Using the color corrector code

Sections

Jetson Nano Startup

The Jetson Nano has a few external peripherals that you must check before turning it on:

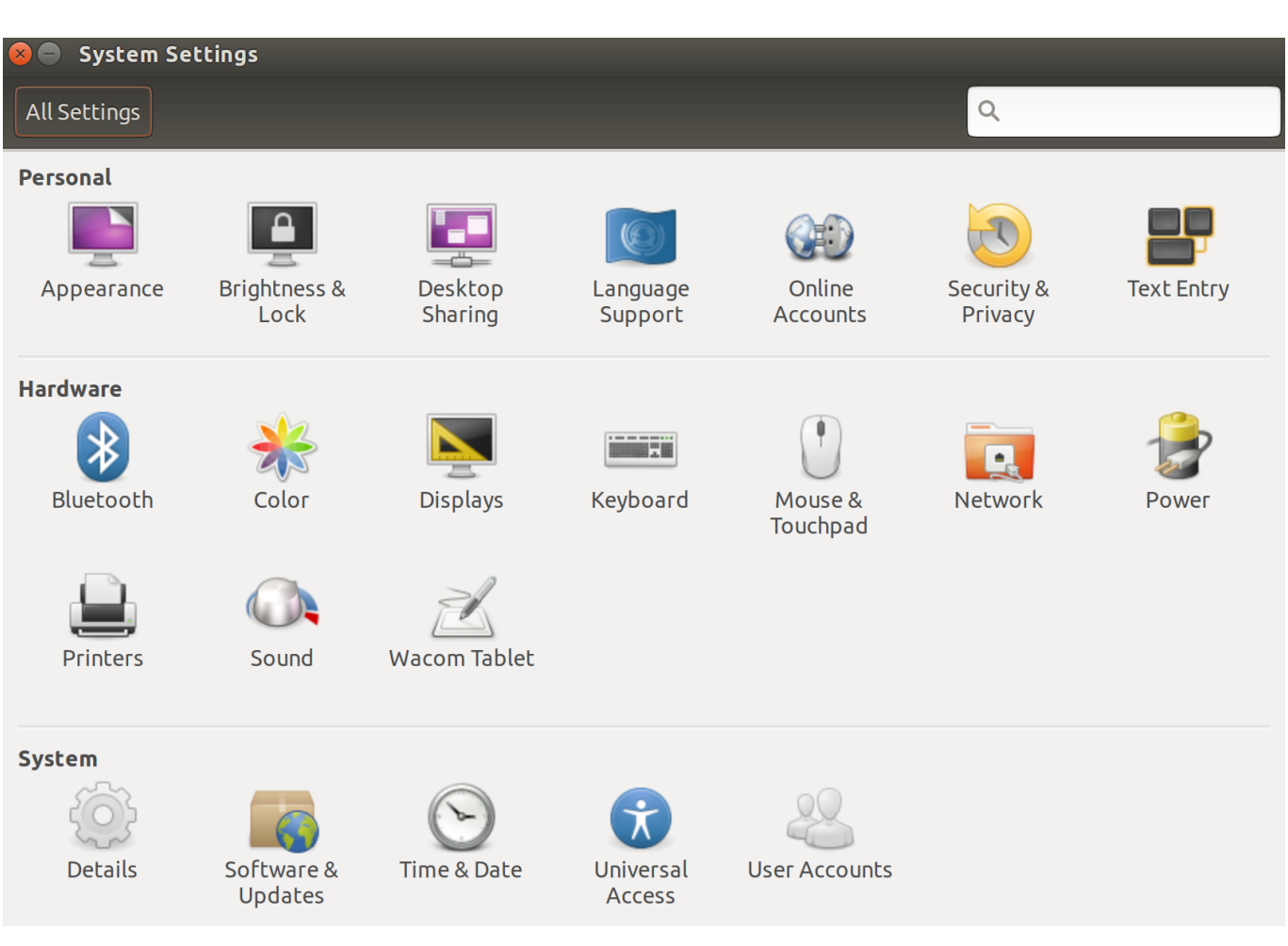
- The prototype's external 500GB Samsung SSD that *has* to be plugged into a USB port. This external drive is for the Nano's swap file.
- The Logitech wireless dongle for the wireless keyboard
- An external display. There are two ports for this, you can either use an HDMI cable or Display Port
- A Webcam if necessary
- The 20A power supply so that the Nano can receive power

To power on the Nano just plug in the powersupply to an wall outlet and it will start up. The start up processes should take from 15-20 seconds and an Nvidia logo will appear as a splashscreen during such.

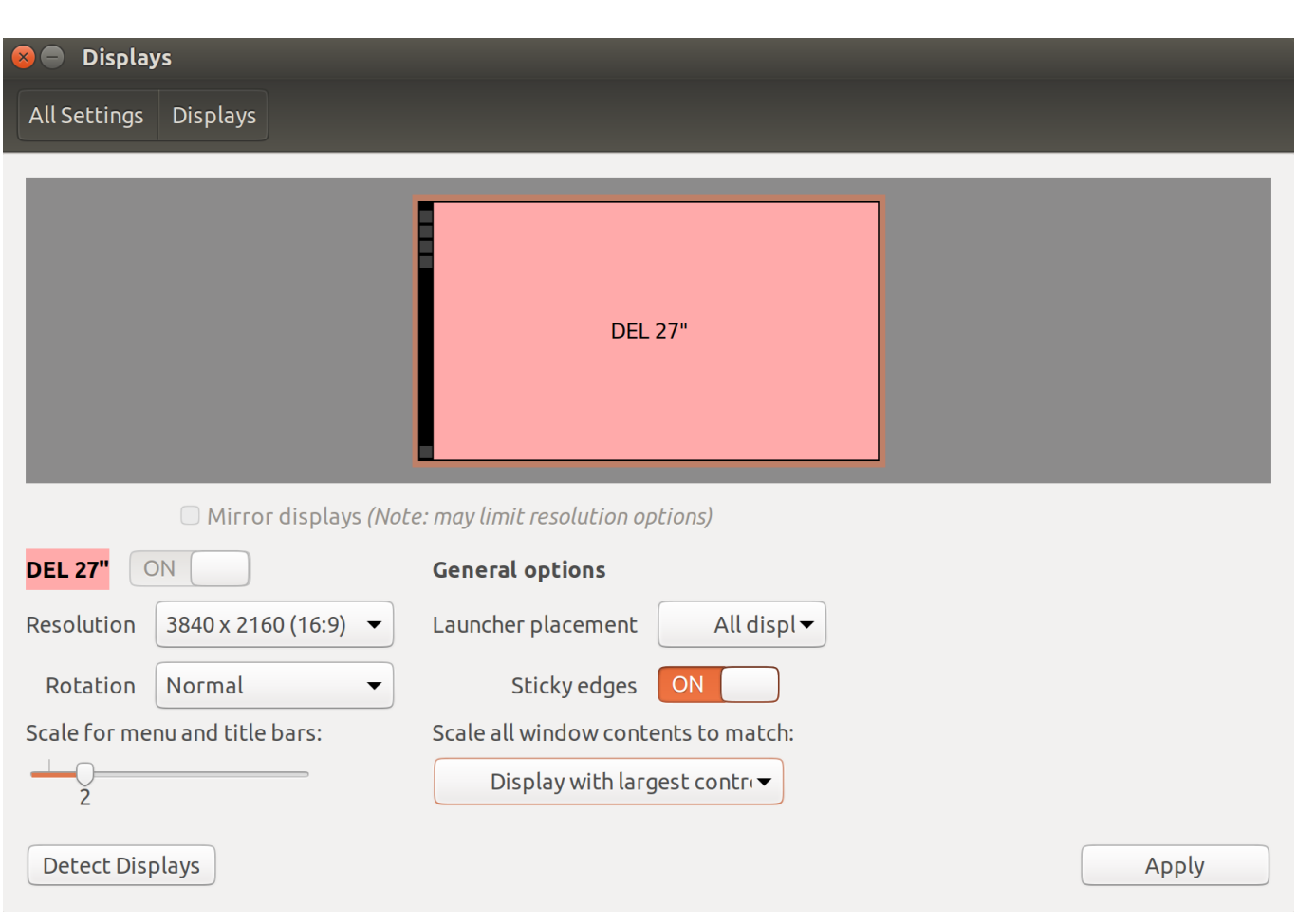
Once the Nano is running, a desktop will be shown like below:

From this, we can accesses the basic computer necessities.

In the case of where the desktop icons are too small, select the **gear icon** in the top right of the screen and select **system settings**. The window shown below should appear:



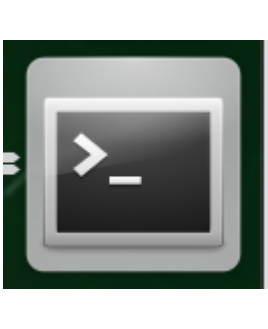
From here select **Displays** under the Hardware section and change the **Scale for menu and title bars** slider until satisfied and click **apply**



If for some reason you need to connect to Wifi while using the Nano, click the **Wifi signal symbol** in the top right and change the appropriate settings.

Another *Mandatory* step we must take is to put the Nano in its maximum usage mode. Normally when you start up the Nano, it is in an eco-power state that varies the the clock speed of the processor depending on the load of the system. The current version of JETPack tends to be hyperactive in changing the clock speed. For this project we want to make sure that the Nano is at maximum usability at all times.

To put the processor at its maximum clock speed, either search for the terminal window shortcut by pressing the **Windows Key** on the Logitech keyboard or select the **Terminal Shortcut** on the **Task Bar** like shown below:



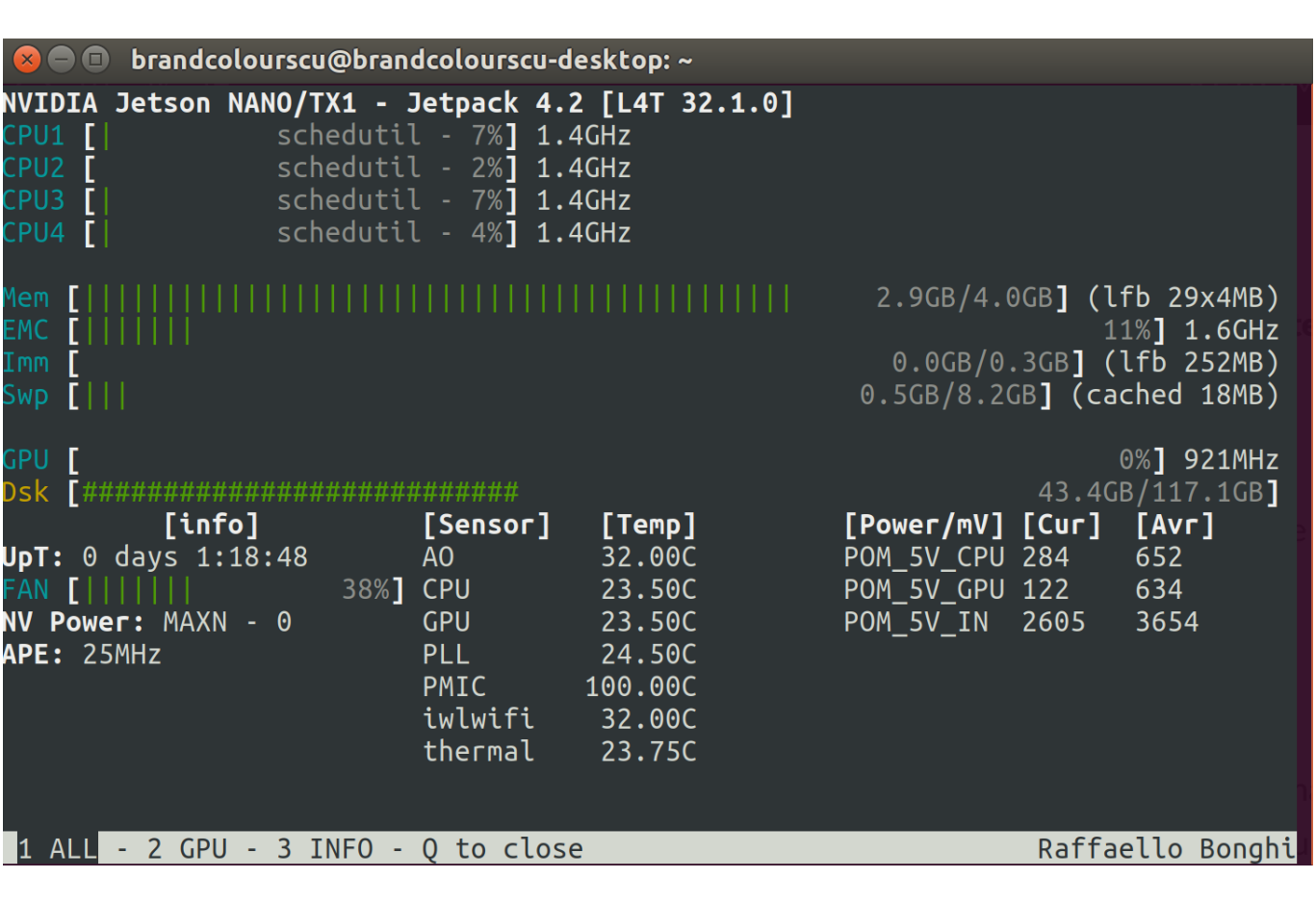
With the Terminal open type the following and press **Enter**:

```
Sudo jetson_clocks
```

To verify that the previous command worked, we can use a program called jtop. To access jtop, type the following in the Terminal:

```
Sudo jtop
```

If done correctly the terminal should now look like the image below and if done correctly the frequency readings for each core should be *1.4GHz*



jtop shows us useful information for looking at the performance of the Nano such as memory usage, power consumption, cpu and gpu load. You can choose to leave this open during the demo or close it by pressing **Q** on the keyboard.

The Nano is now ready for whatever you throw at it!

Navigating to the project directory

The resources for this project are located in a folder named **BrandColorsProj** To view this directory with a file explorer, click the **Files** Shortcut located on the taskbar and select the **BrandColorsProj** folder. This folder contains 3 sub folders named:

- Code
- Data
- Models

What we want to use is the **/BrandColorsProj/Code/** directory. If you have any demo videos you want to run place them in this folder.

Open a Terminal window by either **right clicking** on the **Terminal shortcut** on the taskbar and selecting **New Terminal** or continue with the window you were just using during the jtop setup.

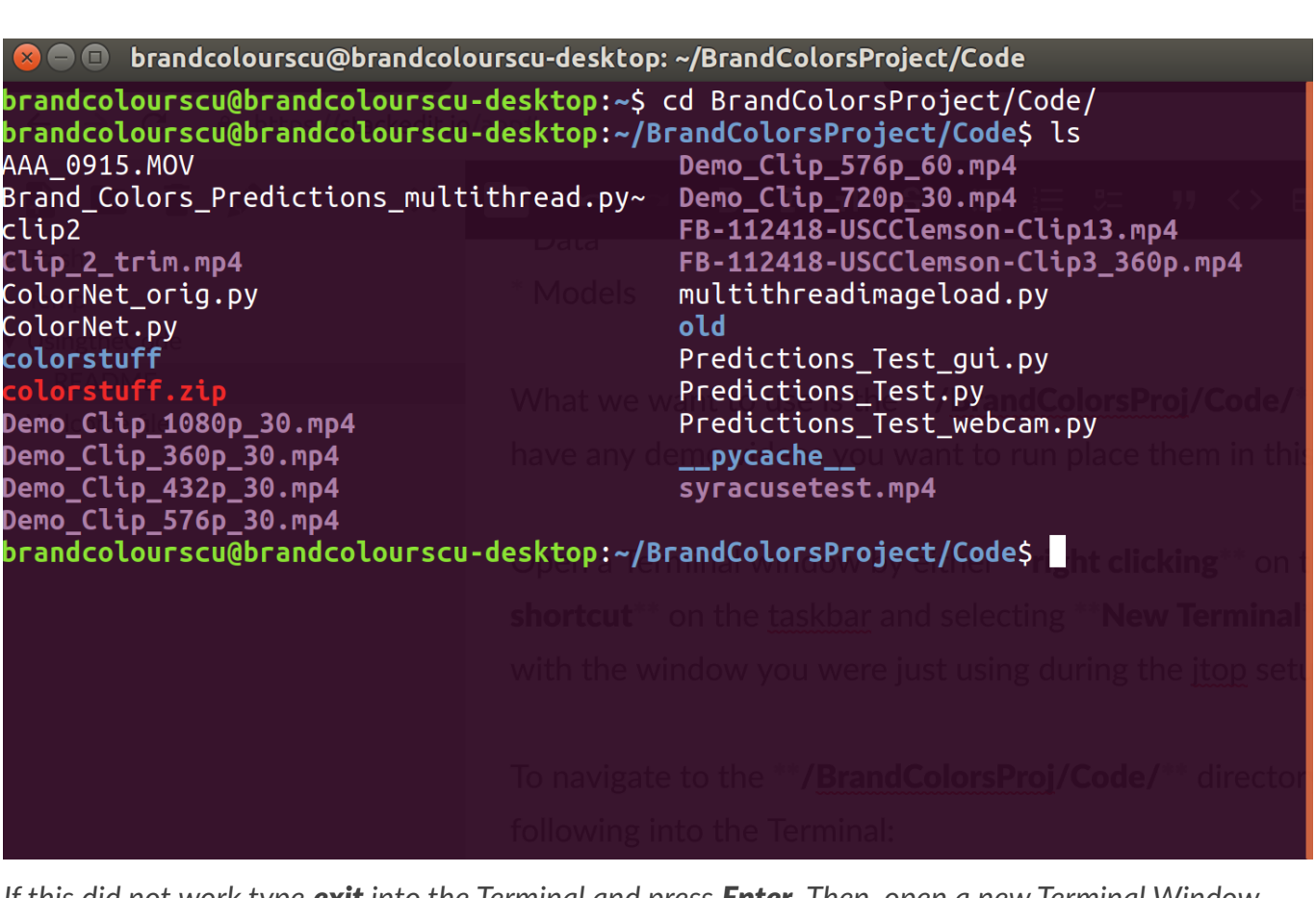
To navigate to the **/BrandColorsProj/Code/** directory, type the following into the Terminal:

```
cd BrandColorsProj/Code/
```

To see if you are in the correct directory enter the following into the Terminal:

```
ls
```

The following should appear:



*If this did not work type **exit** into the Terminal and press **Enter**. Then, open a new Terminal Window and try again.*

Running Different Demos

Now that you have navigated to the correct directory in the Terminal, running the Python scripts is easy. The code folder contains three Python scripts to be used for different demos:

- Basic Predictions Demo (Best FPS)
- GUI Demo with sliders (Smoothest Output)
- Webcam Demo

Each of these demos has arguments you can pass with them to modify the demo:

```
--demo or -d : used to run the demo with the broadcasted vs corrected
--profile or -p : Runs line profiler
--intensity or -i : Scale factor for the correction mask
--cpu or -c : Runs the demo in cpu mode
--help or -h : provides the information above
```

Note: You will most likely use --demo when showing the demo off as it provides the side by side comparison in the output window

If you haven't done so already, put demo footage in the **BrandColorsProj/Code/** directory. If you don't have footage on hand use Demo_Clip_720p_30.mp4

Basic Predictions Demo

The Normal Predictions Demo is the fastest of demos when it comes to frame rate as it is only rendering the video and the demo overlay. However, it is not as smooth as the gui demo due to data being exchange between the onboard RAM and the swap file on the USB SSD. You will notices a slight hang when this occurs. This is solely a limitation of the hardware.

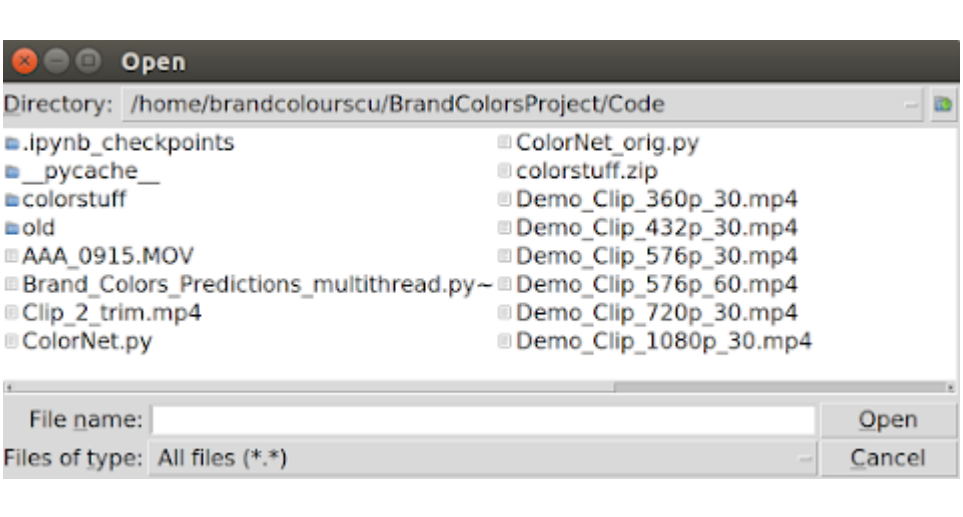
To run the demo follow the steps below:

In the Terminal, with the Code directory selected, use the following code to run the Normal Predictions Demo:

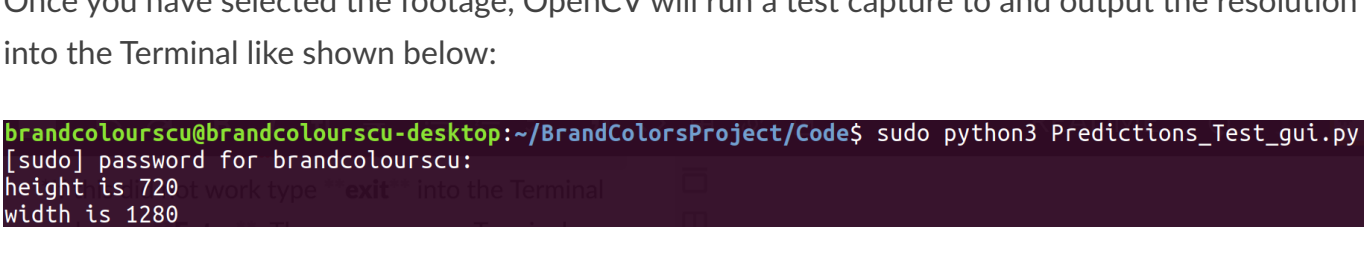
```
sudo python3 Predictions_Test.py -d
```

Note : the -d argument will show the broadcasted image on the left side and the corrected image on the right

When the code starts you will get a file selection box like shown below. Here you can select the footage that you want the demo to run on:



Once you have selected the footage, OpenCV will run a test capture to and output the resolution into the Terminal like shown below:



Note: If the resolution outputed by the Terminal is not correct, press **Ctrl + C** until the demo has stopped and try again

The model will then be loaded on to the GPU and inference will begin. This may take 15-30 seconds to start. If you see an error such as:

```
Failed to load module "canberra-gtk-module"
```

Do not panic, this is a bug with one of the python libraries we are using. This does not effect the demo.

When the demo starts you will now see the footage you selected earlier with its original and corrected frames side by side with the FPS located in the top right corner.

GUI Demo with Sliders

The GUI demo was built for the purpose of showing real time adjustment to the footage. One of the parameters that effect the addition of the layer mask to the original image is the ***Intensity Adjustment Knob*** which will multiply the addition of the image output has shown. While we normally have it as a command line argument and it being a *set it once and forget about it* parameter, with the live GUI this adjustment can be changed with a slider. An added benefit is also being able to move around the Broadcasted vs. Corrected line while inference is running which adds another wow effect.

This demo will perform slightly worse than the Normal Predictions demo. This is because the GUI elements are being rendered on screen. Again, easy problem to solve with more computational horsepower. Even though the performance is worse, the overall playback of the video is smoother. This is because the time wasted to rendering the GUI elements gives time for memory to be shuffled between the onboard RAM and the swapfile on the Nano.

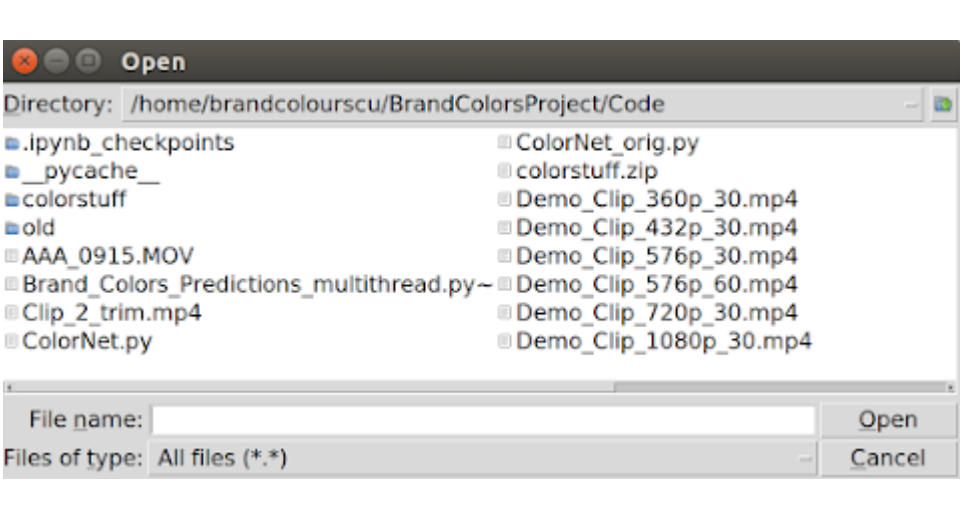
To run the GUI demo follow the steps below:

In the Terminal, with the Code directory selected, use the following code to run the Normal Predictions Demo:

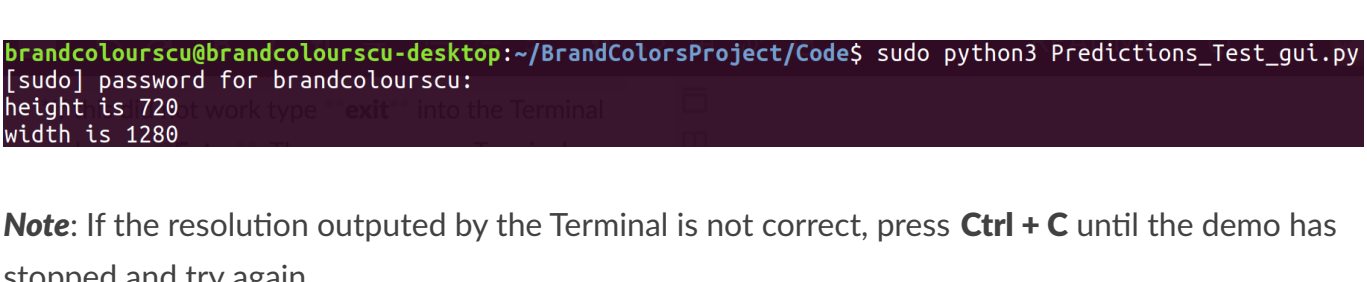
```
sudo python3 Predictions_Test_gui.py -d
```

Note : the -d argument will show the broadcasted image on the left side and the corrected image on the right.

When the code starts you will get a file selection box like shown below. Here you can select the footage that you want the demo to run on:



Once you have selected the footage, OpenCV will run a test capture to and output the resolution into the Terminal like shown below:



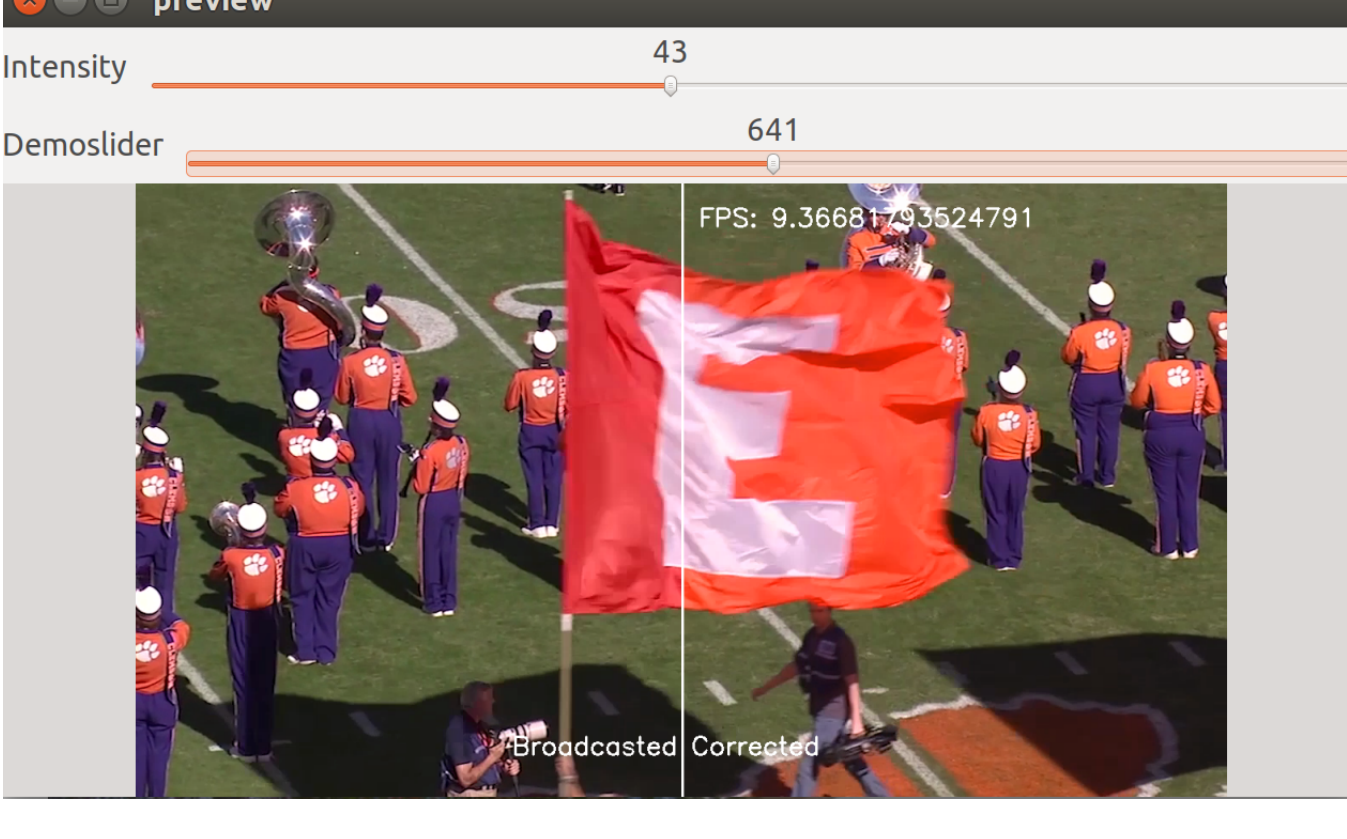
Note: If the resolution outputed by the Terminal is not correct, press **Ctrl + C** until the demo has stopped and try again

The model will then be loaded on to the GPU and inference will begin. This may take 15-30 seconds to start. If you see an error such as:

```
Failed to load module "canberra-gtk-module"
```

Do not panic, this is a bug with one of the python libraries we are using. This does not effect the demo.

An OpenCV window should appear and look like the following:



As shown above the GUI interface contains two sliders: **Intensity** and **Demoslider**. The default value for the **Intensity** slider if 43 (which represents an intensity multiplication of .43). This value, however, we have subjectively chosen and may need changing from scene to scene. The **Demoslider** will change the position of the line that separates the Broadcasted from the Corrected outputs.

This demo has a ***scalable window*** which allows you to resize the demo window to desired size. This will take a hit on performance by 1-4fps.

Webcam Demo

The Webcam Demo is a great way to concretely show that the color correction is in fact happening in real time. However it has a few downsides as of August 13, 2019. The first is that the white balance for the webcam can not be locked, the webcam will change it depending on the scene. This is a limitation of the camera and OpenCV. The second is variability. Currently, the device might change usb ids on the Nano which means you must go in and change parameters in the code for OpenCV to recognize the camera. I plan to fix this is future releases if we get a more usable webcam.

To run the Webcam Demo, follow these steps:

In the Terminal, with the Code directory selected, use the following code to run the Normal Predictions Demo:

```
sudo python3 Predictions_Test_webcam.py -d
```

Note : the -d argument will show the broadcasted image on the left side and the corrected image on the right. Here you can also input the --intensity argument if need be.

An OpenCV window should appear in 15-30 seconds with the Broadcasted vs Corrected Shown.

Note: There will be a noticeable latency in the footage. This is because we are building a inference data stream by threading the input of the frames. This will also cause the frame rate on screen to exceed 30 fps sometimes.

To exit the demo, press **Ctrl + C** at any time.

Last updated: August 15, 2019

Author: John Paul Lineberger (jplineb@clemson.edu)