**Dog Behavior Evaluation Using CNN**

**The system**

The system evaluates the dog’s proximity to a moving / stationary toy.

Whenever a dog or a toy are identified, the system draws boxes around them.

Upon an interaction between the dog and the toy, the system adds a new line in an output file (CSV).

Interaction is defined, as whenever a dog surrounding box overlaps the toy box, or if the distance between the dog’s and the toy’s center of mass is less than 20% of the image width.

**Usage:**

In order to use the system, run NN\_dogs\_behavior\_eval.exe via cmd (No need for python nor tensorflow installation).

Input:

-o: Path to output csv file. Note that the system will generate the file.

-v: Path to the desired video to evaluate.

-g: Path to the trained NN graph.

-l: Path to the label map, which the graph is trained for.

Example for usage (in cmd):

*NN\_dogs\_behavior\_eval.exe  -o csv\_new.csv -g dogs\_toys\_graph2/frozen\_inference\_graph.pb -l data/object-detection.pbtxt -v "\_Alisha Harel TA on 29-12-2017 at 9.08 .mov"*

**The trained graph and label map:**

Our trained graph is “frozen\_inference\_graph.pb” which is located in dogs\_toys\_graph2 folder. The graph is a retrained mobilenet CNN graph.

The label map is named object\_detection.pbtxt and is located in data folder

[Link to mobilenet](https://drive.google.com/open?id=1fWv6fZUplvzT9lyvY8ePyOyTFU-KiFqF).

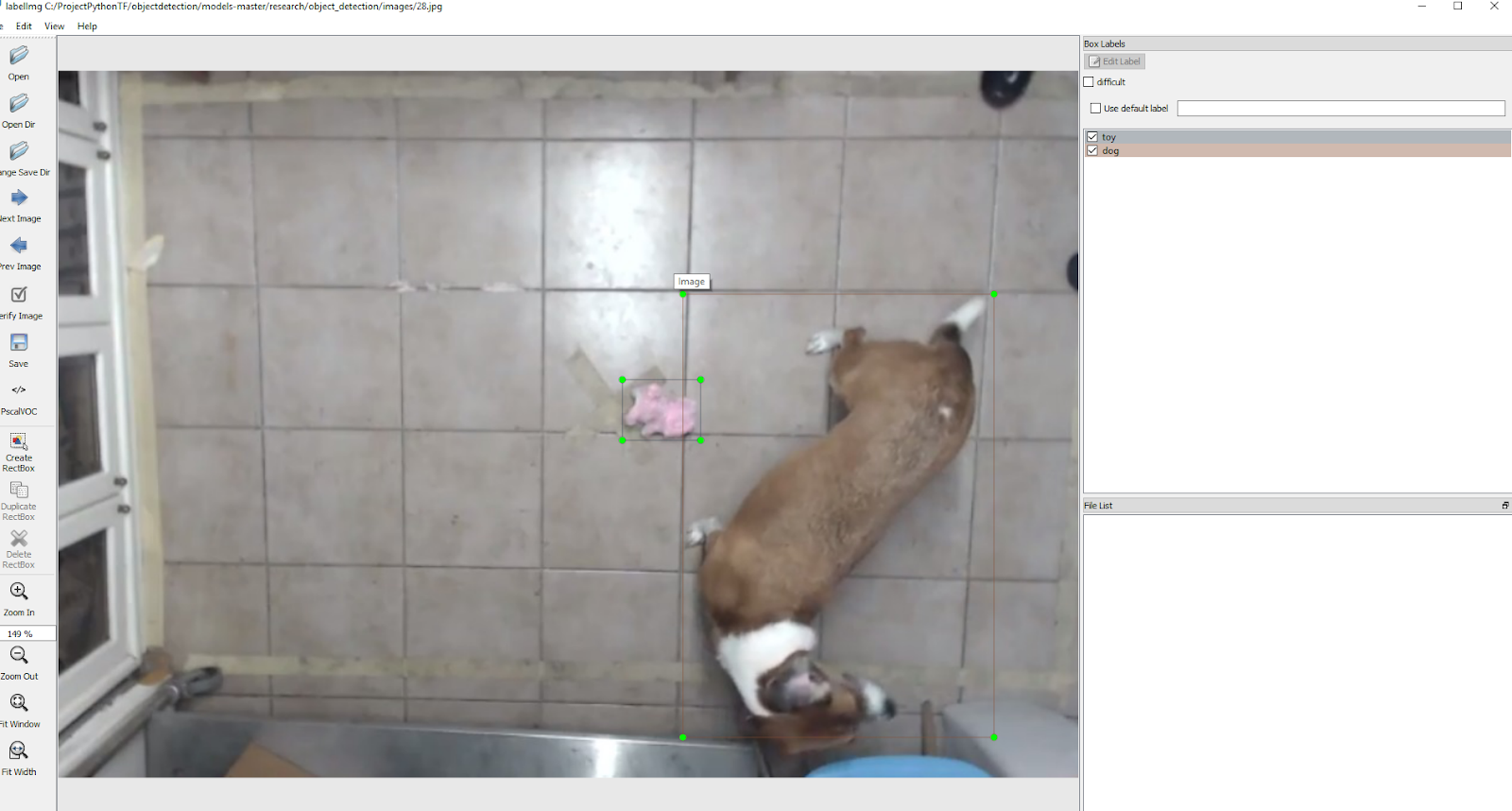
**Note: The system can use a different graph. You will need to train a new one and give it and its label map as an input to the system. Instructions about how to train a new graph is given in the following section.**

**How to create and train a graph for the object detection model**

**Prerequisites:**

**TensorFlow installation guide AT**: <https://github.com/tensorflow/models/blob/master/research/object_detection/README.md>

**STEP 1:** Create CSV files from images

* Use images with relatively small resolutions and with the same file type (like .jpg).
* Put those images in the same directory and name it “images”.
* Using labeling application, label at least 100 images for optimal results. you will receive XML files of the images you labeled. Here’s an example of labeling app usage:
* Inside the “images” directory Make a “train” and a “test” directories.

copy about 10% of the images alongside with their matching XML files

Into “test” and the other 90% images with their XML files into “train”.

* Convert the XML files within “test” and “train” into CSV files using xml\_to\_csv.py.
* You should now have 2 csv files, 1 for the “test” directory and 1 for the “train” directory.

**STEP 2:** Create TFrecords

TensorFlow doesn’t take csv files as an input, it needs record files to train the model.

* Create “data” and “training” directories.
* Use generate\_tfrecord.py to create the TFrecords. You will need to change the script labels according to the labels you choose during Step 1.
* How to exactly use the mentioned script is commented at the top of the script.
* You should now have 2 record files: train.record and test.record.

**STEP 3:** Train the model

* Decide what trained model you want to use. Take notice of the tradeoff between the detection speed and accuracy. The higher the speed the lower the accuracy and vice versa.

(we used the ssd\_mobilenet\_v1\_coco\_2017\_11\_17 model)

* Download the config file for the model you chose.
* Make a file named object-detection.pbtxt which looks similar to this:

item {

id: 1

name: 'dog' }

item {

id: 2

name: 'toy' }

* In the .config file you downloaded, change the num\_classes to the number of classes you chose to detect, in our case it is num\_classes: 2.
* According to your GPU you can decide whether to increase or decrease the batch\_size. If you got a strong GPU, you should consider increasing the batch\_size.
* Create path to the downloaded model like this:

fine\_tune\_checkpoint: ssd\_mobilenet\_v1\_coco\_2017\_11\_17/model.ckpt

* Create path to train and test record files like this:

train\_input\_reader: {    
tf\_record\_input\_reader {     
input\_path: "data/train.record" }  
label\_map\_path: "data/object-detection.pbtxt" }

eval\_input\_reader: {    
tf\_record\_input\_reader {  
input\_path: "data/test.record"  }  
label\_map\_path: "data/object-detection.pbtxt"    
shuffle: false  
num\_readers: 1}

* Use train.py script with this command:

python train.py --logtostderr --train\_dir=training/ --pipeline\_config\_path=training/ ssd\_mobilenet\_v1\_coco\_2017\_11\_17.config.

you should wait until your loss is below 1.

**STEP 4:** Export your Graph and try it out.

* Use the export\_inference\_graph.py with the following command:

python export\_inference\_graph

   --input\_type image\_tensor

--pipeline\_config\_path training/ssd\_mobilenet\_v1\_coco\_2017\_11\_17.config                                                 --trained\_checkpoint\_prefix training/model-numyouhave.ckpt

   --output\_directory exported\_model\_directory\_name

You have now exported your inference graph.

* Finally, Use the NN\_dogs\_behavioural\_evaluation.py with the graph you have just created along with the label\_path and video\_path using the following command:

python NN\_dogs\_behavioural\_evaluation.py -o csv\_new.csv -g exported\_model\_directory\_name/frozen\_inference\_graph.pb -v vid.mov -l data/object-detection.pbtxt

[Link to our datasets](https://drive.google.com/open?id=1xmDkrxHpvtAbNqQ5HIPVzRgihrDDFDJ5)

**Good luck!**