9/10/25

Week-9

Aim:-
To implement a simple Recurrent Neural Network for sequential Data prediction and analyze its performance.

Objective:-

1.) understand the working of RNN's for sequential data.
2.) Train an RNN model on a time-series dataset.
3.) compare predicted and actual values to evaluate performance.

Algorithm:-

→ **Data preprocessing:-**
Normalize the datasets and splits into training and testing sets.

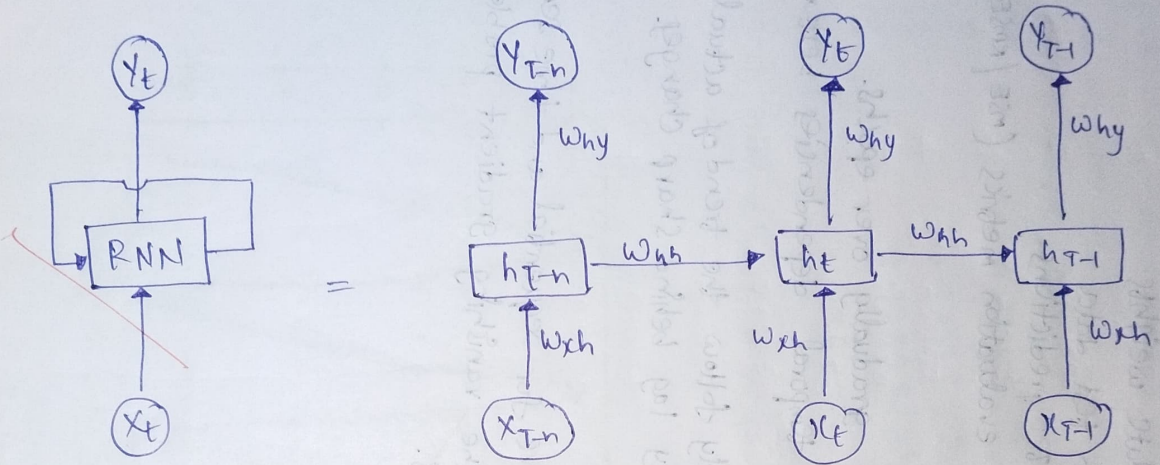
→ **Model design:-**
Define an RNN with Input, hidden and output layers.

→ **Training:-**
Feed sequences into the RNN, compute Loss and update weights using Back propagation through time.

→ **prediction:-**
use the trained RNN to predict the future values.

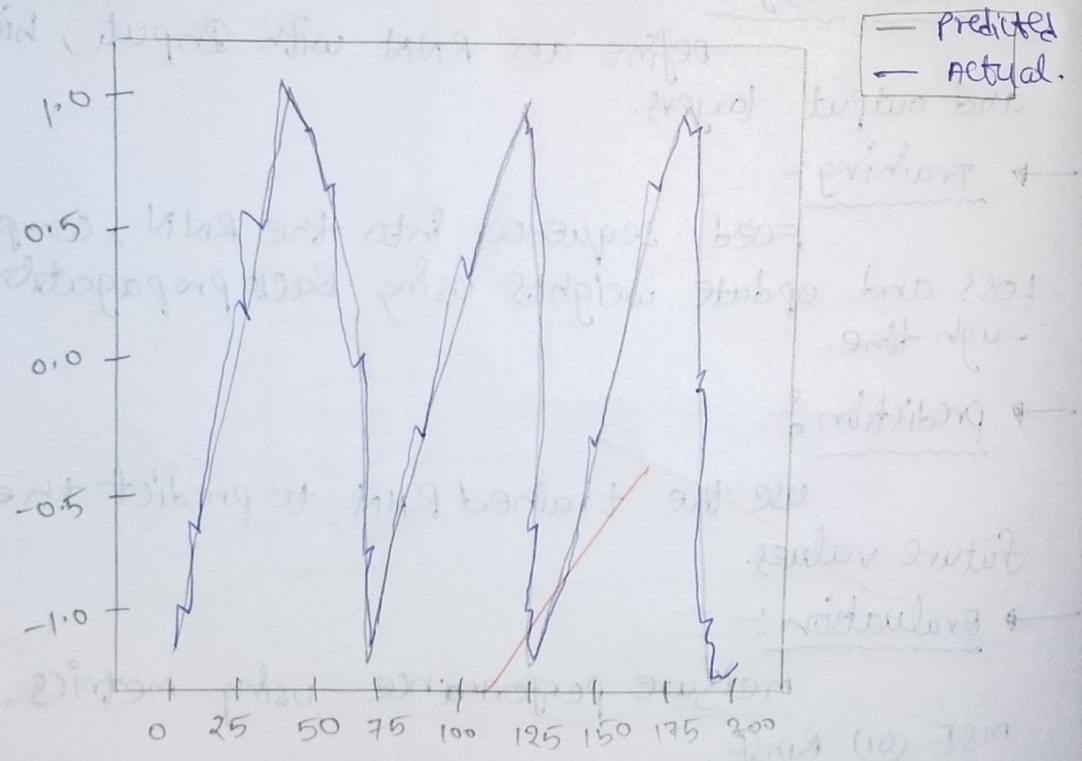
→ **Evaluation:-**
measure performance using metrics like MSE (or) RMSE.

A simple Architecture of the RNN Model



output :-

Epoch	step-loss	val_loss
1	0.5478	0.0334
2	0.0259	0.0181
3	0.0166	0.0196
4	0.0157	0.0172
5	0.0150	0.0161
6	0.0150	0.0153
7	0.0151	0.0178
8	0.0165	0.0157
9	0.0144	0.0170
10	0.0152	0.0153



Pseudo code :-

Load dataset
Normalize Data
split dataset into train and test
Initialize RNN model
for each epoch:
for each batch in training data:
predict output
compute Loss
BACKpropagate loss through time
update weights.
predict on test data
Denormalize predictions
Calculate evaluation metrics.

Observation :-

*) Loss decreases gradually with training, but may faster than LSTM.

*) RNN captures temporal patterns, but struggles with long-term dependencies.

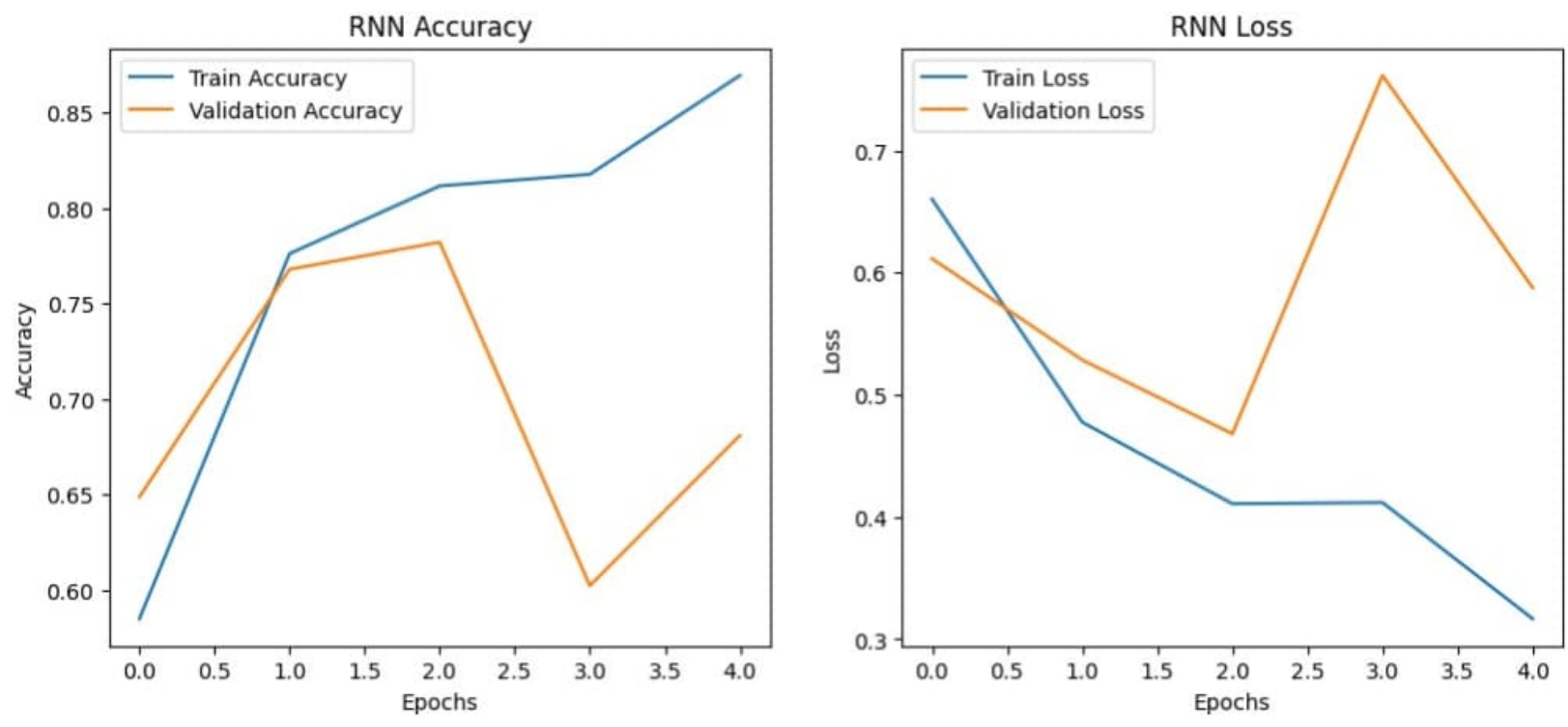
*) predictions follow general trends but may miss sharp fluctuations.

Conclusion :-

*) RNN's are useful for sequential and time-series prediction but have limitations in learning long term dependencies.

```python
# RNN Implementation
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense, Dropout
import matplotlib.pyplot as plt

# Load IMDB dataset
vocab_size = 10000
maxlen = 200

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)

# Build RNN model
rnn_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=maxlen),
    SimpleRNN(128, dropout=0.2),
    Dense(1, activation='sigmoid')
])

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

# Train model
rnn_history = rnn_model.fit(
    x_train, y_train,
    epochs=5,
    batch_size=64,
    validation_split=0.2
)

# Evaluate model
rnn_score = rnn_model.evaluate(x_test, y_test)
print(f"RNN Test Accuracy: {rnn_score[1]*100:.2f}% | Loss: {rnn_score[0]:.4f}")

# Plot Accuracy and Loss
plt.figure(figsize=(12,5))

# Accuracy
plt.subplot(1,2,1)
plt.plot(rnn_history.history['accuracy'], label='Train Accuracy')
plt.plot(rnn_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('RNN Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1,2,2)
plt.plot(rnn_history.history['loss'], label='Train Loss')
plt.plot(rnn_history.history['val_loss'], label='Validation Loss')
plt.title('RNN Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
313/313 ━━━━━━━━━━━━━━━━━━━━ 48s 147ms/step - accuracy: 0.5433 - loss: 0.6817 - val_accuracy: 0.6488 - val_loss: 0.6114
Epoch 2/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 80s 141ms/step - accuracy: 0.7572 - loss: 0.5022 - val_accuracy: 0.7680 - val_loss: 0.5285
Epoch 3/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 44s 140ms/step - accuracy: 0.8231 - loss: 0.4031 - val_accuracy: 0.7822 - val_loss: 0.4681
Epoch 4/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 45s 143ms/step - accuracy: 0.8235 - loss: 0.4016 - val_accuracy: 0.6022 - val_loss: 0.7613
Epoch 5/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 81s 140ms/step - accuracy: 0.8476 - loss: 0.3546 - val_accuracy: 0.6810 - val_loss: 0.5877
782/782 ━━━━━━━━━━━━━━━━━━━━ 18s 23ms/step - accuracy: 0.6824 - loss: 0.5871
RNN Test Accuracy: 68.15% | Loss: 0.5839
```


RNN Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs
RNN Loss
Train Loss
Validation Loss
Loss
Epochs