IMAGE RECOGNITION USING CIFAR-10 DATA

A NAAN MUDHALVAN PROJECT REPORT

Submitted by

GURU PRASAD R

(711721243305)

in partial fulfilment for the award of the degree

of

BACHELOR Of TECHNOLOGY

IN

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
AND
DATA SCIENCE

KGiSL INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this Naan Mudhalvan Project report on "**IMAGE RECOGNITION USING CIFAR-10 DATA**" is the bonafide work of "**GURU PRASAD R**" who belongs to III Year Artificial Intelligence & Data Science "B" during Sixth Semester of Academic Year 2023-2024.

FACULTY INCHARGE **HEAD OF THE DEPARTMENT**

Certified that the candidates were examined by us for NM1009 Generative AI for Engineering Viva held on_____at KGiSL Institute of Technology, Saravanampatti , Coimbatore 641 035.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Chairman and Managing Director Dr. Ashok Bakthavachalam** for providing us with an environment to complete our Naan Mudhalvan project successfully.

We are grateful to our **CEO of Academic Initiatives Mr. Aravind Kumar Rajendran** and our beloved **Secretary Dr. Rajkumar N.** Our sincere thanks to honourable **Principal Dr. Suresh Kumar S** and **Academic Director Dr. Shankar P** for his support, guidance,and blessings.

We would like to thank **Dr. Kalpana S, Head of the Department**, and **Naan Mudhalvan Coordinator Dr.K.Selva Sheela**, Department of Artificial Intelligence and Data Science for her firm support during the entire course of this Naan Mudhalvan Project and who modeled us both technically and morally for achieving greater success in this project work.

We also thank all the faculty members of our department for their help in making this Naan Mudhalvan project a successful one. Finally, we take this opportunity to extend our deep appreciation to our Family and Friends, for all they meant to us during the crucial times of the completion of our project.

TABLE OF CONTENTS

1. Abstract

2. Introduction

3. Proposed Method

4. System Requirements

- Hardware

- Software

- Optional

5.Implementation

6. Result

7. Output Screenshots

8. Conclusion

ABSTRACT

Image recognition is a pivotal field in computer vision with applications ranging from autonomous systems to medical imaging. This project focuses on implementing image recognition using the CIFAR-10 dataset, a benchmark dataset containing 60,000 32x32 color images across ten classes. Leveraging Convolutional Neural Networks (CNNs), a state-of-the-art approach for image classification, we preprocess the CIFAR-10 dataset, design and train a CNN architecture, and evaluate its performance. Through this project, we aim to demonstrate the efficacy of CNNs in accurately classifying images across diverse categories.

INTRODUCTION

In contemporary society, the ability of machines to perceive and understand images has become increasingly important. Image recognition, a subfield of computer vision, facilitates this capability by enabling machines to identify objects, scenes, and patterns within images. The CIFAR-10 dataset stands as a benchmark for evaluating image recognition algorithms, encompassing a wide array of objects across ten distinct classes. This section provides an overview of the project's objectives, the significance of image recognition in modern technology, and the rationale behind selecting the CIFAR-10 dataset for experimentation.

DOMAIN

In this project, we delve into the domain of Computer Vision Applications, focusing on the implementation of image recognition using the CIFAR-10 dataset. Computer vision plays a pivotal role in numerous sectors, including autonomous systems, medical imaging, surveillance, e-commerce, and environmental monitoring. By leveraging Convolutional Neural Networks (CNNs) and the CIFAR-10 dataset, our aim is to develop a robust image recognition system capable of accurately classifying objects across diverse categories. This project holds significance in advancing the capabilities of computer vision technology, with potential applications ranging from autonomous vehicles and medical diagnostics to security systems and environmental monitoring.

PROPOSED METHOD

The proposed method involves a systematic approach to implementing image recognition using the CIFAR-10 dataset. We commence with preprocessing steps, including data normalization to standardize pixel values and data augmentation to enhance the diversity of the dataset. Subsequently, we design a CNN architecture, a deep learning model well-suited for image classification tasks, and train it on the preprocessed CIFAR-10 dataset. The training process entails optimizing model parameters, such as learning rate and batch size, to achieve optimal performance. Finally, we evaluate the trained model's performance using metrics such as accuracy, precision, recall, and F1-score.

SYSTEM REQUIREMENTS

HARDWARE

The implementation of image recognition systems typically requires computational resources capable of handling intensive computations. This may include a computer with a multi-core CPU and GPU acceleration to expedite training processes.

SOFTWARE

Python serves as the primary programming language for implementing the image recognition system, with deep learning frameworks such as TensorFlow or PyTorch providing essential libraries for model development. Additionally, libraries for data visualization (e.g., Matplotlib) and performance analysis (e.g., scikit-learn) may be utilized.

OPTIONAL

Depending on the complexity of the project, additional software tools for model optimization, such as hyperparameter tuning frameworks (e.g., Hyperopt), and cloud computing platforms for distributed training may be employed.

IMPLEMENTATION

This section details the practical implementation of the image recognition system, encompassing data preprocessing, model architecture design, training procedure, and model evaluation. Data preprocessing involves loading the CIFAR-10 dataset, performing normalization to scale pixel values, and augmenting the dataset to increase its size and diversity. The model architecture design entails selecting appropriate CNN layers, such as convolutional, pooling, and fully connected layers, to construct an effective image classification model. The training procedure involves feeding the preprocessed data into the CNN model, optimizing model parameters using optimization algorithms (e.g., stochastic gradient descent), and iteratively adjusting the model's weights to minimize the loss function. Finally, the trained model's performance is evaluated using a separate test set, and performance metrics are computed to assess its accuracy and generalization capabilities.

PROGRAM

```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D

batch_size = 32
num_classes = 10
epochs = 100
data_augmentation = True


# The data, shuffled and split between train and test sets:
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# Convert class vectors to binary class matrices.
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)


model = Sequential()

model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=x_train.shape[1:]))
model.add(Activation('relu'))

model.add(Conv2D(48, (3, 3), padding='same',
                 input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(80, (3, 3), padding='same',
                 input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding='same',
                 input_shape=x_train.shape[1:]))
model.add(GlobalMaxPooling2D())
model.add(Dropout(0.25))


model.add(Dense(500))
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

```
# initiate RMSprop optimizer
opt = keras.optimizers.Adam(lr=0.0001)

# Let's train the model using RMSprop
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
model.summary()



x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
tbCallBack = keras.callbacks.TensorBoard(log_dir='./Graph2',
histogram_freq=0, write_graph=True, write_images=True)


f not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              validation_data=(x_test, y_test),
              shuffle=True, callbacks=[tbCallBack])
else:
    print('Using real-time data augmentation.')
    # This will do preprocessing and realtime data augmentation:

    datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to
180)
        width_shift_range=0.2,  # randomly shift images horizontally (fraction of
total width)
        height_shift_range=0.2,  # randomly shift images vertically (fraction of
total height)
```

```
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False)  # randomly flip images


# Compute quantities required for feature-wise normalization
# (std, mean, and principal components if ZCA whitening is applied).
datagen.fit(x_train)

# Fit the model on the batches generated by datagen.flow().
model.fit_generator(datagen.flow(x_train, y_train,
                                 batch_size=batch_size),
                    steps_per_epoch=x_train.shape[0] // batch_size,
                    epochs=epochs,
                    validation_data=(x_test, y_test), callbacks=[tbCallBack])



scores=model.evaluate(x_test,y_test,verbose=1)
print("Test Loss",scores[0])
print("Test Accuracy",scores[1])
```

RESULT

The results section presents the outcomes of the image recognition experiment, including quantitative performance metrics and qualitative analysis of the trained model's predictions. Performance metrics such as accuracy, precision, recall, and F1-score provide insights into the model's ability to correctly classify images across different categories. Additionally, visualization techniques, such as confusion matrices and ROC curves, may be employed to further elucidate the model's performance characteristics. The results are discussed in the context of the project objectives, highlighting the strengths and limitations of the implemented image recognition system.

OUTPUT SCREENSHOTS

```
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
activation_1 (Activation)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 32, 32, 48)	13872
activation_2 (Activation)	(None, 32, 32, 48)	0
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 48)	0
dropout_1 (Dropout)	(None, 16, 16, 48)	0
conv2d_3 (Conv2D)	(None, 16, 16, 80)	34640
activation_3 (Activation)	(None, 16, 16, 80)	0
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 80)	0
dropout_2 (Dropout)	(None, 8, 8, 80)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	92288
global_max_pooling2d_1 (Glob	(None, 128)	0
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 500)	64500
activation_4 (Activation)	(None, 500)	0
dropout_4 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 10)	5010
activation_5 (Activation)	(None, 10)	0

```
Total params: 211,206
Trainable params: 211,206
Non-trainable params: 0
```

```
Using real-time data augmentation.
Epoch 1/100
1562/1562 [==============================] - 253s 162ms/step - loss: 2.0428 - acc: 0.2263 - val_loss: 1.7488 - val_acc: 0.3685
Epoch 2/100
1562/1562 [==============================] - 245s 157ms/step - loss: 1.7451 - acc: 0.3457 - val_loss: 1.6306 - val_acc: 0.4018
Epoch 3/100
1562/1562 [==============================] - 238s 152ms/step - loss: 1.6335 - acc: 0.3986 - val_loss: 1.5413 - val_acc: 0.4374
Epoch 4/100
1562/1562 [==============================] - 236s 151ms/step - loss: 1.5561 - acc: 0.4282 - val_loss: 1.4778 - val_acc: 0.4630
Epoch 5/100
1562/1562 [==============================] - 237s 152ms/step - loss: 1.4986 - acc: 0.4535 - val_loss: 1.4335 - val_acc: 0.4782
Epoch 6/100
1562/1562 [==============================] - 235s 151ms/step - loss: 1.4423 - acc: 0.4753 - val_loss: 1.3325 - val_acc: 0.5253
Epoch 7/100
1562/1562 [==============================] - 235s 150ms/step - loss: 1.3961 - acc: 0.4943 - val_loss: 1.2703 - val_acc: 0.5431
Epoch 8/100
1562/1562 [==============================] - 238s 152ms/step - loss: 1.3571 - acc: 0.5111 - val_loss: 1.2088 - val_acc: 0.5718
```

```
Epoch 93/100
1562/1562 [==============================] - 229s 146ms/step - loss: 0.6666 - acc: 0.7668 - val_loss: 0.5986 - val_acc: 0.7943
Epoch 94/100
1562/1562 [==============================] - 230s 147ms/step - loss: 0.6674 - acc: 0.7678 - val_loss: 0.6052 - val_acc: 0.7922
Epoch 95/100
1562/1562 [==============================] - 230s 147ms/step - loss: 0.6656 - acc: 0.7676 - val_loss: 0.6026 - val_acc: 0.7919
Epoch 96/100
1562/1562 [==============================] - 228s 146ms/step - loss: 0.6589 - acc: 0.7682 - val_loss: 0.6131 - val_acc: 0.7891
Epoch 97/100
1562/1562 [==============================] - 228s 146ms/step - loss: 0.6607 - acc: 0.7709 - val_loss: 0.5663 - val_acc: 0.8093
Epoch 98/100
1562/1562 [==============================] - 228s 146ms/step - loss: 0.6569 - acc: 0.7703 - val_loss: 0.5620 - val_acc: 0.8037
Epoch 99/100
1562/1562 [==============================] - 227s 146ms/step - loss: 0.6564 - acc: 0.7704 - val_loss: 0.5940 - val_acc: 0.7962
Epoch 100/100
1562/1562 [==============================] - 228s 146ms/step - loss: 0.6492 - acc: 0.7745 - val_loss: 0.5824 - val_acc: 0.8013
```

```
10000/10000 [==============================] - 14s 1ms/step
Test Loss 0.5823626396656036
Test Accuracy 0.8013
```

CONCLUSION

In conclusion, the project demonstrates the feasibility and efficacy of employing CNNs for image recognition tasks, particularly on the CIFAR-10 dataset. By preprocessing the dataset, designing a suitable CNN architecture, and rigorously training and evaluating the model, we have showcased the model's ability to accurately classify images across diverse categories. The project underscores the significance of CNNs in modern image recognition applications and lays the groundwork for future research and development in this domain.

This comprehensive structure encapsulates the entire lifecycle of the project, from conceptualization to implementation and evaluation, providing a holistic view of the image recognition system's design and performance.