Department of Information Technology

University of The Punjab

Gujranwala Campus

Project Name:

Image Classification using Convolutional Neural Networks (CNN) on CIFAR-10 Dataset

Project Documentation

Submitted to

Mrs. Fouqia Zafeer

Submitted by

Anas Karim (Group Lead)

BIT21028 (MORNING)

Certificate

This is to certify that project titled

"Image Classification using Convolutional Neural Networks (CNN) on CIFAR-10 Dataset "

has been completed by following students:

Group Leader: Anas Karim

Project Members:

Name	Roll No
Anas karim	BIT21028
Moon Masih	BIT21053
Umar Wakeel	BIT21014

the Seventh Semester, Bachelor of Information Technology in the year 2025 in partial fulfillment of the requirement to the award of the course **"COMPUTER VISION"**

Project Submitted to:

Mrs.Fouqia Zafeer

Location: University of the Punjab, Gujranwala Campus

Date: February 20, 2025

Table of Contents:

1. Objective:

The primary objective of this project is to build and train a Convolutional Neural Network (CNN) model to classify images from the CIFAR-10 dataset into one of the ten predefined categories. The project aims to demonstrate the application of deep learning techniques in image classification tasks and provide a foundation for further improvements and real-world applications.

2. Scope:

- The project focuses on classifying 32x32 color images into 10 categories: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck.
- The model is trained on the CIFAR-10 dataset, which contains 50,000 training images and 10,000 test images.
- The scope includes building a CNN model, training it, evaluating its performance, and discussing potential improvements.

3. System Architecture:

The system architecture consists of the following components:

1. **Input Layer**: Accepts 32x32 RGB images.
2. **Convolutional Layers**: Extract features from the images.
   a. Conv2D (32 filters, 3x3 kernel, ReLU activation)
   b. MaxPooling2D (2x2 pool size)
   c. Conv2D (64 filters, 3x3 kernel, ReLU activation)
   d. MaxPooling2D (2x2 pool size)
   e. Conv2D (64 filters, 3x3 kernel, ReLU activation)
3. **Flatten Layer**: Converts 2D feature maps into a 1D vector.
4. **Dense Layers**: Fully connected layers for classification.
   a. Dense (64 neurons, ReLU activation)
   b. Dense (10 neurons, Softmax activation for output probabilities)
5. **Output Layer**: Provides probabilities for each of the 10 classes.

4. Setup Instructions:

1. **Install Dependencies**:
   Ensure the following libraries are installed:
   code

pip install tensorflow numpy matplotlib

Download the Dataset:

The CIFAR-10 dataset is automatically downloaded using TensorFlow/Keras.

1. **Run the Code**:
   Execute the provided Python script to load the dataset, build the model, train it, and evaluate its performance.

5. Project Structure:

2. **Data Loading**: Load and normalize the CIFAR-10 dataset.
3. **Model Building**: Define the CNN architecture.
4. **Model Training**: Train the model on the training dataset.
5. **Model Evaluation**: Evaluate the model on the test dataset.
6. **Visualization**: Display sample images with their predicted labels

6. Model Training:

Training Process:

1. The model is trained for 10 epochs using the Adam optimizer and sparse categorical cross-entropy loss.
2. Training and validation accuracy/loss are monitored during training.

Code Explanation

#Load CIFAR-10 dataset

```
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
```

#Normalize pixel values (0-255 → 0-1)

x_train, x_test = x_train / 255.0, x_test / 255.0

#Build CNN Model

model = keras.Sequential([keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)), keras.layers.MaxPooling2D((2,2)), keras.layers.Conv2D(64, (3,3), activation='relu'), keras.layers.MaxPooling2D((2,2)), keras.layers.Conv2D(64, (3,3), activation='relu'), keras.layers.Flatten(), keras.layers.Dense(64, activation='relu'), keras.layers.Dense(10, activation='softmax')])

#Compile the Model

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

#Train the Model

model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

#Evaluate the Model

test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc:.2f}")

Model Summary:

The model.summary() function provides a detailed overview of the model architecture, including the number of parameters in each layer. Here's an example output:

Model: "sequential"

Layer (type) Output Shape Param

conv2d (Conv2D) (None, 30, 30, 32) 896

max_pooling2d (MaxPooling2D) (None, 15, 15, 32) 0

conv2d_1 (Conv2D) (None, 13, 13, 64) 18496

max_pooling2d_1 (MaxPooling2D) (None, 6, 6, 64) 0

conv2d_2 (Conv2D) (None, 4, 4, 64) 36928

flatten (Flatten) (None, 1024) 0

dense (Dense) (None, 64) 65600

dense_1 (Dense) (None, 10) 650

=== Total params: 122,570 Trainable params: 122,570 Non-trainable params: 0

Displaying Sample Images with Labels:

To visualize the dataset and understand the classes, we can display a few sample images from the CIFAR-10 dataset along with their corresponding labels.

Code to Display Sample Images:

import matplotlib.pyplot as plt

Class names in CIFAR-10

class_names = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']

Display sample images

plt.figure(figsize=(10, 5)) for i in range(10): plt.subplot(2, 5, i + 1) plt.xticks([]) # Remove x-axis ticks plt.yticks([]) # Remove y-axis ticks plt.imshow(x_train[i]) # Display the image plt.xlabel(class_names[y_train[i][0]]) # Display the label plt.show()

Explanation:

1. **Class Names**: A list of class names corresponding to the labels in the CIFAR-10 dataset.
2. **Matplotlib**: Used to create a grid of images.
   a. `plt.subplot(2, 5, i + 1)`: Creates a 2x5 grid for displaying 10 images.
   b. `plt.imshow(x_train[i])`: Displays the image at index `i` from the training dataset.
   c. `plt.xlabel(class_names[y_train[i][0]])`: Adds the corresponding label below the image.
3. **Output**: A grid of 10 sample images with their labels.

Sample Output:
The output will be a 2x5 grid of images with labels like:

- Airplane
- Automobile
- Bird
- Cat
- Deer
- Dog
- Frog
- Horse
- Ship
- Truck

Combined Code for Building the Model and Displaying Sample Images:

import tensorflow as tf from tensorflow import keras import numpy as np import matplotlib.pyplot as plt

Load CIFAR-10 dataset

(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()

Normalize pixel values (0-255 → 0-1)

x_train, x_test = x_train / 255.0, x_test / 255.0

Class names in CIFAR-10

class_names = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']

Display sample images

plt.figure(figsize=(10, 5)) for i in range(10): plt.subplot(2, 5, i + 1) plt.xticks([]) plt.yticks([]) plt.imshow(x_train[i]) plt.xlabel(class_names[y_train[i][0]]) plt.show()

Build CNN Model

model = keras.Sequential([keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)), keras.layers.MaxPooling2D((2, 2)), keras.layers.Conv2D(64, (3, 3), activation='relu'), keras.layers.MaxPooling2D((2, 2)), keras.layers.Conv2D(64, (3, 3), activation='relu'), keras.layers.Flatten(), keras.layers.Dense(64, activation='relu'), keras.layers.Dense(10, activation='softmax')])

Compile the Model

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

Print Model Summary

model.summary()

7. Real-Time Testing (Code):

To test the model on a single image:

```
# Load a sample image
sample_image = x_test[0]
sample_label = y_test[0]

# Predict the class
prediction = model.predict(np.expand_dims(sample_image, axis=0))
predicted_class = np.argmax(prediction)

# Display the result
plt.imshow(sample_image)
plt.title(f"Predicted: {class_names[predicted_class]}, Actual:
{class_names[sample_label[0]]}")
plt.show()
```

8. Key Points:

- The model achieves around 70-80% accuracy on the test set.
- Increasing the number of layers or epochs can improve accuracy.
- Data augmentation and transfer learning can further enhance performance.

9. User Instructions:

- Running Real-Time Detection:

- Ensure all dependencies are installed.
- Run the provided Python script.
- The script will automatically download the dataset, train the model, and evaluate its performance.

- To test the model on a custom image, replace the sample image in the real-time testing code with your image.

10. Model Logical Structure:

1. **Input**: 32x32 RGB image.
2. **Feature Extraction**: Convolutional and pooling layers.
3. **Classification**: Fully connected layers with softmax activation.
4. **Output**: Probabilities for each of the 10 classes.

11. Conclusion:

This project demonstrates the application of CNNs for image classification using the CIFAR-10 dataset. The model achieves decent accuracy and can be further improved using advanced techniques like transfer learning and data augmentation.

12. References:

- CIFAR-10 Dataset: https://www.cs.toronto.edu/~kriz/cifar.html
- TensorFlow Documentation: https://www.tensorflow.org/