CIFAR-100 Image Classification using Convolutional Neural Networks

1. Introduction and Background

Image classification is a fundamental task in computer vision, where the goal is to assign a label to an image based on its content. It has a wide range of applications, including object detection, scene understanding, and autonomous driving. The CIFAR-100 dataset is a benchmark dataset commonly used for developing and evaluating image classification algorithms. It presents a challenging problem due to its small image size (32x32 pixels), diverse set of 100 fine-grained classes, and the need for models to generalize across a wide variety of object categories.

In this project, we design and train a convolutional neural network (CNN) to classify images in the CIFAR-100 dataset. Our focus is on developing a baseline architecture that incorporates data augmentation, regularization techniques, and optimization strategies to achieve reasonable performance on this complex dataset. We also explore the model's learning behavior, performance metrics, and areas for potential improvement.

2. Dataset

The CIFAR-100 dataset was introduced by Krizhevsky, Nair, and Hinton (2009) at the University of Toronto. It consists of 60,000 color images (32x32 pixels) divided into 100 distinct classes, each containing 600 images. There are 500 training images and 100 test images per class. The dataset is split into:

- 50,000 training images
- 10,000 test images

Each image belongs to one of 100 fine-grained classes such as "apple," "bicycle," or "whale." The dataset also includes a coarser categorization into 20 superclasses. For this project, we focus solely on the 100 fine-grained classes.

3. Methods

Data Preprocessing

- Images were normalized to the [0,1] range by dividing pixel values by 255.
- Labels were one-hot encoded for multi-class classification.

- The training set was split 80-20 into training and validation sets.

Model Architecture

Our CNN architecture includes:

- Convolutional Layers: Three blocks of increasing filter sizes (64, 128, 256) with ReLU activation, followed by Batch Normalization.
- Pooling: MaxPooling layers reduce spatial dimensions after each block.
- Dropout: Applied after pooling and dense layers to mitigate overfitting (rates of 0.3, 0.4, and 0.5).
- Dense Layers: A fully connected layer with 512 units, followed by a 100-unit softmax output for classification.

Training Setup

- Optimizer: Stochastic Gradient Descent (SGD) with momentum (0.9) and Nesterov acceleration.
- Learning Rate Schedule: `ReduceLROnPlateau` adjusts the learning rate when validation accuracy plateaus.
- Early Stopping: Stops training if validation accuracy does not improve for 30 epochs.
- Data Augmentation: Random rotations, shifts, zooms, and horizontal flips applied during training to improve generalization.
- Mixed Precision Training: Enabled for faster computation.

4. Experiments

Hyperparameters

Parameter	Value
Batch Size	32

Epochs	Up to 200 (early stooping applied)
Initial Learning Rate	0.01
Dropout Rates	0.3,0.4,0.5

Training Performance

- Best Validation Accuracy: 64.35%
- Final Test Accuracy: 64.51%
- Best Validation Loss: 1.3229

Training Behavior

The training and validation accuracy curves show steady improvement, with the model converging around epoch 120. The loss curves decrease over time, but a small gap between training and validation loss suggests mild overfitting, mitigated by regularization techniques like dropout and data augmentation.

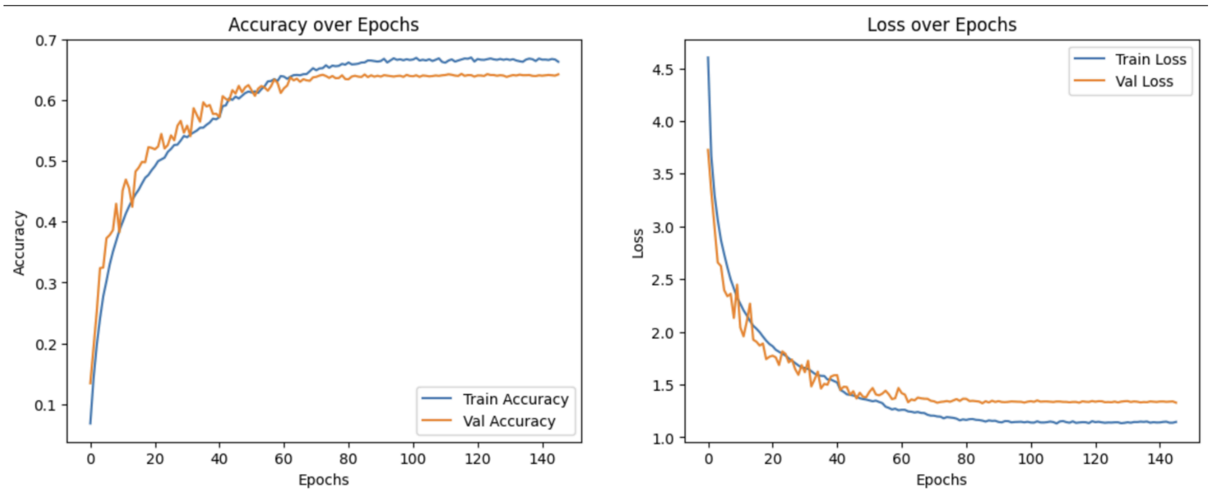
5. Results

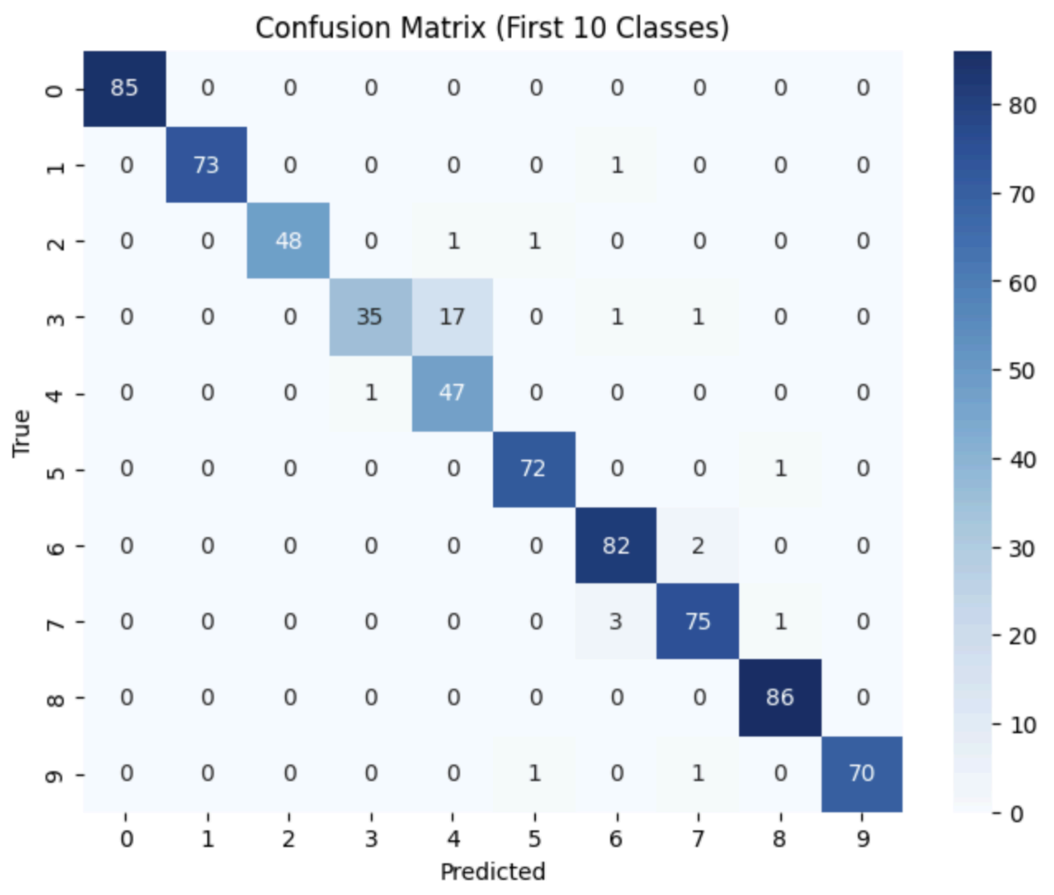
Metric	Value
Best Validation Accuracy	64.35%
Best Validation Loss	1.3229
Final Test Accuracy	64.51%

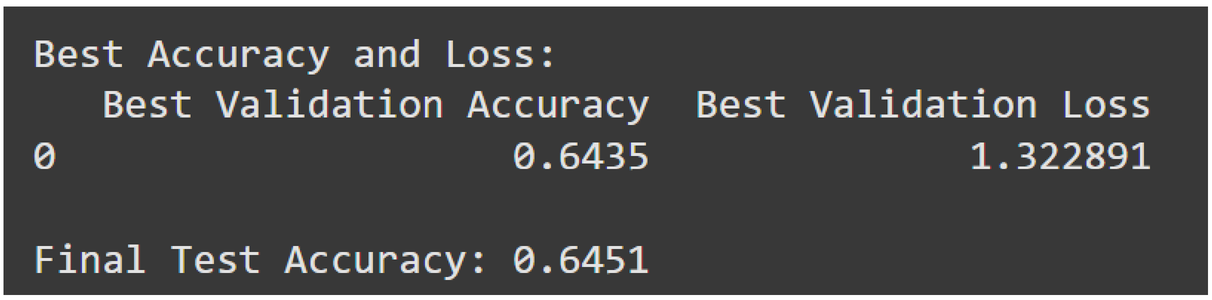
Visualizations

- Accuracy and Loss Curves: Display the model’s learning progress.

- Confusion Matrix (First 10 Classes): Highlights classification strengths and errors. Some classes (e.g., 3 and 4) show confusion, indicating potential areas for improvement.







```
Best Accuracy and Loss:
   Best Validation Accuracy  Best Validation Loss
0                    0.6435              1.322891

Final Test Accuracy: 0.6451
```

6. Discussion and Future Directions

Our CNN model demonstrates reasonable performance on CIFAR-100, achieving around 64.5% test accuracy. Given the complexity of the dataset, this result indicates that the model has learned useful features, though it is far from state-of-the-art. Some observations and potential improvements include:

- Model Depth: Exploring deeper architectures like ResNet or DenseNet could improve performance.
- Advanced Data Augmentation: Adding techniques like CutMix, MixUp, or random erasing may enhance robustness.
- Regularization: Adding weight decay (L2 regularization) or dropout at different layers could further reduce overfitting.
- Learning Rate Strategies: Implementing cosine annealing or cyclical learning rates might yield better convergence.
- Class Imbalance Analysis: Investigating specific classes with low accuracy could inform targeted strategies, such as focal loss or class weighting.

7. References

Krizhevsky, A., Nair, V., & Hinton, G. (2009). *The CIFAR-100 dataset*. University of Toronto. Available at: https://www.cs.toronto.edu/~kriz/cifar.html