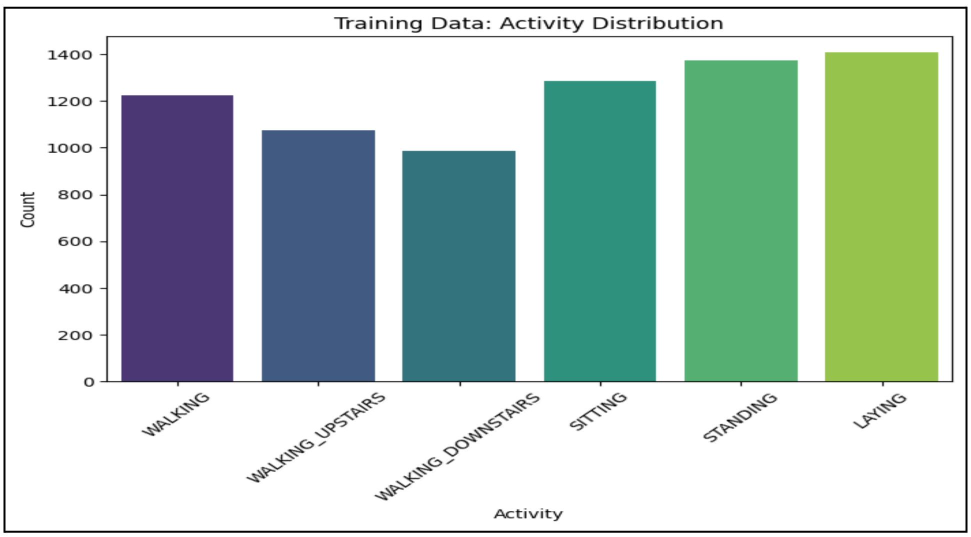
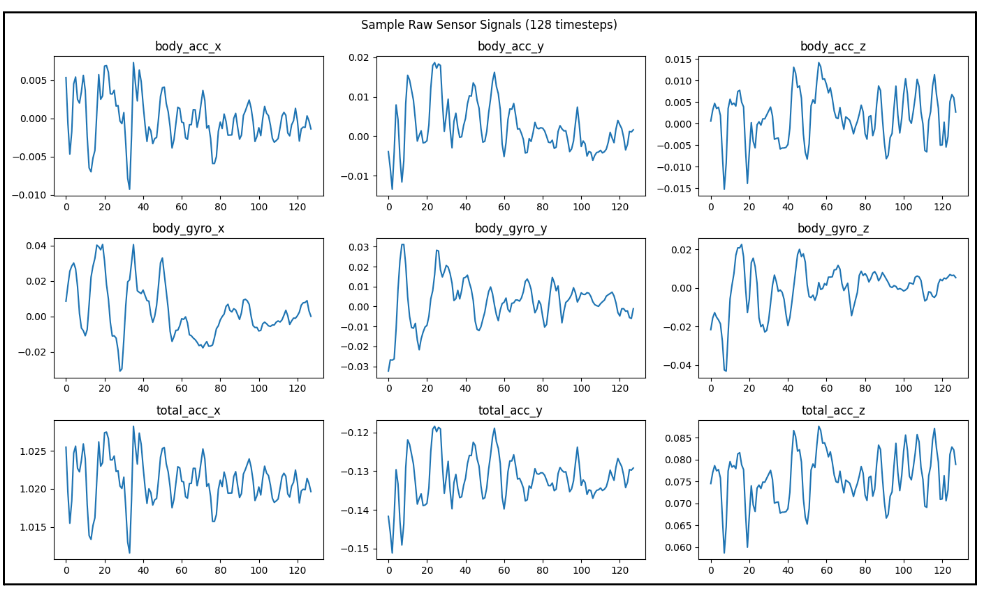
Approach & Observations - Task #1
Human Activity Recognition

1. Data Processing

- I utilized **raw accelerometer and gyroscope data** instead of precomputed features to retain full signal information and allow deep learning models to learn representations directly.
- I applied **normalization and reshaping** to standardize input across subjects, preventing model bias due to varying signal magnitudes.
- **Feature Extraction:** I used **TSFEL** to derive statistical and frequency-based features for machine learning models, ensuring robust feature representation without manual engineering.



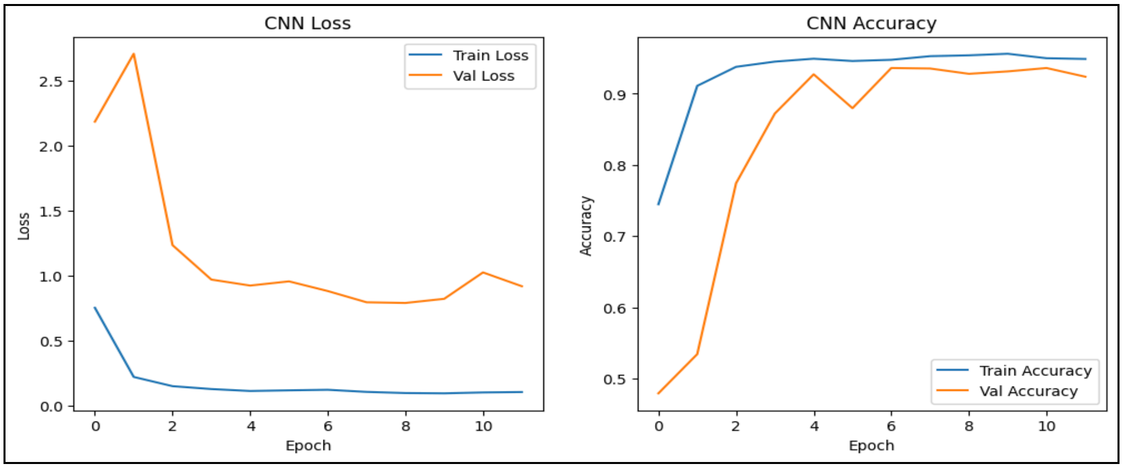


2. Deep Learning Models

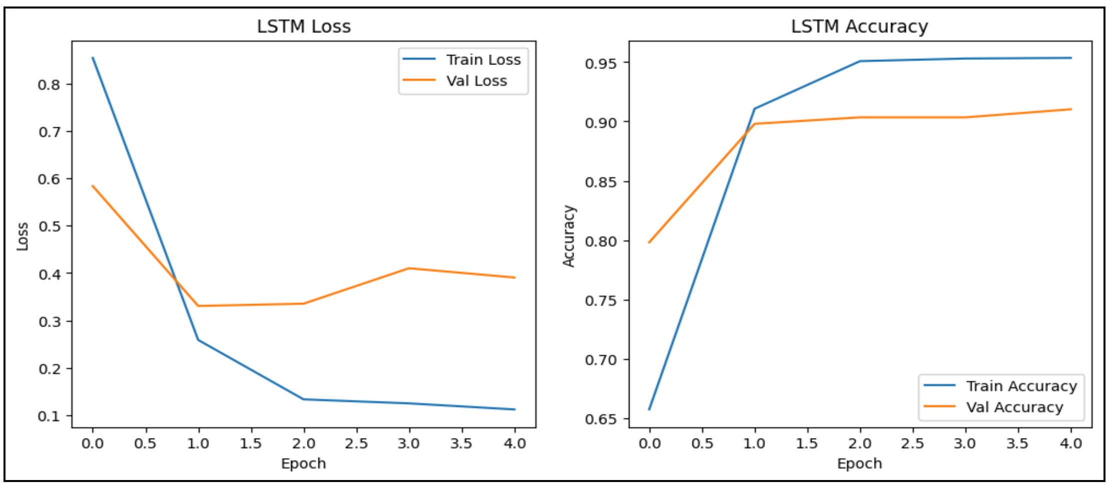
- **LSTM:** It was selected due to its ability to capture temporal dependencies, which is crucial for sequential sensor data.
    - I used **multiple LSTM layers** to enhance long-range dependency capture.
    - Dropout layers were added to prevent overfitting.
    - **Hidden units** were tuned to balance model complexity as well as training efficiency.
    - **Bidirectional LSTMs** were considered but not implemented due to increased computational cost without significant accuracy gains.
- **1D CNN:** It was chosen for its efficiency in learning local patterns and reducing computational complexity.
    - **Kernel size and stride** were optimized to extract short-term activity patterns.
    - **ReLU activation** ensured non-linearity in feature extraction.
    - Batch normalization was used to stabilize training and speed up convergence.
    - **Number of filters** was tuned to balance computational efficiency and representational power.

- **Training Details:**
    - Models were trained directly on raw sensor readings without feature engineering to maintain data integrity.
    - **Adam optimizer** was used for adaptive learning rate optimization, balancing speed and accuracy.
    - **Learning rate scheduling** was employed to adjust learning dynamically, improving convergence.
    - **Batch size and epochs** were fine-tuned using cross-validation to prevent overfitting and underfitting.
    - **Early stopping** was implemented to halt training when validation loss stopped improving, preventing unnecessary computation.

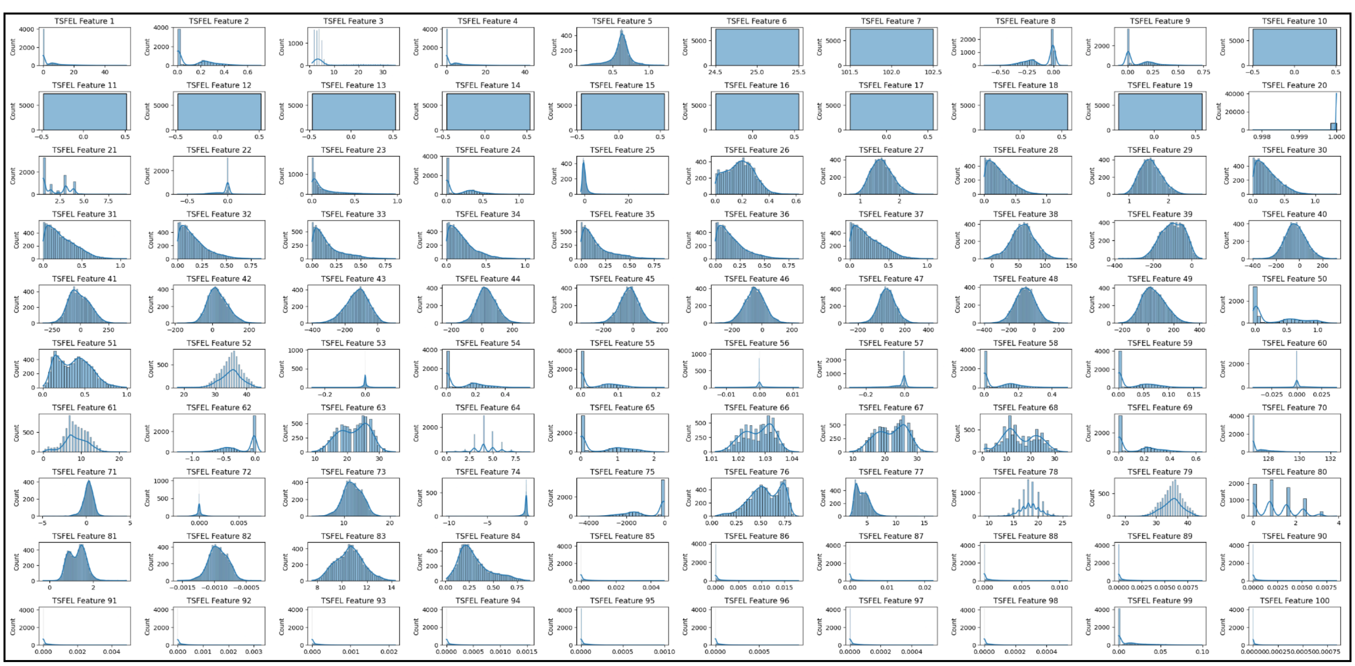
- **Observations:**
    - LSTMs demonstrated superior performance in learning long-term dependencies but required more training time.
    - CNNs effectively captured short-term activity patterns and were computationally efficient.
    - **Regularization techniques**, including dropout and L2 weight decay, were applied to prevent overfitting.



CNN: Loss and Accuracy



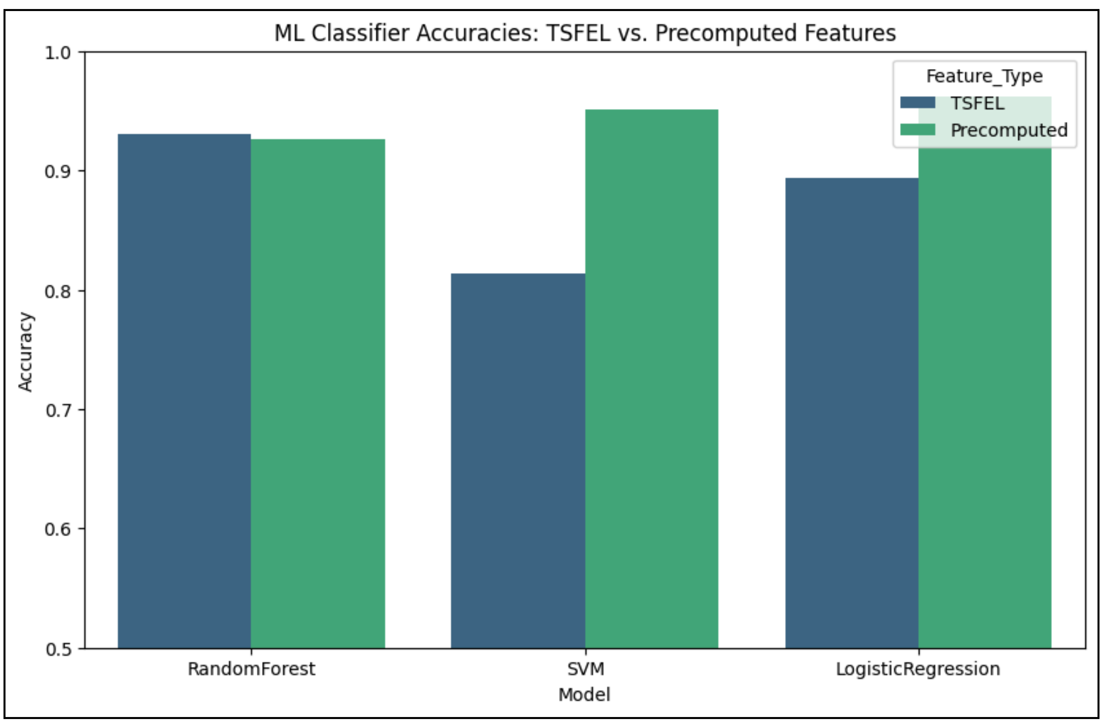
LSTM: Loss and Accuracy



TSFEL Feature Distribution

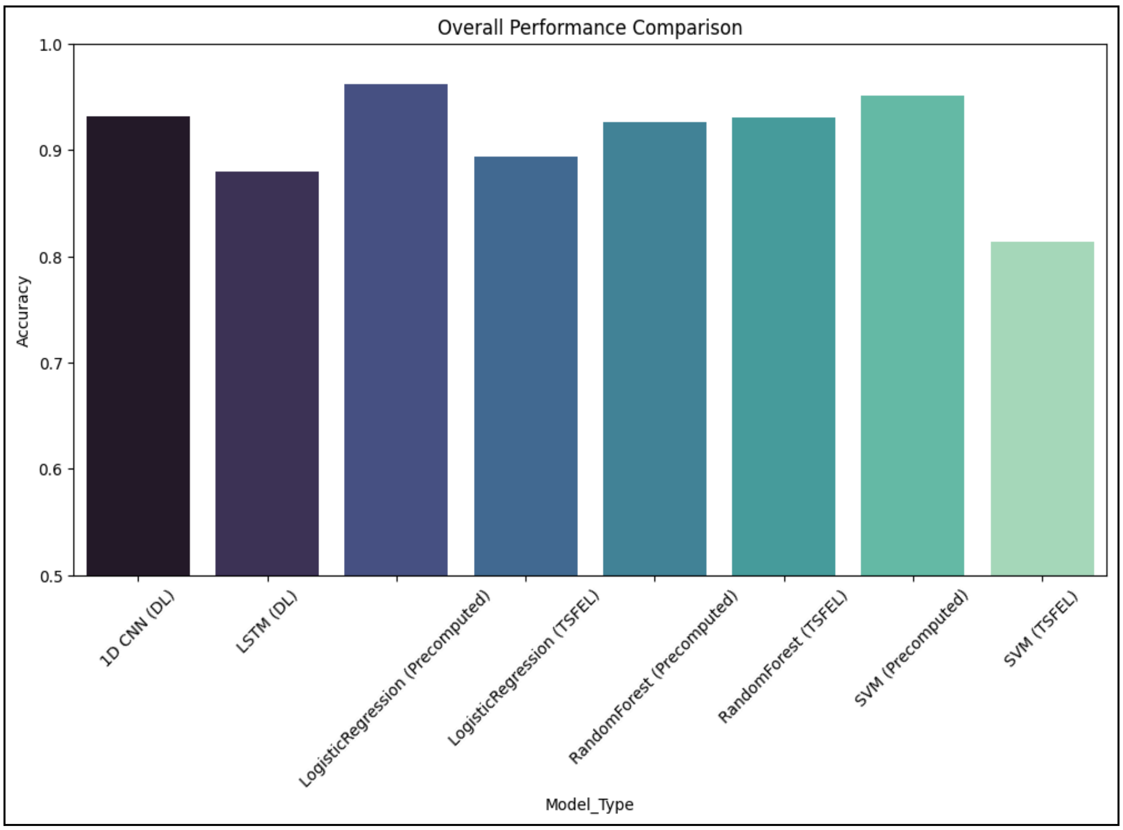
3. Machine Learning Models

- **Feature Engineering:** TSFEL extracted statistical and spectral features from raw sensor data to provide meaningful input for classical models.
- **Models Implemented:**
  - **Random Forest:** Selected for its ability to handle feature complexity and robustness to noise.
  - **SVM:** Effective for high-dimensional data but computationally expensive, requiring careful hyperparameter tuning.
  - **Logistic Regression:** Chosen as a baseline due to its simplicity and interpretability, though less effective for complex patterns.
- **Hyperparameter Tuning:**
  - **Random Forest:** Optimized the number of trees and depth to prevent overfitting while ensuring sufficient complexity.
  - **SVM:** Explored different kernel functions (linear, RBF) to find the best balance between accuracy and computation.
  - **Logistic Regression:** Adjusted regularization parameters to improve generalization.
- **Comparison:**
  - **Random Forest achieved the highest accuracy among ML models** due to its ability to model complex relationships and handle nonlinear separability.
  - **SVM performed well but required extensive computational resources for large feature sets.**



4. Overall Performance Comparison

Model	Feature Set	Accuracy (%)
LSTM	Raw Sensor Data	87.98
1D CNN	Raw Sensor Data	93.18
Random Forest	TSFEL Features	93.04
SVM	TSFEL Features	81.30
Logistic Regression	TSFEL Features	89.38



5. Key Findings & Future Scope

We clearly observe that:

- **End-to-End Deep Learning:** Deep learning models automatically learn temporal patterns from raw sensor data and do not require manual feature engineering.
- **Feature Engineering for ML:** With proper preprocessing and missing-value handling, TSFEL-generated features can match or even surpass the performance of the original 561-dimensional engineered features.
- **Trade-offs:** Deep learning approaches require more computation and data, whereas ML models on engineered features may offer faster training and inference.

Future Scope:

- We may investigate **transformer-based architectures** for improved sequence modeling in HAR.
- We can also optimize deep learning models for **real-time activity recognition** in wearable sensor applications.
- We can explore **hybrid models combining deep learning and ML approaches** to balance performance and interpretability.
- We can experiment with **attention mechanisms** to enhance feature extraction in deep learning architectures.

Submitted by:
Subhrajit Deb
SRIP Intern ID: 21925900
Email: subhrajitd28@gmail.com
Phone: +919863239510
Github Link to Task Submission: https://github.com/dhundhun1111/UCI-HAR.git
Resume Link: Subhrajit_resume.pdf