Convolutional Neural Networks Assignment Report

Nikhil Kumar Sampath

Cats Vs Dogs:

Objective:

To build a convolutional neural network that can recognize cat and dog images and also to get an idea of what kind of impact is the training size going to have during the model-building phase.

Model Building:

In total, we built 15 models with varied layers, nodes, optimizers and other hyper tuning parameters.

We are categorizing the models built into **Scratch Models** and **Pre-Trained Models**

Below are the hyper tunning parameters and the performance evaluation of the models trained from “scratch” and the findings from the models therefore built.

Hyper Tunning Parameters: (Scratch Models)

#No.	Input Layers	Filters	Filter Size	Optimizer	Training	Validation & Test	Dropout
Model 1	5	32 to 256	3	Adam	1000	500 & 500	-
Model 2	5	32 to 256	3	Adam	1000	500 & 500	0.5
Model 3	6	32 to 512	3	Adam	1000	500 & 500	0.5
Model 4	5	64 to 1024	3	Adam	1000	500 & 500	0.6
Model 5	5	32 to 256	3	Adam	2000	500 & 500	0.5
Model 6	5	32 to 256	3	Adam	2000	500 & 500	0.5
Model 7	5	32 to 256	3	Adam	3000	500 & 500	0.5
Model 8	5	32 to 256	3	Adam	3000	500 & 500	0.5
Model 9	5	32 to 512	3	Adam	3000	500 & 500	0.5
Model 10	**5**	**32 to 256**	**3**	**Adam**	**5000**	**500 & 500**	**0.5**

#No.	Max Pooling	Strides	Padding	Augmented Images	Test Performance (Loss, Accuracy)
Model 1	Yes (Pool Size = 2)	-	-	-	(0.645, 0.614)
Model 2	Yes (Pool Size = 2)	-	-	Yes	(0.601, 0.712)
Model 3	Yes (Pool Size = 2)	-	-	Yes	(0.609, 0.692)
Model 4	Yes (Pool Size = 2)	-	-	Yes	(0.666, 0.652)
Model 5	Yes (Pool Size = 2)	-	-	Yes	(0.445, 0.860)
Model 6	-	Yes (Strides = 2)	-	Yes	(0.608, 0.650)
Model 7	Yes (Pool Size = 2)	-	-	Yes	(0.495, 0.818)
Model 8	Yes (Pool Size = 2)	Yes (Strides = 2)	-	Yes	(0.425, 0.856)
Model 9	Yes (Pool Size = 2)	Yes (Strides = 2)	Yes (Same)	Yes	(0.551, 0.782)
Model 10	**Yes (Pool Size = 2)**	**-**	**-**	**Yes**	**(0.182, 0.920)**

Findings:

- When the training size was 1000 samples for the model to learn the features the first four models didn't have higher accuracy but if noted when the model was trained with augmented images i.e., Model 2 the model's accuracy was the best among all the other four models. By this, we can say that Data Augmentation is an efficient way to improvise the model's performance.

- Models 3 and 4 had a change to their filters and the layers i.e., Model 3 was 6 Layers with filters ranging from 32, 64, 128, 256, 256 and 512 whereas Model 4 was with 5 Layers and the filters ranging from 64, 128, 256, 512 and 1024. The increase in filters further and adding layers didn't count to be a better step to increase the model's performance.

- The training sample was further increased to 2000 samples for Models 5 and 6, here we can see that Model 5 had 86% Accuracy and 44.5% Loss. This was the same model which received 71.2% Accuracy with 1000 training samples.

As soon as the same model got to look at more samples plus the augmented versions of the images it got to learn more thereby being able to correctly recognize the images.

- Model 6 didn't have a pooling layer instead of it we had strides as a mechanism to decrease the spatial dimensionality. This change in mechanism from pooling to strides alone didn't show any significant improvement in the model's performance.

- Model 7 was again the replica of the previous models 2 and 5 with just the training sample increasing to 3000, it was interesting to observe when the training sample was increased here from 2000 to 3000 the model's accuracy decreased from 86% to 81.8%, here we can again come up with one observation that not just increasing the sample size will improvise the performance, it's key to choose the right sample size so that the model gets to train well and generalize even better on unseen data.

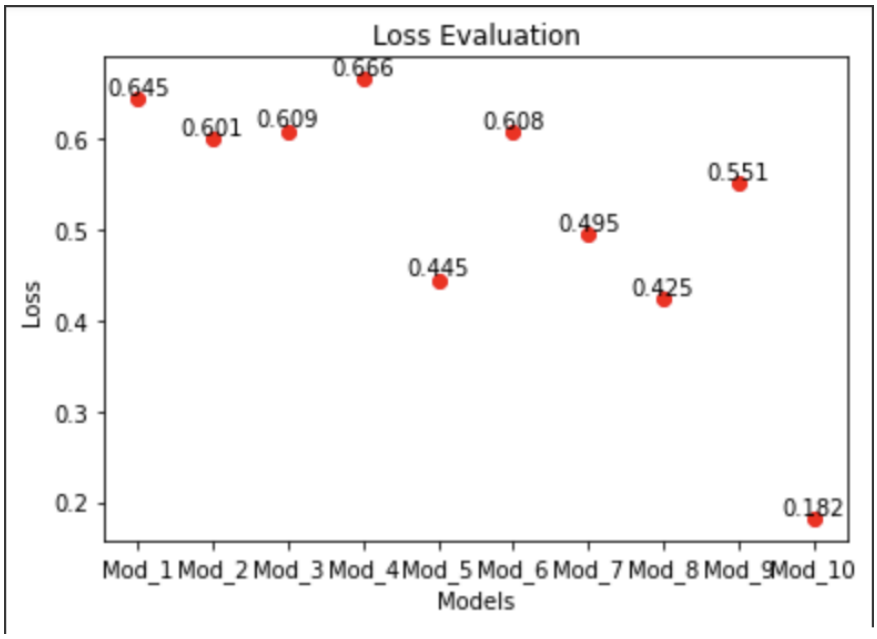
- Model 8 was built with Max Pooling along with Strides this was again a good combination this combination has changed the accuracy and increased from 81.8% to 85.6%.

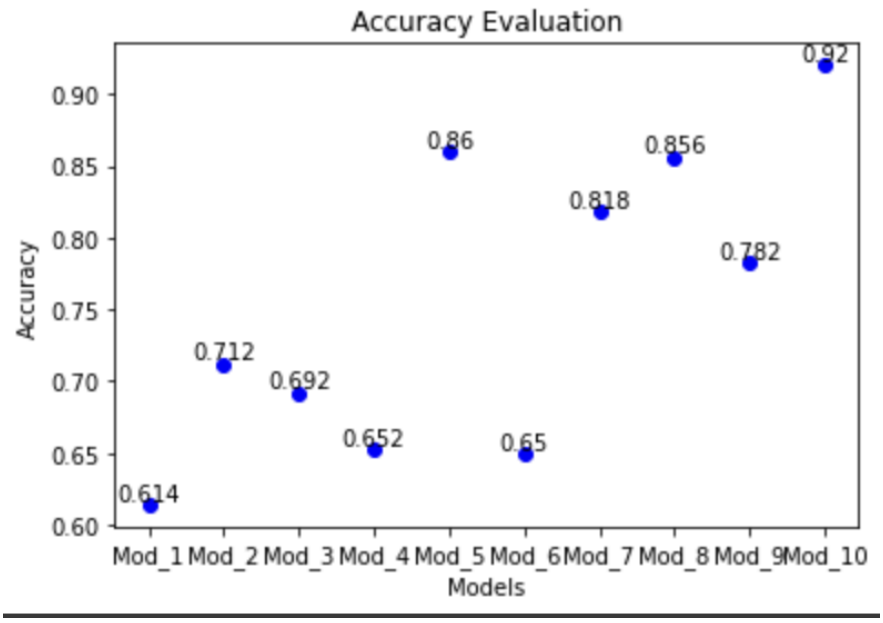
- Model 9 was built with padding turned on; this model didn't have a good performance there was a decline in the model's performance.

- Finally, when the sample was increased to 5000 the models which were having the highest accuracy among different samples were used i.e., Model 2 and Model 5 with an increase in sample size to 5000. This led to an increase in Accuracy to 92% and a Loss of 18.2%

Final Comments:

So, to answer the main question which is to establish the relationship between training sample size and choice of the network, we could say that there is a significant relationship between these two i.e., the size of the sample and the network which we build and use to train the model. Max Pooling Operation with Dropout and Use of Augmented Images counted to be an efficient way in most of the models to see a spike in accuracy and model performance.





Pre-Trained Network:

VGG-16 was used as a pre-trained network for the image recognition task, VGG net is a diversely trained model on thousands and lakhs of images in different categories. It's a powerful pre-trained network which is used widely across the world for image recognition tasks.

We built 5 models out of which 2 of them were the hyper-tuned versions of the previous models.

Hyper Tunning Parameters: (Pre-Trained Models)

#No.	Dense Layer	Training Size	Optimizer	Validation & Test	Dropout
Model 1	1 (256 Nodes)	1000	rmsprop	500 & 500	0.5
Model 2	1 (256 Nodes)	1000	rmsprop	500 & 500	0.5
Model 3	1 (256 Nodes)	1000	rmsprop	500 & 500	0.5
Model 4	1 (256 Nodes)	5000	Adam	500 & 500	0.5
Model 5	1 (256 Nodes)	5000	Adam (1e-5)	500 & 500	0.5

#No.	Pre-Trained Weights Update Set to False	Freeze Layers	Augmented Images	Test Performance (Loss, Accuracy)
Model 1	No	False	No	(12.350, 0.956)
Model 2	Yes	False	Yes	(5.111, 0.958)
Model 3	Yes	True	Yes	(9.085, 0.966)
Model 4	Yes	False	Yes	(0.236, 0.986)
Model 5	Yes	True	Yes	(0.083, 0.994)

Findings:

- Here in the pre-trained network as well training sample size played a prominent role in determining the model's learning characteristics and its performance on the unseen data.

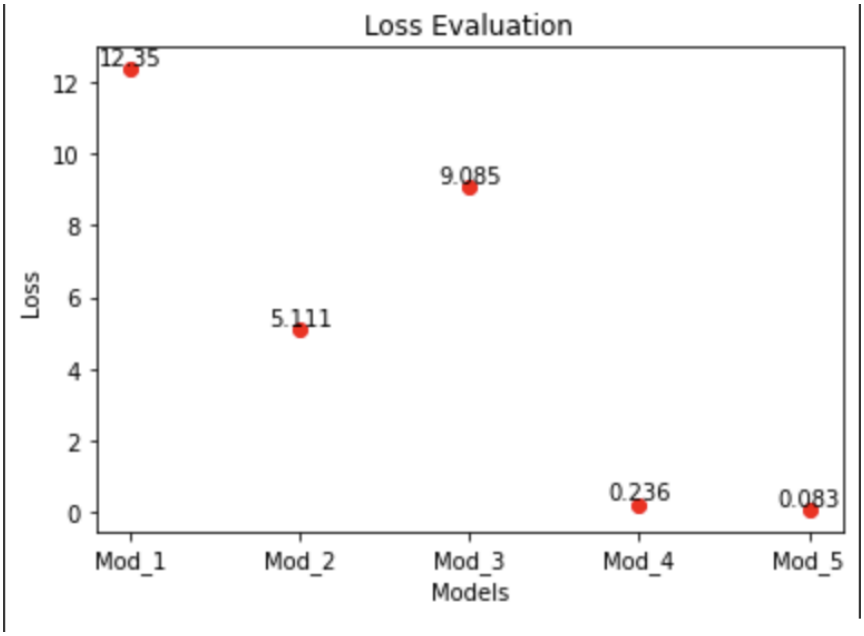
- Although rmsprop is one of the best optimizer functions to use in building a convolutional neural network, Adam still has the upper edge because of its unique mix of using Momentum and rmsprop together to optimize the neural network.

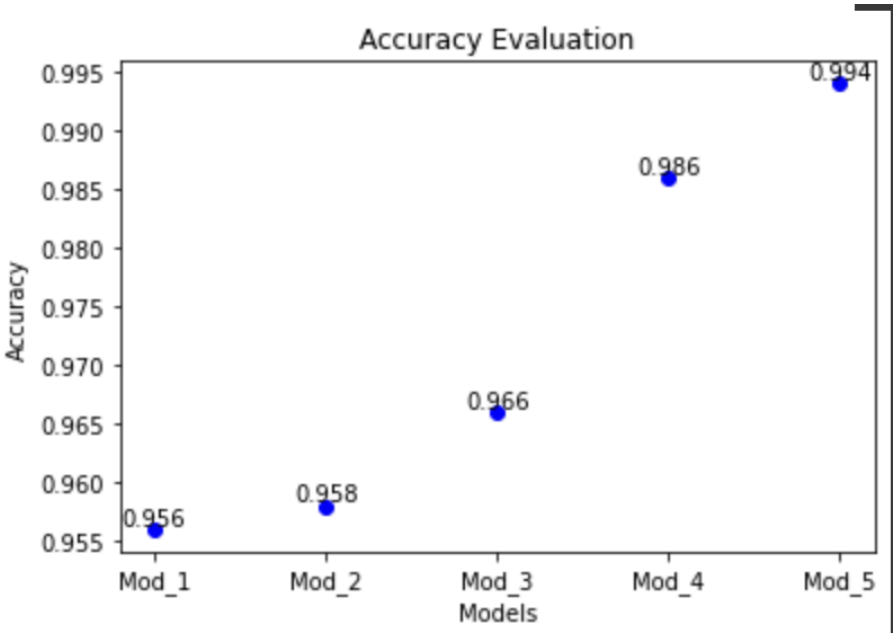
- By not allowing the pre-trained network to not update its weights we are not letting the pre-trained network lose its trained weights, we are just letting the model focus more on training the densely connected classifier layer at the end. This has been a game-changer as we can see Models 2 and 3 have a good improvement in their performance with this measure being set to False. It's also a useful technique to avoid overfitting.

- Since the pre-trained networks are trained on broader categories of images, by freezing the initial layers of the model we are freezing the general categorization layers and forcing the model to especially focus on more specifically towards the image recognition task for which we are building the model. This has proved to be a good measure as we can see in the fine-tuned models i.e., Model 3 and Model 5 these are the highest accurate models within their definite sample sizes.

Final Comments:

Even the pre-trained networks have a significant relationship with the sample size when the sample size is increased from 1000 to 5000, we could see a good change in the model's performance and the model was having a minimal loss value as well. Providing more data for the model to learn with and also an augmented version of the images as well proved to be an efficient way to increase the model's accuracy. Freezing the initial layers of the pre-trained and not letting the pre-trained network update its weights during the training have been proven to be an effective way to control the model from overfitting and to generalize well on the unseen data.


Loss Evaluation
12.35
9.085
5.111
0.236
0.083
Loss
0
2
4
6
8
10
12
Mod_1
Mod_2
Mod_3
Mod_4
Mod_5
Models


Accuracy Evaluation
0.994
0.986
0.966
0.958
0.956
Accuracy
0.955
0.960
0.965
0.970
0.975
0.980
0.985
0.990
0.995
Mod_1
Mod_2
Mod_3
Mod_4
Mod_5
Models