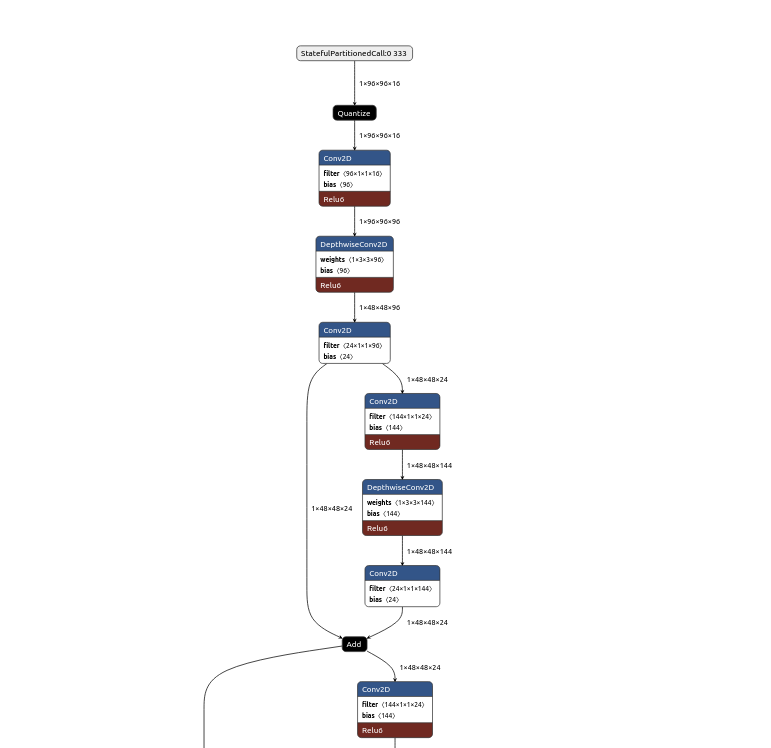
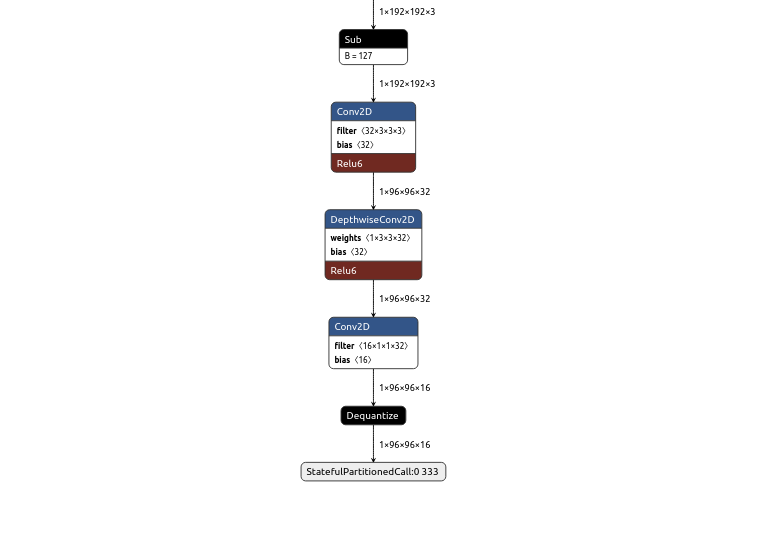
**Assignment #3** – Partitioned, Offloaded DNN Inference

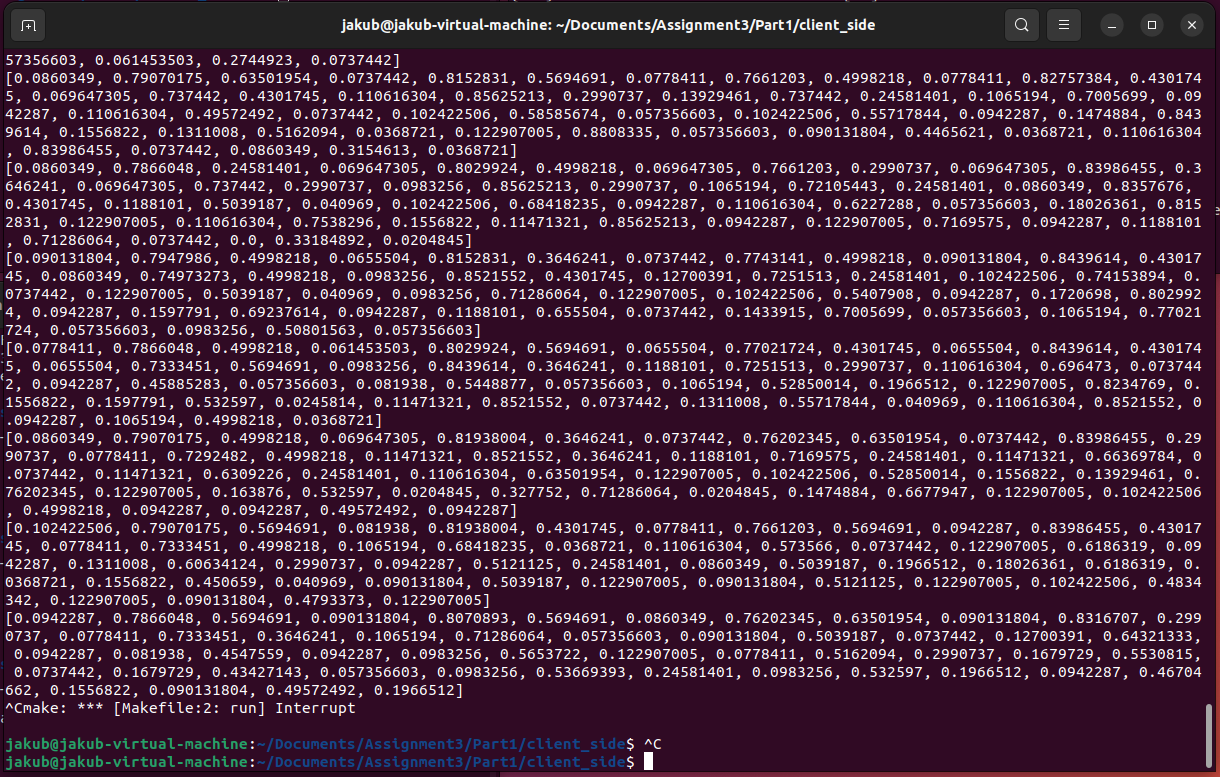
**Part #1** – Partitioned, Offloaded DNN-Based Inference

This part of the assignment was rather simple conceptually, thanks in large part to the Netron app which was very cool to visualize the individual components of the neural net coming together to create the model. In order to do the assignment, I ended up simply using a Python script to convert the model into .json format which I then modified to split it apart. An issue that I ran into while doing this was due to a precision error from within flatc. I addressed this by simply finding all instances of the error (1. / 256 in my case on Node 78), and then simply changing all values to a more precise value. This happened to solve the issue for me. Additionally, I ran into a problem with the data type of the tensors, so I had to convert them into float values and back using the dequantize operator at the end of the local component (left image) and a quantize operator at the beginning of the remote component (right image). After doing this, it was rather simple to get each component to load in the corresponding models and simply use buffers to communicate over the network. I did run into an issue where the output of the OpenCV window solely showed a black screen. I wasn’t able to figure out why it was resulting in a black screen. From the images down below, we see the original display (which can be verified by the path above), and it displays the output along with the 51 float keypoint values that are to be mapped onto the blank canvas. As can be seen in the image above, my **separated** model produces very similar results, however I wasn’t able to figure out how it wasn’t being plotted correctly.

**Local Component/Remote Component**



**Display of Keypoints**



**Original Display**

