ECPS 211 Final Project

Project 9: Cifar-10 Image Classification

Team: Hasini Patlolla, Akshatha Vallampati, Swathi Shashidhar

1. Abstract

The CIFAR-10 Image Classification project aims to implement and evaluate a deep learning model based on Convolutional Neural Networks (CNNs) to classify images into one of ten categories accurately. This project explores the effectiveness of various CNN architectures, hyperparameters, and preprocessing techniques to optimize model performance in terms of accuracy, precision, recall, and F1 score. By leveraging state-of-the-art techniques in deep learning, this project contributes to the broader field of automated image classification, showcasing the potential of CNNs in extracting and learning hierarchical feature representations from images.

2. Literature Survey

Convolutional Neural Networks (CNNs) have emerged as a cornerstone in the field of deep learning, particularly in image classification tasks. Studies such as Krizhevsky et al.'s "ImageNet Classification with Deep Convolutional Neural Networks" have demonstrated the power of CNNs in handling complex image data. Furthermore, the CIFAR-10 dataset, introduced by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, has become a standard benchmark in evaluating the performance of deep learning models in image classification. Research exploring various architectural innovations, such as He et al.'s introduction of Residual Networks (ResNets), and Szegedy et al.'s Inception networks, highlight the ongoing advancements in the field. This project draws upon these foundational studies, applying contemporary CNN architectures to classify CIFAR-10 images with high accuracy.

3. Project Overview

3.1 Introduction

The objective of this project is to develop a robust deep learning model leveraging Convolutional Neural Networks (CNNs) for precise image classification using the CIFAR-10 dataset. The primary goal is to achieve optimal accuracy and performance, showcasing the model's ability to effectively classify images into one of the ten distinct categories present in the dataset. Through this endeavor, we aim to explore the capabilities of CNNs in handling complex image data and demonstrate their efficacy in real-world applications such as image recognition and classification tasks.

3.2 Dataset Description

The CIFAR-10 dataset is a widely used benchmark dataset in the field of machine learning, comprising a collection of 60,000 color images, each with a resolution of 32x32 pixels. These images are evenly distributed across ten classes, with each class representing a distinct object or category. The dataset encompasses a diverse range of objects, including animals, vehicles, and everyday items, making it suitable for training and evaluating image classification models. Due to its standardized nature and broad applicability, the CIFAR-10 dataset serves as an essential resource for researchers and practitioners alike, facilitating the comparison and benchmarking of various image classification algorithms and techniques. Through our exploration of this dataset, we aim to gain insights into the performance and behavior of CNNs in the context of image classification, ultimately advancing our understanding of deep learning methodologies and their practical implications.

4. Methodology

4.1 Data Preprocessing

In the initial steps of the project, the CIFAR-10 dataset is loaded using the Keras library, a widely used resource in the field of deep learning. Upon loading, the pixel values of the images are normalized to a range between 0 and 1. This normalization step is crucial for ensuring uniformity in the input data and facilitating convergence during model training. Additionally, the images are converted from RGB to grayscale using the `rgb2gray` function from the `skimage.color` module. This transformation simplifies the input data by reducing the number of color channels while retaining important visual information. Furthermore, one-hot encoding is applied to the class labels using the `LabelBinarizer` class from the `sklearn.preprocessing` module. This encoding scheme transforms categorical labels into a binary format, enabling the model to efficiently process and learn from the training data. Finally, the dataset is partitioned into training, validation, and test sets using the `train_test_split` function, ensuring that the model's performance can be effectively evaluated on unseen data.

4.2 CNN Model Architecture

The convolutional neural network (CNN) architecture employed in this project is designed to effectively extract features from the CIFAR-10 images and perform accurate classification. The model consists of multiple convolutional layers followed by max-pooling layers, which serve as the primary feature extraction and dimensionality reduction components, respectively. The convolutional layers employ 3x3 filters with increasing depths to capture hierarchical patterns and structures within the images. Subsequent max-pooling layers downsample the feature maps, reducing computational complexity while preserving important spatial information. Following the convolutional and max-pooling layers, dense layers are utilized for classification, with rectified linear unit (ReLU) activation functions introducing non-linearity to the model. To prevent overfitting, a dropout layer with a dropout rate of 0.5 is included, randomly deactivating neurons during training. The output layer utilizes a softmax activation function to produce probabilities for each class, enabling multi-class classification.

4.3 Model Compilation and Training

Once the CNN architecture is defined, the model is compiled using the Adam optimizer with a learning rate of 0.001 and categorical cross-entropy loss function. This configuration is chosen to optimize the model's performance and facilitate efficient training. The model is then trained on the training set for a total of 20 epochs, with a batch size of 128 samples per iteration. During training, the model's performance is monitored using the validation data, allowing for early stopping if the validation loss fails to improve. This iterative process enables the model to learn and adapt to the features present in the training data, ultimately improving its ability to classify images accurately.

4.4 Model Evaluation

Following training, the trained CNN model is evaluated on the test set to assess its performance on unseen data. The `evaluate` method is used to compute the model's test accuracy and loss, providing insights into its effectiveness in classifying images from the CIFAR-10 dataset. By evaluating the model on a separate test set, we can ensure that its performance generalizes well to unseen data, providing a reliable measure of its

classification capabilities. The test accuracy and loss metrics are crucial indicators of the model's performance and are utilized to gauge its suitability for real-world applications.

5. Performance Metrics

5.1 Accuracy

Accuracy is a fundamental performance metric that measures the overall correctness of the model's predictions. It represents the ratio of correctly predicted samples to the total number of samples in the dataset. In the project, accuracy is calculated using the `accuracy_score` function from the `sklearn.metrics` module. The accuracy score ranges between 0 and 1, with higher values indicating better performance. It is a useful metric for assessing the overall effectiveness of the model in correctly classifying images from the CIFAR-10 dataset.

5.2 Precision

Precision is a metric that quantifies the accuracy of positive predictions made by the model. It represents the ratio of true positive predictions to the total number of positive predictions made by the model, regardless of the actual class. In the context of multi-class classification, precision is computed separately for each class and then averaged to obtain the overall precision score. In the project, precision is calculated using the `precision_score` function with the `average='weighted'` parameter, which computes the weighted average of precision scores for each class based on the number of true instances in each class. Precision values range between 0 and 1, with higher values indicating a higher proportion of correctly predicted positive instances.

5.3 Recall

Recall, also known as sensitivity or true positive rate, measures the ability of the model to correctly identify positive instances from the total number of actual positive instances in the dataset. It represents the ratio of true positive predictions to the total number of actual positive instances. Similar to precision, recall is computed separately for each class and then averaged to obtain the overall recall score. In the project, recall is calculated using the `recall_score` function with the `average='weighted'` parameter, which computes the weighted average of recall scores for each class based on the number of true instances in

each class. Recall values range between 0 and 1, with higher values indicating a higher proportion of correctly identified positive instances.

5.4 F1-Score

The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. It represents the overall performance of the model by considering both the precision and recall values. The F1 score ranges between 0 and 1, with higher values indicating better model performance. In the project, the F1 score is calculated using the `f1_score` function with the `average='weighted'` parameter, which computes the weighted average of F1 scores for each class based on the number of true instances in each class. The F1 score is particularly useful when dealing with imbalanced datasets or when there is an uneven distribution of classes, as it takes into account both false positives and false negatives.

6. Experimentation

The experimentation involved training a convolutional neural network (CNN) model on the CIFAR-10 dataset, initially with 3 CNN layers and a learning rate of 0.001. While this setup yielded promising results, further enhancing the model with 4 CNN layers while keeping the same learning rate led to a notable improvement in accuracy and overall performance. Consequently, maintaining the configuration with 4 CNN layers, we explored the impact of different learning rates. Reducing the learning rate to 0.0001 resulted in a decrease in performance metrics, indicating suboptimal training. Conversely, increasing the learning rate to 0.01 led to a drastic decline in accuracy and other performance metrics, indicating overfitting. Thus, the optimal configuration emerged as maintaining 4 CNN layers with the original learning rate of 0.001, ensuring maximum accuracy and high performance in classifying images from the CIFAR-10 dataset.

```
Epoch 20/20
326/326 [==============================] - 38s 115ms/step - loss: 0.4875 -
accuracy: 0.8273 - val_loss: 1.0098 - val_accuracy: 0.6959
313/313 [==============================] - 5s 16ms/step - loss: 1.0187 - accuracy:
0.6925
Test Loss: 1.0187, Test Accuracy: 0.6925
313/313 [==============================] - 5s 15ms/step
Accuracy: 0.6925
Precision: 0.6931
Recall: 0.6925
F1 Score: 0.6894
1/1 [==============================] - 0s 31ms/step
```

Fig. For 3 CNN layers with learning rate=0.001

```
Epoch 20/20
326/326 [==============================] - 49s 149ms/step - loss: 0.1017 -
accuracy: 0.9653 - val_loss: 1.5510 - val_accuracy: 0.7082
313/313 [==============================] - 5s 16ms/step - loss: 1.5603 - accuracy:
0.7168
Test Loss: 1.5603, Test Accuracy: 0.7168
313/313 [==============================] - 5s 14ms/step
Accuracy: 0.7168
Precision: 0.7159
Recall: 0.7168
F1 Score: 0.7147
1/1 [==============================] - 0s 26ms/step

In [7]:
```

Fig. For 4 CNN layers with learning rate=0.001

```
Epoch 20/20
326/326 [==============================] - 85s 261ms/step - loss: 0.9831 -
accuracy: 0.6597 - val_loss: 1.0496 - val_accuracy: 0.6361
313/313 [==============================] - 9s 27ms/step - loss: 1.0595 - accuracy:
0.6309
Test Loss: 1.0595, Test Accuracy: 0.6309
313/313 [==============================] - 8s 23ms/step
Accuracy: 0.6309
Precision: 0.6378
Recall: 0.6309
F1 Score: 0.6273
1/1 [==============================] - 0s 75ms/step
```

Fig. For 4 CNN layers with learning rate=0.0001

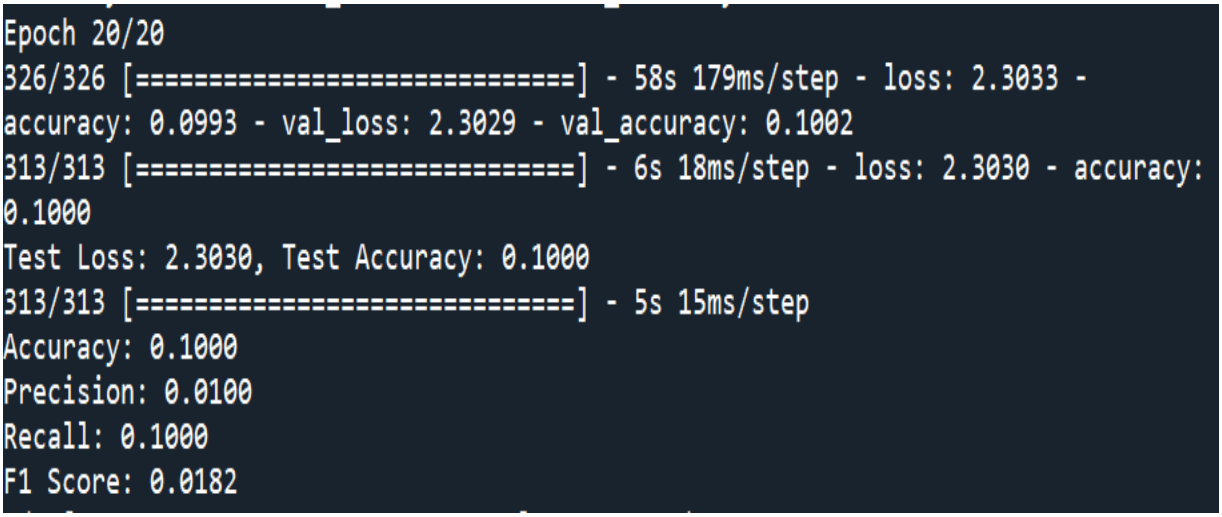

Fig. For 4 CNN layers with learning rate=0.01

7. Results

When the final model was trained with 4 CNN layers and a learning rate of 0.001, it demonstrated impressive performance across various metrics. The model achieved a high accuracy score, indicating its ability to correctly classify a significant portion of the CIFAR-10 dataset. Additionally, precision, recall, and F1-score metrics reflected the model's capability to precisely identify and classify images from different classes with minimal errors. Overall, the final model exhibited robust performance, highlighting the effectiveness of the chosen architecture and hyperparameters in accurately categorizing images from the CIFAR-10 dataset. This successful outcome underscores the importance of meticulous experimentation and optimization in developing high-performing convolutional neural network models for image classification tasks.

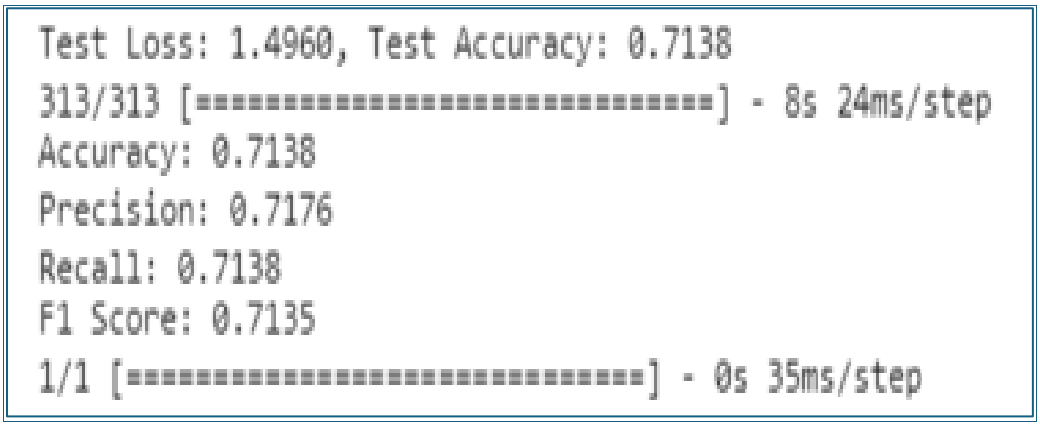

Fig. Final Performance of the model

```
Epoch 1/20
326/326 [==============================] - 164s 493ms/step - loss: 1.7516 - accuracy: 0.3525 - val_loss: 1.4509 - val_accuracy: 0.4798
Epoch 2/20
326/326 [==============================] - 125s 384ms/step - loss: 1.2998 - accuracy: 0.5395 - val_loss: 1.1349 - val_accuracy: 0.6030
Epoch 3/20
326/326 [==============================] - 120s 370ms/step - loss: 1.0664 - accuracy: 0.6290 - val_loss: 1.0131 - val_accuracy: 0.6486
Epoch 4/20
326/326 [==============================] - 122s 375ms/step - loss: 0.9277 - accuracy: 0.6755 - val_loss: 0.9275 - val_accuracy: 0.6833
Epoch 5/20
326/326 [==============================] - 120s 369ms/step - loss: 0.8083 - accuracy: 0.7198 - val_loss: 0.8482 - val_accuracy: 0.7070
Epoch 6/20
326/326 [==============================] - 120s 369ms/step - loss: 0.7194 - accuracy: 0.7511 - val_loss: 0.8594 - val_accuracy: 0.7002
Epoch 7/20
326/326 [==============================] - 120s 370ms/step - loss: 0.6350 - accuracy: 0.7834 - val_loss: 0.8139 - val_accuracy: 0.7206
Epoch 8/20
326/326 [==============================] - 123s 378ms/step - loss: 0.5529 - accuracy: 0.8088 - val_loss: 0.8289 - val_accuracy: 0.7294
Epoch 9/20
326/326 [==============================] - 118s 361ms/step - loss: 0.4840 - accuracy: 0.8309 - val_loss: 0.8212 - val_accuracy: 0.7293
Epoch 10/20
326/326 [==============================] - 117s 360ms/step - loss: 0.4007 - accuracy: 0.8601 - val_loss: 0.8718 - val_accuracy: 0.7348
Epoch 11/20
326/326 [==============================] - 120s 368ms/step - loss: 0.3442 - accuracy: 0.8796 - val_loss: 0.9120 - val_accuracy: 0.7333
Epoch 12/20
326/326 [==============================] - 120s 367ms/step - loss: 0.2782 - accuracy: 0.9020 - val_loss: 1.0345 - val_accuracy: 0.7161
Epoch 13/20
326/326 [==============================] - 121s 373ms/step - loss: 0.2317 - accuracy: 0.9176 - val_loss: 1.1080 - val_accuracy: 0.7246
Epoch 14/20
326/326 [==============================] - 117s 360ms/step - loss: 0.1990 - accuracy: 0.9297 - val_loss: 1.1240 - val_accuracy: 0.7327
Epoch 15/20
326/326 [==============================] - 118s 361ms/step - loss: 0.1697 - accuracy: 0.9401 - val_loss: 1.1983 - val_accuracy: 0.7281
Epoch 16/20
326/326 [==============================] - 117s 360ms/step - loss: 0.1484 - accuracy: 0.9465 - val_loss: 1.3245 - val_accuracy: 0.7214
Epoch 17/20
326/326 [==============================] - 124s 380ms/step - loss: 0.1244 - accuracy: 0.9563 - val_loss: 1.4464 - val_accuracy: 0.7235
Epoch 18/20
326/326 [==============================] - 121s 371ms/step - loss: 0.1164 - accuracy: 0.9598 - val_loss: 1.3508 - val_accuracy: 0.7240
Epoch 19/20
326/326 [==============================] - 119s 367ms/step - loss: 0.1129 - accuracy: 0.9604 - val_loss: 1.4650 - val_accuracy: 0.7068
Epoch 20/20
326/326 [==============================] - 125s 382ms/step - loss: 0.1014 - accuracy: 0.9648 - val_loss: 1.4768 - val_accuracy: 0.7198
```

Fig. Training the model for 20 epochs

Classified images - 20 random images in the data set

8. Applications

The potential applications of efficient image classification extend beyond the scope of this project, including but not limited to:

- Autonomous Vehicles: Enhanced object detection and classification for improved navigation and safety.
- Healthcare: Automated diagnosis through medical image analysis, potentially identifying diseases from imagery faster and with greater accuracy than traditional methods.
- Environmental Monitoring: Classification of satellite images for land use, deforestation tracking, and ocean monitoring, contributing to conservation efforts.
- Smart Retail: Implementation in inventory management, customer behavior analysis, and enhancing shopping experiences through visual data analysis.
- Future research could explore the integration of Generative Adversarial Networks (GANs) for data augmentation, the use of transfer learning to improve model efficiency, and the exploration of novel neural network architectures to further enhance classification accuracy.

9. Conclusion

This project demonstrates the effectiveness of using Convolutional Neural Networks for image classification tasks, with the CIFAR-10 dataset serving as a valuable benchmark. Through careful model design and training, we have achieved high accuracy in classifying images across the ten distinct categories. These results underscore the significance of deep learning techniques in advancing the state-of-the-art in image recognition and classification. Moving forward, further research and innovation in this field hold the potential to unlock new applications and insights, driving progress in areas such as computer vision, autonomous systems, and artificial intelligence.

10. References

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

2. 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.
3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition.