Q1

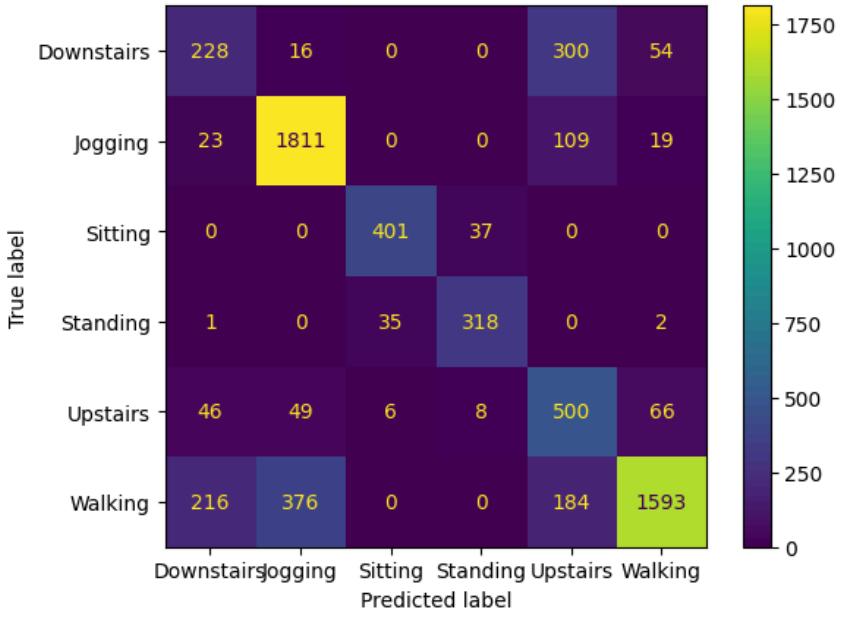
Project Objective

The objective of this project is to develop a Human Activity Recognition (HAR) system that classifies human activities using accelerometer data. A three-layer Multi-Layer Perceptron (MLP) model was designed to classify six activities (“downstairs”, “jogging”, “sitting”, “standing”, “upstairs”, and “walking”) based on ten features extracted from three-axis accelerometer signals. The trained model was deployed and executed on an embedded system using the Mbed environment.

Model Training (Python / TensorFlow)

The model was trained using the WISDM accelerometer dataset in a Python environment. Raw sensor data were segmented into windows of 80 samples with a step size of 40. From each window, ten statistical features were extracted and used as input to the classifier.

The MLP model consists of an input layer with 10 neurons, two hidden layers with 100 neurons each using ReLU activation, and an output layer with 6 neurons using Softmax activation. The model was trained for 50 epochs using the Adam optimizer and categorical cross-entropy loss function. After training, the Keras model was converted to TensorFlow Lite format and then exported as a C array for embedded deployment.



Mbed Integration

The embedded implementation was performed on the STM32 DISCO-F746NG development board, featuring an ARM Cortex-M7 processor running at 216 MHz. The model was integrated into an Mbed OS project and executed using optimized memory settings to fit the available RAM and Flash constraints.

During integration, several configuration challenges were addressed, including toolchain compatibility and memory optimization. After resolving these issues, the model was successfully compiled and executed on the target hardware.

Experimental Results

The project was successfully built, and the generated binary was deployed to the development board. Inference was performed using test feature vectors, and prediction results were printed via the serial interface. LED indicators were used to provide visual feedback for different activity classes. The system demonstrated reliable inference performance with acceptable latency on the embedded platform.

Lessons Learned

This project highlighted the challenges of deploying machine learning models on resource-constrained embedded systems. Key lessons include the importance of model size optimization, careful toolchain configuration, and efficient integration of machine learning workflows with embedded software development.

Conclusion

In this project, a machine learning–based human activity recognition system was successfully developed and deployed on an embedded platform. The results demonstrate the feasibility of running neural network models on Mbed OS–based microcontroller systems for practical HAR applications.

Q2

Project Objective

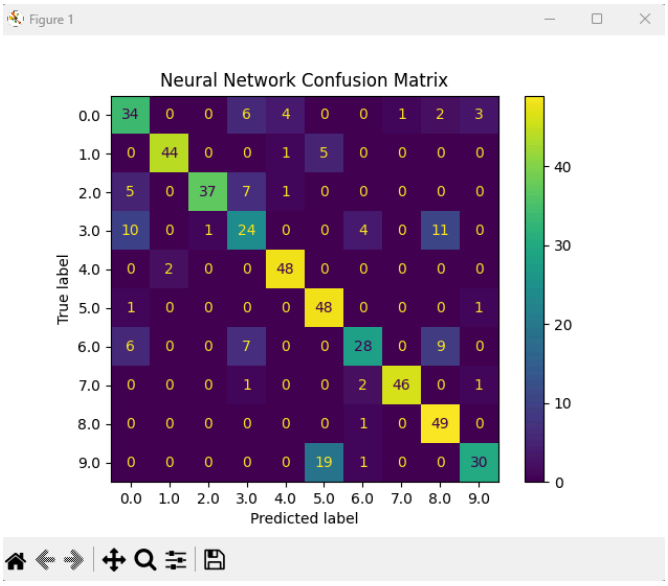
The objective of this project is to develop and deploy a machine learning–based temperature prediction system on an embedded platform. A Multi-Layer Perceptron (MLP) model was designed to predict future room temperature values based on a fixed number of past temperature measurements. The trained model was converted and integrated into an Mbed OS environment and executed on an STM32-based microcontroller.

Model Training (Python / TensorFlow)

The model was developed and trained in a Python environment using TensorFlow and Keras. Historical temperature data were loaded and preprocessed in `main.py`. To capture temporal dependencies, the dataset was transformed into input–output pairs using a sliding window approach, where a fixed number of previous temperature samples were used to predict the next temperature value.

The MLP model consists of an input layer corresponding to the selected number of previous temperature values, one or more hidden layers with ReLU activation functions, and a single-neuron output layer that produces a continuous temperature prediction. The model was trained using the Adam optimizer and mean absolute error as the loss function. After training, the model was evaluated on a test dataset to verify prediction accuracy.

Once training was completed, the Keras model was converted to TensorFlow Lite format and exported as a C-compatible array (`.cc` and `.h` files) for deployment on the embedded system.



Feature Processing

Supporting signal-processing and feature-handling functionality was implemented in `mfcc_func.py`. Although originally designed for signal-based feature extraction, this module provided reusable preprocessing utilities and demonstrated modular separation between data handling and model training logic. This structure improved code clarity and maintainability.

Mbed Integration

The embedded implementation was performed using Mbed OS on the STM32F746NG development board, which features an ARM Cortex-M7 processor running at 216 MHz. The TensorFlow Lite Micro framework was used to execute the neural network model on the microcontroller.

The generated model files were included in the Mbed project and loaded into memory as constant C arrays. A TensorFlow Lite Micro interpreter was configured with a statically allocated tensor arena to meet the memory constraints of the embedded platform. Careful selection of tensor arena size and operator resolvers ensured successful model initialization and inference execution.

The application logic, implemented in `main.cpp`, provided input data to the model, invoked inference, and retrieved the predicted temperature values.

Experimental Results

The project was successfully compiled and deployed to the STM32F746NG development board. Inference was performed using test temperature sequences, and predicted values were output via the serial console. The model executed reliably within the available memory limits and demonstrated stable inference behavior with low latency.

The results confirmed that the trained MLP model could be executed on a resource-constrained embedded system while maintaining acceptable prediction accuracy.

Lessons Learned

This project highlighted several important aspects of deploying machine learning models on embedded systems. Key lessons include the importance of simplifying model architecture for memory efficiency, managing toolchain and library compatibility, and carefully handling data formats when converting models from Python to embedded C representations. Additionally, the

project emphasized the value of modular software design when integrating machine learning workflows with embedded development environments.

Conclusion

In this project, a machine learning–based temperature prediction system was successfully developed, converted, and deployed on an embedded platform using Mbed OS. The results demonstrate that neural network models trained in high-level Python environments can be effectively integrated into microcontroller-based systems. This work confirms the feasibility of deploying predictive machine learning applications on embedded hardware for real-world sensing and monitoring tasks.

Q3

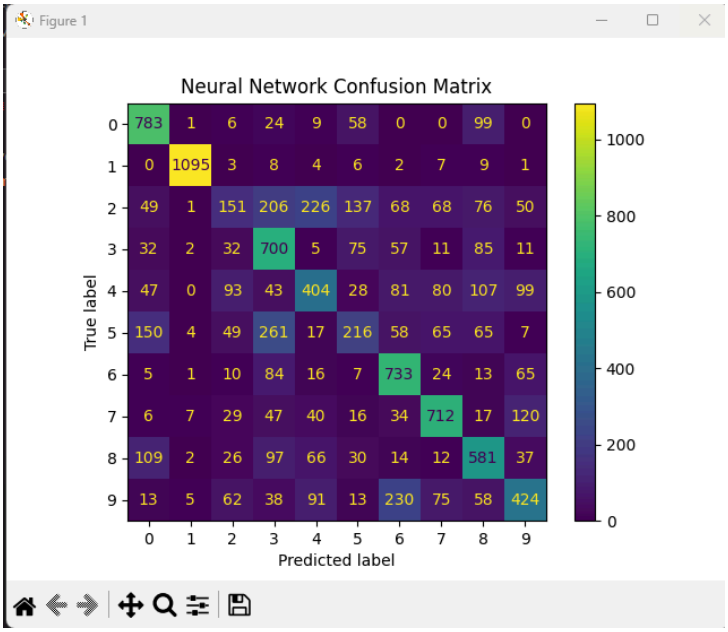
Project Objective

The objective of this project is to develop a handwritten digit recognition system capable of classifying digits from **0 to 9** using digital image data. Unlike binary classification tasks such as distinguishing “zero” versus “not zero,” this application addresses a **multi-class classification problem** by employing a multilayer neural network. The system is designed to extract discriminative features from grayscale digit images and use these features to accurately identify the corresponding digit class.

Dataset and Feature Extraction

The MNIST handwritten digit dataset was used in this study. The dataset consists of grayscale images of handwritten digits, each labeled with an integer value between 0 and 9. Instead of using raw pixel values directly, **Hu invariant moments** were extracted from each image to form a compact and rotation-invariant feature representation.

For every image, seven Hu moments were computed using OpenCV’s moment calculation functions. These seven features were used as inputs to the neural network, significantly reducing input dimensionality while preserving the shape characteristics of handwritten digits.



Neural Network Model

A multilayer perceptron (MLP) architecture with three layers was implemented for digit classification. The input layer accepts the seven Hu moment features. This is followed by two hidden layers, each containing 100 neurons and using the ReLU activation function to introduce nonlinearity and improve learning capability. The output layer consists of ten neurons, corresponding to the ten digit classes (0–9), and employs the Softmax activation function to produce class probabilities.

Since this is a multi-class classification task with integer labels, the model was trained using the **Sparse Categorical Cross-Entropy** loss function. The Adam optimizer was selected due to its efficient convergence properties.

Training and Evaluation

The neural network was trained using the training portion of the MNIST dataset. To improve training efficiency and avoid overfitting, two Keras callbacks were employed. The **ModelCheckpoint** callback was used to save the best-performing model during training, while **EarlyStopping** was applied to terminate training when the loss stopped improving for a predefined number of epochs.

After training, the model was evaluated on the test dataset. Predictions were obtained by selecting the class with the highest Softmax probability. A confusion matrix was generated to analyze classification performance across all digit classes, providing insight into misclassification patterns and overall accuracy.

Results and Discussion

The trained multilayer neural network successfully classified handwritten digits into ten distinct classes. The confusion matrix visualization demonstrated that most digits were recognized accurately, with higher confusion occurring between visually similar digits. The use of Hu moments proved effective in capturing essential shape information while keeping the feature vector compact.

Overall, the results confirm that a relatively simple MLP architecture, combined with carefully selected image features, can achieve reliable performance for handwritten digit recognition tasks.

Conclusion

In this project, a handwritten digit recognition system was implemented using a multilayer neural network and Hu moment features extracted from MNIST images. The model successfully performed multi-class classification across ten digit categories. This study demonstrates the effectiveness of combining classical image feature extraction techniques with neural network–based classifiers for pattern recognition applications.

Q4

Project Objective

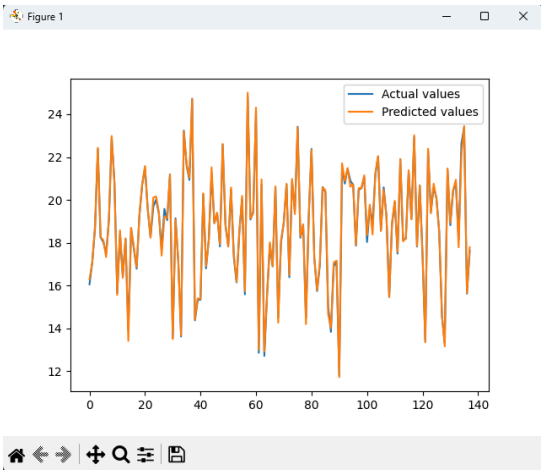
The objective of this project is to develop and deploy a machine learning–based temperature prediction system on an embedded platform. A Multi-Layer Perceptron (MLP) model was designed to predict future room temperature values using a fixed number of past temperature measurements as input. The trained model was converted and executed on an embedded system using the Mbed environment.

Model Training (Python / TensorFlow)

The model was trained in a Python environment using TensorFlow and Keras. Temperature data were loaded and processed in `main.py`. To model temporal behavior, a sliding window approach was applied, where a predefined number of previous temperature values were used as input features to predict the next temperature value.

The MLP model consists of an input layer corresponding to the number of previous temperature samples, one or more hidden layers with ReLU activation functions, and a single-neuron output layer for regression. The model was trained using the Adam optimizer and mean absolute error as the loss function. After training, the model was evaluated on a held-out test set to verify prediction performance.

Once training was completed, the Keras model was converted to TensorFlow Lite format and then exported as a C array to enable deployment on the embedded platform.



Feature Processing

Additional preprocessing and signal-related functionality was organized in `mfcc_func.py`. Although primarily designed for signal feature extraction, this file provided reusable utility functions and demonstrated a modular approach to separating data processing logic from model training. This structure improved code readability and maintainability.

Mbed Integration

The embedded implementation was carried out on the STM32 DISCO-F746NG development board, which features an ARM Cortex-M7 processor operating at 216 MHz. The converted TensorFlow Lite model was integrated into an Mbed OS project and executed using TensorFlow Lite Micro.

The model data were included in the project as static C arrays, and a TensorFlow Lite Micro interpreter was configured with a statically allocated tensor arena to meet the memory constraints of the microcontroller. Several integration challenges were addressed, including correct toolchain configuration, memory allocation sizing, and compatibility between the generated model and the embedded runtime. After resolving these issues, the model was successfully compiled and executed on the target hardware.

Experimental Results

The project was successfully built, and the generated binary was deployed to the STM32F746NG development board. Inference was performed using sample temperature sequences, and predicted temperature values were printed via the serial interface. The system demonstrated stable inference behavior and acceptable latency when running on the embedded platform.

Lessons Learned

This project highlighted the challenges involved in deploying machine learning models on resource-constrained embedded systems. Key lessons include the importance of selecting appropriate model complexity, managing memory usage, handling model format conversion correctly, and ensuring compatibility between high-level machine learning frameworks and embedded software environments.

Conclusion

In this project, a machine learning–based temperature prediction system was successfully developed, converted, and deployed on an embedded platform using Mbed OS. The results demonstrate that neural network models trained in Python can be effectively executed on

microcontroller-based systems. This work confirms the feasibility of applying machine learning techniques to embedded sensing and prediction applications under practical resource constraints.

BONUS

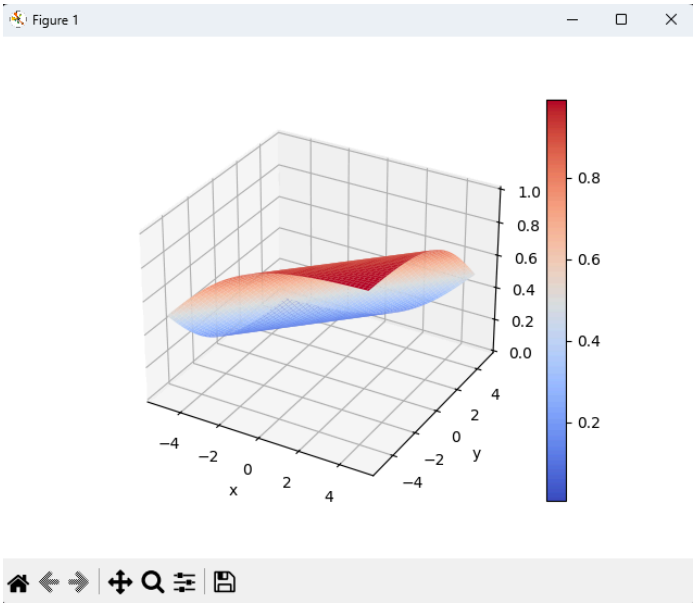
Project Objective

The objective of this project is to develop a temperature prediction system using a Multi-Layer Perceptron (MLP) neural network and deploy it on an embedded platform. The system predicts future temperature values based on a fixed number of past temperature measurements. The trained model was converted and executed on an embedded system using the Mbed environment.

Model Training (Python / TensorFlow)

The model was trained in a Python environment using TensorFlow and Keras. Temperature data were loaded and processed in `main.py`. A sliding window approach was applied to the dataset, where consecutive temperature samples were grouped to form input vectors and the following temperature value was used as the prediction target.

The MLP model consists of an input layer corresponding to the selected number of previous temperature values, hidden layers with ReLU activation functions, and a single output neuron for regression. The model was trained using the Adam optimizer and mean absolute error loss function. After training, the model was evaluated on a test dataset to validate its prediction performance. The trained Keras model was then converted to TensorFlow Lite format and exported as a C array for embedded deployment.



Mbed Integration

The embedded implementation was performed on the STM32 DISCO-F746NG development board, which is based on an ARM Cortex-M7 processor operating at 216 MHz. The converted TensorFlow Lite model was integrated into an Mbed OS project using the TensorFlow Lite Micro framework.

The model was included in the project as static C source and header files. A TensorFlow Lite Micro interpreter was configured with a statically allocated tensor arena to ensure compatibility with the memory constraints of the embedded system. During integration, issues related to library compatibility and memory allocation were addressed. After resolving these issues, the project was successfully compiled and executed on the target hardware.

Experimental Results

The project was successfully built, and the generated binary was deployed to the STM32F746NG development board. Inference was performed using test temperature input sequences, and predicted temperature values were output via the serial interface. The system demonstrated stable inference execution and acceptable performance on the embedded platform.

Lessons Learned

This project highlighted the challenges of deploying machine learning models on resource-constrained embedded systems. Key lessons include the importance of appropriate model sizing, careful memory management, and correct conversion of trained models for embedded execution. The project also emphasized the need for proper integration between machine learning frameworks and embedded software toolchains.

Conclusion

In this project, a machine learning–based temperature prediction system was successfully developed and deployed on an embedded platform using Mbed OS. The results demonstrate that neural network models trained in Python can be effectively executed on

microcontroller-based systems. This confirms the feasibility of applying machine learning techniques to embedded prediction and monitoring applications.