AI Hardware Final Report

Nathan Hersel, Owen Singley, Josiah Suwargo

Engineering School - University of Virginia

ECE 4501

Dr. Mircea Stan

12/10/2024

Objectives:

For this project, we worked to complete multiple objectives. We wanted to perform image classification on the Google Coral Micro TPU using the MobileNetV1, MobileNetV2, and InceptionV2 models. We then wanted to perform image classification on Google Colab TPU using the MobileNetV1, MobileNetV2, and InceptionV4 models. The next objective was to measure the execution time for image classification on both platforms using the three different models on each platform. The Google Coral measures the time output in ticks and those have to be converted to seconds. Google Colab measures the time output in seconds per iteration and that has to be converted to seconds so that it can be compared to the Google Coral. The final objective was to determine whether having access to a physical TPU gives enough advantages to make it worth the cost compared to cloud ML applications. This will mainly be done by comparing the time on both platforms and also looking into the complications that come with both platforms.

Expected Results:

We expect that there will be very little variation in the execution time between the two platforms. Since both the Google Coral hardware and the cloud AI are running the same models and the cloud is running on top-of-the-line GPUs, it is expected that the time to be about the same on both platforms. If there is variation between the two platforms, we would expect the cloud-based platform to execute quicker because the Coral TPU hardware is limited to a 4 TOPS peak performance TPU. However, we expect that having access to the physical hardware will provide improved customizability compared with the cloud platform and this will allow for more efficient performance on the Coral Edge TPU.

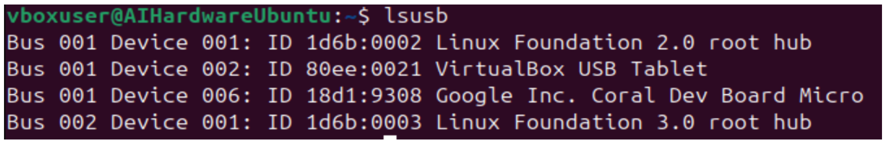
How Does it Work:

For the Google Coral, to get the board running, we first plug the board into a laptop using a USB-C cable. Next, we log into a Linux Virtual Machine because Linux provides the capability to flash our program to the Google Coral TPU Dev Board Micro. Then the board can be flashed and the output can then be examined on the serial console. For Google Colab, the program written is simply executed and the results are printed to the output.

Tutorial:

Step 1: Power the board using a USB-C cable connected to a laptop running Linux or MacOS with Python3 installed.

Step 2: Verify that the Coral Micro board is connected to the machine using the lsusb command.



```
vboxuser@AIHardwareUbuntu:~$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 006: ID 18d1:9308 Google Inc. Coral Dev Board Micro
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
```

Figure 1. Linux Terminal with “lsusb” Command

Step 3: Set up a directory for FreeRTOS development with command: git clone --recurse-submodules -j8 https://github.com/google-coral/coralmicro

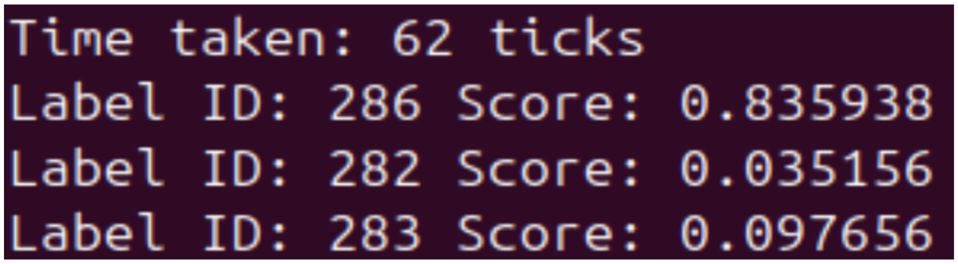
Step 4: Navigate to the coralmicro directory you just created and run: bash setup.sh and bash build.sh

Step 5: Now flash the board using the command: python3 scripts/flashtool.py -e <EXAMPLE_NAME>

Step 6: Once the board is flashed, view the output of the program. This will be different depending on the program flashed to the board. Some may require launching a Python application, while others make use of the serial console.

Step 7 (optional): Our program made use of the serial console so we will show how to access the console here. First, run the command "ls /dev/ttyACM*" to detect to which line the board is connected.

Step 8 (optional): Now based on the result of the previous command, run the command "screen /dev/ttyACM0 115200" (ttyACM0 was the line our board connected to). This command will connect you to the Coral Micro serial console, where you will be able to view the printed results.



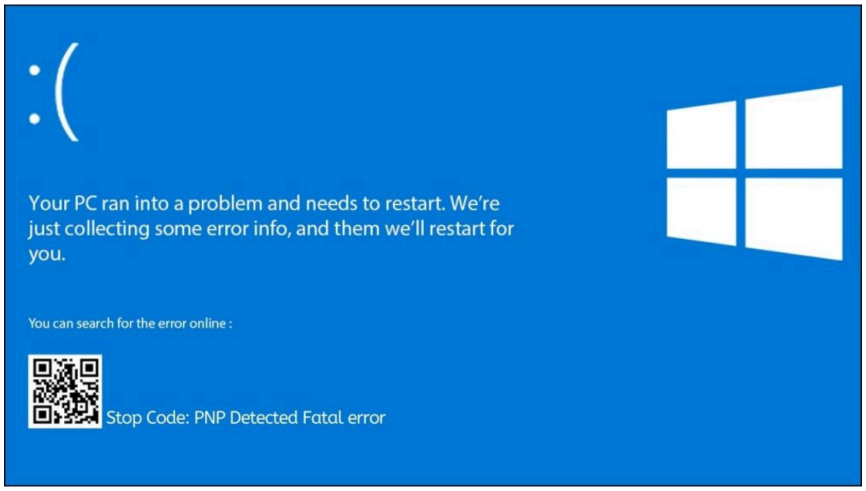
```
Time taken: 62 ticks
Label ID: 286 Score: 0.835938
Label ID: 282 Score: 0.035156
Label ID: 283 Score: 0.097656
```

Figure 2. Example Output from Serial Console

Additionally, if you want to alter program code to fit your needs, all you need to do is use a code editor, make your changes, and then rebuild the development environment and reflash the board. If you are keeping the same machine learning model for the program, you can use the --nodata flag in the python flashtool command to only flash program code and skip over the .tflite files. This could save you lots of time during the debugging process.

Known Issues and How to Fix:

During this project, we ran into a lot of miscellaneous issues that slowed us down greatly. The first issue we ran into was using a USB-C cable that was not capable of data transport and thus only powered the board. This was solved by getting a new cable. Another major issue that we encountered was the serial console printing inconsistently. Sometimes the information comes up as expected, but other times nothing appears and the serial console must be restarted. We expect that this is due to our Linux machine being a VirtualBox VM on a Windows laptop. Sometimes plugging the board into the laptop caused the laptop to blue screen with a PNP_FATAL_ERROR. This is shown in figure 3 below. We are still completely unsure why this happens. It could be because the TPU is in use and it causes a power spike occasionally when powering up. Sometimes the board is not recognized by the computer when it is plugged in. This is usually an issue that comes up if an error occurs in flashing the board. To fix this, we put the board into Setup Downloader mode, plug it back into the computer, and reflash.


:(

Your PC ran into a problem and needs to restart. We're just collecting some error info, and them we'll restart for you.

You can search for the error online :

Stop Code: PNP Detected Fatal error

Figure 3. Blue Screen Encountered Randomly When Plugging in Board

Results and Conclusion:

For the Google Coral TPU, the MobileNetV1 model performed image classification in 62 milliseconds, the MobileNetV2 model finished in 61 milliseconds, and the InceptionV2 model finished in 155 milliseconds. Surprisingly, the MobileNet models were much faster than the Inception V2 model. This is likely because the program was specifically designed for this model. We had to use InceptionV2 on the coral micro because V4 is not supported for the micro board. As for the Google Colab TPU, the MobileNetV1 was not available from the available libraries as we suspect it might be too old of a model to keep, but we still managed to get results for the MobileNetV2 and InceptionV4, with their image classification time being 35 milliseconds and 36 milliseconds respectively.

	MobileNetV1	MobileNetV2	InceptionV4
Coral Micro TPU	62 ms	61 ms	155 ms
Colab Cloud TPU	N/A	35 ms	36 ms
Notes	MobileNetV1 was not working on Colab		InceptionV4 was not compatible with Coral so InceptionV2 was used instead

Table 1. Results from Experimentation with Coral and Colab Image Classification

Based on the data we've acquired, we've determined that the project was a success, the time threshold was within expectation as the Google Coral TPU will always be slower than the Colab TPU as it is hardware compared to software. From this comparison, we learned that the Google Coral Micro Dev

Board is capable of numerous things that we are yet to try, despite it having a harder learning curve when compared to Google Colab, we think that the payoff from the projects done with the Dev Board will be much more rewarding.