cs577 Assignment 4

Sahil Riyaz Sheikh(A20518693)

Department of Computer Science
Illinois Institute of Technology

November 12, 2022

Abstract

In this assignment, I'll work on two datasets i) Cats and Dog ii) CIFAR10.I'll perform classification task on both of datasets. There are few specific requirements I need to complete for each dataset and perform specific tasks on them. I'll make use of transfer learning with help of VGG16 model, inception blocks, residual blocks. I'll compute the graph for training loss and validation loss, training accuracy and validation accuracy. In the end I'll summarize my results with observed performance results.

1. Problem Statement:

I was familiar with the CIFAR-10 dataset, consisting of 60,000 images out which 10,000 are reserved for testing purposes. There are 10 classes in CIFAR-10 such as aeroplanes, horse, cats, etc. I was working on the Cats and Dog dataset for the first time. It is a large dataset consisting of 12,501 images of Cats and 12,501 images of Dogs. For this assignment I can only use 2000 images of Cats and 2000 images of Dogs. I'll make a model to implement binary classification on Cats and Dogs dataset, and for CIFAR-10, I'll create a model to implement multiclass classification.

2. Proposed Solution:

For Cats and Dogs, I'll make a not to deep neural network with several convolution layers and have dense layer with 1 node for output as there are only two classes. I'll make use of Relu as activation function in hidden layers and in the output Sigmoid will be the desired activation function. Lastly Binary cross entropy will be the suitable loss function.

For CIFAR-10, as it has 10 class, I'll one hot encode them before feeding them into the network. I'll try to keep the network not too deep and make use of convolutions layers along side with max pooling layers. For the dense layer, it will be a single layer with 10 nodes acting as the output layer. All the layers other than output layer will have Relu as activation function and lastly the output layer will have Sigmoid as its activation function.

For the Abalone dataset, I'll be using Softmax as the activation function in the output layer and since there are more than two classes in target, I need to one-hot-encode the target values. The other dense layers could have either sigmoid or Relu as activation function depending upon the performance of the network. I'll use Categorical cross entropy as loss function.

The output layer will have Softmax as activation function and the hidden layers will consist of sigmoid activation. I will use Categorical Cross Entropy loss and Kullback–Leibler divergence for loss function.

3. Implementation Detail:

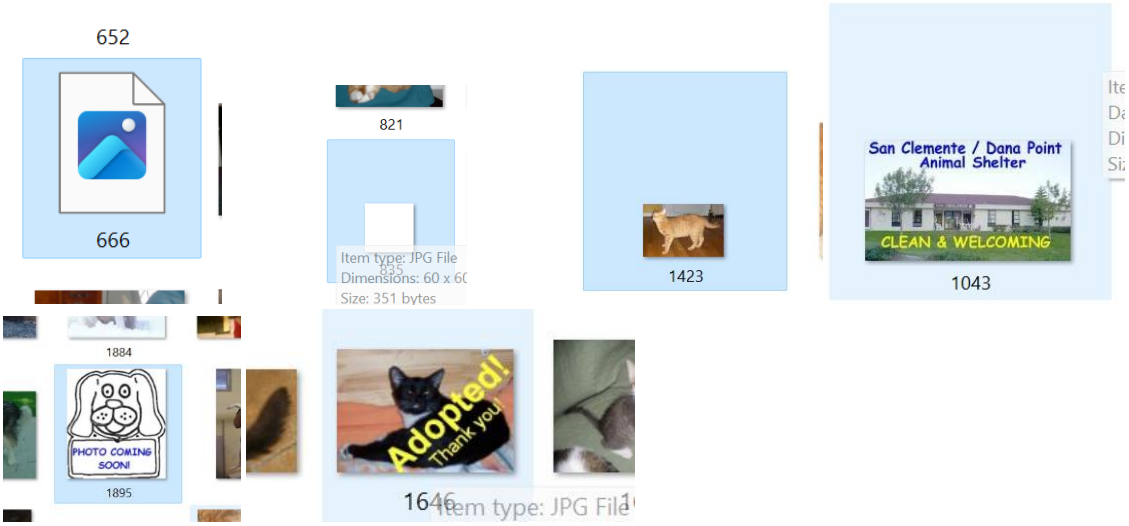
For loading the data for Cats and Dog, I made use of data generators. They work when the data is arranged in the following manner:

Data/
Data/Train/1.cats 2.dogs
Data/Vali/1.cats 2.dogs
Data/Test/1.cats 2.dogs

So after downloading, extracting the data, I re arranged it the specified manner to load the data using data generators. The first major issue I ran into was that I had not provided padding in my model which resulted in OOM error occurring stating: I had run out of memory so added padding to every layer to avoid this form a happening again.

The second major issue which I ran into was the model threw an error during training saying it was unable to identify a file. I was perplexed as this was first time, I encountered this type of error. After a few hours of trying

to find a solution, I decided to examine my data and found a corrupt file and some other files which were really small in size as well had text written on them in front of the faces of the animals due to which the model was unable to extract features.



These were some of the pictures which I found in my dataset which were causing a error during the training phase of the model. I replaced them as well as check the entire dataset and cleaned it.

Next part was implementing the model, recording the losses and accuracy, plotting the graph. Then training the model on training and validation data combined, and evaluate the performance on test set. I saved weights of this model. Now I visualized the out of different layers by creating a new model and utilizing the saved weights of trained model and visualize the few channels of different layers. Finally, I plotted the visualization of every layer using mat plot library.

Now I imported the VGG16 model from keras. applications library. I created two models and added the imported VGG16 model in both of these models. The first layers of this model had the VGG16 model layer with frozen convolution base. In one model the layers were untrainable and in other they were trainable. The deeper layers of these models we same as that of my first model as I had just replaced the first of my first model layers with layers from VGG16. Then I trained both the models on entire data and evaluated their performance.

In the end I created data generator to augment the images and feed this to the model with frozen weights and recorded its performance.

For CIFAR-10, this was relatively easier to code and implement. I had to download the CIFAR-10 and load it using the pickle library. As this dataset is stored in batches, I had to separately unpickle each batch, reshape the data and concatenate all the batches to together to form the training data. Test set is separate batch. Now I split the training data into validation data and training data. I created the model and trained it on training set and evaluated the performance. Then I trained this model on entire dataset, saved the weights and evaluated on test set. I added batch normalization to see the difference in results.

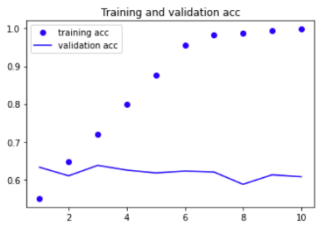
I created two models, one with inception block and other with residual block, trained them and evaluated their performances.

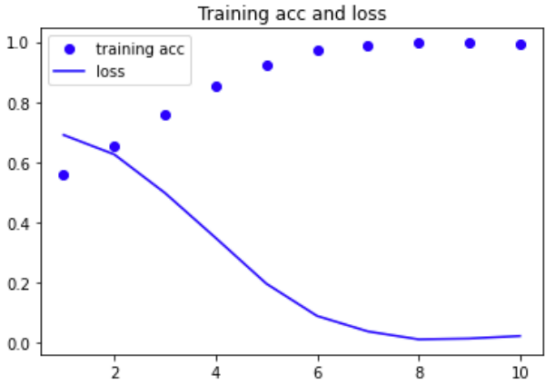
Both the codes are simple to run. One has to run all cells to obtain the desired outputs.

4. Results:

For Cats and Dogs dataset:

The model trained for 10 epochs. But the model overfits after 4th epoch and achieves an accuracy of 74% percent according to the graph. (right)






```
20/20 [==============================] - 1s 43ms/step - loss: 1.9176 - accuracy: 0.6275
```

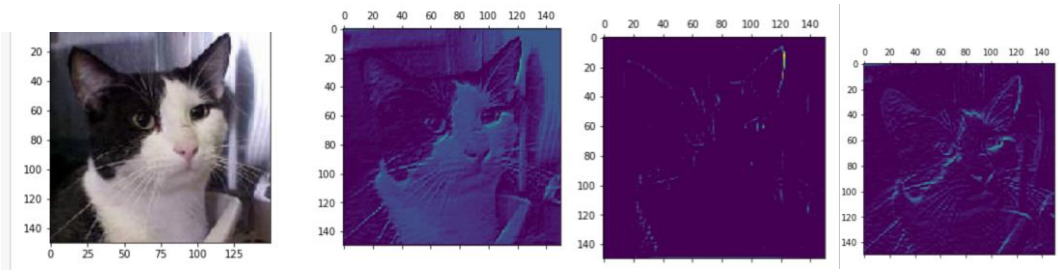
Out[42]: `[1.9176138639450073, 0.6274999976158142]`

The above graph depicts the model trained on entire dataset and provides a accuracy of 62.74% on evaluation.

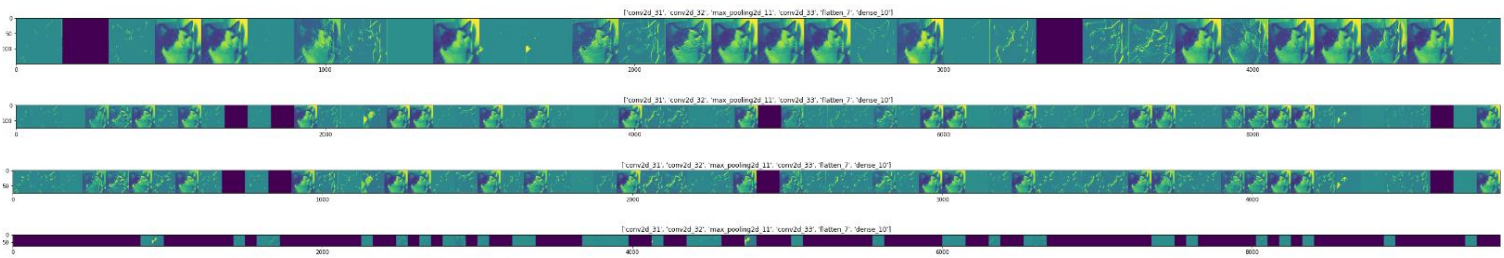
Model summary of new model with saved weights from previous model.

```
from keras.models import load_model
new_model = load_model("final_model.h5")
new_model.summary()
imgpath = 'C:/Users/sahil/Downloads/kagglecatsanddogs_5340/PetImages/Cat/3012.jpg'
```

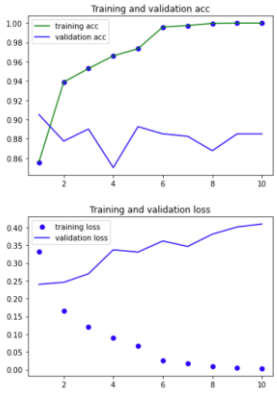
```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_31 (Conv2D)          (None, 150, 150, 32)      896

 conv2d_32 (Conv2D)          (None, 150, 150, 64)      18496

 max_pooling2d_11 (MaxPoolin  (None, 75, 75, 64)       0
 g2D)

 conv2d_33 (Conv2D)          (None, 75, 75, 128)       73856

 flatten_7 (Flatten)         (None, 720000)            0

 dense_10 (Dense)            (None, 1)                 720001

=================================================================
Total params: 813,249
Trainable params: 813,249
Non-trainable params: 0
```

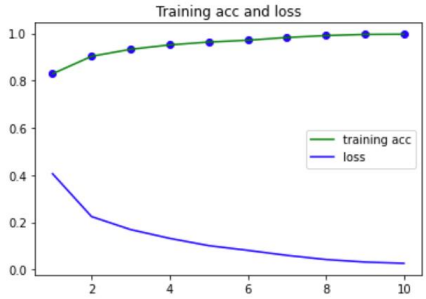


These are the visualizations from second layer. We can see the model detects the edges of cats



This the visualization of all the layers of in the network. We can see in the upper layers edges of cat is getting detected and in the deeper layer we can see finer features of the cat are being identified such as whiskers



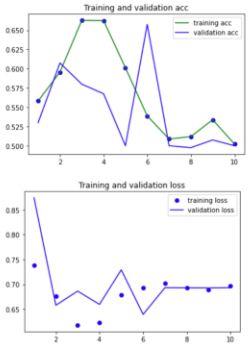


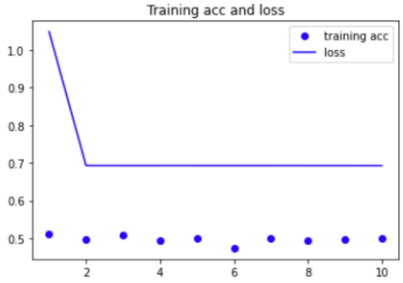

```
20/20 [==============================] - 1s 37ms/step - loss: 0.2760 - accuracy: 0.8925

[0.2760259807109833, 0.8924999833106995]
```

The above are the results from VGG16 model with frozen weights. The graph on left shows training on just training data and on right shows the graph when model is trained on entire dataset.

Graph(left) model overfits around the 4th epoch when ran for 10 epochs and achieves an accuracy of 94% percent

And when evaluated on test set, the model achieved an accuracy of 89.25%.






```
20/20 [==============================] - 1s 49ms/step - loss: 0.6931 - accuracy: 0.5000
Out[16]: [0.6931479573249817, 0.5]
```

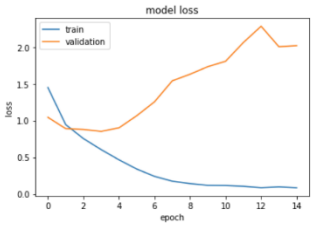
These are the results from when we use VGG16 model with trainable layers. I ran the model for 10 epochs. The model overfits after the 2nd epoch when trained on just training set and reaches a training accuracy of 58%. Whereas when trained on entire dataset, the accuracy is 50%

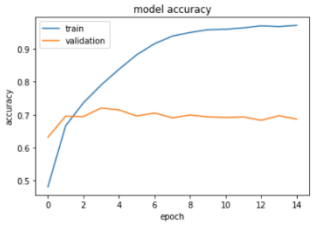
Conclusion: The model performs the best when the VGG16 model has untrainable layers, with a accuracy of 89.25%, when compared to other models. The first model performs fairly better than VGG16 model with trainable layers as the weights gets destroyed.

Drawbacks: (All models) The model overfits quickly, but this could be due to the small subset of dataset.

(First model) The network seems a little deep and I could remove some layer to reduce the complexity.

(VGG16-untrainable layer): Could perform better with other hyper-parameters.



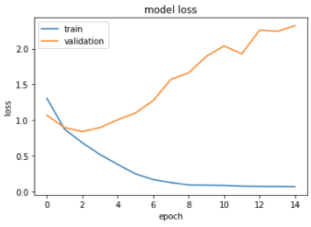


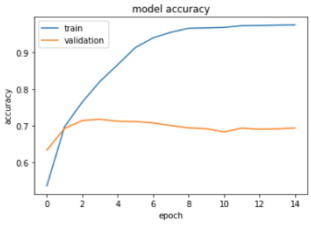
I created two models for classification. The results of model 1 are on right and of model 2 are on left. Both them perform very similarly and have no major differences.

The models overfits after 3rd epoch on right and one on left overfits between 3rd and 4th epoch.

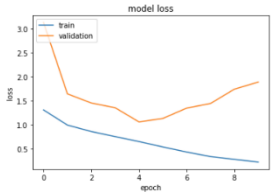
Right: accuracy – 72%

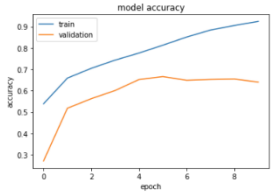
Left: accuracy – 72%



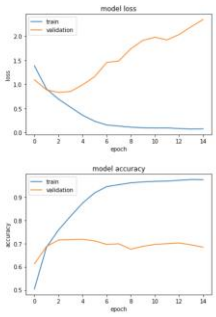


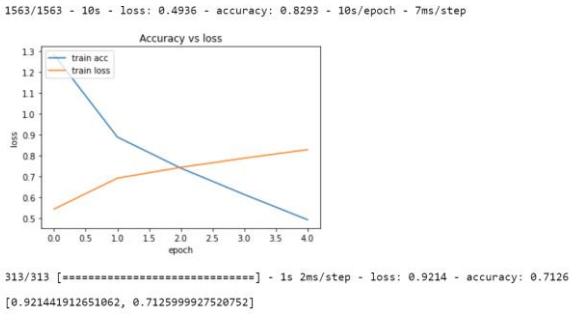
These are the results after applying batch normalization on 1st model and we see the model overfits after 4th epoch rather than 3rd epoch like before.

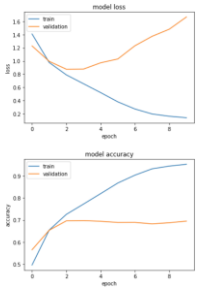


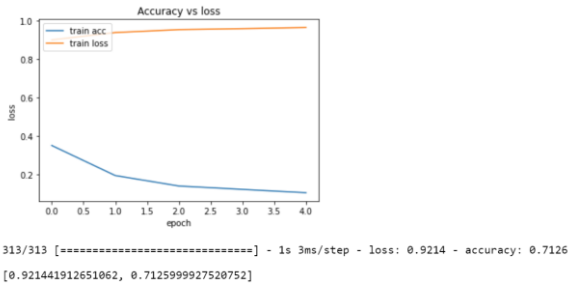


Below are the performances of model with inception block, the model reached an accuracy of 70% when trained just on training set and overfitted after 1st epoch(left). When trained on entire dataset, achieved an accuracy of 72.16% when evaluated on test data. (right)









The above results are from the model with residual block which achieved an accuracy of 75% and overfits after 4th epoch(left)
And the model on right when trained on entire dataset achieved an accuracy of 71.26% on evaluating on test set.

Drawbacks: Model overfits quickly, could try reducing the numbers of convolution filters

5. References:

Lecture notes

https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

http://marubon-ds.blogspot.com/2018/06/how-to-write-inception-module.html

https://www.analyticsvidhya.com/blog/2021/08/how-to-code-your-resnet-from-scratch-in-tensorflow/

https://becominghuman.ai/understanding-and-coding-inception-module-in-keras-eb56e9056b4b

https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/

https://machinelearningmastery.com/how-to-implement-major-architecture-innovations-for-convolutional-neural-networks/

https://www.youtube.com/watch?v=XxBrvCb6pqc

https://towardsdatascience.com/fine-tuning-pre-trained-model-vgg-16-1277268c537f

https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/