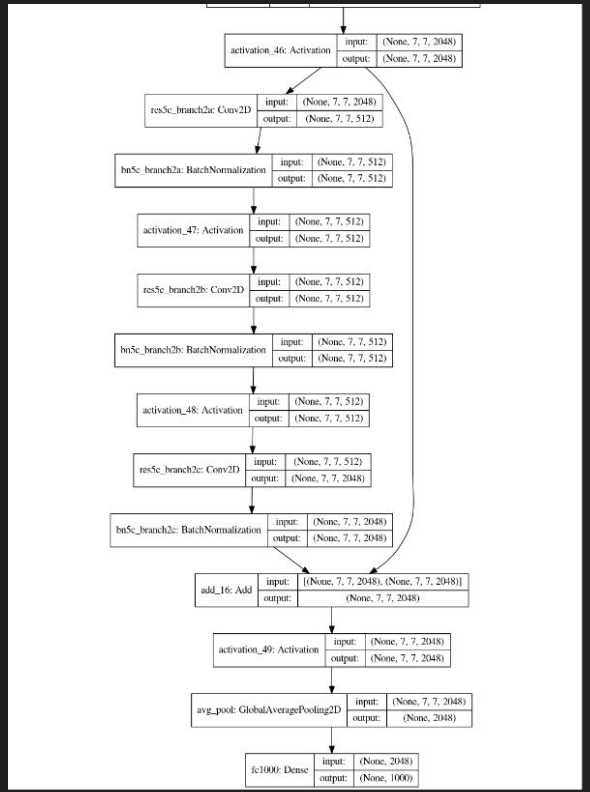
M3-Week04-Group02

ResNet50 model selected.



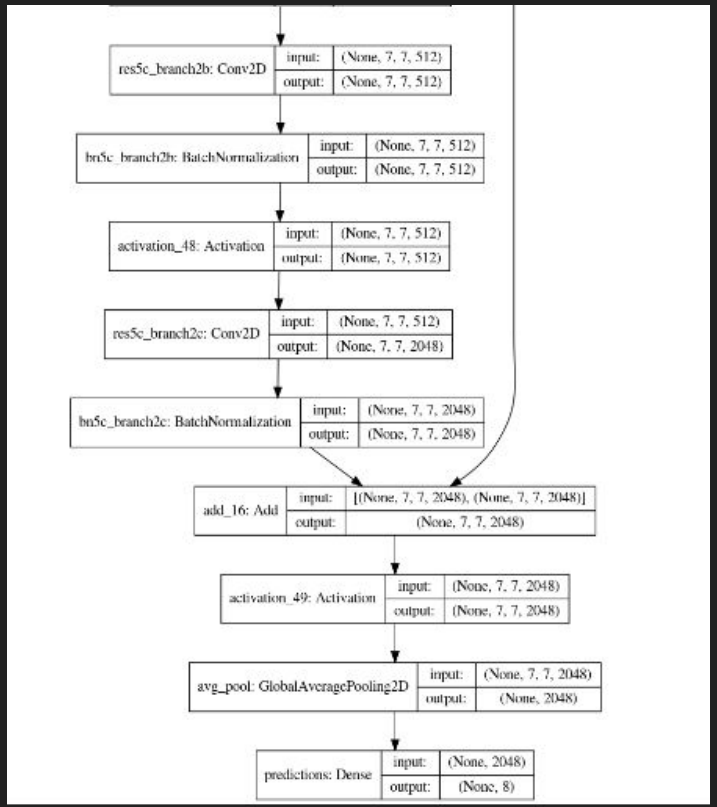
ResNet model architecture consists of blocks of main activation layers. Here showing 49th activation layer, block before is activation_46.

All files and code can be found on github

Click me!

First changed the last dense layer to 8 neurons representing our 8 classes

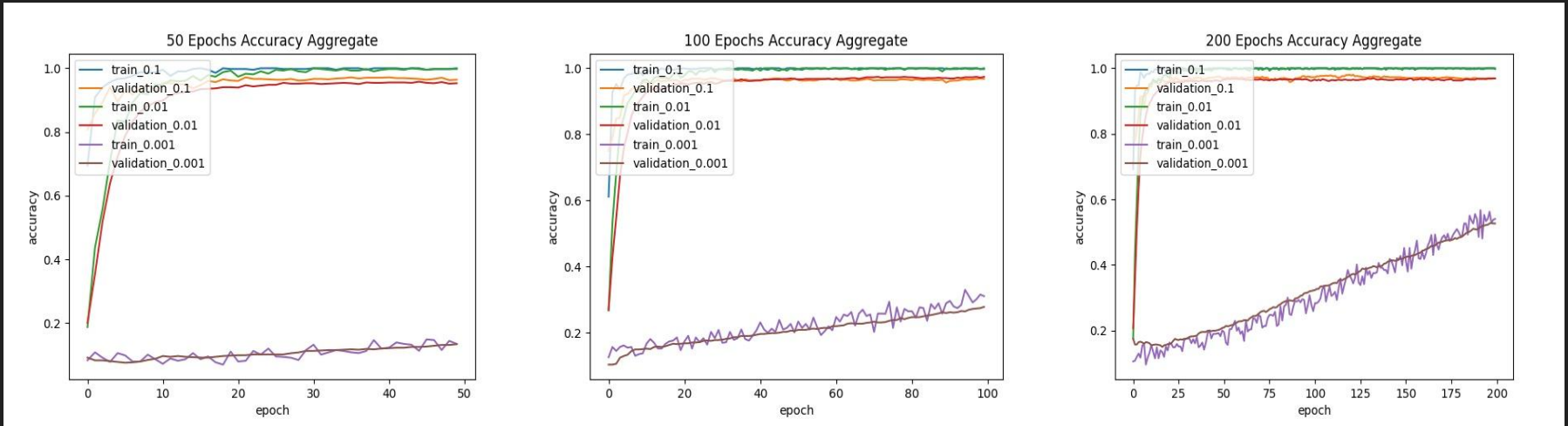
Then we have tried to unfreeze some of the layers -3,-4,-5 to -15 but we noticed it is not the right approach due to architecture of our network.



Example of unfreezing layers

We can see that unfreezing different layers does not change the output of a network and does not improve accuracy. Which was expected as we are adjusting only some layers and backpropagation is not adjusting whole network to work better for end result.

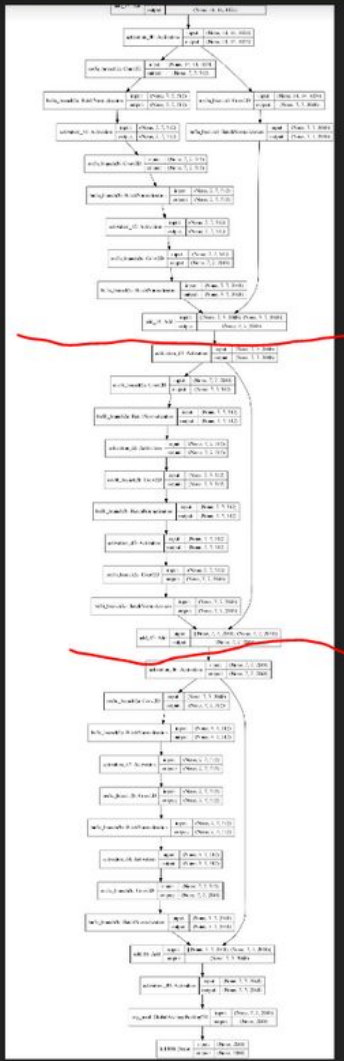
Learning rate variation on Resnet50



At all of the tested epoch numbers, the learning rate of 0.01 seems to be the best case. It might be the case that for epochs much, much greater than this number the accuracy of LR <= 0.001 outperforms the one we selected, but we believe the execution difference makes up for this. Our best validation_accuracy is for 100 Epochs, LR = 0.01 at 0.9690

Other change was to cut the network by blocks

Since ResNet is created with blocks of layers connected to each other in a pattern, changing the size of the model required cutting in modules



Results of a different size of models from ResNet50

We have decided to cut the network and different layers of main activation layers: Activation_46,43,40,37,22,7 to see how the network responds.

We could see that for 20 epochs best results are yield by 43, however we noticed that results are growing therefore we run it for 50 epochs

Name	Smoothed	Value	Step	Time	Relative
base_with_layer-2_cut_activation_22\validation	0.7479	0.8253	19	Tue Jan 26, 17:06:51	1m 3s
base_with_layer-2_cut_activation_37\validation	0.8537	0.8563	19	Tue Jan 26, 16:47:07	1m 17s
base_with_layer-2_cut_activation_40\validation	0.8705	0.8476	19	Tue Jan 26, 16:33:00	1m 22s
base_with_layer-2_cut_activation_43\validation	0.9117	0.9157	19	Tue Jan 26, 16:24:10	1m 24s
base_with_layer-2_cut_activation_46\validation	0.8524	0.8191	19	Tue Jan 26, 15:55:42	1m 26s
base_with_layer-2_cut_activation_7\validation	0.6054	0.6369	19	Tue Jan 26, 17:11:50	55s

Running different size of model with 50 epochs

We selected two sizes of the model that were increasing and run them for 50 epochs

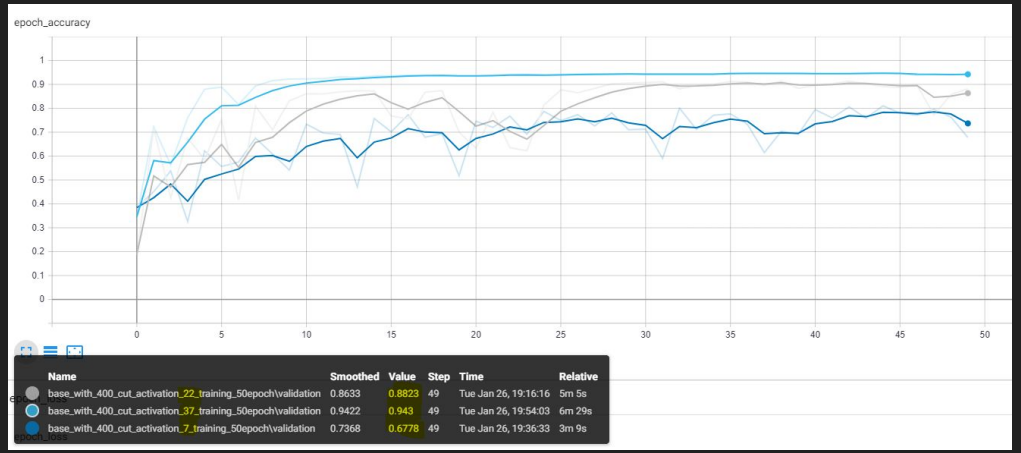
We noticed interesting behaviour for cut off at 37 activation layer model where around epoch 40 validation accuracy reached 53% and then went up again to 95%

Running different size of model with 50 epochs and 379 images

After initial analysis of smaller sizes of a model we took only 379 images for training and run the models for 50 epochs to have better window of improvement

We saw similar behaviour as before for model with cut off at 22nd activation layer at epoch 23 going down to 62% then go up again to almost 90%

Best result was seen for model with cut off at 37th activation layer that reached 94% at 20 epoch already

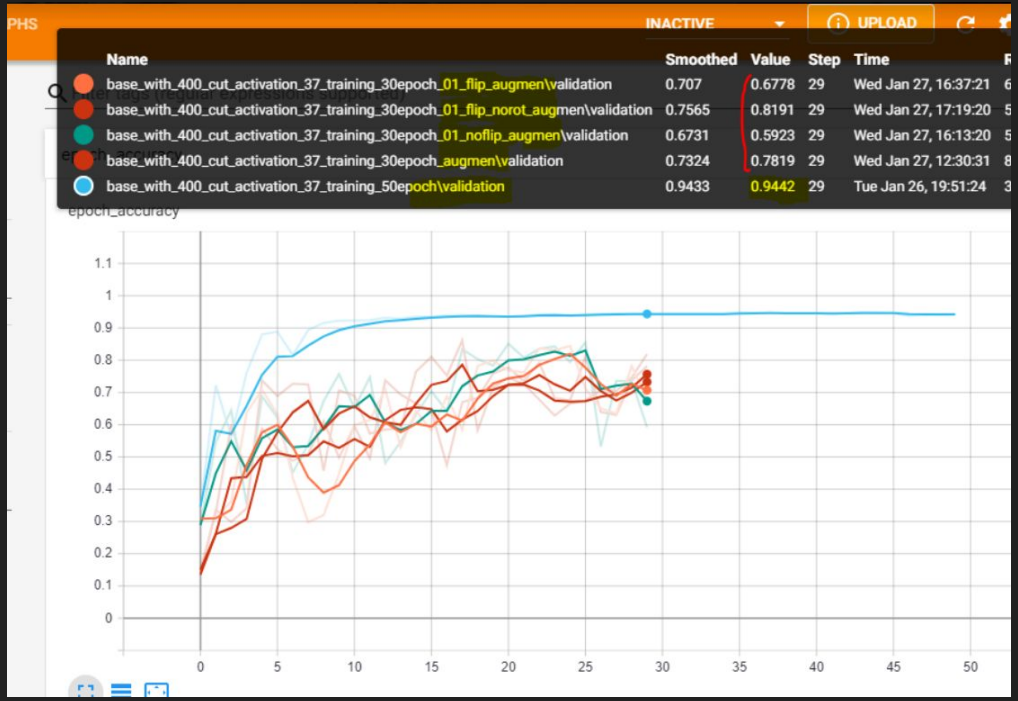


Using augmentation for model cut_activation_37

Good practice for enhancing performance of the network with small amount of images is using data augmentation to generate variance in the dataset.

It was expected that the use of data augmentation would improve the performance of the neural network in a scenario where data is scarce.

However, in our case results showed otherwise



Using augmentation for model cut_activation_37

In order to further understand the results of our network with data augmentation, we ran a series of controlled experiments separating the following data augmentation techniques:

- Rotation range: 20 degrees
- Width shift: 20% (of the total width of the image in pixels)
- Height shift: 20% (of the total height of the image in pixels)
- Shear range: 20%
- Zoom range: 20%
- Horizontal flip
- Vertical flip

	Train accuracy	Validation accuracy
Rotation	1	0.84
Width shift	0.12	0.14
Height shift	0.99	0.85
Shear	1	0.93
Zoom	0.13	0.14
Horizontal flip	1	0.93
Vertical flip	0.99	0.91
All together	0.96	0.66

All models were trained for 50 epochs

Using augmentation for model cut_activation_37

This experiment showed that most techniques produce extreme overfitting, we believe this is due to a very light distortion applied to the images that allows the network to entirely *learn* the train set while failing to generalize on the validation set.

Some types of data augmentation, such as random *Zoom* and *Width shift,* produced poor results even in the training set. This may be produced by an excessive distortion of the original images when applied isolated from other techniques.

A combination of all techniques derived in overfitting and a poor ability to generalize.

	Train accuracy	Validation accuracy
Rotation	1	0.84
Width shift	0.12	0.14
Height shift	0.99	0.85
Shear	1	0.93
Zoom	0.13	0.14
Horizontal flip	1	0.93
Vertical flip	0.99	0.91
All together	0.96	0.66

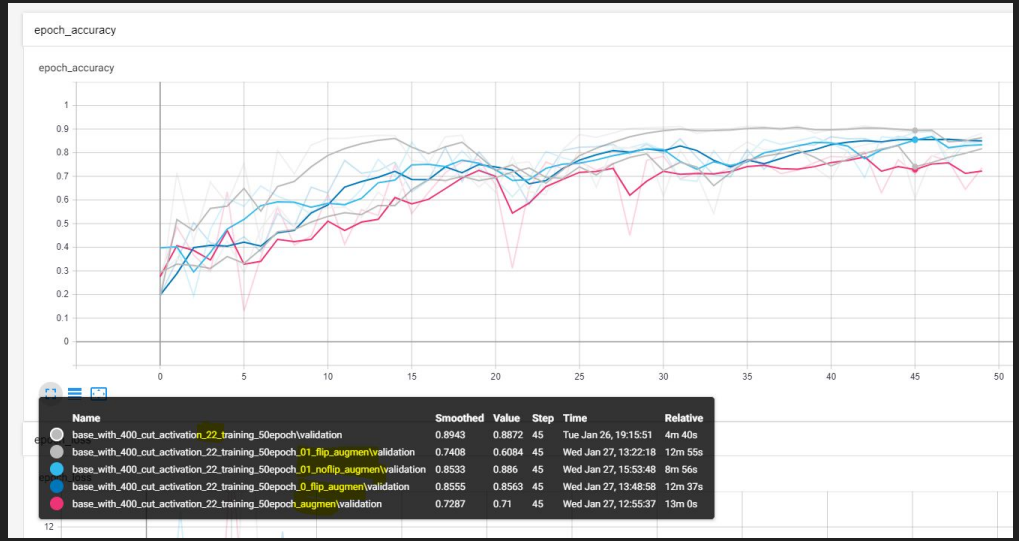
In the case of *Height shift* and *Vertical flip*, overfitting (although present) was not as excessive.

Using augmentation cut_activation_22

As it striked us odd that augmentation does not bring better results we verified the same for different size model.

Same conclusion was drawn, that model without augmentation performs better than models with augmentation of data.

Combining zoom, shift augmentation along with vertical flip brought worst results. Where using either zoom, shift OR flip allowed to get similar result to no augmentation model for this case.

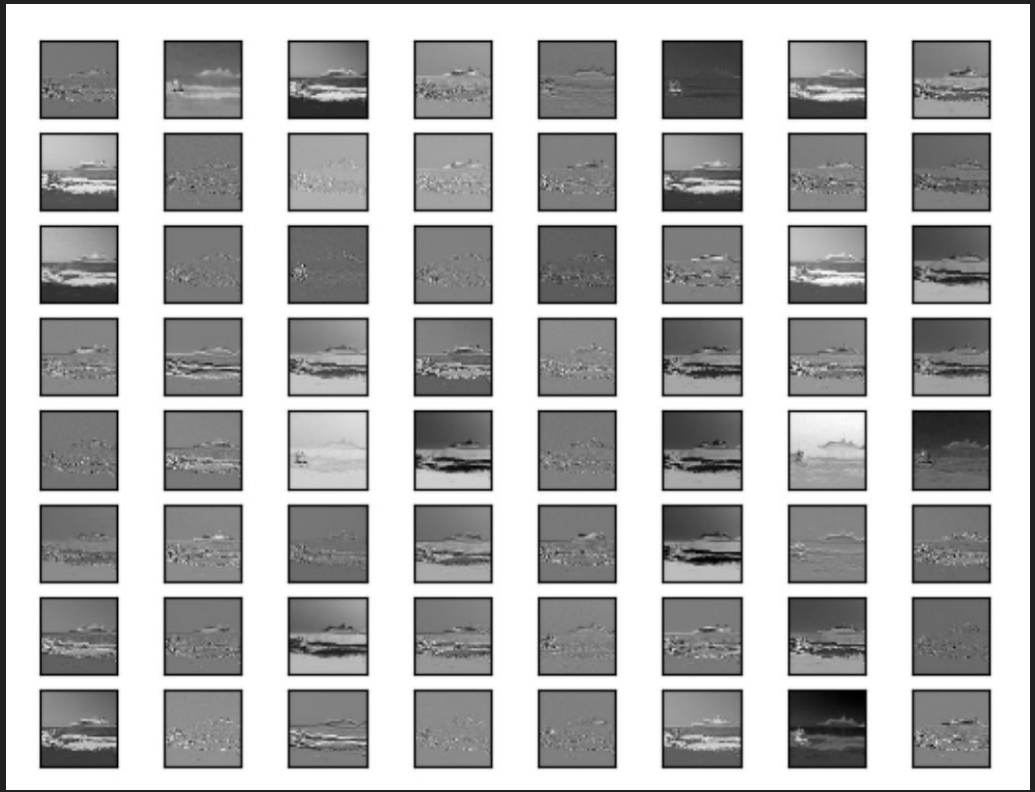


Even though results state otherwise we could assume that using data augmentation probably allow model to generalize better in the future regardless of our validation set yielding better results without augmentation.

Visualizing output from first convolution layer in Resnet50

First conv layer of Resnet having 64 channels allows us to see 64 images that shows the way network is processing this one image of a coast.

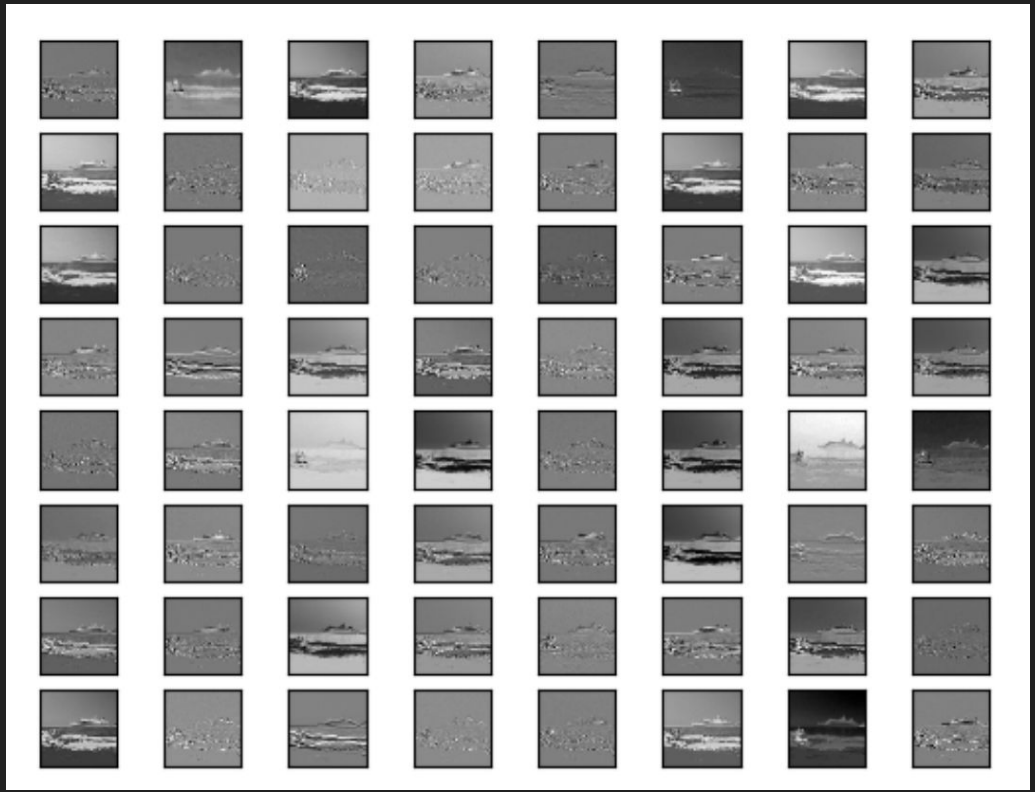
We can see, for instance, that some neurons specialized on separating sea from the sky in an image of a beach.



Visualizing output from first convolution layer in Resnet50

First conv layer of Resnet having 64 channels allows us to see 64 images that shows the way network is processing this one image of a coast.

We can see that some neurons specializes on separating sea from the sky



Hyperparameter optimization

For hyperparameter optimization, we used two different search methods

```
tuner = kt.RandomSearch(base_model,
    objective="val_accuracy",
    max_trials=15)
```

```
tuner = kt.Hyperband(base_model,
    objective='val_accuracy',
    max_epochs=HYPERBAND_EPOCHS,
    factor=HYPERBAND_FACTOR,
    directory=f'hyperband_E{HYPERBAND_EPOCHS}_F{HYPERBAND_FACTOR}',
    project_name=F'hyperparam_opt_E{HYPERBAND_EPOCHS}_F{HYPERBAND_FACTOR}')
```

Hyperband

Hyperband works similiar to randomSearch, except it “pits” results against each other in a tournament format, choosing the “winner” of each run and optimizing parameter selection after every trial.

```
tuner = kt.Hyperband(base_model,
    objective='val_accuracy',
    max_epochs=10,
    factor=2,
    directory='hyperband_10',
    project_name='hyperparam_opt')
```

```
tuner.search(train_generator, validation_data = (test_generator),  epochs=10, callbacks=[ClearTrainingOutput()] )
```

Hyperband

These are the parameters optimized through HyperBand.

```
########################################

Search space summary
Default search space size: 5
conv3_depth (Choice)
{'default': 4, 'conditions': [], 'values': [4, 8], 'ordered': True}


conv4_depth (Choice)
{'default': 23, 'conditions': [], 'values': [23, 36], 'ordered': True}


pooling (Choice)
{'default': 'avg', 'conditions': [], 'values': ['avg', 'max'], 'ordered': False}


optimizer (Choice)
{'default': 'adam', 'conditions': [], 'values': ['adam', 'sgd'], 'ordered': False}


learning_rate (Choice)
{'default': 0.01, 'conditions': [], 'values': [0.1, 0.01], 'ordered': True}



########################################
```

Hyperband

Hyperband outputs the status of the search at the start of every trial

```
Search: Running Trial #1

Hyperparameter    |Value           |Best Value So Far
conv3_depth       |4               |?
conv4_depth       |23              |?
pooling           |max             |?
optimizer         |adam            |?
learning_rate     |0.1             |?
tuner/epochs      |2               |?
tuner/initial_e...|0               |?
tuner/bracket     |2               |?
tuner/round       |0               |?
```

```
Trial 25 Complete [00h 01m 01s]
val_accuracy: 0.15831135213375092

Best val_accuracy So Far: 0.38786280155181885
Total elapsed time: 00h 14m 55s

Search: Running Trial #26

Hyperparameter    |Value           |Best Value So Far
conv3_depth       |8               |4
conv4_depth       |36              |23
pooling           |avg             |avg
optimizer         |sgd             |sgd
learning_rate     |0.01            |0.1
tuner/epochs      |10              |10
tuner/initial_e...|0               |0
tuner/bracket     |0               |0
tuner/round       |0               |0
```

Hyperband

Hyperband can also summarize a trial from a saved execution

```
Trial summary
Hyperparameters:
conv3_depth: 8
conv4_depth: 36
pooling: max
optimizer: adam
learning_rate: 0.01
tuner/epochs: 2
tuner/initial_epoch: 0
tuner/bracket: 2
tuner/round: 0
Score: 0.1556728184223175
```

Hyperband - Results

This is the result of taking the best parameters returned by Hyperband for max_epochs = 20 and factor = 5. Final validation accuracy was 0.697 after 100 Epochs.