

# Deep Learning Walkthrough - 07

Code in [github.com/google-aai/sc17](https://github.com/google-aai/sc17)

**Cassie Kozyrkov**

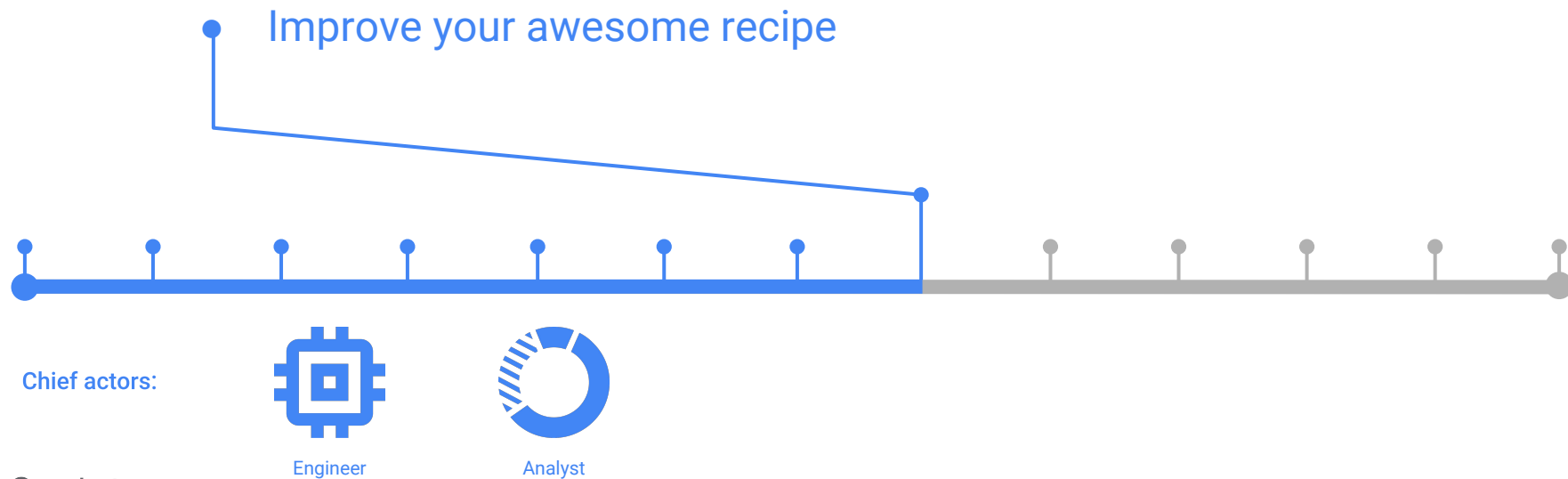
Chief Decision Scientist, Google Cloud

GitHub: [kozyrkov](#); Twitter: [@quaesita](#)

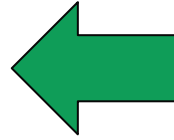
Google Cloud



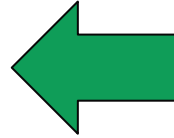
# Step 7 | Tune and debug



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



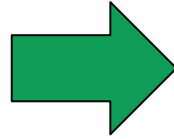
Training-ish



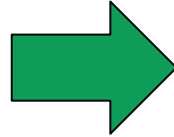
Validation-ish



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



Tuning



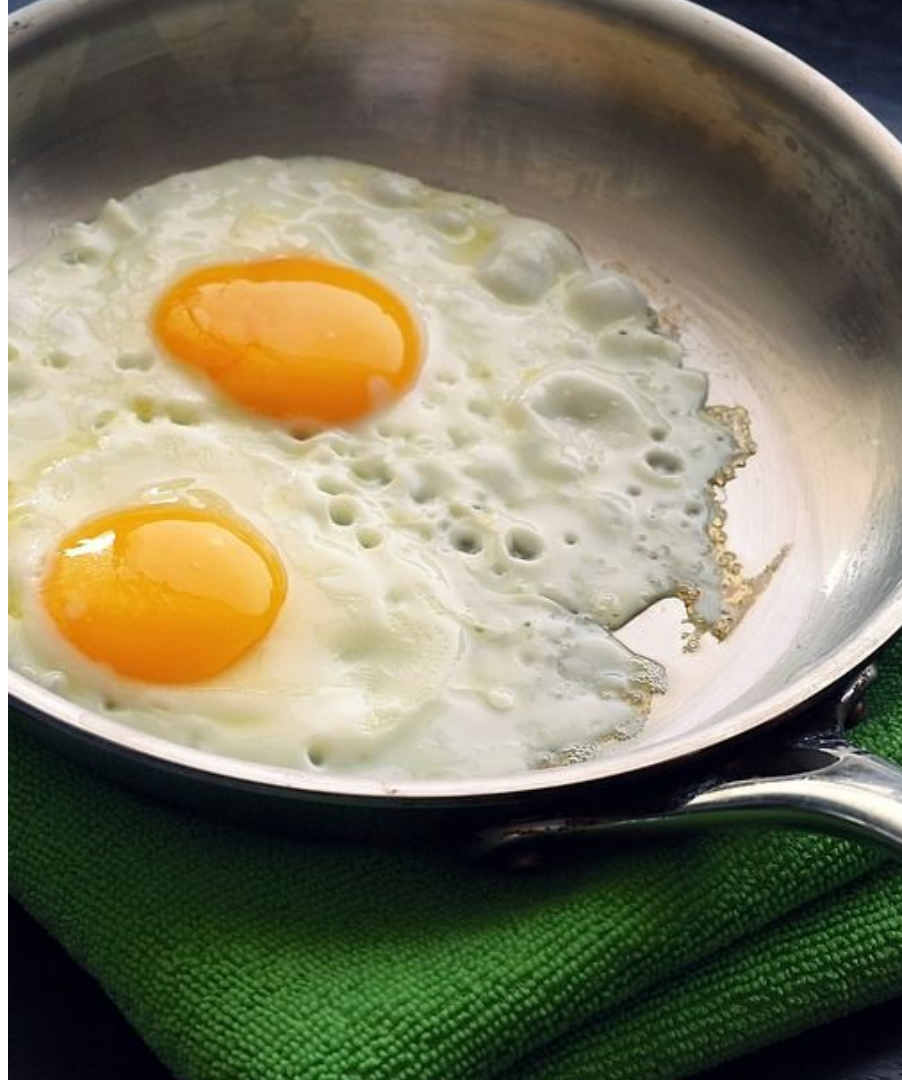
Debugging



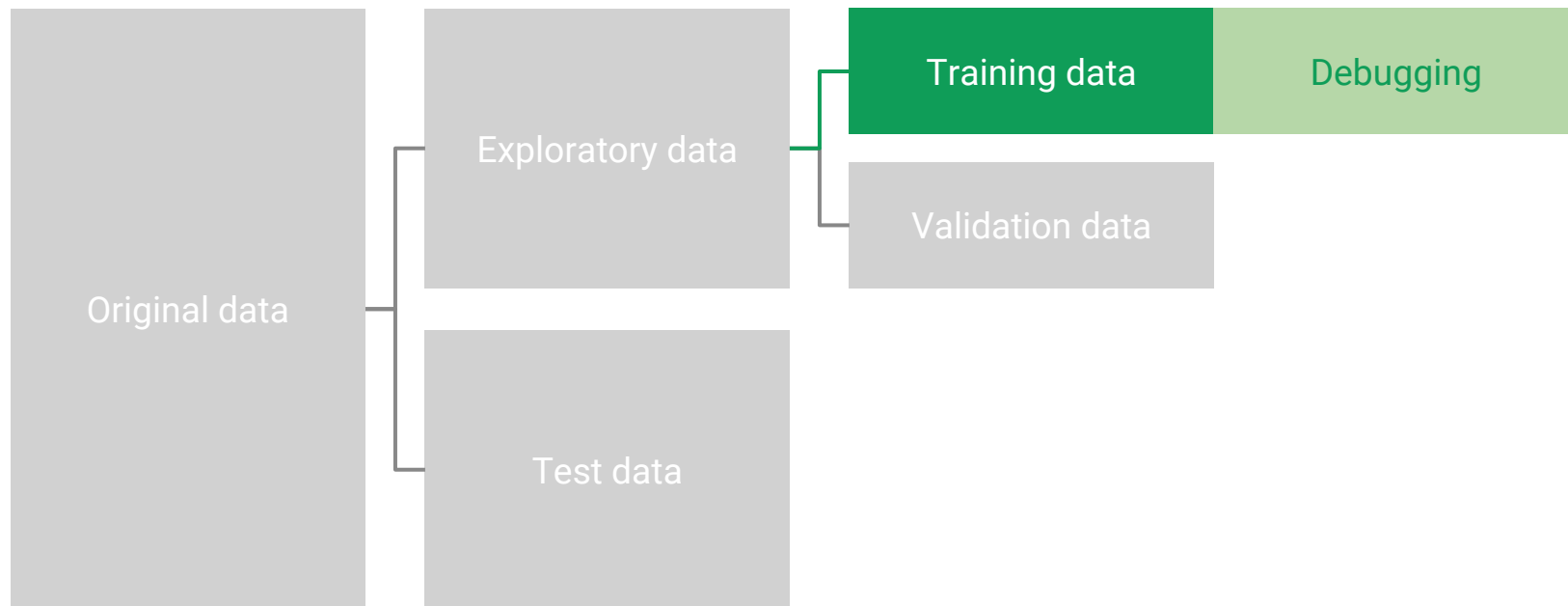
# In practice



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

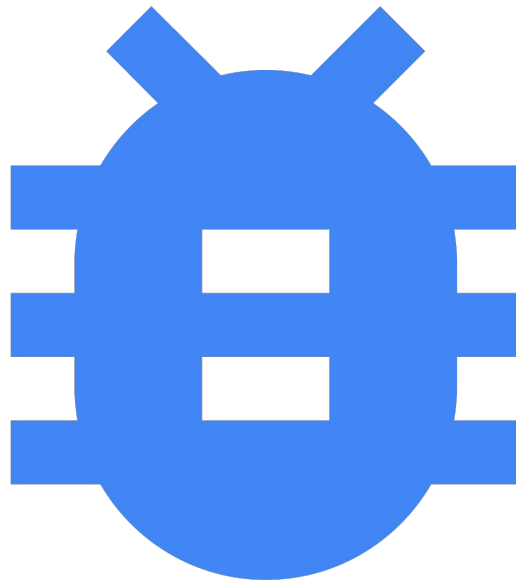


# Debugging


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







# 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

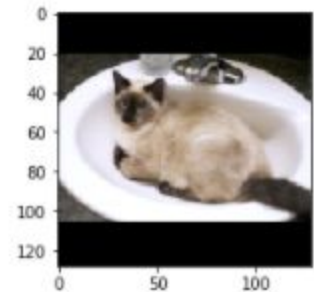
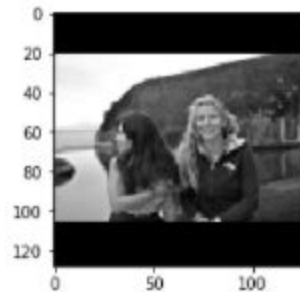
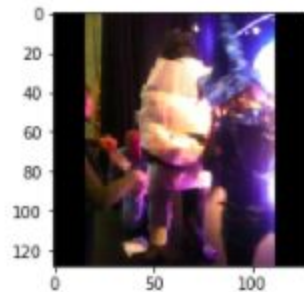
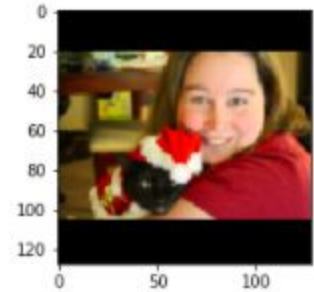
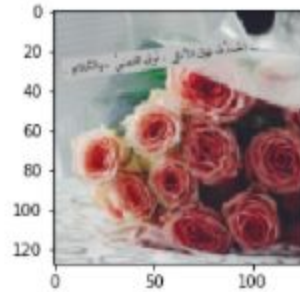
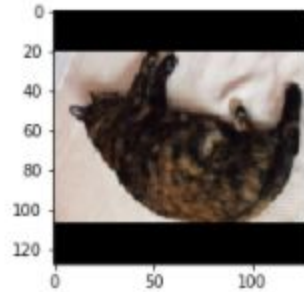
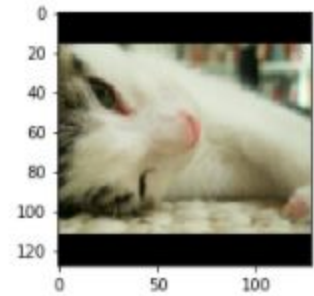
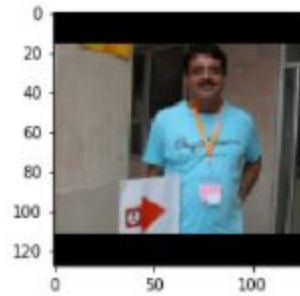
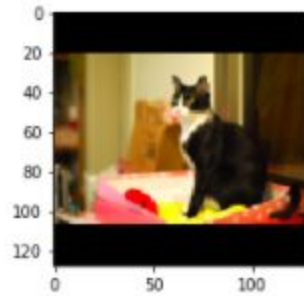
13713\_0.1715\_1.png

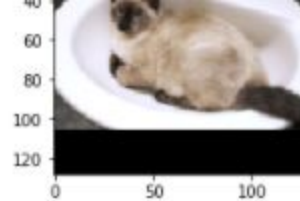
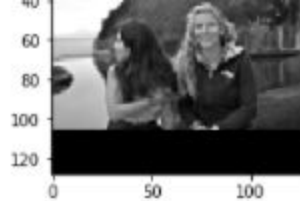
13824\_0.1728\_0.png

13834\_0.1729\_0.png

14281\_0.1784\_0.png

14978\_0.1870\_1.png





Out[23]: 128

```
In [24]: # 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

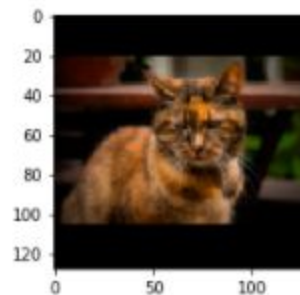
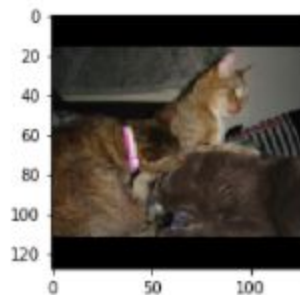
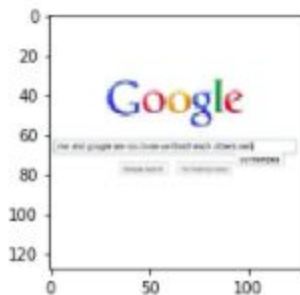
1414\_0.0192\_1.png

14151\_0.1771\_1.png

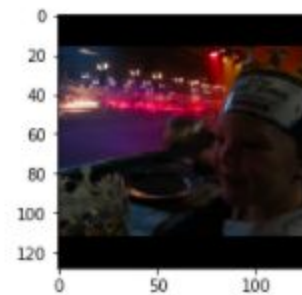
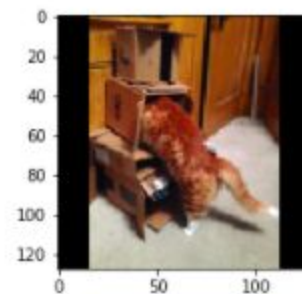
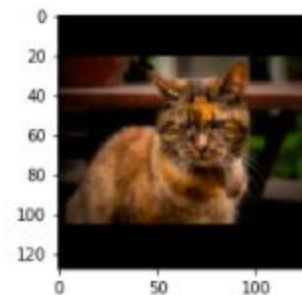
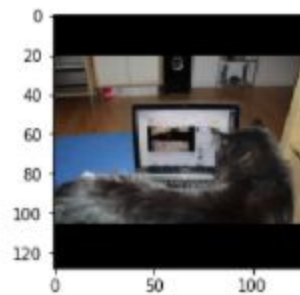
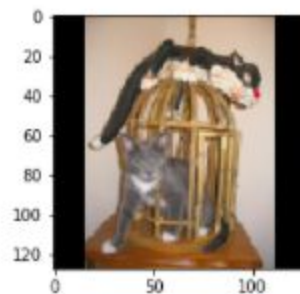
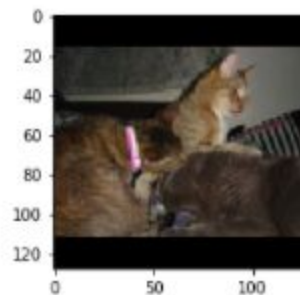
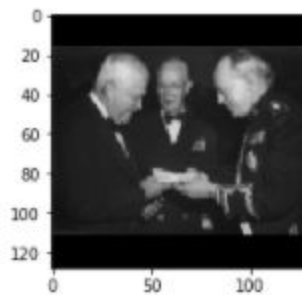
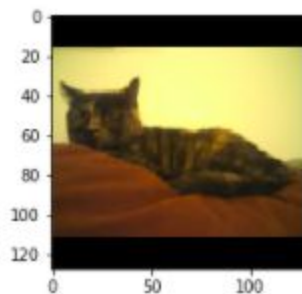
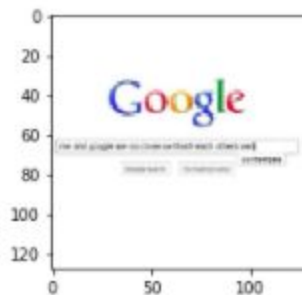
1346\_0.0183\_0.png

13897\_0.1737\_1.png

13937\_0.1742\_0.png



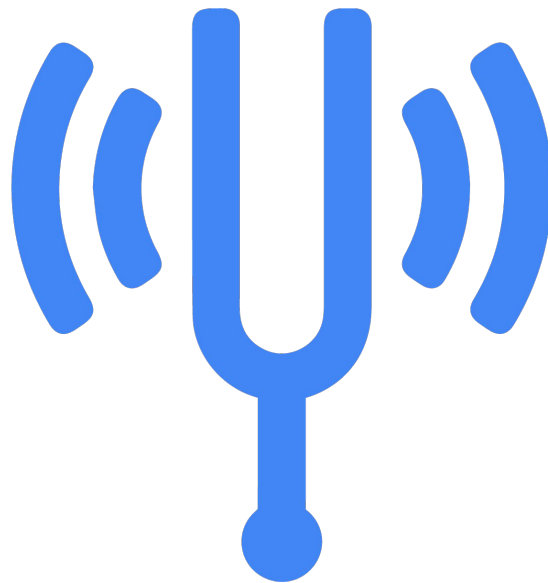
1346\_0.0183\_0.png  
13897\_0.1737\_1.png  
13937\_0.1742\_0.png



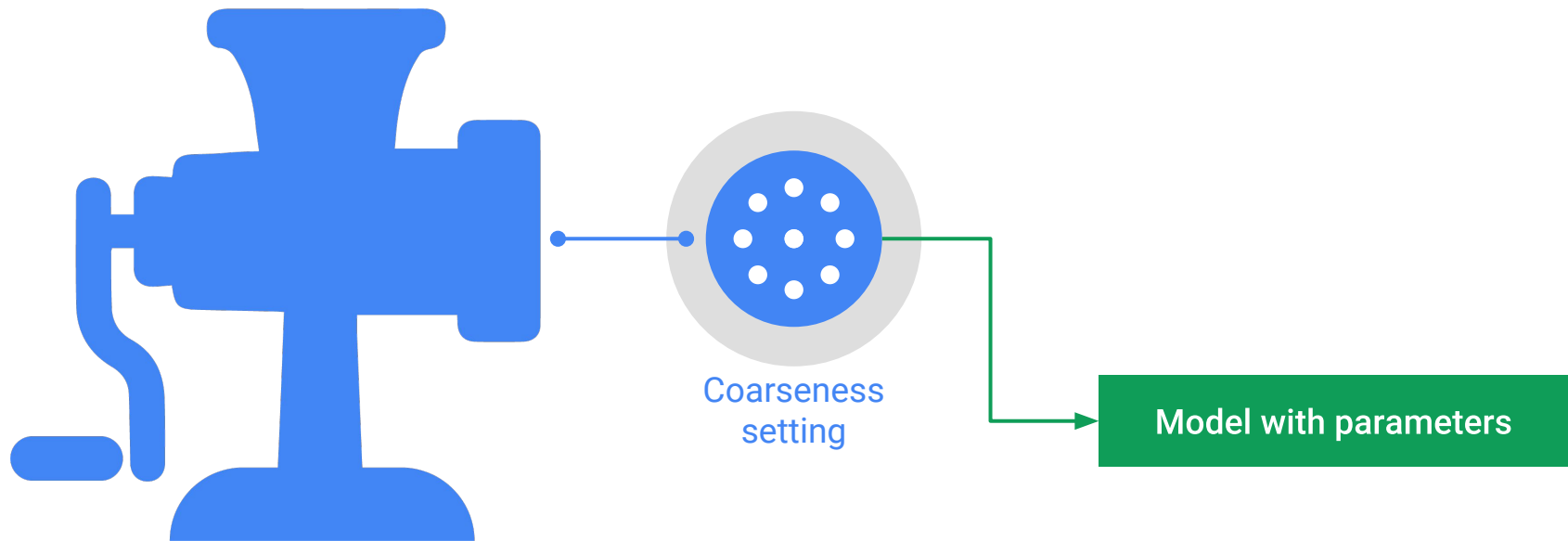
# Tuning

You just ran an algorithm

Hey, where did that setting you used  
come from?

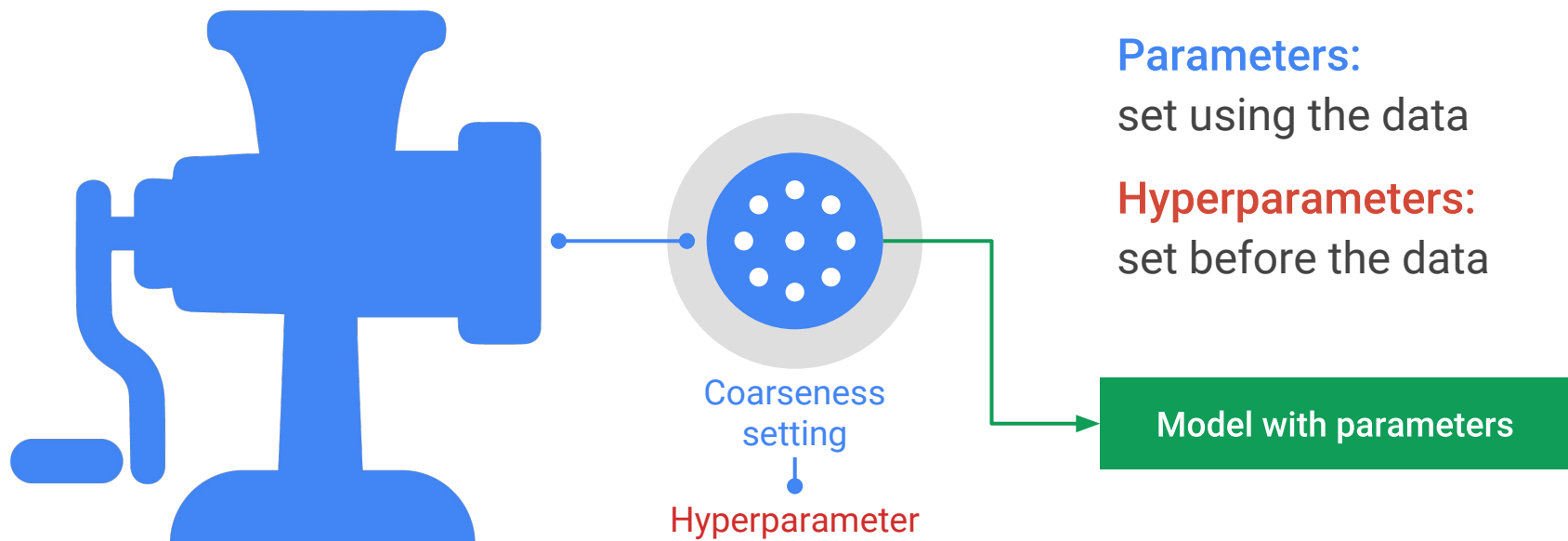


# Spot the hyperparameter



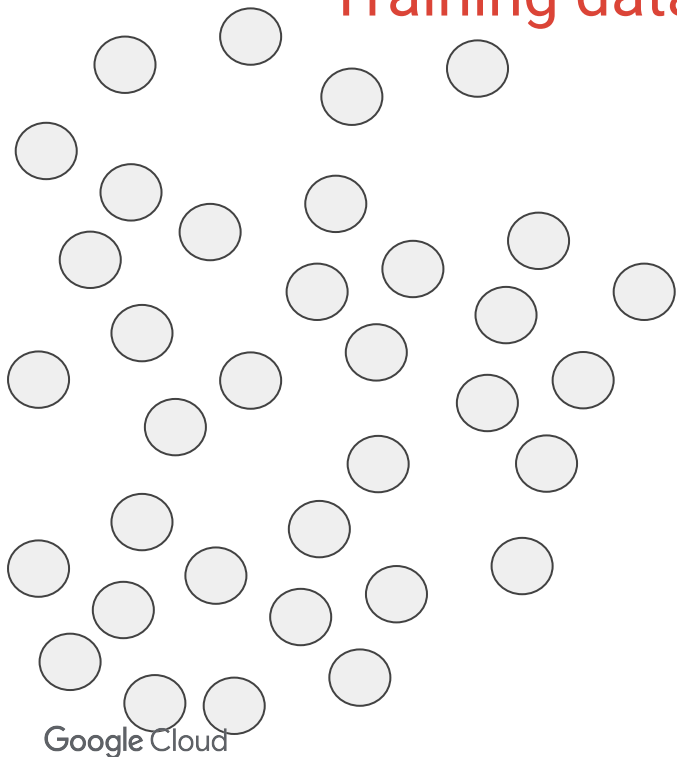


# Spot the hyperparameter



# Previous step

Training dataset



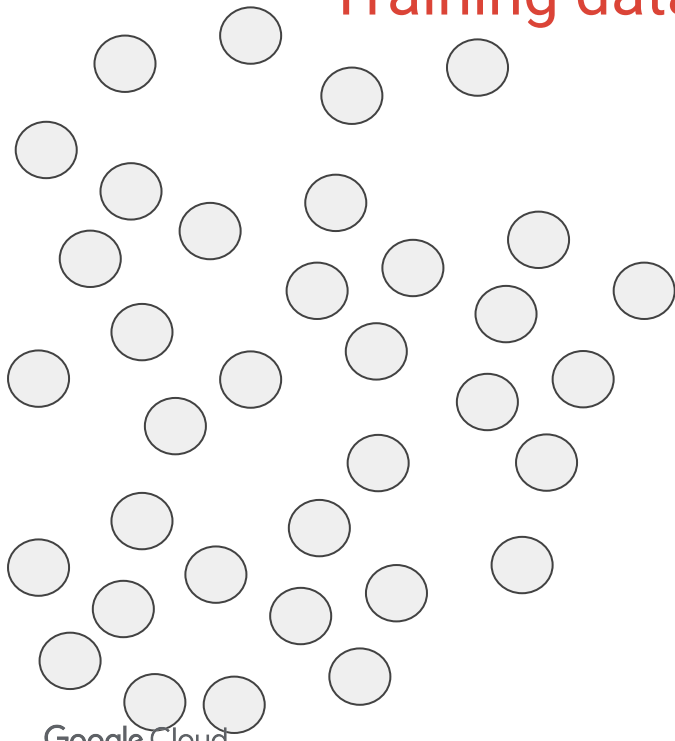
During the training step, you used your entire training dataset to try out different algorithms

You found one you liked

Tune it

# Basic tuning (holdout method)

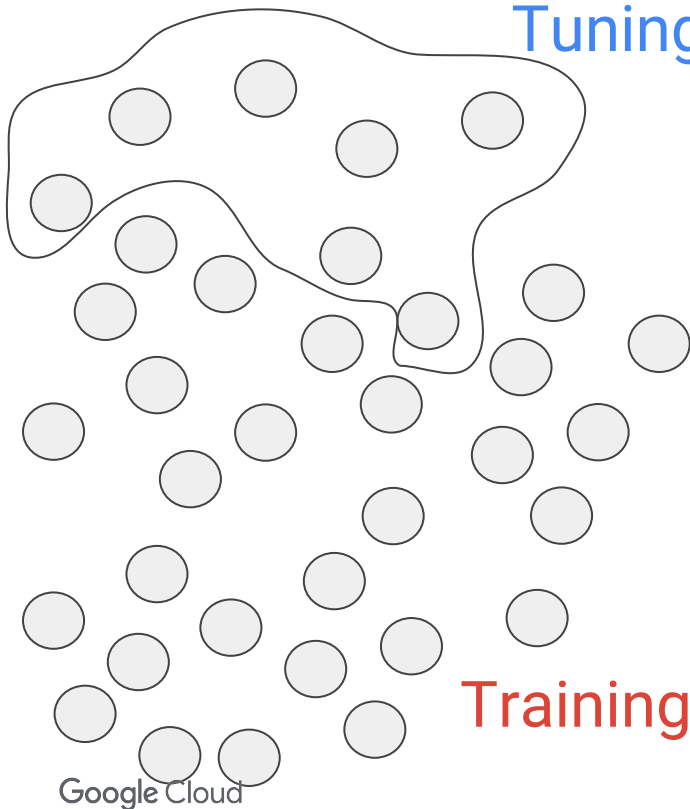
Training dataset



If you don't already have a tuning dataset,

# Basic tuning (holdout method)

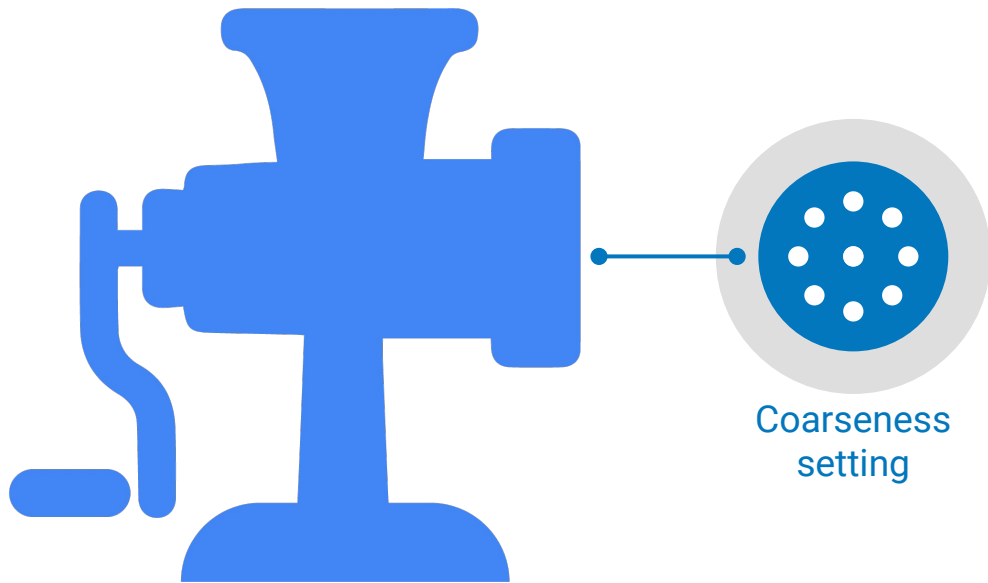
Tuning dataset



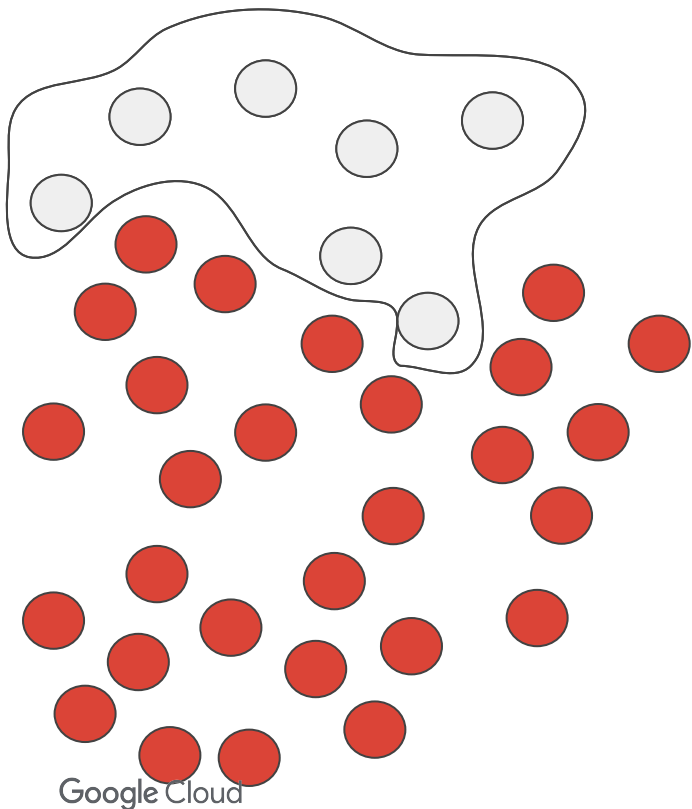
If you don't already have a tuning dataset, temporarily make one from your training data

# Basic tuning (holdout method)

Permitted settings  
for hyperparameter



# Basic tuning (holdout method)

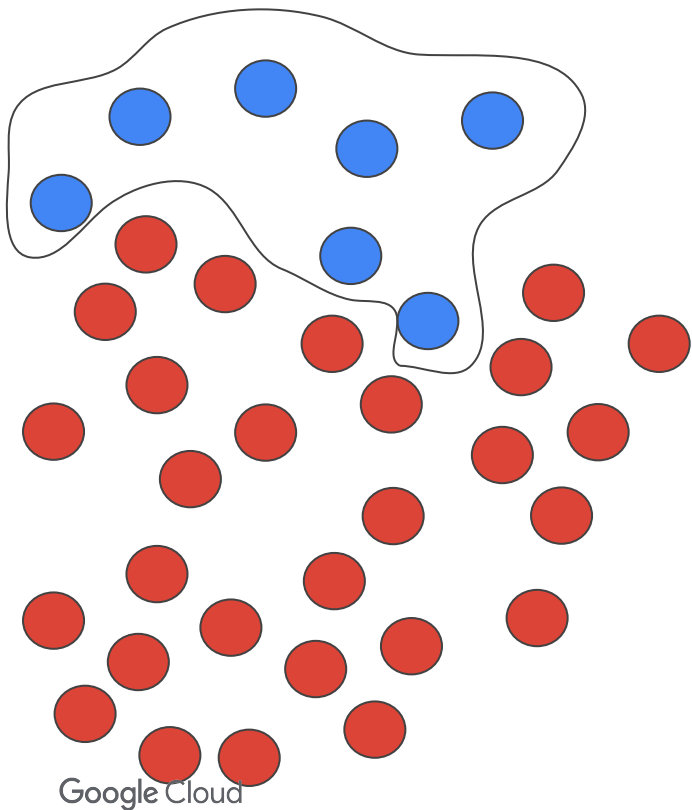


set hyperparameter = 1

Train model



# Basic tuning (holdout method)



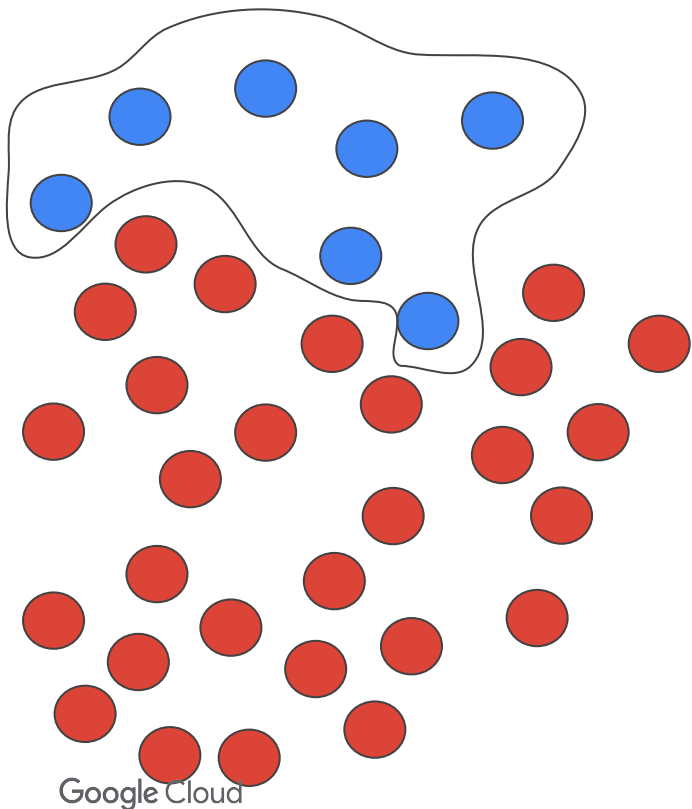
set hyperparameter = 1

Train model

Evaluate performance

Store

# Basic tuning (holdout method)



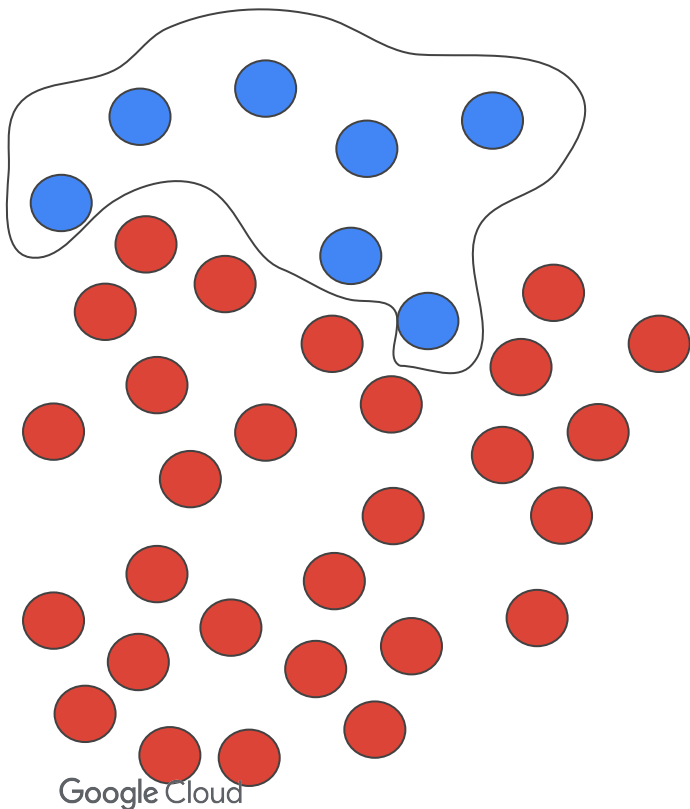
set hyperparameter = 2

Train model

Evaluate performance

Store

# Basic tuning (holdout method)



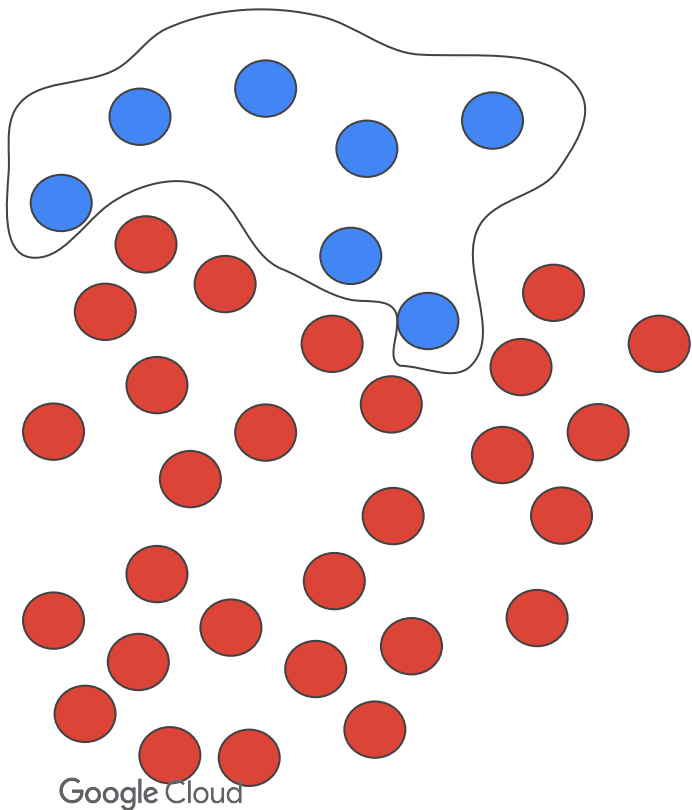
set hyperparameter = 3

Train model

Evaluate performance

Store

# Basic tuning (holdout method)



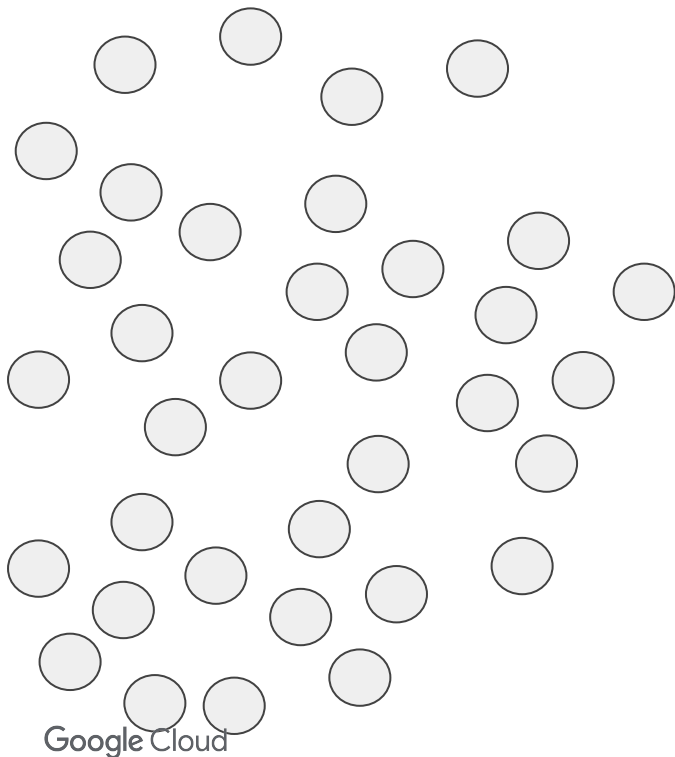
set hyperparameter = 4

Train model

Evaluate performance

Store

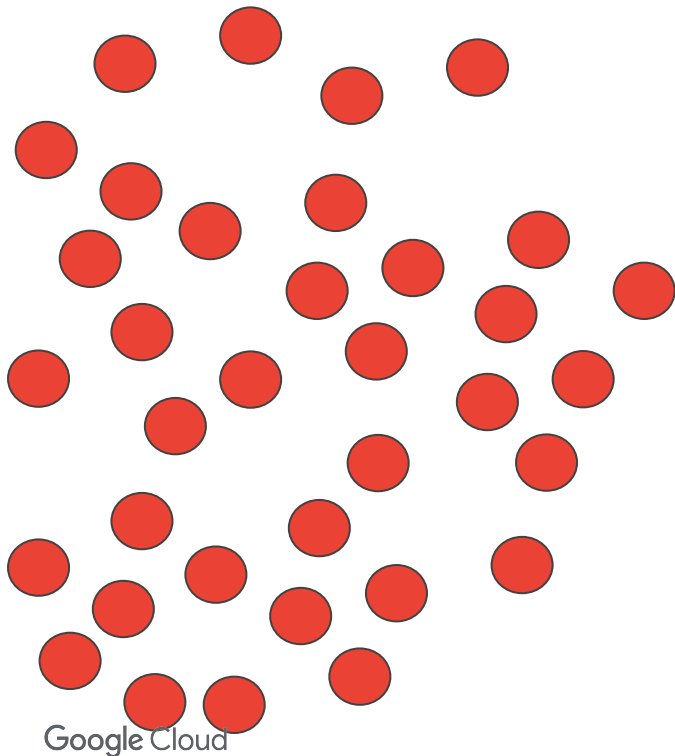
# Basic tuning (holdout method)



Choose the hyperparameter setting which gives you the **best performance**

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

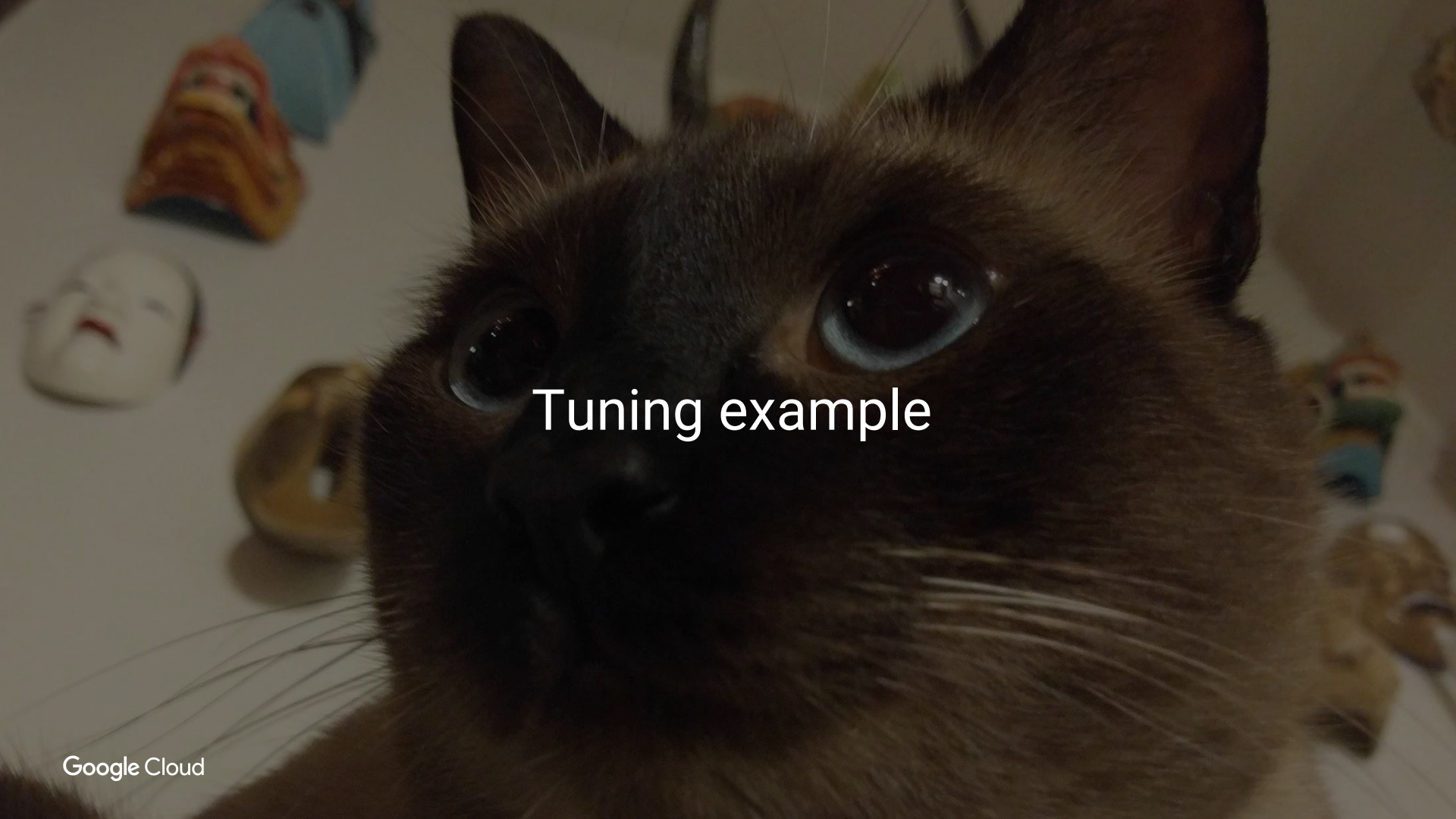
# Basic tuning (holdout method)



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

The result is your **tuned model**





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

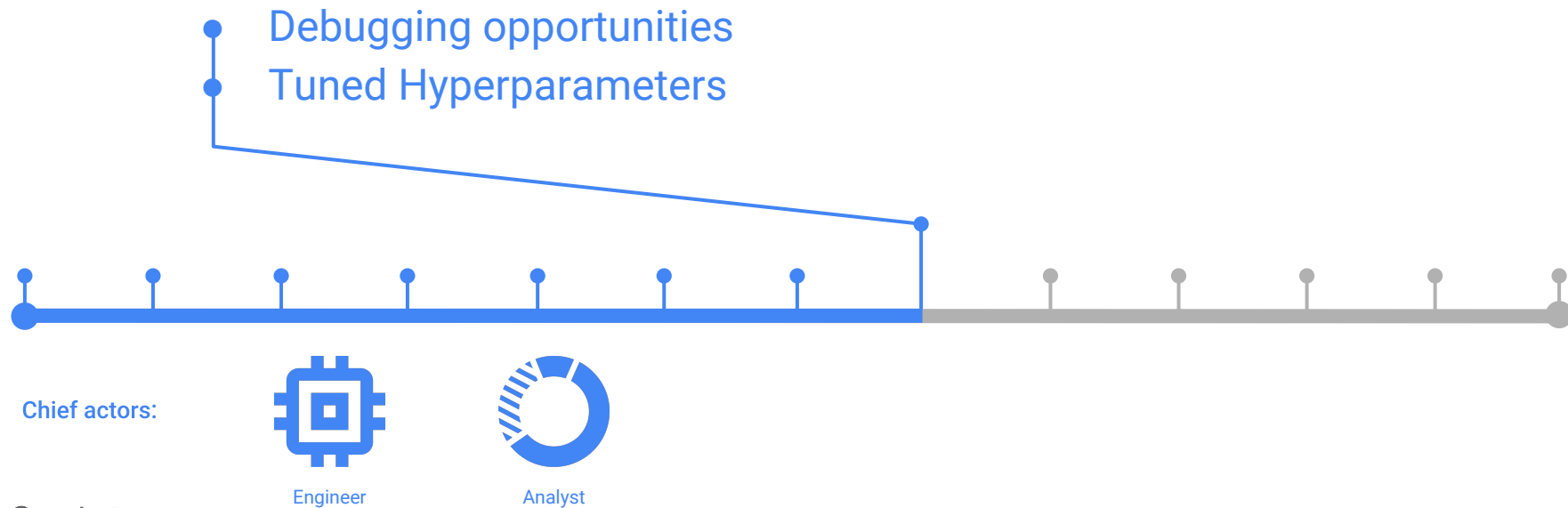


Allocate some training data for debugging & tuning, either at the start of the project or on the fly

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



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



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

