AI Systems Lab Notebook

Jules Udahemuka
judahemu@andrew.cmu.edu

LAB #2

Jan 27, 2025

Objective: Setting up the assignment RPi and Arduino for the assignment.

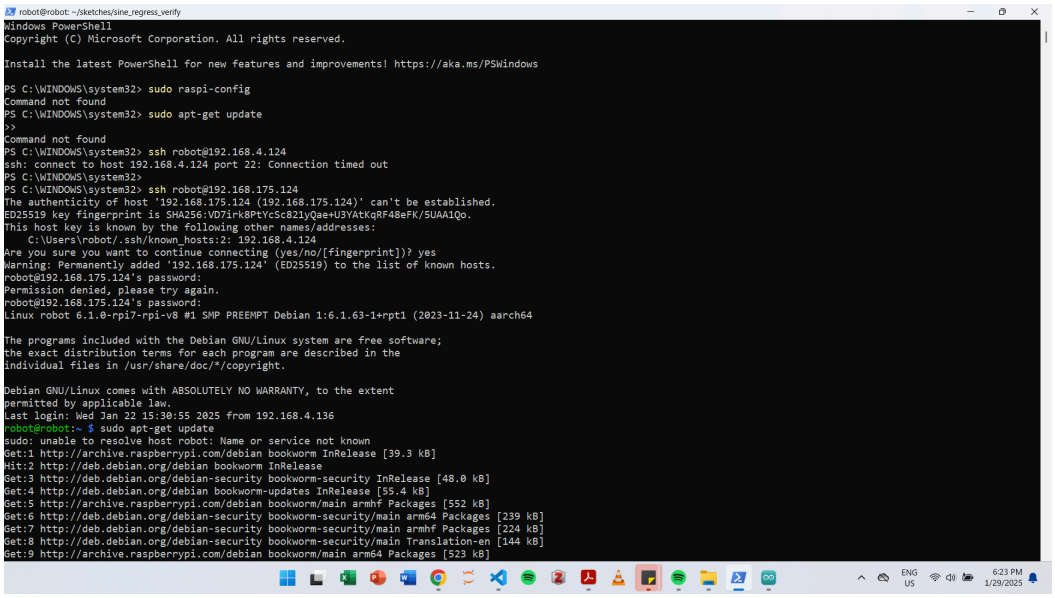
1. **Initial File Transfer Issue**
    a. **Problem**: The **sine_regress_verify.tar** file needed to be transferred from laptop to Raspberry Pi. It took me a while to figure it out how to actually do this. Most of the attemp failed until I came across an article which described how to actually do it using SCP
    b. **Solution**: Used SCP (Secure Copy Protocol) to transfer the file to the correct directory

2. **TensorFlow Lite Compilation Issues**
    a. **Problem**: First compilation was taking very long and showing precompiled library errors
    b. **Solution**: Modified the library.properties file to disable precompiled libraries by setting (This allowed the sketch to compile successfully, though taking longer):

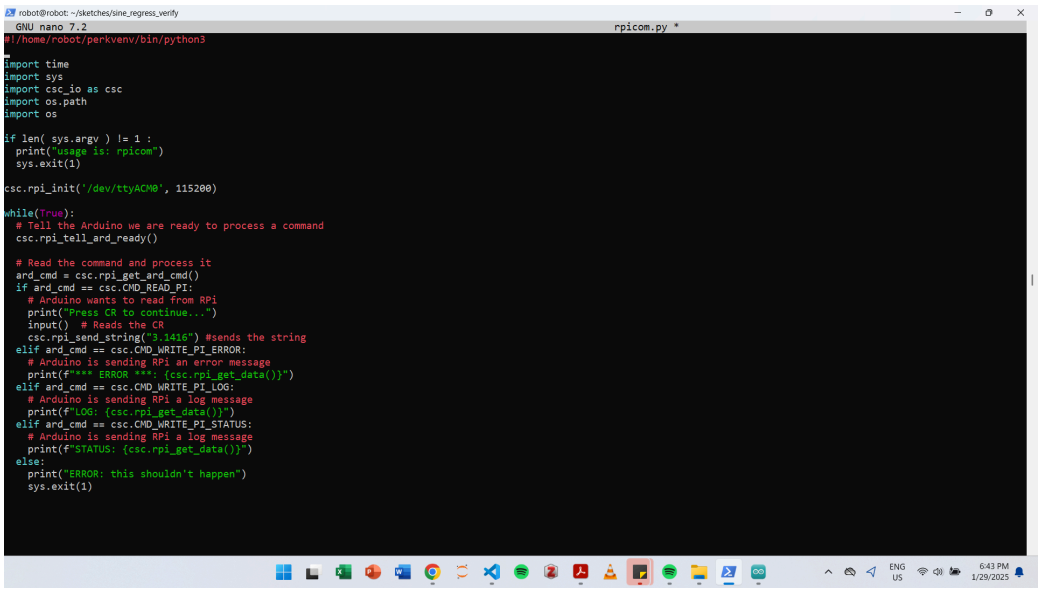
```
Unset
precompiled=false
dot_a_linkage=false
```

3. **Python Script (rpicom.py) Execution Issues**
    a. **Problem**: Script had incorrect line pointing to wrong Python path as I was using the one from the starter code. So it gave a lot of errors and took few minutes to figure it out.
    b. **Solution:** Modified the first line to point to the correct Python interpreter path:

```
Unset
#!/home/robot/perkvenv/bin/python3
```



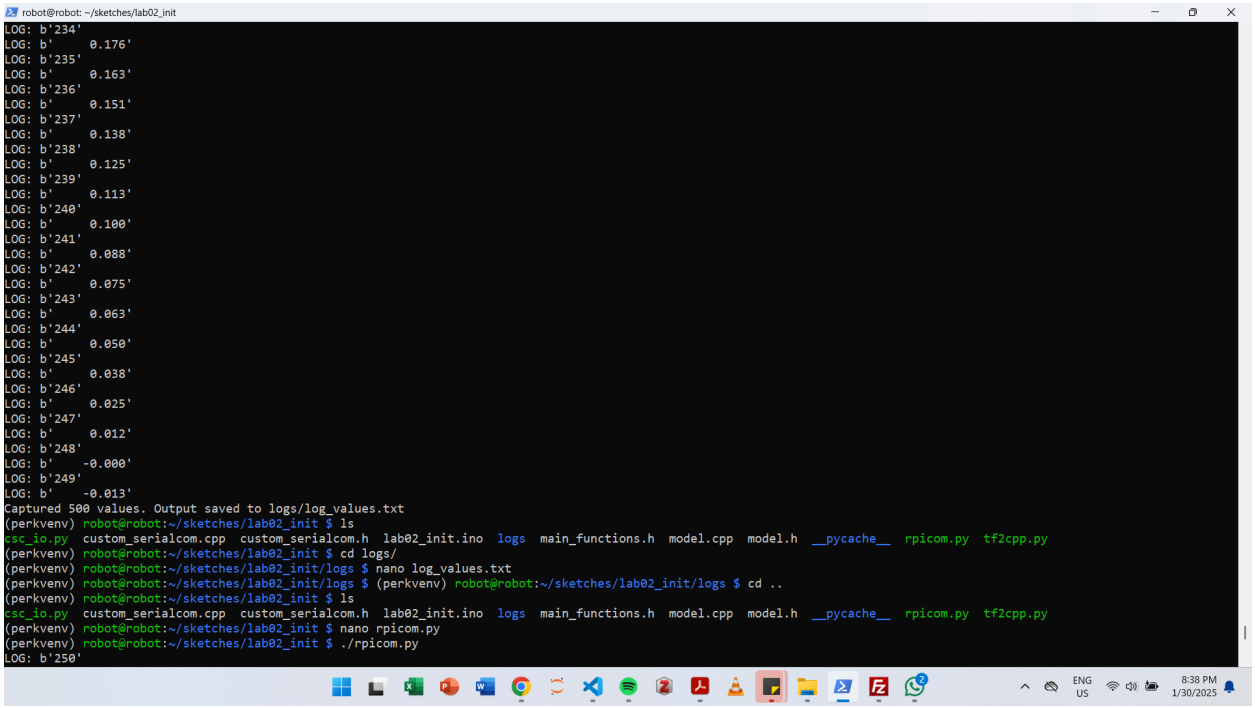
4. **Arduino Upload Issues**
   a. **Problem**: "No device found on ttyACM0" errors
   b. Attempted Solutions:
      i. Checking USB connection and device detection: Initially, I thought the device was not connected, which indeed was true. Arduino wasn't connected successfully via USB. After doing this, the command continues to show the same issues as before. So, I tried the next step.
      ii. Using verbose mode for uploading: This didn't resolve the issue to as the device continue to return empty when I run the ttyACMO command.
      iii. I went back to the Lab 1 to see how I set it and I found that there was a command that I forget to run to create the Arduino environment. After running the command, the issue resolved successfully.



Jan 29, 2025

Objective: Finalize the coding part with a rough draft of the Lab Notebook

1. Initial changes to rpicom.py went successfully, as it was able to print and save 500 logs as it was described in the report.



2. **Initial Data Format Issue for X and Y changes**
    a. **Problem**: The output showed pairs of values, but the y-values were sequential numbers (3, 4, 5...) instead of actual model predictions
    b. **Diagnosis**: Through debug logging, I discovered that the inference count was being sent instead of the model's output values
    c. **Solution**: Modified the HandleOutput_ml function to properly format and send both x and y

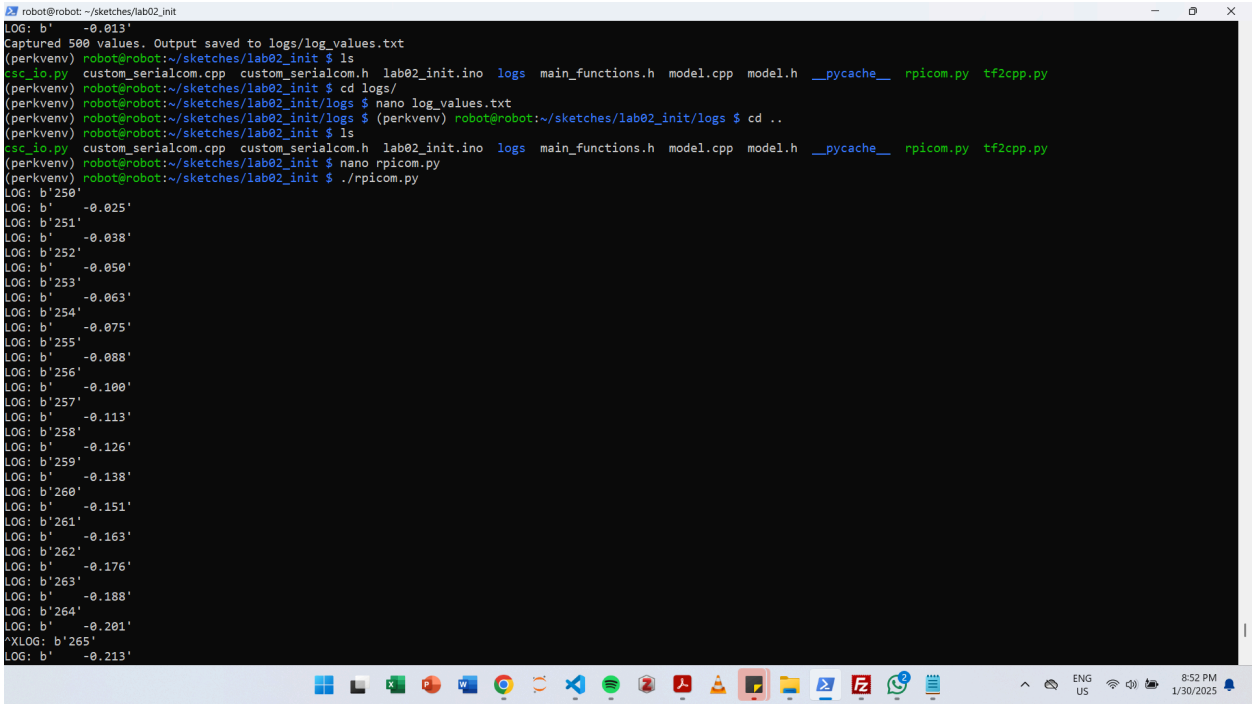
Unset

```
void HandleOutput_ml(double x, double y) {
    dtostrf(x, 10, 3, varbuf);
    csc_write_data(CSC_CMD_WRITE_PI_LOG, (byte*)varbuf,
strlen(varbuf));
    dtostrf(y, 10, 3, varbuf);
    csc_write_data(CSC_CMD_WRITE_PI_LOG, (byte*)varbuf,
strlen(varbuf));
}
```

3. **Communication Protocol Confusion**
    a. **Problem**: Data wasn't being properly paired in the Python script
    b. **Diagnosis**: Added extensive debugging to track each message and its format which apparently didn’t resolve the issue until I checked the code again and realized that I was forgetting to add the proper venv at the beginning of my code (happened twice yesterday and today).
    c. **Solution**: Created a debug logging system in rpicom.py to track message flow:

Unset

```
def debug_print(message):
    print(message)
    debug_file.write(f"{message}\n")
    debug_file.flush()
```

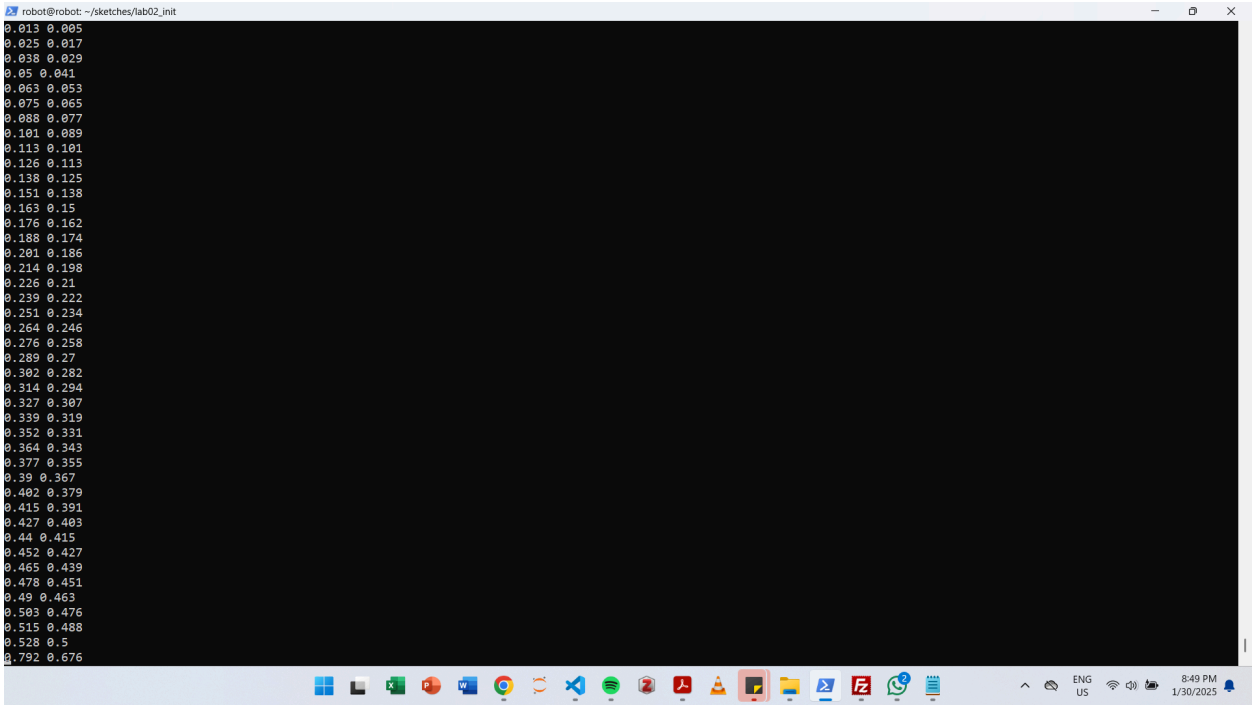


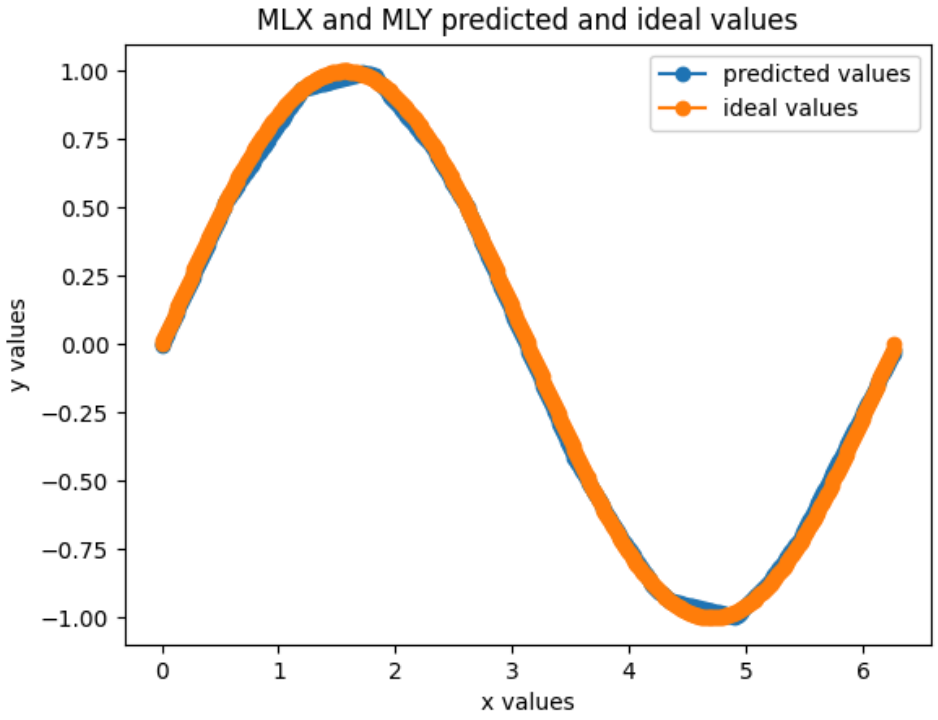
4. **Data Synchronization Issue**
    a. **Problem**: X and Y values weren't being properly paired in the log file
    b. **Diagnosis**: Through debug output, saw that messages were being received but not properly paired
    c. **Solution**: Implemented a state machine approach in Python to track and pair values, but this didn’t help to resolve the issue:

Unset

```
if is_value:
    current_x = value
    is_value = False
else:
    log_file.write(f"{current_x},{value}\n")
    captured_pairs += 1
    is_value = True
```

After all the diagnoses, I was able to resolve the issue and output the file file below:



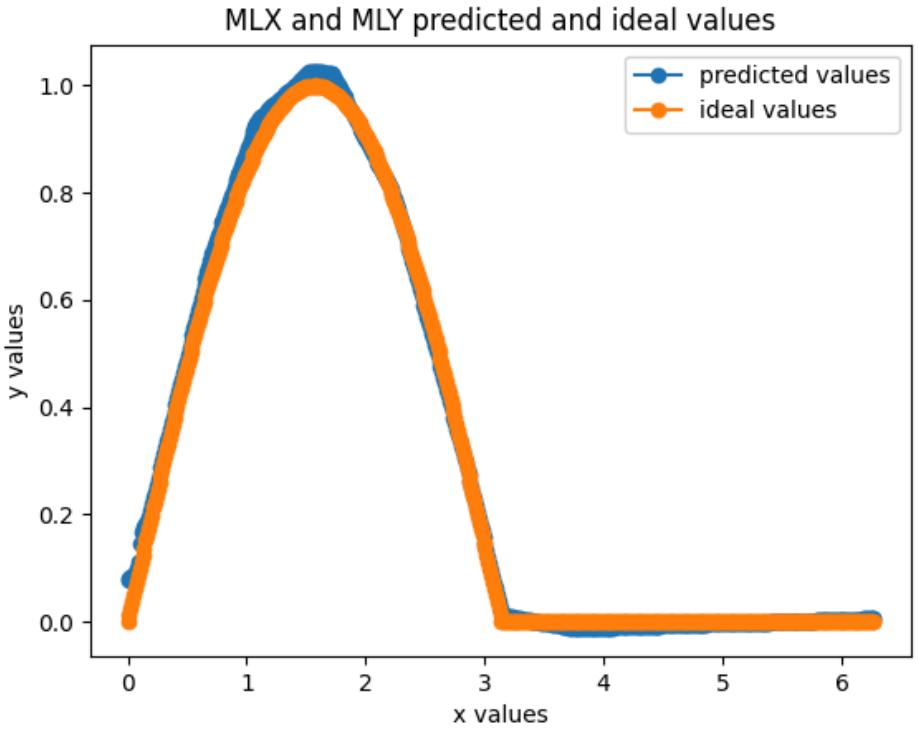


5. **Change the seed and record:**

- ➔ Through experimentation with different random seed values, I observed notable variations in model performance. I tested three different seeds to initialize the model weights: 12601, 12345, and 34457. The evaluation metrics used were test loss (mean squared error) and mean absolute error (MAE), which provide complementary insights into model performance.
- ➔ The seed value 12601 yielded relatively modest performance with a test loss of 0.0149 and MAE of 0.0975. When we changed to seed 12345, we saw an improvement in both metrics, with the test loss decreasing to 0.0114 and MAE improving to 0.0901. However, the best performance was achieved with seed 34457, which produced the lowest test loss of 0.0107 and MAE of 0.0828. This represents approximately a 15% improvement in MAE compared to our initial seed.
- ➔ The results showed the impact of weight initialization on model performance. The significant variation in performance across different seeds demonstrates the importance of trying multiple initializations when training neural networks, even for relatively simple regression tasks (I read a lot about this topic as it was fascinating to see how different initialization impact learning in the model, one article that stood out is this one taking about random initialization but it talked about other initialization technics: https://pub.aimind.so/random-initialization-vanishing-and-exploding-gradients-c10ba7a52728?gi=4b5b7e8be9d2).
- ➔ *The seed 34457 likely provided an initial weight distribution that allowed the optimizer to find a better local minimum during training, resulting in superior generalization performance on the test set.*

Jan 30, 2025

6. **Half-wave experimentation**: The half-wave rectified sine wave experiments involved several model architecture modifications and hyperparameter tuning. Looking at the graph, you can observe excellent alignment between predicted and ideal values, with the model accurately capturing both the sinusoidal portion and the rectification at zero, indicating successful learning of the non-linear behavior.



My experimentation revealed several interesting findings:

1. Initially, increasing the number of epochs from the default to 500 yielded modest improvements, with test loss decreasing to 0.0102 and MAE to 0.0811. This suggests the model benefited slightly from additional training iterations.
2. Modifying the network architecture by adjusting the number of neurons to 32 in the dense layer with ReLU activation produced a small but positive impact, reducing test loss to 0.0101 and MAE to 0.0808. This indicates that the increased model capacity helped capture the function's complexity marginally better.
3. Further architectural changes, including adding another layer (32 neurons followed by 16 neurons), actually led to slightly worse performance with test loss increasing to 0.0104 and MAE to 0.0812. This suggests that the additional complexity may have been unnecessary for this particular problem.
4. Experimenting with different activation functions, specifically switching from ReLU to tanh, resulted in slightly degraded performance (test loss: 0.0105, MAE: 0.0819). This indicates that ReLU was indeed a more suitable activation function for this task, possibly due to its effectiveness in handling the rectification behavior of the target function.