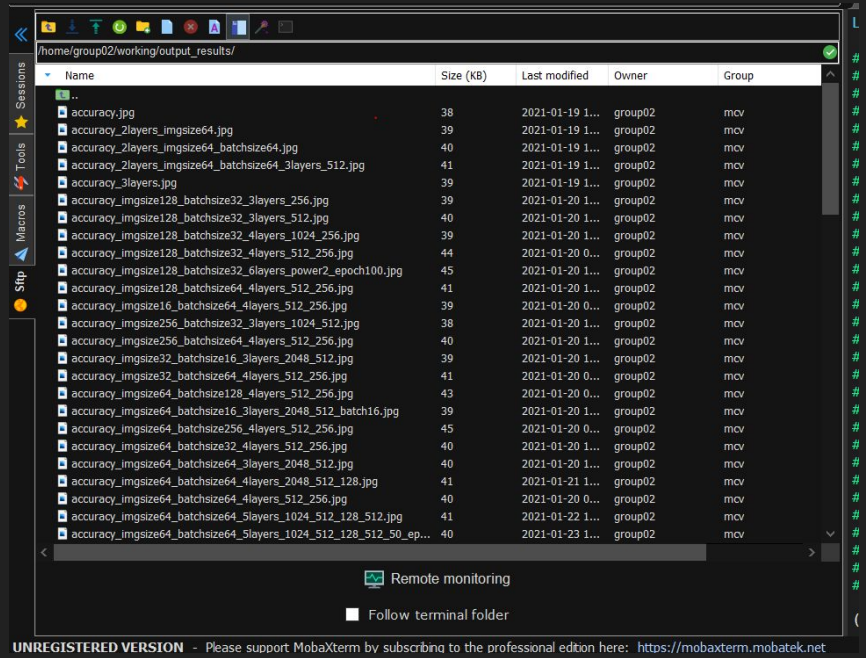
M3-Week03-Group02

Tested different combinations of MLP parameters and layers to choose representative one for precise tests.



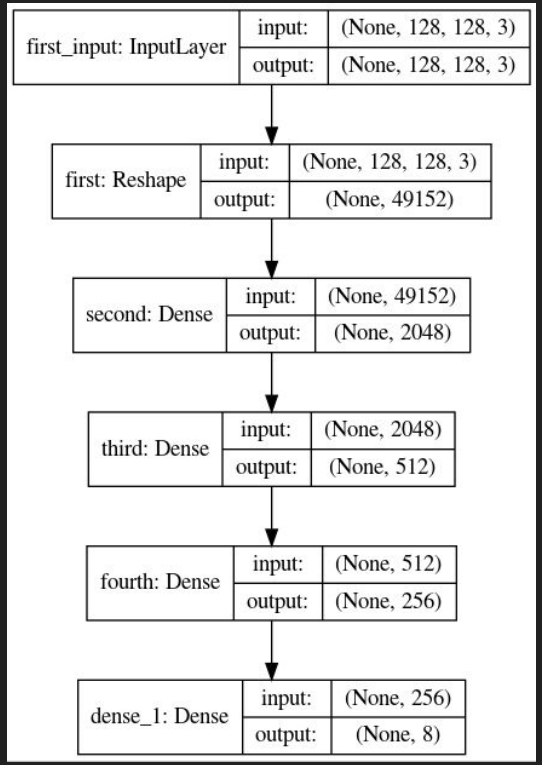
We have chosen 4 layers model presented on next slide

All files and code can be found on github

Click me!

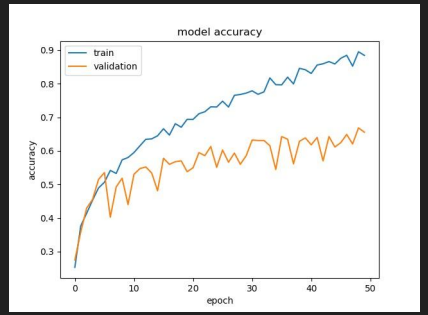
Layer Model used for image and batch size analysis

We used architecture on the right with 50 Epochs

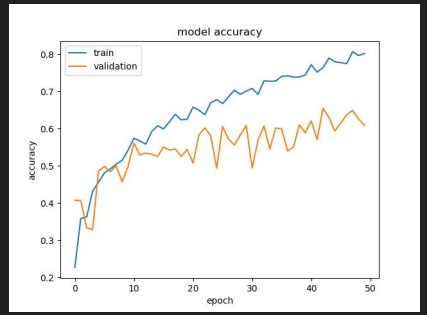


Different MLP combinations based on batch size

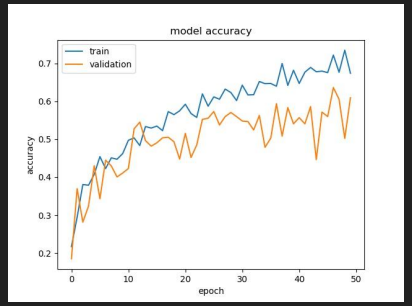
We have run same architecture with fixed image size to 64 and different size of batch: 32, 64, 128, 256. All trained with 50 epochs.



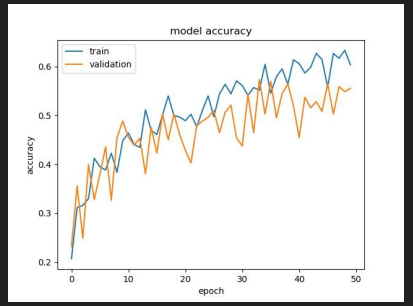
Img_size = 64
Batch_size = 32



Img_size = 64
Batch_size = 64



Img_size = 64
Batch_size = 128

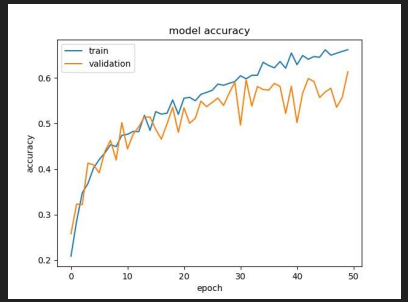


Img_size = 64
Batch_size = 256

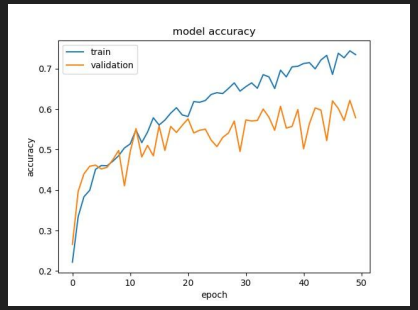
We can see that as batch size is increased the validation and test comes together therefore overfitting is decreased.

Different MLP combinations based on img size

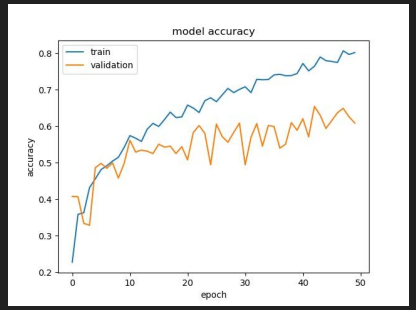
We have run same architecture with fixed batch size to 64 and different size of image: 16, 32, 64, 128. All trained with 50 epochs.



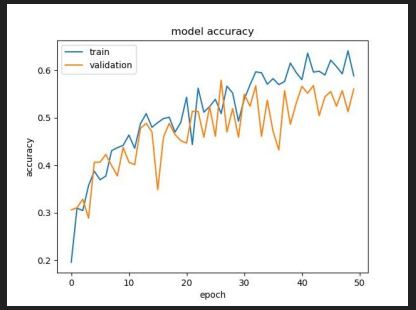
Img_size = 16
Batch_size = 64



Img_size = 32
Batch_size = 64



Img_size = 64
Batch_size = 64



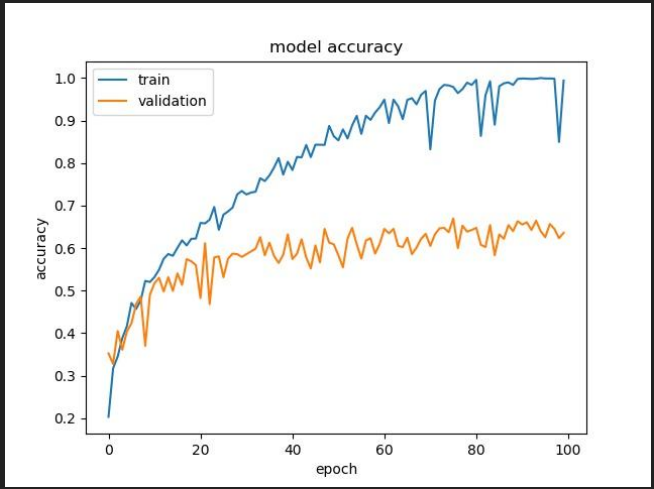
Img_size = 128
Batch_size = 64

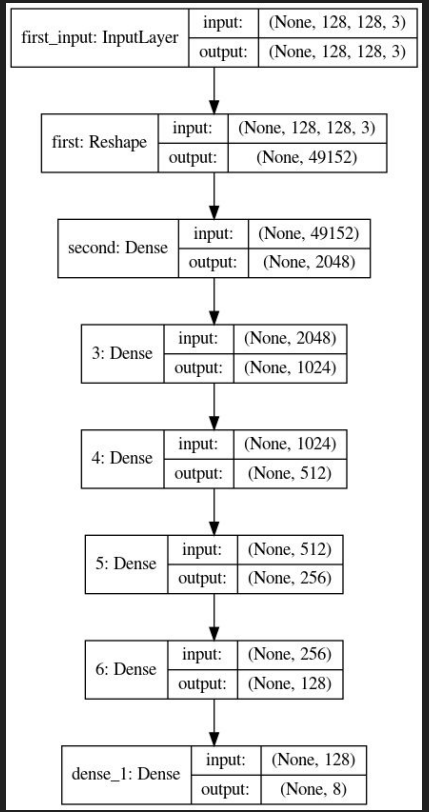
We can see that as image size is increased the validation stays more or less on the same level. For the small image size and largest (128) we can see overfitting is happening less than for 32 and 64

Different MLP combinations - 6 layers

Img_size = 128
Batch_size = 32
Epoch = 100

We have tried also deeper networks to see the behaviour. As we can see for more layers and epochs model is overfitting even more. Variance is increased a lot allowing training set to reach almost 1 in accuracy which is clear overfit.

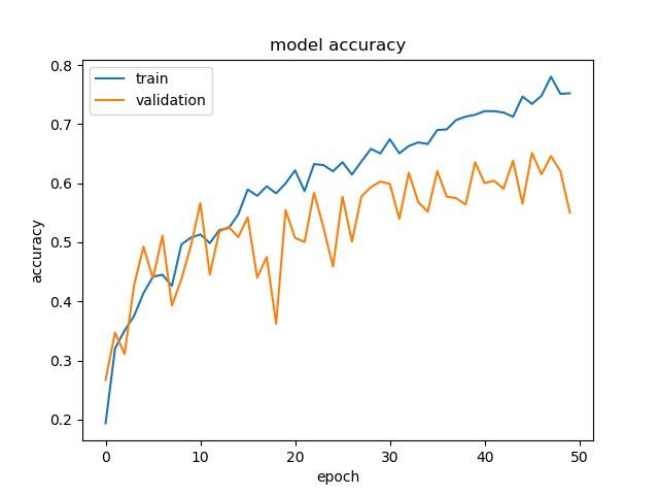


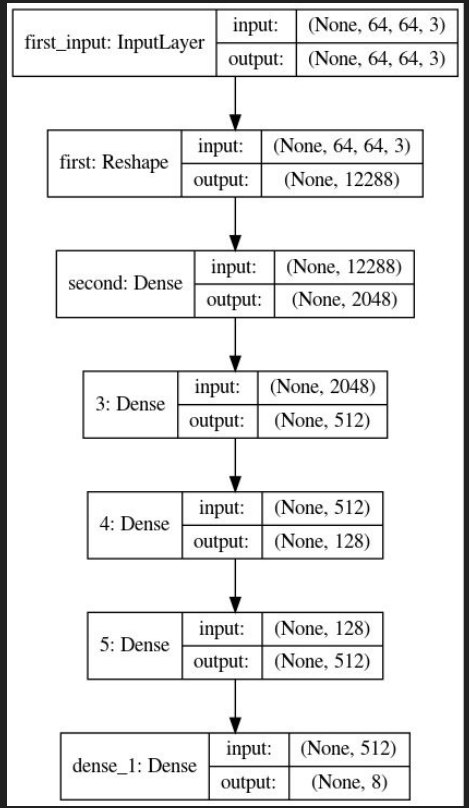


Different MLP combinations - 5 layers

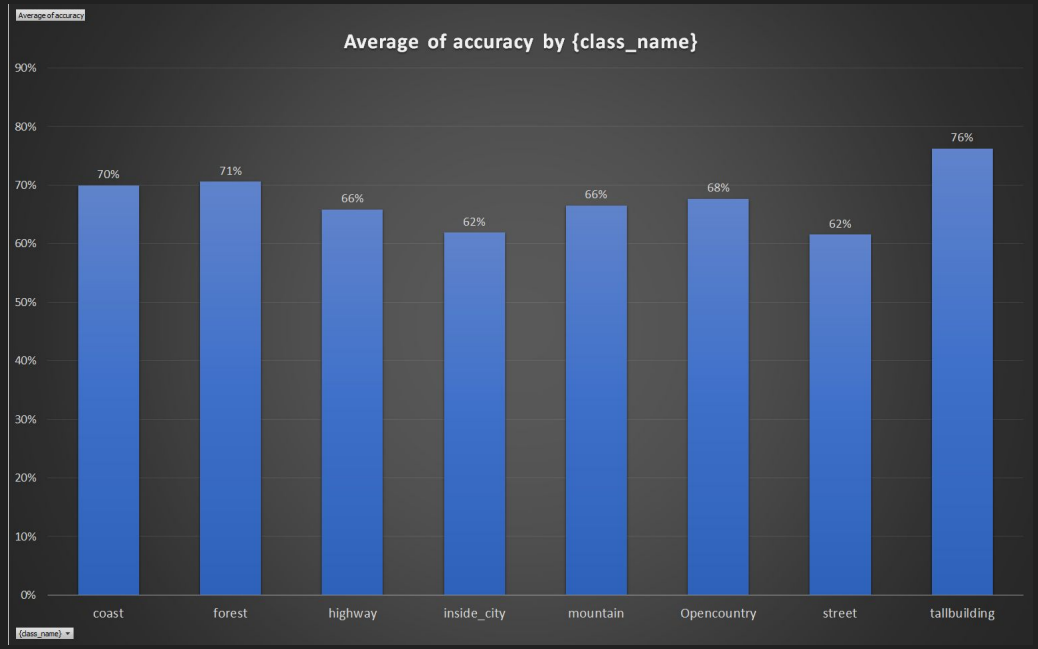
Img_size = 64
Batch_size = 64
Epoch = 50

Testing 5 layers we can see less overfitting than for 6 layers. Also we can notice that less number of epochs helps model to avoid overfitting. Usage of moderate batch size and image size is important to avoid it as well.



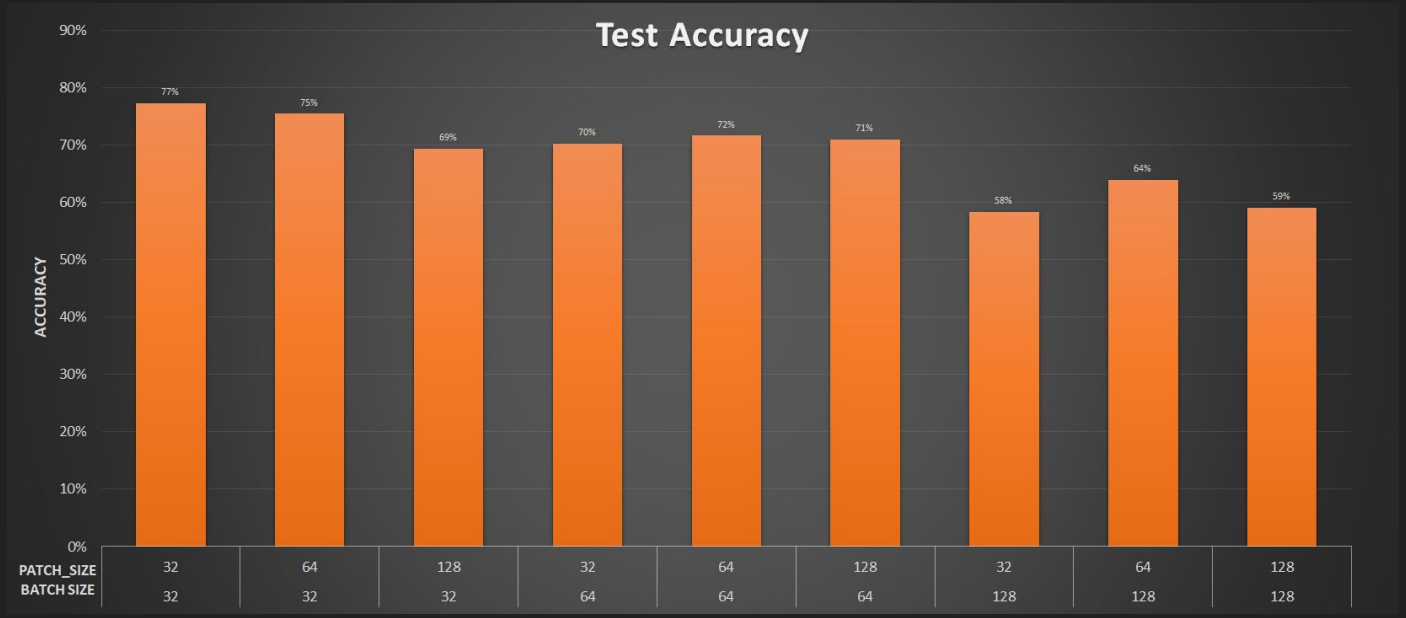


Average accuracy per class name (50 + 100 epochs)



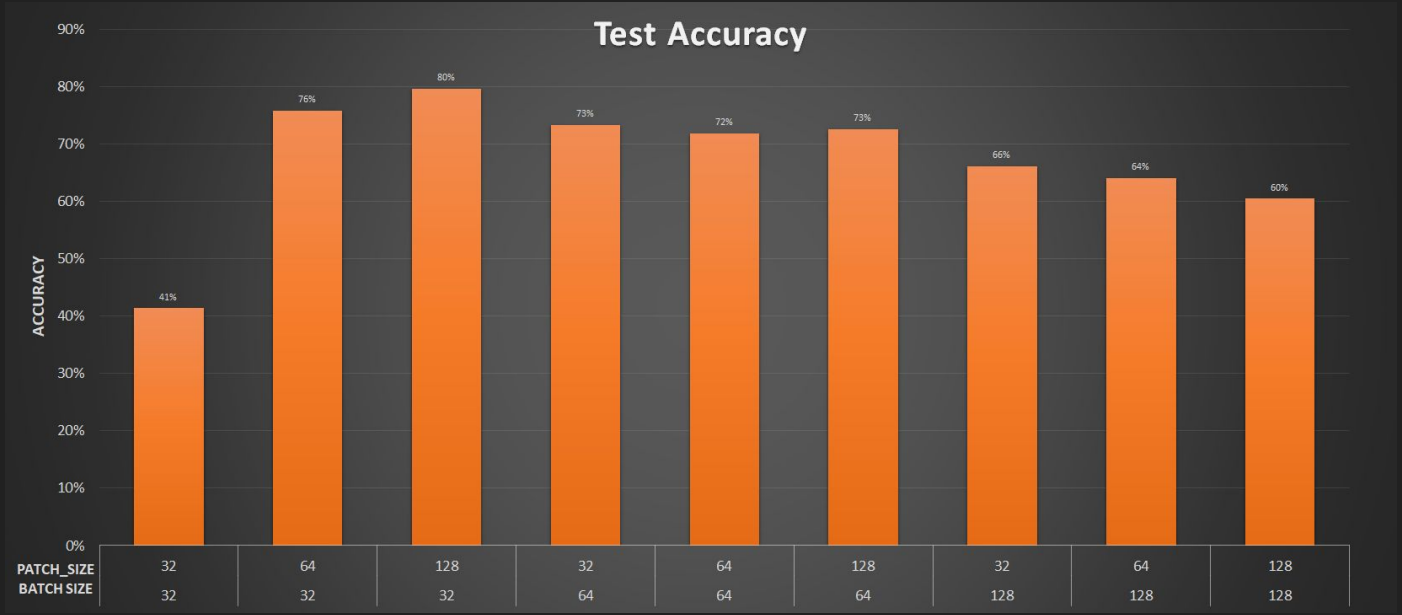
We found it interesting to look at how patch size and batch size affected the accuracy within the classes. Here are the results

Patch size and batch size - 50 epoch



At 50 epochs, the best overall results come from size 32 patches with 32 batches. These are accuracy values for the test image set.

Patch size and batch size - 100 epoch

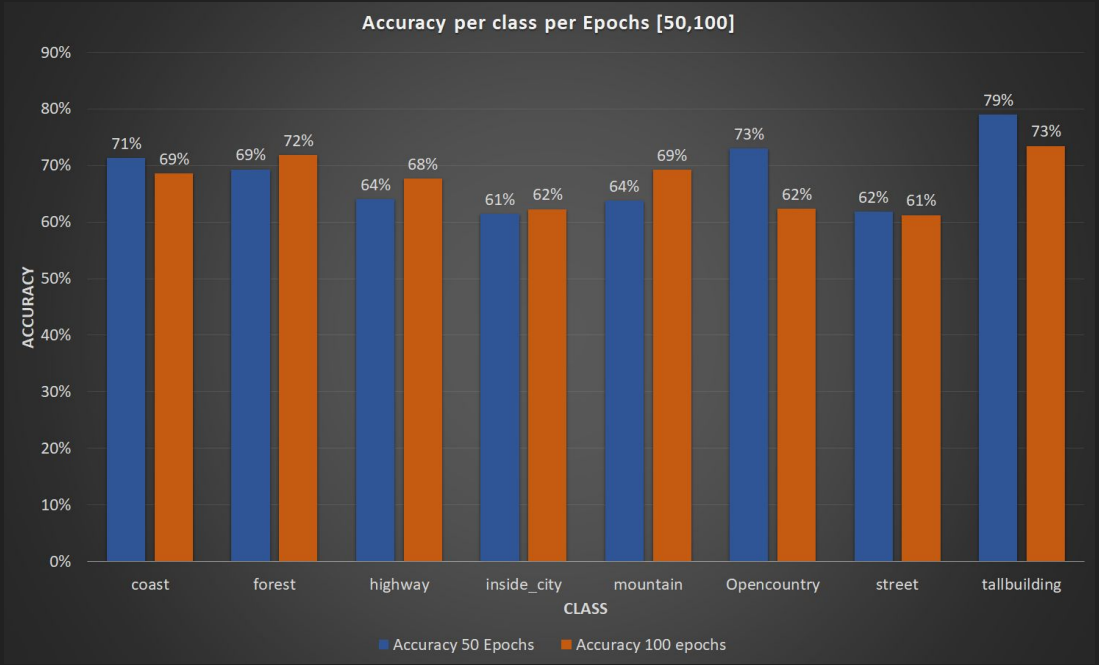


At 100 epochs, the best overall results come from size 128 patches with 32 batches. Batch size 64 seems to provide the most consistent results.

Patch size and batch size - 50 epochs vs 100 epoch

Comparing both,100 epochs has a higher average accuracy for forest, highway, inside_city and mountain classes, while 50 epochs has a better average accuracy for coast, opencountry, street and tallbuilding.

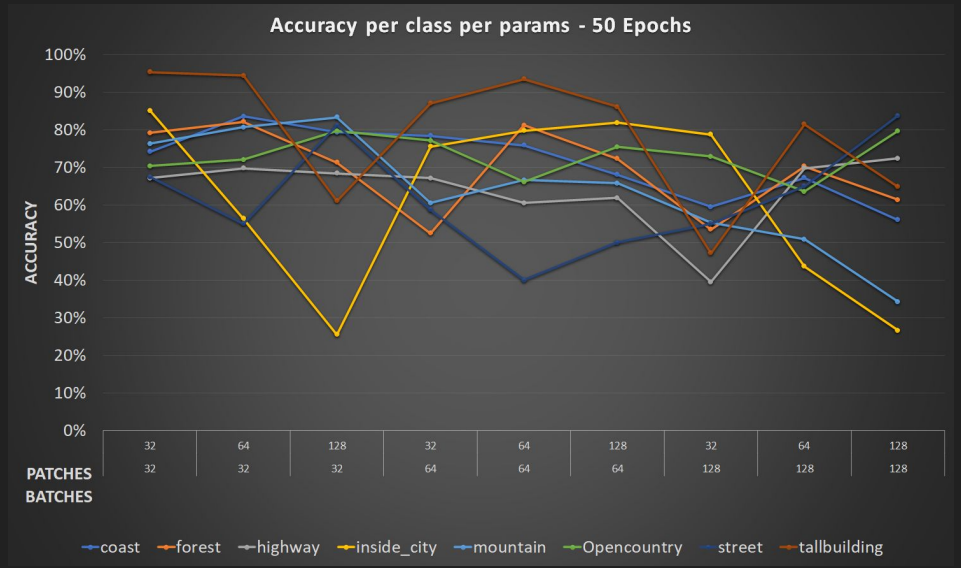
Of note is that the classes that perform better for 100 epochs contain more “noisy” features (tree leaves, “messy” urban features like windows and doors, jagged mountain ranges), while the classes that perform well on 50 epochs contain more “clean” features (clean horizon lines on the ocean, clean lines on buildings).



Accuracy per batch/patch size per class name (50 epochs)

It’s interesting that at 50 epochs, tallbuilding has a very high accuracy when patch_size is 64 regardless of its other values. This is probably related to features in buildings that are more recognizable at that size, but disappear at lower patch sizes, or become irrelevant at higher patch sizes.

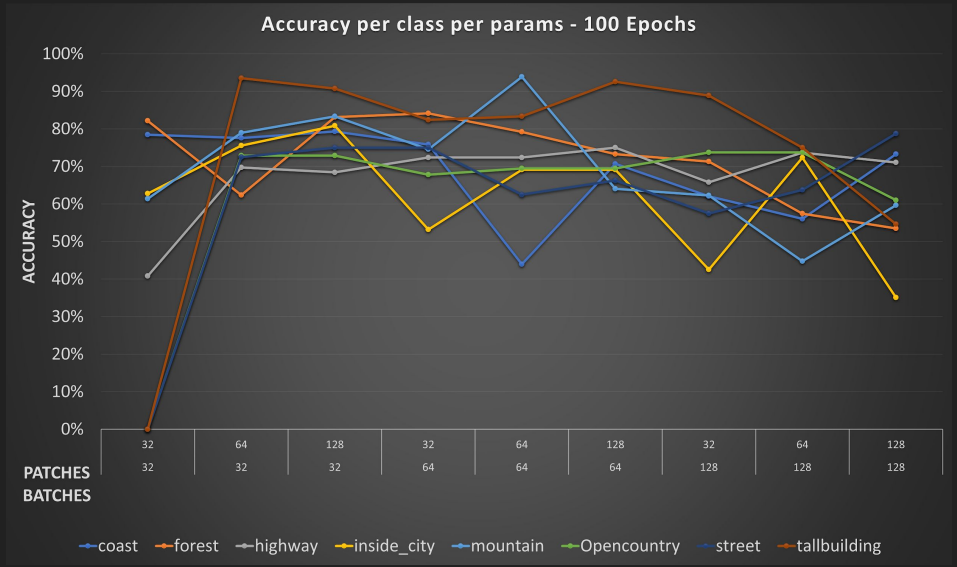
It is hard to get many conclusions from this graph and the following, as there is a lot going on, but it’s interesting to see trends like the tallbuilding’s class compared to other classes.



Accuracy per batch/patch size per class name (100 epochs)

For Patch_size and Batch_size 32, tensorflow ran out of training data required to complete street, tallbuilding, and opencountry.

Tallbuilding's preference for patch_size = 64 does not follow the 50 epochs trend.



Interestingly enough, the mountain class' graph forms a mountain, with its highest point at patch and batch size = 64.

Model used for using features generated by MLP layers in SVM and BoW

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
first (Reshape)              (None, 12288)             0
_________________________________________________________________
second (Dense)               (None, 2048)              25167872
_________________________________________________________________
3 (Dense)                    (None, 512)               1049088
_________________________________________________________________
4 (Dense)                    (None, 128)               65664
_________________________________________________________________
dense (Dense)                (None, 8)                 1032
=================================================================
Total params: 26,283,656
Trainable params: 26,283,656
Non-trainable params: 0
_________________________________________________________________
None
Done!
```

Extracted features from all layers but reshape and apply directly to SVM or thru all BoW

model_layer = Model(inputs=model.input, outputs=model.get_layer('4').output)

Accuracy score for SVM using features extracted from different level of layers - no patches

Network architecture	RBF	LINEAR
(32*32*3)>2048>512>128	32,21	27,380
(32*32*3)>2048>512	34,32	28,360
(32*32*3)>2048	33,95	29,730

As we can see it is possible to apply features generated by MLP to other algorithms. However from what we can see it is not generating good results.

Probably because MLP has more levels of freedom than SVM kernels, therefore SVM cannot take a good use of features from MLP.

Accuracy score for BoW using features extracted from different level of layers - no patches

Network architecture	RBF	LINEAR
(32*32*3)>2048>512>128	30,60	28,980
(32*32*3)>2048>512	27,75	27,880
(32*32*3)>2048	28,87	28,740

As we can see it is possible to apply features generated by MLP to other methods like BoW. However from what we can see it is not generating good results.

Probably because MLP has more levels of freedom than BoW method, therefore BoW cannot take a good use of features from MLP.

Accuracy for BoW using features extracted from different level of layers - **Patches**

Patch_size	Network architecture	RBF	LINEAR
32x32	(32*32*3)>2048>512>128	14,62	46,952
32x32	(32*32*3)>2048>512	14,62	43,355
32x32	(32*32*3)>2048	14,62	46,950

Using patches of images did show an important boost in results for the SVM with linear kernel applying BoW algorithm in contrast with using whole images. However, these results are still far from those of the MLP alone.
It is also to be noted that RBF performed poorly for this experiment.