Deep Learning Walkthrough - 07

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

Google Cloud

Step 7 | Tune and debug

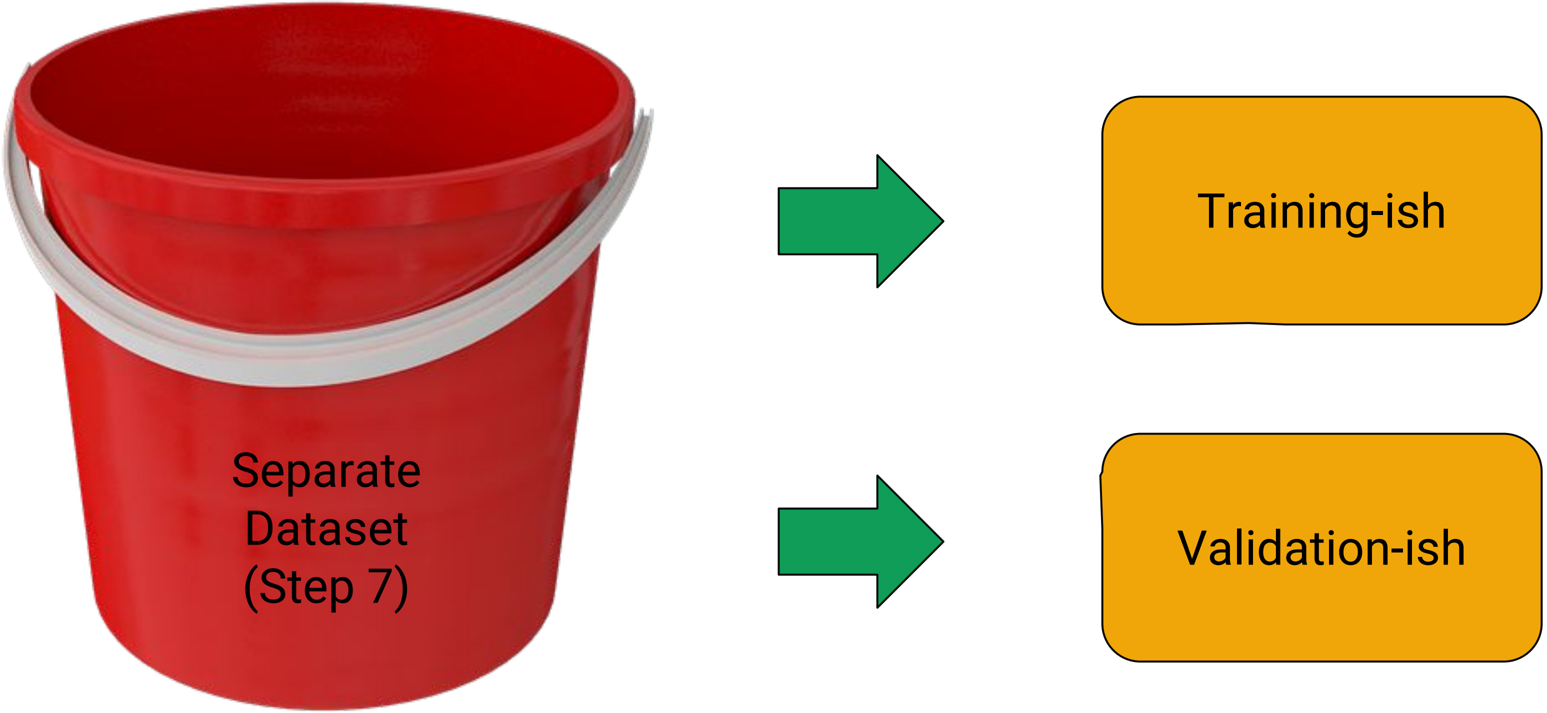
Improve your awesome recipe

Chief actors:

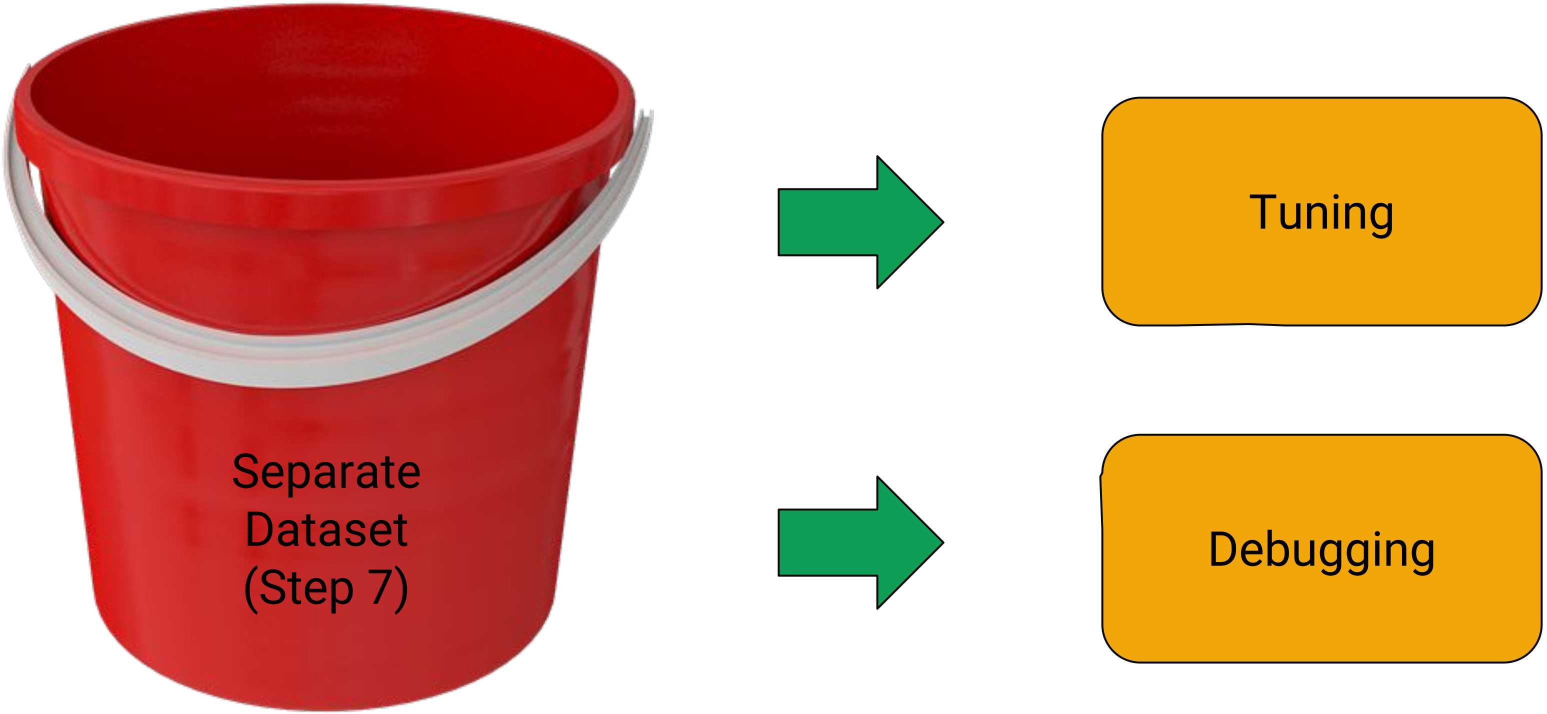
Engineer

Analyst

Call it what you like, but please do it properly!



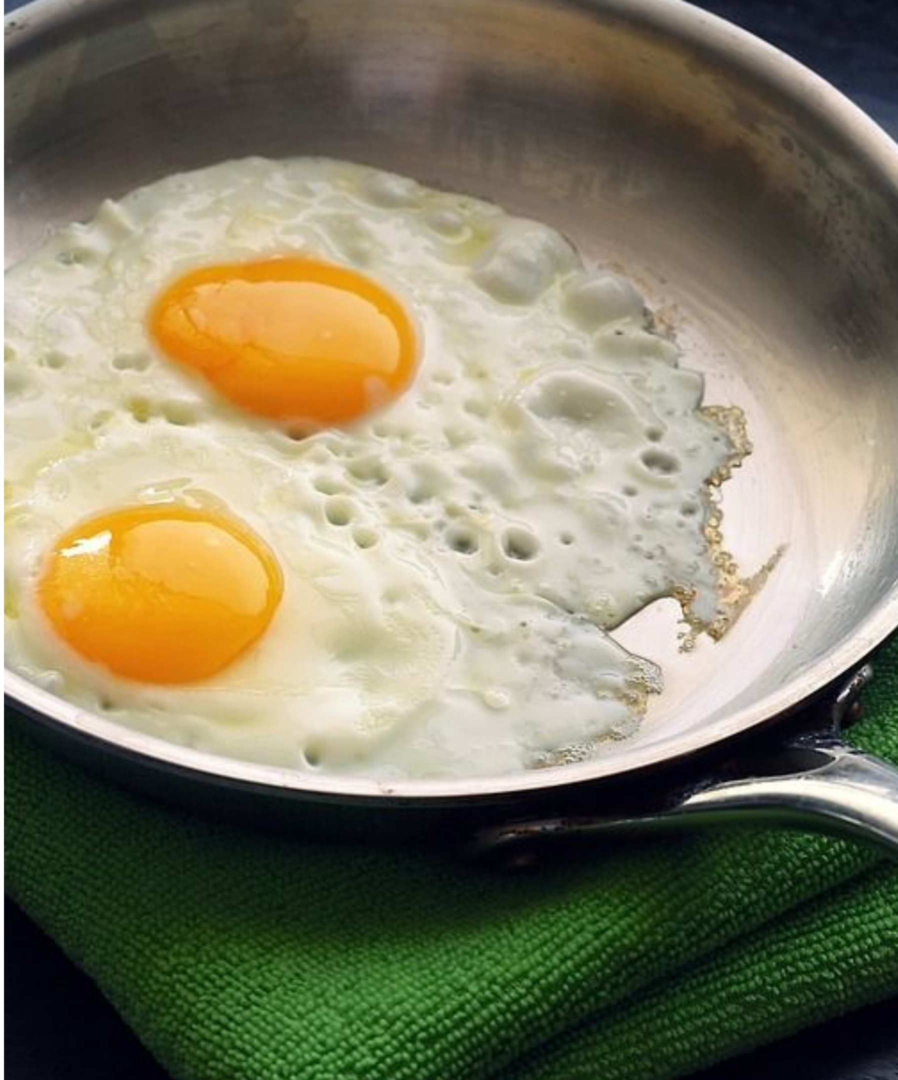
Call it what you like, but please do it properly!



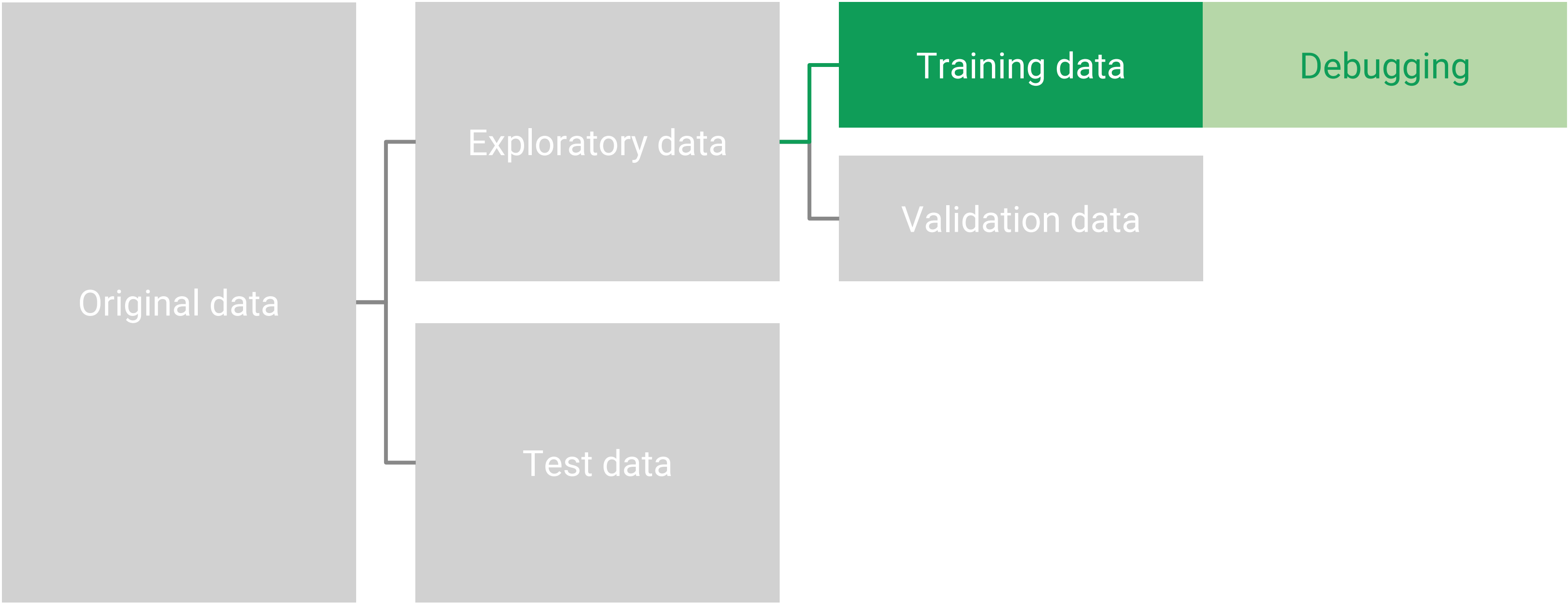
In practice

!

You may **pull debugging/ tuning datasets** out of your training set as needed or have a fixed set at the start



Use this, not that

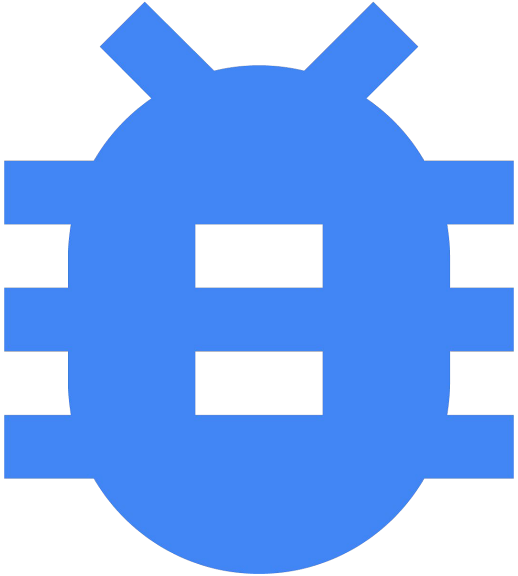


Debugging

If you want to do debugging, do it with your Step 7 dataset

How?

- Fit a model in training data
- Check performance in debugging data
- Look at instances model got wrong



Confidential & Proprietary

Debugging example

Debugging

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

In [19]:
```
files = os.listdir(DEBUG_DIR)
model_version = OUTPUT_DIR + 'model.ckpt-' + str(TRAIN_STEPS)
predicted = cat_finder(DEBUG_DIR, model_version)
observed = get_labels(DEBUG_DIR)
```

```
INFO:tensorflow:Restoring parameters from ../../data/output/model.ckpt-100
```

In [20]:
```
print('Debugging accuracy is ' + str(get_accuracy(observed, predicted)))
```

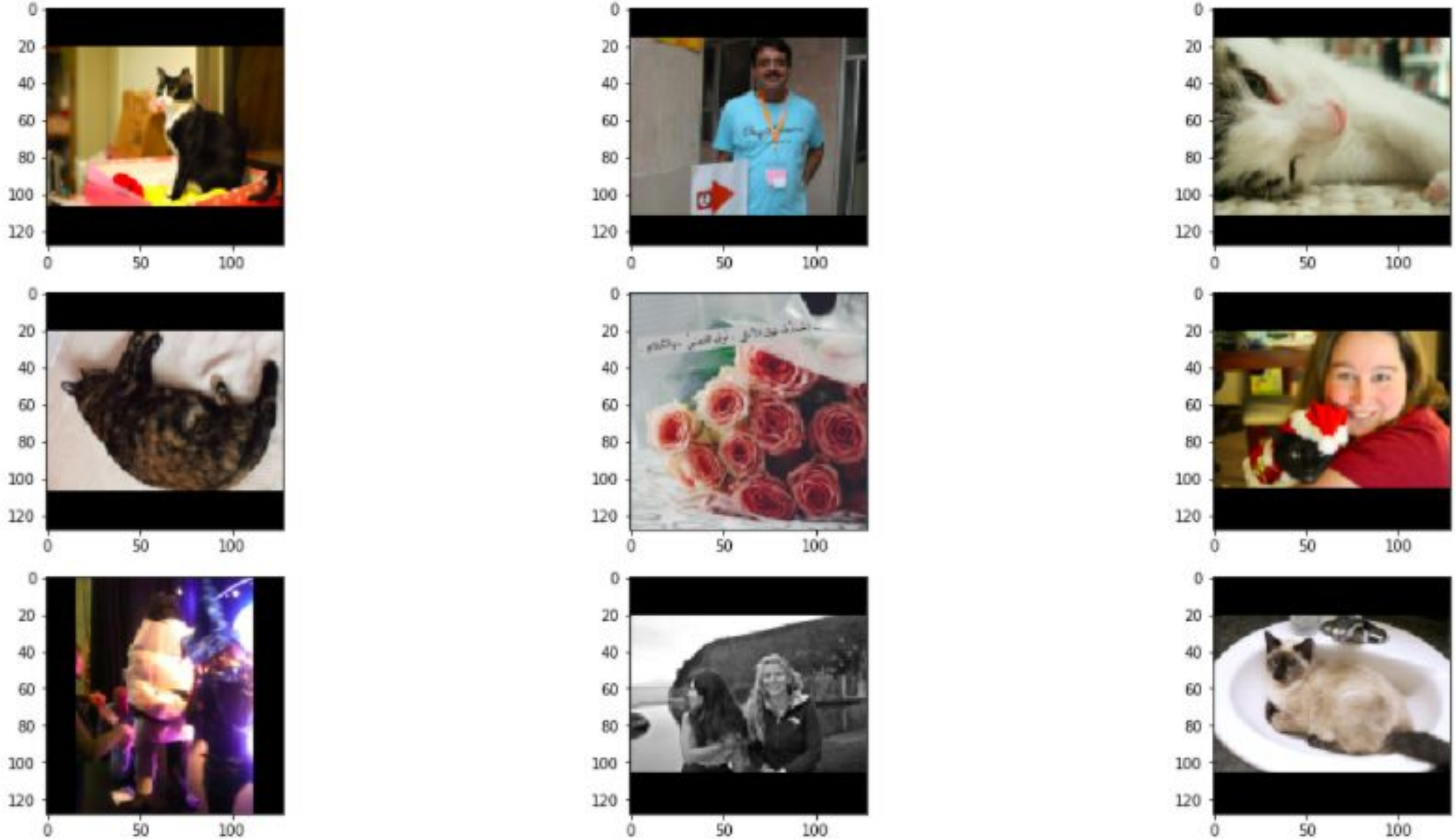
```
Debugging accuracy is 0.7
```

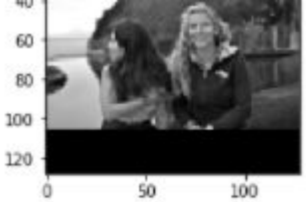
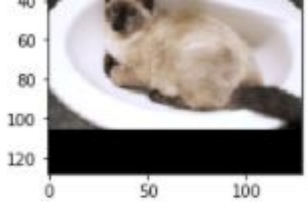
In [21]:
```
df = pd.DataFrame({'files': files, 'predicted': predicted, 'observed': observed})
hit = df.files[df.observed == df.predicted]
miss = df.files[df.observed != df.predicted]
```

In [23]:
```
# Show successful classifications:
show_inputs(DEBUG_DIR, hit, 3)
```

```
File names:
14091_0.1764_1.png
13209_0.1656_0.png
13817_0.1727_1.png
14132_0.1769_1.png
13919_0.1740_0.png
13713_0.1715_1.png
```

13834_0.1729_0.png
14281_0.1784_0.png
14978_0.1870_1.png

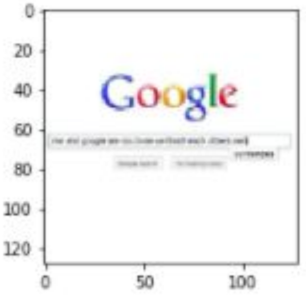


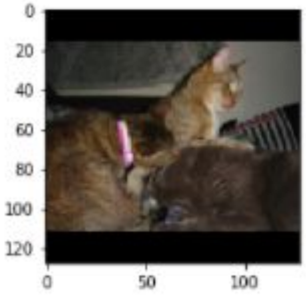
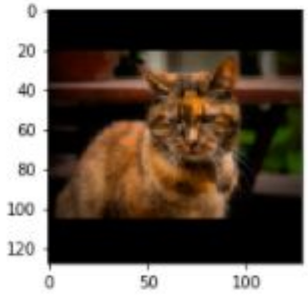




Out[23]: 128

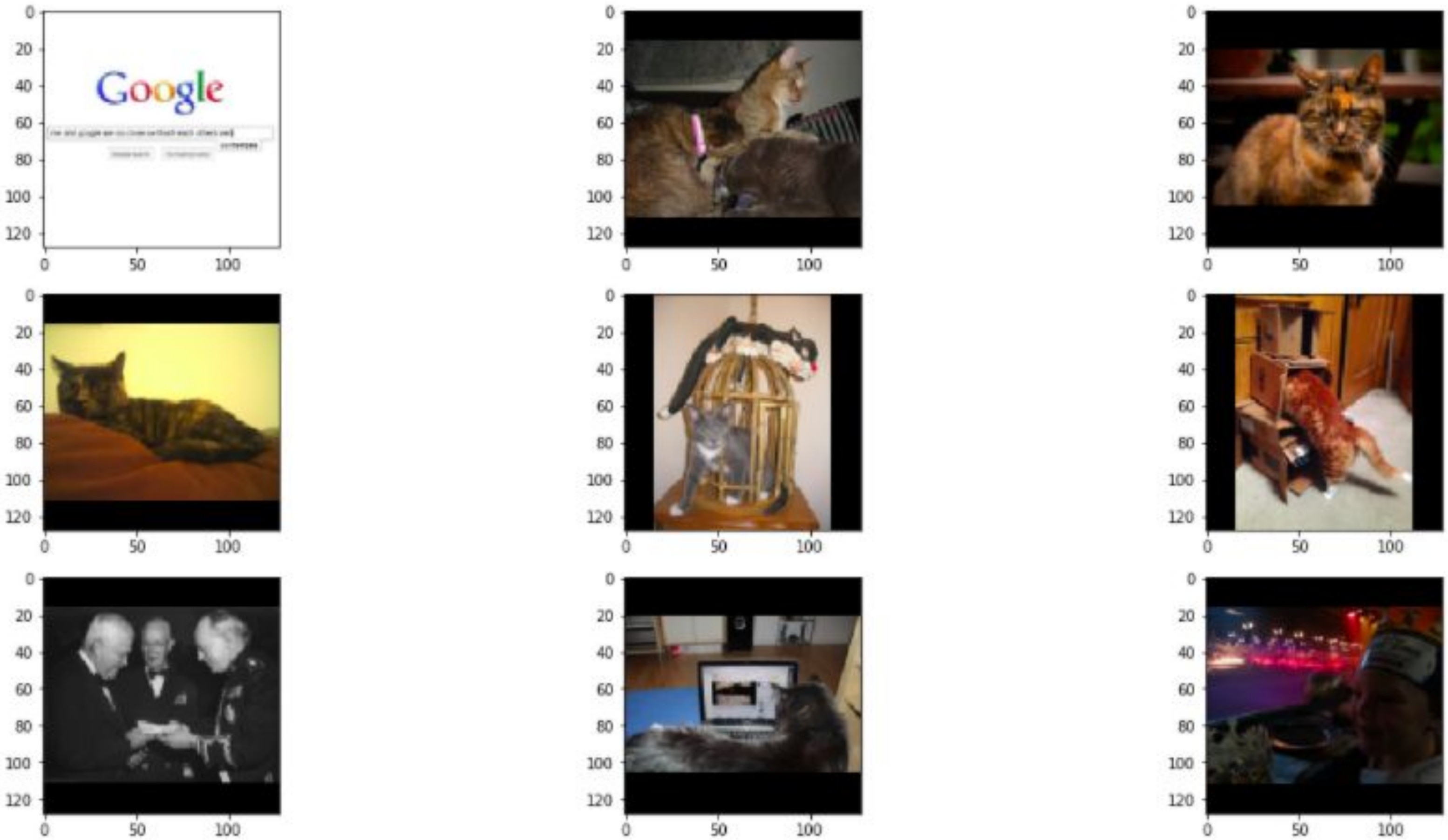
In [24]:
```python
# Show unsuccessful classifications:
show_inputs(DEBUG_DIR, miss, 3)
```

```
File names:
13306_0.1669_0.png
14869_0.1857_1.png
14848_0.1855_1.png
14937_0.1864_1.png
1414_0.0192_1.png
14151_0.1771_1.png
1346_0.0183_0.png
13897_0.1737_1.png
13937_0.1742_0.png
```



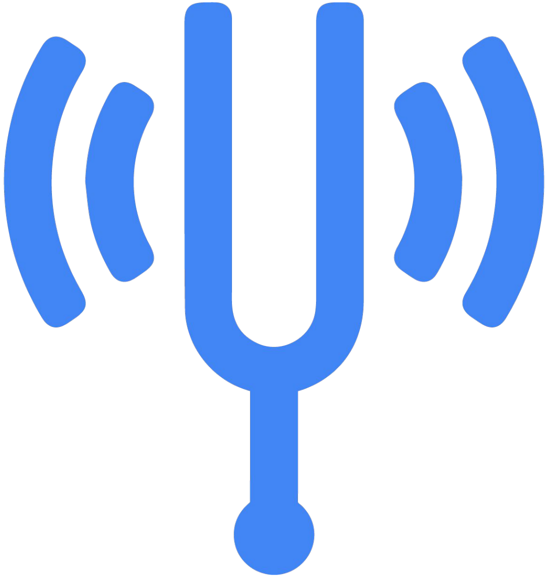
1346_0.0183_0.png
13897_0.1737_1.png
13937_0.1742_0.png



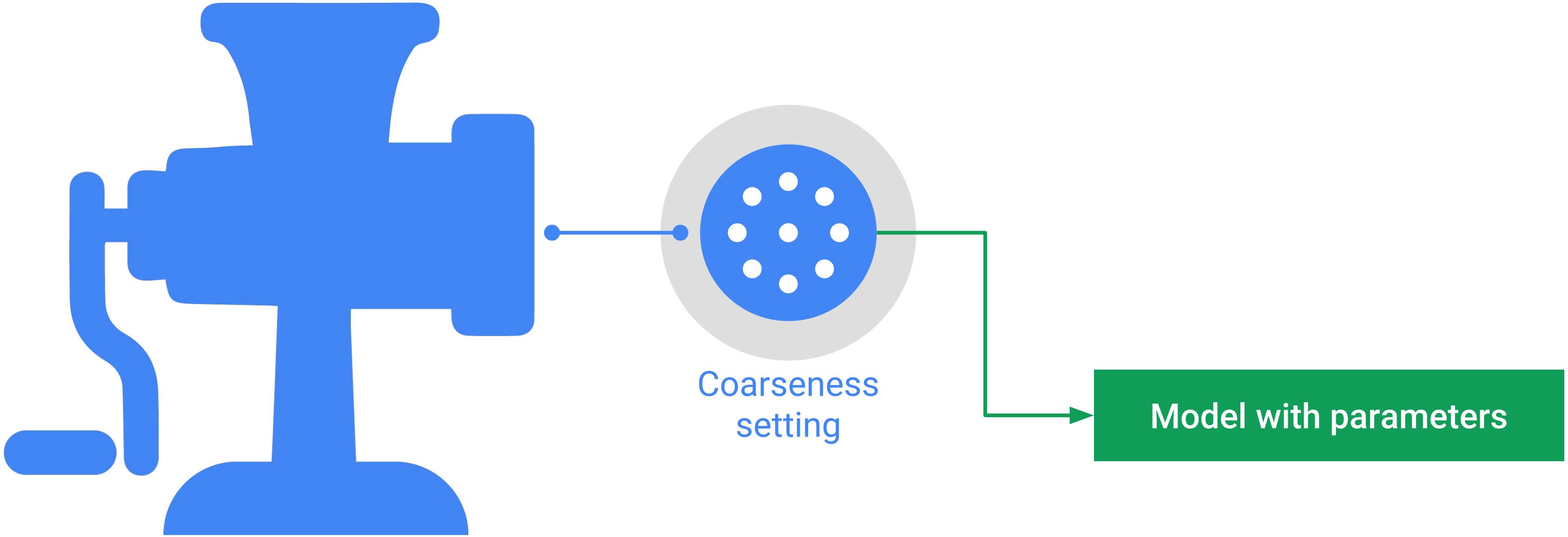
Tuning

You just ran an algorithm

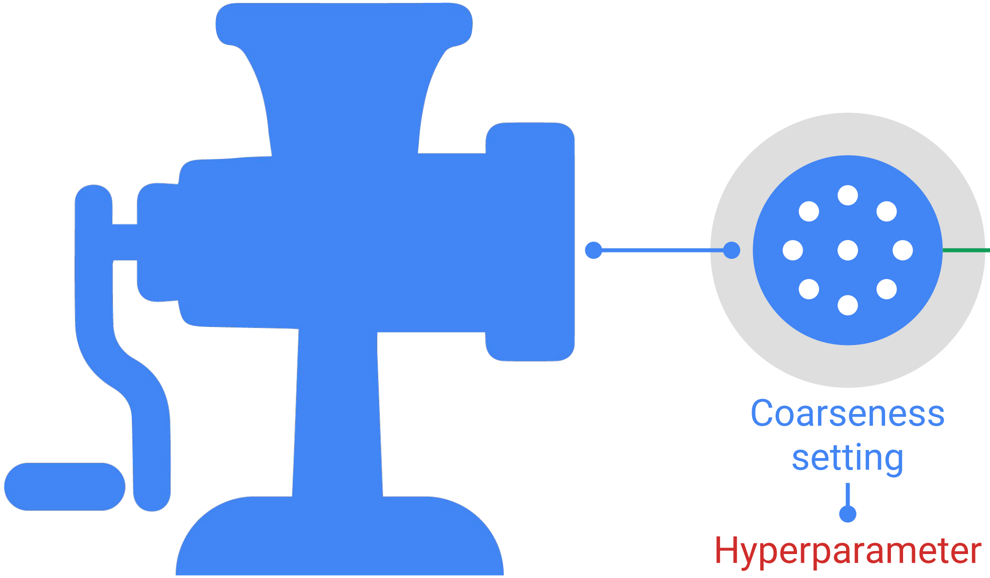
Hey, where did that setting you used come from?



Spot the hyperparameter



Spot the hyperparameter



Parameters:
set using the data

Hyperparameters:
set before the data

Model with parameters

Confidential & Proprietary

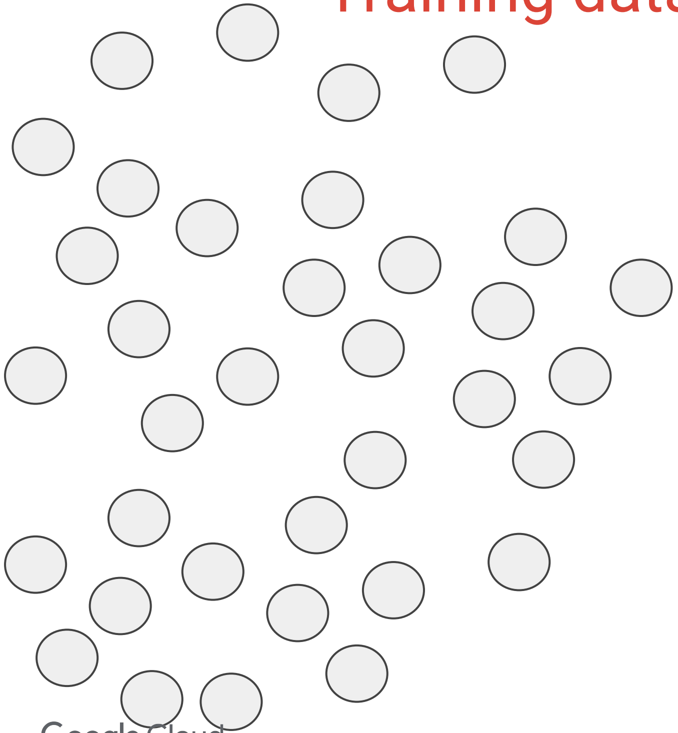
Tune your hyperparameters

* Examples: learning rate, SVM cost parameter, Bayesian prior parameters, mixing parameter, LASSO shrinkage parameter, etc. Regularized model? Quick, spot the hyperparameter!

If your algorithm has some numerical knobs and dials*, you'll need to tune them

Previous step

Training dataset



During the training step, you used your entire training dataset to try out different algorithms

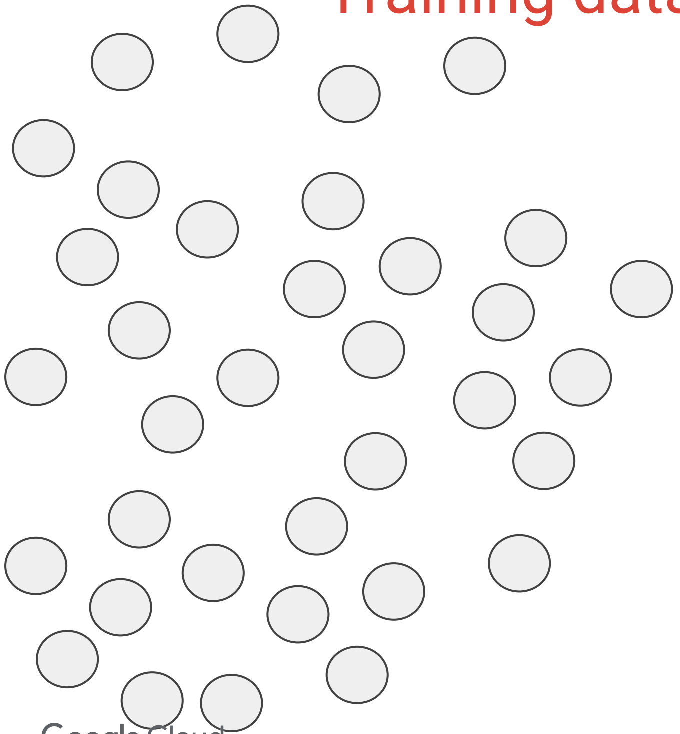
You found one you liked

Tune it

Confidential & Proprietary

Basic tuning (holdout method)

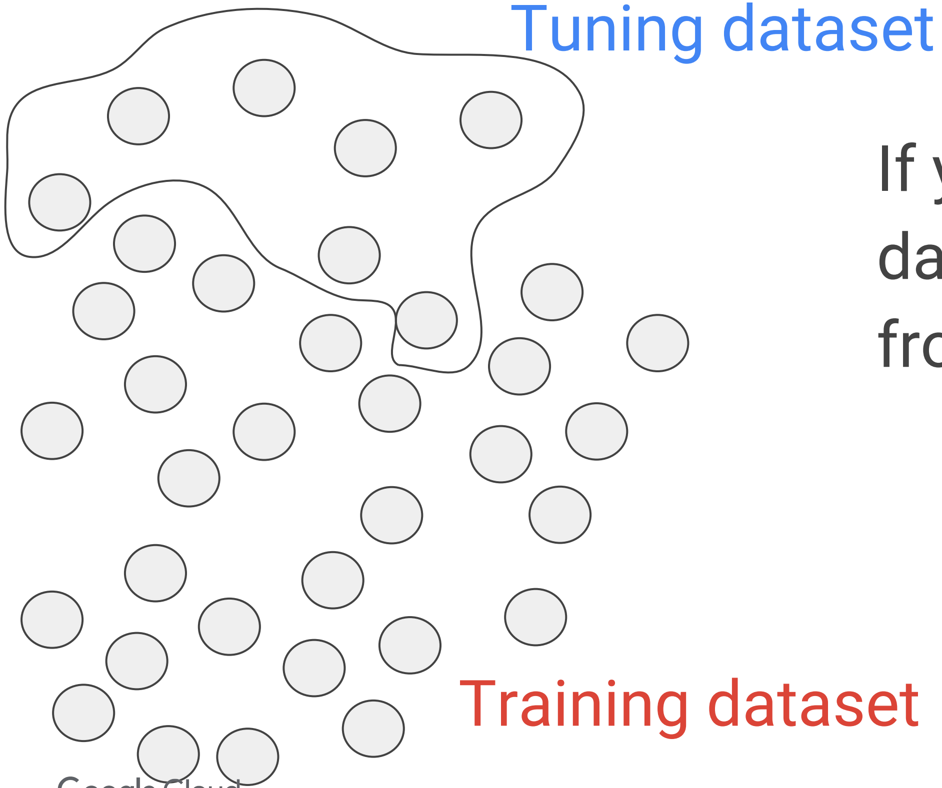
Training dataset



If you don't already have a tuning dataset,

Confidential & Proprietary

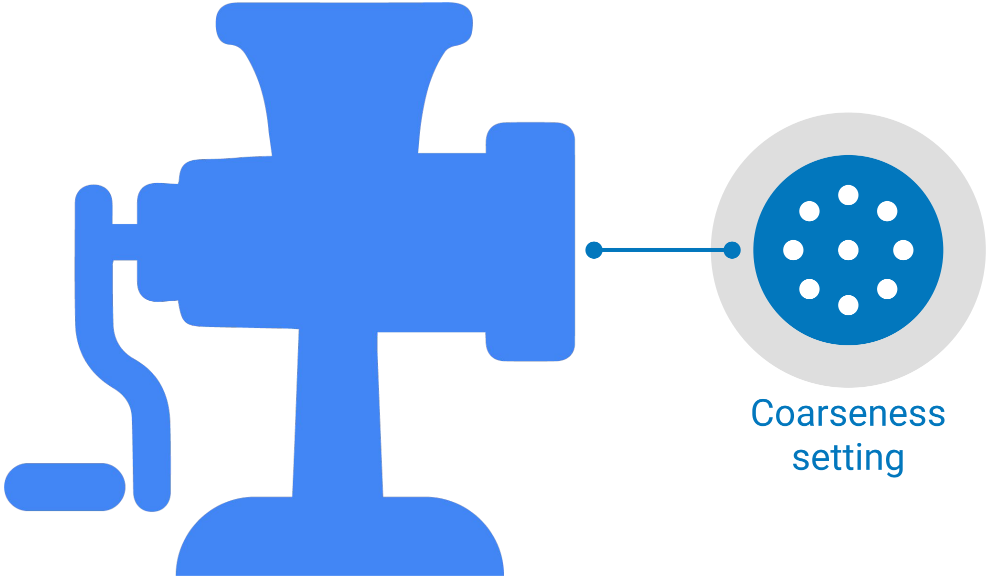
Basic tuning (holdout method)



If you don’t already have a tuning dataset, temporarily make one from your training data

Basic tuning (holdout method)

Permitted settings for hyperparameter

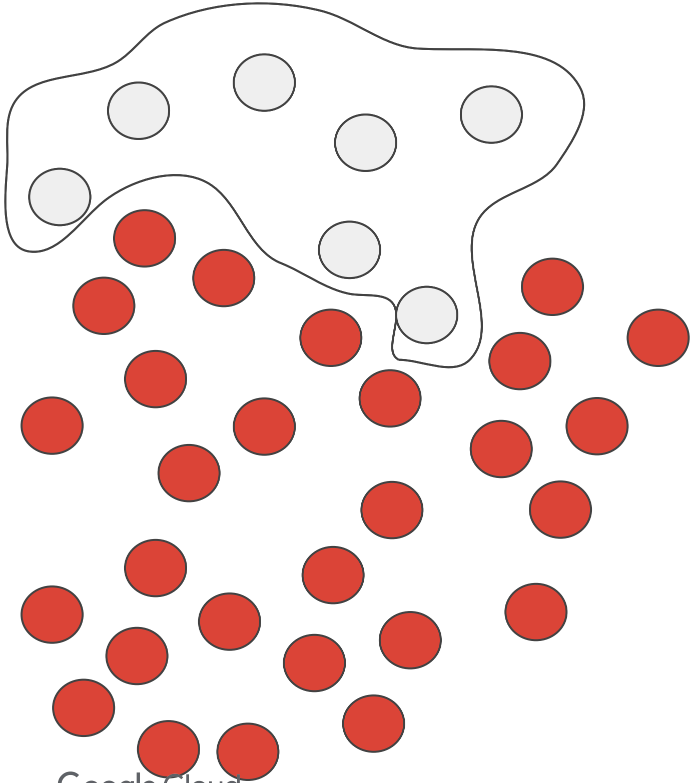


Basic tuning (holdout method)

hyperparameter = 1

Confidential & Proprietary

Basic tuning (holdout method)

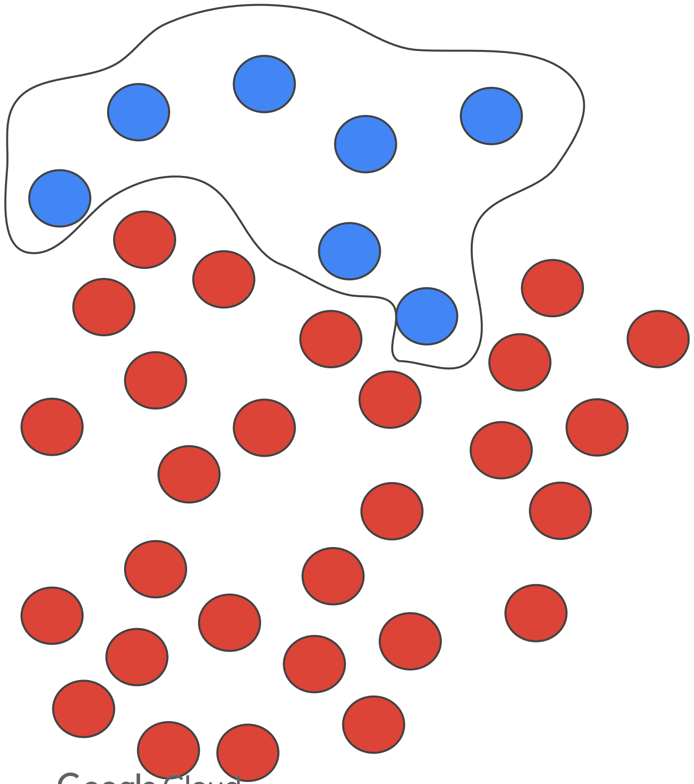


set hyperparameter = 1

Train model

Confidential & Proprietary

Basic tuning (holdout method)



set hyperparameter = 1

Train model

Evaluate performance

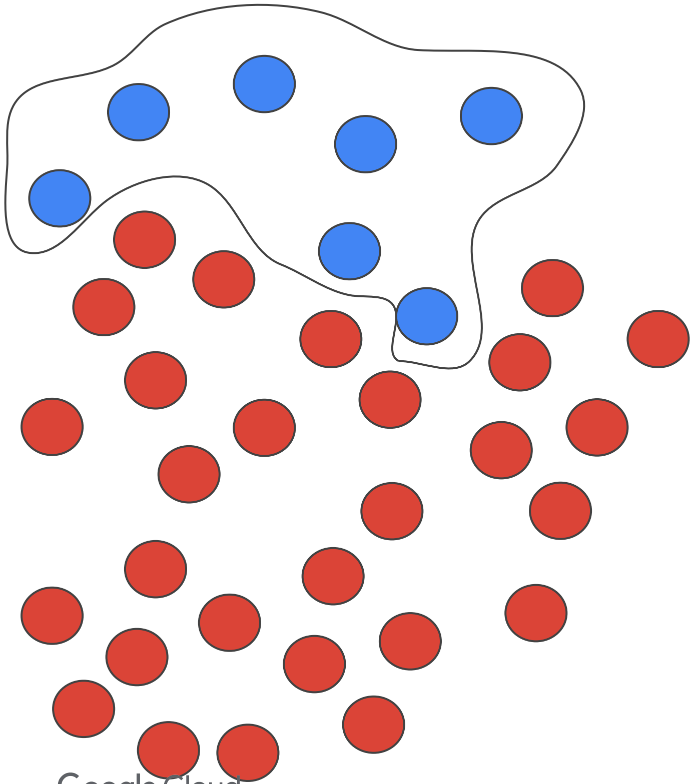
Store

Basic tuning (holdout method)

hyperparameter = 2

Confidential & Proprietary

Basic tuning (holdout method)



set hyperparameter = 2

Train model

Evaluate performance

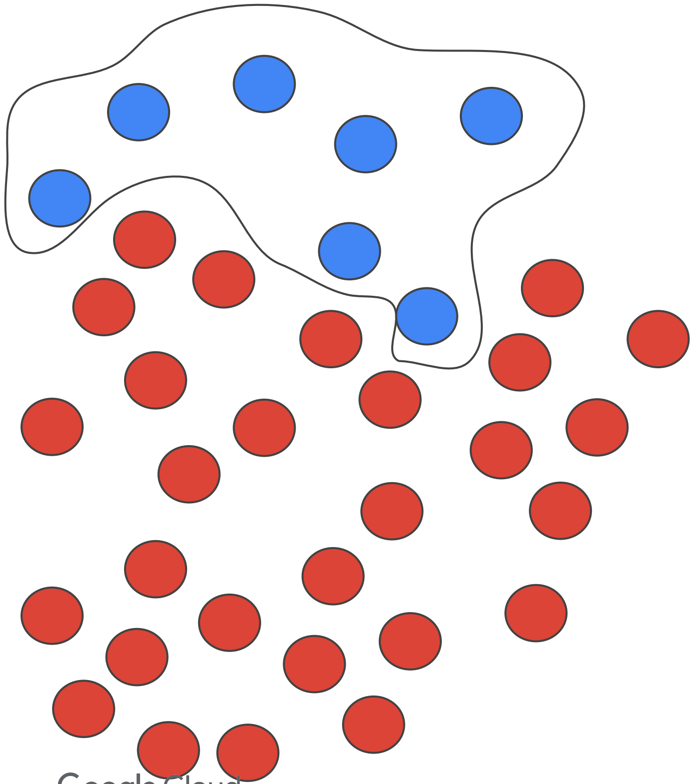
Store

Basic tuning (holdout method)

hyperparameter = 3

Confidential & Proprietary

Basic tuning (holdout method)



set hyperparameter = 3

Train model

Evaluate performance

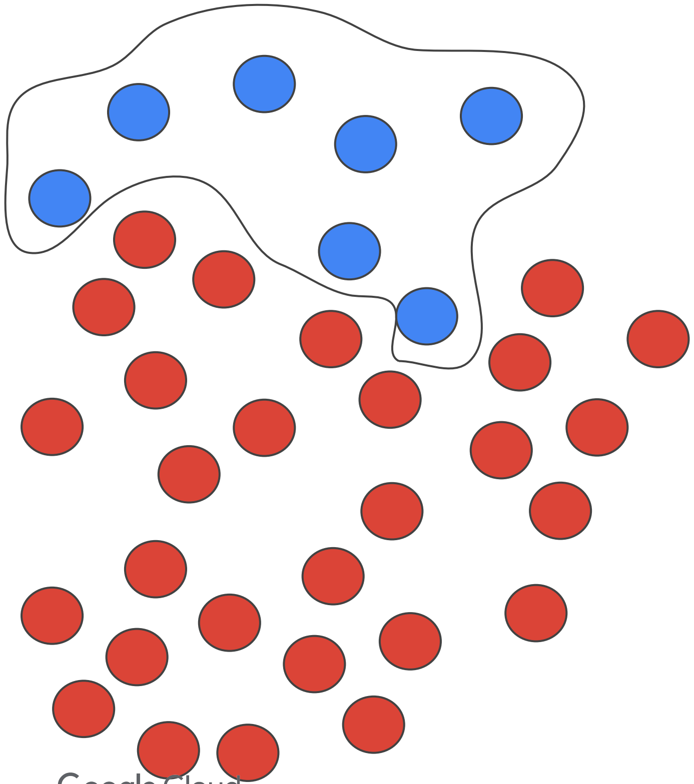
Store

Basic tuning (holdout method)

hyperparameter = 4

Confidential & Proprietary

Basic tuning (holdout method)



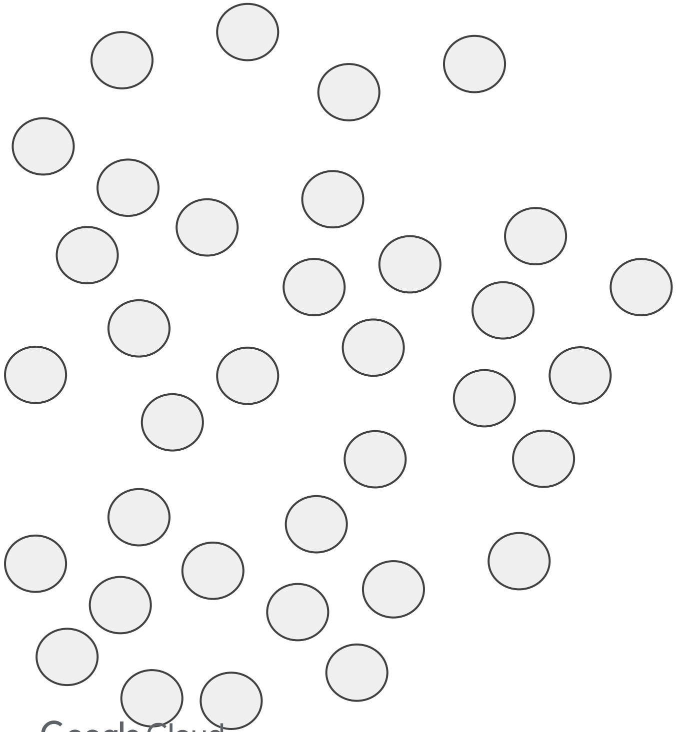
set hyperparameter = 4

Train model

Evaluate performance

Store

Basic tuning (holdout method)

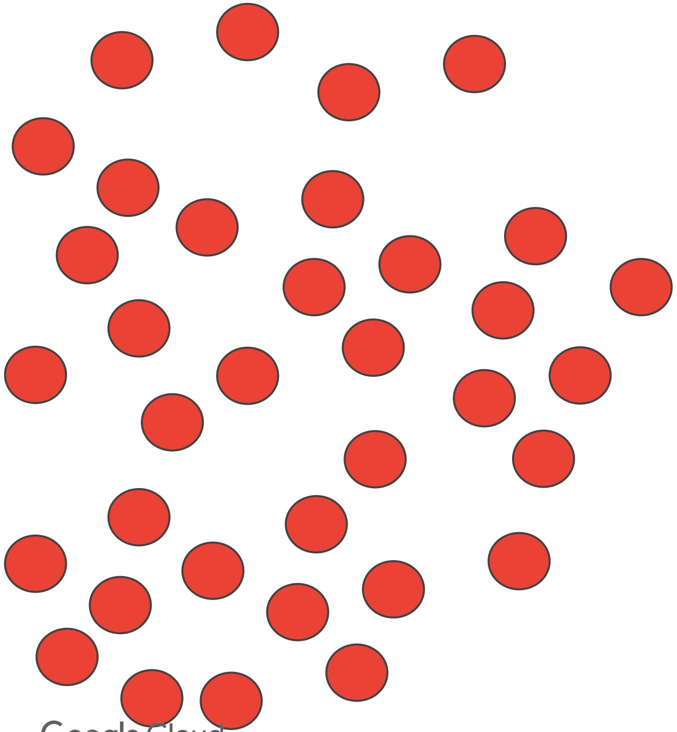


Choose the hyperparameter setting which gives you the best performance

Hyperparameter	Accuracy
1	66%
2	74%
3	94%
4	87%

Confidential & Proprietary

Basic tuning (holdout method)

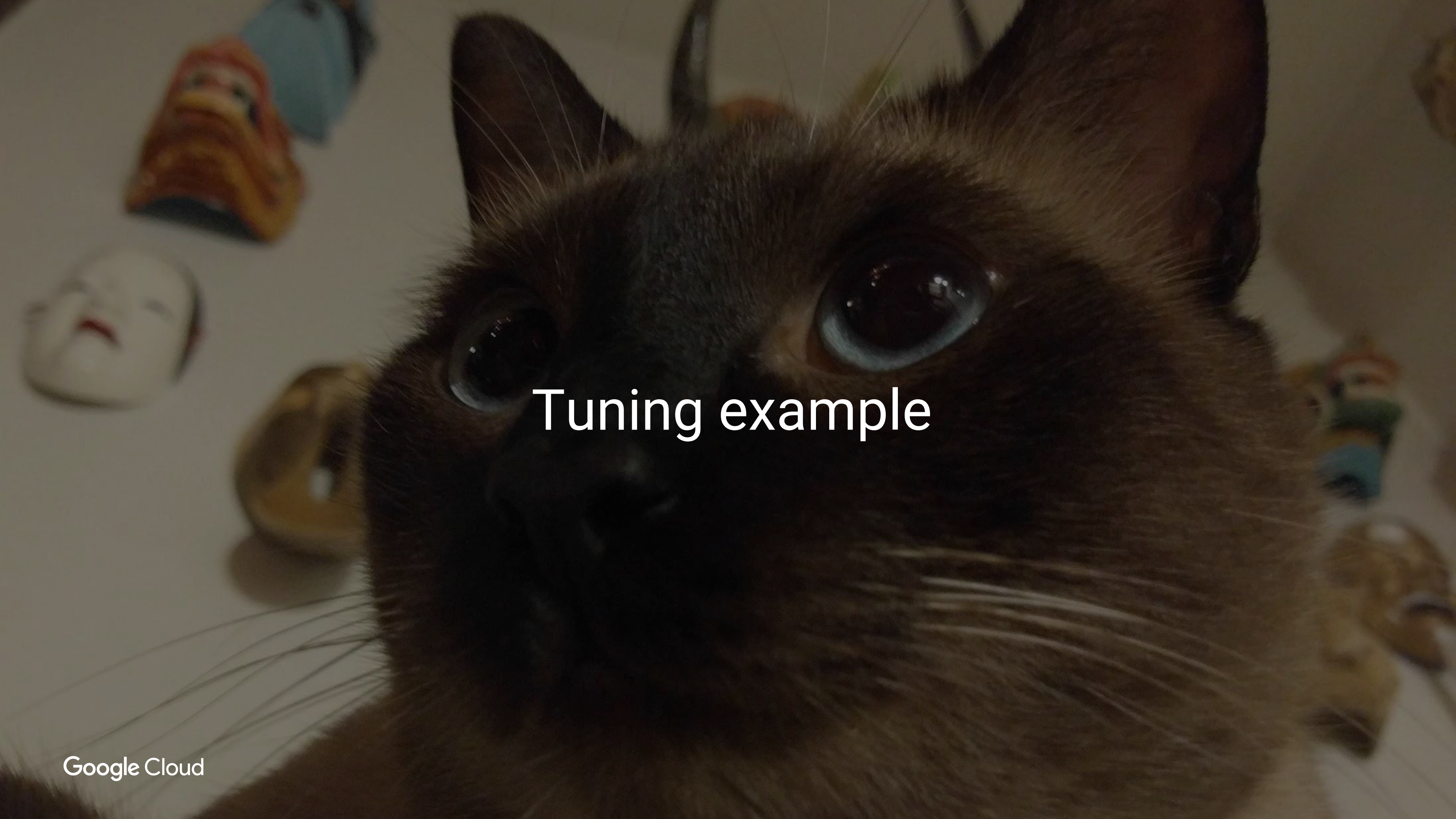


Re-run your algorithm with this setting on your training data

The result is your tuned model

Confidential & Proprietary

Tuning example



Tuning

I'll show you extremely simple tuning of the dropout rate. There are plenty of more sophisticated options (check out Cloud ML Engine!) but I'd like to show you that the principles are simple. We'll just step through an array of options and see which dropout setting gets you the best accuracy, then we'll select that one when we train with more data.

In [340]:

```
# Disable TF verbose output:
tf.logging.set_verbosity(tf.logging.FATAL)

# Get output:
dropouts = np.array([])
accuracies = np.array([])
for i in range(9):
    tune_output_dir = OUTPUT_DIR + 'dropout0.' + str(i + 1) + '/'
    tune_dropout = (float(i) + 1) / 10
    print("It's {:%H:%M} in London".format(datetime.datetime.now()) + ' --- Dropout setting is '
    # Try a new dropout setting for TF Estimator:
    estimator = tf.estimator.Estimator(model_fn=generate_model_fn(tune_dropout),
                                       model_dir=tune_output_dir,
                                       config=RunConfig(
                                           save_checkpoints_secs=CHECKPOINT_PERIOD_SECS,
                                           keep_checkpoint_max=20,
                                           save_summary_steps=100,
                                           log_step_count_steps=100)
                                       )
    # Train it!
    learn_runner.run(generate_experiment_fn(), tune_output_dir)
    # Identify the model version:
    tuned_model = tune_output_dir + 'model.ckpt-' + str(TRAIN_STEPS)
    # Output predicted and observed labels:
    predicted = cat_finder(DEBUG_DIR, model_version=tuned_model)
```

```
    # Train it!
    learn_runner.run(generate_experiment_fn(), tune_output_dir)
    # Identify the model version:
    tuned_model = tune_output_dir + 'model.ckpt-' + str(TRAIN_STEPS)
    # Output predicted and observed labels:
    predicted = cat_finder(DEBUG_DIR, model_version=tuned_model)
    observed = get_labels(DEBUG_DIR)
    # Compute performance metric:
    accuracy = get_accuracy(truth=observed, predictions=predicted)
    print('Accuracy is: ' + str(accuracy))
    # Append to array:
    dropouts = np.append(dropouts, tune_dropout)
    accuracies = np.append(accuracies, accuracy)
```

```
It's 03:28 in London --- Dropout setting is 0.1
Accuracy is: 0.7
It's 03:31 in London --- Dropout setting is 0.2
Accuracy is: 0.67
It's 03:33 in London --- Dropout setting is 0.3
Accuracy is: 0.63
It's 03:35 in London --- Dropout setting is 0.4
Accuracy is: 0.64
It's 03:37 in London --- Dropout setting is 0.5
Accuracy is: 0.68
It's 03:39 in London --- Dropout setting is 0.6
Accuracy is: 0.68
It's 03:42 in London --- Dropout setting is 0.7
Accuracy is: 0.65
It's 03:44 in London --- Dropout setting is 0.8
Accuracy is: 0.62
It's 03:46 in London --- Dropout setting is 0.9
Accuracy is: 0.63
```

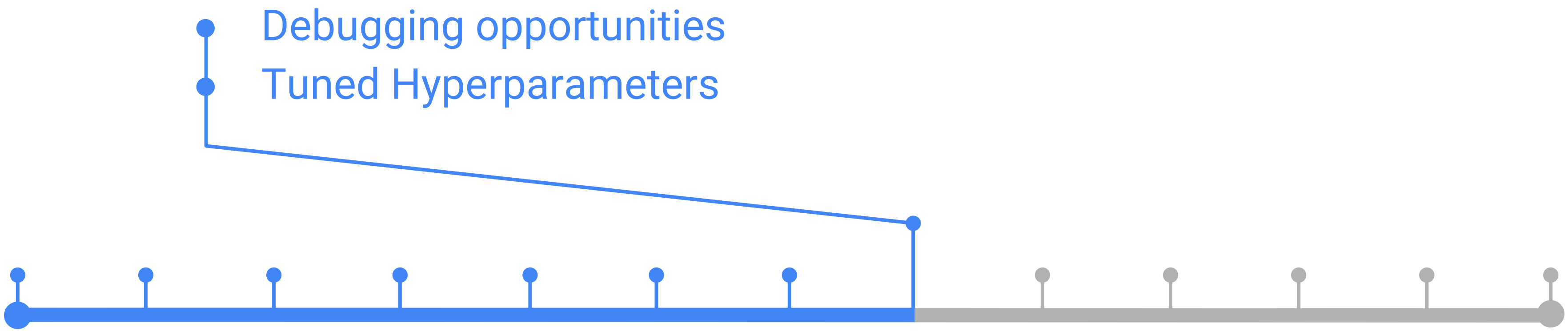
Key message

!

Allocate some training data for debugging & tuning, either at the start of the project or on the fly

1. Debugging gets you insights
2. Tuning saves you from poor hyperparameter choices

Step 7 is finished | You specifically allocated data to get:



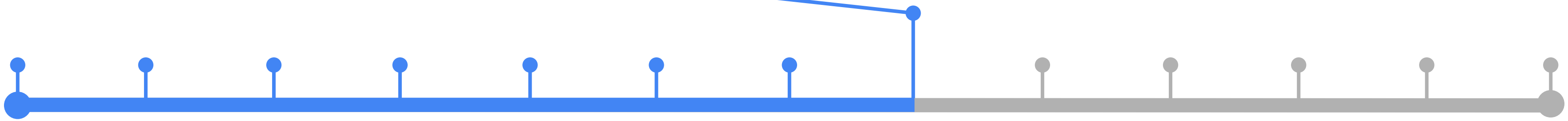
Chief actors:

Engineer

Analyst

Step 7 is finished | You specifically allocated data to get:

- Perhaps looked at images we failed on (shown in next chapter)
- Dropout is set to 0.6



Chief actors:

Engineer

Analyst

Google Cloud

Confidential & Proprietary