Assignment 2

Jacob Joy

Kent State University

Dr. CJ Wu

3/23/2025

Setup

The initial setup for computer vision using the cat's vs dog's data set required connecting google drive that contained the data to google colab. The next set was to create multiple training sets using different sample sizes. These random samples were 1000, 1500, 2000 each set had 50% dogs and 50% cats. A reduced validation set, and reduced testing set as requested was created containing 500 samples each with 50% dogs and 50% cats.

Procedure

1. Data preprocessing was conducted on each training, validation, and test set which defined the batch size that would be used and the size of the images (180x180). This step needs to be conducted whenever a training set is changed or when a batch size is changed. This is done for each different value of training data.

2. Building of many different models without any data augmentation.

3. The models were compiled when needed and trained using a callback to keep track of the best validation loss and epoch. To better see the results additional formatting was applied to the checkpoint file that allowed for the epoch number to be amended to the file.

4. The training and validation; accuracy and loss were recorded.

5. The best epoch model was loaded from the checkpoint and used to obtain accuracy and loss when used on the test dataset.

6. This process was repeated for each model.

7. Next, data augmentation was added to many different models that were developed based on the results of the models that had no data augmentation applied. Steps 3 – 6 are repeated with an increase in epochs used to 100 because of the later overfitting due to the data augmentation.

8. Pretrained models were used next with VGG16 being selected as was suggested in the text.

9. The features of VGG16 need to be selected and processed using VGG16 functions. Labels will also need to be separated for our data to be read correctly when using VGG16.

10. The pretrained model is used along with steps 3 – 6 for each version of the training data and batch size used.

11. Another model is created using VGG16 as a layer of the model to run the dense layers together. Steps 3 – 6 for each version of the training data and batch size used.

12. Finally, another model is created using VGG16 but the bottom 4 layers are unfrozen in order to update based on the training data being used. Again steps 3 – 6 for each version of the training data and batch size used.

Summary

Initially only 1000 samples were in the training data set and many different techniques were utilized to try and find the best performing model with such a low sample size. Techniques such as padding, dropout, batch Normalization, adding additional conv2D layers, data augmentation and residuals were all used in

verting combinations. What was found was no matter the combination of optimization techniques data augmentation was a must in order to improve data validation accuracy. After that was determined it was found that using the recommended optimization that the book provided with a change to the batch size to 16 produced the best result on Validation of 75.6% while staying close to the accuracy of the training data. Some other methods were experimented with one interesting experiment using multiple additional conv2D layers. At first this model did not really train as indicated in the results table. For the model to train residuals were added back into the model. This model produced high results on the training data and similar to the best model on validation.

Next a training set of 1500 was used and based on the previous methods the best models that had no data augmentation were tested and showed improvement to validation accuracy similar to a lower sized training set but including data augmentation. Again, data augmentation was shown to be beneficial, and it was determined that the book example with no changes provided the best result on validation which was 78.8%. It is important to note that this was similar to adding more dropout after each max pooling layer and including padding as well as having the batch size be 16 or 32.

Next a training set of 2000 was used and the same models as with the 1500 data set were used it was determined that the use of multiple conv2D layers with residuals added back provided the best validation accuracy at 83.8%. One additional experiment was conducted using multiple layers but slowing down the training rate this showed an overall decrease in Accuracy but did decrease the difference between validation loss and training loss. However, the difference was very minor for such a decrease in accuracy going from 83.3% - 72.4%.

Three pretrained models were used: the whole model, data augmentation, and fine tuning. For the 1000 training samples the pretrained fined tuned model had a validation of 97.6%. For the 1500 training samples again the pretrained fined tuned model had a validation of 98.4%. For the 2000 training samples the pretrained model frozen but using a dense layer and a slower learning rate of 1e-10 resulted in a 97.4% validation accuracy. This is what I consider to be the best optimized model even though its validation accuracy is a little lower in comparison to other models. The reason being that the loss rates are much lower, providing a more accurate and confident model. This model performed at 97.8% accuracy on the test data.

Conclusion

In conclusion if you have limited amounts of data it appears to be best to use a pretrained model, however, additional steps will still need to be taken to optimize the pretrained network for use with your specific use case. In general, it is always better to have as much training data as you can, however there is a trade off in time and computational power when you increase the training data amount. Additionally, increasing the validation and test data amounts can also help highlight better real use cases of a model as the loss amounts may not be as high when these amounts are increased. While I was finishing this lab a thought occurred to me to include data augmentation with the unfrozen layers and a quick test was done and while this did not improve validation accuracy it greatly lowered the loss on validation, training, and test. This would be a future area to explore.

Model	Training_Data	Validation_Data	Test_Data	Training_Acc	Training_Loss	Validation_Acc	Validation_Loss	Delta Loss Train_Val	Test_Acc	Test_Loss
Base Model	1000	500	500	0.6388	0.6473	0.594	0.6536	-0.0063	0.64	0.647
Base and layer	1000	500	500	0.6919	0.6307	0.652	0.6458	-0.0151	0.634	0.672
Base and padding	1000	500	500	0.7546	0.5483	0.686	0.5975	-0.0492	0.68	0.605
Base and dropout	1000	500	500	0.688	0.5878	0.642	0.6443	-0.0565	0.644	0.652
Base and batchNormalization	1000	500	500	0.8225	0.4572	0.552	0.6928	-0.2356	0.554	0.686
Base, batchNormalization, and padding	1000	500	500	0.7339	1.0764	0.562	0.6946	0.3818	0.56	0.697
Base, padding, dropout	1000	500	500	0.7432	0.5206	0.724	0.5549	-0.0343	0.694	0.572
Base, padding, and layer	1000	500	500	0.7144	0.5744	0.63	0.6581	-0.0837	0.67	0.653
Base, padding, layer, and batchnormalization	1000	500	500	0.6277	1.0979	0.514	0.7003	0.3976	0.518	0.702
padding dropout and batch no layer	1000	500	500	0.754	0.5212	0.578	0.6774	-0.1562	0.618	0.687
base padding dropout and layer	1000	500	500	0.7217	0.5597	0.664	0.6494	-0.0897	0.678	0.654
Data Augmentation dropout before dense	1000	500	500	0.7683	0.4984	0.744	0.5282	-0.0298	0.736	0.602
Data Augmentation dropout before dense, batch 16	1000	500	500	0.7748	0.4732	0.788	0.5005	-0.0273	0.756	0.5384
Data Augmentation dropout before dense, padding same.	1000	500	500	0.8001	0.4604	0.742	0.5993	-0.1389	0.76	0.641
just padding	1000	500	500	0.8067	0.4499	0.718	0.5703	-0.1204	0.696	0.581
Data Augmentation dropout after pooling and before dense, padding same.	1000	500	500	0.8434	0.3934	0.754	0.5294	-0.136	0.75	0.553
Try above but add padding to the pooling	1000	500	500	0.7748	0.4783	0.754	0.5776	-0.0993	0.728	0.576
An extra layer	1000	500	500	0.5619	0.6864	0.598	0.6684	0.018	0.618	0.675
multipler layers	1000	500	500					0	0.5	0.693
multiple layers added residuals	1000	500	500	0.814	0.4085	0.738	0.6096	-0.2011	0.7	0.683
multiplr layers added residuals added batchnormalization	1000	500	500	0.5848	1.5388	0.576	0.7499	0.7889	0.554	0.786
above with dropout	1000	500	500	0.7972	0.5298	0.736	0.6558	-0.126	0.728	0.649
dropout	1000	500	500	0.7487	0.4971	0.644	0.6347	-0.1376	0.624	0.671
dropout and batach	1000	500	500	0.7018	0.8784	0.582	0.6798	0.1986	0.588	0.675
Base and padding	1500	500	500	0.7308	0.5479	0.712	0.5658	-0.0179	0.706	0.554
Base, padding, dropout	1500	500	500	0.8001	0.4357	0.752	0.5782	-0.1425	0.702	0.591
Data Augmentation dropout before dense	1500	500	500	0.8237	0.391	0.788	0.4684	-0.0774	0.79	0.483
Data Augmentation dropout after pooling and before dense, padding same. Ba	1500	500	500	0.8018	0.4663	0.814	0.4465	0.0198	0.758	0.576
Data Augmentation dropout after pooling and before dense, padding same.	1500	500	500	0.8528	0.3474	0.796	0.4545	-0.1071	0.78	0.524
multiple layers added residuals	1500	500	500	0.7699	0.4808	0.77	0.4936	-0.0128	0.714	0.558
Base and padding	2000	500	500	0.7799	0.4909	0.698	0.6275	-0.1366	0.684	0.566
Base, padding, dropout	2000	500	500	0.7822	0.4661	0.736	0.5793	-0.1132	0.726	0.597
Data Augmentation dropout before dense	2000	500	500	0.885	0.276	0.85	0.3991	-0.1231	0.816	0.431
Data Augmentation dropout before dense 16 batch size	2000	500	500	0.823	0.4012	0.814	0.4064	-0.0052	0.762	0.494
Data Augmentation dropout after pooling and before dense, padding same.	2000	500	500	0.8452	0.3604	0.834	0.4377	-0.0773	0.812	0.432
multiple layers added residuals	2000	500	500	0.8621	0.3463	0.838	0.4044	-0.0581	0.812	0.448
multiple layers added residuals, slower learning rate	2000	500	500	0.7568	0.4893	0.724	0.5217	-0.0324	0.726	0.548
Pretrained	1000	500	500	0.991	0.8189	0.974	4.1621	-3.3432	0.966	5.433
Pretrained batch 16	1000	500	500	0.9773	2.2901	0.95	8.6734	-6.3833	0.9375	14.8806
Pretrained, own dense	1000	500	500	0.9843	1.9293	0.972	3.4481	-1.5188	0.968	4.913
Pretrained , own dense, batch 16	1000	500	500	0.9814	1.7634	0.974	4.1392	-2.3758	0.972	6.693
Pretrained,fine tune	1000	500	500	0.9981	0.1552	0.976	3.1833	-3.0281	0.98	3.405
Pretrained, fine tuned, batch 16	1000	500	500	0.9953	0.3047	0.976	4.3661	-4.0614	0.976	6.215
Pretrained	1500	500	500	0.9953	0.907	0.962	4.0546	-3.1476	0.974	6.261
Pretrained batch 16	1500	500	500	0.9947	1.1252	0.972	3.5741	-2.4489	0.97	6.283
Pretrained, own dense	1500	500	500	0.9914	0.371	0.976	2.8322	-2.4612	0.978	4.238
Pretrained , own dense, batch 16	1500	500	500	0.9932	0.275	0.98	2.7154	-2.4404	0.976	4.1
Pretrained,fine tune	1500	500	500	0.9952	0.1007	0.984	1.8956	-1.7949	0.978	2.877
Pretrained, fine tuned, batch 16	1500	500	500	0.9947	0.2291	0.982	2.1004	-1.8713	0.987	3.246
Pretrained	2000	500	500	0.9782	2.4132	0.972	3.7203	-1.3071	0.966	4.86
Pretrained batch 16	2000	500	500	0.9778	2.765	0.964	4.341	-1.576	0.958	8.646
Pretrained batch 64	2000	500	500	0.8613	47.2641	0.964	2.9842	44.2799	0.962	6.221
Pretrained, own dense	2000	500	500	0.9746	3.0891	0.982	2.8258	0.2633	0.976	2.11
Pretrained , own dense, batch 16	2000	500	500	0.9792	1.2059	0.976	2.8657	-1.6598	0.976	2.228
Pretrained , own dense, batch 64	2000	500	500	0.9928	0.4747	0.974	3.477	-3.0023	0.982	2.387
Pretrained,fine tune	2000	500	500	0.9981	0.0474	0.98	2.2109	-2.1635	0.98	2.27
Pretrained, fine tuned, batch 16	2000	500	500	0.994	0.1765	0.98	1.8053	-1.6288	0.98	1.63
Pretrained, fine tuned, batch 64	2000	500	500	0.9935	0.2659	0.978	2.5788	-2.3129	0.974	2.803
Pretrained,fine tune slower learning rate	2000	500	500	0.9956	0.3959	0.98	2.2099	-1.814	0.98	2.27
Pretrained, fine tuned data augmentation	2000	500	500	0.9773	0.0951	0.964	0.1355	-0.0404	0.98	0.198
Pretrained, own dense slower learning rate	2000	500	500	0.9697	0.4184	0.974	0.6266	-0.2082	0.978	0.671
Pretrained,fine tune slower learning rate unfrezze 8 layers	2000	500	500	0.9815	0.3328	0.976	0.6652	-0.3324	0.976	0.622