Cats and Dogs classification

Using 5 convolutional layers and tuning hyper-parameters, validation accuracy of 85.62% is achieved against training accuracy of 92.18%. VGG style filter depths provided best results according to the experiments. Experiment 15 provided the best results.

Pre-processing:

The input files were separated in two different folders train and validation. Division is such that 9000 images are kept for training and 3500 for validation for each cats and dogs classes.

Keras **ImageDataGenerator** class is used to generate batches of tensor image data with real-time data augmentation. The data is be looped over (in batches) from the source directories. Transformations like zoom and flip are used to augment data for training. This technique ensures that different variations of the same data are fed to the network for training that result in better generalization. Furthermore data normalization is also done as part of preprocessing.

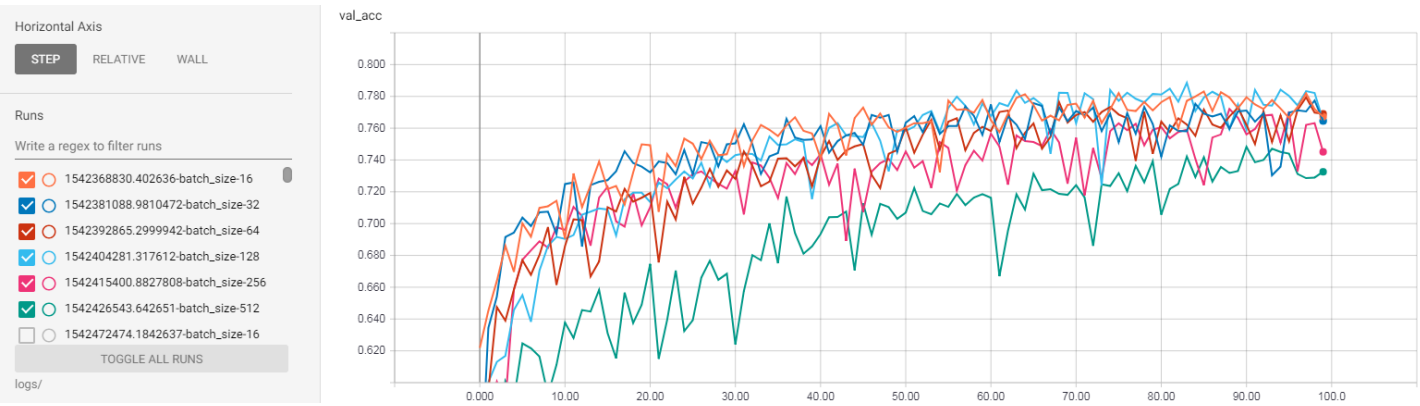
Following are the results of various experiments

experiment #	Description	epochs	train accuracy	validation accuracy
First using the same network architecture as provided in the project, impact of various hyper-parameters was measured. See results of experiment number 1 to 6. We observe that validation accuracy is best with batch size 128. We select batch size 128 for further experiments. Batch size 512 performed worst of all. See Graph-1 for validation accuracy plots.				
1	Batch Size= 16 (Rest is default)	94	0.7951	0.7723
2	Batch Size= 32 (Rest is default)	94	0.8188	0.7358
3	Batch Size= 64 (Rest is default)	94	0.7977	0.7680
4	Batch Size= 128 (Rest is default)	94	0.8051	**0.7842**
5	Batch Size= 256 (Rest is default)	94	0.7749	0.7509
6	Batch Size= 512 (Rest is default)	94	0.7402	0.7451
Experiment 7-10 list results of different weight initializers. Experiment 7 is same as experiment 4 because glorot_uniform is default weight initializer. The default weight initializer wins again. See Graph-2 for validation accuracy plots.				
7	Batch Size= 128, weight initializer= glorot_uniform (Rest is default)	94	0.8051	**0.7842**
8	Batch Size= 128, weight initializer= glorot_normal (Rest is default)	94	0.7973	0.7803
9	Batch Size= 128, weight initializer= he_uniform (Rest is default)	94	0.7843	0.7772
10	Batch Size= 128, weight initializer= he_normal	94	0.8032	0.7616

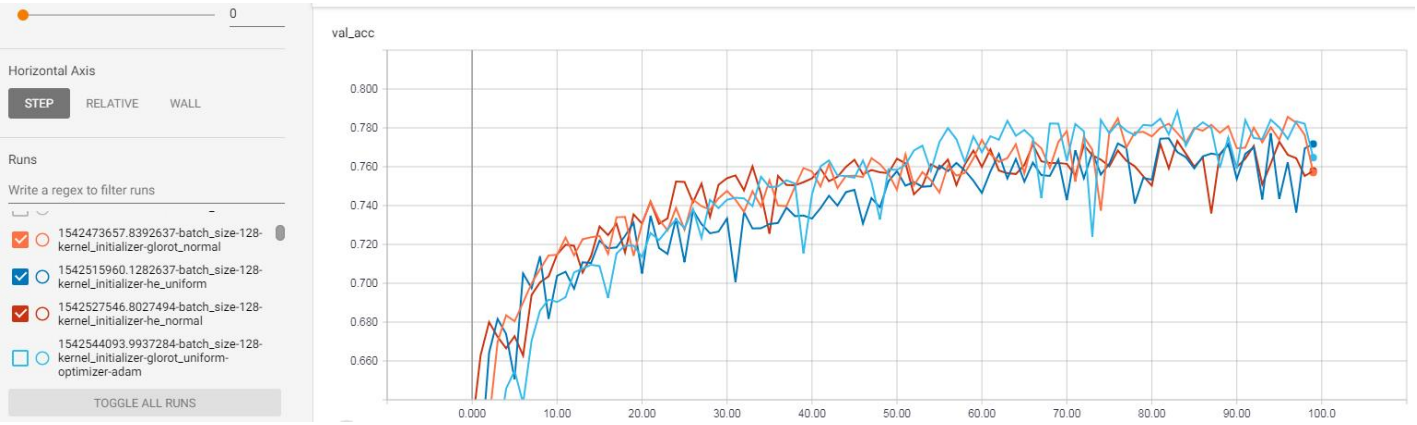
	(Rest is default)			
Experiment 11-14 list results of different optimizers Experiment 11 is same as experiment 4&7 because adadelta is default optimizer. We observe that adam perform better so we select it for further experiments. See Graph-3 for validation accuracy plots.				
11	Batch Size= 128, weight initializer= glorot_uniform, optimizer=adadelta (Rest is default)	94	0.8051	0.7842
12	Batch Size= 128, weight initializer= glorot_uniform, optimizer=adam (Rest is default)	94	0.8010	**0.7939**
13	Batch Size= 128, weight initializer= glorot_uniform, optimizer=adamax (Rest is default)	94	0.7721	0.7397
14	Batch Size= 128, weight initializer= glorot_uniform, optimizer=nadam (Rest is default)	94	0.8264	0.7923
Now the network configuration is changed keeping convolutional layers as 5. Changed parameters following VGG style. Less number of filters in beginning and more number of filters towards end in convolutional layers. Also some increase in the dense layer size. See Table-1 for model summary Validation accuracy is increase with the new network configuration. See Graph-4 for validation accuracy				
15	Batch Size= 128, weight initializer= glorot_uniform, optimizer=adam With increased parameters (Rest is default)	82	0.9218	**0.8562**
Add more parameters to see if validation accuracy can be pushed beyond 85%. See Table-2 for model summary. Validation accuracy is still staying below as compared to experiment 15 (See Graph-4)				
16	Batch Size= 128, weight initializer= glorot_uniform, optimizer=adam Further increased parameters in the configuration of experiment 15 (Rest is default)	30	0.8672	0.8290
Using the same network as of experiment 15, decrease the learning rate to see if the overfitting (gap between validation and training accuracy) decreases. Results showed no improvement.				
Using the same network as of experiment 15, decrease the dropout from 0.5 to 0.2 to see if learning can be increased. Results showed no improvement.				

Validation Accuracy graphs

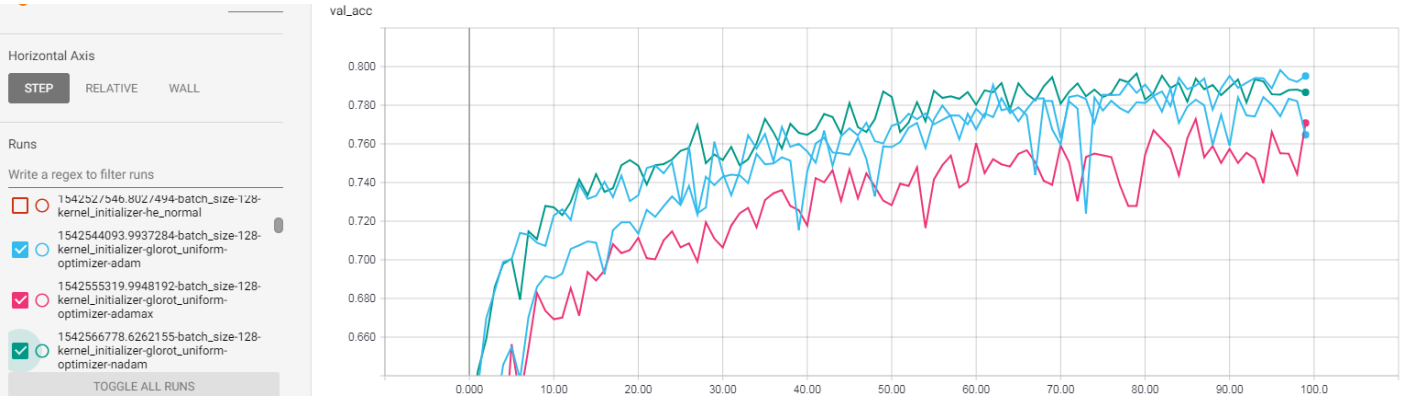
Graph-1: Validation accuracy for experiments 1 to 6 for different batch sizes



Graph-2: Validation accuracy for experiments 11 to 14 for different optimizers



Graph-3: Validation accuracy for experiments 7 to 10 for different weight initializers



Graph-4: Validation accuracy for experiments 15(magenta) &16 (blue) after increased parameters

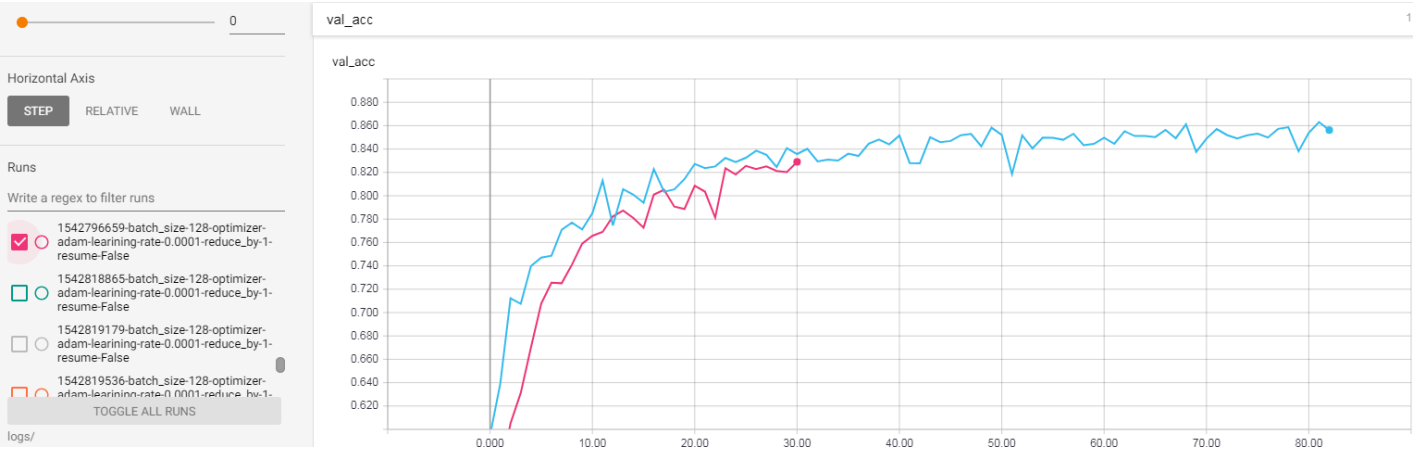


Table-1: Experiment 15

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 16)	448
conv2d_2 (Conv2D)	(None, 64, 64, 16)	2320
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 16)	0
conv2d_3 (Conv2D)	(None, 32, 32, 32)	4640
conv2d_4 (Conv2D)	(None, 32, 32, 32)	9248
conv2d_5 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 32)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 128)	1048704
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129

Total params: 1,074,737
Trainable params: 1,074,737
Non-trainable params: 0

Table-2: Experiment 16

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 64, 64, 32)	896
conv2d_2 (Conv2D)	(None, 64, 64, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 32, 32, 32)	0
conv2d_3 (Conv2D)	(None, 32, 32, 64)	18496
conv2d_4 (Conv2D)	(None, 32, 32, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 64)	0
conv2d_5 (Conv2D)	(None, 16, 16, 128)	73856
flatten_1 (Flatten)	(None, 32768)	0
dense_1 (Dense)	(None, 128)	4194432
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 128)	16512
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 4,350,497
Trainable params: 4,350,497
Non-trainable params: 0