

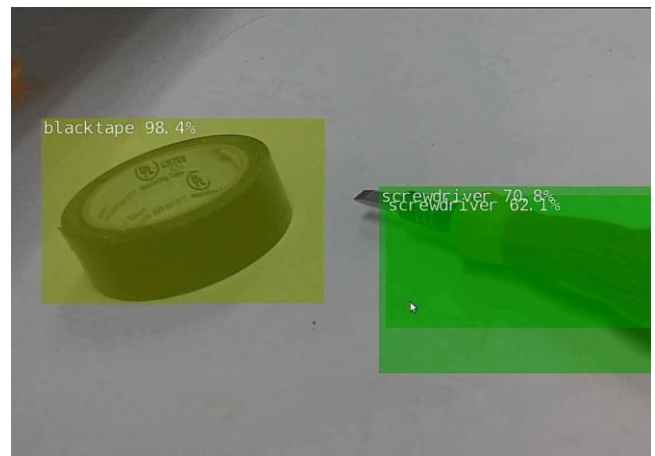
# Train your Jetson Nano to detect objects using Jetson Inference Package

In this guide, you will learn how to train a neural network (SSD-mobilenet v2) to recognise daily objects using the development kit provided by the jetson-inference.

After this the nano should be able to perform the following on live camera footage



From this



To this

# 1. Capturing Training Dataset

Now we may capture the required pictures for helping the neural network to recognise the objects.

In this guide we will train the network to recognise these three objects



**Left:** Black tape

**Middle:** Clear tape

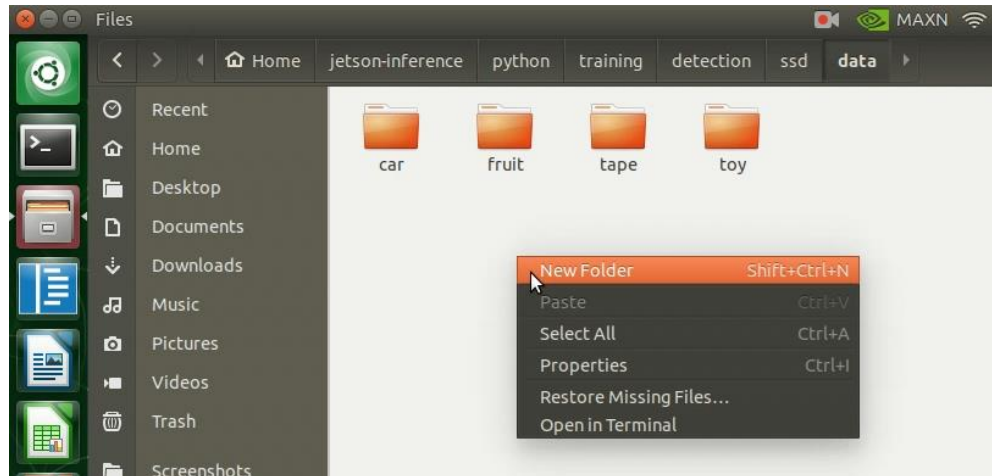
**Right:** Screwdriver

**Note:** After this guide you may use other objects to train the network

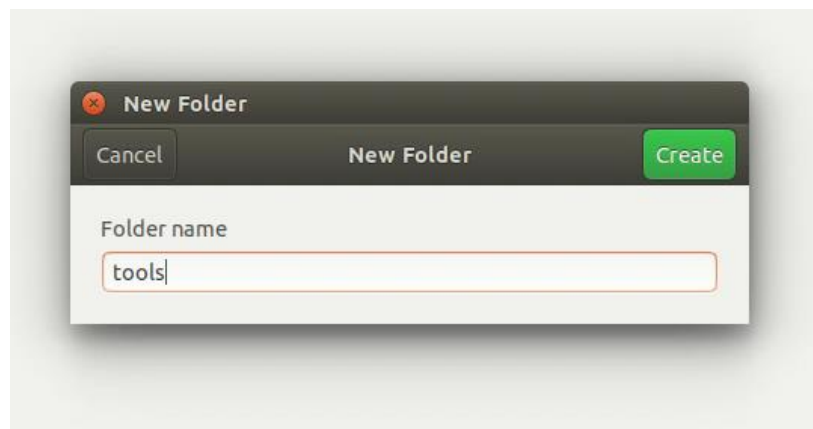
### Step 3.1

Prepare folder

Once the objects are selected, open file manager and navigate to jetson-inference/python/training/detection/ssd/data



Create a new folder by right clicking on empty space nearby, name the folder with tools



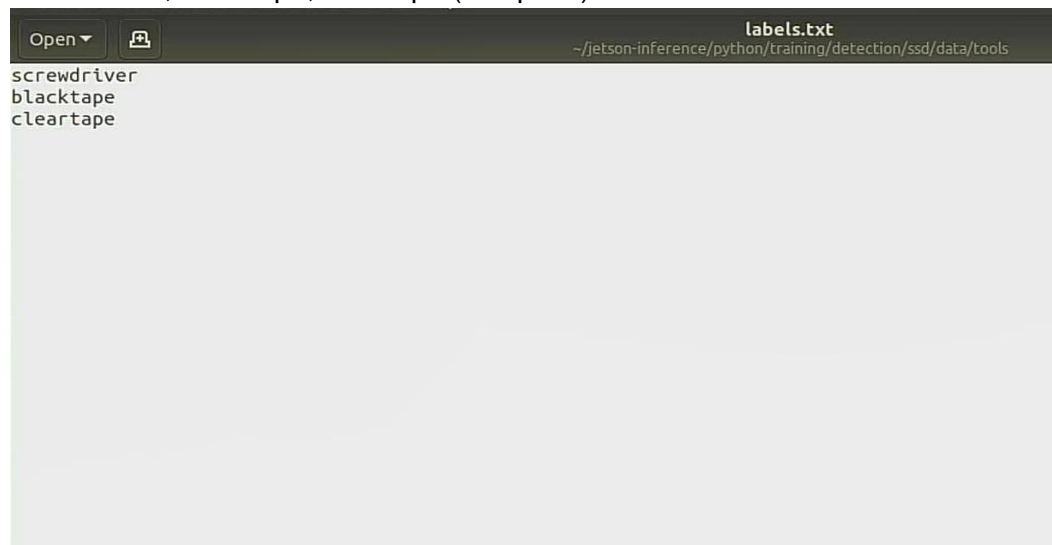
Open the folder and open gedit text editor (click the ubuntu icon and search for "editor" without double quotes)



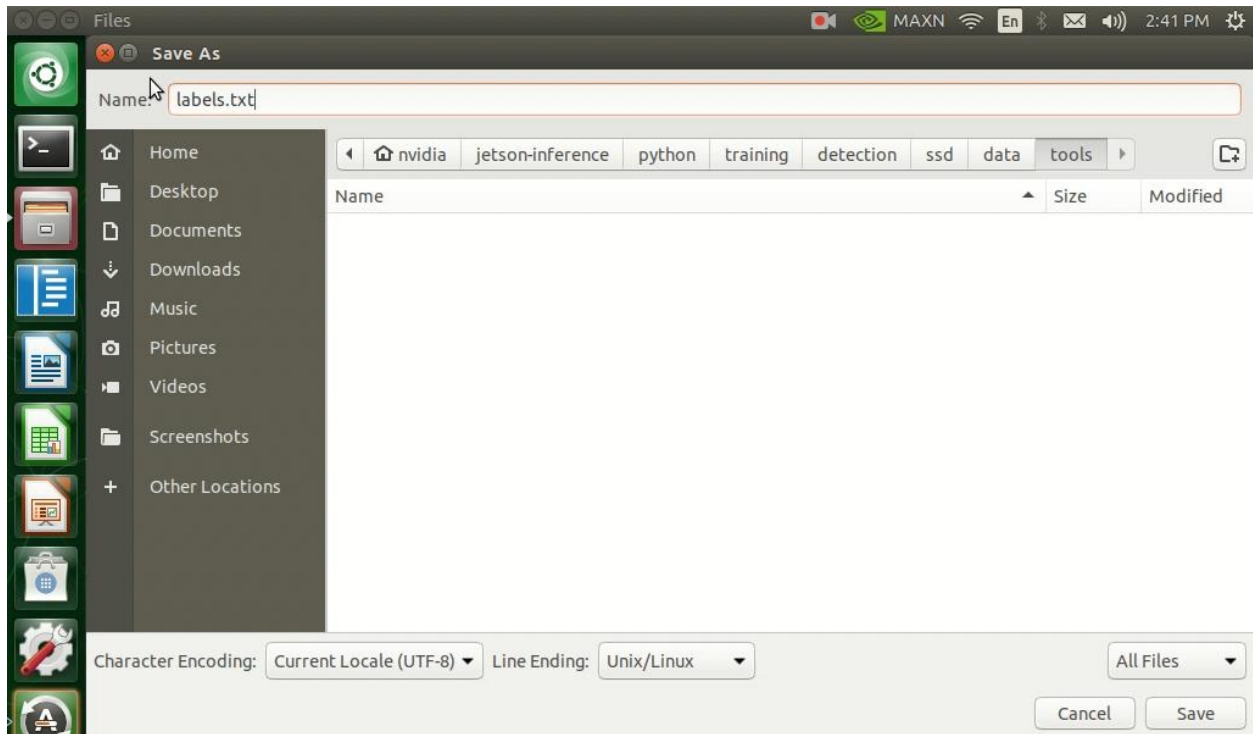
Enter the name of the objects, ONE PER LINE.

**Note: Do not add any spaces**

Screwdriver, blacktape, cleartape (no space)



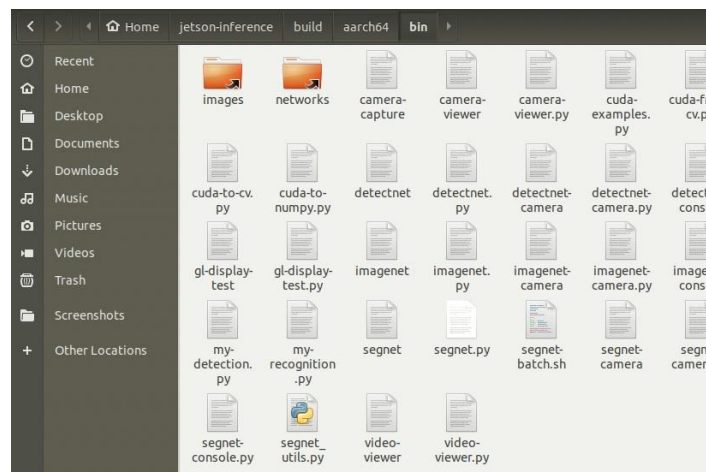
Save as "labels.txt" inside the new folder you created.



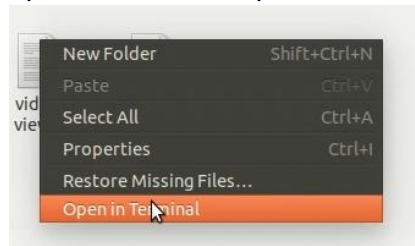
## Step 3.2

Start capturing

Open a file manager and navigate to the folder `jetson-inference/build/aarch64/bin`

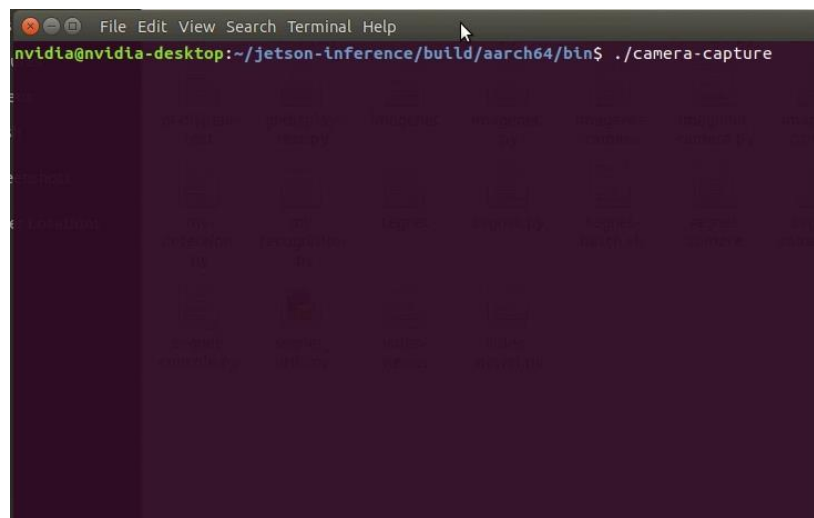


Right click on the nearby empty space and select open terminal here

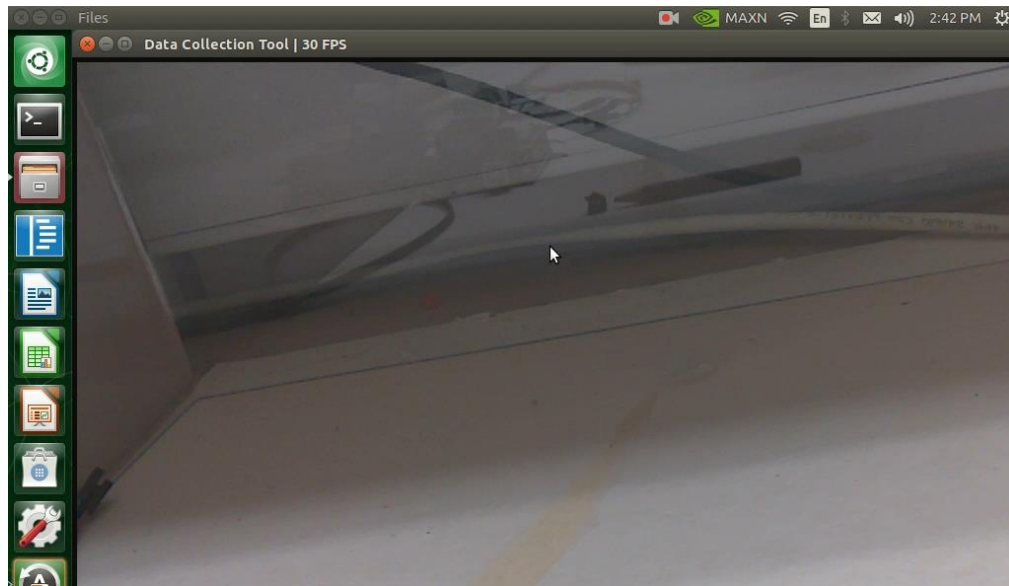


On the terminal type `./camera-capture` without the double quotes

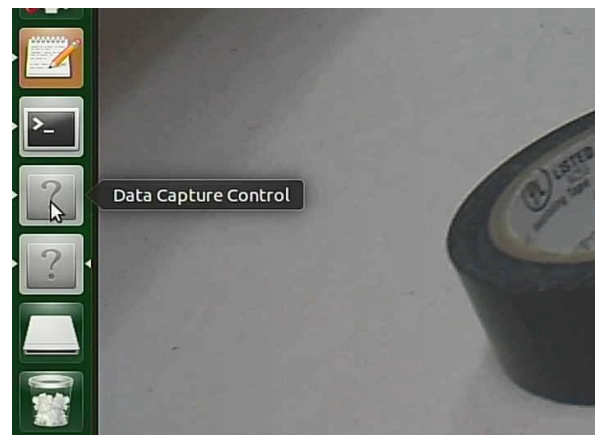
Then Hit Enter

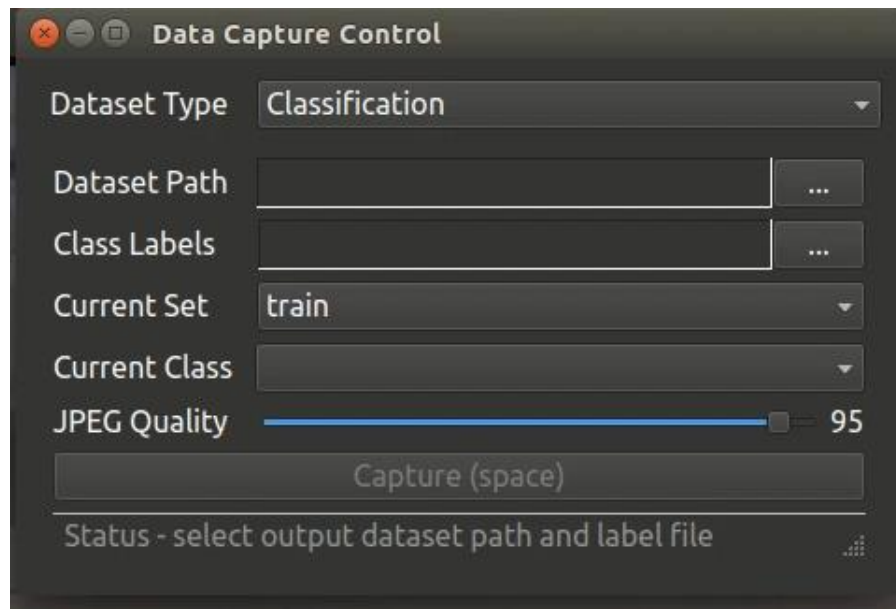


An new window will appear, showing the camera view



Now select another new window

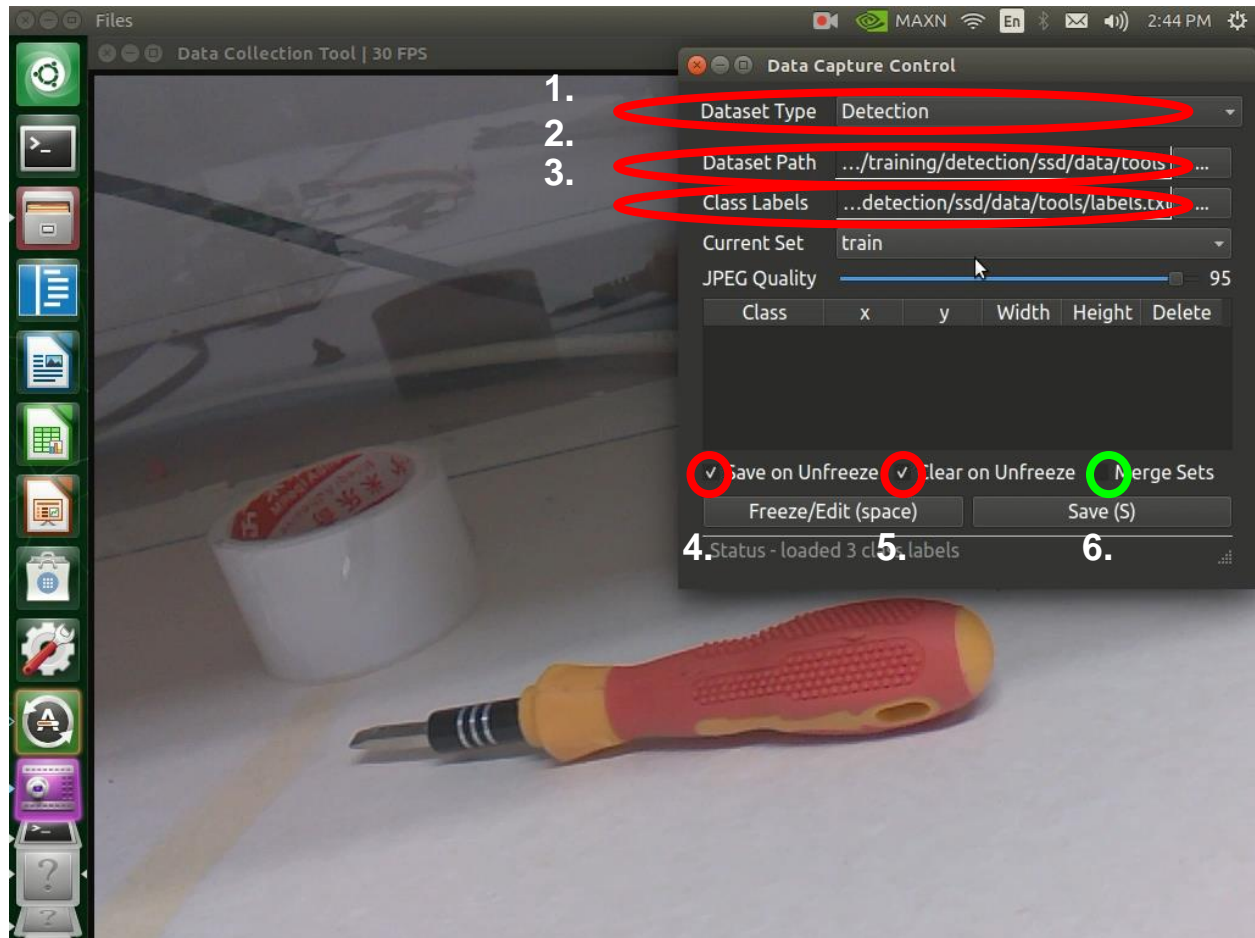




The capture interface will appear



Now select the following



Set the following on the interface

1. Change the Dataset type to detection
2. Select the data set path as the new folder you've created
3. Select class labels as the labels.txt inside that folder
4. Uncheck "Save on Freeze"
5. Uncheck "Clear on Unfreeze"
6. Check "Merge Sets"

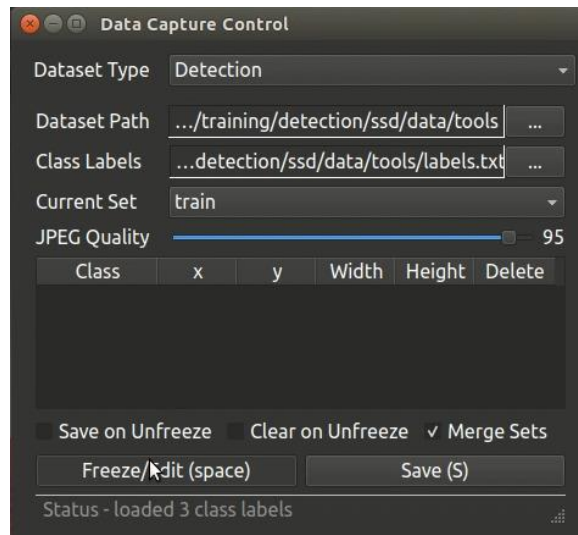
And now you may begin to pose the objects for capturing, follow the following rules

1. Capture the whole object
2. If there are multiple objects, one object must separate from one another
3. Make sure there are enough like to show the object clearly

### Step 3.3

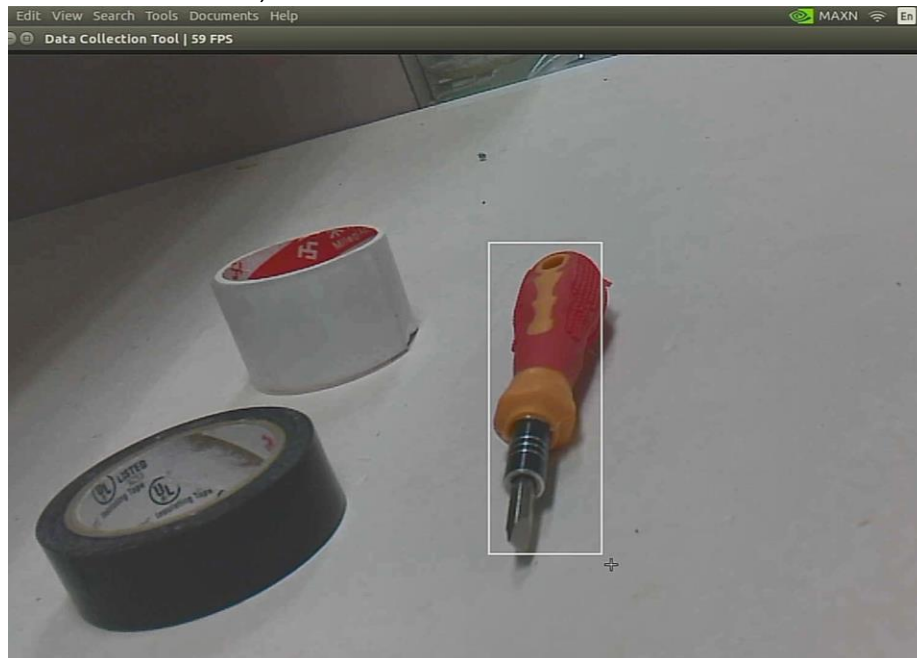
#### Freezing

Now after posing your object, click the freeze button



The screen will now freeze, switch to the window with camera view.

To start tagging the objects, drag the cursor across the object to draw a box (retake without reflection)

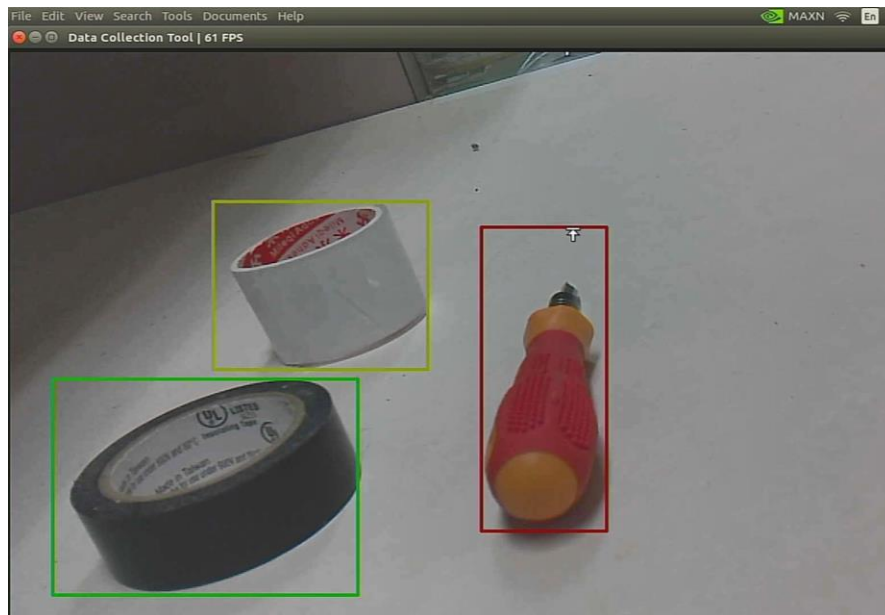


Follow the following rules to create a good tag of the object

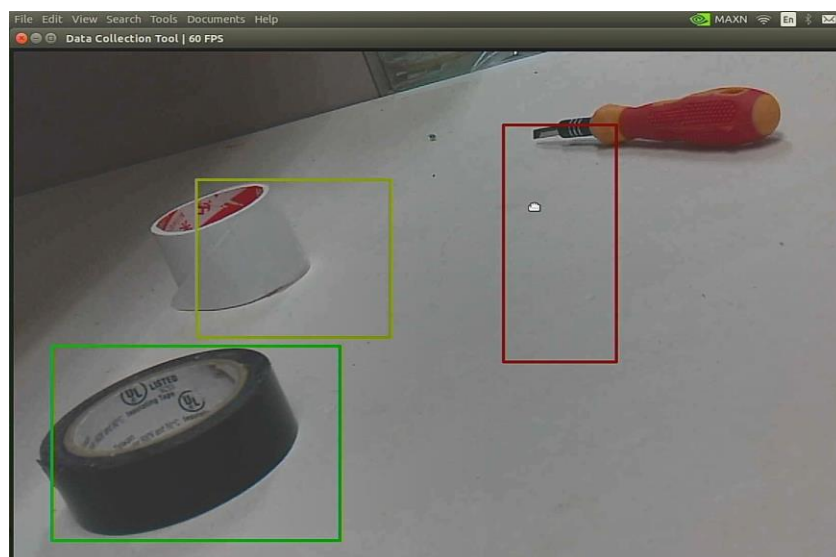
1. **The box must tightly bound the object**
2. **Avoid overlapping of the boxes**

If you have made a mistake, use the following techniques to adjust the boxes:

To adjust the size of the box, put the cursor near the edge of the box until the following icon appears. Then you may stretch it or shrink it along the height or the width.



To adjust the position of the box, place the cursor on the centre of the box and drag it.



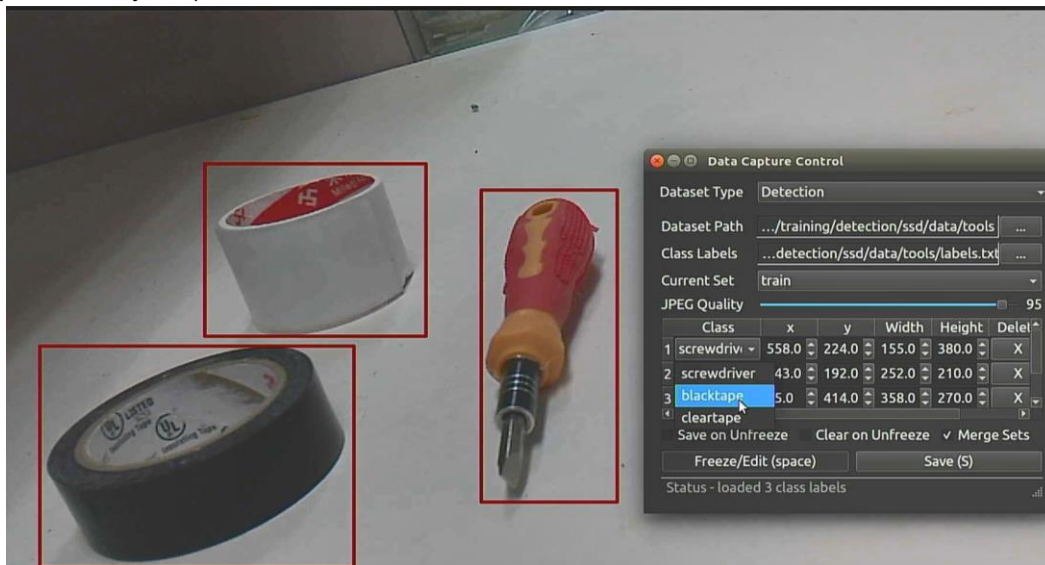


### Step 3.4

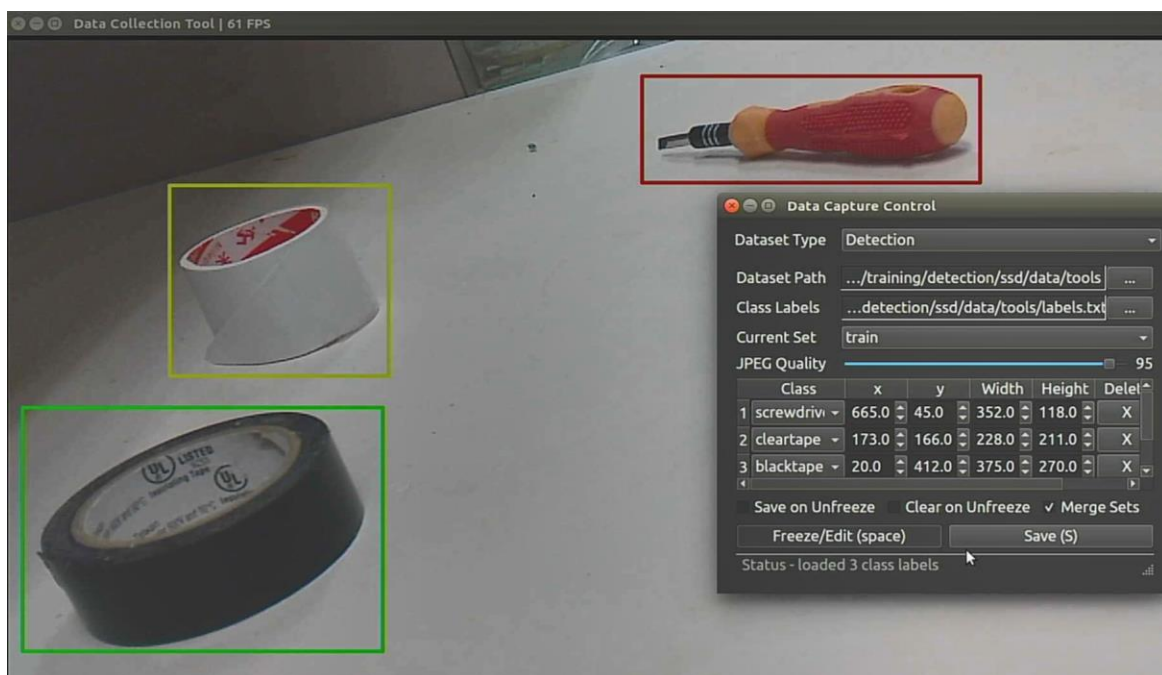
Saving

After EACH tag, do the following:

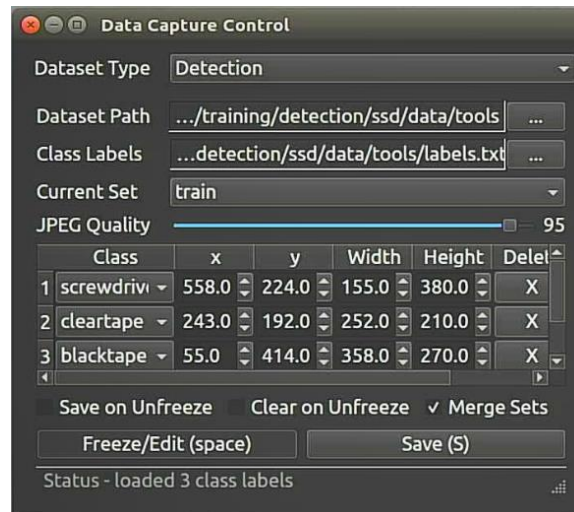
Switch to the capture interface and select the correct class  
(Take pic- diff objects)



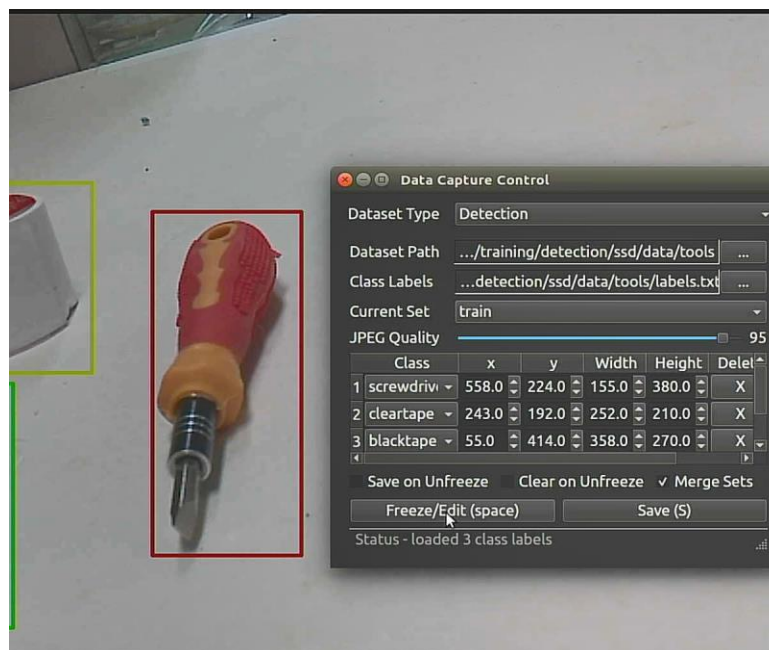
If the classes are clear the boxes would have different colours



If you are done tagging the entire frame, click save  
(retake pic)



Unfreeze the frame and reposition your objects to collect more pictures, that way the neural network can recognise more details of the objects.  
(retake pic)



Once you have done repositioning, freeze the frame and start from Step 3.3 again.

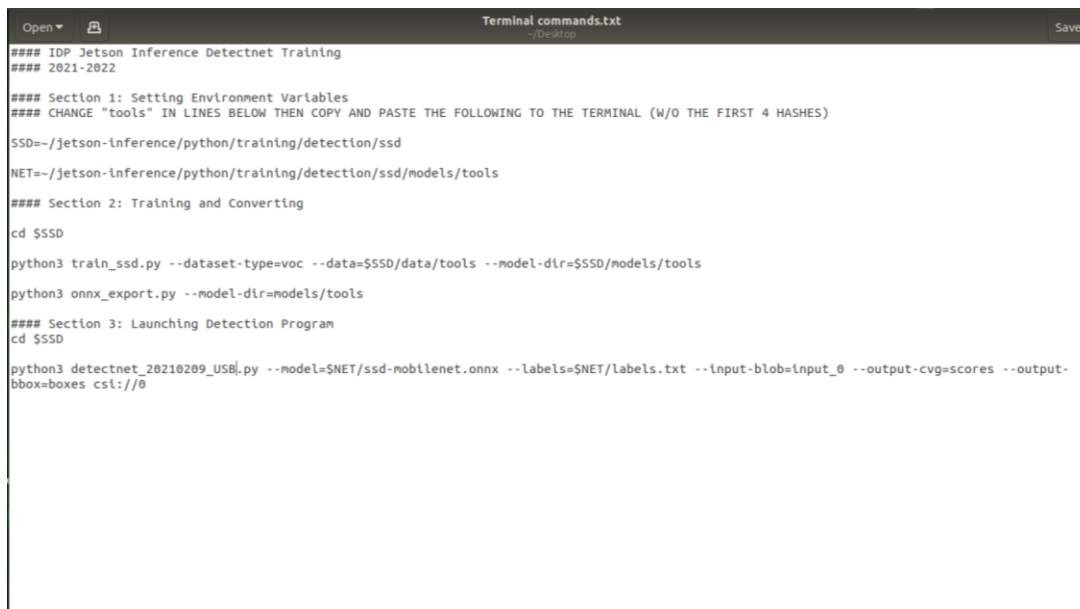
Collect at least 20 pictures to let the network get a firm grip of the objects' details.

## 2. Training the Model

To start training, do the following

### Step 4.1

Open the "Terminal Commands.txt" on the desktop



```
Open ▾  Terminal commands.txt  Save
##### IDP Jetson Inference Detectnet Training
##### 2021-2022

##### Section 1: Setting Environment Variables
##### CHANGE "tools" IN LINES BELOW THEN COPY AND PASTE THE FOLLOWING TO THE TERMINAL (W/O THE FIRST 4 HASHES)

SSD=~/jetson-inference/python/training/detection/ssd
NET=~/jetson-inference/python/training/detection/ssd/models/tools

##### Section 2: Training and Converting

cd $SSD

python3 train_ssd.py --dataset-type=voc --data=$SSD/data/tools --model-dir=$SSD/models/tools
python3 onnx_export.py --model-dir=models/tools

##### Section 3: Launching Detection Program

cd $SSD

python3 detectnet_20210209_USB.py --model=$NET/ssd-mobilenet.onnx --labels=$NET/labels.txt --input-blob=input_0 --output-cvg=scores --output-bbox=boxes csl://0
```

Open a terminal window by clicking on the terminal icon



Copy the two commands from section one to a terminal line-by-line

Now check your vehicle for the following:

1. Make sure the power module is plugged
2. Make sure the charger shows red light



## Step 4.2

Now copy the 3 commands from section 1 of the txt file, LINE-BY-LINE, on to the terminal  
If you've done correctly, the following prompt will appear

```
2021-06-17 15:11:36 - Namespace(balance_data=False, base_net=None, base_net_lr=0.001, batch_size=4, checkpoint_folder='/home/nvidia/jetson-inference/python/training/detection/ssd/models/tools', dataset_type='voc', datasets=['/home/nvidia/jetson-inference/python/training/detection/ssd/data/tools'], debug_steps=10, extra_layers_lr=None, freeze_base_net=False, freeze_net=False, gamma=0.1, lr=0.01, mb2_width_mult=1.0, milestones='80,100', momentum=0.9, net='mb1-ssd', num_epochs=30, num_workers=2, pretrained_ssd='models/mobilenet-v1-ssd-mp-0.675.pth', resume=None, scheduler='cosine', t_max=100, use_cuda=True, validation_epochs=1, weight_decay=0.0005)
2021-06-17 15:11:36 - Prepare training datasets.
2021-06-17 15:11:36 - VOC Labels read from file: ('BACKGROUND', 'screwdriver', 'adhesive', 'screwset')
2021-06-17 15:11:36 - Stored labels into file /home/nvidia/jetson-inference/python/training/detection/ssd/models/tools/labels.txt.
2021-06-17 15:11:36 - Train dataset size: 18
2021-06-17 15:11:36 - Prepare Validation datasets.
2021-06-17 15:11:36 - VOC Labels read from file: ('BACKGROUND', 'screwdriver', 'adhesive', 'screwset')
2021-06-17 15:11:36 - Validation dataset size: 18
2021-06-17 15:11:36 - Build network.
2021-06-17 15:11:37 - Init from pretrained ssd models/mobilenet-v1-ssd-mp-0.675.pth
2021-06-17 15:11:37 - Took 0.51 seconds to load the model.
```

Wait for a minute or two, the training would start and doing training of epochs (epoch here means going through the entire dataset once)

```
2021-06-17 15:11:36 - VOC Labels read from file: ('BACKGROUND', 'screwdriver', 'adhesive', 'screwset')
2021-06-17 15:11:36 - Validation dataset size: 18
2021-06-17 15:11:36 - Build network.
2021-06-17 15:11:37 - Init from pretrained ssd models/mobilenet-v1-ssd-mp-0.675.pth
2021-06-17 15:11:37 - Took 0.51 seconds to load the model.
2021-06-17 15:11:51 - Learning rate: 0.01, Base net learning rate: 0.001, Extra Layers learning rate: 0.01.
2021-06-17 15:11:51 - Uses CosineAnnealingLR scheduler.
2021-06-17 15:11:51 - Start training from epoch 0.
/home/nvidia/.local/lib/python3.6/site-packages/torch/optim/lr_scheduler.py:123:
UserWarning: Detected call of `lr_scheduler.step()` before `optimizer.step()`.
In PyTorch 1.1.0 and later, you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step()`. Failure to do this will result in PyTorch skipping the first value of the learning rate schedule. See more details at
https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
"https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
/home/nvidia/.local/lib/python3.6/site-packages/torch/nn/_reduction.py:44: UserWarning: size_average and reduce args will be deprecated, please use reduction='sum' instead.
warnings.warn(warning.format(ret))
```

**Note: If you have find yourself stuck in the following prompt longer than 5 minutes**



```
File Edit View Search Terminal Help
2021-06-17 15:11:36 - VOC Labels read from file: ('BACKGROUND', 'screwdriver', '
adhesive', 'screwset')
2021-06-17 15:11:36 - Validation dataset size: 18
2021-06-17 15:11:36 - Build network.
2021-06-17 15:11:37 - Init from pretrained ssd models/mobilenet-v1-ssd-mp-0_675.
pth
2021-06-17 15:11:37 - Took 0.51 seconds to load the model.
2021-06-17 15:11:51 - Learning rate: 0.01, Base net learning rate: 0.001, Extra
Layers learning rate: 0.01.
2021-06-17 15:11:51 - Uses CosineAnnealingLR scheduler.
2021-06-17 15:11:51 - Start training from epoch 0.
/home/nvidia/.local/lib/python3.6/site-packages/torch/optim/lr_scheduler.py:123:
UserWarning: Detected call of `lr_scheduler.step()` before `optimizer.step()`.
In PyTorch 1.1.0 and later, you should call them in the opposite order: `optimiz
er.step()` before `lr_scheduler.step()`. Failure to do this will result in PyTo
rch skipping the first value of the learning rate schedule. See more details at
https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
"https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", User
Warning)
/home/nvidia/.local/lib/python3.6/site-packages/torch/nn/_reduction.py:44: UserW
arning: size_average and reduce args will be deprecated, please use reduction='s
um' instead.
warnings.warn(warning.format(ret))
```

Hit Ctrl-C and enter the previous command again

If the training is going smoothly, the command line would show something like the following

```
File Edit View Search Terminal Help
https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
"https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", User
Warning)
/home/nvidia/.local/lib/python3.6/site-packages/torch/nn/_reduction.py:44: UserW
arning: size_average and reduce args will be deprecated, please use reduction='s
um' instead.
  warnings.warn(warning.format(ret))
2021-06-17 15:15:48 - Epoch: 0, Validation Loss: 7.6973, Validation Regression L
oss 2.6682, Validation Classification Loss: 5.0291
2021-06-17 15:15:48 - Saved model /home/nvidia/jetson-inference/python/training/
detection/ssd/models/tools/mb1-ssd-Epoch-0-Loss-7.697316265106201.pth
2021-06-17 15:15:56 - Epoch: 1, Validation Loss: 5.9695, Validation Regression L
oss 2.5941, Validation Classification Loss: 3.3754
2021-06-17 15:15:56 - Saved model /home/nvidia/jetson-inference/python/training/
detection/ssd/models/tools/mb1-ssd-Epoch-1-Loss-5.96950330734253.pth
2021-06-17 15:16:07 - Epoch: 2, Validation Loss: 5.7521, Validation Regression L
oss 2.4730, Validation Classification Loss: 3.2791
2021-06-17 15:16:07 - Saved model /home/nvidia/jetson-inference/python/training/
detection/ssd/models/tools/mb1-ssd-Epoch-2-Loss-5.752058124542236.pth
2021-06-17 15:16:14 - Epoch: 3, Validation Loss: 4.6599, Validation Regression L
oss 1.7022, Validation Classification Loss: 2.9577
2021-06-17 15:16:15 - Saved model /home/nvidia/jetson-inference/python/training/
detection/ssd/models/tools/mb1-ssd-Epoch-3-Loss-4.659869146347046.pth
```

Enter the last command and the training is done

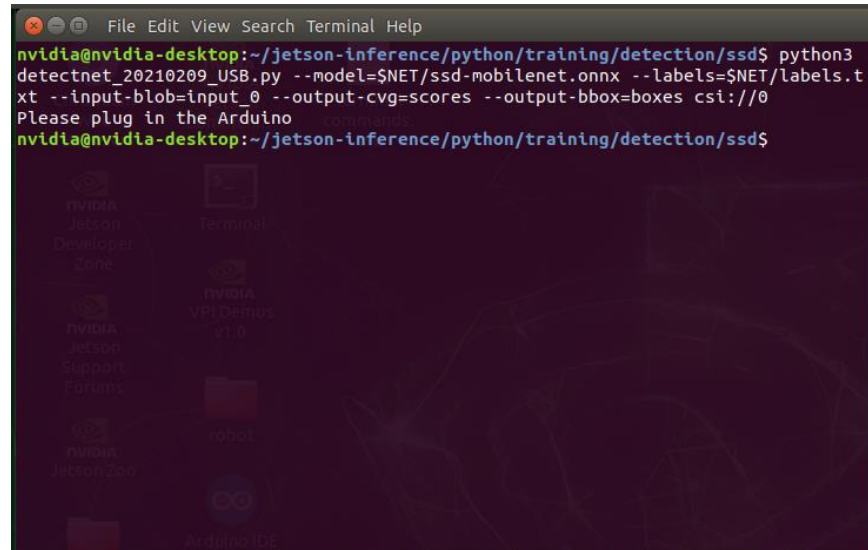
```
File Edit View Search Terminal Help
nvidia@nvidia-desktop:~/jetson-inference/python/training/detection/ssd$ python3
onnx_export.py --model-dir=models/tools
Namespace(batch_size=1, height=300, input='', labels='labels.txt', model_dir='mo
dels/tools', net='ssd-mobilenet', output='', width=300)
running on device cuda:0
found best checkpoint with loss 1.035489 (models/tools/mb1-ssd-Epoch-29-Loss-1.0
354889392852784.pth)
creating network: ssd-mobilenet
num classes: 4
loading checkpoint: models/tools/mb1-ssd-Epoch-29-Loss-1.0354889392852784.pth
exporting model to ONNX...
```

### 3. Launch the Detection Program

#### Step 5.1

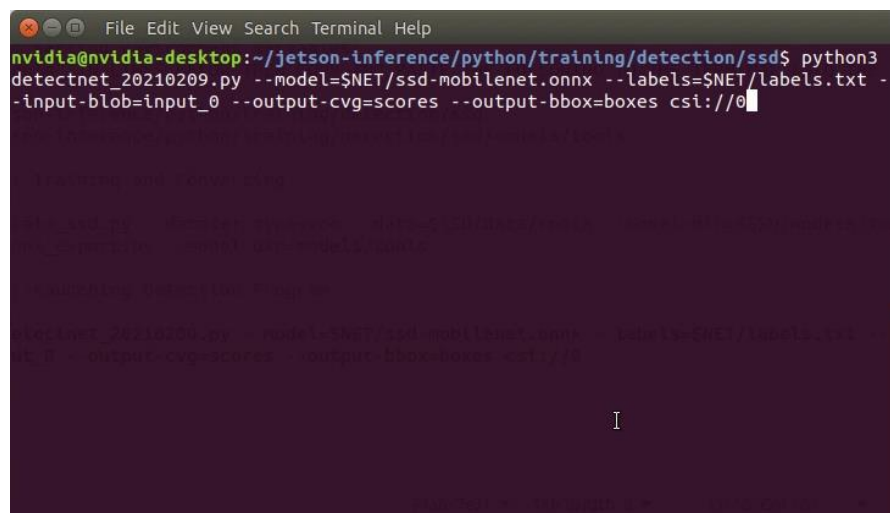
Make sure the Arduino is connected to the Nano via a USB cable

**Note: If the Arduino not plugged in, the following error would appear**



```
nvidia@nvidia-desktop:~/jetson-inference/python/training/detection/ssd$ python3
detectnet_20210209_USB.py --model=$NET/ssd-mobilenet.onnx --labels=$NET/labels.t
xt --input-blob=input_0 --output-cvg=scores --output-bbox=boxes csi://0
Please plug in the Arduino
nvidia@nvidia-desktop:~/jetson-inference/python/training/detection/ssd$
```

Now open "Terminal Commands.txt" again, copy commands from section 3, line-by-line, onto the terminal.



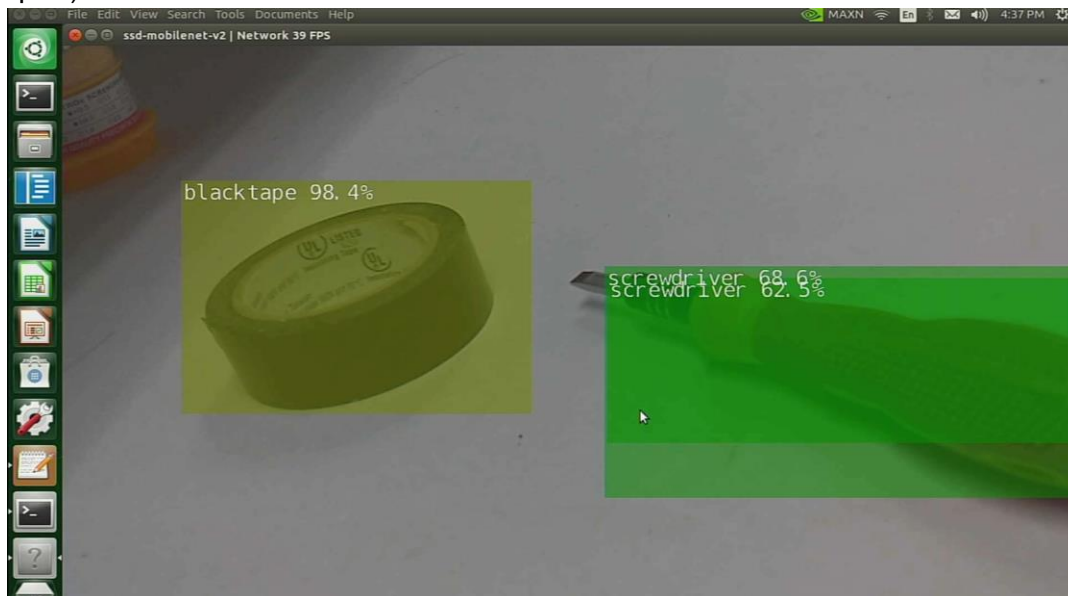
```
nvidia@nvidia-desktop:~/jetson-inference/python/training/detection/ssd$ python3
detectnet_20210209.py --model=$NET/ssd-mobilenet.onnx --labels=$NET/labels.txt -
-input-blob=input_0 --output-cvg=scores --output-bbox=boxes csi://0
```

And hit enter

First time loading the model would take a few minutes, please be patient

```
File Edit View Search Terminal Help
[TRT] Replacing slice Slice_210 with copy from 460 to 461
[TRT] Eliminating concatenation Concat_194
[TRT] Generating copy for 295 to 445
[TRT] Generating copy for 331 to 445
[TRT] Generating copy for 359 to 445
[TRT] Generating copy for 387 to 445
[TRT] Generating copy for 415 to 445
[TRT] Generating copy for 443 to 445
[TRT] Eliminating concatenation Concat_193
[TRT] Generating copy for 283 to 444
[TRT] Generating copy for 319 to 444
[TRT] Generating copy for 347 to 444
[TRT] Generating copy for 375 to 444
[TRT] Generating copy for 403 to 444
[TRT] Generating copy for 431 to 444
[TRT] Eliminating concatenation Concat_209
[TRT] Generating copy for 453 to 460
[TRT] Generating copy for 459 to 460
[TRT] Eliminating concatenation Concat_220
[TRT] Generating copy for 465 to boxes
[TRT] Generating copy for 470 to boxes
[TRT] After concat removal: 85 layers
[TRT] Graph construction and optimization completed in 0.109292 seconds.
```

At last you will see a window with the camera view, but this time the objects are being detected (retake pics)



The more pictures you used to train, there will be less error in detecting these objects.

Try to place only a single object in front of the camera and move it slightly left or right, the camera should follow

## 4. References

<https://github.com/dusty-nv/jetson-inference/>

### Criteria

To shorten the learning process, the objects of choice are recommended to have the following.

1. Short in length (NO sudden change in shape)
2. Distinct colour from the surroundings
3. Small

Examples:



Suitable



NOT Suitable