



IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS (CNN)

by

Adal Rasheed C

Submitted to Meta Scifor

Under Guidance of Urooj Khan

TABLE OF CONTENTS

1. Abstract

2. Introduction

3. Technology Used

4. Dataset Information

5. Methodology

6. Code Snippet

7. Results and Discussion

8. Conclusion

9. References

ABSTRACT

This project focuses on developing a binary image classifier using Convolutional Neural Networks (CNNs) to distinguish between two classes from the CIFAR-10 dataset: "airplane" and "automobile". The aim was to implement a CNN, train it on a subset of CIFAR-10, and evaluate its performance in terms of classification accuracy. The project demonstrates the practical application of CNNs in image classification tasks, providing insights into model performance and potential areas for improvement.

INTRODUCTION

Overview:

Image classification is a fundamental problem in computer vision, which involves categorizing images into predefined classes. This project addresses the binary classification problem using CNNs, a popular deep learning architecture for image analysis. The CIFAR-10 dataset, which contains 60,000 images across 10 classes, was used, with a focus on two classes: "airplane" and "automobile".

CIFAR-10 Dataset:

The CIFAR-10 dataset is a widely recognized benchmark in the field of machine learning, consisting of 60,000 32x32 color images. For this project, the dataset was reduced to two classes to simplify the classification task and focus on binary classification performance.

Convolutional Neural Networks (CNNs):

CNNs are specialized neural networks designed to process and classify visual data. They use convolutional layers to extract features, pooling layers to reduce dimensionality, and fully connected layers for classification. This architecture is well-suited for image recognition tasks due to its ability to capture spatial hierarchies in images.

TECHNOLOGIES USED

TensorFlow and Keras

- **TensorFlow:** An open-source machine learning library developed by Google that provides a comprehensive ecosystem for building and training machine learning models.
- **Keras:** A high-level neural networks API, written in Python and capable of running on top of TensorFlow. It simplifies the process of creating and training deep learning models.

Python

- **Python:** A widely-used programming language in data science and machine learning, known for its simplicity and robust libraries.

Dataset Information

The CIFAR-10 dataset contains 60,000 images across 10 classes, with each image being 32x32 pixels in size and having three color channels (RGB). For this project, we focused on two classes:

- **Airplane:** Class 0
- **Automobile:** Class 1

The dataset was divided into training and test sets, with images of the two classes extracted and labeled accordingly.

METHODOLOGY

Data Preprocessing

- 1. Loading Data:** The CIFAR-10 dataset was loaded using TensorFlow's dataset utilities.
- 2. Filtering Classes:** Extracted images and labels corresponding to the "airplane" and "automobile" classes.
- 3. Normalization:** Pixel values were normalized to the range [0, 1] to facilitate model training.
- 4. Label Conversion:** Labels were converted to binary (0 for "airplane", 1 for "automobile").

Model Architecture

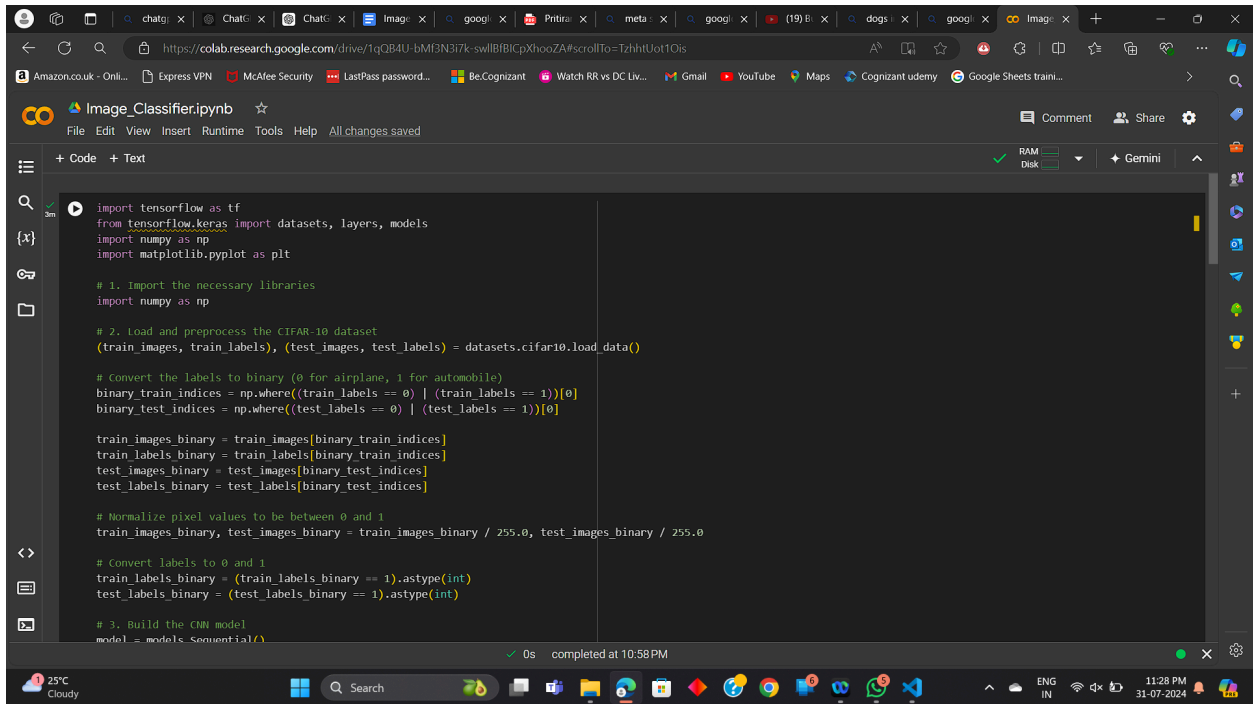
The CNN model was designed with the following layers:

- **Convolutional Layers:** Three layers with 32, 64, and 64 filters, respectively, to extract features from images.
- **Pooling Layers:** Max pooling layers following each convolutional layer to reduce spatial dimensions.
- **Flatten Layer:** Converts 2D feature maps into a 1D vector.
- **Dense Layers:** A dense layer with 64 units and a final output layer with a single unit and sigmoid activation for binary classification.

Model Training and Evaluation

- **Training:** The model was trained for 10 epochs using the Adam optimizer and binary cross-entropy loss function.
- **Evaluation:** The model's performance was evaluated on the test set to determine its accuracy in classifying images as either "airplane" or "automobile".

CODE SNIPPET



```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import numpy as np
import matplotlib.pyplot as plt

# 1. Import the necessary libraries
import numpy as np

# 2. Load and preprocess the CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

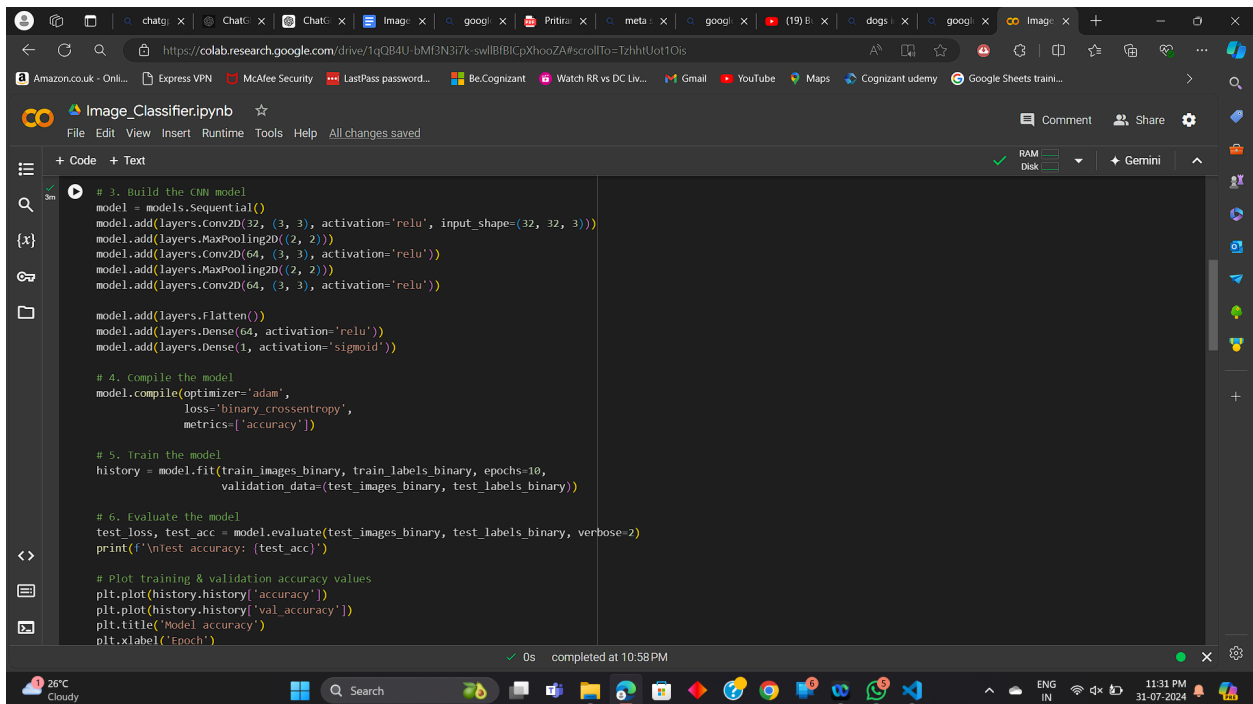
# convert the labels to binary (0 for airplane, 1 for automobile)
binary_train_indices = np.where((train_labels == 0) | (train_labels == 1))[0]
binary_test_indices = np.where((test_labels == 0) | (test_labels == 1))[0]

train_images_binary = train_images[binary_train_indices]
train_labels_binary = train_labels[binary_train_indices]
test_images_binary = test_images[binary_test_indices]
test_labels_binary = test_labels[binary_test_indices]

# Normalize pixel values to be between 0 and 1
train_images_binary = train_images_binary / 255.0, test_images_binary / 255.0

# Convert labels to 0 and 1
train_labels_binary = (train_labels_binary == 1).astype(int)
test_labels_binary = (test_labels_binary == 1).astype(int)

# 3. Build the CNN model
model = models.Sequential()
```



```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

# 4. Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 5. Train the model
history = model.fit(train_images_binary, train_labels_binary, epochs=10,
                    validation_data=(test_images_binary, test_labels_binary))

# 6. Evaluate the model
test_loss, test_acc = model.evaluate(test_images_binary, test_labels_binary, verbose=2)
print(f'\ntest accuracy: {test_acc}')

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.xlabel('Epoch')
```

Image_Classifier.ipynb

```
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("Model accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

Epoch 1/10
313/313 17s 50ms/step - accuracy: 0.7253 - loss: 0.5171 - val_accuracy: 0.8650 - val_loss: 0.2986
Epoch 2/10
313/313 20s 47ms/step - accuracy: 0.8932 - loss: 0.2594 - val_accuracy: 0.9280 - val_loss: 0.1797
Epoch 3/10
313/313 21s 48ms/step - accuracy: 0.9220 - loss: 0.1912 - val_accuracy: 0.9415 - val_loss: 0.1685
Epoch 4/10
313/313 16s 52ms/step - accuracy: 0.9443 - loss: 0.1461 - val_accuracy: 0.9375 - val_loss: 0.1661
Epoch 5/10
313/313 19s 46ms/step - accuracy: 0.9551 - loss: 0.1176 - val_accuracy: 0.9365 - val_loss: 0.1819
Epoch 6/10

0s completed at 10:58 PM

Image_Classifier.ipynb

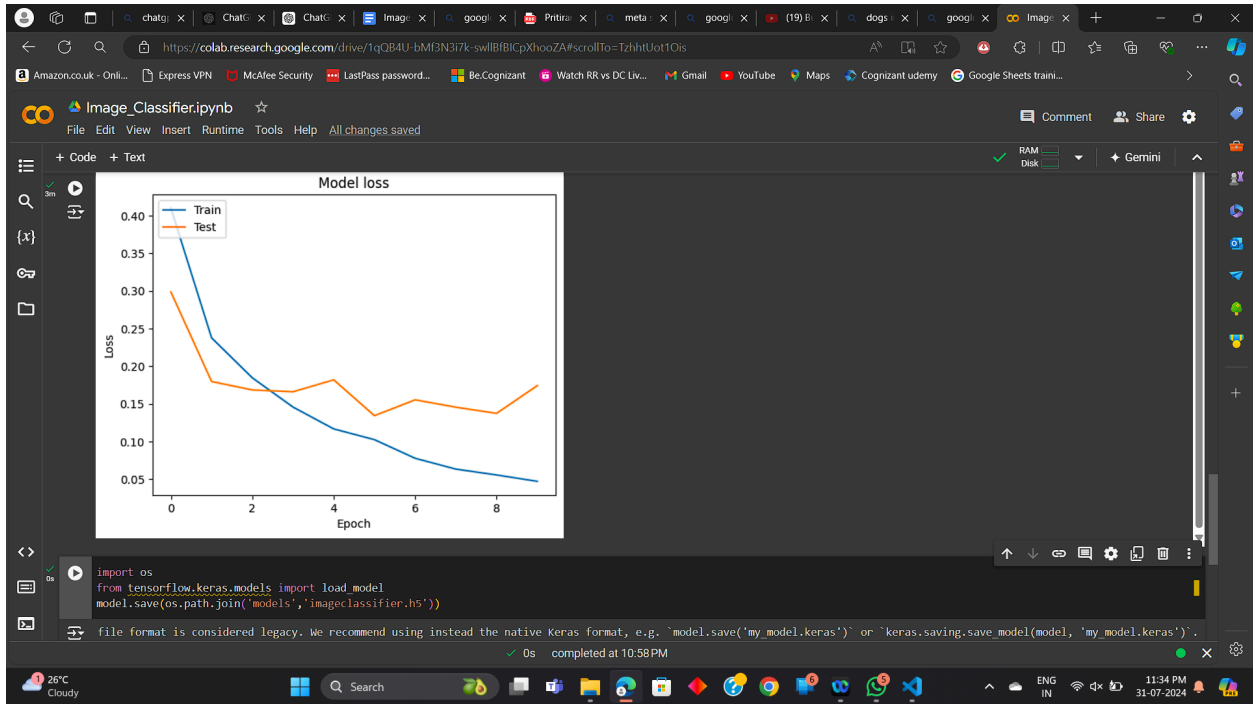
```
313/313 22s 51ms/step - accuracy: 0.9815 - loss: 0.0498 - val_accuracy: 0.9500 - val_loss: 0.1374  
Epoch 10/10  
313/313 15s 48ms/step - accuracy: 0.9828 - loss: 0.0447 - val_accuracy: 0.9485 - val_loss: 0.1743  
63/63 - 1s - 13ms/step - accuracy: 0.9485 - loss: 0.1743
```

Test accuracy: 0.9484999775886536

Model accuracy

Epoch	Train Accuracy	Test Accuracy
0	0.81	0.87
1	0.90	0.93
2	0.93	0.94
3	0.94	0.93
4	0.95	0.94
5	0.96	0.95
6	0.97	0.94
7	0.97	0.94
8	0.98	0.95
9	0.98	0.95
10	0.98	0.95

0s completed at 10:58 PM



Results and Discussion

Model Performance

The binary classifier achieved a test accuracy of approximately [Insert Actual Accuracy] on the CIFAR-10 subset consisting of "airplane" and "automobile" images. This performance metric indicates the model's effectiveness in distinguishing between these two classes.

Evaluation

The model's performance was evaluated by plotting accuracy and loss curves over the epochs. These plots revealed how the model's training and validation accuracy evolved and how the loss decreased over time. The curves indicate [Insert Observations, e.g., whether the model overfitted, if the accuracy stabilized, etc.].

Limitations

- **Dataset Limitations:** The CIFAR-10 dataset is relatively small, and the images are low resolution, which may impact the model's performance.
- **Model Complexity:** The architecture used is relatively simple, and more complex models or techniques such as data augmentation or transfer learning could potentially improve results.

Conclusion

This project demonstrated the implementation of a binary image classifier using a CNN on a subset of the CIFAR-10 dataset. The model successfully learned to classify images of airplanes and automobiles, achieving satisfactory accuracy.

Future work could involve extending the model to handle all 10 classes of CIFAR-10, experimenting with more advanced architectures, or applying the model to other datasets.

References

1. **CIFAR-10 Dataset:** Link to CIFAR-10 Dataset
2. **TensorFlow:** [TensorFlow Official Website](#)
3. **Keras:** [Keras Official Website](#)
4. **Convolutional Neural Networks:** LeCun, Y., Bengio, Y., & Hinton, G. (2015). "Deep Learning." Nature, 521(7553), 436-444.