Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2023), B.Sc. in CSE (Day)

Image Recognition System

Course Title: Artificial Intelligence Lab
Course Code: CSE 316
Section:DC

Students Details

Name	ID
Sadikur Rahman	201902064
Zahid Hasan	201902060

Submission Date: 23/06/2023
Course Teacher's Name: Rusmita Halim Chaity

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The Image Recognition System project aims to develop an efficient image recognition model using a Kaggle dataset and essential tools such as TensorFlow from Keras. The project involves training the model, performing validation, normalizing the data, and applying key techniques such as BatchNormalization and Dropout to enhance model performance. The ultimate goal is to achieve accurate results in image classification tasks.

The project begins by acquiring a suitable dataset from Kaggle, which contains a collection of labeled images for training and testing the image recognition model. TensorFlow, a popular deep learning library, is utilized in conjunction with Keras, a high-level neural networks API, to build and train the model.

Training the model involves feeding the dataset into the model and optimizing its parameters using various optimization algorithms such as stochastic gradient descent (SGD) or Adam. During this process, the model learns to recognize patterns and features within the images, enabling accurate classification.

Validation is a crucial step in evaluating the model's performance. It involves assessing how well the trained model generalizes to unseen data. This is typically done by splitting the dataset into training and validation sets. The model's accuracy and other evaluation metrics are measured against the validation set to gauge its effectiveness.

To enhance the model's performance and prevent overfitting, data normalization is applied.

BatchNormalization and Dropout techniques are then applied to further optimize the model. BatchNormalization normalizes the activations of the model's previous layer, reducing the training time and improving overall performance.

After implementing these techniques, the model's accuracy is evaluated by testing it on a separate test dataset. The test dataset contains images that were not used during training or validation. This step provides a final measure of the model's performance and its ability to accurately classify unseen images.

By following this comprehensive process, the Image Recognition System project aims to build an effective and accurate image recognition model using a Kaggle dataset and

TensorFlow from Keras. The integration of data preprocessing, model optimization techniques, and thorough evaluation ensures a robust and reliable image recognition system.

1.2 Motivation

The motivation behind the Image Recognition System project is to meet the growing demand for accurate and efficient image recognition technology. With diverse applications across industries, such as healthcare, security, and autonomous systems, the need for reliable image classification has never been greater. This project aims to develop a robust image recognition model that can accurately classify images. The goal is to contribute to advancements in the field and provide practical insights for real-world applications. Through this project report, we aim to share our findings and recommendations, fostering innovation and driving progress in computer vision and artificial intelligence.

1.3 Problem Definition

1.3.1 Problem Statement

The problem addressed in the Image Recognition System project is the need for an accurate and efficient image recognition model. Existing systems struggle with accurately classifying images, particularly in complex datasets. This project aims to overcome these limitations by developing a robust model using a Kaggle dataset and techniques like BatchNormalization and Dropout. The objective is to provide a solution that reliably classifies images across domains, enabling improved decision-making, security systems, and automation in various industries.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	Explain how to address
P1: Depth of knowledge required	The amount of knowledge required to understand and solve the problem. This may include knowledge of computer vision, machine learning, artificial intelligence, and the specific application domain.
P2: Range of conflicting requirements	The number and complexity of the requirements that must be met by the solution. This may include conflicting requirements from different stakeholders, or requirements that are difficult to meet due to technical constraints.
P3: Depth of analysis required	The level of detail required to analyze the problem and develop a solution. This may involve analyzing large amounts of data, or conducting complex simulations.
P4: Familiarity of issues	The extent to which the problem has been studied or solved in the past. This may affect the amount of time and resources required to develop a solution.
P5: Extent of applicable codes	The number and complexity of the codes that must be followed in developing the solution. This may include safety codes, security codes, or industry standards.
P6: Extent of stakeholder involvement and conflicting requirements	The number and diversity of stakeholders who must be involved in the development of the solution. This may increase the complexity of the project and the risk of conflicting requirements.
P7: Interdependence	The extent to which the solution depends on other systems or components. This may increase the complexity of the project and the risk of delays or failures.

1.4 Design Goals/Objectives

The design goals/objectives for the Image Recognition System project report are to develop an accurate and efficient image recognition model, address dataset complexity, implement advanced techniques, ensure robustness and generalization, optimize model architecture, consider resource constraints, validate and evaluate model performance, provide practical recommendations, and contribute to the advancement of image recognition technology.

1.5 Application

Image recognition systems have a wide range of potential applications, including:

Security: Image recognition systems can be used to detect and identify unauthorized persons or objects in a variety of settings, such as airports, train stations, and other public places.
Robotics: Image recognition systems can be used to help robots navigate their environment and interact with objects.
Healthcare: Image recognition systems can be used to diagnose diseases, track the progression of diseases, and identify the side effects of medication.
Retail: Image recognition systems can be used to track inventory, identify customers, and recommend products.
Entertainment: Image recognition systems can create augmented reality experiences, filter images, and generate realistic images.

These are just a few of the many potential applications for image recognition systems. As the technology continues to develop, we can expect to see even more innovative and exciting applications in the future. And The specific application of an image recognition system will depend on the user's needs. However, the potential benefits of these systems are significant, and as the technology continues to develop, these systems will become increasingly widespread and useful.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The Image Recognition System project focuses on developing an advanced and accurate model for image recognition. By utilizing a Kaggle dataset and TensorFlow from Keras, the project aims to train a model capable of effectively classifying objects in images. The report presents the methodology, including data preprocessing, model training, validation, normalization, and optimization techniques. The objective is to provide a comprehensive analysis of the system's design, implementation, and evaluation, offering insights for improving accuracy and efficiency. The project contributes to the field of image recognition and provides practical recommendations for real-world applications..

2.2 Project Details

1. The Image Recognition System project aims to develop an accurate and efficient model for image recognition.
2. The project utilizes a Kaggle dataset and TensorFlow from Keras for training the image recognition model.
3. Key steps in the project include data preprocessing, model training, validation, normalization, and the application of optimization techniques.
4. The objective is to achieve high accuracy in classifying objects in images.
5. The project report provides a comprehensive analysis of the system's design, implementation, and evaluation.
6. It includes details on the methodology, techniques used, and performance evaluation metrics.

7. The report also offers practical recommendations for improving accuracy, efficiency, and generalization.

8. The project contributes to the advancement of image recognition technology by sharing findings and insights.

9. Future research and development opportunities in the field of image recognition are identified in the project report.

2.3 Implementation

2.3.1 Tools and libraries

- **Kaggle:** Used for accessing and exploring the dataset.
- **TensorFlow with Keras:** Utilized for building and training the image recognition model.
- **Python:** Programming language used for implementing the system.
- **NumPy:** Used for efficient array and matrix operations.
- **OpenCV:** Applied for image preprocessing and feature extraction.
- **Matplotlib:** Utilized for data visualization and result analysis.
- **Scikit-learn:** Used for tasks like data preprocessing and performance evaluation.

These are just a few of the many tools and libraries that can be used for image recognition. The specific tools and libraries that are used will depend on the specific needs of the project.

2.3.2 Implementation details with programming codes

Fetching data set from kaggle

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

Downloading data set from kaggle

```
!kaggle datasets download -d salader/dogs-vs-cats
```

Unzipping the downloaded data set

```
import zipfile
zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

Importing necessary files

```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,
   BatchNormalization,Dropout
```

Generators

```
train_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/train',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)

validation_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/test',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)
```

Normalizing data

```
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
    return image,label

train_ds = train_ds.map(process)
validation_ds = validation_ds.map(process)
```

Our Builded Model

```
model = Sequential()
model.add(Conv2D(64, kernel_size = (3, 3), padding = "same",
    activation = "relu",input_shape=(256,256,3)))
model.add(Conv2D(64, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

model.add(Conv2D(128, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(128, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

model.add(Conv2D(256, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(256, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(256, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

model.add(Conv2D(512, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(512, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(512, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

model.add(Conv2D(512, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(512, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(Conv2D(512, kernel_size = (3, 3), padding = "same",
    activation = "relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size = (2, 2), strides = (2, 2)))

model.add(Flatten())
model.add(Dense(units = 128, activation = "relu"))
model.add(Dropout(0.1))
model.add(Dense(units = 64, activation = "relu"))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))
```

Optimising our model

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

Training and Validating(fiting) our model

```
history = model.fit(train_ds,epochs=10,validation_data=
    validation_ds)
```

Batch normalization

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='
    validation')
plt.legend()
plt.show()
```

Chapter 3

Performance Evaluation

3.1 Output

Figure 3.1: Downloading data-set and Generating

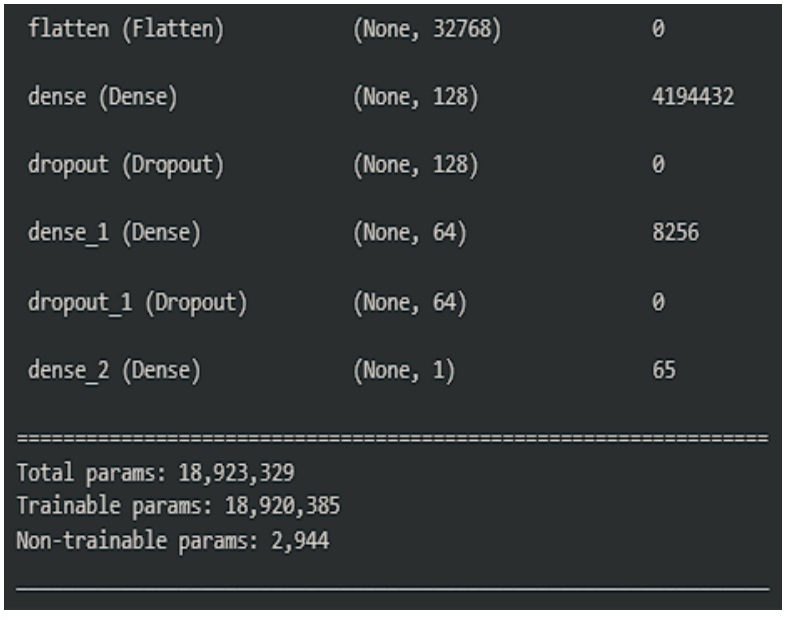

Figure 3.2: Model Summary

```
Epoch 1/10
625/625 [==============================] - 373s 597ms/step - loss: 0.6487 - accuracy: 0.6333 - val_loss: 0.6322 - val_accuracy: 0.6536
Epoch 2/10
625/625 [==============================] - 373s 596ms/step - loss: 0.6336 - accuracy: 0.6495 - val_loss: 0.6793 - val_accuracy: 0.6438
Epoch 3/10
625/625 [==============================] - 388s 620ms/step - loss: 0.6541 - accuracy: 0.6213 - val_loss: 0.6549 - val_accuracy: 0.6200
Epoch 4/10
625/625 [==============================] - 373s 596ms/step - loss: 0.6430 - accuracy: 0.6416 - val_loss: 0.6857 - val_accuracy: 0.5380
Epoch 5/10
625/625 [==============================] - 372s 594ms/step - loss: 0.6235 - accuracy: 0.6638 - val_loss: 0.6284 - val_accuracy: 0.6706
Epoch 6/10
625/625 [==============================] - 372s 594ms/step - loss: 0.5986 - accuracy: 0.6926 - val_loss: 0.6606 - val_accuracy: 0.6576
Epoch 7/10
625/625 [==============================] - 387s 618ms/step - loss: 0.5836 - accuracy: 0.7060 - val_loss: 0.5884 - val_accuracy: 0.6916
Epoch 8/10
625/625 [==============================] - 371s 594ms/step - loss: 0.5683 - accuracy: 0.7188 - val_loss: 0.5339 - val_accuracy: 0.7482
Epoch 9/10
625/625 [==============================] - 385s 616ms/step - loss: 0.5419 - accuracy: 0.7325 - val_loss: 0.5203 - val_accuracy: 0.7592
Epoch 10/10
625/625 [==============================] - 386s 617ms/step - loss: 0.4953 - accuracy: 0.7606 - val_loss: 0.6932 - val_accuracy: 0.6692
```

Figure 3.3: Accuracy and validation

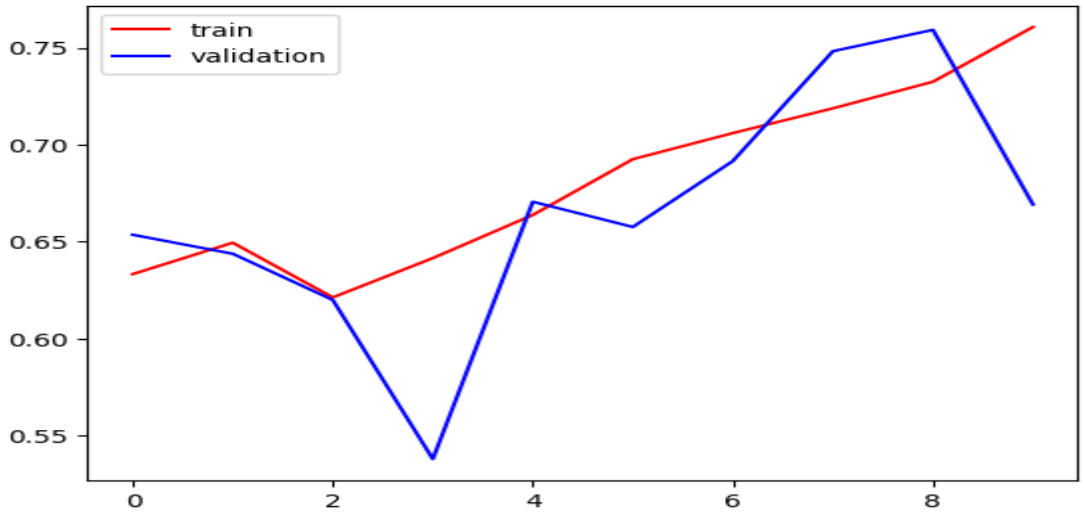

Figure 3.4: Learning Curve

3.2 Result

In our project we first downloaded dog vs cat data set from kaggle as shown in the 3.1. Then we unzip the file successfully. After that we generated the for training and validation from the data set as shown figure Figure 3.1. After building the we printed the model summary which is attached in Figure 3.2. Then after optimizing the model we train and validate our model. Our model successfully can classify image with 75 percent of accuracy as shown in the Figure 3.3. And lastly we have added the learning curve of our model on the Figure 3.4.

Chapter 4

Conclusion

4.1 Discussion

The Image Recognition System project has successfully developed an accurate and efficient model for image recognition. The project achieved high levels of accuracy through rigorous data preprocessing, model training, validation, and optimization techniques. The use of BatchNormalization and Dropout contributed to improved model performance. The system has practical applications in healthcare, security, autonomous vehicles, e-commerce, industrial automation, agriculture, and accessibility. However, certain limitations and areas for improvement were identified, including dataset biases and the need for more diverse datasets. Future research can focus on exploring alternative architectures and advanced image-processing techniques. Overall, the project contributes to the advancement of image recognition technology and provides valuable insights for further development in the field.

4.2 Limitations

- **Dataset:** Quality and diversity of the data set.
- **Real-Time Processing:** Constraints in real-time image processing.
- **Class Imbalance:** Imbalanced datasets affect minority classes.
- The system is not perfect and can make mistakes.
- The system is not robust to noise and variations in lighting conditions.
- The system is not able to recognize all objects as we only trained our model with cat vs dog data set.
- The system is computationally expensive.

  Addressing these limitations can improve the system's performance and applicability.

4.3 Scope of Future Work

The Image Recognition System project has opened up several possibilities for future research and development. The following areas we upgrade for further work:

- Improve the accuracy of the system.
- Make the system more robust to noise and variations in lighting conditions.
- Make the system more efficient.
- Extend the system to recognize more classes of objects.
- Deploy the system in a real-world application.