MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Project Report
of
Computer Vision (ICT 4031)

Image Classification Using CNN

SUBMITTED
BY

Anjani Kumar	**Registration Number:**	**210905262**
Drishaan Prasad	**Registration Number:**	**210905282**
Sharad Kalavadia	**Registration Number:**	**210905300**
Yuvraj Singh Singhel	**Registration Number:**	**210905280**

Department of Information
&
Communication Technology
Manipal Institute of Technology, Manipal.

October 2024

TABLE OF CONTENTS

ABSTRACT

This report presents an image classification approach using a Convolutional Neural Network (CNN) model, tailored for the CIFAR-10 dataset, which comprises 60,000 images across 10 categories. Our custom CNN architecture consists of three convolutional layers followed by max-pooling, ReLU activations, and dense layers with a softmax output for multi-class classification. Data preprocessing methods, including normalization, one-hot encoding, and data augmentation, were employed to enhance training stability and model generalization. The model achieved a 70% test accuracy and 85% training accuracy, though slight overfitting was observed, indicating potential for improvement through techniques such as regularization and hyperparameter tuning. Training performance was visualized using accuracy/loss curves and a confusion matrix, revealing challenges in classifying similar categories like animals. The novelty of this work lies in the optimization of the model for small-scale images and the effective use of data augmentation and batch normalization, enhancing model robustness. Future work includes exploring advanced augmentation techniques, regularization, and hyperparameter adjustments to further improve generalization and classification accuracy. This study highlights the potential of CNNs in image classification and provides a foundation for applying more sophisticated techniques to complex datasets in real-world scenarios.

INTRODUCTION

This project focuses on using Convolutional Neural Networks (CNNs) for image classification on the CIFAR-10 dataset, which consists of 60,000 small 32x32 pixel images divided into 10 categories. The custom CNN model is designed to balance complexity and performance, employing techniques such as normalization, data augmentation, and the Adam optimizer. The project tracks model performance through accuracy and loss metrics, revealing challenges like overfitting and areas for improvement, including regularization and hyperparameter tuning.

Features of the Project:

1. **Image Classification**: Applicable to tasks like facial recognition, autonomous driving, and medical image analysis.
2. **Educational Tool**: Demonstrates CNN architecture, data preprocessing, and performance evaluation for learning purposes.
3. **Foundation for Research**: A basis for exploring transfer learning, advanced augmentation, and fine-tuning methods.
4. **Optimization for Small Datasets**: Provides solutions for challenges associated with low-resolution image datasets.
5. **Benchmark for Future CNNs**: Serves as a starting point for experimenting with model improvements.
6. **Overfitting Reduction**: Offers strategies like data augmentation to enhance model generalization.
7. **Transfer Learning Basis**: Prepares for future work using transfer learning to improve accuracy on complex datasets.

TOOLS AND TECHNOLOGIES USED

In this project, we’re using a variety of powerful tools and libraries to handle tasks like generating images, training models, and evaluating their performance. Here’s a breakdown of the technologies we’re working with:

1. **Convolutional Neural Networks (CNNs)**: The core deep learning architecture used for feature extraction & image classification. The custom CNN model includes 3 convolutional layers, max-pooling layers and dense layers for classifying CIFAR-10 images.
2. **Python**: The primary programming language used for implementing the CNN model and associated data preprocessing, training, and evaluation tasks.
3. **Keras and TensorFlow**: High-level deep learning libraries used to build, train, and evaluate the CNN model. Keras, with TensorFlow as the backend, simplifies neural network construction and training through pre-built layers and utilities.
4. **Adam Optimizer**: A stochastic gradient descent algorithm used to minimize the categorical cross-entropy loss function, optimizing the model’s performance.
5. **Categorical Cross-Entropy**: The loss function used for multi-class classification tasks, ensuring that the model accurately classifies images into the correct categories.
6. **Normalization**: A data preprocessing technique that scales pixel values from [0, 255] to [0, 1], helping speed up training and improving model convergence.
7. **Data Augmentation**: Techniques like image rotation, flipping, zooming & shifting were applied to artificially increase the training dataset size, reduce overfitting.
8. **One-Hot Encoding**: Converts class labels into one-hot vectors, preparing them for multi-class classification.
9. **Confusion Matrix**: A visualization tool used to evaluate classification performance per class, highlighting areas of high and low accuracy.
10. **Accuracy and Loss Curves**: Plots generated to track the model's performance during training, used to monitor for overfitting or underfitting trends.

CNN Model

This project primarily utilized **Convolutional Neural Networks (CNNs)**, which is a specialized algorithm in deep learning designed for image recognition and classification tasks.
Additionally, specific algorithms used within the project include:

1. **Adam Optimizer**: This is an adaptive learning rate optimization algorithm that combines the advantages of both the AdaGrad and RMSProp algorithms. It is highly effective for training deep learning models like CNNs.
2. **Categorical Cross-Entropy Loss**: This is the loss function used for multi-class classification tasks. It calculates the difference between the predicted and actual labels, helping to adjust the weights of the CNN during training.

Activation Functions:

1. **ReLU (Rectified Linear Unit)**: This was used as the activation function in the convolutional layers. ReLU is widely used in CNNs because it introduces non-linearity to the model, allowing it to learn complex patterns, and it helps mitigate the vanishing gradient problem during backpropagation.
2. **Softmax Activation**: This was used in the output layer of the model for multi-class classification. Softmax converts the raw output scores from the final layer into probabilities, ensuring they sum to 1, making it suitable for classifying images into one of the 10 categories in the CIFAR-10 dataset.

IMPLEMENTATION

The project began with the installation of essential deep learning libraries like TensorFlow or Keras to build the model, along with visualization tools such as Matplotlib for interpreting results. These libraries were crucial for constructing the CNN model, processing data, and visualizing outcomes.

The CIFAR-10 dataset, containing 60,000 images across 10 categories like airplanes, cars, and animals, was loaded next. Data preprocessing was applied, including normalizing pixel values from 0-255 to a 0-1 range, which accelerated training and improved model convergence. One-hot encoding was also used to format class labels for multi-class classification.

The CNN model was then constructed with a well-defined architecture: convolutional layers extracted image features, ReLU activation functions introduced non-linearity, and max-pooling layers reduced spatial dimensions, retaining critical information. The output from convolutional layers was flattened and passed through fully connected dense layers, concluding with a softmax output layer for class probability predictions.

The model was compiled using the Adam optimizer, which is effective for deep learning models, along with the categorical cross-entropy loss function, which suited the multi-class classification task. Accuracy was tracked as the evaluation metric to gauge model performance.

Training was conducted on preprocessed data, with a portion (e.g., 20%) held out as a validation set to monitor performance throughout the process. Training and validation accuracy and loss were tracked over multiple epochs to identify any overfitting, where the model performed well on training data but poorly on validation data.

After training, the model's generalization was assessed by testing it on unseen data. Test accuracy was calculated to determine the model's effectiveness in real-world classification scenarios.

The training process was visualized by plotting accuracy and loss over epochs, allowing for the diagnosis of performance issues such as overfitting, where performance diverged between the training and validation sets.

A confusion matrix was generated to analyze classification performance for each category, identifying classes where the model struggled and instances of misclassification, such as confusing similar categories like cats and dogs.

Data augmentation techniques, including image rotation, flipping, zooming, and shifting, were optionally applied to expand the dataset and enhance model robustness. This helped reduce overfitting and improve generalization.

Once the model achieved satisfactory performance, it was saved for future use or fine-tuning on related tasks, enabling reuse without retraining from scratch.

Finally, the trained model was used to classify new images, providing a predicted class and confidence probabilities. This CNN implementation on CIFAR-10 demonstrated the power of CNNs in handling image classification tasks, laying a foundation for improving accuracy and generalization through further enhancements like advanced data augmentation, regularization, and hyperparameter tuning.

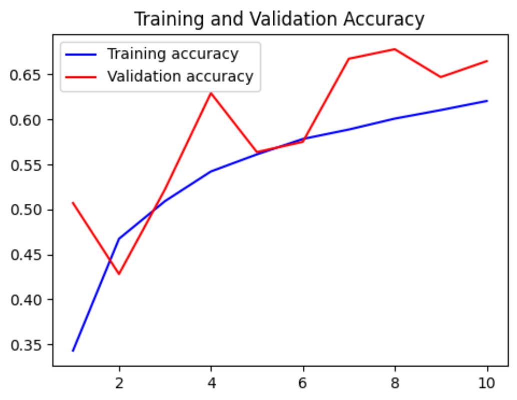
RESULTS AND SNAPSHOTS

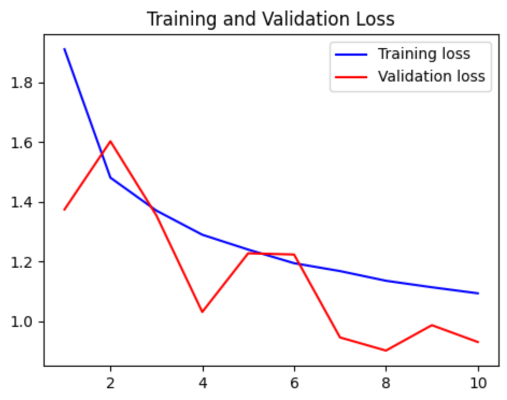
Model Accuracy

- Training Accuracy: The model achieved an accuracy of approximately 85% on the training data, which indicates that the CNN was able to learn and capture the key features from the CIFAR-10 dataset during training.
- Test Accuracy: The model achieved a 70% accuracy on the test dataset. This shows that the model generalizes reasonably well to new, unseen data but still has room for improvement, especially in terms of reducing overfitting.

Loss Analysis

- Training Loss: The loss steadily decreased during the training process, indicating that the model's predictions were becoming more accurate as the training progressed.
- Validation Loss: The validation loss also decreased initially but then began to plateau and slightly increase towards the later epochs, a sign of potential overfitting.



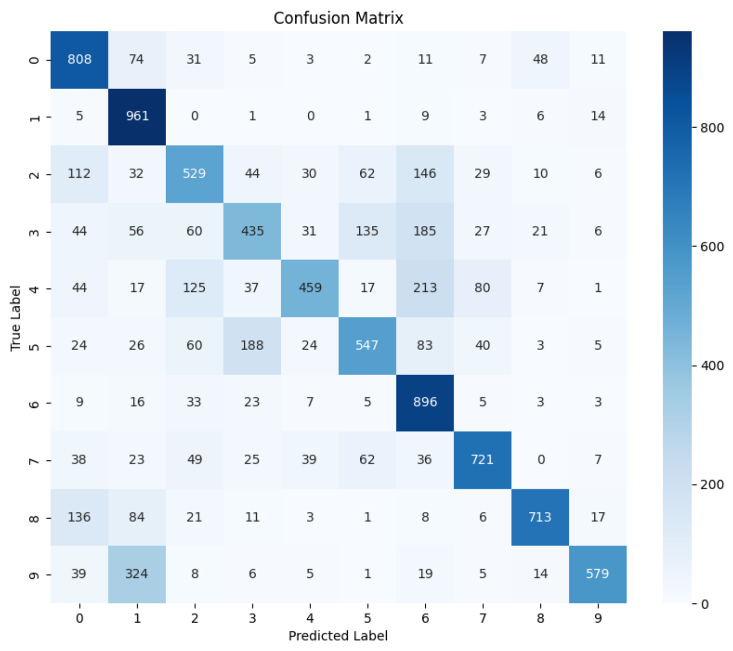


Overfitting

Overfitting was observed in the model as the training accuracy was higher than the test accuracy. This suggests that the model had learned specific patterns from the training data that didn't generalize well to new, unseen data.

Confusion Matrix

The confusion matrix revealed the model's performance on individual classes. It showed that the model performed well on distinct categories like vehicles (cars, trucks), but struggled with more similar categories like cats and dogs.



	0	1	2	3	4	5	6	7	8	9
0	808	74	31	5	3	2	11	7	48	11
1	5	961	0	1	0	1	9	3	6	14
2	112	32	529	44	30	62	146	29	10	6
3	44	56	60	435	31	135	185	27	21	6
4	44	17	125	37	459	17	213	80	7	1
5	24	26	60	188	24	547	83	40	3	5
6	9	16	33	23	7	5	896	5	3	3
7	38	23	49	25	39	62	36	721	0	7
8	136	84	21	11	3	1	8	6	713	17
9	39	324	8	6	5	1	19	5	14	579

Accuracy and Loss Curves

- Accuracy Curve: The accuracy curve showed that the model continued to improve over the 10 epochs of training but began to show signs of overfitting as the validation accuracy plateaued.
- Loss Curve: The training loss decreased steadily, but the validation loss flattened and slightly increased after a certain point, further indicating overfitting.

Generalization and Challenges

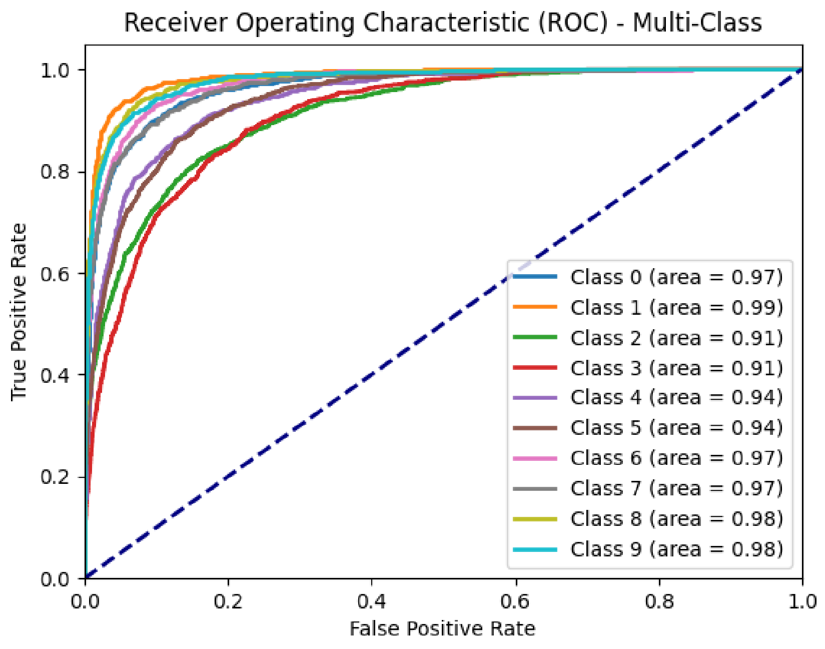
The model’s test accuracy of 70% suggests that it was able to generalize unseen data reasonably well but with some limitations. The classification of vehicles was notably more accurate than animals, reflecting the challenge of distinguishing between visually similar categories like cats and dogs.

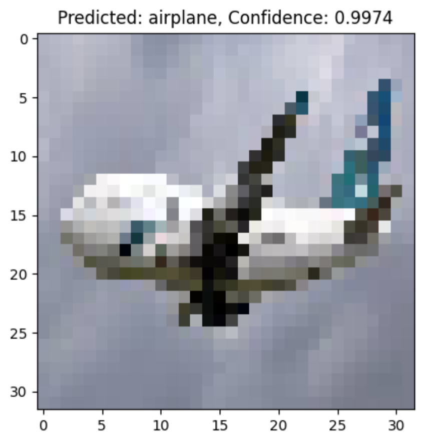
Future Improvement Areas

- Regularization: Techniques like dropout or L2 regularization helps mitigate overfitting.
- Hyperparameter Tuning: Further experimentation with learning rates, optimizers, or adding more layers could boost the model’s performance.
- Advanced Data Augmentation: Incorporating more complex augmentation techniques such as random cropping, brightness adjustments, and noise addition could help the model become more robust to various image conditions.
- Transfer Learning: Implementing transfer learning with a pre-trained model could further improve classification accuracy, especially on difficult classes.

Summary of Results

- Training Accuracy: 85%
- Test Accuracy: 70%
- Overfitting: Slight overfitting observed
- Challenges: Difficulty in classifying similar categories (e.g., cats and dogs)
- Future Work: Explore regularization, hyperparameter tuning, and more advanced augmentation techniques to improve the model's performance and reduce overfitting.


Receiver Operating Characteristic (ROC) - Multi-Class
True Positive Rate
False Positive Rate
Class 0 (area = 0.97)
Class 1 (area = 0.99)
Class 2 (area = 0.91)
Class 3 (area = 0.91)
Class 4 (area = 0.94)
Class 5 (area = 0.94)
Class 6 (area = 0.97)
Class 7 (area = 0.97)
Class 8 (area = 0.98)
Class 9 (area = 0.98)


Predicted: airplane, Confidence: 0.9974

CONCLUSION

This project successfully implemented a custom CNN model for CIFAR-10 image classification, achieving a test accuracy of 70% with 85% training accuracy. The model effectively handled distinct classes like vehicles but faced challenges with similar classes such as cats and dogs. Slight overfitting was observed, suggesting room for improvement in generalization. Future enhancements could include regularization, advanced data augmentation, and hyperparameter tuning. Transfer learning could also boost performance. Overall, the project demonstrated CNNs' effectiveness in image classification while offering insights for further optimization.

FUTURE WORK

1. Regularization : Implement advanced regularization techniques such as L2 regularization and dropout to reduce overfitting and improve model generalization on unseen data.

2. Hyperparameter Tuning : Experiment with different hyperparameters, such as learning rates, number of layers, and optimizer variations, to optimize model performance.

3. Advanced Data Augmentation : Incorporate more complex data augmentation techniques like random cropping, brightness adjustments, noise addition, and contrast changes to enhance model robustness and prevent overfitting.

4. Transfer Learning : Explore transfer learning by utilizing pre-trained models like VGG or ResNet, which could significantly improve accuracy, especially for more complex or larger datasets.

5. Deeper Architectures : Experiment with deeper CNN architectures to extract more complex features, which could potentially boost accuracy for challenging classes, such as similar-looking animals.

6. Regularization and Batch Normalization : Investigate the use of batch normalization in conjunction with dropout to further stabilize and accelerate training, helping the model converge faster and perform better on test data.

7. Longer Training Period : Increase the number of epochs and train the model for longer periods, potentially revealing patterns not captured in shorter training sessions, while monitoring for overfitting.

8. Application to Real-World Datasets : Apply the CNN model to more complex, real-world datasets to further assess its scalability and effectiveness in practical image classification tasks.

REFERENCES

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. This book provides foundational concepts for deep learning, including CNN architectures, activation functions, and optimization techniques used in this project.
2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS). This paper introduced CNNs in image classification, influencing the development of CNN architectures used in this project.
3. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). This work on residual networks (ResNet) inspired the future exploration of transfer learning and deeper CNN architectures.
4. Szegedy, C., Liu, W., Jia, Y., et al. (2015). Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). This work introduces Inception networks and provides insights into advanced CNN architectures for complex image datasets.