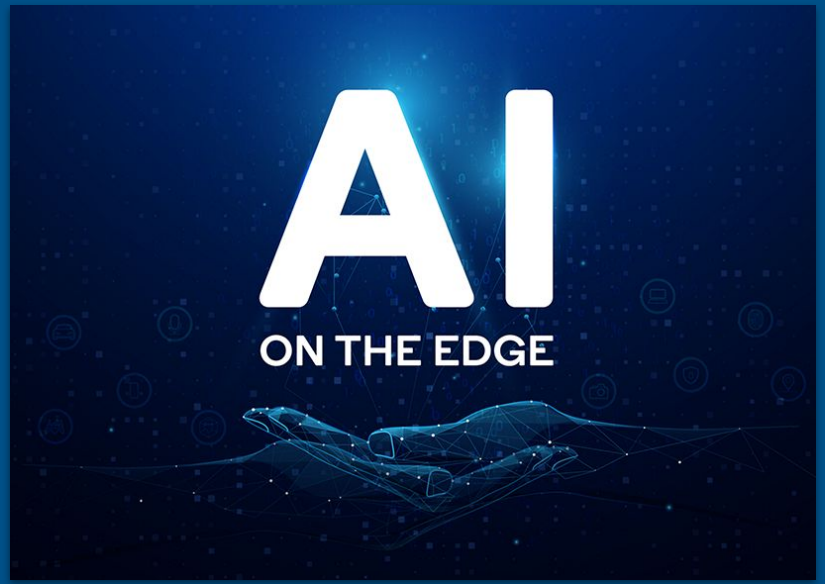
07-03-25

Sequence Models in TinyML

P A Gautham Nair
DSA 2024 Batch4
ICT Academy of Kerala



Problem Statement

- Deploying sequence models like RNN, LSTM, and GRU in TinyML is challenging due to limited computational power, memory, and energy constraints. Optimizing these models while maintaining accuracy and efficiency is crucial, but the trade-offs between model complexity, inference speed, and size remain unclear. A comparative study is needed to evaluate their performance in TinyML environments and identify the most suitable model for real-time, resource-constrained applications.

Objective

- To conduct a comparative study of RNN, LSTM, and GRU models in TinyML by evaluating their accuracy, inference time, and model size.

- To analyze the impact of model optimizations (TFLite conversion and quantization) on performance to determine the most suitable model for real-time TinyML applications.

Dataset

Human Activity Recognition Using Smartphones

Authors : Jorge Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra
https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones

📌 **Description:**

- Collected from 30 individuals performing six activities (walking, walking upstairs, walking downstairs, sitting, standing, lying) while carrying a smartphone(Samsung Galaxy S II).
- Sensor data from the **accelerometer** and **gyroscope** was recorded at **50Hz**.

📌 **Features:**

- **Raw data** from the accelerometer and gyroscope
- **561 time and frequency domain features** extracted from raw sensor signals.
- Includes **mean, standard deviation, signal magnitude area, correlation, and more**

📌 **Use in This Study:**

- Used to train and compare **RNN, LSTM, and GRU** models for TinyML.
- Evaluated models on **accuracy, inference time, and model size** after optimization (TFLite & quantization).

Preprocessing

Merging Sensor Data & Assigning Labels

📌 Merging Raw Sensor Data

- The dataset consists of multiple text files, each representing accelerometer and gyroscope readings for individual experiments across 30 participants.
- We combined all these files into a **single dataset**, ensuring sensor readings were aligned.

📌 Assigning Activity Labels

- labels.txt provides mapping information: [experiment_id, user_id, activity_id, start_sample, end_sample].
- For each experiment, activity labels were assigned to corresponding time steps based on the **start and end sample indices**.
- This resulted in a structured dataset with sensor readings and labeled activities.

Preprocessing

Checking Data Balance & Applying SMOTE

📌 Imbalance in Activity Labels

- The dataset contains **12 distinct activity labels**, but their distribution was highly imbalanced.
- Some activities had significantly fewer samples, which could bias model training.

📌 Balancing Data Using SMOTE

- **Synthetic Minority Over-sampling Technique** (SMOTE) was applied to generate synthetic samples for underrepresented activities.
- This helped ensure **all activities had a balanced representation**, improving model generalization and performance.

Preprocessing

Preparing Data for Sequence Models

📌 Need for Sequential Input

- RNNs, LSTMs, and GRUs require sequential input rather than individual time points.
- We **transformed raw sensor data into fixed-length time steps** to capture temporal dependencies.

📌 Creating Overlapping Sequences

- Each sequence contains **50 time steps** (1 second of sensor data at 50Hz).
- The label assigned to a sequence corresponds to the **last time step** in that window.

📌 Final Data Shape

- Transformed dataset has the shape **(Samples, Time Steps, Features) → (N, 50, 6)**.
- This ensures that models learn from continuous motion patterns instead of isolated readings.

RNN Model

Recurrent Neural networks have loops that allow information to persist, making them well-suited for tasks involving time-series data, speech recognition, and natural language processing.

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 50, 256)	67,328
dropout (Dropout)	(None, 50, 256)	0
simple_rnn_1 (SimpleRNN)	(None, 50, 128)	49,280
dropout_1 (Dropout)	(None, 50, 128)	0
simple_rnn_2 (SimpleRNN)	(None, 64)	12,352
dense (Dense)	(None, 13)	845

Total params: 129,805 (507.05 KB)

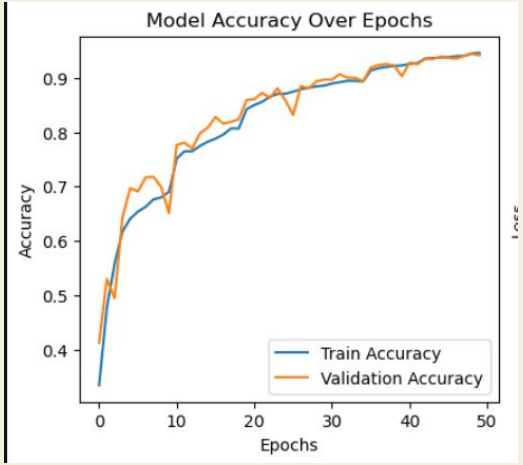
Trainable params: 129,805 (507.05 KB)

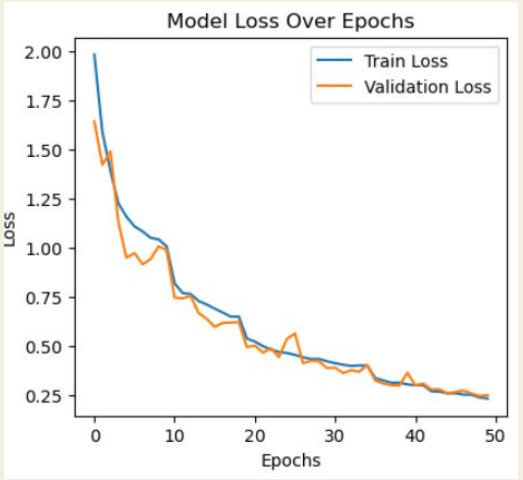
Non-trainable params: 0 (0.00 B)

RNN Full Model Performance

Accuracy: 94.46%
F1-score: 94.24%
Inference Time: 0.581 ms/sample

- The accuracy improves steadily, reaching above **90%** after 50 epochs.
- The training and validation accuracy curves follow a similar trend, indicating **minimal overfitting**.
- The validation loss closely follows the training loss, suggesting **good generalization**.





LSTM Model

LSTM is an advanced type of RNN designed to overcome the vanishing gradient problem. It introduces **gates (forget, input, and output gates)** that regulate information flow, allowing the network to remember long-term dependencies.

Layer (type)	Output Shape
lstm (LSTM)	(None, 50, 128)
dropout_2 (Dropout)	(None, 50, 128)
lstm_1 (LSTM)	(None, 50, 64)
dropout_3 (Dropout)	(None, 50, 64)
lstm_2 (LSTM)	(None, 32)
dense_1 (Dense)	(None, 64)
dropout_4 (Dropout)	(None, 64)
dense_2 (Dense)	(None, 13)

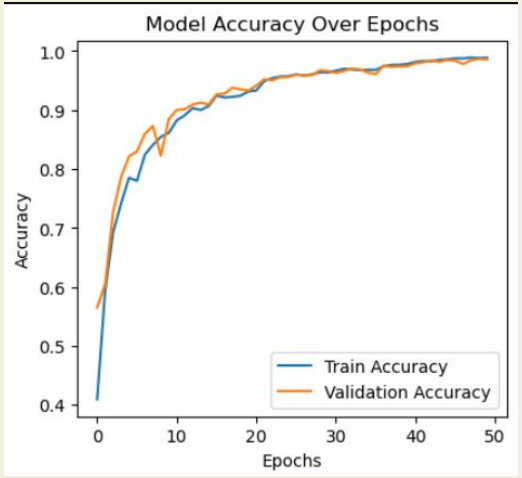
Total params: 133,901 (523.05 KB)
Trainable params: 133,901 (523.05 KB)
Non-trainable params: 0 (0.00 B)

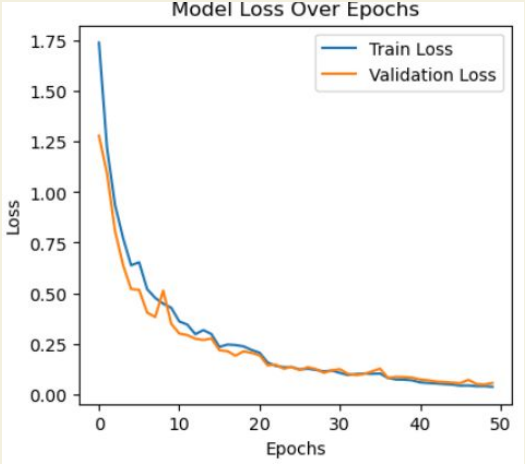
LSTM Full Model Performance

- Accuracy: 98.35%
- F1-score: 98.34%
- Inference Time: 0.554 ms/sample

Model Loss Over Epochs

- Loss declines consistently and approaches **zero**, implying effective model convergence.
- The validation loss follows the training loss, showing **good generalization**.





GRU Model

GRU is a simplified version of LSTM that retains efficiency while maintaining performance. It has only **two gates (reset and update gates)** instead of three, reducing computational cost.

GRUs are computationally faster than LSTMs and work well in **sequential modeling tasks**, making them a popular choice for real-time applications.

Layer (type)	Output Shape
gru (GRU)	(None, 50, 128)
dropout_5 (Dropout)	(None, 50, 128)
gru_1 (GRU)	(None, 50, 64)
dropout_6 (Dropout)	(None, 50, 64)
gru_2 (GRU)	(None, 32)
dense_3 (Dense)	(None, 64)
dropout_7 (Dropout)	(None, 64)
dense_4 (Dense)	(None, 13)

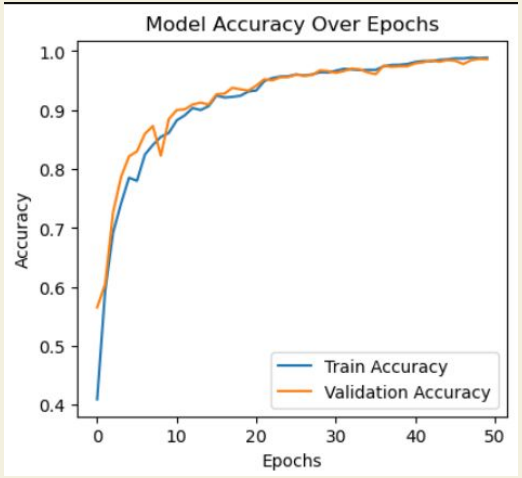
Total params: 101,837 (397.80 KB)

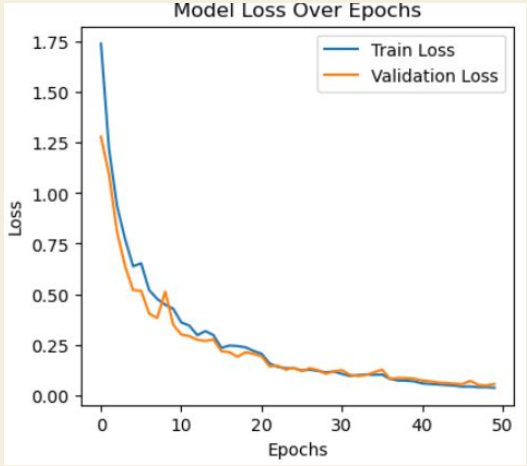
Trainable params: 101,837 (397.80 KB)

Non-trainable params: 0 (0.00 B)

GRU Full Model Performance

- accuracy: 0.9894 - loss: 0.0384
- The training and validation accuracy curves closely align, suggesting **minimal overfitting**.
- Loss declines consistently and approaches **zero**, implying effective model convergence.
- The validation loss follows the training loss, showing **good generalization**.





Downsizing Models for TinyML

📌 **Using TFLite Conversion**

- Convert full **RNN, LSTM, and GRU** models to **TensorFlow Lite (TFLite)** format.
- Reduces **model size** while maintaining inference capabilities.
- Ideal for **mobile & embedded applications**.

📌 **Applying Quantization for Further Optimization**

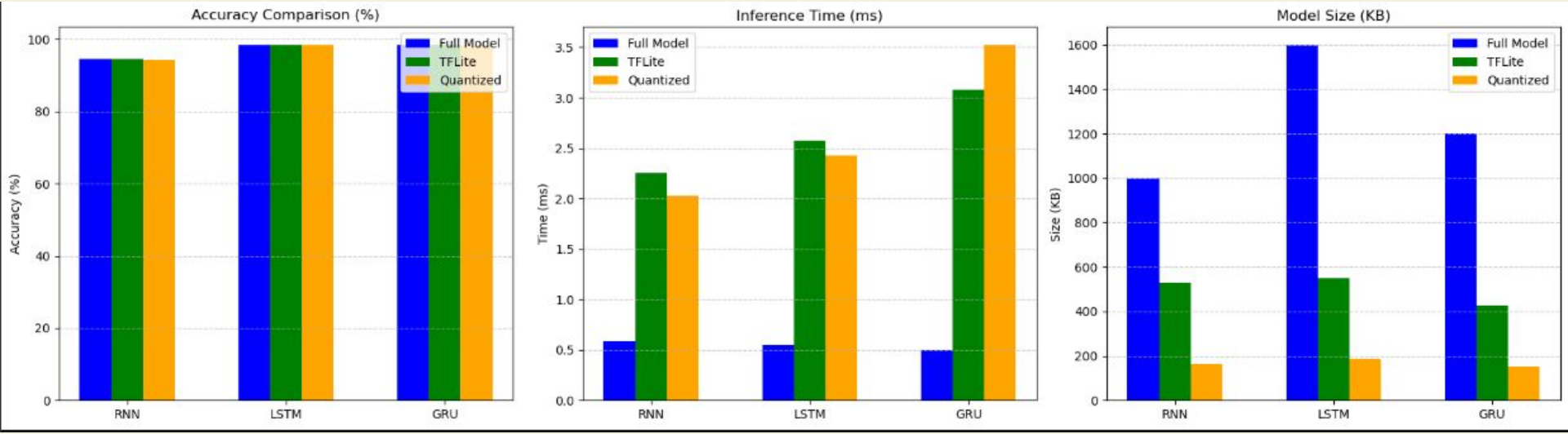
- **Post-training quantization**: minimizing memory footprint.
- **Benefits:**
  Smaller model size (up to **4x** reduction).
  Faster inference with lower power consumption.
  Minimal accuracy loss

Combining **TFLite + Quantization** enables **efficient deployment** of deep learning models on TinyML hardware with optimal speed and memory usage.

Comparative Study

Model	TFLite Model	Quantized Model
RNN	Accuracy: 94.46%	Accuracy: 94.38%
	Inference Time: 2.200 ms	Inference Time: 2.034 ms
	Model Size: 527.84 KB	Model Size: 162.03 KB
LSTM	Accuracy: 98.35%	Accuracy: 98.31%
	Inference Time: 2.587 ms	Inference Time: 2.587 ms
	Model Size: 550.32 KB	Model Size: 185.17 KB
GRU	Accuracy: 98.50%	Accuracy: 98.35%
	Inference Time: 3.076 ms	Inference Time: 2.928 ms
	Model Size: 427.55 KB	Model Size: 152.16 KB

Comparative Study



Conclusion

Quantization significantly reduces model size, making it ideal for TinyML deployment, with models shrinking by **6-10x** while maintaining **high accuracy** (minimal drop of **0.04%-0.15%**). However, inference time increases post-quantization, with **GRU being the slowest**, making **RNN and LSTM more efficient choices** for real-time applications. TFLite conversion alone retains full accuracy while cutting model size by **~50%**, but quantization provides the best trade-off between **memory efficiency and performance**. Among the three models, **LSTM strikes the best balance** between accuracy, size reduction, and inference speed, making it the most suitable for TinyML applications.