Final Project Report: Image Classification Using Deep Learning

Rahman Heydarov

December 23, 2024

Abstract

This project develops convolutional neural networks (CNNs) for CIFAR-10 image classification using TensorFlow and Keras. Three models were implemented: a basic CNN, a regularized model, and a hyperparameter-tuned model. The basic model, with convolutional blocks and max pooling, achieved 80% accuracy. The regularized model, incorporating batch normalization, dropout, early stopping, and a learning rate scheduler, improved accuracy to 89%. Hyperparameter tuning via random search optimized parameters like filter size and dropout rate, yielding an 85% accuracy. The results demonstrate the effectiveness of regularization and tuning in enhancing CNN performance.

Contents

1 Introduction

Image classification is a key task in computer vision where the goal is to assign an image to one of several predefined categories. It has many practical uses, such as identifying objects in autonomous vehicles, recognizing products in online shopping, and analyzing medical images to assist in diagnosis. With the increasing amount of image data available today, there is a need for methods that can classify images accurately and efficiently. Traditional approaches often relied on manually defining features, but these methods struggle with complex and diverse datasets. Deep learning has emerged as a powerful alternative, offering automatic feature extraction and high performance.

Convolutional neural networks (CNNs) are a type of deep learning model specifically designed for tasks like image classification. CNNs use layers like convolutional, pooling, and fully connected layers to process image data. These layers help the model learn patterns, such as edges and shapes, and combine them to recognize more complex features. Because of this, CNNs have become the standard for many image classification tasks. In this project, the CIFAR-10 dataset is used to test CNN performance. This dataset includes 60,000 color images in 10 categories, such as airplanes, cars, and animals. Each image is small, only 32x32 pixels, making it an ideal starting point for testing and improving CNN models.

This project builds and compares three models for classifying CIFAR-10 images. The first is a basic CNN that uses standard convolutional and pooling layers. The second is a regularized CNN that includes batch normalization and dropout layers to prevent overfitting. It also uses training strategies like early stopping and a learning rate scheduler. The third model applies hyperparameter tuning using random search to find the best settings for the CNN, such as the number of filters, kernel size, and dropout rate. All models are trained using the Adam optimizer and evaluated based on their accuracy.

The goal of this project is to explore how regularization and hyperparameter tuning affect model performance. By comparing these approaches, this work demonstrates the steps involved in building and refining CNN models for practical use. Additionally, GitHub is used for version control and project tracking, ensuring a well-documented workflow.

2 Methodology

2.1 Dataset

The CIFAR-10 dataset was selected for this project due to its widespread use in benchmarking image classification models. It contains 60,000 color images, each with a resolution of 32x32 pixels, divided into 10 classes, including airplanes, cars, and animals. Each class contains 6,000 images, with a training set of 50,000 images and a test set of 10,000 images. The dataset is balanced, ensuring an equal distribution of samples across all categories, which is essential for training models without introducing bias. Its manageable size and resolution make it suitable for developing and testing convolutional neural networks (CNNs) efficiently.

2.2 Data Preprocessing

Data preprocessing included standardization and augmentation. Images were normalized to scale pixel values between 0 and 1, ensuring uniformity and accelerating training. Data augmentation techniques such as random rotations, flips, shifts, and zooms were applied to increase dataset variability and reduce overfitting. These augmentations helped the model generalize better by exposing it to slightly altered versions of the same images. The training set was shuffled before feeding it into the model to prevent bias during training. Additionally, the test set was kept separate and untouched during training to ensure unbiased evaluation.

2.3 Model Architecture

Three CNN architectures were implemented. The basic model consisted of sequential convolutional blocks with ReLU activation functions, followed by max pooling to reduce spatial dimensions. The final dense layers included a softmax-activated output layer for multi-class classification.

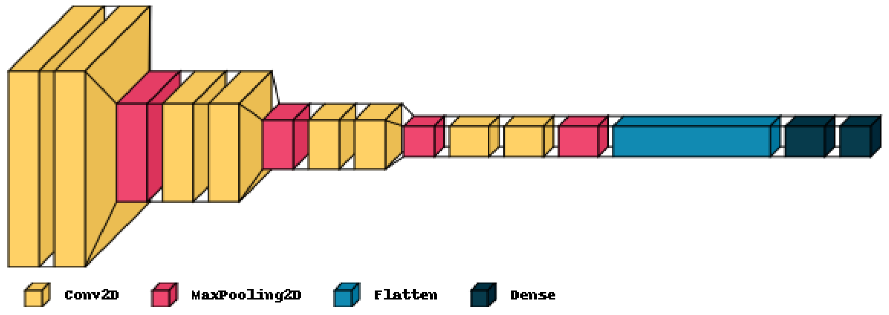


Figure 1: Basic Model Architecture

The regularized model added batch normalization after each convolutional layer to stabilize learning and dropout layers to reduce overfitting by randomly deactivating neurons during training. Early stopping was used to halt training when validation performance plateaued, and a ReduceLROnPlateau scheduler was applied to dynamically adjust the learning rate, enabling better convergence.

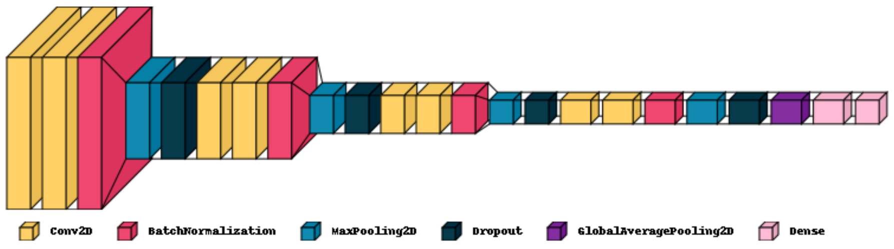


Figure 2: Regularized Model Architecture

The hyperparameter-tuned model used random search to optimize parameters such as the number of filters, kernel size, units in dense layers, and dropout rates. This approach allowed experimentation with different configurations to find the most effective architecture for the dataset.

2.4 Hyperparameter Tuning and Regularization

Hyperparameter tuning involved a random search over a predefined range of values for key parameters, such as filter sizes (32, 64, 128), kernel sizes (3x3, 5x5), dense layer units (128, 256), and dropout rates (0.2, 0.5). Models were trained on combinations of these parameters, and the configuration with the highest validation accuracy was selected. Regularization techniques, including dropout and batch normalization, reduced overfitting by introducing randomness and stabilizing weight updates. The early stopping and learning rate scheduler further ensured that training focused on optimal epochs, preventing unnecessary computations.

2.5 Evaluation Metrics

The models were evaluated using several metrics to ensure comprehensive performance analysis. Accuracy was used as the primary metric to measure the percentage of correctly classified images. Additionally, a classification report was generated, providing precision, recall, and F1-score for each class to evaluate the model's performance in handling imbalanced classes, if any. A confusion matrix was also used to visualize the model's predictions, highlighting the number of correct and incorrect classifications for each class. Together, these metrics provided a detailed understanding of the model's strengths and areas for improvement.

2.6 GitHub Integration

GitHub was used throughout the project for version control. The repository tracked code changes, making it easier to manage updates and revert to earlier versions if needed. Commit messages provided a clear log of progress. This ensured a well-organized workflow and facilitated submission of the project.

3 Results

The performance of the models was evaluated using multiple metrics, including accuracy, loss, precision, recall, F1-score, and a confusion matrix. The following table summarizes the test accuracy of the three models:

Model	Accuracy
Basic Model	80%
Regularized Model	89%
Hyperparameter-Tuned Model	85%

3.1 Visualizations

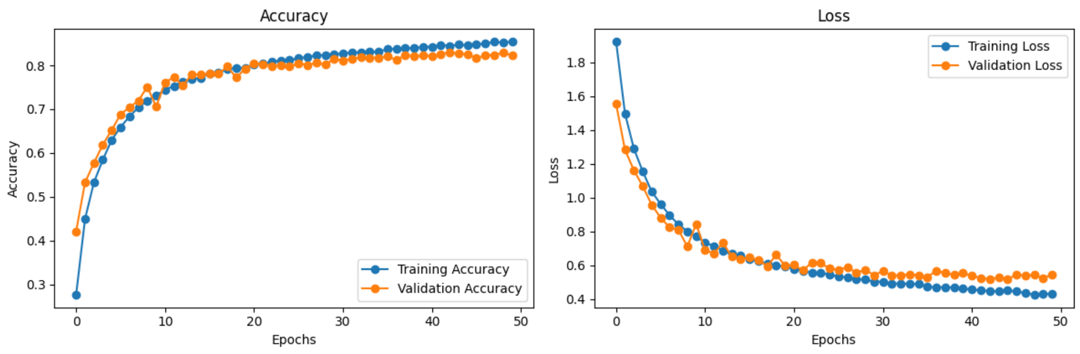
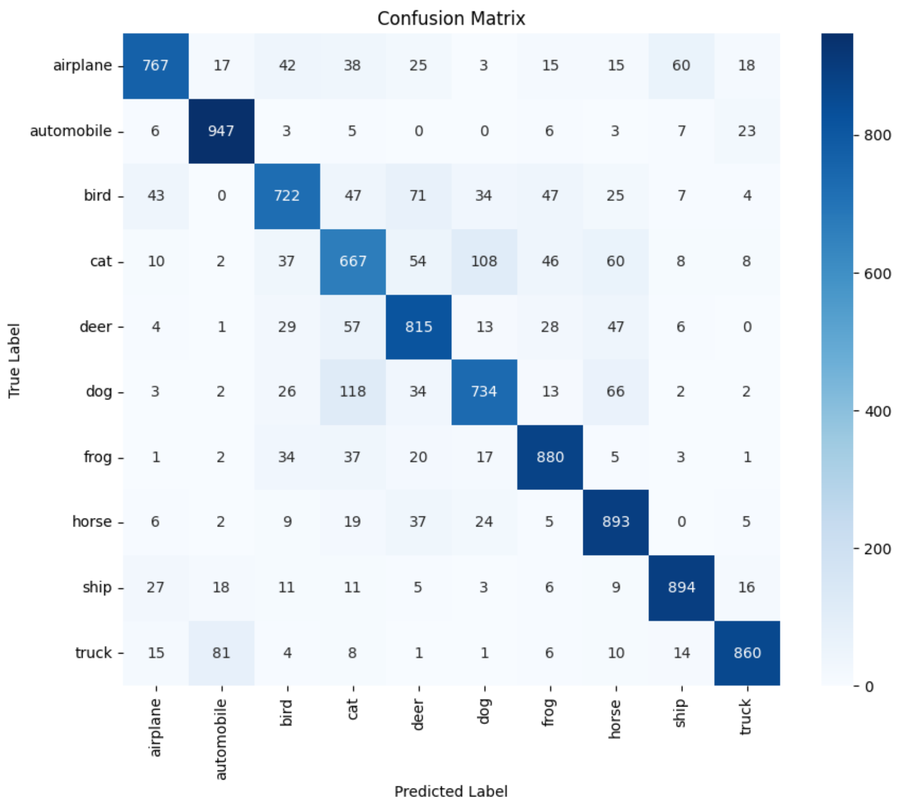


Figure 3: Basic Model Curves



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	767	17	42	38	25	3	15	15	60	18
automobile	6	947	3	5	0	0	6	3	7	23
bird	43	0	722	47	71	34	47	25	7	4
cat	10	2	37	667	54	108	46	60	8	8
deer	4	1	29	57	815	13	28	47	6	0
dog	3	2	26	118	34	734	13	66	2	2
frog	1	2	34	37	20	17	880	5	3	1
horse	6	2	9	19	37	24	5	893	0	5
ship	27	18	11	11	5	3	6	9	894	16
truck	15	81	4	8	1	1	6	10	14	860

Figure 4: Basic Model Confusion Matrix

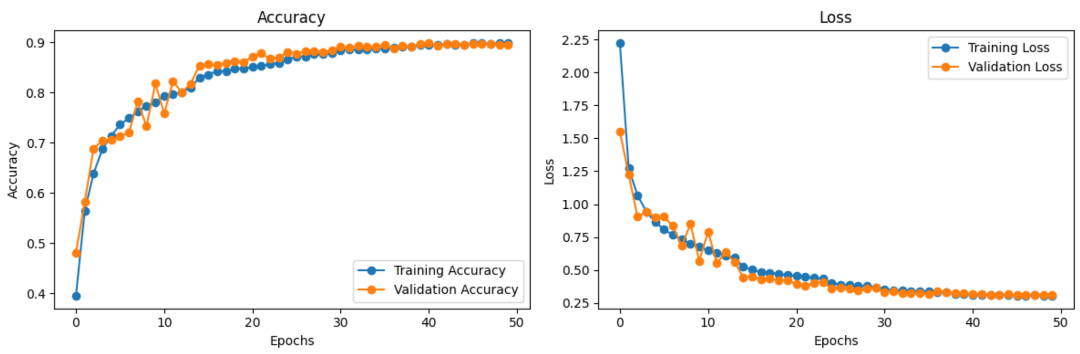
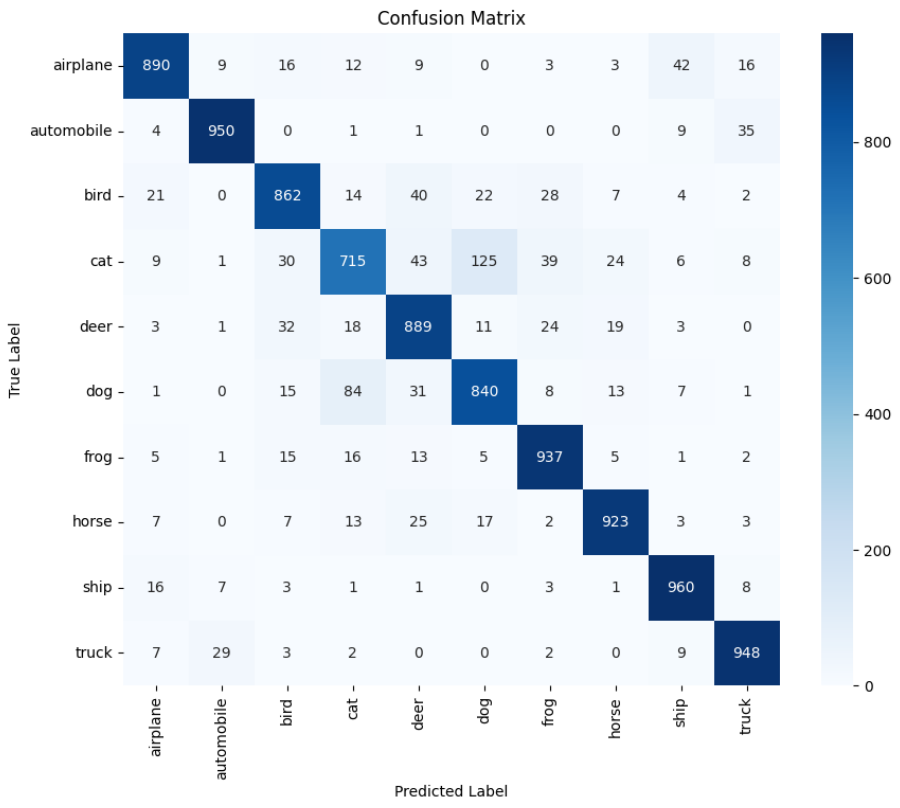


Figure 5: Regularized Model Curves



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	890	9	16	12	9	0	3	3	42	16
automobile	4	950	0	1	1	0	0	0	9	35
bird	21	0	862	14	40	22	28	7	4	2
cat	9	1	30	715	43	125	39	24	6	8
deer	3	1	32	18	889	11	24	19	3	0
dog	1	0	15	84	31	840	8	13	7	1
frog	5	1	15	16	13	5	937	5	1	2
horse	7	0	7	13	25	17	2	923	3	3
ship	16	7	3	1	1	0	3	1	960	8
truck	7	29	3	2	0	0	2	0	9	948

Figure 6: Regularized Model Confusion Matrix

4 Conclusion

This project successfully developed a deep learning pipeline for classifying images from the CIFAR-10 dataset, achieving the stated objectives. A basic model, a regularized model, and a hyperparameter-tuned model were implemented and evaluated. Among these, the regularized model achieved the highest test accuracy of 89

The inclusion of evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices provided a comprehensive understanding of the model's strengths and weaknesses. The integration of GitHub ensured efficient version control and collaboration, aligning with the project goals.

Future improvements could involve experimenting with more advanced architectures, such as ResNet or EfficientNet, to further enhance performance. Transfer learning could also be explored, particularly for smaller datasets, to leverage pre-trained weights. Additionally, fine-tuning the data augmentation strategies and conducting a more exhaustive hyperparameter search may yield better results. Expanding the project to include more diverse datasets would also test the model's adaptability and robustness in handling real-world scenarios.