Ex No: 6

BUILD A RECURRENT NEURAL NETWORK

Aim:

To build a recurrent neural network with Keras/TensorFlow.

Procedure:

1. Import required libraries for numerical computation and deep learning, such as NumPy and TensorFlow.
2. Define a function to generate dummy sequential data with random features and one-hot encoded labels.
3. Generate training data with a specified number of samples, sequence length, and output classes using the data generation function.
4. Build a sequential model using Keras with a SimpleRNN layer for processing sequential data and a Dense output layer for classification.
5. Compile the model with the Adam optimizer, categorical cross-entropy loss, and accuracy as a performance metric.
6. Train the model on the generated training data for a specified number of epochs and batch size.
7. Generate a separate set of sequential test data to evaluate the trained model's performance.
8. Evaluate the model using the test dataset and print the loss and accuracy values.
9. Generate new random sequential data for prediction.
10. Use the trained model to predict the class probabilities for the new data samples and print the predictions.

Code:

```
import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

# Generate dummy sequential data

def generate_data(num_samples=1000, sequence_length=10, num_classes=2):

    X = np.random.rand(num_samples, sequence_length, 1)  # 3D input for RNN
```

```
    y = np.random.randint(num_classes, size=(num_samples, num_classes))  # One-hot encoded labels
    return X, y
X, y = generate_data()
model = keras.Sequential([
    layers.SimpleRNN(50, activation='relu', input_shape=(X.shape[1], X.shape[2])),
    layers.Dense(2, activation='softmax')  # Adjust the number of units for your classes
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
model.fit(X, y, epochs=10, batch_size=32)
# Generate test data
X_test, y_test = generate_data(num_samples=200)
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {accuracy:.4f}')
# Generate some new data for prediction
X_new = np.random.rand(5, 10, 1)  # 5 new samples
predictions = model.predict(X_new)
print(predictions)
```

Output:

```
Epoch 1/10
C:\Users\mannu\anaconda3\envs\dlc\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do
layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the
  super().__init__(**kwargs)
32/32 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.3543 - loss: 0.7056
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3662 - loss: 0.7122
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3485 - loss: 0.6976
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.2966 - loss: 0.7087
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6222 - loss: 0.7392
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6153 - loss: 0.7082
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3386 - loss: 0.7176
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3980 - loss: 0.7233
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3866 - loss: 0.7133
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4784 - loss: 0.6931
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4753 - loss: 0.6428
Test Accuracy: 0.4900
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 200ms/step
[[0.47517636 0.52482367]
 [0.504462   0.495538  ]
 [0.4892657  0.51073426]
 [0.49312642 0.50687355]
 [0.5224694  0.47753054]]
```

Result:

Thus the program to build a recurrent neural network with Keras/TensorFlow has been executed successfully.