**Experiment 8**

**Aim: Design and implement GRU model with tensorflow/keras and check accuracy**

This program demonstrates the use of a Gated Recurrent Unit (GRU)-based deep learning model for binary sequence classification. The aim is to show how GRUs process sequential data and classify it into one of two categories.

#### ****Description:****

GRU stands for Gated Recurrent Unit, which is a type of recurrent neural network (RNN) architecture that is similar to LSTM (Long Short-Term Memory).

Like LSTM, GRU is designed to model sequential data by allowing information to be selectively remembered or forgotten over time. However, GRU has a simpler architecture than LSTM, with fewer parameters, which can make it easier to train and more computationally efficient.

The main difference between GRU and LSTM is the way they handle the memory cell state. In LSTM, the memory cell state is maintained separately from the hidden state and is updated using three gates: the input gate, output gate, and forget gate. In GRU, the memory cell state is replaced with a “candidate activation vector,” which is updated using two gates: the reset gate and update gate.

The reset gate determines how much of the previous hidden state to forget, while the update gate determines how much of the candidate activation vector to incorporate into the new hidden state.

**Model Architecture:**

A **GRU layer** with 32 units processes sequential data, capturing temporal dependencies.

A **Dense layer** with a sigmoid activation function outputs probabilities for binary classification.

**Dataset:**

Input: Randomly generated sequences of numerical data with shape (100, 10, 1).

Labels: Randomly generated binary labels (0 or 1).

**Training Configuration:**

* + Loss Function: binary\_crossentropy to measure error in binary classification.
  + Optimizer: adam for efficient and adaptive gradient descent.
  + Metric: Accuracy to evaluate model performance.
  + Batch Size: 32.
  + Epochs: 10.

**Program:**

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import GRU, Dense, Dropout

from tensorflow.keras.optimizers import Adam

from sklearn.model\_selection import train\_test\_split

import numpy as np

# Data Preparation

X = np.random.rand(1000, 10, 1) # 1000 samples, 10 timesteps, 1 feature

y = np.random.randint(0, 2, (1000, 1)) # Binary target labels

# Split data into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Define the model with best hyperparameters

model = Sequential([

GRU(64, input\_shape=(10, 1)), # Best GRU Units

Dropout(0.3), # Best Dropout Rate

Dense(1, activation='sigmoid') # Output layer for binary classification

])

# Compile the model

model.compile(optimizer=Adam(learning\_rate=0.0001), loss='binary\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train, epochs=20, batch\_size=64, validation\_data=(X\_val, y\_val))

# Evaluate the model

loss, accuracy = model.evaluate(X\_val, y\_val)

print(f"Validation Loss: {loss}, Validation Accuracy: {accuracy}")

# Optional: Visualize training and validation performance

import matplotlib.pyplot as plt

# Plot training and validation accuracy

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.title('Accuracy vs Epochs')

plt.show()

# Plot training and validation loss

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.title('Loss vs Epochs')

plt.show()

**Output:**

**Result Analysis:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GRU Units** | **Dropout Rate** | **Learning Rate** | **Batch Size** | **Training Loss** | **Validation Loss** | **Training Accuracy** | **Validation Accuracy** |
| 32 | 0.1 | 0.01 | 16 |  |  |  |  |
| 32 | 0.2 | 0.001 | 32 |  |  |  |  |
| 32 | 0.3 | 0.0001 | 64 |  |  |  |  |
| 64 | 0.1 | 0.01 | 16 |  |  |  |  |
| 64 | 0.2 | 0.001 | 32 |  |  |  |  |
| 64 | 0.3 | 0.0001 | 64 |  |  |  |  |
| 128 | 0.1 | 0.01 | 16 |  |  |  |  |
| 128 | 0.2 | 0.001 | 32 |  |  |  |  |
| 128 | 0.3 | 0.0001 | 64 |  |  |  |  |

**Conclusion:**