Big data, machine learning and maps: lessons learned on aerial imagery

Olivia Wilson
Research Scientist

Olivia.Wilson@os.uk
@oew1v07

Ordnance Survey

OS is known for

- **Accurate, authoritative data**

  When it needs to be right

- **Trusted and respected worldwide**

  A domestic focus with international reach

- **225 years experience**

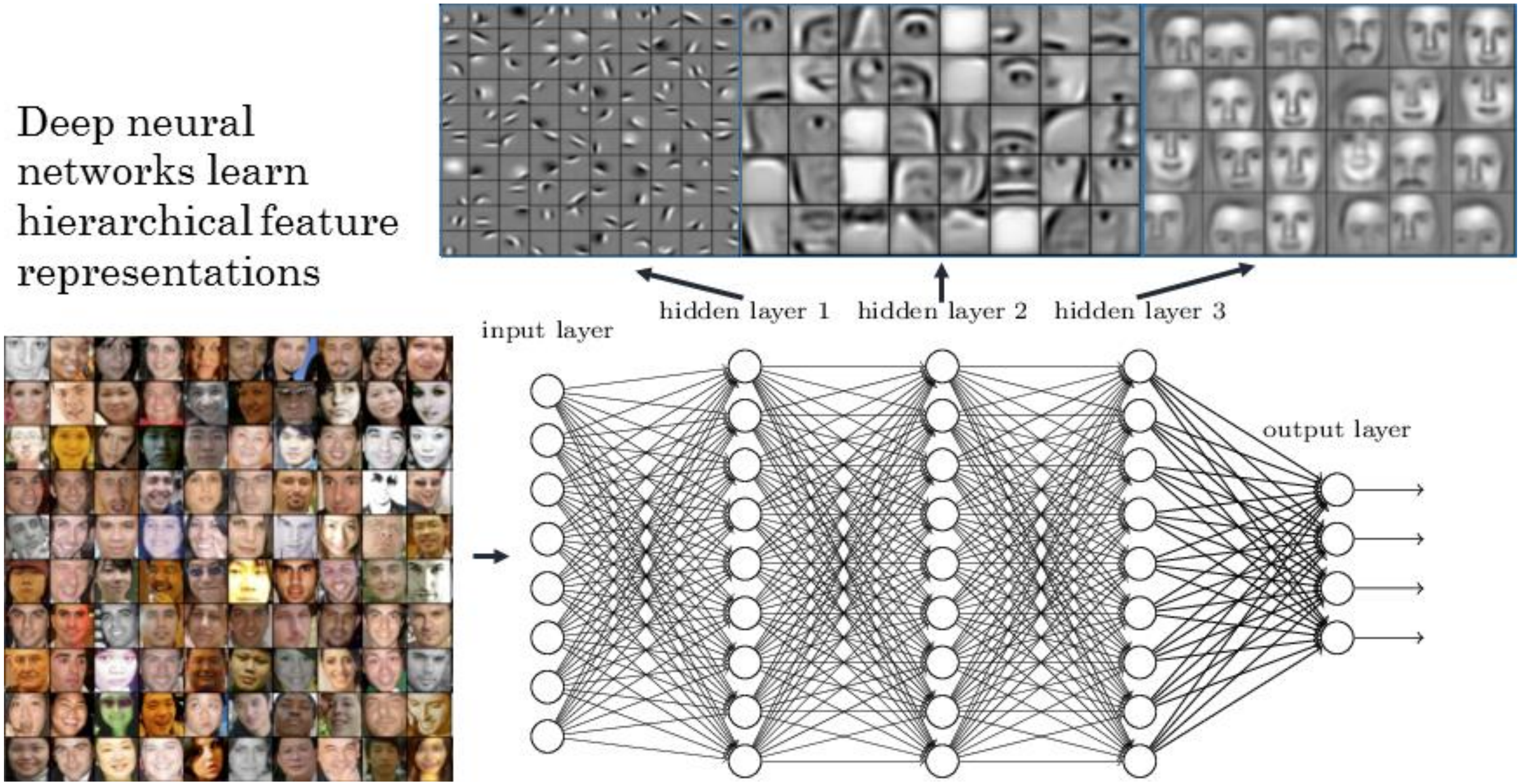
  The world's most experienced geospatial intelligence organisation.

Ordnance Survey

What OS actually does

- Paper maps now a legacy – less than 6% of our business but still hugely significant
- One of the UK's largest stores of geographic digital data
- Making **100,000+** updates every day to over **650+ million** physical features.

Convolutional Neural Networks (CNNs)



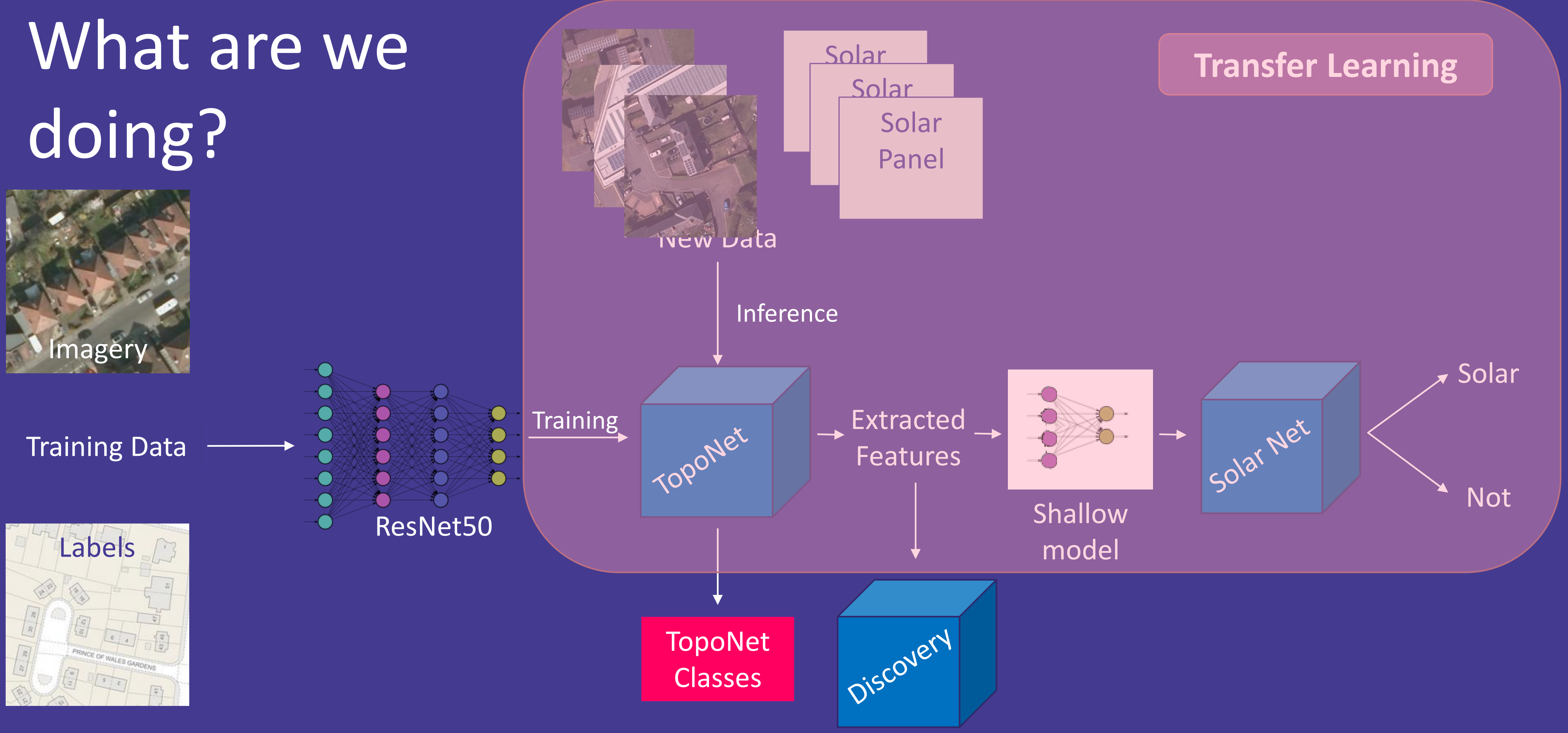
Learning deep representations

Convolutional Neural Networks (CNNs)

Ordnance Survey

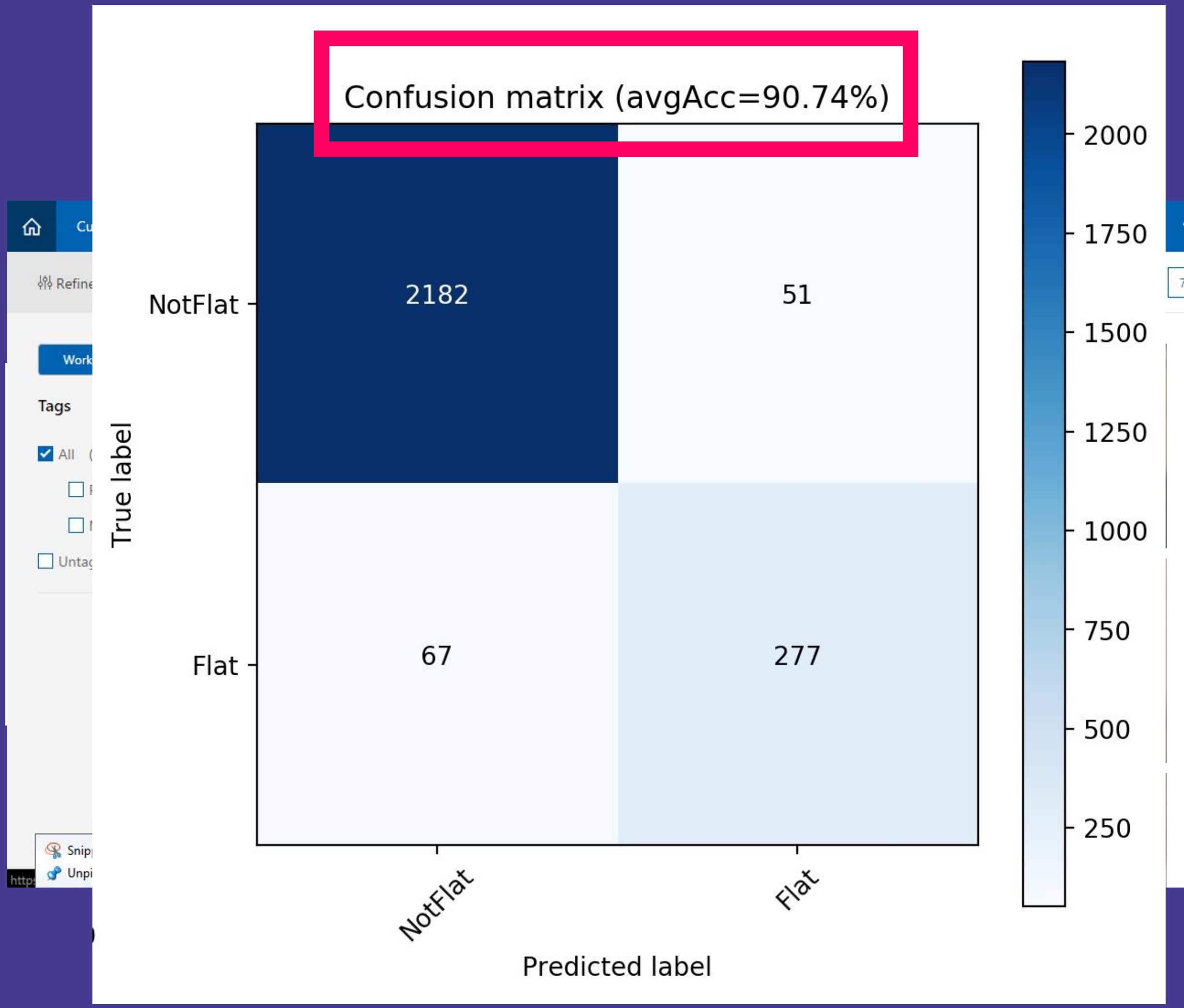
ImageNet

What are we doing?



Microsoft Roof Hack

- 20,000 labels
- Only in Hull and Southampton
- Webservice creating an SVM using imagenet features
- … but we weren't using equal classes

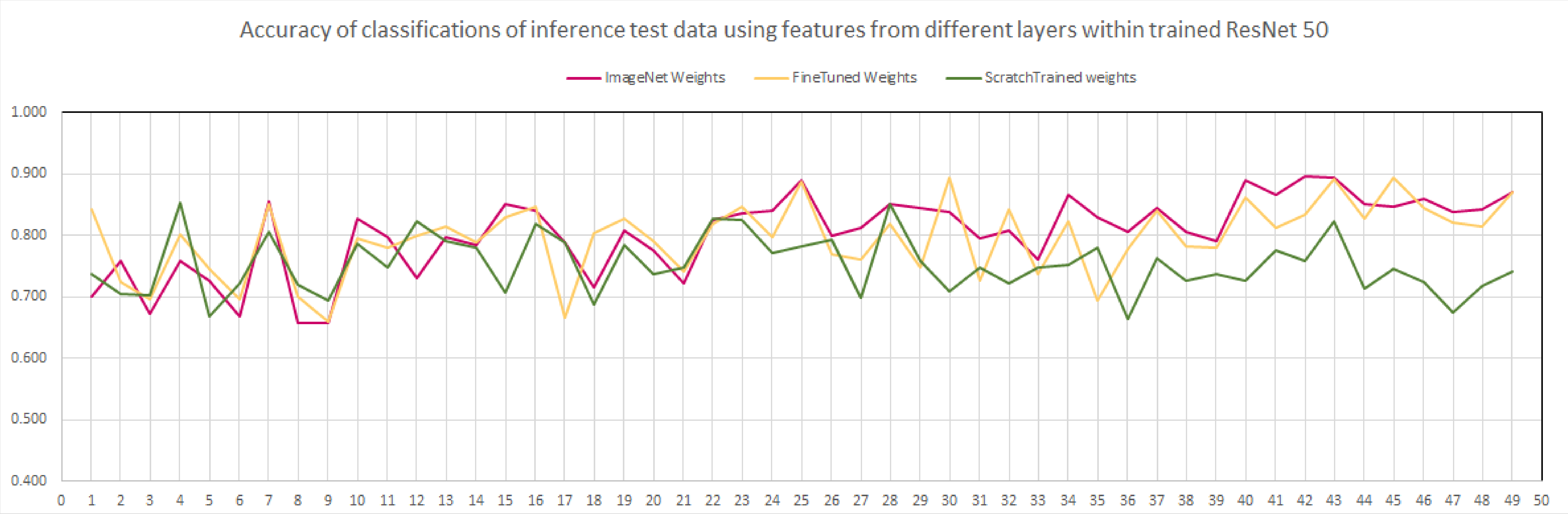


Training a CNN

- 1 GPU
- 50 layers ResNet50 architecture
- 1.2 million image-class pairs (Southampton area)

- Train from scratch with aerial imagery (“scratch train”)
- Steal the ImageNet weights (“imagenet”)
- Or, start with imagenet and then train using aerial imagery (“fine-tune”)

First Results of Transfer Learning

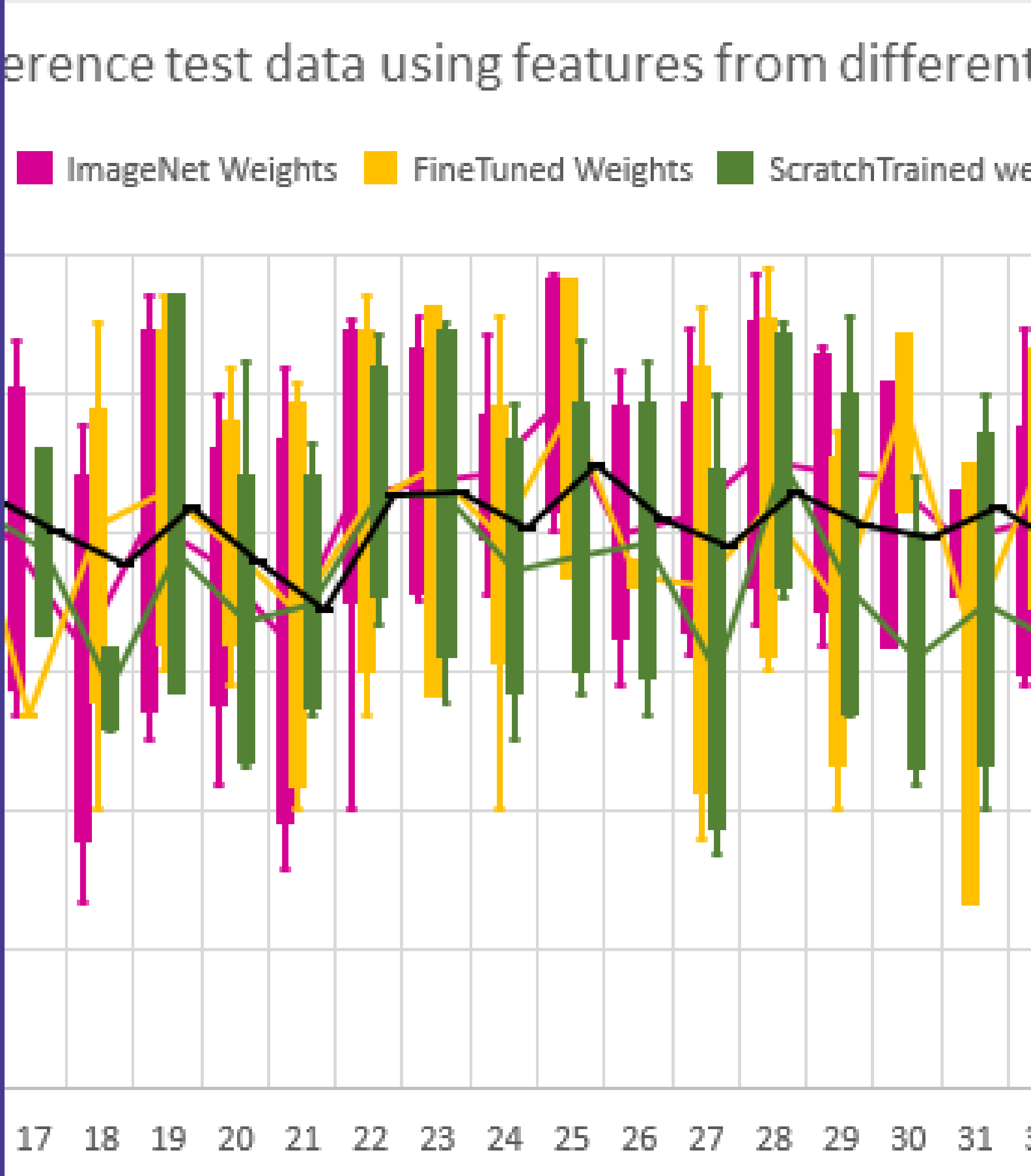


Ordnance Survey

WHAT?

3 weeks to train (x 2)

Yet, at best AS GOOD AS ImageNet weights



Ordnance Survey

But…

Aerial imagery is different to ImageNet data because:

- No foreground/background relationship in the scene
- Not ‘composed’, e.g. with objects arranged in the middle
- Different colour and texture

So training with aerial imagery MUST be valid

So…

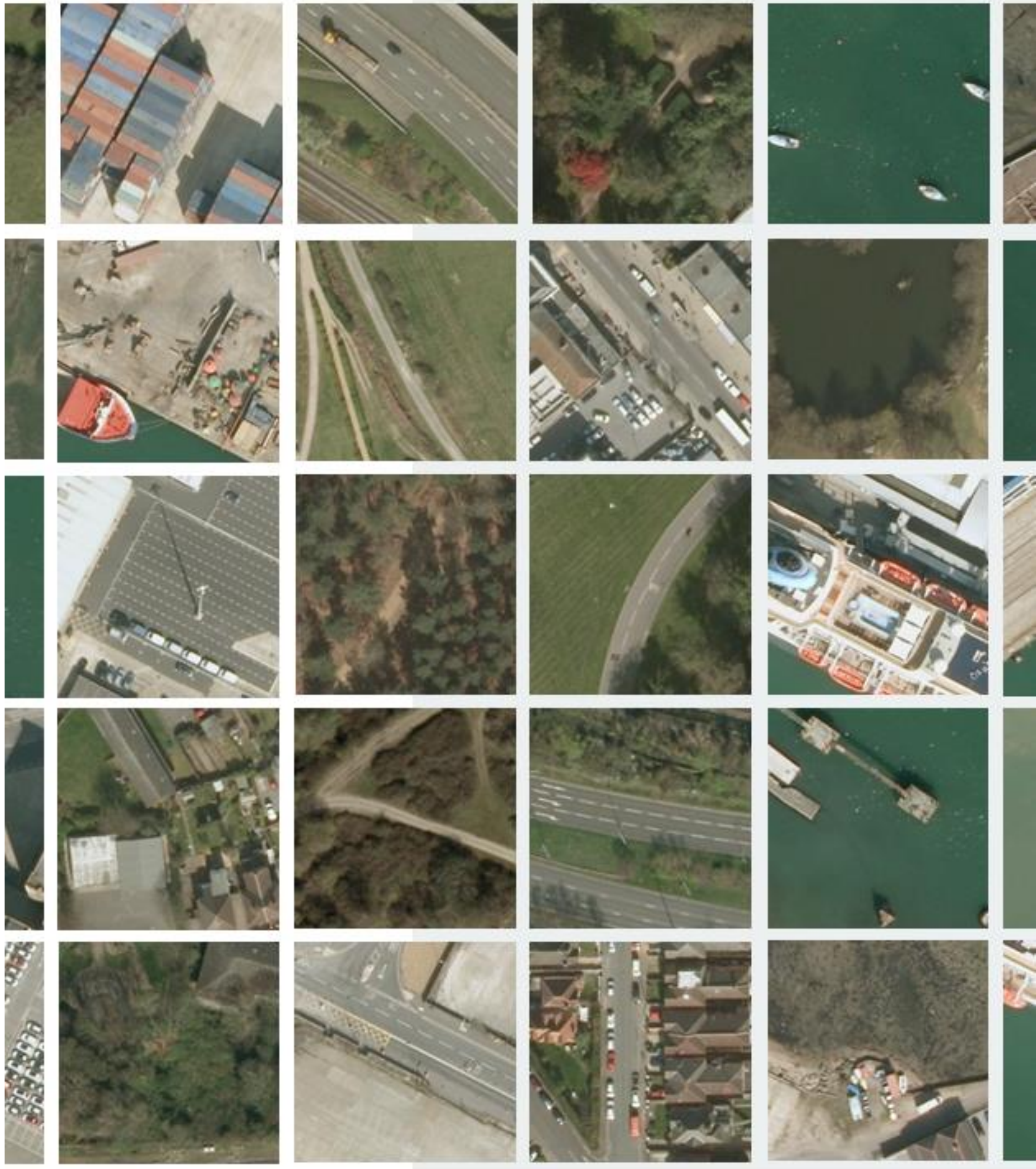
We need to experiment to improve our model.

It's a many dimensional search space:

- Architecture
- Training time
- Hyperparameters

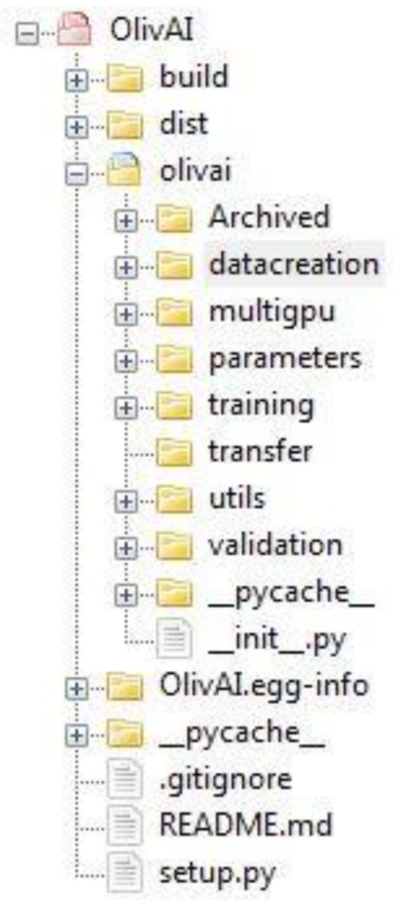
But…

3 weeks is too long



olivai

- 79 files
- ~ 3000 lines of code
- ~ 1200 lines of commenting (!)
- 'Olivia' → OlivAI



olivai

- Built on Python, TensorFlow and Keras

Why Keras?

- Well used and documented
- More seriously, multiple backends
- Easier than raw TensorFlow
- PyTorch was in its infancy when we started
- Better error handling than Neon

Running

```python
from olivai.training import data_gen, train_on_gen
args = parse_args()

# Create the generator for the data for the model to train on
gen = data_gen(args.folder, target_size=(224,224),
               batch_size=args.batch_size)

serial_model = train_on_gen(args.arch, gen, args.batch_size,
                            args.epochs, args.gpus,
                            args.parameter_server,
                            args.method, args.num_class)
```

Pipeline

```
def precompile(model_name, num_classes, gpus, parallel, epochs,
               run_test, batch_size, method,
               validation_data=None):
    (parallel_model, serial_model) = create_model(model_name,
                                                  num_classes,
                                                  gpus, method
                                                  parallel)
    callbacks = default_callbacks(parallel, serial_model,
                                  batch_size)
    fit_dict = create_fit_dict(callbacks, epochs, run_test,
                               validation_data, parallel)
    return (parallel_model, serial_model, fit_dict)
```

Pipeline

```
def train_on_gen(model_name, gen, batch_size, epochs, gpu_count,
                 num_classes=12):
    out = precompile(model_name, num_classes, gpu_count,
                     epochs, batch_size, validation_data,
                     parallel=True)
    (parallel_model, serial_model, fit_dict) = out
    parallel_model.compile(**compile_dict)
    parallel_model.fit_generator(gen, **fit_dict)
    return serial_model
```

Parallelisation

- Parallelisation is hard!
- Data vs model parallelism?

What actually happens

- Create serial model (on CPU)
- Pass serial model to `make_parallel` (rossumai/keras-multi-gpu)
- `make_parallel` distributes model onto GPUs
- When fitting the data batch (128 in my case) is divided between the GPUs (32 on each)
- On batch end each model comes back with different weights and updates them centrally

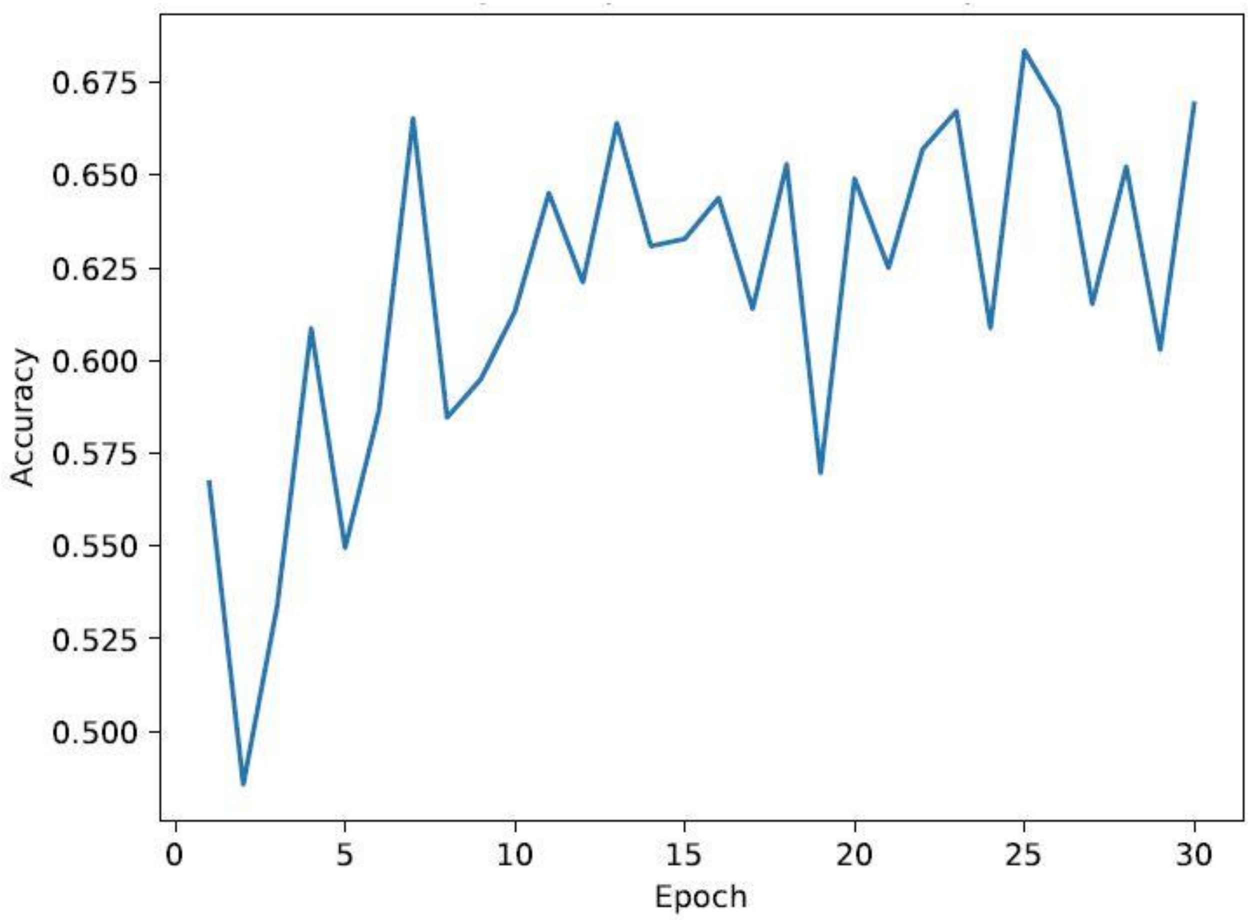
3 Weeks - > 2 days ☺

What decreased training time

- Parallelisation not using buggy parallelisation
- Also conda installs of keras and tensorflow (on keras > 2.2.2)

Results

Test accuracy of ResNet at each epoch



TopoNet V2

Speed ups mean we able to iterate quickly

TopoNet is fully trained

We can iterate quickly

We can quickly change parameters

We have many to choose from

So which TopoNet is best for what task?

Transfer Learning

```python
def create_feature_model(model, trained_weights, num_acts=None,
                         upper_layer=None):
    model.load_weights(trained_weights)
    acts = [l for l in model.layers if isinstance(l, keras.layers.Activation)]

    if upper_layer is not None:
        layer_names = [l.name for l in acts[:upper_layer + 1]]
    else:
        layer_names = [l.name for l in acts]

    outputs = [model.get_layer(layer).output for layer in layer_names]
    inp = model.get_input_at(0)

    feature_model = keras.Model(inputs=inp, outputs=outputs)
    return feature_model
```

Transfer Learning

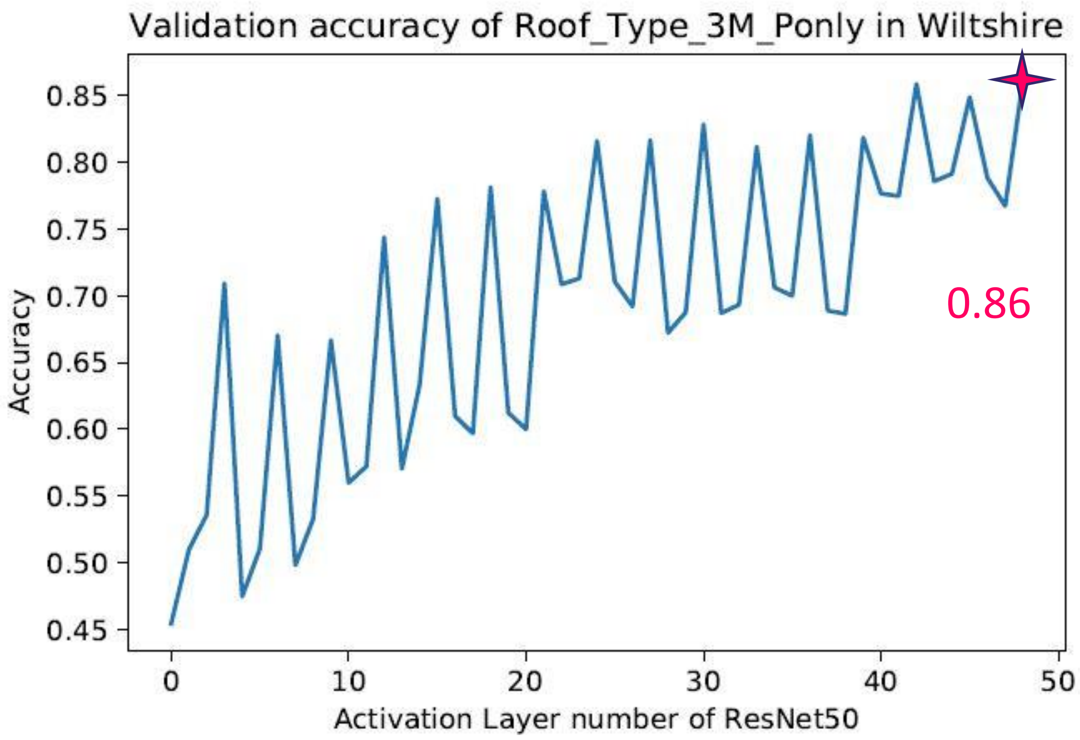
```
def create_features(feature_model, folder, batch_size,
                    target_size):
    gen = data_gen(folder, target_size=target_size,
                   batch_size=batch_size)

    features = feature_model.predict_generator(gen)

    return features
```

- This returns a list of arrays for each layer of the network (chosen using `create_feature_model`)
- Each array is of shape `(num_images, num_filters, size_of_filters)`

So where are we now?

Complex
Flat
Gabled
Hipped
Other

Concrete
Glass
Metal
Other
Thatched
Tiled



Testing

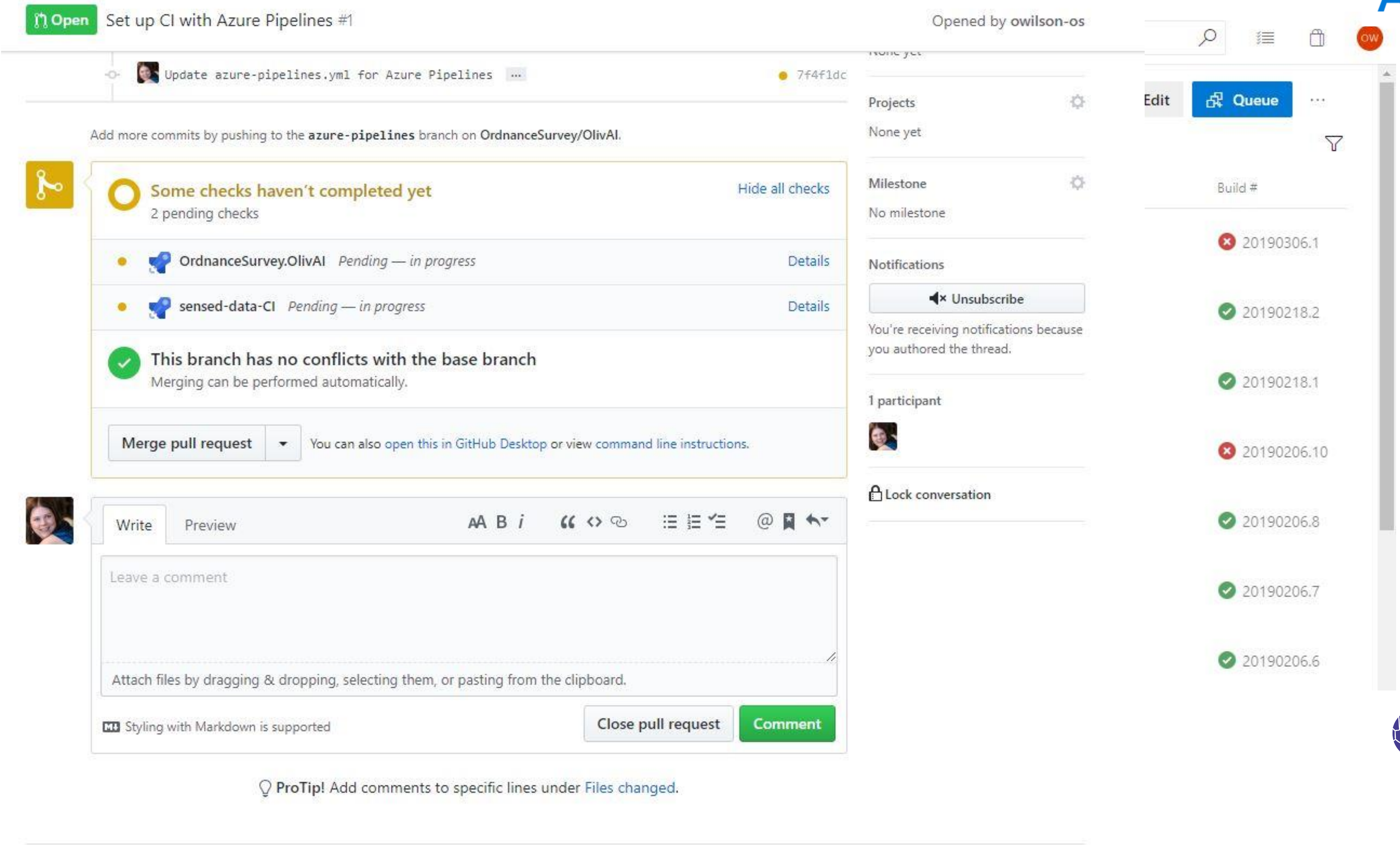
cmd (Admin)

<1> cmd

```
(olivai) OWilson@afahpoc1w006 C:\OS_Code\OlivAI\tests
# pytest -m "not gputest"
===================================== test session starts ===================
platform win32 -- Python 3.6.8, pytest-4.3.0, py-1.8.0, pluggy-0.9.0
rootdir: C:\OS_Code\OlivAI\tests, inifile: pytest.ini
collected 170 items / 30 deselected / 140 selected

test_parameters\test_parameters.py ..
test_training\test_callbacks.py ...
test_training\test_data.py ....FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFF [ 47%]
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF                                [ 77%]
test_training\test_model.py ...................                           [ 92%]
test_training\test_training.py ss                                         [ 93%]
test_transfer\test_transfer.py sssss                                      [ 97%]
test_utils\test_model_utils.py sss.                                       [100%]

================================= FAILURES =================================
________________________ test_shuffle_filenames[0] _________________________

seed = 0

    @pytest.mark.parametrize("seed",
                             range(100))
    def test_shuffle_filenames(seed):
>       list_fol, list_names, lookup_features = load_shuffle_jbls()

test_training\test_data.py:66:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
test_training\test_data.py:119: in load_shuffle_jbls
    list_fol, list_names = joblib.load(jbl_list)
```

Continuous Integration

Azure Pipelines



Ordnance Survey

Conclusions

- Developed a robust python library (**olivai**) with documentation, testing and continuous integration with necessary functionality.
- Data remains ‘90%’ of the work, and probably always will (long live the domain expert)
- We can now iterate quickly; essential to our work
- We’ve trained some deep networks and are now in a position to train in anger rather than mild frustration

Any Questions?

Olivia Wilson
Research Scientist

Olivia.Wilson@os.uk
@oew1v07

Ordnance Survey