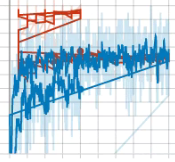
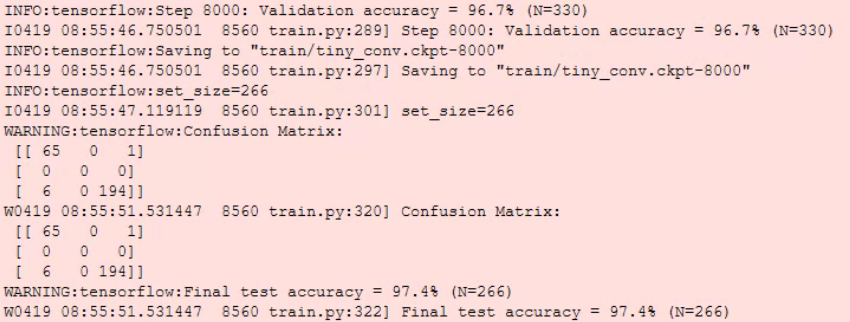
The primary goal for developing a prototype was to be able to run a covid detection algorithm in a small and cheap package. The main goals for the prototype were to detect coughs in a crowded environment, check if those coughs showed that someone had COVID, count the amount of people in the room, and to send the information to a server to calculate the distribution and concentration of positive cases. The prototype also needed to be cheap to allow widespread adoption.

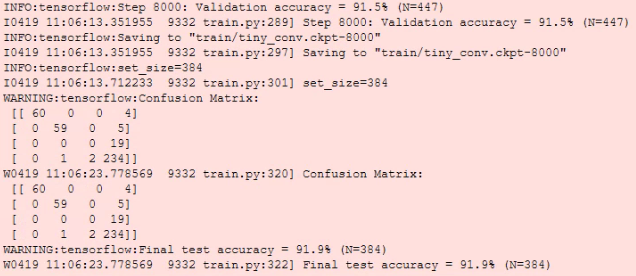
The first design was an ESP-32 with a camera and attached microphone that could run cough detection with tensorflow lite and identify the amount of people in a room with the camera. The first goal of getting cough and covid detection running on the ESP-32 was accomplished by adapting a lightweight speech detection example included with tensorflow. Cough data was easily found on Github, and almost all of the datasets were designed for COVID testing. This made training a model simple. The model was trained using a modified version of the example training code on Google Colab. The original version of the model only reached 80-85% accuracy, and after using a larger dataset and removing silence from the dataset the accuracy moved to above 90%. I also switched to setting up Jupyter locally to run training due to the difficulty of moving data to Google Colab, allowing me to modify the model parameters and dataset faster.



Graph of overlaid accuracy of cough detection and coronavirus testing models during training.



Final iteration of cough detection model.



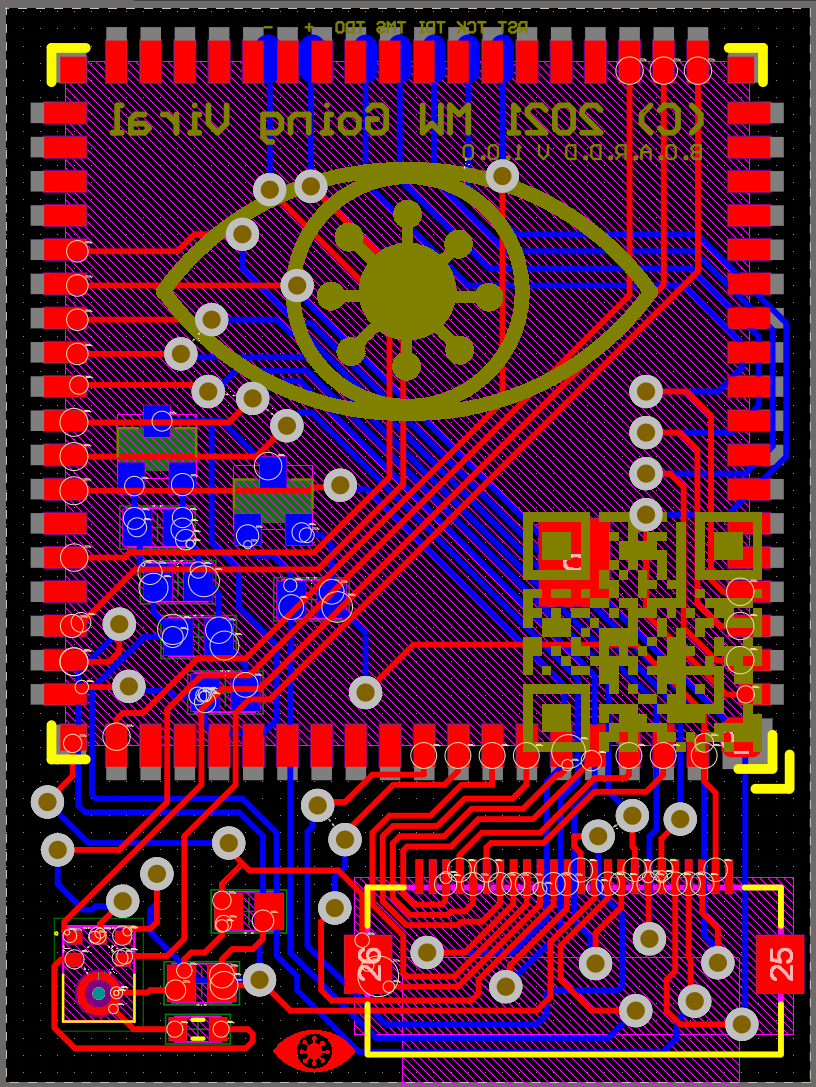
Final iteration of coronavirus detection model.

Although the ESP-32 successfully ran the cough detection model, attempts to convert existing object detection and crowd counting models proved that the ESP-32 was not powerful enough for this application. Most models that used images wanted hardware support for computer vision in the form of CUDA or OpenCV, which the ESP-32 lacked. The only clear solution at the time was to run the person counting on a much more powerful Nvidia Jetson Nano. Even though the setup worked to identify the amount of people in the room and could detect coughs on the ESP-32, the high cost of the Jetson Nano made the prototype unviable for further development. The Jetson Nano ran YOLOv4 object detection, despite my attempts to find a dedicated crowd counting model. The Jetson Nano used an ARM64 cpu, which was unsupported by most dependencies.

Because of the high power of the Jetson, multiple ESP-32 video streams could be processed simultaneously allowing for a lower cost per individual point of monitoring. This setup would work as intended, but was limited to large scale deployments with a high entry cost. I began looking for a lower cost chip similar to the one found in the Intel Neural Compute Stick that would allow for all processing to be ran on one board. After looking through dedicated object detection boards, I found the Sipeed M1 and M1W. The M1W contains a two core RISC-V cpu with a deticiated AI coprocessor allowing it to run YOLOv2 object detection and an integrated ESP8266 for wifi. Adding a small pcb with a microphone and a camera creates an all in one COVID detection system for a price under $15.

Because the M1W was identified late in development, I have not been able to port the cough detection yet. Most of the time spent so far was working on finding the most lightweight models, it should be simple enough to convert the example code to run on the M1W. After removing silence in the COVID dataset, the model can distinguish positivity with ~90% accuracy and identify coughs from background noise with ~97% accuracy. The camera only needs to detect people near the device so the person detection does not need to be accurate.

The custom PCB was created based on the schematic of the Sipeed Maixduino, a board similar to the final goal of the prototype. The board was designed to be as lightweight and cheap as possible, and only contains the processor, camera, and microphone. Due to the long wait times for shipping the SMD parts and PCB from China, the board would not be ready by the final presentation. Despite this, the board is fully functional and I plan on finishing assembly after the competition is over for experience.



Early version of the PCB.

After this pcb design was made, I discovered that the existing cough detection model could not be modified to work with the Sipeed M1. An ESP32 or similar Arduino board would need to be added. I have decided to finish the breadboard prototype before finishing the PCB.

Main accomplishments:

* Found a working audio detection example that supported the ESP32
* Found and formatted COVID training data
* Trained a working COVID testing model
* Verified that the model worked on the ESP32
* Looked into crowd counting models before settling on YOLO object detection
* Found a processor that would support YOLO object detection
* Verified that the processor would work for detecting people
* Created a PCB for a final product
* Created a new name and logo for the prototype

Things that still need would need to be done for demonstration:

* Interface ESP32 and M1 dev boards to verify functionality

Required steps for full prototype completion:

* Finish the PCB with verified wiring
* Create a case to protect the PCB
* Connect ESP32 to the backend to display infection data.

Possible ways to improve accuracy of device:

* Train custom YOLO model to improve person detection
* Create more expensive device with better microphone and camera

Despite the slight benefits these options are time prohibitive and not necessary for an initial prototype.