

# Building a Caffe Workflow



Mia Polansky

Rice University RECG

# Overview

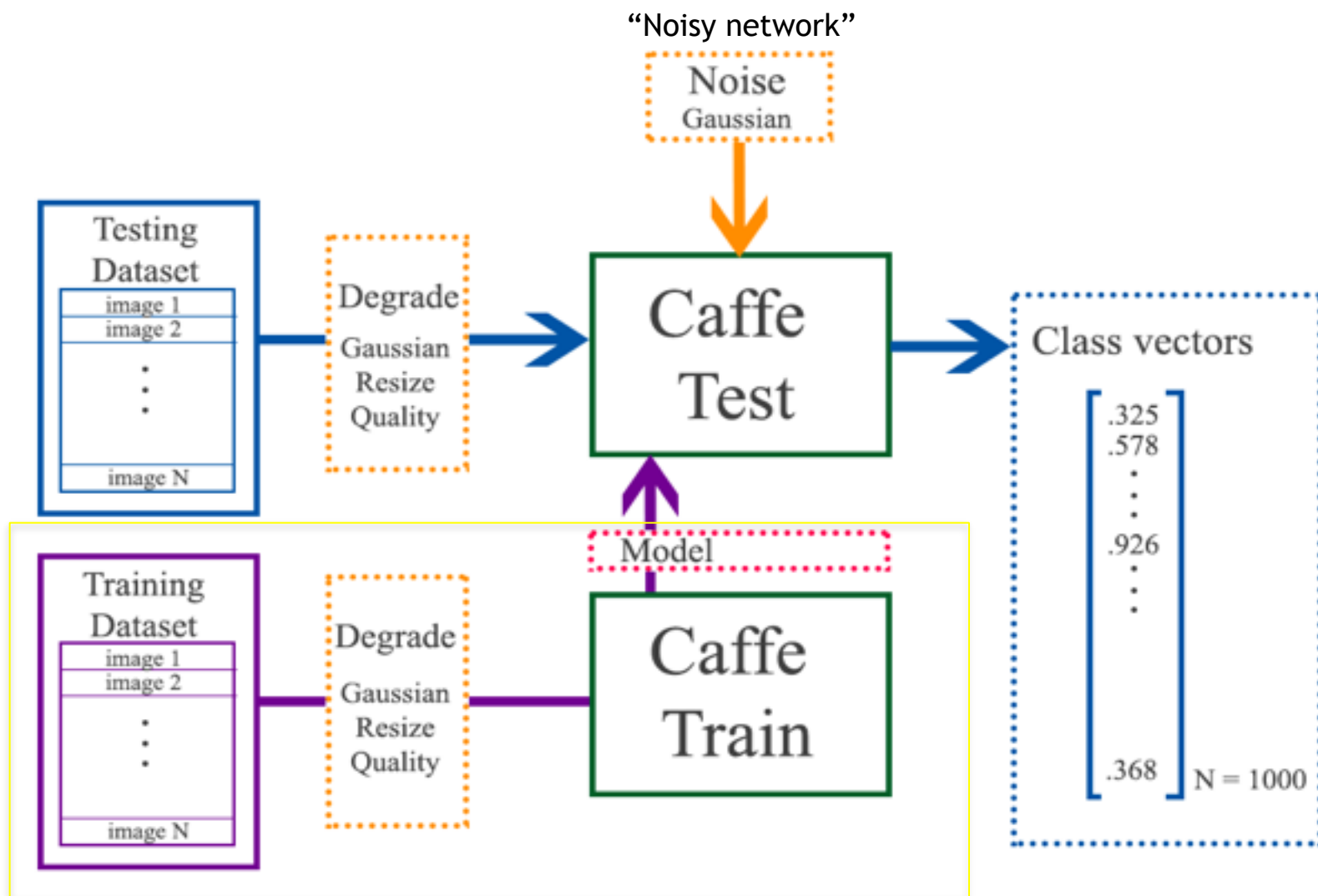
---

Motivation: To create a workflow that will allow for streamlined caffe testing and generation of statistics from the testing results

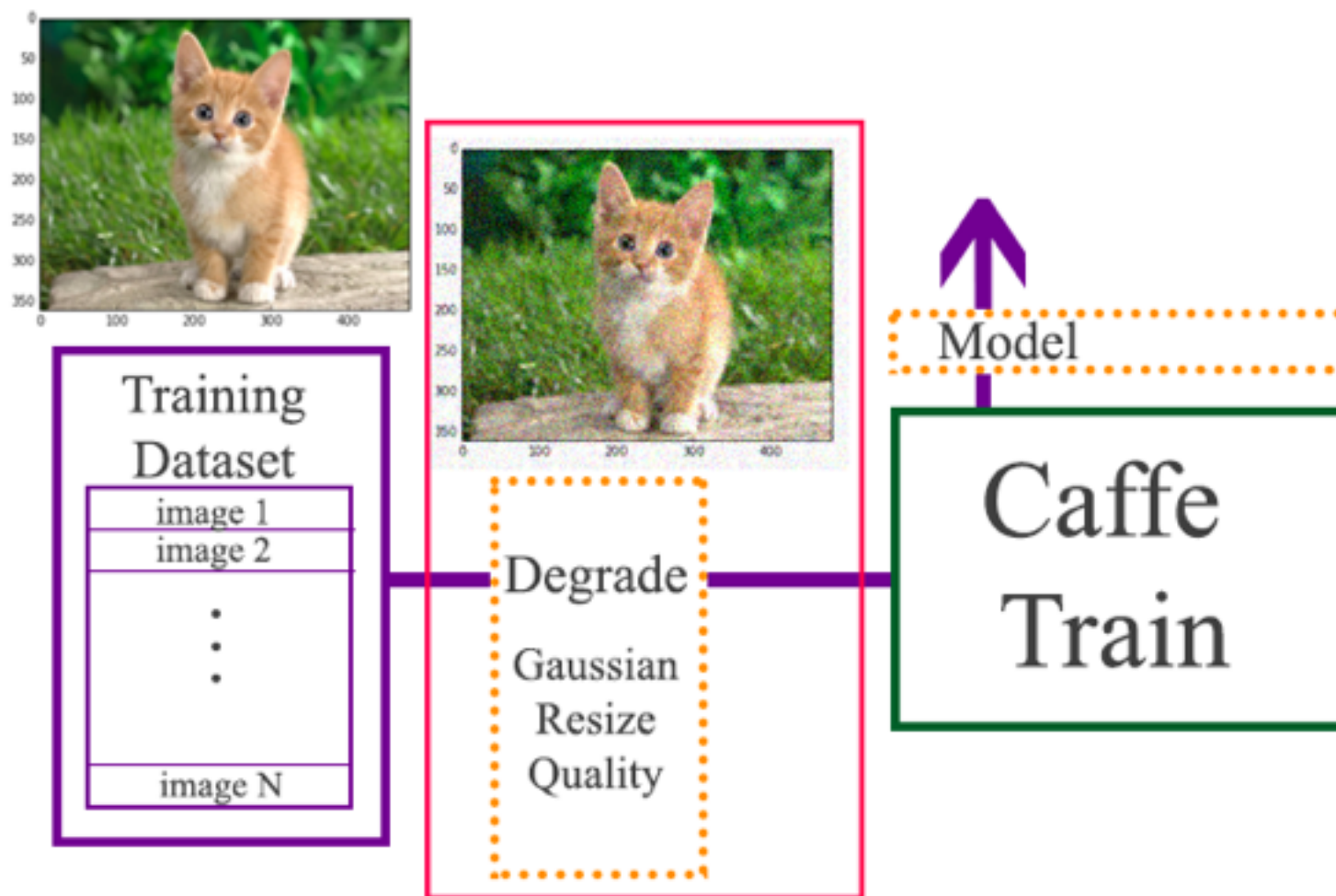
## Outline:

- Caffe Refresher
- Overview of the desired workflow
- Automation of testing runs
- Creation of a testing GUI
- Generating statistics

# Caffe Refresher: CaffeTrain

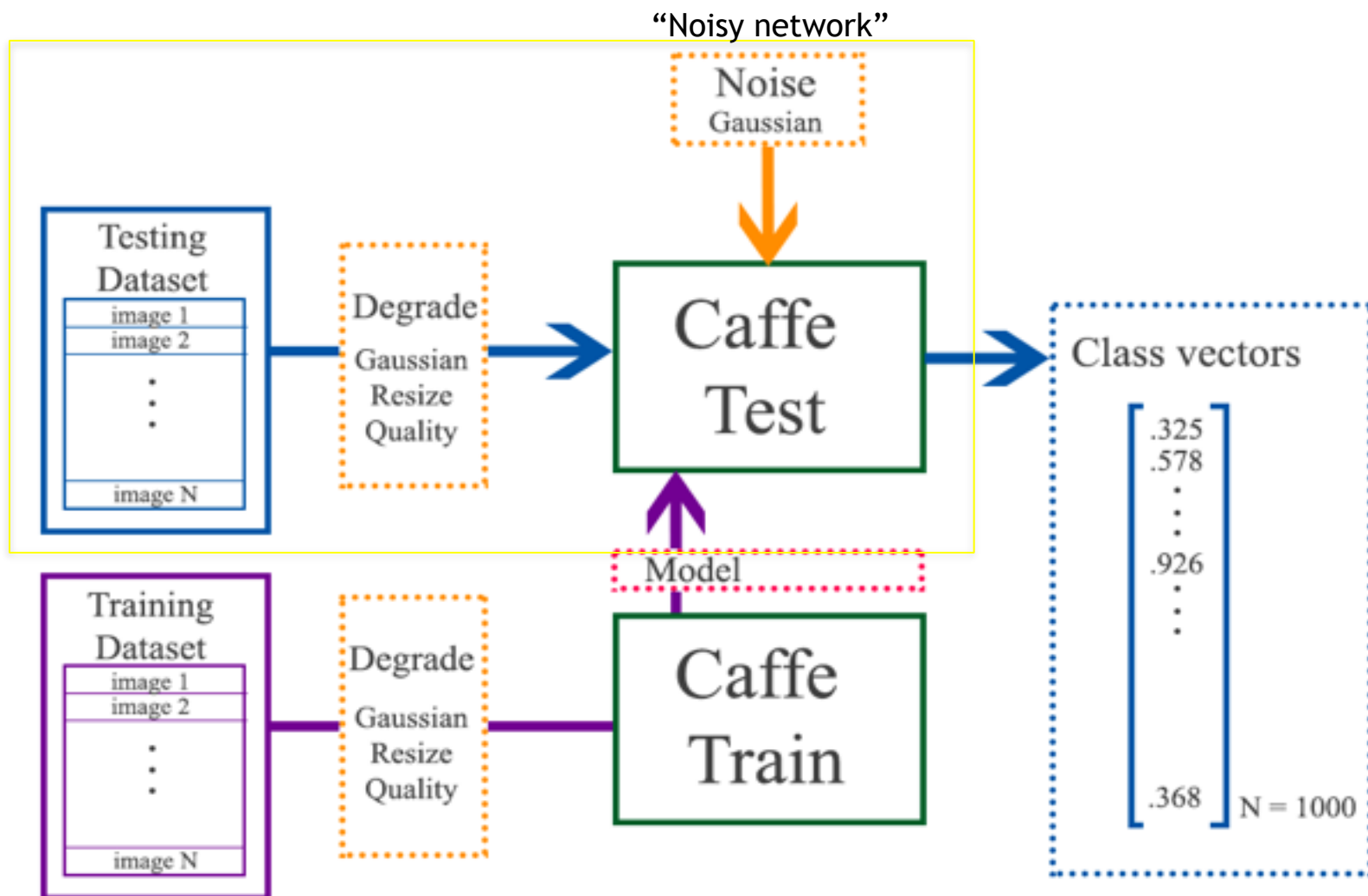


# Caffe Refresher: CaffeTrain

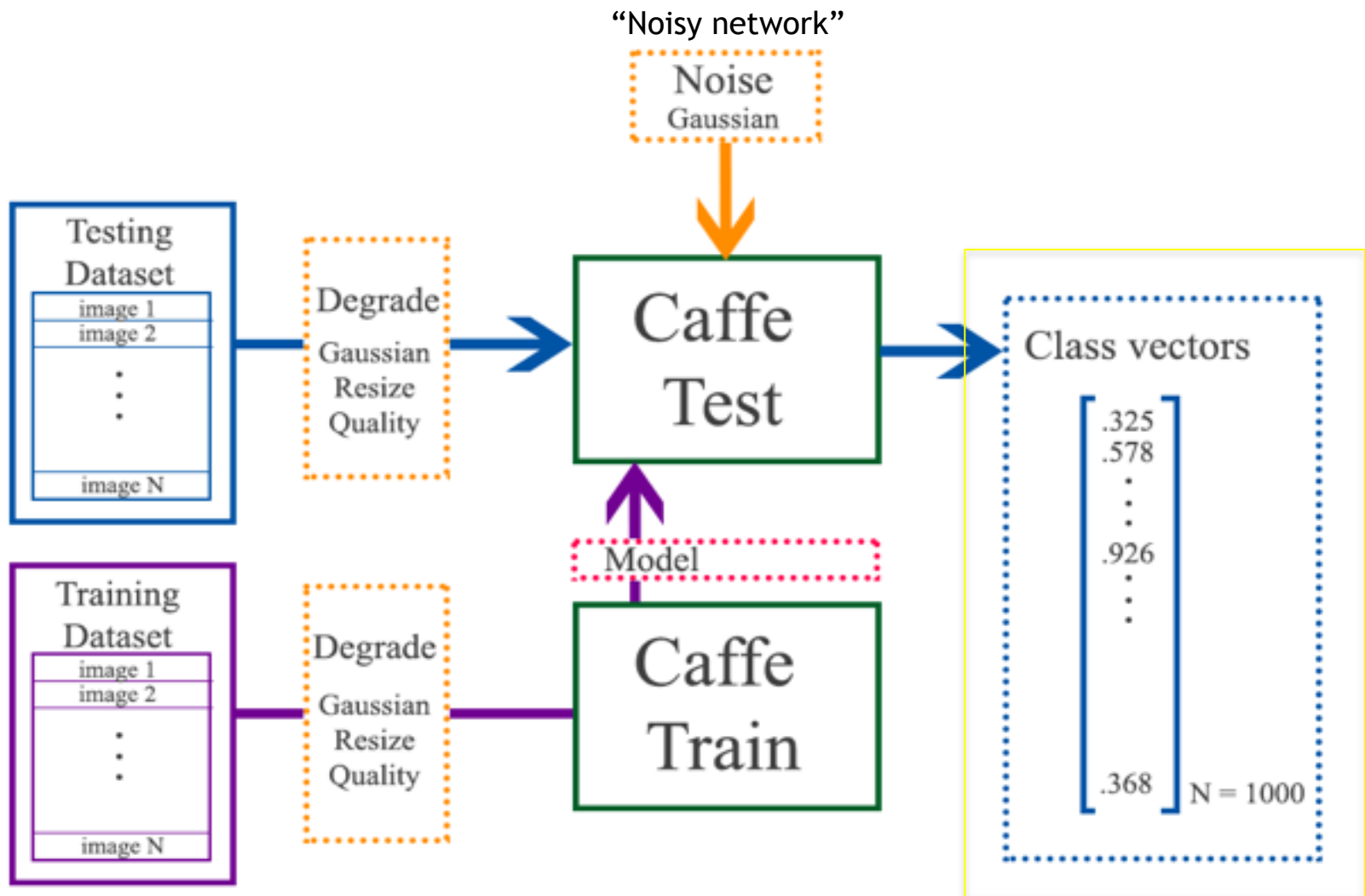


This produces the degraded model

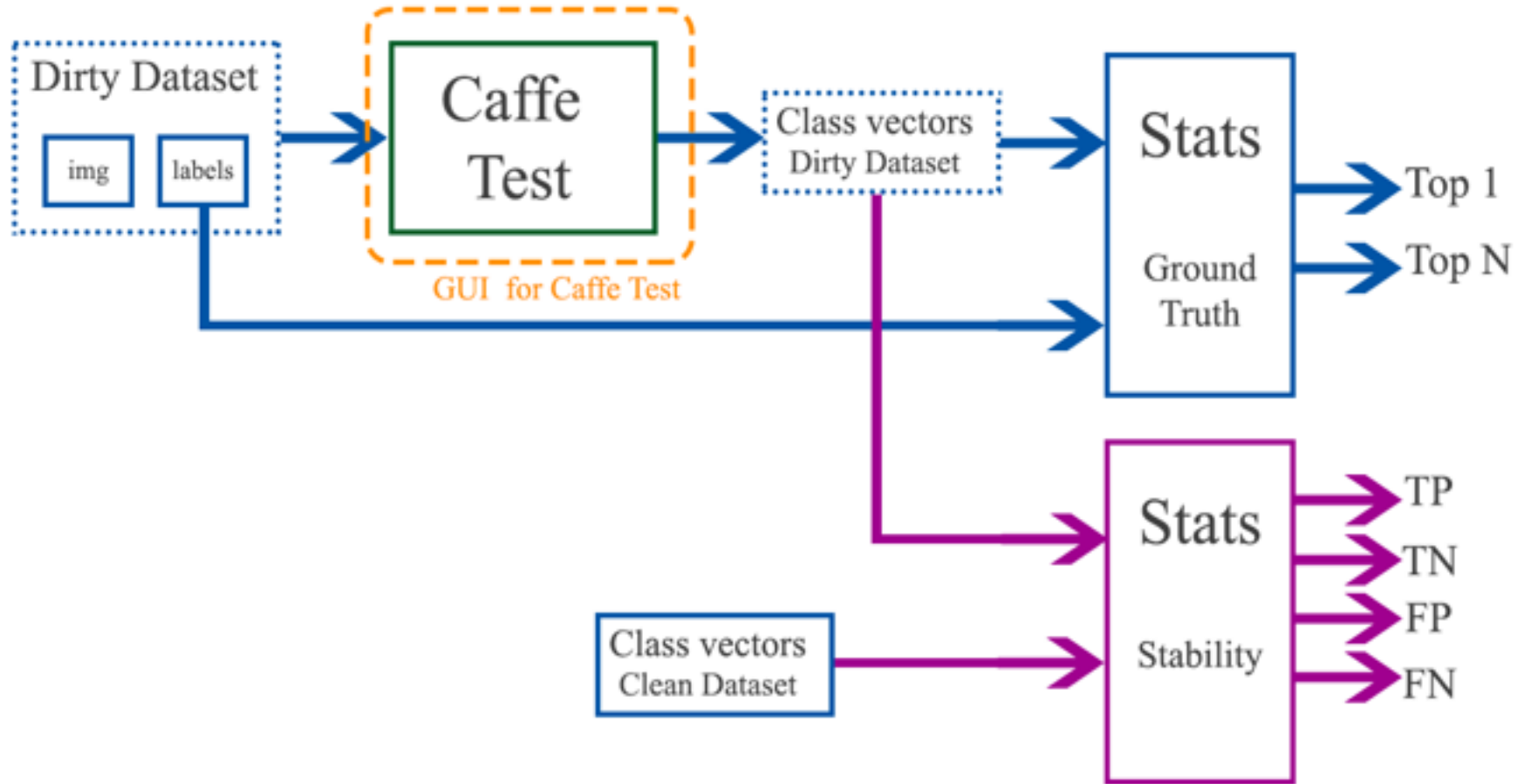
# Caffe Refresher: The Output



# Caffe Refresher



# CaffeTab Workflow Overview



# Testing Parameters

---

- Many parameters needed for testing:
  - General parameters
  - Run parameters
  - Degradation parameters



# CaffeTab Workflow (1): Multiple Runs in a Row

- Need to run the Caffe Test for 500 or more folders



```
275 runs = 500
276 numBatchesPerRun = 1
277 numImagesInBatchFolder=100
278
279 for i in range(runs):
280     #startingBatch = i * numPerBatch;
281     new3 = CaffeTester(croot, iroot, 'googlenet', '', i, numBatchesPerRun, numImagesInBatchFolder, 'new_gaussian13')
282     new3.degrade(['resize'], .1, 5, .15, 5, i, 0.5)
283     #new3.degrade(['clean'], .001, 1, 1, 10, i, 0.5)
284
```

# CaffeTab Workflow (2): A Testing GUI

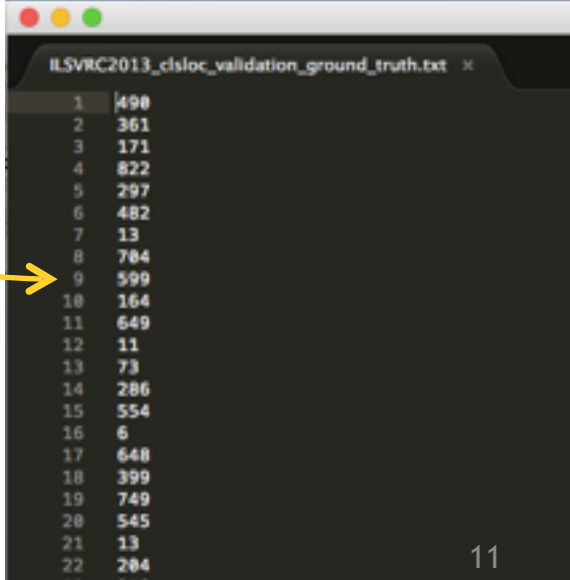
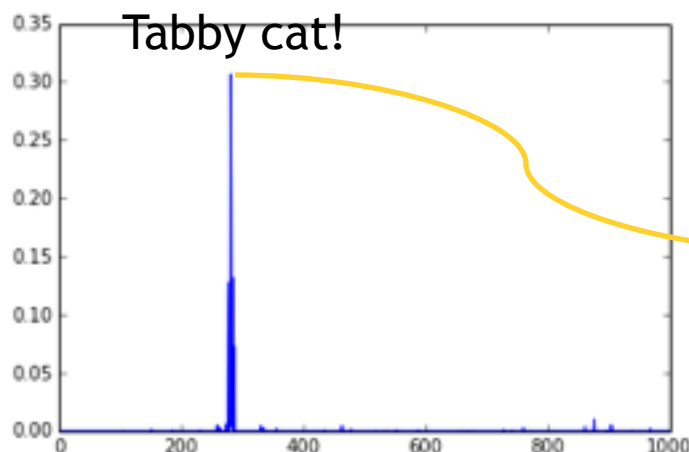
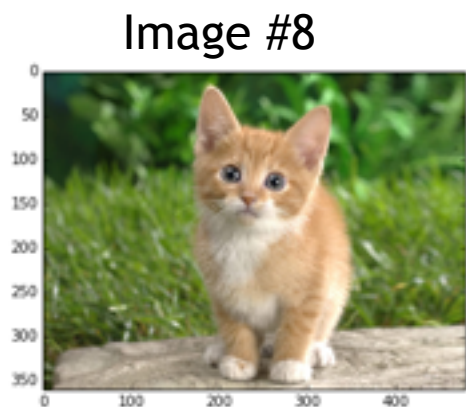
```
1 import sys
2 import Tkinter as tk
3 import subprocess
4 import testmodule
5 croot = '../caffe/'
6 #iroot = '/home/roblkw/Work/caffe/work'
7 iroot = '/hard2/ILSVRC2014_img_val'
8
9 class MyOptionsMenu(tk.OptionMenu):
10     def __init__(self, master, status, *options):
11         self.var = tk.StringVar(master)
12         self.var.set(status)
13         tk.OptionMenu.__init__(self, master, self.var, *options)
14         self.config(
15             font=('calibri', (10)), bg='white',
16             self['menu'].config(font=('calibri', (10)), bg='white', fg='dark blue')
17
18         @tk.Entry.__init__(parent, textvar=self.var)
19
20     def callback(self):
21         val = '{}'.format(self.var.get())
22         print(val)
23         # subprocess.call([val])
24
25
26 def submit_data2():
27     val = mymenu2.get()
28     print val
29
30 def submit_data3():
31     val = mymenu3.get()
32     print val
33
34 def submit_data4():
35     val = mymenu4.get()
36     print val
37
38 def submit_data5():
39     val = mymenu5.get()
40     print val
```

The screenshot shows a window titled "TestmoduleGUI" with a light blue background. It contains several sections of controls:

- Model Parameters:** Includes a "Base Model" dropdown menu (currently showing "Select base model") and a "Log Name (name of log file to output the test results)" text input field.
- Run Parameters:** Contains three text input fields labeled "Number of Batches per Run", "Total Runs", and "Number of Images in Batch Folder".
- Degradation Parameters:** Includes a "Degradation Type" dropdown menu (currently showing "Select deg type"), an "Initial Degradation" text input field, and two more text input fields labeled "Number of Steps" and "Step Size".
- Top N (Top N values to select):** A text input field at the bottom left.
- Run Button:** A button labeled "Run Testmodule.py" at the bottom center.

# Analyzing the Results

- Caffe outputs 1000 x 1 class vectors that contain the probability of each class
- These are “guesses” → We need to compare against the ground truth



Index	Ground Truth Label
1	490
2	361
3	171
4	822
5	297
6	482
7	13
8	784
9	599
10	164
11	649
12	11
13	73
14	286
15	554
16	6
17	648
18	399
19	749
20	545
21	13
22	284
23	310

# Mapping the Ground Truth Data

ILSVRC2013\_clsloc\_validation\_ground\_truth.txt

1	498
2	361
3	171
4	822
5	297
6	482
7	13
8	784
9	599
10	164
11	649
12	11
13	73
14	286
15	554
16	6
17	648
18	399
19	749
20	545
21	13

Matlab  
Image Labels

Matlab Numbering

synsets

1x1860 struct with 8 fields

Fields	ILSVRC2014_ID	WNID	words	gloss
1	1	'n021197...	'kit fox, ...	'small gr...
2	2	'n021007...	'English s...	'an Englis...
3	3	'n021101...	'Siberian ...	'breed of ...
4	4	'n020962...	'Australia...	'small gr...
5	5	'n021020...	'English s...	'a breed ...
6	6	'n020662...	'grey wha...	'medium...
7	7	'n025098...	'lesser pa...	'reddish...
8	8	'n021240...	'Egyptian...	'a domes...

synset\_words.txt

1	n05448764	tench, Tinca tinca
2	n05443537	goldfish, Carassius auratus
3	n0484858	great white shark, white shark, man-eater, man-eating shark, Ca
4	n04891361	tiger shark, Galeorhinus galeus
5	n0494475	hammerhead, hammerhead shark
6	n0496331	electric ray, crampfish, numbfish, torpedo
7	n0498041	stingray
8	n0514668	cock
9	n0514859	hen
10	n0518878	ostrich, Struthio camelus
11	n0518879	brambling, Fringilla montifringilla
12	n0518880	goldfinch, Carduelis carduelis
13	n0518881	house finch, linnet, Carpodacus mexicanus
14	n0518882	juncos, snowbird
15	n0518883	indigo bunting, indigo finch, indigo bird, Passerina cyanea
16	n0518884	robin, American robin, Turdus migratorius
17	n0518885	bulbul
18	n0518886	jay
19	n0518887	magpie
20	n0518888	chickadee
21	n0518889	water ouzel, dipper
22	n0518890	kite
23	n0518891	bald eagle, American eagle, Haliaeetus leucocephalus

Caffe Numbering

A	B	C	D	E	F	G
Caffe	Caffe #	MATLAB 2014	MATLAB 2014 #	FieldName	Caffe #	MATLAB #
n02119764	0	n02119764	440	n02119764	278	2
n02119764	1	n02119764	450	n02119764	282	3
n02119764	2	n02119764	460	n02119764	290	4
n02119764	3	n02119764	470	n02119764	300	5
n02119764	4	n02119764	480	n02119764	310	6
n02119764	5	n02119764	490	n02119764	320	7
n02119764	6	n02119764	500	n02119764	330	8
n02119764	7	n02119764	510	n02119764	340	9
n02119764	8	n02119764	520	n02119764	350	10
n02119764	9	n02119764	530	n02119764	360	11
n02119764	10	n02119764	540	n02119764	370	12
n02119764	11	n02119764	550	n02119764	380	13
n02119764	12	n02119764	560	n02119764	390	14
n02119764	13	n02119764	570	n02119764	400	15
n02119764	14	n02119764	580	n02119764	410	16
n02119764	15	n02119764	590	n02119764	420	17
n02119764	16	n02119764	600	n02119764	430	18
n02119764	17	n02119764	610	n02119764	440	19
n02119764	18	n02119764	620	n02119764	450	20
n02119764	19	n02119764	630	n02119764	460	21
n02119764	20	n02119764	640	n02119764	470	22

Caffe Image Labels

# Comparing against the Ground Truth

Compares the class vectors and finds “hits”:

- Is the correct class same as top predicted class?
- Is the correct class within top N predicted classes?

```
import csv
import numpy as np
import glob
import os
from sys import argv
import argparse

script, num_batches, max_deg_level, img_per_batch, deg_type, toprint = argv

gt = np.loadtxt("Ground_Truth_%s.txt" % deg_type)

def f(a,N):
    return np.argsort(a)[::-1][:N]

#top_1 = np.argmax(gt)

#max_deg_level = 1
#num_batches = 500
#img_per_batch = 100

match5 = np.zeros(int(max_deg_level)+1)
match1 = np.zeros(int(max_deg_level)+1)

for deg_lvl in range(0,int(max_deg_level)+1): #+1?
    for batch in range(0,int(num_batches)):
        #compare1 = np.load("/Users/miaipensky/caffe_exp/classvec_2013/googlenet_"+str(batch)+"_"+str(deg_lvl)+"_"+str(batch)+".npy")
        compare1 = np.load("/home/roblkw/caffe_exp/classvec_2013/googlenet_"+str(batch)+"_"+str(deg_lvl)+"_"+str(batch)+".npy")
        #print compare1.shape
        for img in range(0,int(img_per_batch)):
            top_5 = f(compare1[img,:],5)
            #print top_5
            top_1 = np.argmax(compare1[img,:])

            if gt[img*100+batch] == top_1:
                match1[deg_lvl] += 1

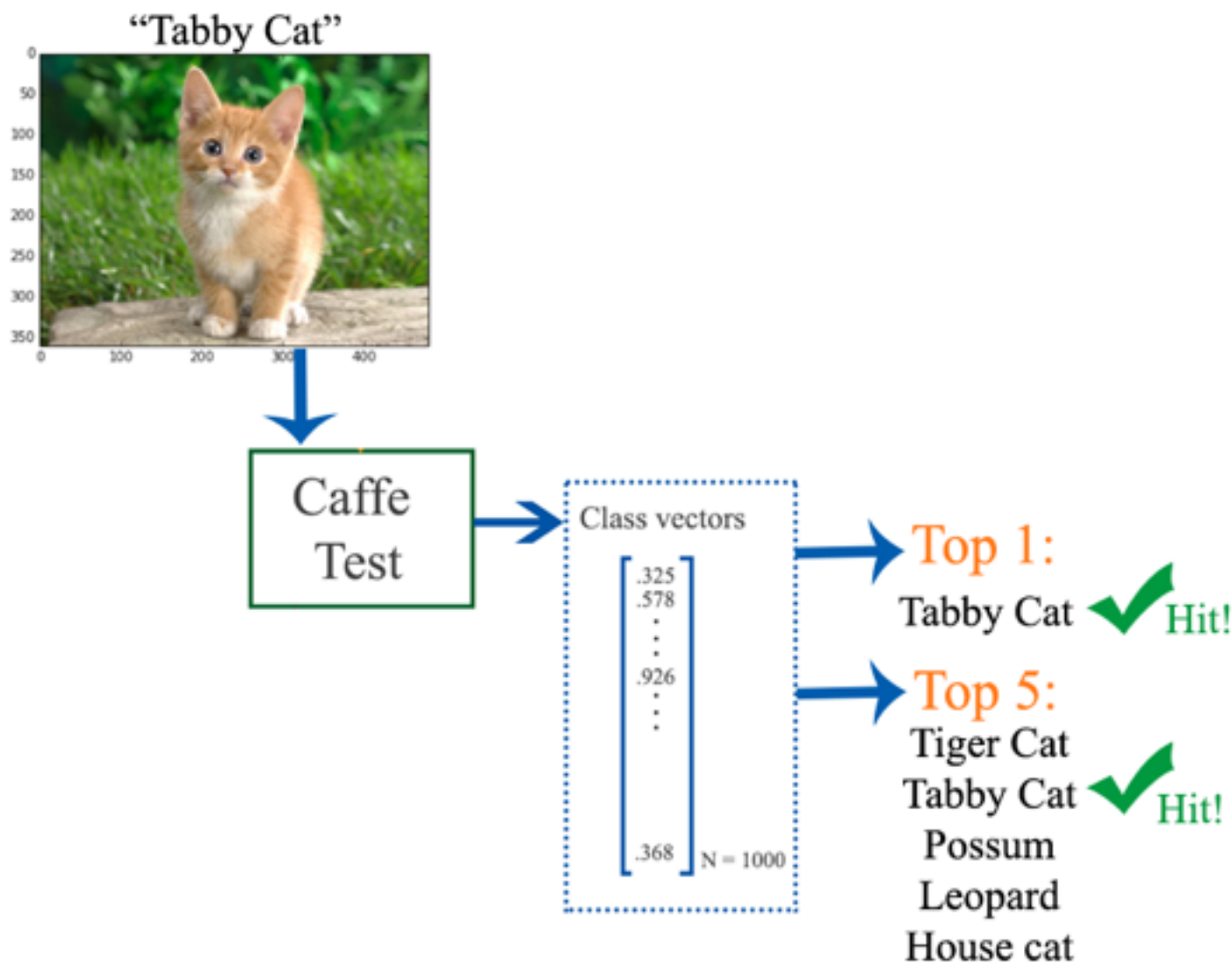
            for num in range(0,5):
                if gt[img*100+batch] == top_5[num]:
                    match5[deg_lvl] += 1

np.savetxt("top_5_hit_rate_"+deg_type+"_"+max_deg_level+".csv", match5/float(500), delimiter=",")
np.savetxt("top_1_hit_rate_"+deg_type+"_"+max_deg_level+".csv", match1/float(500), delimiter=",")

#print "top 5:", match5/float(int(img_per_batch)*int(num_batches)), '\n', "top 1:", match1/float(int(img_per_batch)*int(num_batches))

if toprint.lower() == "true":
    print "top 5:", match5/float(int(img_per_batch)*int(num_batches)), '\n', "top 1:", match1/float(int(img_per_batch)*int(num_batches))
```

# Comparing against the Ground Truth



# Simplifying the Process

- Still tedious: need to manually input # batches, max degradation level, images per batch, degradation type, and toggle printing results
- → Use argv to define parameters from command line

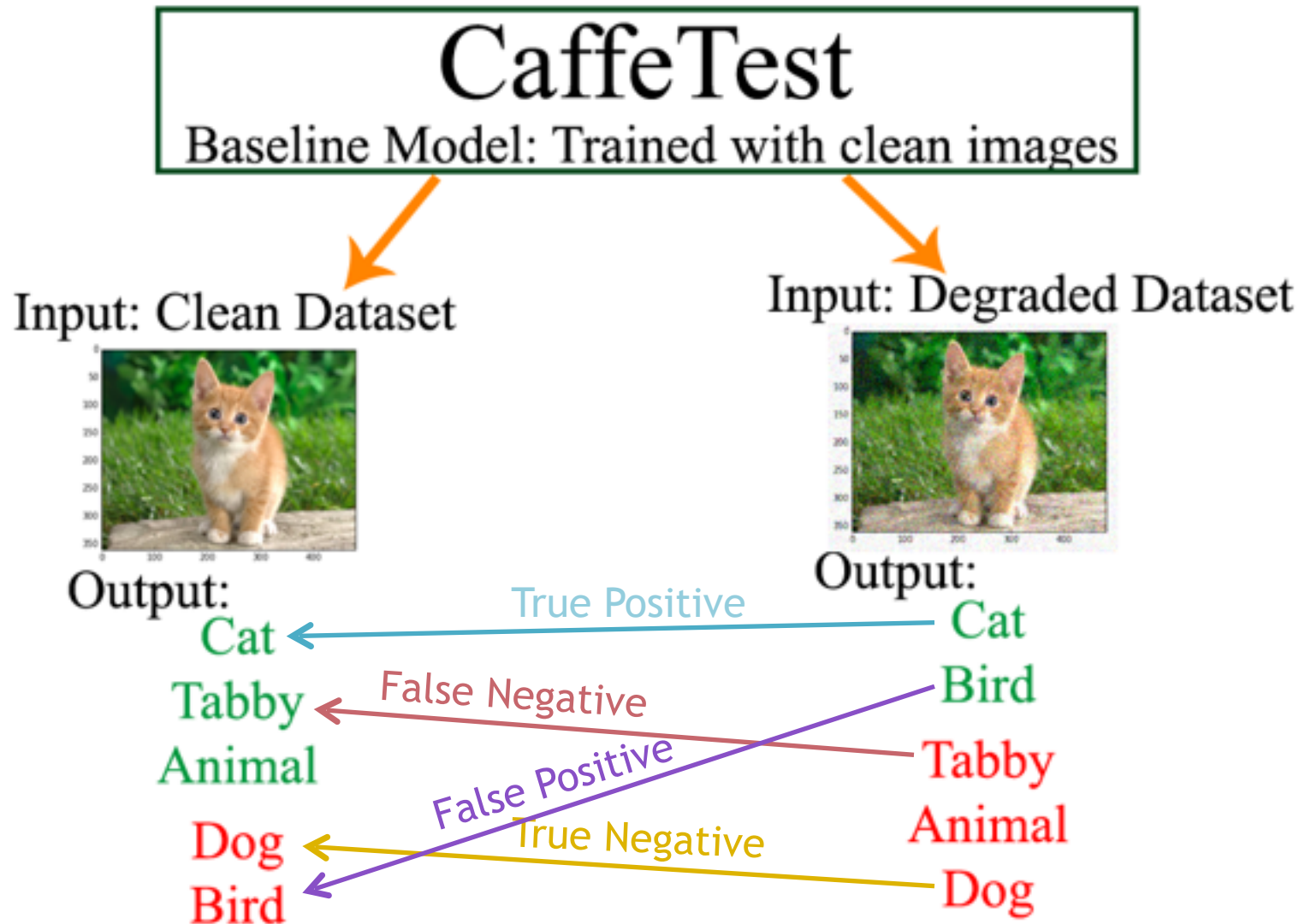
```
Ground_Truth_Stats.py x
1  #Instructions for running the script:
2  #This script runs on terminal and takes in 6 arguments (5 if you don't include the file name)
3  #The arguments are as follows:
4      # 1) filename.py --> Ground_Truth_Stats.py
5      # 2) num_batches --> Number of batches being analyzed (corresponds to largest first/third number of numpy array file names)
6      # 3) max_deg_level --> The maximum degradation level (corresponds to largest second number of numpy array file names)
7      # 4) img_per_batch --> number of images per batch
8      # 5) deg_type --> degradation type
9      # 5) toprint --> Whether to print results; either true or false
10 # Example:
11 # miapolansky$ Ground_Truth_Stats.py 500 0 100 Gaussian True
12
```

```
Mia:~ miapolansky$ cd caffe/caffe_exp/tools
```

```
Mia:tools miapolansky$ python Ground_Truth_Stats.py 500 0 100 Gaussian True
```



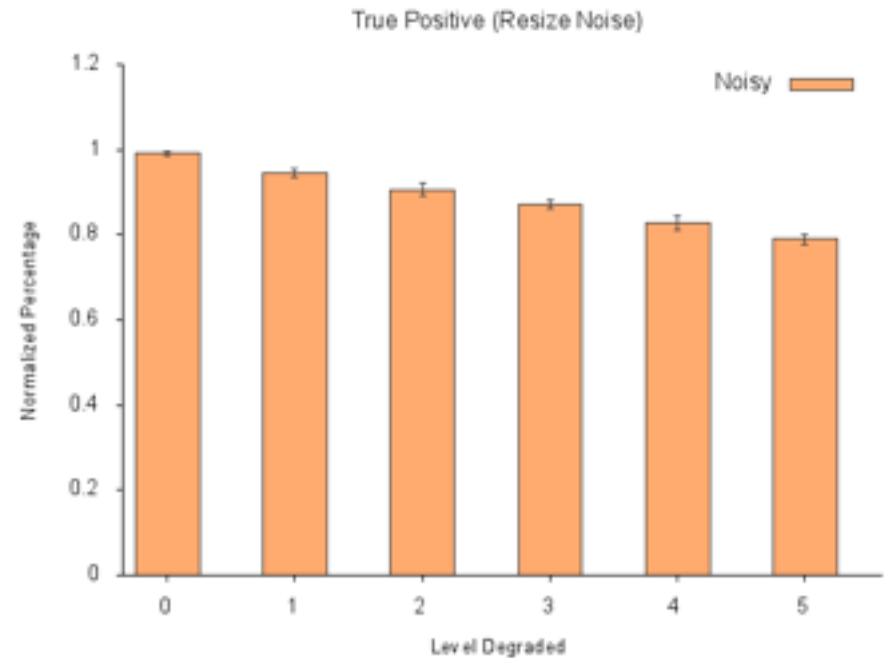
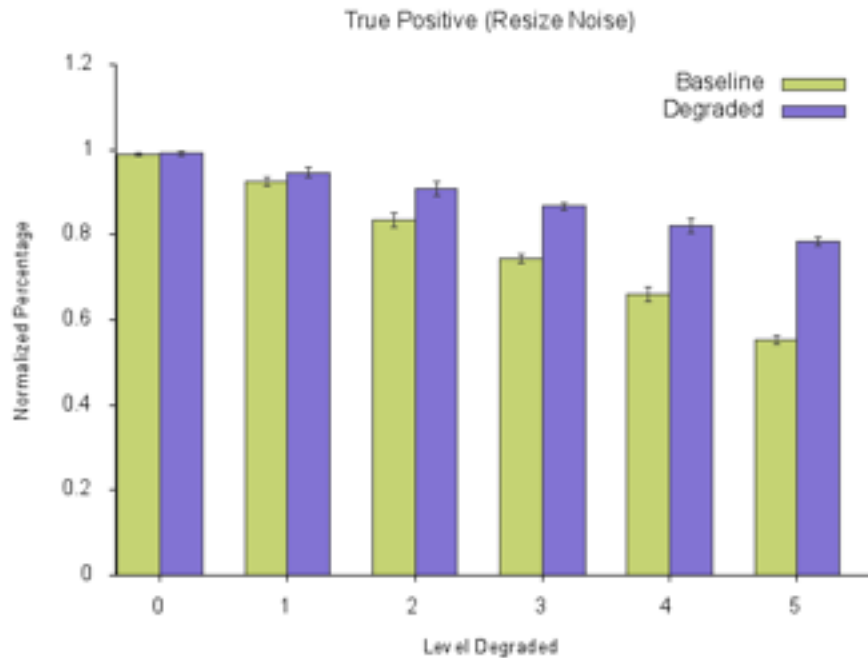
# Calculating TP, TN, FP, FN





# Visualizing the Results

- Afterwards, graph data with gnuplot



Baseline: Model trained on clean dataset, run on clean network

Degraded: Model trained on degraded dataset, run of clean network

Noisy: Model trained on noisy network and degraded dataset, run on noisy network