Image Classification with Deep Learning

Selvaraj.X(2577127)

13-NOV-2023

Introduction

- Image classification is a fundamental task in computer vision, which involves assigning predefined labels to images. Deep learning has revolutionized image classification, achieving state-of-the-art results on various benchmarks.
- This presentation will discuss how to develop an image classification system using deep learning with TensorFlow and Keras.

Problem Statement

- Develop an image classification system for a new application using TensorFlow with the Keras API
- Classify images into predefined categories
- Dataset: CIFAR10

Neural Network Architecture

- Design an ANN with convolutional, pooling, and dense layers
- Justify architecture choices based on the problem

```
# Define the Neural Network architecture
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(10, activation='softmax'))
```

Model Training

- Compile the model with an optimizer, loss function, and evaluation metric
- Train the model on the training set for a specified number of epochs
- Monitor training and use validation data to prevent overfitting

```
# Train the model
history = model.fit(train_images, train_labels, epochs=10, validation_data=(val_

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

Model Evaluation

- Evaluate the trained model on the test set
- Report key performance metrics (accuracy, precision, recall)
- Use visualizations (confusion matrices, ROC curves) to analyze performance.

```
# Model evaluation
print(f'Test accuracy: {test_acc}')
predictions = model.predict(test_images)
predicted_labels = tf.argmax(predictions, axis=1)
```

```
Test accuracy: 0.7088000178337097
313/313 [==============================] - 3s 11ms/step
```

Fine-Tuning and Optimization

- Experiment with hyperparameter tuning or model modifications to improve performance
- Discuss any challenges encountered during training and how they were addressed.

```
# Hypertuning model

model.add(layers.Dropout(0.25))  # Adding dropout to reduce overfitting
```

```
# Train the model
history = model.fit(train_images, train_labels, epochs=10, validation_data=(val_

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

Model Comparison

Before Tuning……..

```
# Model evaluation
print(f'Test accuracy: {test_acc}')
predictions = model.predict(test_images)
predicted_labels = tf.argmax(predictions, axis=1)
```

```
Test accuracy: 0.7088000178337097
313/313 [==============================] - 3s 11ms/step
```

After Tuning…….

```
# Model evaluation
print(f'Test accuracy: {test_acc}')
predictions = model.predict(test_images)
predicted_labels = tf.argmax(predictions, axis=1)
```

```
Test accuracy: 0.6866999864578247
313/313 [==============================] - 3s 11ms/step
```

Conclusion

- In this presentation, we have discussed the key steps involved in developing an image classification system using deep learning with TensorFlow and Keras. We have also highlighted some important considerations for deployment and documentation.
- Deep learning is a powerful tool for image classification, but it is important to choose the right architecture and hyperparameters for the specific problem. Additionally, it is essential to monitor the training process and use validation data to prevent overfitting.