**Hardware Setup and Implementation steps:**To successfully run code on the Edge TPU Corel Dev Board, you need to follow several steps. Here’s a comprehensive guide to get your code up and running:

**1. Set Up and Boot the Board**

* **Gather requirements**:
  + Ensure you have a host computer running Linux, Mac, or Windows 10.
  + Install Python 3 on your host computer.
  + Prepare a microSD card with at least 8 GB capacity.
  + A USB-C power supply (2-3 A / 5 V) and USB-C to USB-A cable.
  + Wi-Fi or Ethernet connection.
* **Flash the board**:
  + Download and unzip the SD card image (enterprise-eagle-flashcard-20211117215217.zip).
  + Use balenaEtcher to flash the image to the microSD card.
  + Set the boot mode switches to **SD card** to boot from the microSD card.
  + Power the board and let it flash the system image to the eMMC memory. The process takes 5-10 minutes.
  + Once done, change the boot mode switches to **eMMC** mode and boot up the board.
* **Access the Dev Board’s shell**:
  + Install the **Mendel Development Tool (MDT)** on your host computer:
  + python3 -m pip install --user mendel-development-tool
  + If necessary, add ~/.local/bin to your PATH:
  + echo 'export PATH="$PATH:$HOME/.local/bin"' >> ~/.bash\_profile
  + source ~/.bash\_profile
  + **Windows users**: Set up an alias for MDT in Git Bash:
  + echo "alias mdt='winpty mdt'" >> ~/.bash\_profile
  + source ~/.bash\_profile
  + Connect the Dev Board to your computer using a USB-C cable.
  + Run mdt devices to ensure MDT can detect the board.
  + If it’s ready, you should see the board's hostname and IP address, such as:
  + orange-horse (192.168.100.2)
  + Now, run mdt shell to access the board’s shell.

**2. Connect the Board to the Internet**

* You need the board online for software updates, model downloads, etc.
* Use either Wi-Fi or Ethernet for internet access.
  + For Wi-Fi, use nmtui to select and activate a network.
  + Alternatively, connect to Wi-Fi using the command:
  + nmcli dev wifi connect <NETWORK\_NAME> password <PASSWORD> ifname wlan0

**3. Prepare the Environment**

* **Update the board’s software**:
  + Run the following commands to ensure your board is up to date:
  + sudo apt-get update
  + sudo apt-get dist-upgrade

**4. Transfer Files to the Board**

* **Using mdt push**: To transfer files to the board, use mdt push. Here’s an example of how to push files to the board:
* mdt push <local\_file\_path> <remote\_file\_path>

For example, to push a Python script:

mdt push ./your\_script.py /home/board\_user/

This will copy the file from your local machine to the Dev Board.

**5. Open Git Bash and MDT Shell**

* **Git Bash** (on Windows):
  + Git Bash allows you to run shell commands with a Unix-like environment.
  + You can open it by searching for "Git Bash" in the Start Menu after you’ve installed Git for Windows.
* **MDT Shell**:
  + Use mdt shell to open the shell of the Dev Board, allowing you to run commands directly on the board.

**6. Run Code on Edge TPU**

* Once you have your files on the board, you can start running the code.
* Make sure any dependencies (like scikit-learn, numpy, etc.) are installed on the board.
* sudo apt-get install python3-numpy python3-sklearn
* Once everything is set up, navigate to your file directory on the Dev Board:
* cd /home/board\_user/
* Run your Python script:
* python3 your\_script.py
* The Edge TPU will accelerate the inference process, improving the efficiency of your model.

**7. Access the Board’s Files (Optional)**

* **Git Access**: If you want to manage files using Git, you can clone repositories directly onto the Dev Board by running:
* git clone <repository\_url>

**8. References and Documentation**

* Official **Coral Dev Board Documentation**:
  + Coral Dev Board Get Started Guide
  + <https://coral.ai/docs/dev-board/get-started/#update-mendel>
* **Mendel Development Tool Documentation**:
  + MDT Documentation

**Files attached :  
1. Hardware Implementation.zip**

It contains the code that we ran on edge TPU board. It also contains train data, labels, test data and test labels.

**1. All\_5\_models.zip**

This folder contains the **core ML experiment results** for the 5 models (RandomForest, XGBoost, LightGBM, KNN, DecisionTree).  
It has both **CSV results** and **plots** (for binary & multiclass, 70/30 and 80/20 splits).

* **CSV Results**
  + combined\_results\_all\_models.csv → Consolidated accuracy/metrics of all models.
  + kfold\_summary\_best\_models.csv → Cross-validation summary with best models.
  + models\_holdout\_results\_80\_70.csv → Holdout validation results for 70/30 & 80/20 splits.
* **Plots per Model & Setting**
  + Each model has **binary** and **multiclass** results shown in:
    - \*\_counts.png → Raw counts of predictions (confusion-matrix like).
    - \*\_percent.png → Percent accuracy per class.
    - Example:
      * RandomForest\_binary\_70\_30\_counts.png → RandomForest performance on binary dataset (70/30 split, counts).
      * LightGBM\_multiclass\_80\_20\_percent.png → LightGBM accuracy percentages for multiclass (80/20 split).

This is the main **experimental results package** for binary & multiclass robustness evaluation of the 5 models.

## ****2. crytpohardening.zip****

This folder is about **cryptographic performance analysis**.

* Performance Results.docx → Report/document summarizing encryption overhead analysis.
* 1.png, 2.png, 3.png, 4.png → Plots showing latency, decryption time, overhead % for encryption in ICS/IoT.

Complements the robustness study with **encryption hardening results**, showing security overhead and real-time feasibility.

## ****3. TCN.zip****

This folder has the **Temporal Convolutional Network (TCN) experiments**.

* **CSV Results**
  + classical\_results\_binary.csv → Classical model results for binary classification (baseline vs TCN).
  + classical\_results\_multiclass.csv → Classical model results for multiclass classification.
* **Plots**
  + tcn\_binary\_counts.png, tcn\_binary\_percent.png → Binary results for TCN.
  + tcn\_multiclass\_counts.png, tcn\_multiclass\_percent.png → Multiclass results for TCN.
  + TCN\_binary\_learning\_curves.png → Training vs validation curves for binary TCN.
  + TCN\_multiclass\_learning\_curves.png → Training vs validation curves for multiclass TCN.
* **Report**
  + TCN.docx → Document summarizing TCN results, comparisons, and findings.

Provides the **deep learning baseline (TCN)** for binary and multiclass, compared against the classical ML models.

**Binary\_robust\_gaussian\_all\_models.png** & **Multiclass\_robust\_gaussian\_all\_models.png** → Visual summary of model robustness under Gaussian noise.