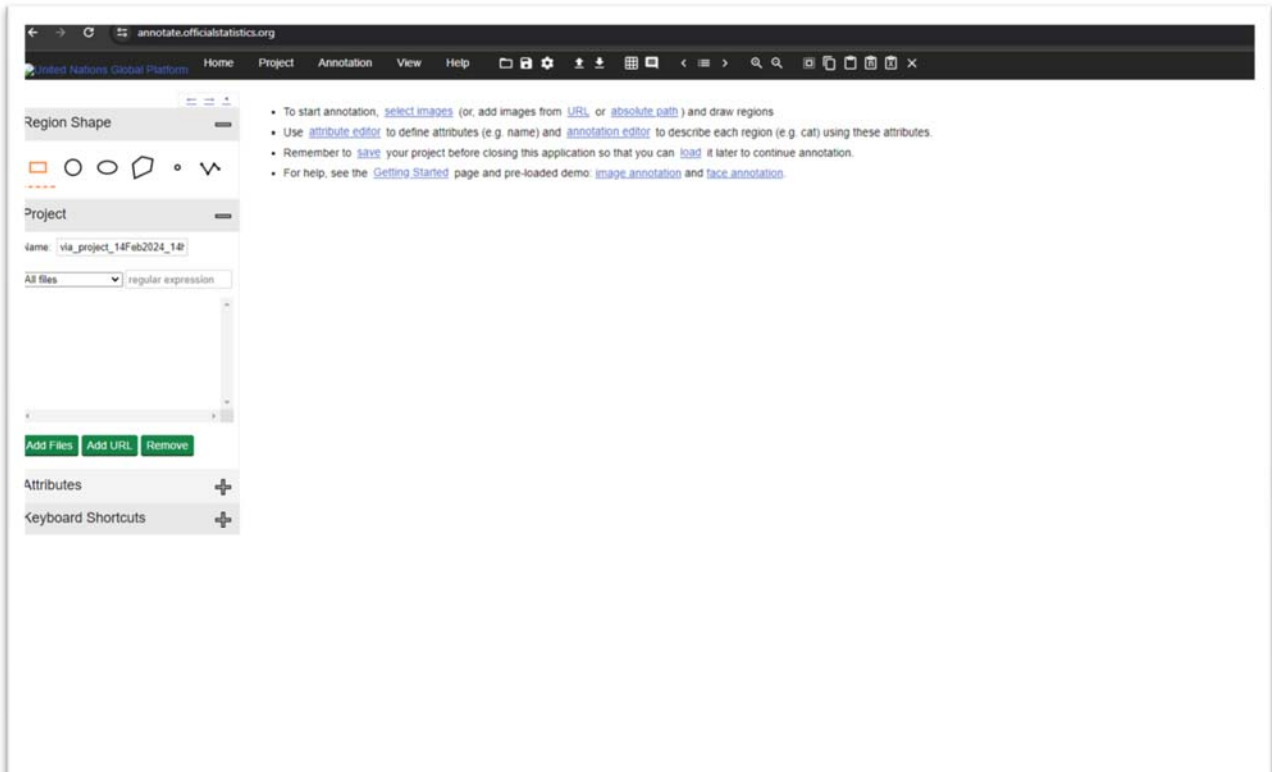# GUI Help Guide

## Quick Setup
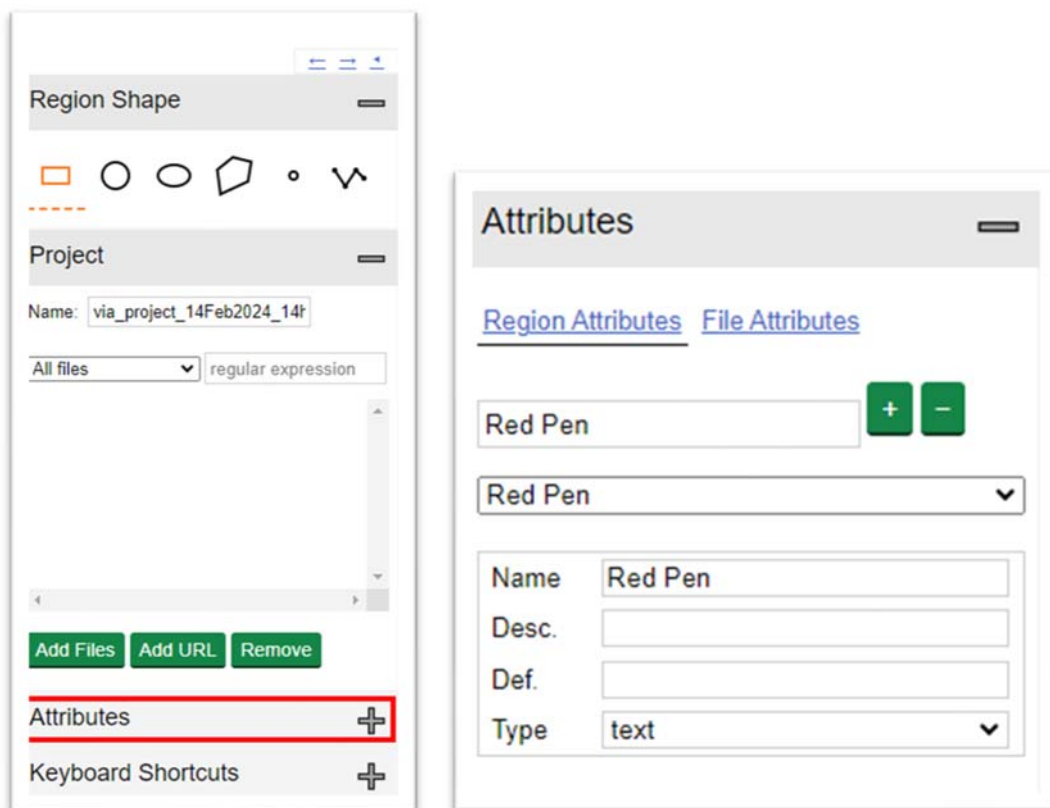
1. Gather all images
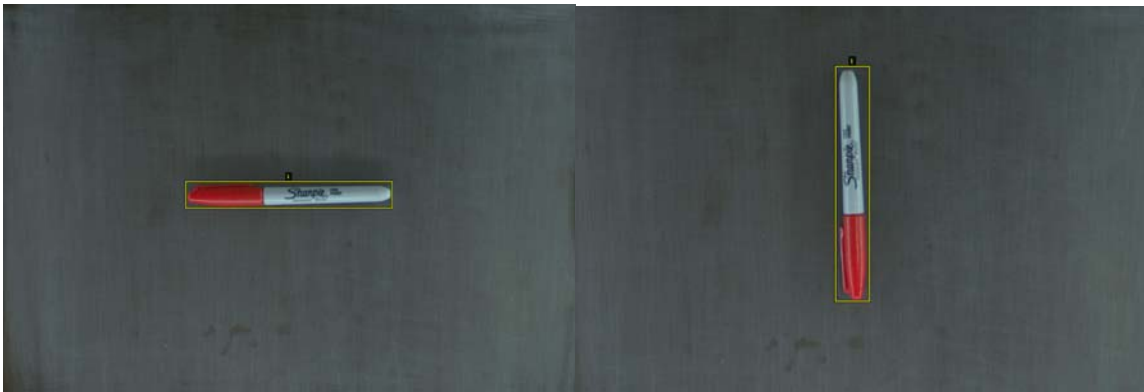2. Annotate images using vgg annotation tool. https://annotate.officialstatistics.org/

3. Add your images
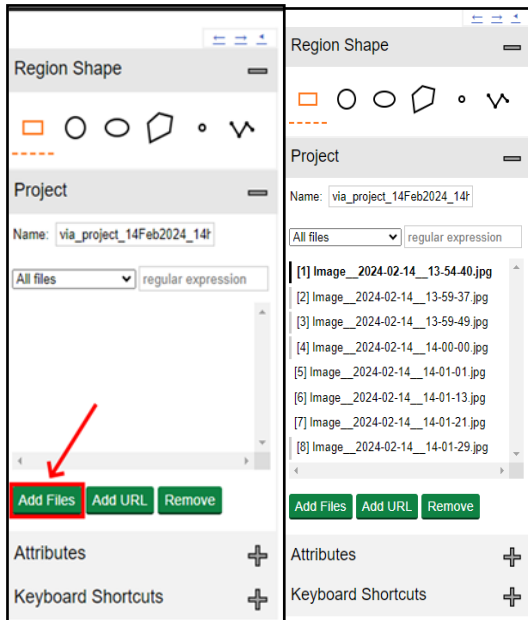


4. Add an attribute (object name).

5. Create bounding boxes around each object of this type in ALL images.



    a.  If training to detect multiple objects:

    b.  After all objects of attribute "X" have been labled, DELETE the attribute and create a new attribute.
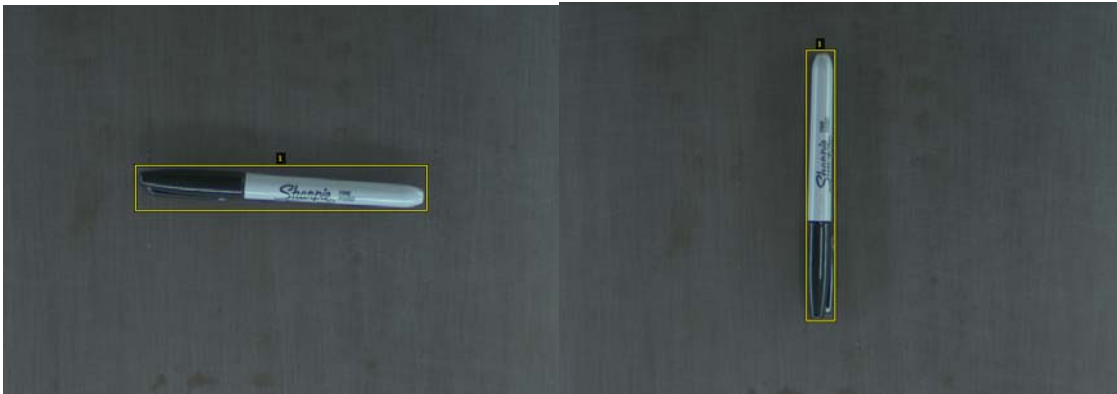
c. If multiple attributes exist at a single time the annotation file will be corrupted. The attribute drop-down MUST only contain a single attribute.

6. Once all object and object types are labled, export the annotations as a .CSV file.
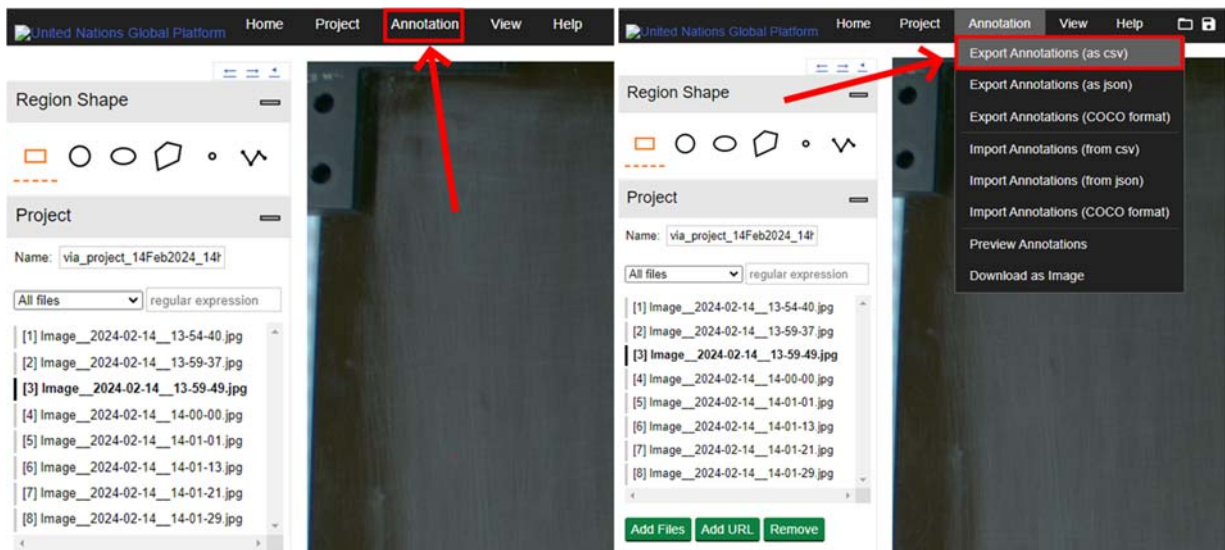


7. Launch application
8. Go to File->Create New Project
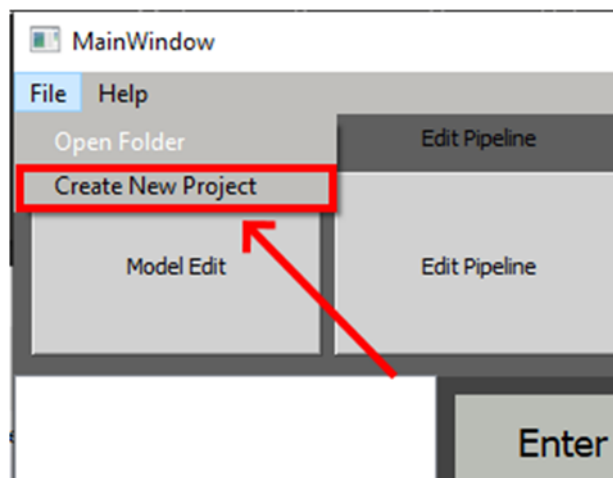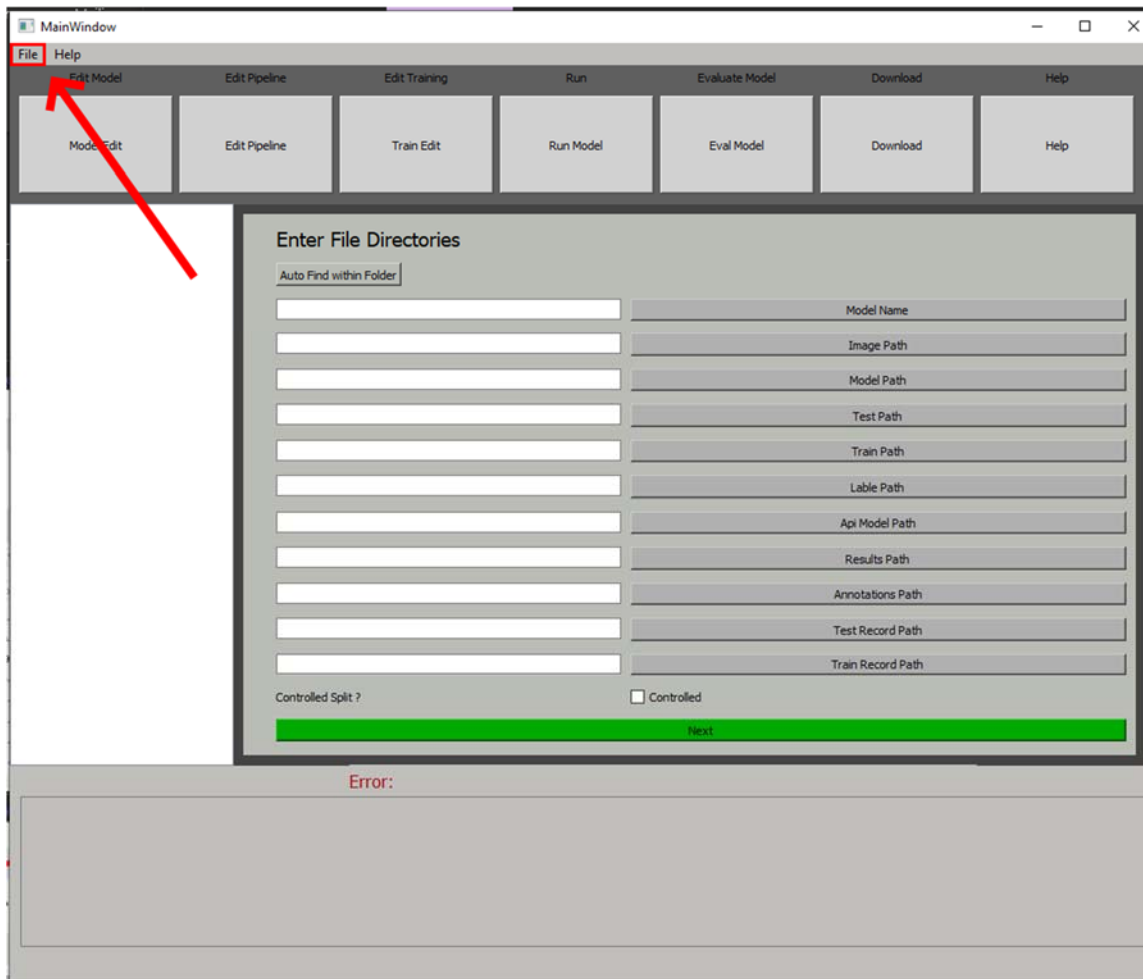
9. Once the create project dialog opens enter a project name and select a save location for your project.
   a. If desired detection models can be downloaded directly from the create project page. This is the suggested method of downloading tensorflow models.

10. Click "Next"
11. The next page loads all user-created files into the project folder. Enter images and annotations (From VGG). If images are not the same size as the selected model image size use the resize option to resize all images. Note images are expected to be square (ie:

640x640 or 1024x1024) for training to be successful.



a. A controlled split ensures that training/test images have all annotations that exist in a single image available. If a controlled split is not selected and there are multiple objects in a single image annotations may be split in ways that cause the model to not converge properly.

12. Once images and annotations have been provided press "Next" to progress to the next page.

a. Label maps are dynamically created must labels must match the labels used during the annotation step. This does include capital letters!!

Dialog                                    ?    ✕

# Create a new Project

Object

Enter Object Lables followed by ",".

|                                        |

Example: Dog, Cat, Gofer, Turtle

| Confirm |

Downloading...

| Finish |

13. Type all lables followed by a comma. Once all labels have been added press "Confirm" and they will appear in the right side table. If satisfied press "Finish".



Dialog                                    ?    ✕

# Create a new Project

Object

Enter Object Lables followed by ",".

| Red Pen, Black Pen |

| 1 | Red Pen |
| 2 | Black Pen |

Example: Dog, Cat, Gofer, Turtle

| Confirm |

Note: the model will now be downloaded and extracted. This may take a while depending on internet speeds. To ensure the model is finished downloading check the model name if it is "Detection Model" the download is complete. Else the download/extraction is still in progress.

14. Once "Finished" is pressed the dialog will close.

## Importing a project

1. Click on File->Open Folder
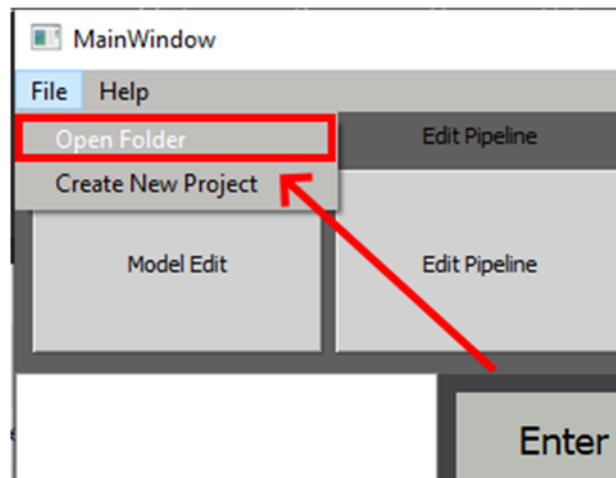


2. Open the project folder.
   a. The project directory will then be displayed in the left hand table.
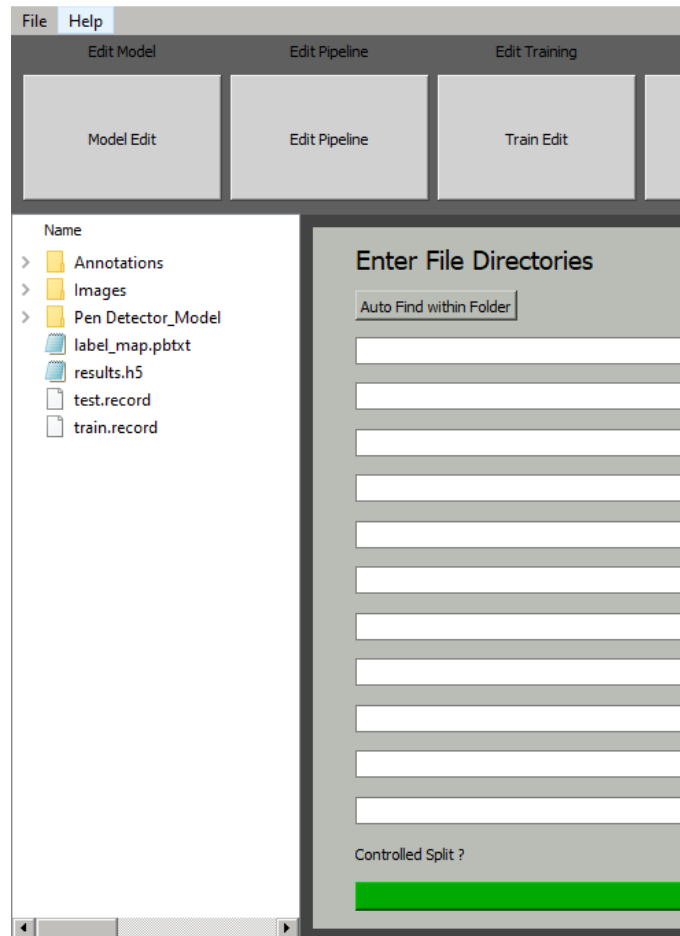
## Training a Model

1. With a project loaded press "Auto Find within Folder". This will automatically load all model training parameters. EXCEPT the location of TensorFlow installation directory, this must be entered manually.

**Enter File Directories**

Auto Find within Folder

| | |
|---|---|
| Detection_Model | Model Name |
| C:/Users/sseaberry/Documents/test/Pen Detector/Images | Image Path |
| Pen Detector_Model/trained_models/checkpoints/Detection_Model | Model Path |
| C:/Users/sseaberry/Documents/test/Pen Detector | Test Path |
| C:/Users/sseaberry/Documents/test/Pen Detector | Train Path |
| C:/Users/sseaberry/Documents/test/Pen Detector/label_map.pbtxt | Lable Path |
| C:/Users/sseaberry/Documents/TensorFlow/models-master | Api Model Path |
| C:/Users/sseaberry/Documents/test/Pen Detector/results.h5 | Results Path |
| C:/Users/sseaberry/Documents/test/Pen Detector/Annotations | Annotations Path |
| C:/Users/sseaberry/Documents/test/Pen Detector/train.record | Test Record Path |
| C:/Users/sseaberry/Documents/test/Pen Detector/test.record | Train Record Path |

Controlled Split ?            ☐ Controlled

Next

2. Click on the API model path and select the tensorflow/models-master/ folder. As shown above.
    a. If using a controlled annotations split click on the "Controlled" checkbox.
3. Press "Next" to advance to the next page.
    a. The next page gives the ability to edit the models training parameters

**Model Pipeline Editor**

Faster Rcnn (Inception or other) ☑
Centernet ☐
SSD ☐
Efficent Det ☐

| | |
|---|---|
| Number of Classes | 2 |
| Batch Size | 1 |
| Object Classification Weight | 1 |
| Max Number of Boxes | 1 |
| Object Localization Weight | 1| |

[ Next ]

4. Select the correct model type (here FRCNN).
5. Number of classes: number of independent annotations.
6. Batch size: number of images to train on before updating weights/biases. If training on a single GPU/CPU set this value to 1-6. (NOTE: larger numbers will need more computation resources. If getting a OOM error when training lower this value.)
7. Object Classification Weight/Object Localization Weight: Set these to values floating point numbers above 1. If classification accuracy is more important set this value larger than the localization and vise versa.
8. Number of boxes: Set to the largest number of boxes in a single image. NOTE: If training on images with > 100 objects in a single image, see fine-tuning section.
9. Press "Next" to progress to the next page.
10. On the "Train Inputs" page:

## Train Inputs

| | |
|---|---|
| 0.1 | Learning Rate |
| 100000 | # of Steps |

**Source Image Shape (W, H)**

| |
|---|
| 2064 |
| 3088 |

**Trained Image Shape**

| |
|---|
| 640 |

If Controlled Data Split Used Enter your Train and Test Annotation Files Below

| | |
|---|---|
| | Train Annotations |
| | Test Annotations |

<div style="background-color:green">Load Model</div>

a. Enter a learning rate between 1 and 0.001. This value has a large impact on the models training losses. Too large and losses will increase, too small and they will decrease too slowly/overfit. It is suggested to start at 0.1 and decrease by half if the model loss is increasing.

b. # of steps: Set to value above 100,000 idea is 300,000+.

c. Source image shape: Used to correctly scale annotations, set these values to the Width/heigh of the original images use to create annotations.

d. Trained image shape: Size of model training images. Selected during the project creation process.

e. If using a controlled split:

   i. Double click and drag training/test annotation files from the left-hand directory viewer into their respective boxes.

15. Press "Load Model".

   a. If model loading is successful, the Error display will show "None".

16. From the top taskbar select "Run Model"

17. If there are no errors and the Status reads "Loaded", press "Run"
18. If successful the Status will read "Running" and logging output will start to roll out on the log display below.
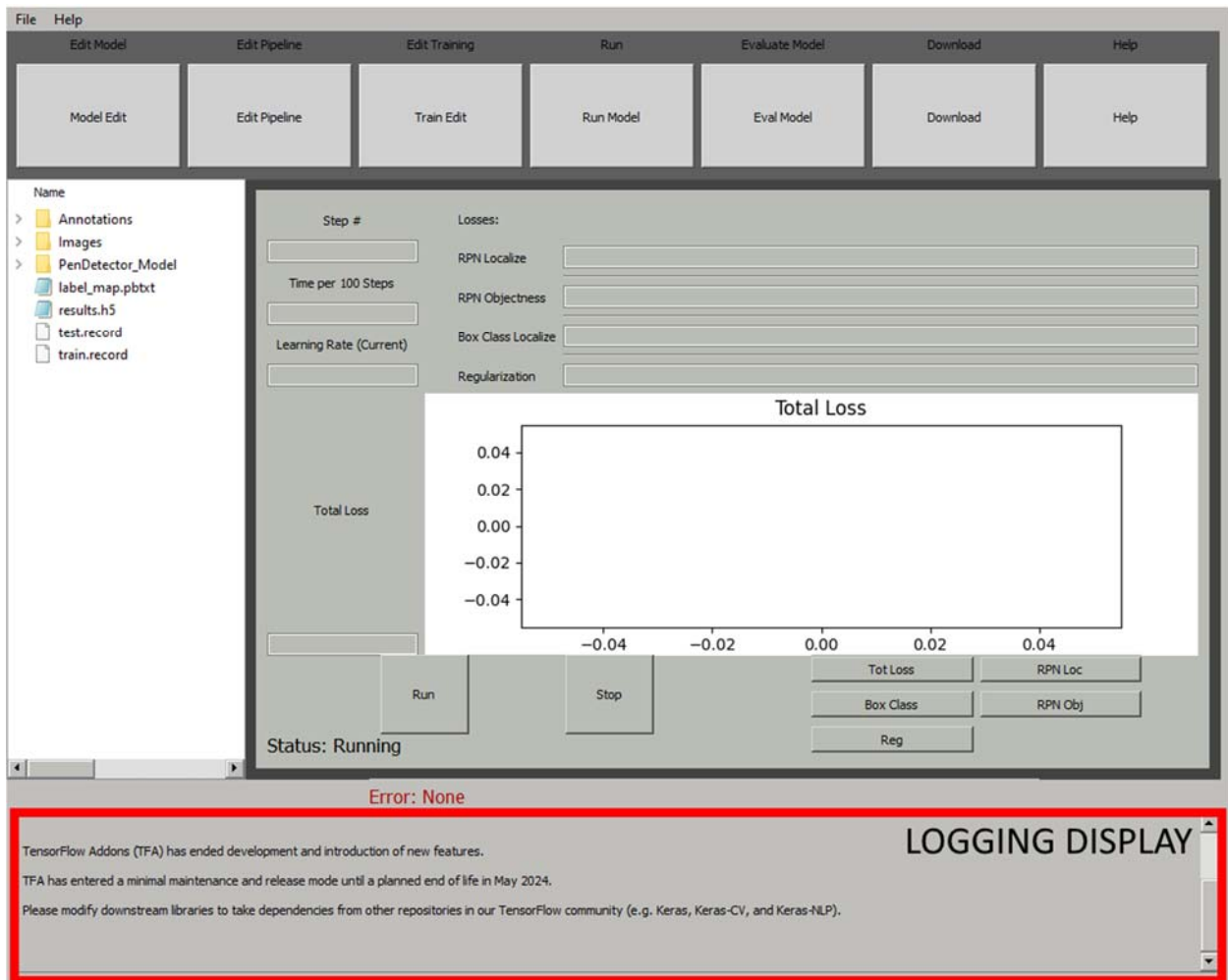
During the models training the losses are updated and saved so that graphs of each loss can be viewed in real time. NOTE: If training on a single GPU/CPU loss values can take a long time to update due to slow model training speeds. It is typical to see +1h of delay between loss updates. This is VERY model/image size dependent.

## Evaluating a Model

Click on "Eval Model" in the top taskbar



The model eval page:

**Evaluate Model**

☐ Evaulate from Video (Camera)    ☑ Evaluate Single Image

| | |
|---|---|
| Detection_Model | Model Name |
| ctor_Model/trained_models/checkpoints/Detection_Model/pipeline.config | Pipeline Path |
| ector/PenDetector_Model/trained_models/checkpoints/Detection_Model | Checkpoint Path |
| cuments/test/PenDetector/Images/Image__2024-02-14__13-59-49.jpg | Image Path |
| C:/Users/sseaberry/Documents/test/PenDetector/label_map.pbtxt | Lable Path |
| C:/Users/sseaberry/Pictures | Save Path |

| | |
|---|---|
| Checkpoint Number | 3 |
| Threshold Value | 0.1 |

**Evaluate**

Status: Evaluating

First choose to either evaluate the model of a single static image (Evaluate Single Image) or to evaluate the model on a constantly updated video (Evaluate from Video).

As before on the Run page type the model's name. Note if the project was set up using the Create Project function this will be "Detection_Model".

Pipeline path should point to the pipeline.config file in the model s folder.

Checkpoint path should point to the folder containing the checkpoints.

Image path: If evaluating from a single image enter the path to the image. If evaluating from camera this entry can be set to anything.

Label path should point to the labelmap.pbtxt of the model.

Save path location to save images too.

Checkpoint number is an integer value corresponding to the checkpoint of the model.

Threshold value is a floating-point number between 1 and 0 corresponding to the accepted certainty levels for detected objects.

Evaluation from Video will pick up video from whatever camera is attached to the user's machine. If no camera is detected the evaluation will terminate unexpectedly.

# Fine Tuning

Fine-tuning is done by editing the models pipeline.config file. For a full explanation of the pipeline file and all its options visit:
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md

If there are more than 100 objects per image the pipeline.config must be edited to include this line in the EVAL INPUT READER section:

max_number_of_boxes: 500 (or whatever is the max number of objects in a single image)

If training on images that are not similar to those in the COCO dataset users may want to start training a model from scratch. To do so, remove the FINE_TUNE_CHECKPOINT lines in the pipeline.config file.

If the objects/images are not subjected to large variations in lighting, or coloring users may want to train without data augmentation. To do so; remove all the data augmentation sections from the pipeline.config.

# Setup Without Using Create Project

## Model Training:

To train the model, several parameters must be given before the program can run. Most of these are paths to files or folders.

- Model Name, STRING corresponding to saved model folder name
- Image Path, FOLDER PATH containing all images to be used for both training and testing
- Model Path, FOLDER PATH to downloaded model folder
- Config Path, FILE PATH pointing to pipeline.config file
- Test Path, FOLDER PATH to location of test.record
- Train Path, FOLDER PATH to location of test.record
- Lable Path, FILE PATH pointing to lablemap in .pbtxt form
- Apimodel Path, FOLDER PATH to tensorflow object detection models folder
- Checkpoint Path, FOLDER PATH to selected models checkpoint folder (rename this from default to checkpoint0 if not already called checkpoint0)
- Result Path, FILE PATH pointing to models .h5 file (Can be located anywhere and can be created by user)
- Annotations Path, FILE PATH pointing to annotations .txt file (use vgg annotator (https://annotate.officialstatistics.org/) and convert .csv to .txt by changing thenendswitch using F2 -> rename DO NOT open in excel and save to .txt)

- Test Record, FILE PATH pointing to models test.record file (Can be located anywhere and can be created by user)
- Train Record, FILE PATH pointing to models train.record file (Can be located anywhere and can be created by user)
- controlled, set to 1 if user needs controlled spilt of train and test data select 0 if random split is OK

# Train page

Learning rate:

The learning rate corresponds to how much each tensor is changed each epoch. Its is recommended to use a value between 1 and 0.1. Setting the learning rate to a larger value will result in a faster convergence but may cause the model to converge on a non-optimal solution. A smaller value will give a slower convergence but will result in a more optimal solution. Of course, this depends greatly on the model and objects you are trying to locate, but this is true for the general case.

It is also recommended to use a variable learning rate. This can be achieved by using a stepped learning rate or cosine decay learning rate. Both options decrease the learning rate as the model progresses. Note that if cosine or another uncontrolled variable learning rate can cause model to stagnate if the initial learning rate is set to low as it will tend to zero and will not recover unless the model is reset

## # of steps

Minimum for the model to properly converge is 150,000 – 200,000 ideal is 300,000+. With less training images the number of steps needed to converge will increase.

## Test & Train Annotations

Input the split train and test annotations. These must be split by image ie: images 1-7 are used for training and 8-10 are for evaluation, test annotations will only contain annotations for images 1-7 and 8-10 only for training annotations. Failure to split annotations properly will result in a sub-optimal convergence.

# Creating Annotations

To create annotations for object detection use https://annotate.officialstatistics.org/.
Steps:
1. Create annotations for all objects in each image
   a. If there is more than one object that needs to be classified, make sure to use multiple attributes and use the proper attribute label for the object.
      i. Ie: Trays and setters if you create a bounding box around a setter make sure in VGG, that the box is linked to the setter attribute label.
2. Export the complete annotations as a csv file.
   a. Top bar -> Annotations -> Export Annotations (as csv)

# Evaluation

Paths are the same as they are in model edit. Save location needs to point to an EXISTING image file of the same size (640x640 for resnet 640).

## Checkpoint number: INT value

The checkpoint that the model should be evaluated at. Commonly this is the most recent checkpoint. See file structure to find where to obtain the checkpoint number from.

## Threshold value: FLOAT (MUST BE BETWEEN 0-1)

This input thresholds the minimum confidence for an object to be classified. Suggested values are between 0.5 and 0.7.  As the threshold value tends to 1 there will be less and less objects classified.

## File structure

Modelname_Folder

    |-------> Pipeline.config FILE

    |-------> ckpt-*(Checkpoint files)

    |-------> checkpoint0 FOLDER

        |-------> checkpoint

        |-------> ckpt-0

    |-------> eval FOLDER

        |-------> events.out.* FILE(s) Filled if eval has been run on model

    |-------> train folder

        |-------> events.out.* FILE(s) Filled at regular intervals

    |-------> Saved_model FOLDER

        |-------> Saved_model.pb FILE

    |-------> Variables FOLDER

        |-------> variables.* FILE(s)

## Downloading of new models

Downloadable models can be found here:

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Steps:

1. Right Click on model name
2. Click "copy link address"
3. Paste into form

## Image shapes

Most of the time users will want to annotate using full-resolution images to be able to accurately define the bounding boxes, and then down-sample the images for object detection training. If original scale images are used there can be an overallocation of resources, resulting in crashes. Due to this, there are fields for Source images (full-resolution) and Train Images (down-sampled) in the GUI. Note that incorrect inputs here will have catastrophic consequences on the detection algorithm, either with crashing or bounding boxes being in incorrect locations.

Example inputs:

- Source images, input field 1 = width
  - input field 2 = height
- Train image, input field = width and height (assumed square image)
  - **** SINGLE VALUE HERE (ie 512x512 image enter just 512)

## First Time Setup

1. Initial Setup Images:
    1. Using a camera in the same position as you intend to be detecting from, take pictures of multiple different tray setups (different setters in different orientations).
        1. The more images the better detection result
    2. Keeping the images at full resolution, annotate using the annotation web tool defined above.
        1. Keep the boxes you annotate with as close to the edge of the part as possible to ensure the detection found boxes are tight to their physical location.
    3. Edit/create the lablemap.pbtxt to include ONLY the labels used for annotations.
    5. If a controlled split is wanted:
        1. Split the annotations data into separate train and test annotations files.
    6. Copy all images into a sperate folder
        2. Set image resize to the dimensions that the detection model will use
            1. For Example: Resnet640 use (640,640)
            2. Centernet512 use (512,512)
2. Using the detection GUI download a model.
3. Starting from the top of the left-hand bar select Model Edit and fill out all boxes.
    1. See above for help.
4. Press next and continue to Pipeline Editor.
    1. See above for help
5. Press next and continue to Train Inputs.
    1. See above for help
6. Press load model
    1. If successful, no error will be displayed in the top error bar
7. Go to Run Model and press run
8. The model will now be training in the background.
    1. To verify this use NVTOP (linux only) and look at GPU memory and clock speed. These should both be close to maximum values.
        1. It is possible that the GPU is NOT used for training due to tensorflow being unable to locate appropriate GPU drivers. In this case the CPU will be utilized. Check CPU usage to verify.
    1. Windows users use task manager to see CPU/GPU usage
    2. Another method is to monitor RAM usage.
        1. Ram usage will slowly increase and then stabilize after
    3. Crashes are common in the beginning of the model run due to overallocation of resources.
        1. To mitigate the crashes:
            1. Use a model with a lower image resolution.
            2. Change the batch size to half of its current value.

## Setting up Programming Environment (linux)

1. Create new virtual environment with python 3.8.10
    1. Other Python versions may work but have not been tested. Check tensorflow documentation for list of compatible python versions.
2. Ensure the latest pip version is installed

3. Run apt-get install -y protobuf-compiler python-pil python-lxml
4. Pip install numpy and cython
5. With the virtual environment activated run pip install –r on requirements.txt
6. Run apt-get update && apt-get install ffmpeg libsm6 libxext6 -y
7. Clone the tensorflow repository into a known location
    1. It's suggested to create a folder called tensorflow/models and clone into there
8. Navigate to the cloned repository and cd to the research folder
    1. If using folder naming as defined in 7.1 this folder can be found at tensorflow/models/research
9. Run protoc object_detection/protos/*.proto --python_out=.
10. Run cp object_detection/packages/tf2/setup.py .
11. Run pip install .
12. To utilize a GPU for training follow instructions found here: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#gpu-support-optional
    1. Ensure to follow the instructions for the GPU installed on your machine. Model training will not be done on GPU if incorrect drivers, driver versions, or cudnn versions are installed.