs. The baseline model consistently shows superior performance, reaching the highest accuracy by the 10th epoch. The depthwise separable convolution model trails behind, suggesting that despite its potential for efficiency, it may require further optimization to reach the baseline's effectiveness. The mixup model shows promising improvement over epochs, indicating its potential to enhance model generalization and performance.

Discussion

The experiments utilized several regularization and optimization techniques to enhance model performance and prevent overfitting. Specifically, dropout and batch normalization were employed as regularization techniques. Dropout randomly drops out neurons during the training process to prevent them from co-adapting too much, while batch normalization normalizes the input to each activation function to stabilize learning. The Adam optimizer was used across all experiments for its adaptive learning rate properties, which help in faster convergence.

The baseline model provided a solid foundation, demonstrating high accuracy and other metrics. The depthwise separable convolution model, despite its reduced computational complexity, showed a decrease in performance, which

might be attributed to its lower capacity to capture complex features compared to the baseline model. However, this model significantly increased the number of parameters, suggesting an over-parameterization that did not translate into better accuracy, likely due to the increased difficulty in training. The mixup augmentation experiment showed promising results in enhancing model generalization, although its accuracy slightly lagged behind the baseline model. This could be due to the regularization effect of mixup, which might require further fine-tuning of the model parameters and training regimen.

Conclusion

Our experiments highlight the trade-offs between model complexity, computational efficiency, and performance in deep learning for image classification. While depthwise separable convolutions offer a promising avenue for reducing computational costs, they may require careful tuning to match the performance of standard convolutions. Mixup augmentation presents a valuable technique for improving model robustness, although its effectiveness may depend on the model architecture and training setup.

Appendix: Hyperparameters Table

Hyperparameter	Baseline Model	Depthwise/Mixup Models
Learning Rate	0.001	0.001
Batch Size	128	128
Optimizer	Adam	Adam
Dropout Rate	0.25	0.5 (Depthwise) / 0.25 (Mixup)
Epochs	10	10

Table 1: Hyperparameters used in the experiments.