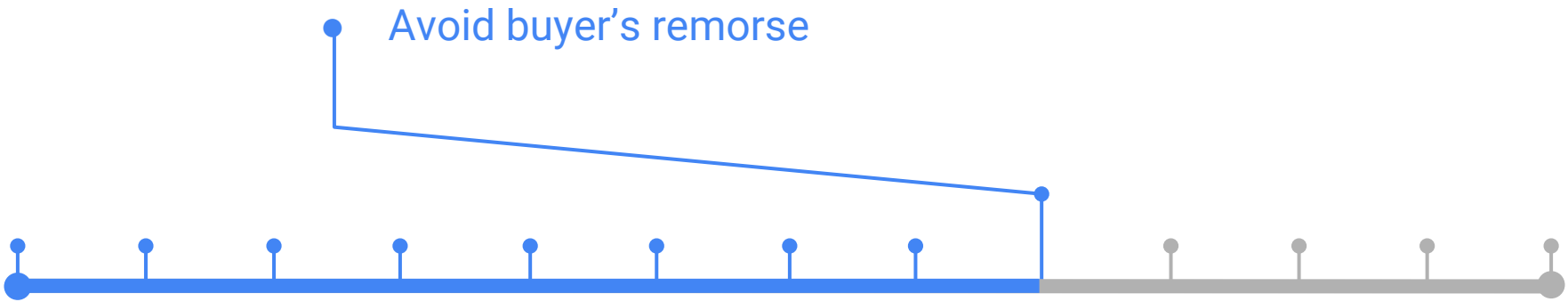
Deep Learning Walkthrough - 08

Code in github.com/google-aai/sc17

Cassie Kozyrkov
Chief Decision Scientist, Google Cloud
GitHub: kozyrkov; Twitter: @quaesita

Google Cloud

Step 8 | Validate your model



Chief actors:

Analyst

What if you overfit?

You'll be oblivious to overfitting unless you check performance on fresh data.

Use validation

How?

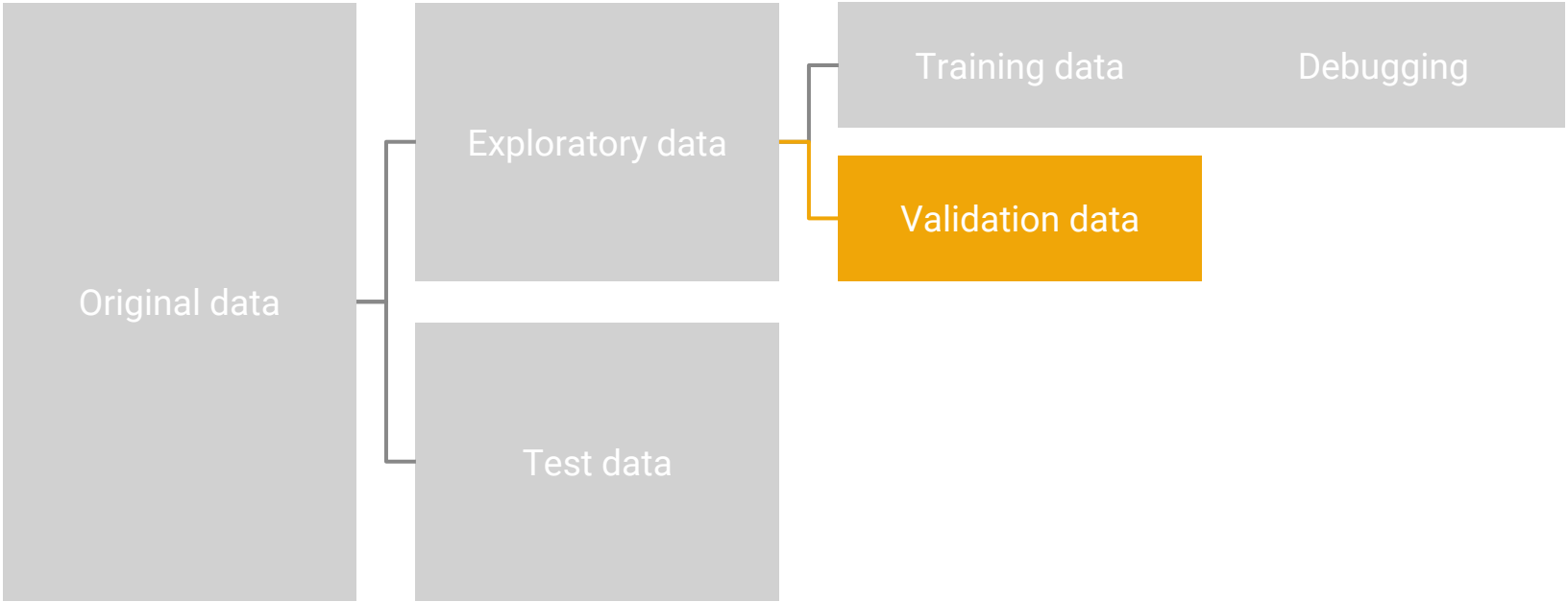
Simply evaluate your model on a different dataset from the one you trained on

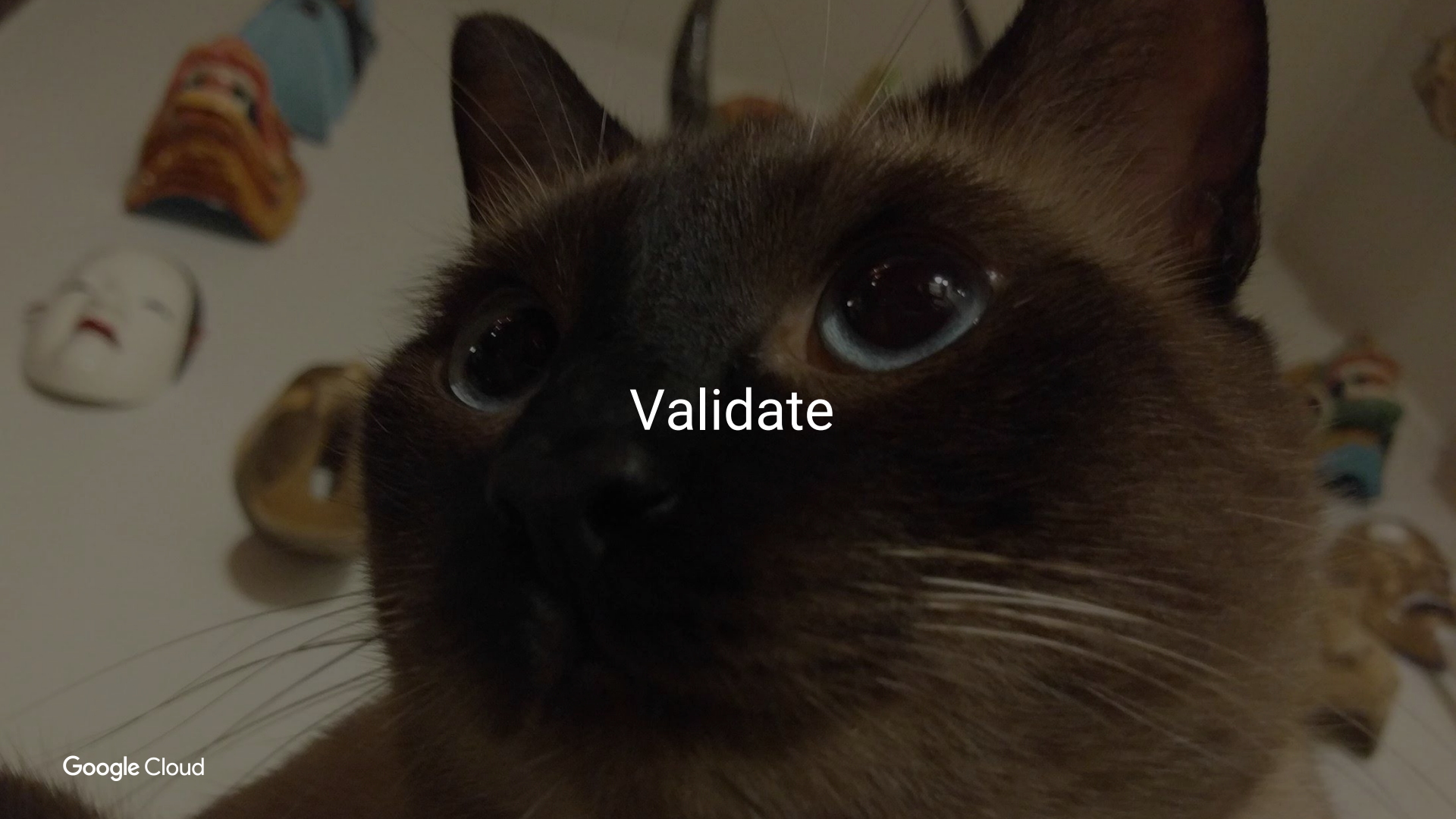
Do you like your performance

Yes? Go to next step

No? Go back to training, try again

Use this, not that




Validate

Step 8 - Validation

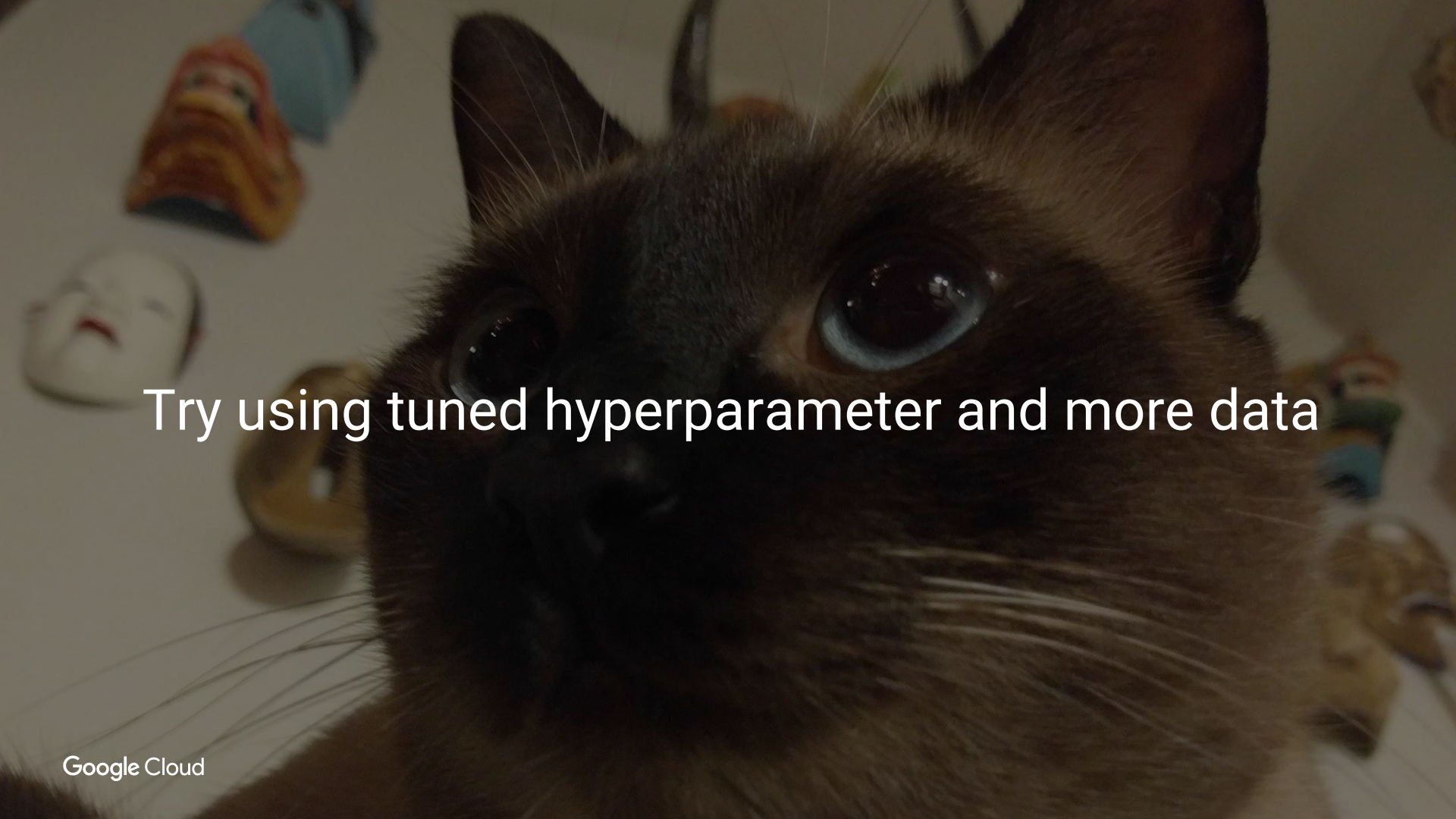
Imagine we had skipped Step 7 (don't worry, we'll use the tuned hyperparameter when we train in the next notebook) and we considered launching the CNN model we just trained in Step 6. The training accuracy looked good, right? Never fear, validation will keep you safe! It's important to validate before trusting a model, so let's do that now and apply cat_finder() to our validation dataset. Since this is validation, we'll only look at the final performance metric (accuracy) and nothing else.

In [24]:
```
files = os.listdir(VALID_DIR)
predicted = cat_finder(VALID_DIR, model_version)
observed = get_labels(VALID_DIR)
print('\nValidation accuracy is ' + str(get_accuracy(observed, predicted)))
```

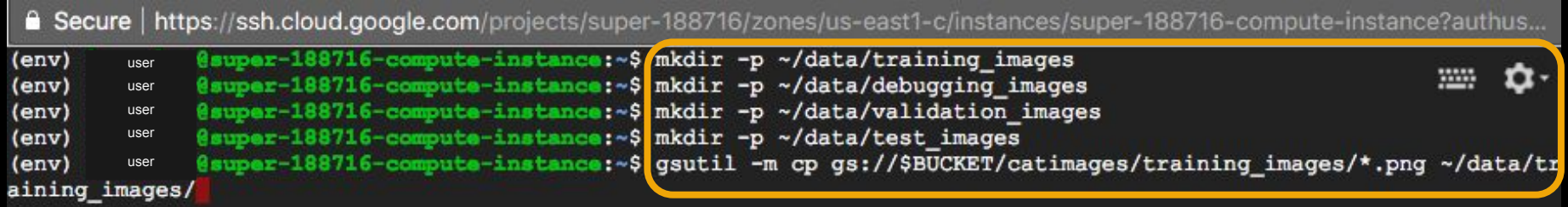
```
1000 predictions completed (out of 23572)...
2000 predictions completed (out of 23572)...
3000 predictions completed (out of 23572)...
4000 predictions completed (out of 23572)...
5000 predictions completed (out of 23572)...
6000 predictions completed (out of 23572)...
7000 predictions completed (out of 23572)...
8000 predictions completed (out of 23572)...
9000 predictions completed (out of 23572)...
10000 predictions completed (out of 23572)...
11000 predictions completed (out of 23572)...
12000 predictions completed (out of 23572)...
13000 predictions completed (out of 23572)...
14000 predictions completed (out of 23572)...
15000 predictions completed (out of 23572)...
16000 predictions completed (out of 23572)...
17000 predictions completed (out of 23572)...
18000 predictions completed (out of 23572)...
19000 predictions completed (out of 23572)...
```

In [24]:
```
files = os.listdir(VALID_DIR)
predicted = cat_finder(VALID_DIR, model_version)
observed = get_labels(VALID_DIR)
print('\nValidation accuracy is ' + str(get_accuracy(observed, predicted)))
```

```
1000 predictions completed (out of 23572)...
2000 predictions completed (out of 23572)...
3000 predictions completed (out of 23572)...
4000 predictions completed (out of 23572)...
5000 predictions completed (out of 23572)...
6000 predictions completed (out of 23572)...
7000 predictions completed (out of 23572)...
8000 predictions completed (out of 23572)...
9000 predictions completed (out of 23572)...
10000 predictions completed (out of 23572)...
11000 predictions completed (out of 23572)...
12000 predictions completed (out of 23572)...
13000 predictions completed (out of 23572)...
14000 predictions completed (out of 23572)...
15000 predictions completed (out of 23572)...
16000 predictions completed (out of 23572)...
17000 predictions completed (out of 23572)...
18000 predictions completed (out of 23572)...
19000 predictions completed (out of 23572)...
20000 predictions completed (out of 23572)...
21000 predictions completed (out of 23572)...
22000 predictions completed (out of 23572)...
23000 predictions completed (out of 23572)...
23572 predictions completed (out of 23572)...

Validation accuracy is 0.74
```


Try using tuned hyperparameter and more data

Copy over the full training, validation, and test sets



mkdir -p ~/data/training_images

mkdir -p ~/data/debugging_images

mkdir -p ~/data/validation_images

mkdir -p ~/data/test_images

gsutil -m cp gs://$BUCKET/catimages/training_images/*.png ~/data/training_images/

gsutil -m cp gs://$BUCKET/catimages/validation_images/*.png ~/data/validation_images/

gsutil -m cp gs://$BUCKET/catimages/test_images/*.png ~/data/test_images/

mv ~/data/training_images/000*.png ~/data/debugging_images/

mv ~/data/training_images/001*.png ~/data/debugging_images/

Make folders and copy over data

echo "done!"

Google Cloud

```
(env)  user  @super-188716-compute-instance:~$ mkdir -p ~/data/training_images
(env)  user  @super-188716-compute-instance:~$ mkdir -p ~/data/debugging_images
(env)  user  @super-188716-compute-instance:~$ mkdir -p ~/data/validation_images
(env)  user  @super-188716-compute-instance:~$ mkdir -p ~/data/test_images
(env)  user  @super-188716-compute-instance:~$ gsutil -m cp gs://$BUCKET/catimages/training_images/*.png ~/data/tr
aining_images/
Copying gs://super-188716-bucket/catimages/training_images/000001_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000002_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000003_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000004_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000005_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000007_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000006_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000008_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000009_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000010_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000011_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000013_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000012_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000015_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000016_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000017_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000018_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000019_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000020_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000021_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000022_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000023_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000024_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000025_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000026_000_0.png...
Copying gs://super-188716-bucket/catimages/training_images/000027_000_1.png...
Copying gs://super-188716-bucket/catimages/training_images/000028_000_1.png...
```

Secure | https://ssh.cloud.google.com/projects/super-188716/zones/us-east1-c/instances/super-188716-compute-instance?authus...

```
Copying gs://super-188716-bucket/catimages/test_images/080161_999_1.png...
Copying gs://super-188716-bucket/catimages/test_images/080162_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080163_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080164_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080165_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080166_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080167_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080168_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080169_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080170_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080171_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080172_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080173_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080174_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080175_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080176_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080177_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080178_999_1.png...
Copying gs://super-188716-bucket/catimages/test_images/080181_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080180_999_1.png...
Copying gs://super-188716-bucket/catimages/test_images/080182_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080183_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080184_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080185_999_0.png...
Copying gs://super-188716-bucket/catimages/test_images/080186_999_0.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080187_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080188_999_1.png...:00
Copying gs://super-188716-bucket/catimages/test_images/080189_999_0.png...:00
 [15.7k/15.7k files][365.3 MiB/365.3 MiB] 100% Done   1.8 MiB/s ETA 00:00:00
Operation completed over 15.7k objects/365.3 MiB.
(env) user@super-188716-compute-instance:~$ mv ~/data/training_images/000*.png ~/data/debugging_images/
(env) user@super-188716-compute-instance:~$ mv ~/data/training_images/001*.png ~/data/debugging_images/
(env) user@super-188716-compute-instance:~$ echo "done!"
done!
(env) user@super-188716-compute-instance:~$
```

Rerun on entire training set

jupyter step_5_to_8_part2 Last Checkpoint: 8 minutes ago (autosaved)

Logout

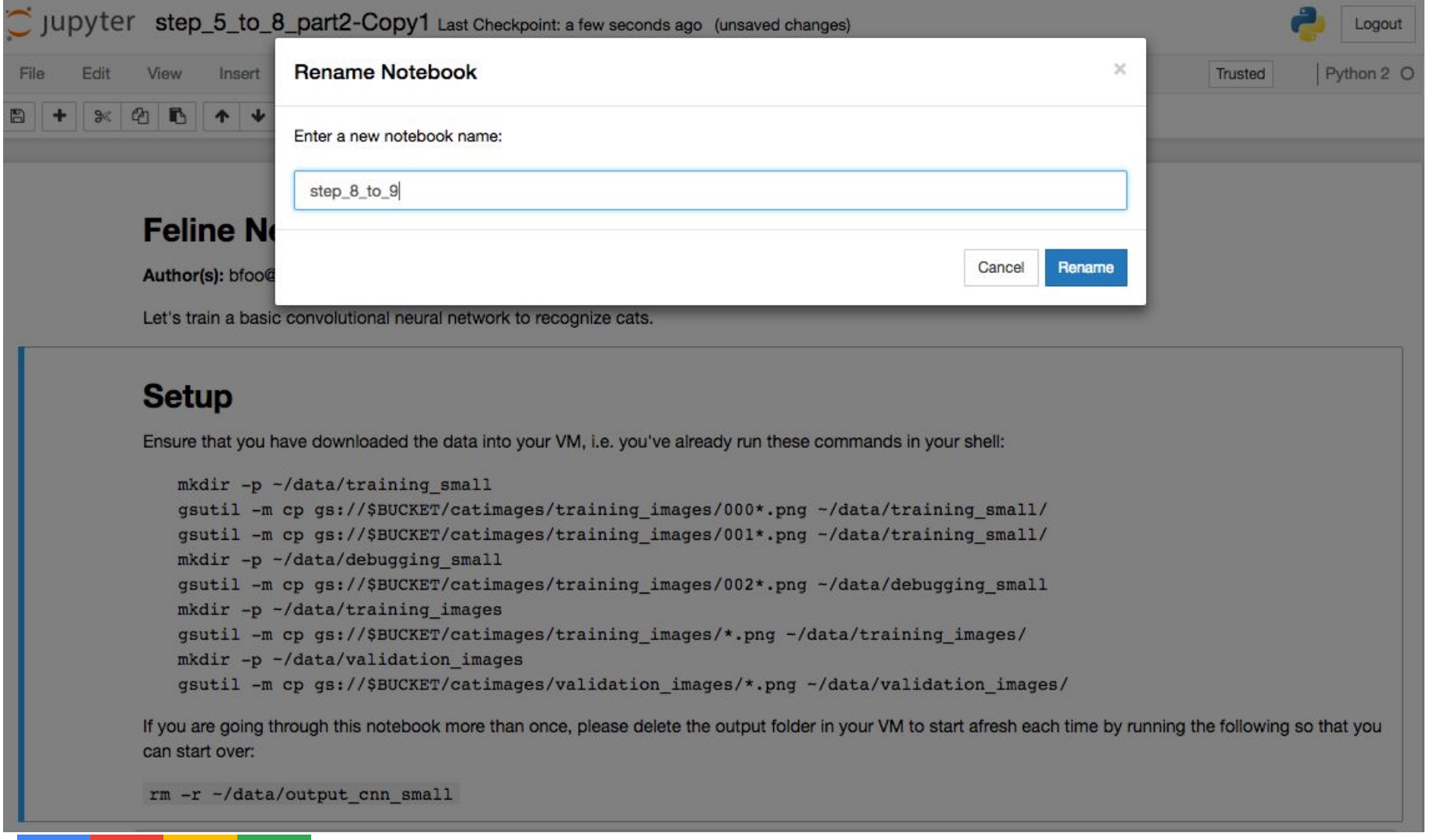
File Edit View Insert Cell Kernel Help

Trusted | Python 2

New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Revert to Checkpoint
Print Preview
Download as
Trusted Notebook
Close and Halt

Run Markdown

```
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
redictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
predictions completed (out of 23572)...
19000 predictions completed (out of 23572)...
20000 predictions completed (out of 23572)...
21000 predictions completed (out of 23572)...
22000 predictions completed (out of 23572)...
23000 predictions completed (out of 23572)...
23572 predictions completed (out of 23572)...

Validation accuracy is 0.74
```


jupyter step_5_to_8_part2-Copy1 Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert
Trusted
Python 2
Rename Notebook
Enter a new notebook name:
step_8_to_9
Cancel
Rename
Feline N
Author(s): bfoo@
Let's train a basic convolutional neural network to recognize cats.
Setup
Ensure that you have downloaded the data into your VM, i.e. you've already run these commands in your shell:
mkdir -p ~/data/training_small
gsutil -m cp gs://$BUCKET/catimages/training_images/000*.png ~/data/training_small/
gsutil -m cp gs://$BUCKET/catimages/training_images/001*.png ~/data/training_small/
mkdir -p ~/data/debugging_small
gsutil -m cp gs://$BUCKET/catimages/training_images/002*.png ~/data/debugging_small
mkdir -p ~/data/training_images
gsutil -m cp gs://$BUCKET/catimages/training_images/*.png ~/data/training_images/
mkdir -p ~/data/validation_images
gsutil -m cp gs://$BUCKET/catimages/validation_images/*.png ~/data/validation_images/
If you are going through this notebook more than once, please delete the output folder in your VM to start afresh each time by running the following so that you can start over:
rm -r ~/data/output_cnn_small

Feline Neural Network - Part 2

Author(s): kozyr@google.com, bfoo@google.com

Let's train a basic convolutional neural network to recognize cats.

Setup

Ensure that you have downloaded the data into your VM, i.e. you've already run these commands in your shell:

```
mkdir -p ~/data/training_small
gsutil -m cp gs://$BUCKET/catimages/training_images/000*.png ~/data/training_small/
gsutil -m cp gs://$BUCKET/catimages/training_images/001*.png ~/data/training_small/
mkdir -p ~/data/debugging_small
gsutil -m cp gs://$BUCKET/catimages/training_images/002*.png ~/data/debugging_small
echo "done!"
mkdir -p ~/data/training_images
gsutil -m cp gs://$BUCKET/catimages/training_images/*.png ~/data/training_images/
mkdir -p ~/data/validation_images
gsutil -m cp gs://$BUCKET/catimages/validation_images/*.png ~/data/validation_images/
mkdir -p ~/data/test_images
gsutil -m cp gs://$BUCKET/catimages/test_images/*.png ~/data/test_images/
mkdir -p ~/data/debugging_images
mv ~/data/training_images/000*.png ~/data/debugging_images/
mv ~/data/training_images/001*.png ~/data/debugging_images/
```

Setup

Ensure that you have downloaded the data into your VM, i.e. you've already run these commands in your shell:

```
mkdir -p ~/data/training_small
gsutil -m cp gs://$BUCKET/catimages/training_images/000*.png ~/data/training_small/
gsutil -m cp gs://$BUCKET/catimages/training_images/001*.png ~/data/training_small/
mkdir -p ~/data/debugging_small
gsutil -m cp gs://$BUCKET/catimages/training_images/002*.png ~/data/debugging_small
echo "done!"
mkdir -p ~/data/training_images
gsutil -m cp gs://$BUCKET/catimages/training_images/*.png ~/data/training_images/
mkdir -p ~/data/validation_images
gsutil -m cp gs://$BUCKET/catimages/validation_images/*.png ~/data/validation_images/
mkdir -p ~/data/test_images
gsutil -m cp gs://$BUCKET/catimages/test_images/*.png ~/data/test_images/
mkdir -p ~/data/debugging_images
mv ~/data/training_images/000*.png ~/data/debugging_images/
mv ~/data/training_images/001*.png ~/data/debugging_images/
echo "done!"
```

If you are going through this notebook more than once, please delete the output folder in your VM to start afresh each time by running the following so that you can start over:

```
rm -r ~/data/output_cnn_big
```

In [1]:
```
# Enter your username:
YOUR_GMAIL_ACCOUNT = '******' # Whatever is before @gmail.com in your email address
```

In [2]:

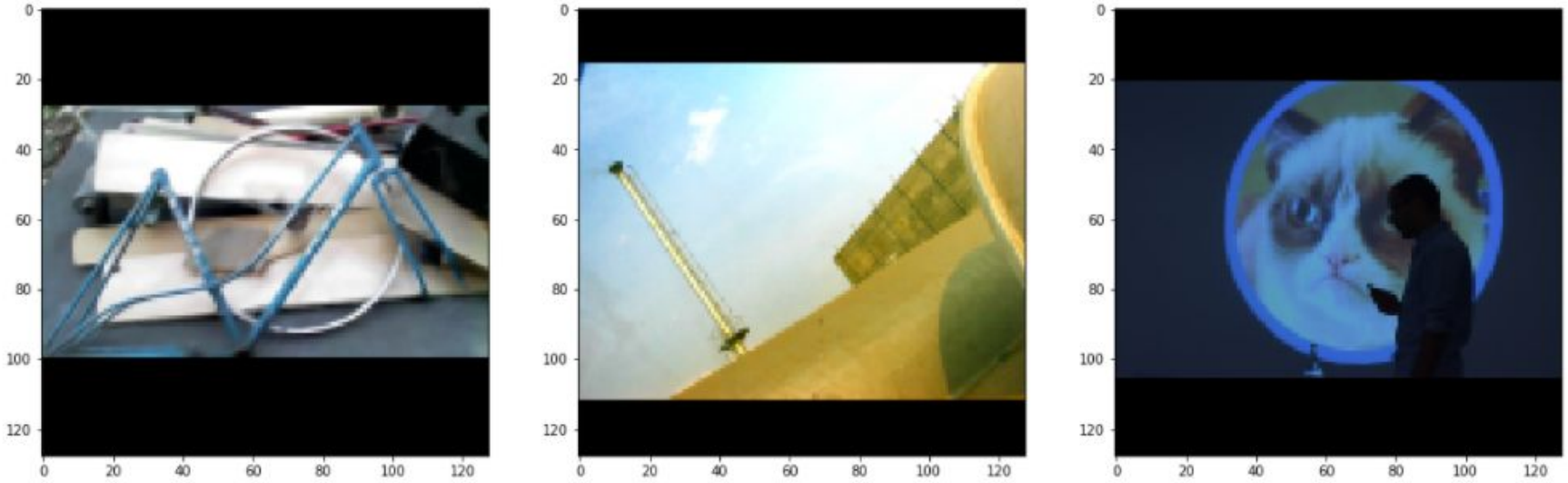
```
# Libraries for this section:
import os
import datetime
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import tensorflow as tf
from tensorflow.contrib.learn import RunConfig, Experiment
from tensorflow.contrib.learn.python.learn import learn_runner
```

In [3]:
```
# Directory settings:
TRAIN_DIR = os.path.join('../..', YOUR_GMAIL_ACCOUNT, 'data/training_images/')  # Where the training dataset lives.
DEBUG_DIR = os.path.join('../..', YOUR_GMAIL_ACCOUNT, 'data/debugging_images/')  # Where the debugging dataset lives.
VALID_DIR = os.path.join('../..', YOUR_GMAIL_ACCOUNT, 'data/validation_images/')  # Where the validation dataset lives.
TEST_DIR = os.path.join('../..', YOUR_GMAIL_ACCOUNT, 'data/test_images/')  # Where the test dataset lives.
OUTPUT_DIR = os.path.join('../..', YOUR_GMAIL_ACCOUNT, 'data/output_cnn_big/')  # Where we store our logging and models

# TensorFlow setup:
NUM_CLASSES = 2  # This code can be generalized beyond 2 classes (binary classification).
QUEUE_CAP = 5000  # Number of images the TensorFlow queue can store during training.
# For debugging, QUEUE_CAP is ignored in favor of using all images available.
TRAIN_BATCH_SIZE = 128  # Number of images processed every training iteration.
DEBUG_BATCH_SIZE = 64  # Number of images processed every debugging iteration.
TRAIN_STEPS = 3000 # Number of batches to use for training.
DEBUG_STEPS = 2 # Number of batches to use for debugging.
# Example: If dataset is 5 batches ABCDE, train_steps = 2 uses AB, train_steps = 7 uses ABCDEAB).

# Monitoring setup:
TRAINING_LOG_PERIOD_SECS = 60  # How often we want to log training metrics (from training hook in our model_fn).
CHECKPOINT_PERIOD_SECS = 60  # How often we want to save a checkpoint.

# Hyperparameters we'll tune in the tutorial:
DROPOUT = 0.6  # Regularization parameter for neural networks - must be between 0 and 1.

# Additional hyperparameters:
LEARNING_RATE = 0.001  # Rate at which weights update.
CNN_KERNEL_SIZE = 3  # Receptive field will be square window with this many pixels per side.
CNN_STRIDES = 2  # Distance between consecutive receptive fields.
CNN_FILTERS = 16  # Number of filters (new receptive fields to train, i.e. new channels) in first convolutional layer.
FC_HIDDEN_UNITS = 512  # Number of hidden units in the fully connected layer of the network.
```

In [4]:
```
def show_inputs(dir, filelist=None, img_rows=1, img_cols=3, figsize=(20, 10)):
  """Display the first few images.

  Args:
    dir: directory where the files are stored
    filelist: list of filenames to pull from, if left as default, all files will be used
    img_rows: number of rows of images to display
    img_cols: number of columns of images to display
    figsize: sizing for inline plots

  Returns:
    pixel_dims: pixel dimensions (height and width) of the image
  """
  if filelist is None:
    filelist = os.listdir(dir)  # Grab all the files in the directory
  filelist = np.array(filelist)
  plt.close('all')
  fig = plt.figure(figsize=figsize)
  print('File names:')

  for i in range(img_rows * img_cols):
    print(str(filelist[i]))
    a=fig.add_subplot(img_rows, img_cols,i + 1)
    img = mpimg.imread(os.path.join(dir, str(filelist[i])))
    plt.imshow(img)
  plt.show()
  return np.shape(img)
```

In [5]:
```
pixel_dim = show_inputs(TRAIN_DIR)
print('Images have ' + str(pixel_dim[0]) + 'x' + str(pixel_dim[1]) + ' pixels.')
pixels = pixel_dim[0] * pixel_dim[1]
```

```
File names:
037831_473_0.png
034435_431_0.png
025375_317_1.png
```




```
Images have 128x128 pixels.
```

TF Estimator

This is where it all comes together: TF Estimator takes in as input everything we've created thus far and when executed it will output everything that is necessary for training (fits a model), evaluation (outputs metrics), or prediction (outputs predictions).

In [9]:
```
# TF Estimator:
# WARNING: Don't run this block of code more than once without first changing OUTPUT_DIR.
estimator = tf.estimator.Estimator(
  model_fn=generate_model_fn(DROPOUT),  # Call our generate_model_fn to create model function
  model_dir=OUTPUT_DIR,  # Where to look for data and also to paste output.
  config=RunConfig(
    save_checkpoints_secs=CHECKPOINT_PERIOD_SECS,
    keep_checkpoint_max=20,
    save_summary_steps=100,
    log_step_count_steps=100)
)
```

```
INFO:tensorflow:Using config: {'_save_checkpoints_secs': 60, '_num_ps_replicas': 0, '_keep_checkpoint_max': 20, '_tas
k_type': None, '_is_chief': True, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7f2
0e4931890>, '_model_dir': '../../kozyrkov/data/output_cnn_big/', '_save_checkpoints_steps': None, '_keep_checkpoint_e
very_n_hours': 10000, '_session_config': None, '_tf_random_seed': None, '_save_summary_steps': 100, '_environment':
'local', '_num_worker_replicas': 0, '_task_id': 0, '_log_step_count_steps': 100, '_tf_config': gpu_options {
  per_process_gpu_memory_fraction: 1.0
}
, '_evaluation_master': '', '_master': ''}
```

TF Experiment

A TF Experiment defines **how** to run your TF estimator during training and debugging only. TF Experiments are not necessary for prediction once training is complete.

TERMINOLOGY WARNING: The word "experiment" here is not used the way it is used by typical scientists and statisticians.

In [10]:
```
# TF Experiment:
def experiment_fn(output_dir):
    """Create _experiment_fn which returns a TF experiment

    To be used with learn_runner, which we imported from tf.

    Args:
      output_dir: which is where we write our models to.
    Returns:
      a TF Experiment
    """

    return Experiment(
      estimator=estimator,
      train_input_fn=generate_input_fn(TRAIN_DIR, TRAIN_BATCH_SIZE, QUEUE_CAP),  # Generate input function above.
      eval_input_fn=generate_input_fn(DEBUG_DIR, DEBUG_BATCH_SIZE, QUEUE_CAP),
      train_steps=TRAIN_STEPS,  # Number of batches to use for training.
      eval_steps=DEBUG_STEPS, # Number of batches to use for eval.
      min_eval_frequency=1 # Run eval once every min_eval_frequency number of checkpoints.
    )
```

Step 6 - Train a model!

Let's run our lovely creation on our training data. In order to train, we need learn_runner(), which we imported from TensorFlow above. For prediction, we will only need estimator.predict().

In [11]:
```
# Enable TF verbose output:
tf.logging.set_verbosity(tf.logging.INFO)
start_time = datetime.datetime.now()
print('It\'s {:%H:%M} in London'.format(start_time) + ' --- Let\'s get started!')
# Let the learning commence!  Run the TF Experiment here.
learn_runner.run(experiment_fn, OUTPUT_DIR)
# Output lines using the word "Validation" are giving our metric on the non-training dataset (from DEBUG_DIR).
end_time = datetime.datetime.now()
print('\nIt was {:%H:%M} in London when we started.'.format(start_time))
print('\nWe\'re finished and it\'s {:%H:%M} in London'.format(end_time))
print('\nCongratulations!  Training is complete!')
```

```
It's 13:19 in London --- Let's get started!
WARNING:tensorflow:From <ipython-input-10-deff8ba6df81>:20: calling __init__ (from tensorflow.contrib.learn.python.le
arn.experiment) with local_eval_frequency is deprecated and will be removed after 2016-10-23.
Instructions for updating:
local_eval_frequency is deprecated as local_run will be renamed to train_and_evaluate. Use min_eval_frequency and cal
l train_and_evaluate instead. Note, however, that the default for min_eval_frequency is 1, meaning models will be eva
luated every time a new checkpoint is available. In contrast, the default for local_eval_frequency is None, resulting
in evaluation occurring only after training has completed. min_eval_frequency is ignored when calling the deprecated
local_run.
WARNING:tensorflow:From /home/ user /env/local/lib/python2.7/site-packages/tensorflow/contrib/learn/python/learn/mo
nitors.py:269: __init__ (from tensorflow.contrib.learn.python.learn.monitors) is deprecated and will be removed after
2016-12-05.
```

In [11]:
```
# Enable TF verbose output:
tf.logging.set_verbosity(tf.logging.INFO)
start_time = datetime.datetime.now()
print('It\'s {:%H:%M} in London'.format(start_time) + ' --- Let\'s get started!')
# Let the learning commence!  Run the TF Experiment here.
learn_runner.run(experiment_fn, OUTPUT_DIR)
# Output lines using the word "Validation" are giving our metric on the non-training dataset (from DEBUG_DIR).
end_time = datetime.datetime.now()
print('\nIt was {:%H:%M} in London when we started.'.format(start_time))
print('\nWe\'re finished and it\'s {:%H:%M} in London'.format(end_time))
print('\nCongratulations!  Training is complete!')
```

```
INFO:tensorflow:Restoring parameters from ../../user/data/output_cnn_big/model.ckpt-2952
INFO:tensorflow:Evaluation [1/2]
INFO:tensorflow:Evaluation [2/2]
INFO:tensorflow:Finished evaluation at 2018-01-03-13:37:01
INFO:tensorflow:Saving dict for global step 2952: accuracy = 0.84375, global_step = 2952, loss = 0.520053
INFO:tensorflow:Validation (step 2952): loss = 0.520053, global_step = 2952, accuracy = 0.84375
INFO:tensorflow:Saving checkpoints for 3000 into ../../user/data/output_cnn_big/model.ckpt.
INFO:tensorflow:Loss for final step: 0.0658541.
INFO:tensorflow:Starting evaluation at 2018-01-03-13:37:07
INFO:tensorflow:Restoring parameters from ../../user/data/output_cnn_big/model.ckpt-3000
INFO:tensorflow:Evaluation [1/2]
INFO:tensorflow:Evaluation [2/2]
INFO:tensorflow:Finished evaluation at 2018-01-03-13:37:07
INFO:tensorflow:Saving dict for global step 3000: accuracy = 0.875, global_step = 3000, loss = 0.349303

It was 13:19 in London when we started.

We're finished and it's 13:37 in London

Congratulations!  Training is complete!
```

Get predictions and performance metrics

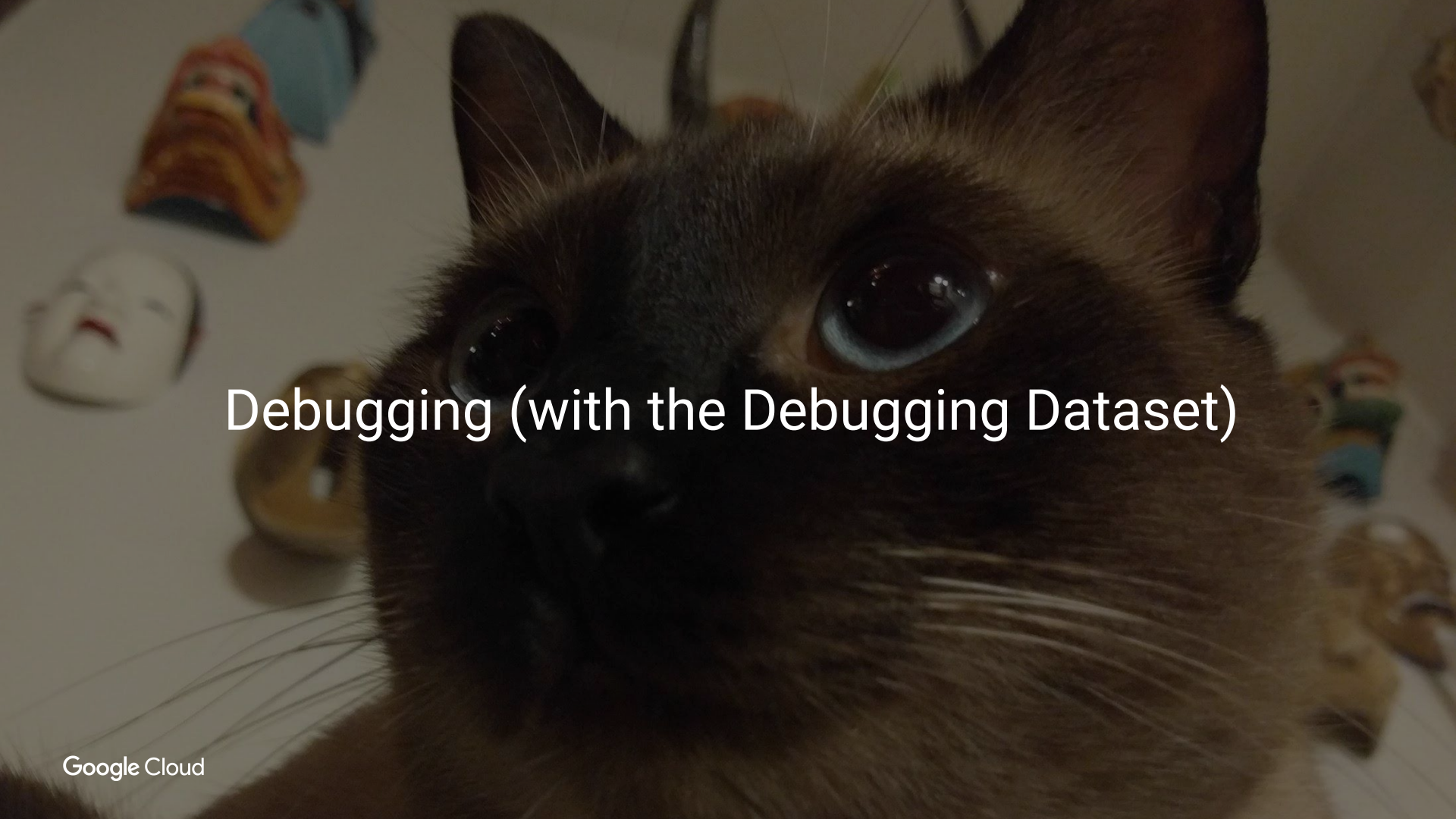
Create functions for outputting observed labels, predicted labels, and accuracy. Filenames must be in the following format: number_number_label.extension

In [16]:
```
files = os.listdir(TRAIN_DIR)
model_version = os.path.join(OUTPUT_DIR, 'model.ckpt-' + str(TRAIN_STEPS))
predicted = cat_finder(TRAIN_DIR, model_version)
observed = get_labels(TRAIN_DIR)
```

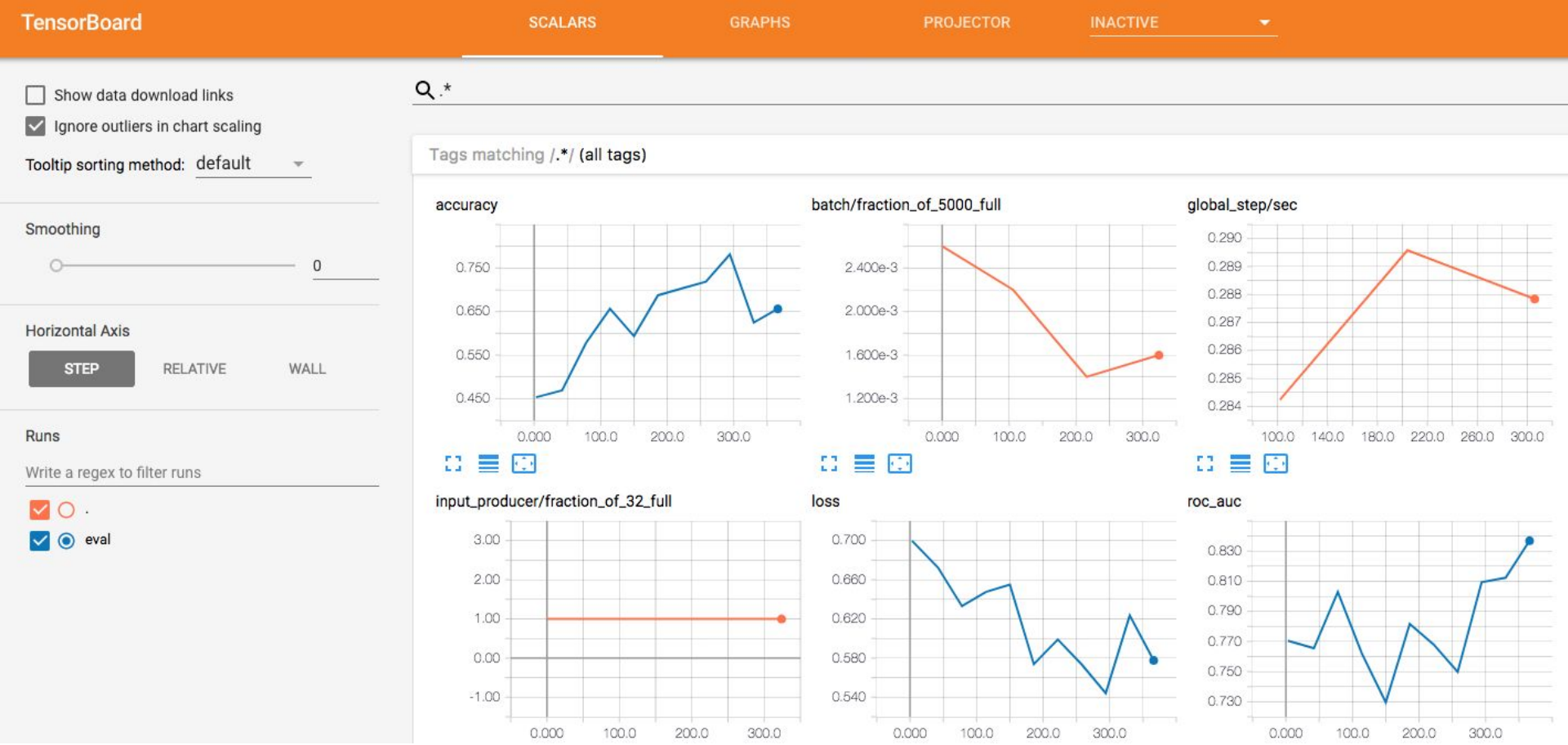
```
INFO:tensorflow:Restoring parameters from ../../ user /data/output_cnn_big/model.ckpt-3000
1000 predictions completed (out of 37040)...
2000 predictions completed (out of 37040)...
3000 predictions completed (out of 37040)...
4000 predictions completed (out of 37040)...
5000 predictions completed (out of 37040)...
6000 predictions completed (out of 37040)...
7000 predictions completed (out of 37040)...
8000 predictions completed (out of 37040)...
9000 predictions completed (out of 37040)...
10000 predictions completed (out of 37040)...
11000 predictions completed (out of 37040)...
12000 predictions completed (out of 37040)...
13000 predictions completed (out of 37040)...
...
37000 predictions completed (out of 37040)...
37040 predictions completed (out of 37040)...
```

In [17]:
```
print('Training accuracy is ' + str(get_accuracy(observed, predicted)))
```

```
Training accuracy is 0.95
```



Google Cloud



Connect TensorBoard to model output directory (summaries and checkpoints) to check progress.

tensorboard --port 8080 --logdir gs://bucket/model-output-dir

TensorBoard SCALARS GRAPHS PROJECTOR INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

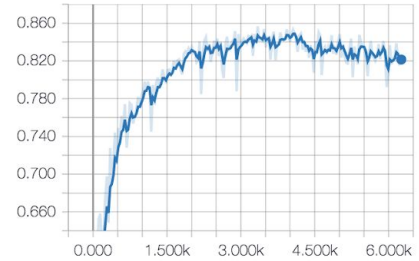
.

eval

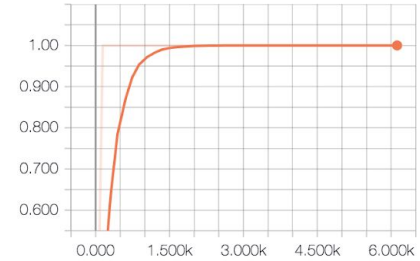
.*

Tags matching /.*/ (all tags)

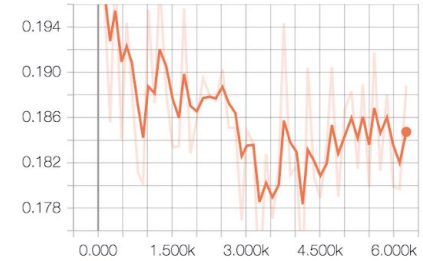
accuracy



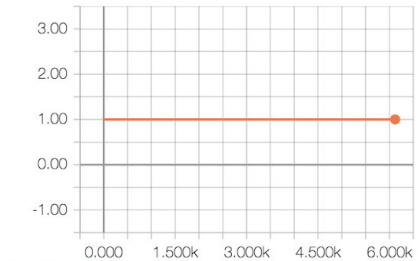
batch/fraction_of_5000_full



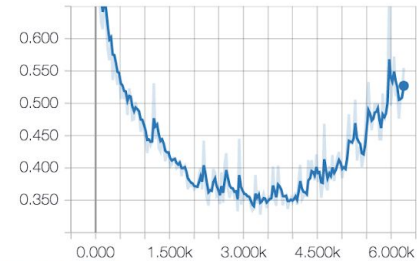
global_step/sec



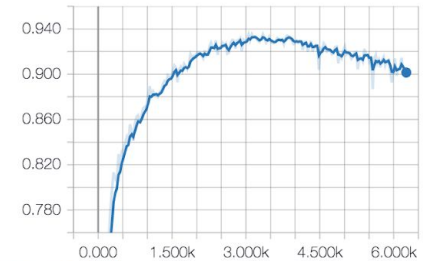
input_producer/fraction_of_32_full



loss



roc_auc



Step 7 - Debugging and Tuning

Debugging

It's worth taking a look to see if there's something special about the images we misclassified.

In [18]:
```
files = os.listdir(DEBUG_DIR)
model_version = os.path.join(OUTPUT_DIR + 'model.ckpt-' + str(TRAIN_STEPS))
predicted = cat_finder(DEBUG_DIR, model_version)
observed = get_labels(DEBUG_DIR)
```

```
INFO:tensorflow:Restoring parameters from ../../ user /data/output_cnn_big/model.ckpt-3000
1000 predictions completed (out of 1960)...
1960 predictions completed (out of 1960)...
```

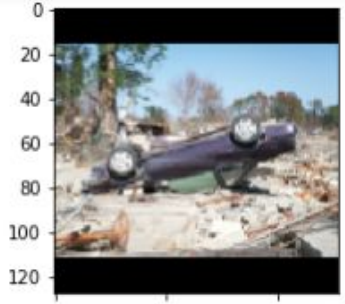
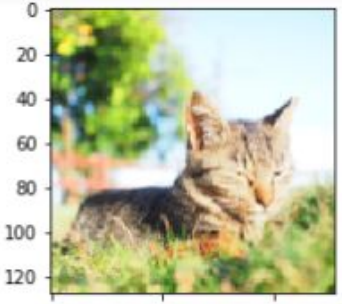
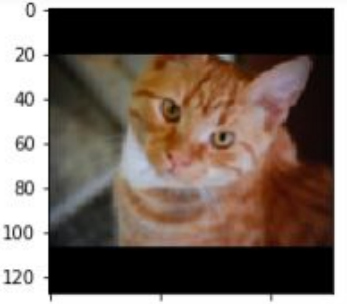
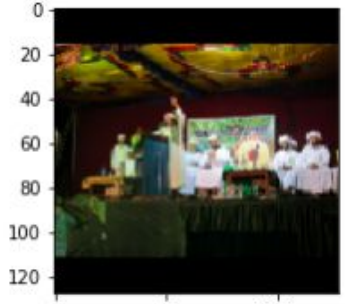
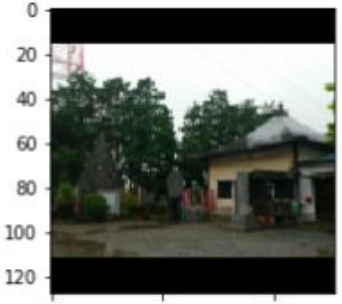
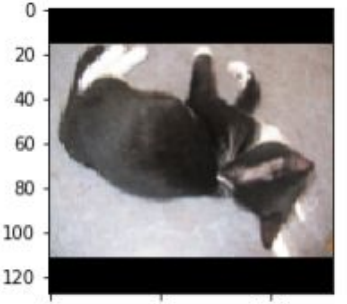
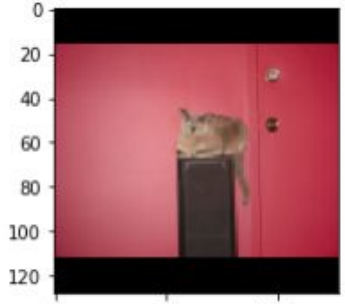
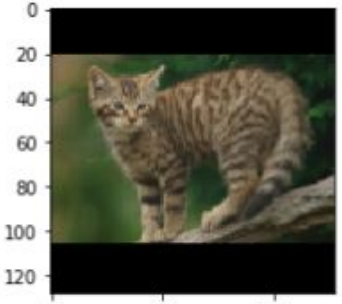
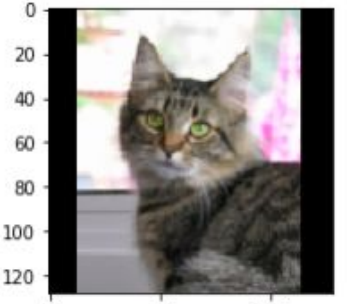
In [19]:
```
print('Debugging accuracy is ' + str(get_accuracy(observed, predicted)))
```

```
Debugging accuracy is 0.85
```

In [20]:
```
df = pd.DataFrame({'files': files, 'predicted': predicted, 'observed': observed})
hit = df.files[df.observed == df.predicted]
miss = df.files[df.observed != df.predicted]
```

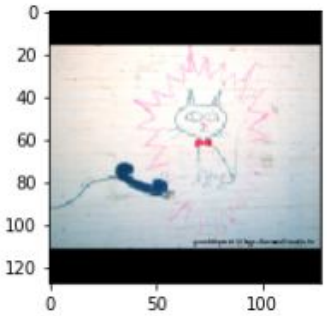
In [21]:
```
# Show successful classifications:
show_inputs(DEBUG_DIR, hit, 3)
```

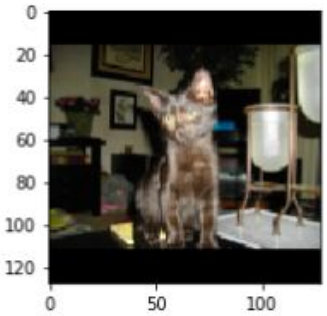
```
File names:
001639_021_0.png
001297_016_1.png
```

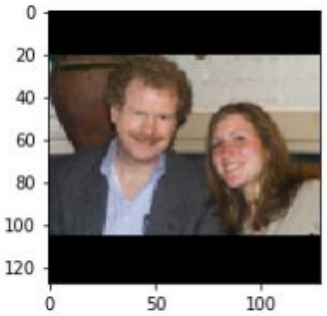










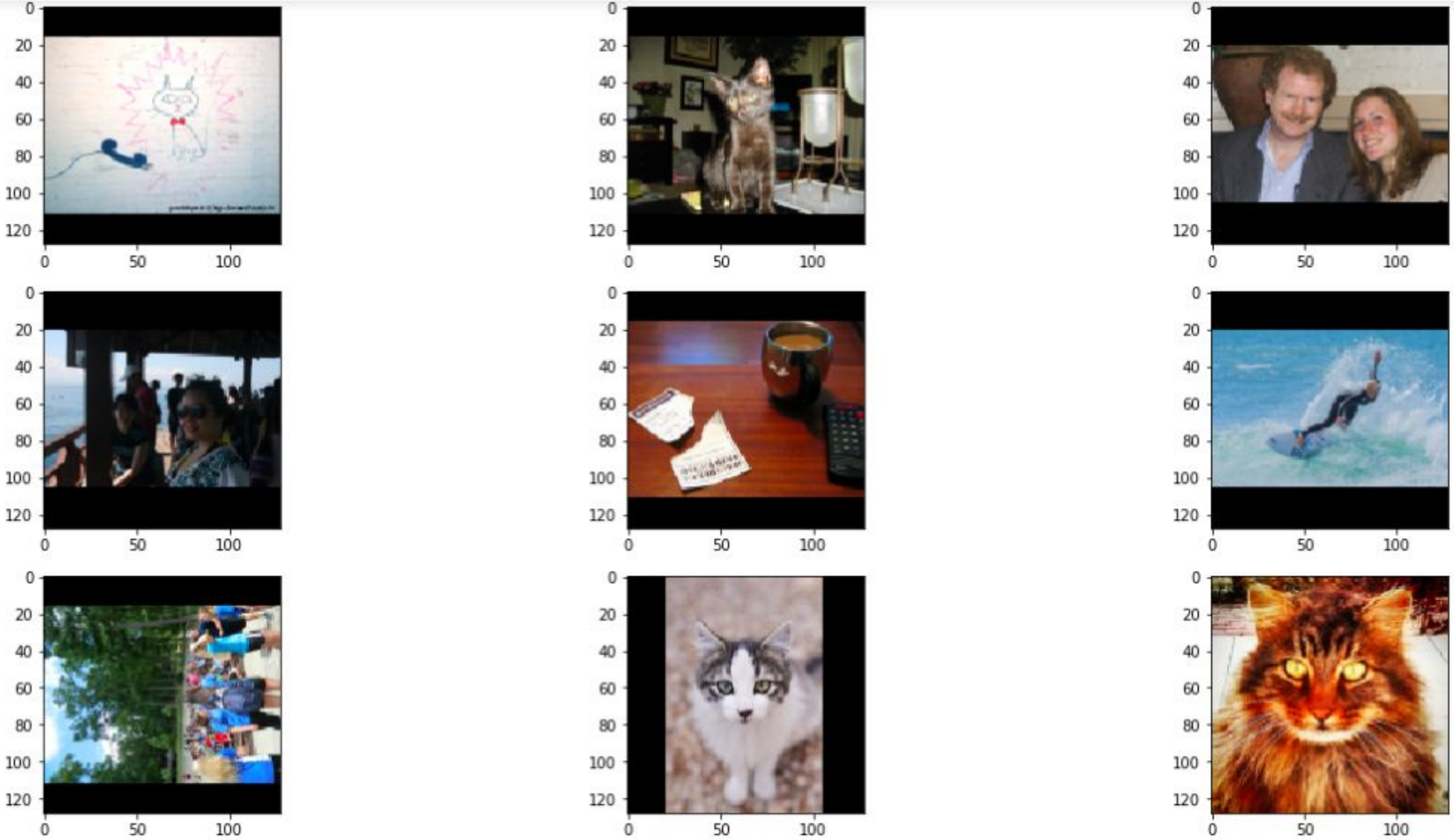
In [22]:
```
# Show unsuccessful classifications:
show_inputs(DEBUG_DIR, miss, 3)
```

```
File names:
001782_023_1.png
000658_009_1.png
000450_006_0.png
000052_001_0.png
001046_013_0.png
001411_018_0.png
000919_012_0.png
001506_019_1.png
000490_006_1.png
```









Tuning vs Debugging

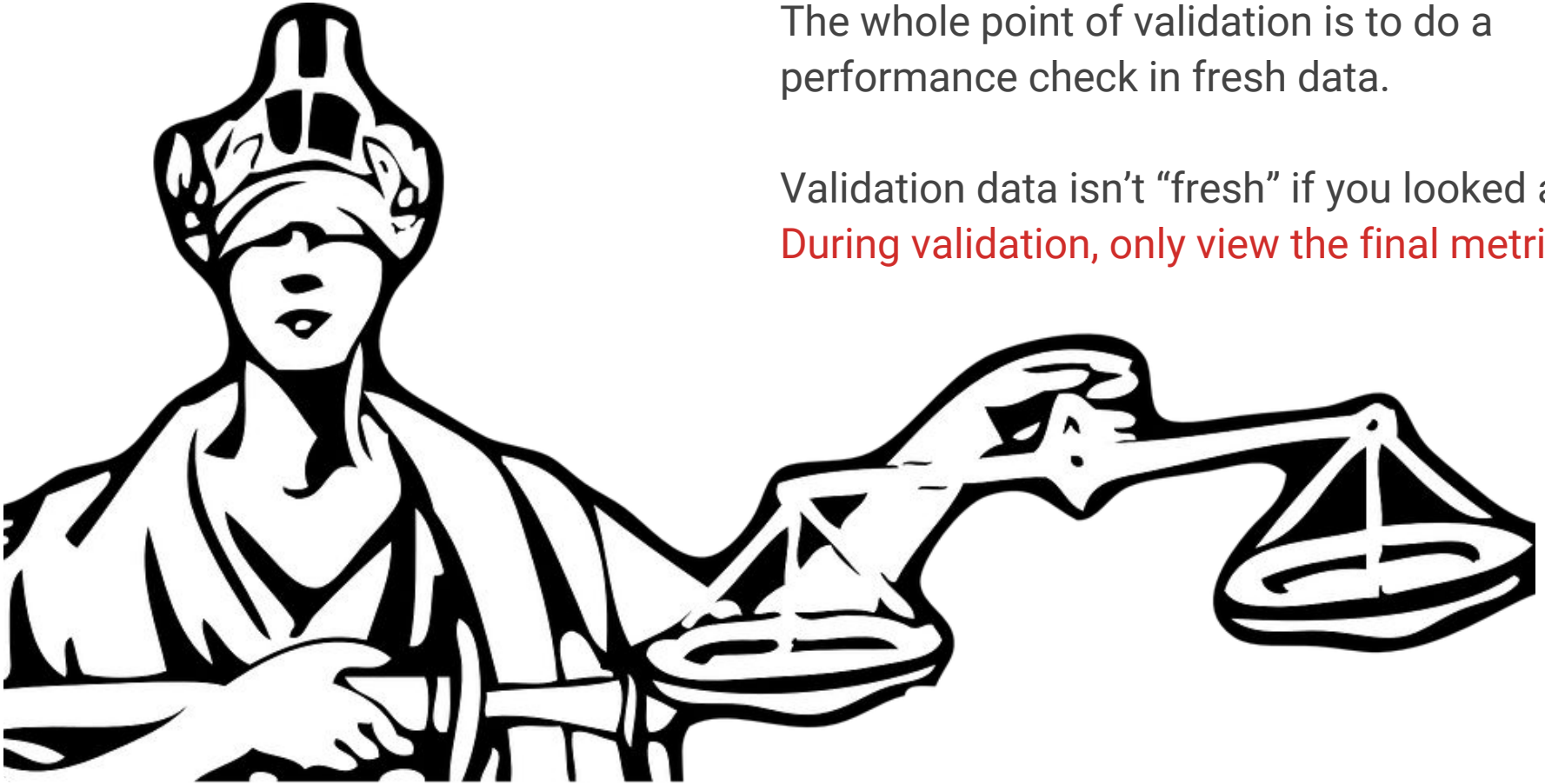
Tuning could be considered part of the training process. Why?

We're allowed to look at and tinker with training and tuning data. May debug using them.

Google Cloud

* Is it possible to overfit even if you did tuning with cross-validation? Yes

Validation is blind!



The whole point of validation is to do a performance check in fresh data.

Validation data isn't "fresh" if you looked at it.
During validation, only view the final metric.

Danger! Pitfall alert

Don’t debug with your validation data!

You’ll lose the benefits of validation.

Anything you look at effectively joins the training set. The only dataset you should be looking at is your training data - use it to make debugging data as and when you need to.

Key message

Validation is all about checking if your model succeeds on a new dataset

This protects you from blindly overfitting

Only view the final metric, not individual validation instances


Validation time!

Step 8 - Validation

Apply cat_finder() to the validation dataset. Since this is validation, we'll only look at the final performance metric (accuracy) and nothing else.

In [23]:
```
files = os.listdir(VALID_DIR)
predicted = cat_finder(VALID_DIR, model_version)
observed = get_labels(VALID_DIR)
print('\nValidation accuracy is ' + str(get_accuracy(observed, predicted)))
```

```
INFO:tensorflow:Restoring parameters from ../../user/data/output_cnn_big/model.ckpt-3000
1000 predictions completed (out of 23572)...
2000 predictions completed (out of 23572)...
3000 predictions completed (out of 23572)...
4000 predictions completed (out of 23572)...
5000 predictions completed (out of 23572)...
6000 predictions completed (out of 23572)...
7000 predictions completed (out of 23572)...
8000 predictions completed (out of 23572)...
9000 predictions completed (out of 23572)...
10000 predictions completed (out of 23572)...
11000 predictions completed (out of 23572)...
12000 predictions completed (out of 23572)...
13000 predictions completed (out of 23572)...
14000 predictions completed (out of 23572)...
15000 predictions completed (out of 23572)...
16000 predictions completed (out of 23572)...
17000 predictions completed (out of 23572)...
18000 predictions completed (out of 23572)...
19000 predictions completed (out of 23572)...
23000 predictions completed (out of 23572)...
23572 predictions completed (out of 23572)...

Validation accuracy is 0.84
```

Step 8 is finished | You have your best candidate model and now you've evaluated

Chief actors:

Analyst

Step 8 is finished | You have your best candidate model and now you've evaluated

84% accuracy
Ready to proceed to testing

Chief actors:

Analyst

Google Cloud