DM873 Project 1

Niels Peter Roest$^{[niroe18]}$

SDU Odense

Abstract. I developed and trained a neural network capable of recognizing between a cat and a dog with a 73% accuracy.

Keywords: Computer Science · Neural networks

1 Intro

The object of this project is to create a neural network able to distinguish between cats and dogs.

2 The network

Initially i used the VGG-16 network structure, but while trying VGG-16 i encountered several issues with my hyperparameters, firstly the learning rate. I quickly found that Keras default learning rate of 1E-3 was quite big, and resulted in my network not learning anything after many epochs. The accuracy was dropping below 50% while training, which meant that a coin flip was better. This was solved by testing different learning rates, and finding that 1E-5 was appropriate for learning, since it now slowly encroached upon the correct value, while presenting more consistent accuracy along the way.

Another issue was the jumpiness, while training i found that my accuracy would jump up and down alot from epoch to epoch. This means that the data i was ingesting was simply too small to be representative of most of the data in the network. To solve this i increased the batch size from 32 to 128 images, which greatly increased the consistency of the results from each epoch.

While training i also needed a way to find out how many epochs were optimal, so i employed early stopping, which i set to check if the validation loss hasn't decreased within 5 epochs, in which case it restores the best weights and stops the training. At this point VGG-16 got me about 70% accuracy with around 50 epochs.

3 The new network

While training VGG-16, i found that it took a long time, and the results weren't too impressive compared to the time spent training. VGG-16 is a big network designed for thousands of classifications instead of just 2, so i crafted a smaller network to achieve the same task, but more effectively. This network looks like:

(150,150,3)− >Conv2D(32)− >MaxPool(2,2)− >Conv2D(64)− >MaxPool(2,2)− > Conv2D(128)− >MaxPool(2,2)− >Flatten− >Dense(1024)− >Dense(2,SoftMax).

All these use ReLU, except the output which is Softmax. Its important to note that i uses 2 output classes (and Softmax) instead of 1 output class (and sigmoid), this is mainly because i had a bug in tensorflow preventing me from observing the accuracy while training if i used binary classification. While running this network, i observed that my training accuracy was higher than my validation accuracy, this means that i was overfitting to the training data. The model's accuracy was 62%, while the training data accuracy was 65%. To solve this discrepancy in accuracy i employed image augmentations (and dropout).

4 The augmentations

The image augmentations used are image rotation, shear (dragging), zoom and later rescale (for normalizing colors, it was observed to help in accuracy). This was done to attempt to force the network to generalize the training data better, and thus create a better network. I also employed dropout on the first dense layer, initially with a probability of 20%, but increased to 50% after observing my training accuracy increase substantially more than my validation accuracy. While dropout did help, it did not fix the problem of overfitting, so i determined that i had to make the network smaller again.

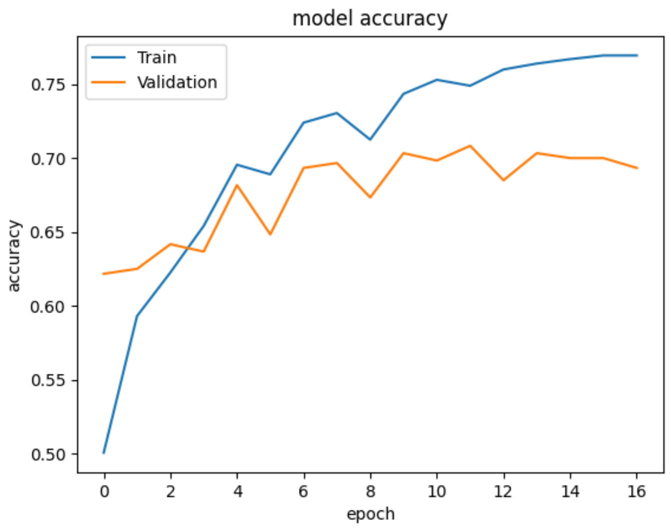
5 The new new network

The new new network looks like:

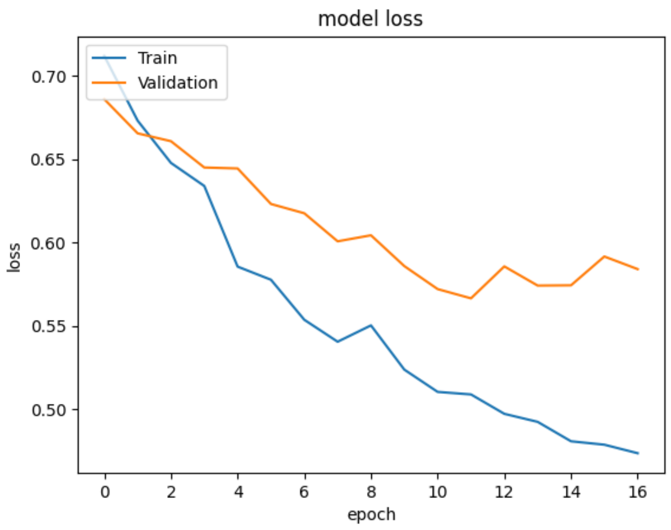
(200,200,3)− >Conv2D(32)− >MaxPool(2,2)− >Conv2D(64)− >MaxPool(2,2)− > Conv2D(128)− >MaxPool(2,2)− >Flatten− >Dense(512)− >Dense(2,SoftMax).

I also increased the image size to allow more details to be learned. These changes allowed the network to achieve a accuracy of 72.5% (on the test data) and loss of 52.6% after only 17 epochs (meaning it was alot faster to train and test on).

6 The training



As can be seen on the accuracy figure, the validation accuracy peaked at 11 epochs with my network, where if afterwards started to dwindle.



As can be seen on the loss figure, the least amount of loss for validation was also at epoch 11. Which meant that training beyond that point would only overfit the data, and not be helpful. It can also be seen, that the network overfits greatly, which means that my network could be futher improved by reducing the number of neurons, or using greater regularization.

7 Layer weight regularization

There are alot of tactics involved in forcing the network to learn less, and generalize more. One of which is L1 and L2 regularization, which attempts to penalize using more weights, thus force the remainder to generalize to learn more. This is great, as it theoretically improves the performance of neural networks. In practice, it also turns traning my network from a 15 minute task to a 4 hours task, decided against using it.

8 Conclusion

I have created a neural network capable of identifying if a dog or a cat is the focus of an image, with a 72% accuracy, while testing various hyperparameters along the way.