Deep-Learning

Practical 4

Aim : Basics of Tensorflow for Nueral network

Code:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models

# Step 1: Create a small dataset
x_train = np.array([[0.1, 0.2], [0.4, 0.3], [0.5, 0.6], [0.9,
0.8], [0.7, 0.3],
                    [0.2, 0.1], [0.8, 0.5], [0.4, 0.6], [0.3,
0.7], [0.6, 0.9]])

y_train = (x_train[:, 1] > x_train[:, 0]).astype(int)

# Step 2: Define the Model
model = models.Sequential([
    layers.Dense(8, activation='relu', input_shape=(2,)),
    layers.Dense(1, activation='sigmoid')
])

# Step 3: Compile the Model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Step 4: Train the Model
model.fit(x_train, y_train, epochs=5, batch_size=2)

# Step 5: Make Predictions
predictions = model.predict(x_train)
print("Predictions (rounded):",
np.round(predictions).astype(int))
print("Actual Labels:", y_train)

# Step 6: Evaluate the Model
test_loss, test_acc = model.evaluate(x_train, y_train)
```

```
print(f'Test Accuracy: {test_acc}')

# To visualize the model structure
model.summary()
```

Output

```
jayraj@jayrajs-MacBook-Air deep-learning % python3 pr-4.py
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
t to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 805us/step - accuracy: 0.5000 - loss: 0.7321
Epoch 2/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 517us/step - accuracy: 0.4722 - loss: 0.7697
Epoch 3/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 479us/step - accuracy: 0.5417 - loss: 0.7265
Epoch 4/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 540us/step - accuracy: 0.6458 - loss: 0.6832
Epoch 5/5
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 527us/step - accuracy: 0.5833 - loss: 0.6731
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Predictions (rounded): [[0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]]
Actual Labels: [1 0 1 0 0 0 0 1 1 1]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.5000 - loss: 0.7234
Test Accuracy: 0.5
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 8)	24
dense_1 (Dense)	(None, 1)	9

```
Total params: 101 (408.00 B)
Trainable params: 33 (132.00 B)
Non-trainable params: 0 (0.00 B)
Optimizer params: 68 (276.00 B)
jayraj@jayrajs-MacBook-Air deep-learning %
```

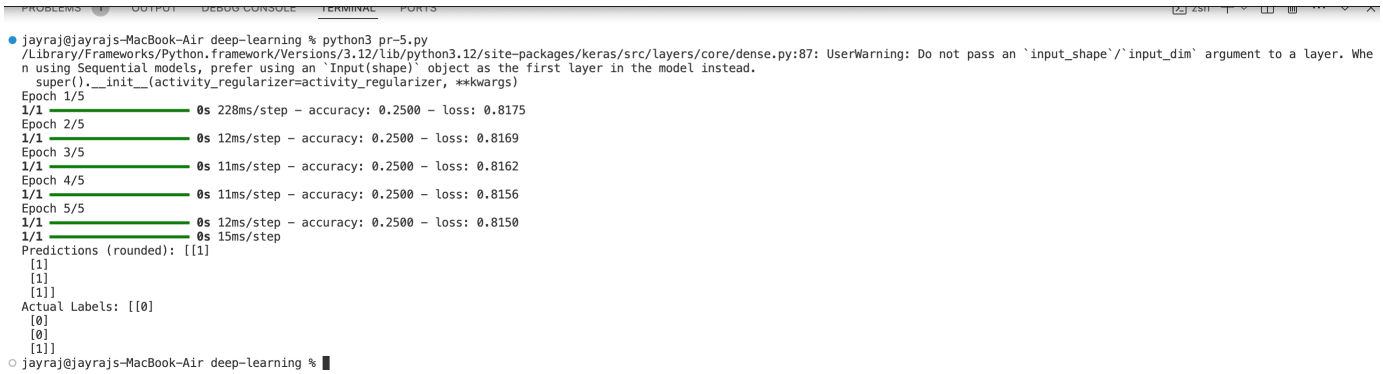
Practical 5

Aim : Write a python program to implement perceptron using tensorflow

Code:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models


x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [0], [0], [1]])


model = models.Sequential([
    layers.Dense(1, activation='sigmoid', input_shape=(2,))
])
weights = np.array([[0.4], [0.6]])
bias = np.array([0.2])

# Set weights and bias for the Dense layer
model.layers[0].set_weights([weights, bias])

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

# Train the model
model.fit(x_train, y_train, epochs=5)

# Make predictions
predictions = model.predict(x_train)
print("Predictions (rounded):",
np.round(predictions).astype(int))
print("Actual Labels:", y_train)
```

Output:



Practical 6

Aim : Write a program to implement an autoencoder for image reconstruction

Code:-

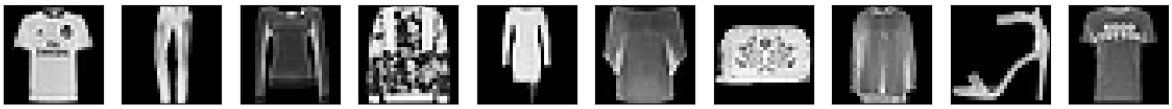
```
import numpy as np
import os
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
import pandas as pd

def load_fashion_mnist_data():
    base_path = './fashion-mnist/'
    # Load data directly if CSV format is provided
    x_train = pd.read_csv(os.path.join(base_path, 'fashion-
mnist_train.csv')).values[:, 1:] / 255.0
    x_test = pd.read_csv(os.path.join(base_path, 'fashion-
mnist_test.csv')).values[:, 1:] / 255.0
    return x_train, x_test


x_train, x_test = load_fashion_mnist_data()

encoding_dim = 64

input_img = layers.Input(shape=(784,))
encoded = layers.Dense(encoding_dim, activation='relu')
(input_img)

decoded = layers.Dense(784, activation='sigmoid')(encoded)

# Combine encoder and decoder into the autoencoder model
autoencoder = models.Model(input_img, decoded)


autoencoder.compile(optimizer='adam',
loss='binary_crossentropy')



autoencoder.fit(x_train, x_train,
                epochs=5,
                batch_size=256,
```

```
                shuffle=True,
                validation_data=(x_test, x_test))

# Encode and decode some images (reconstruction)
decoded_imgs = autoencoder.predict(x_test)

# Reshape images back to 28x28 for visualization
decoded_imgs = decoded_imgs.reshape((x_test.shape[0], 28, 28))


n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Display original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Display reconstructed images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i], cmap='gray')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

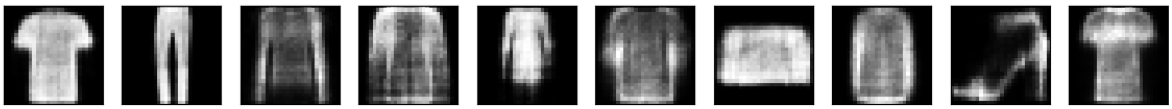
Output:-

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

jayraj@jayrajs-MacBook-Air deep-learning % python3 pr-6.py
Epoch 1/5
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.4743 - val_loss: 0.3256
Epoch 2/5
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.3202 - val_loss: 0.3044
Epoch 3/5
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.3018 - val_loss: 0.2923
Epoch 4/5
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.2909 - val_loss: 0.2855
Epoch 5/5
235/235 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.2866 - val_loss: 0.2815
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 468us/step
```

original images



reconstructed images



Practical 7

Aim :Write a program in python for image classification using CNN (using tensorflow.

Code

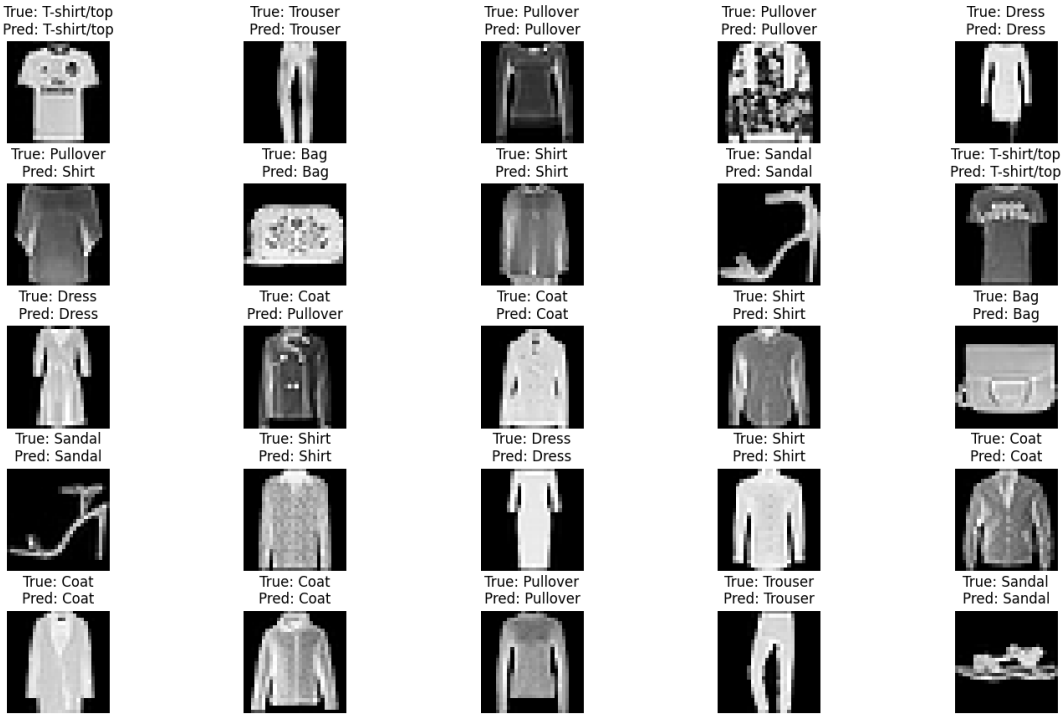
```
import numpy as np
import os
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models

# Load the Fashion MNIST dataset
def load_fashion_mnist_data():
    base_path = './fashion-mnist/'
    x_train = pd.read_csv(os.path.join(base_path, 'fashion-
mnist_train.csv')).values[:, 1:]
    x_test = pd.read_csv(os.path.join(base_path, 'fashion-
mnist_test.csv')).values[:, 1:]

    # Reshape data to 28x28 and scale to [0, 1]
    x_train = x_train.reshape(-1, 28, 28, 1) / 255.0
    x_test = x_test.reshape(-1, 28, 28, 1) / 255.0

    # Extract labels
    y_train = pd.read_csv(os.path.join(base_path, 'fashion-
mnist_train.csv')).values[:, 0]
    y_test = pd.read_csv(os.path.join(base_path, 'fashion-
mnist_test.csv')).values[:, 0]

    return x_train, y_train, x_test, y_test


x_train, y_train, x_test, y_test = load_fashion_mnist_data()

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
```

```
 layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')  # 10 classes for
Fashion MNIST
])

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


model.fit(x_train, y_train, epochs=5, batch_size=300,
validation_split=0.1)


test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)


predictions = model.predict(x_test)


def plot_predictions(x, y_true, y_pred, class_names):
    plt.figure(figsize=(10, 10))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.imshow(x[i].reshape(28, 28), cmap='gray')
        plt.title(f"True: {class_names[y_true[i]]}\nPred:
{class_names[np.argmax(y_pred[i])]}")
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Define class names for Fashion MNIST
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress',
'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle
boot']


# Plot the predictions
```

```
plot_predictions(x_test, y_test, predictions, class_names)
```

Output

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

jayraj@jayrajs-MacBook-Air deep-learning % python3 pr-7.py
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument t
o a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
180/180 ━━━━━━━━━━━━━━━━━━━━ 8s 42ms/step - accuracy: 0.6039 - loss: 1.1729 - val_accuracy: 0.8077 - val_loss: 0.5283
Epoch 2/5
180/180 ━━━━━━━━━━━━━━━━━━━━ 8s 43ms/step - accuracy: 0.8223 - loss: 0.4835 - val_accuracy: 0.8472 - val_loss: 0.4302
Epoch 3/5
180/180 ━━━━━━━━━━━━━━━━━━━━ 8s 43ms/step - accuracy: 0.8564 - loss: 0.4033 - val_accuracy: 0.8672 - val_loss: 0.3845
Epoch 4/5
180/180 ━━━━━━━━━━━━━━━━━━━━ 8s 44ms/step - accuracy: 0.8719 - loss: 0.3604 - val_accuracy: 0.8782 - val_loss: 0.3540
Epoch 5/5
180/180 ━━━━━━━━━━━━━━━━━━━━ 8s 44ms/step - accuracy: 0.8837 - loss: 0.3250 - val_accuracy: 0.8723 - val_loss: 0.3500
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8736 - loss: 0.3390
Test accuracy: 0.878000020980835
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
jayraj@jayrajs-MacBook-Air deep-learning %

Low Battery
Your Mac will sleep soon unless plugged into a power outlet.
```



Practical 8

Aim : Write a program to use a pre-trained model (e.g., VGG16, Reset) for a custom image classification task.

Code:-

```
from google.colab import files
uploaded = files.upload()
import keras
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input,
decode_predictions
import numpy as np
from keras.applications.resnet50 import ResNet50
# Load the pre-trained ResNet50 model
model = ResNet50(weights='imagenet')
img_path= 'Camera.jpg'
img = keras.utils.load_img(img_path, target_size=(224, 224))
x= keras.utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
preds = model.predict(x)
print('Predicted:', decode_predictions (preds, top=3) [0])
```

Output:-

```
Choose Files  Camera.jpg
• Camera.jpg(image/jpeg) - 271358 bytes, last modified: 11/20/2021 - 100% done
Saving Camera.jpg to Camera (2).jpg
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102967424/102967424 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Predicted: [('n04069434', 'reflex_camera', 0.94990647), ('n03976467', 'Polaroid_camera', 0.016023183), ('n03657121', 'lens_cap', 0.014319241)]
```

Practical 9

Aim : Write a program to fine-tune the pre-trained model on a new dataset and Compare the performance of the fine-tuned model with a model trained from scratch.

Code:-

```
import os
import zipfile
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

os.makedirs(os.path.expanduser("~/.kaggle"), exist_ok=True)
kaggle_json_path = os.path.expanduser("~/.kaggle/kaggle.json")
if not os.path.exists(kaggle_json_path):
    raise FileNotFoundError("kaggle.json file not found.
Please set up Kaggle API key as described.")

# Download Fashion MNIST dataset from Kaggle
os.system('kaggle datasets download -d zalando-research/
fashionmnist')

# Unzip the dataset
with zipfile.ZipFile('fashionmnist.zip', 'r') as zip_ref:
    zip_ref.extractall('fashionmnist')

train_data = pd.read_csv('fashionmnist/fashion-
mnist_train.csv')
test_data = pd.read_csv('fashionmnist/fashion-mnist_test.csv')

x_train = train_data.iloc[:, 1:].values
y_train = train_data.iloc[:, 0].values
x_test = test_data.iloc[:, 1:].values
y_test = test_data.iloc[:, 0].values
x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
# Convert labels to categorical (one-hot encoding)
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Split train set into train and validation set
x_train, x_val, y_train, y_val = train_test_split(x_train,
y_train, test_size=0.2, random_state=42)

# Build the model
model = Sequential([
    Dense(128, activation='relu',
input_shape=(x_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')  # 10 classes for Fashion
MNIST
])

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

# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=32,
validation_data=(x_val, y_val))

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')

# Save the model
model.save('fashion_mnist_model.h5')
```

Output:-

```
jayraj@jayrajs-MacBook-Air deep-learning % /usr/local/bin/python3 /Users/jayraj/Desktop/study/python/deep-learning/pr-9.py
Dataset URL: https://www.kaggle.com/datasets/zalando-research/fashionmnist
License(s): other
fashionmnist.zip: Skipping, found more recently modified local copy (use --force to force download)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7738 - loss: 0.6466 - val_accuracy: 0.8488 - val_loss: 0.4101
Epoch 2/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 1s 985us/step - accuracy: 0.8571 - loss: 0.3937 - val_accuracy: 0.8403 - val_loss: 0.3965
Epoch 3/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 1s 981us/step - accuracy: 0.8718 - loss: 0.3528 - val_accuracy: 0.8757 - val_loss: 0.3359
Epoch 4/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8821 - loss: 0.3179 - val_accuracy: 0.8727 - val_loss: 0.3487
Epoch 5/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 1s 975us/step - accuracy: 0.8892 - loss: 0.3058 - val_accuracy: 0.8789 - val_loss: 0.3429
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 345us/step - accuracy: 0.8748 - loss: 0.3317
Test accuracy: 0.878000020980835
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
jayraj@jayrajs-MacBook-Air deep-learning %
```

Practical 10

Aim : Write a program to implement an RNN/LSTM for text generation.

Code:-

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding,
Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequences


text = """The quick brown fox jumps over the lazy dog. The
quick brown fox jumps over the lazy dog."""


tokenizer = Tokenizer(char_level=True)
tokenizer.fit_on_texts([text])
total_chars = len(tokenizer.word_index) + 1

input_sequences = []
for i in range(1, len(text)):
    seq = text[:i+1]
    input_sequences.append(tokenizer.texts_to_sequences([seq])
[0])

max_sequence_len = max([len(seq) for seq in input_sequences])
input_sequences = pad_sequences(input_sequences,
maxlen=max_sequence_len, padding='pre')


input_sequences = np.array(input_sequences)
X = input_sequences[:, :-1]
y = input_sequences[:, -1]
y = tf.keras.utils.to_categorical(y, num_classes=total_chars)

# Build the LSTM model
model = Sequential()
```

```
model.add(Embedding(total_chars, 50,
input_length=max_sequence_len - 1))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dense(total_chars, activation='softmax'))

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

# Train the model
history = model.fit(X, y, epochs=5)

# Function to generate text
def generate_text(seed_text, next_chars=10):
    for _ in range(next_chars):
        tokenized_seq =
tokenizer.texts_to_sequences([seed_text])[0]
        tokenized_seq = pad_sequences([tokenized_seq],
maxlen=max_sequence_len, padding='pre')
        predicted_char_index =
np.argmax(model.predict(tokenized_seq), axis=-1)
        next_char =
tokenizer.index_word[predicted_char_index[0]]
        seed_text += next_char
    return seed_text


seed_text = "The quick"
generated_text = generate_text(seed_text, next_chars=20)
print("Generated Text: \n", generated_text)
```

Output:-

ITM(SLS) BARODA UNIVERSITY VADODARA
SCHOOL OF COMPUTER SCIENCE ENGINEERING AND TECHNOLOGY

```
jayraj@jayrajs-MacBook-Air deep-learning % python3 pr-10.py
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Epoch 1/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 87ms/step - accuracy: 0.1520 - loss: 3.3646
Epoch 2/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - accuracy: 0.1982 - loss: 3.3406
Epoch 3/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 0.1825 - loss: 3.2812
Epoch 4/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 87ms/step - accuracy: 0.2177 - loss: 3.1429
Epoch 5/5
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - accuracy: 0.1786 - loss: 3.1313
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 181ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Generated Text:
 The quick
jayraj@jayrajs-MacBook-Air deep-learning %
```

Practical 11

Aim : Write a program to train the model on a text corpus (e.g., Shakespeare's works).

Code:-

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding,
Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequences

text = """Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date:
Sometime too hot the eye of heaven shines,
And often is his gold complexion dimm'd;
And every fair from fair sometime declines,
By chance or nature's changing course untrimm'd;
But thy eternal summer shall not fade
Nor lose possession of that fair thou owest;
Nor shall Death brag thou wanderest in his shade,
When in eternal lines to time thou growest:
So long as men can breathe or eyes can see,
So long lives this, and this gives life to thee."""


tokenizer = Tokenizer(char_level=True)
tokenizer.fit_on_texts([text])
total_chars = len(tokenizer.word_index) + 1  # Total unique
characters


input_sequences = []
for i in range(1, len(text)):
    seq = text[:i+1]
    input_sequences.append(tokenizer.texts_to_sequences([seq])
[0])
```

```
max_sequence_len = max([len(seq) for seq in input_sequences])
input_sequences = pad_sequences(input_sequences,
maxlen=max_sequence_len, padding='pre')

input_sequences = np.array(input_sequences)
X = input_sequences[:, :-1]
y = input_sequences[:, -1]
y = tf.keras.utils.to_categorical(y, num_classes=total_chars)
# Build the LSTM model
model = Sequential()
model.add(Embedding(total_chars, 50,
input_length=max_sequence_len - 1))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dense(total_chars, activation='softmax'))

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

# Train the model
history = model.fit(X, y, epochs=10, verbose=1)

# Function to generate text
def generate_text(seed_text, next_chars=100):
    for _ in range(next_chars):
        tokenized_seq =
tokenizer.texts_to_sequences([seed_text])[0]
        tokenized_seq = pad_sequences([tokenized_seq],
maxlen=max_sequence_len, padding='pre')
        predicted_char_index =
np.argmax(model.predict(tokenized_seq), axis=-1)
        next_char =
tokenizer.index_word[predicted_char_index[0]]
        seed_text += next_char
    return seed_text

seed_text = "Shall I compare"
generated_text = generate_text(seed_text, next_chars=100)
print("Generated Text: \n", generated_text)
```

ITM(SLS) BARODA UNIVERSITY VADODARA
SCHOOL OF COMPUTER SCIENCE ENGINEERING AND TECHNOLOGY

Output:-

```
jayraj@jayrajs-MacBook-Air deep-learning % python3 pr-11.py
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Epoch 1/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 16s 701ms/step - accuracy: 0.1416 - loss: 3.3449
Epoch 2/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 722ms/step - accuracy: 0.0958 - loss: 2.9883
Epoch 3/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 716ms/step - accuracy: 0.1537 - loss: 2.9433
Epoch 4/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 707ms/step - accuracy: 0.1846 - loss: 2.9065
Epoch 5/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 709ms/step - accuracy: 0.1629 - loss: 2.8947
Epoch 6/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 712ms/step - accuracy: 0.1758 - loss: 2.9242
Epoch 7/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 709ms/step - accuracy: 0.1801 - loss: 2.8854
Epoch 8/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 714ms/step - accuracy: 0.1575 - loss: 2.9633
Epoch 9/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 682ms/step - accuracy: 0.1537 - loss: 2.9552
Epoch 10/10
20/20 ━━━━━━━━━━━━━━━━━━━━ 14s 679ms/step - accuracy: 0.1850 - loss: 2.9111
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 239ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step
Generated Text:
 Shall I compare      e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e
jayraj@jayrajs-MacBook-Air deep-learning %
```

Practical 12

Aim : Write a program to implement an RNN/LSTM for sentiment analysis for any text data such as tweets, instagram comment etc.

Code:-

```
import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense,
Dropout
from tensorflow.keras.preprocessing.sequence import
pad_sequences

# Load the dataset
dataset, info = tfds.load("imdb_reviews", with_info=True,
as_supervised=True)
train_data, test_data = dataset['train'], dataset['test']

vocab_size = 10000
max_length = 200
embedding_dim = 64
batch_size = 300


train_texts = []
for text, label in train_data:
    train_texts.append(text.numpy().decode('utf-8'))  # Decode
from bytes to string


tokenizer =
tf.keras.preprocessing.text.Tokenizer(num_words=vocab_size,
oov_token="<OOV>")
tokenizer.fit_on_texts(train_texts)

def encode_and_pad(text, label):
    # Tokenize and pad the text
    text =
tokenizer.texts_to_sequences([text.numpy().decode('utf-8')])
    text = pad_sequences(text, maxlen=max_length,
padding='post', truncating='post')
```

```
    return tf.convert_to_tensor(text[0], dtype=tf.int32),
tf.convert_to_tensor(label, dtype=tf.int64)

def encode_and_pad_tf(text, label):
    text, label = tf.py_function(func=encode_and_pad,
inp=[text, label], Tout=(tf.int32, tf.int64))
    text.set_shape([max_length])
    label.set_shape([])
    return text, label

# Apply the transformation
train_data = train_data.map(encode_and_pad_tf)
test_data = test_data.map(encode_and_pad_tf)

# Shuffle, batch, and prefetch the datasets
train_data =
train_data.shuffle(10000).batch(batch_size).prefetch(tf.data.e
xperimental.AUTOTUNE)
test_data =
test_data.batch(batch_size).prefetch(tf.data.experimental.AUTO
TUNE)

# LSTM model
model = Sequential([
    Embedding(vocab_size, embedding_dim),  # Removed
input_length
    LSTM(64, return_sequences=True),
    Dropout(0.5),
    LSTM(32),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

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

steps_per_epoch = len(train_data) // batch_size
model.fit(train_data, epochs=5, validation_data=test_data,
steps_per_epoch=steps_per_epoch)

loss, accuracy = model.evaluate(test_data)
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

```
new_texts = ["This movie was fantastic!", "I did not like the
film at all."]
new_sequences = tokenizer.texts_to_sequences(new_texts)
new_padded_sequences = pad_sequences(new_sequences,
maxlen=max_length, padding='post')

predictions = model.predict(new_padded_sequences)
for i, text in enumerate(new_texts):
    sentiment = "positive" if predictions[i] > 0.5 else
"negative"
    print(f"Text: {text} | Predicted Sentiment: {sentiment}")
```

Output:-

```
PROBLEMS 9   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

/usr/local/bin/python3 /Users/jayraj/Desktop/study/python/deep-learning/pr-12.py
jayraj@jayrajs-MacBook-Air deep-learning % /usr/local/bin/python3 /Users/jayraj/Desktop/study/python/deep-learning/pr-12.py
2024-10-19 09:14:38.120098: I tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: E
d of sequence
Epoch 1/5
84/84 ━━━━━━━━━━━━━━━━━━━━ 48s 540ms/step - accuracy: 0.5153 - loss: 0.6923 - val_accuracy: 0.5285 - val_loss: 0.6867
Epoch 2/5
84/84 ━━━━━━━━━━━━━━━━━━━━ 46s 525ms/step - accuracy: 0.6005 - loss: 0.6552 - val_accuracy: 0.5082 - val_loss: 0.6948
Epoch 3/5
84/84 ━━━━━━━━━━━━━━━━━━━━ 47s 538ms/step - accuracy: 0.5149 - loss: 0.6965 - val_accuracy: 0.5506 - val_loss: 0.6873
Epoch 4/5
84/84 ━━━━━━━━━━━━━━━━━━━━ 46s 536ms/step - accuracy: 0.5781 - loss: 0.6781 - val_accuracy: 0.7372 - val_loss: 0.5674
Epoch 5/5
84/84 ━━━━━━━━━━━━━━━━━━━━ 61s 706ms/step - accuracy: 0.7330 - loss: 0.5690 - val_accuracy: 0.5000 - val_loss: 0.6936
84/84 ━━━━━━━━━━━━━━━━━━━━ 12s 145ms/step - accuracy: 0.4980 - loss: 0.6937
Test Accuracy: 50.00%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
Text: This movie was fantastic! | Predicted Sentiment: positive
Text: I did not like the film at all. | Predicted Sentiment: positive
jayraj@jayrajs-MacBook-Air deep-learning %
```