EE 622 Advanced Machine Learning
Assignment 2 : Classification of cat vs. dog images

Name: Immidisetti Rakhil
Roll No: 130102026
Department: ECE

Dependencies

Numpy, Scipy, Theano, Keras.

Running Instructions

- Command for training the model:
  THEANO_FLAGS=device=gpu,floatX=float32,lib.cnmem=0.8,mode=FAST_RUN python final_train.py
- The data is shuffled and stored in numpy arrays in '.npz' format. So no need to normalize it to float everytime one runs the code.
- For every epoch, the weights are saved as *'params.h5'* which overwrites that of the previous epoch.
- The predictions for test data are in *'predictions.txt'* file.

Report

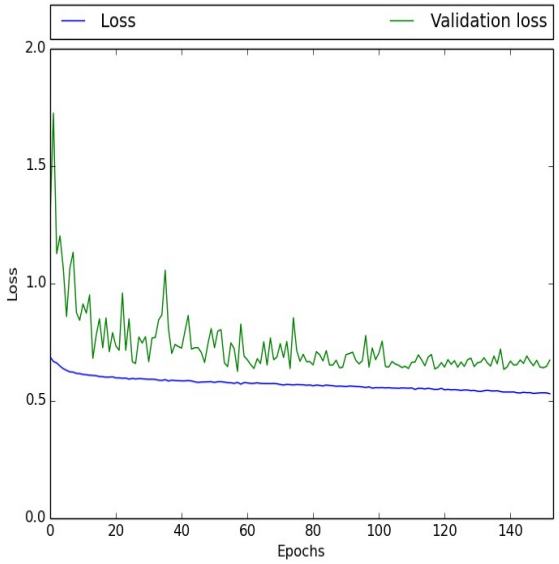
Final architecture:

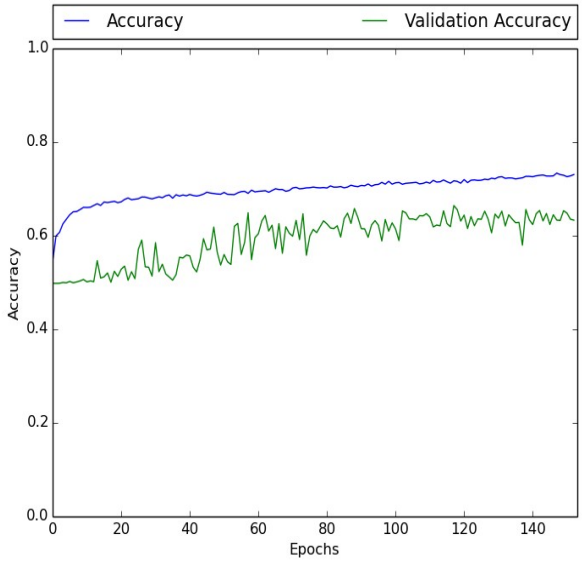
- The given data is shuffled and is split into training(0.8) and validation(0.2).
- Fits the model on data generated batch-by-batch by a Python generator. The generator is run in parallel to the model, for efficiency. This allows real-time data augmentation on images on CPU in parallel to training your model on GPU.
- The max generator queue size is 10. After every epoch the weights are saved and the metrics like loss and accuracy of that respective epoch are saved into a text file using callback.
- Used Image Data Generator in keras for data augmentation. Random horizontal flipping, shifting and rotation were done on the training images.
- Adadelta loss optimizer with a default learning rate of 0.1 and categorical cross-entropy as loss function were used.
- Activation function- LeakyRelu with alpha=0.3.

- A 8-layer architecture with 6 conv and 2 fc layers. 2 max-pooling layers were used after the first two conv layers.
- The output to the first fully connected layer is of shape 6x6x256.
- A high droput of 0.7 was used in the final fully connected layers.

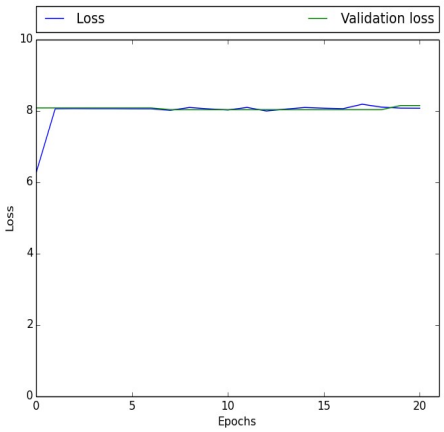
Training methods and architectures:

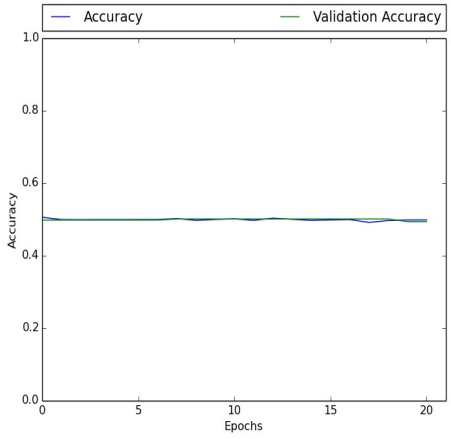
- First, I started out with a 7 layer architecture with 5 conv and 2 fc layers with SGD (Simple Gradient descent) as optimizer. And the learning rate was 0.01. I used sample-wise centre as one of the pre-processing techniques. Dropout for fc layers is 0.6. Below are the plots for accuracy and loss. (Green indicates validation. Blue indicates Training). The accuracy after 140 epochs was 75% and 62% respectively for training and validation. We can clearly see that the network is not learning properly and is also overfitting.



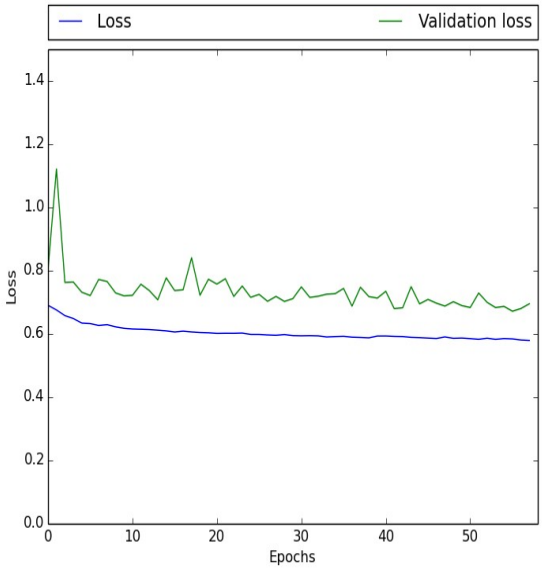


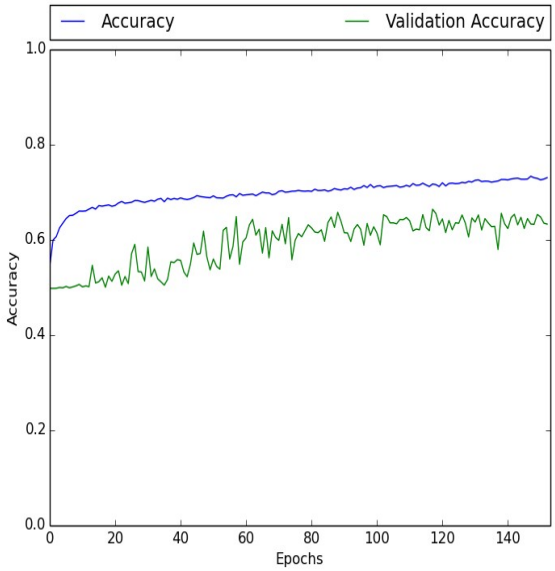
- Second. Instead of the traditional SGD optimizer I have experimented with using NADAM optimizer. The perfomance was worse and the model was not able to learn anything properly so it was stopped at 20 epochs.



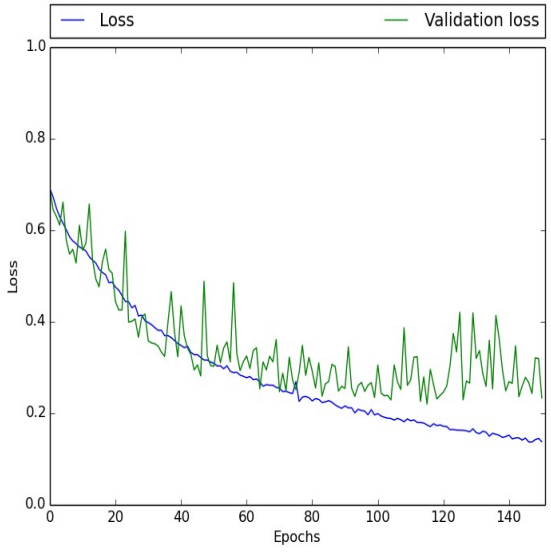


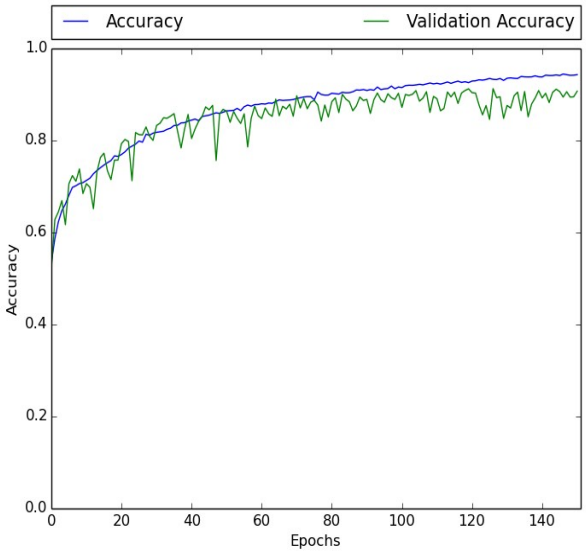
- Third. Then I went back to SGD and increased one convolutional layer and dropout to 0.7 . I decreased the learning rate manually at certain intervals with increase in epochs. There was not much difference between this and the first model.



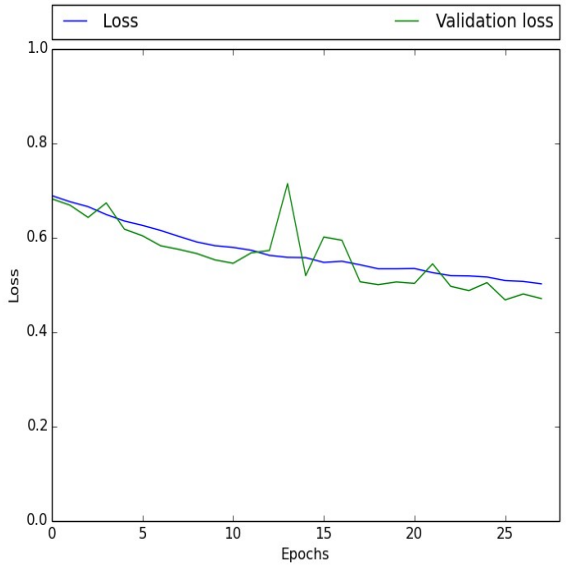


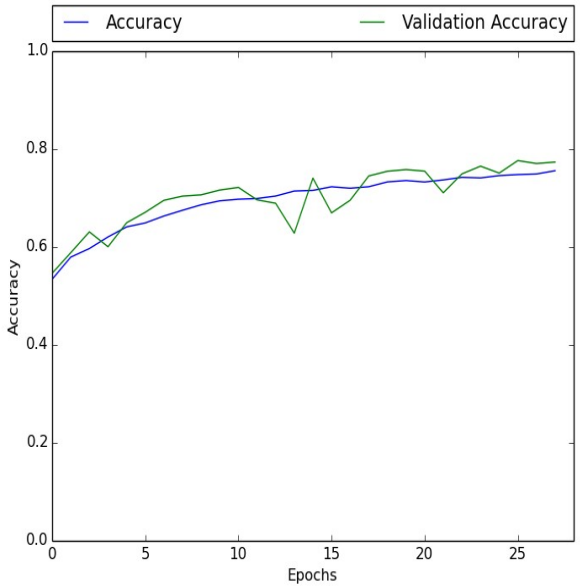
- Fourth. I felt something was fishy with the sample-wise centering of data and removed it. I used the same architecture as first model but decreased the dropout of fc layers to 0.4 and changed the optimizer to Adadelta. A vast improvement in accuracy and loss is observed.



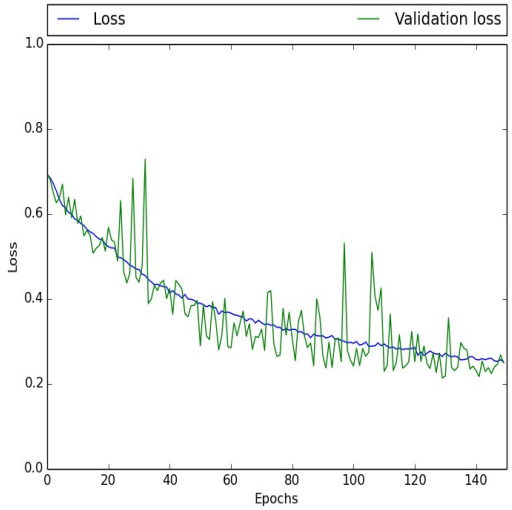


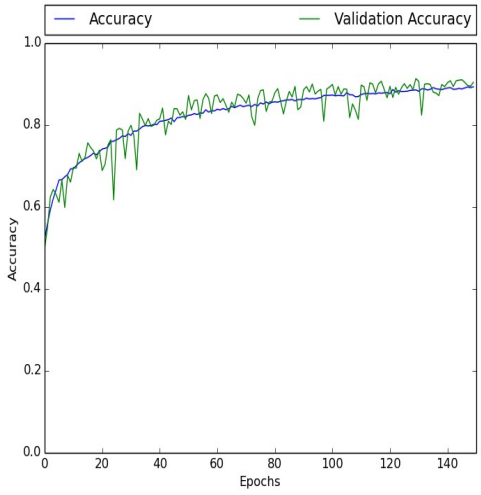
- Fifth. I confirmed by doubt on the sample-wise centering of input data by using the same model as fourth with processing the input by sample-wise centering. We see that from the below figures that using sample-wise centering of image reduces the performance of our model. Generally sample-wise centering is used to make the input data robust from brightness and contrast changes. But for this dataset, I think this plays an important feature for recognising cats and dogs.





- Sixth. In this model, I added one more convoutional layer to the fourth model. It can be observed that the validation accuracy does not deviate away from training accuracy at higher epochs when compared to 4th model due to more denser architecture.





Other Inferences:

- It can be obsereved that there are large variations in the validation accuracy due to smaller validation split.
- Adadelta was better in training the model when compared to SGD which can be observed from comparing 4^{th} and 3^{rd} models.
- Final Training Accuracy: **89.35%** , Final Validation Accuracy: **90.46%**
- Final Loss: **0.251**, Final Validation Loss: **0.249**