Topic: Human Activity Recognition using Recurrent Neural Networks
Domain: Sensor-Based Time-Series Classification
Dataset: UCI Human Activity Recognition Using Smartphones
Course: Machine Learning
Assignment: 5.4
Group Members:

- Eman Shahbaz (BITF22M516)
- Dua Sarwar (BITF22M530)

1. Objective

The objective of this assignment is to implement **Recurrent Neural Network (RNN)-based classification** for time-series sensor data. Unlike previous deliverables that relied on engineered or flattened features, this assignment focuses on preserving **temporal dependencies** in raw sensor signals. Three recurrent architectures were implemented and evaluated:

- Simple RNN
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

2. Dataset Description

The **UCI HAR Dataset** contains recordings from a smartphone's accelerometer and gyroscope sensors. The phone was worn on the waist while subjects performed daily activities.

Sensors used:

- 3-axis accelerometer (x, y, z)
- 3-axis gyroscope (x, y, z)

Activities (6 classes):

1. Walking
2. Walking Upstairs
3. Walking Downstairs

4. Sitting
5. Standing
6. Lying

Dataset size:

- Training samples: 7,352
- Testing samples: 2,947

Each sample is a **time window of 128 timesteps**.

3. Preprocessing for Sequential Data

For recurrent models, **raw inertial signals** were used from the *Inertial Signals* directory instead of pre-extracted feature vectors.
Each sample was represented as a **3D tensor**:
`(samples, timesteps, features) = (N, 128, 6)`
Where:

- Timesteps = 128
- Features = 6 sensor channels (acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z)

Normalization

Signals were normalized **per channel** using training-set mean and standard deviation to ensure stable learning and avoid scale imbalance.

4. Recurrent Models Implemented

Three recurrent architectures were implemented using TensorFlow/Keras:

4.1 Simple RNN

- Single recurrent layer
- Suffers from vanishing gradient problem
- Serves as a baseline recurrent model

4.2 LSTM

- Uses input, forget, and output gates
- Designed to capture long-term temporal dependencies
- More stable than Simple RNN

4.3 GRU

- Simplified gated architecture
- Fewer parameters than LSTM
- Faster convergence and good generalization

All models used:

- Adam optimizer
- Sparse categorical cross-entropy loss
- Softmax output layer for 6-class classification

5. Evaluation Metrics

For **each model**, the following metrics were reported:

- Accuracy
- Weighted F1-score
- Confusion Matrix
- Training vs Validation Loss Curves

6. Results

6.1 RNN Model Comparison

Model	Accuracy	Weighted F1
Simple RNN	0.4404	0.3866
LSTM	0.6339	0.5590
GRU	**0.6624**	**0.6179**

Best Recurrent Model: GRU

6.2 Training vs Validation Behavior

- **Simple RNN** showed unstable convergence and underfitting, with poor validation performance.
- **LSTM** demonstrated smoother training and improved validation loss.
- **GRU** achieved the most stable learning curves, indicating better generalization and efficient learning.

7. Comparison with Fully Connected Neural Network

Model	Accuracy
FCNN (Assignment 5.3)	**0.9437**
Best RNN (GRU)	0.6624

Accuracy Difference:

```
0.6624 - 0.9437 = -0.2813
```

Interpretation

The FCNN outperformed all recurrent models because it was trained on **engineered, window-based features** that already encode discriminative motion patterns. In contrast, RNN-based models operate on **raw sensor signals**, which are noisier and require more complex temporal modeling and tuning.

8. Limitations of Recurrent Models

- Simple RNN suffers from vanishing gradients.
- LSTM and GRU increase computational cost.
- Training time is longer compared to FCNN.
- Performance is sensitive to hyperparameter tuning.
- RNNs may underperform when datasets already provide strong engineered features.

9. Conclusion

This assignment successfully implemented and evaluated **Simple RNN, LSTM, and GRU** models for time-series classification using raw sensor data. GRU achieved the best recurrent performance, while FCNN from Assignment 5.3 remained superior due to richer feature representations. The results highlight the trade-offs between sequential modeling and feature-based learning.