SAINT MARY'S UNIVERSITY SINCE 1802

One University. One World. Yours.

Master of Science in Computing and Data Analytics

Data and Text Mining

MCDA 5580

Deep Learning Assignment

Submitted To:

Chris Malloy

Submitted By:

Hemanchal Joshi (A00433394)

Haozhou Wang (A00431268)

Rishi Karki (A00432524)

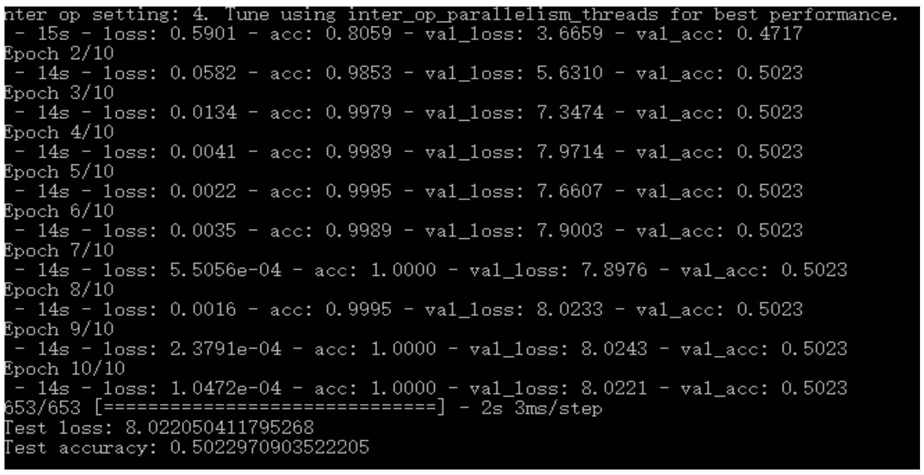
The CNN Model and it's work flow

The Classifier model we build used various computer vision libraries including openCV2, TensorFlow and many more. We build a python Object CNN which takes values from the get_images function. The get_images function loads images to the script and converts the image coordinates to arrays. We split the training and testing set from the loaded data and then move towards the CNN objects where all the values are loaded, normalized and then preparation for building the model is done.

The function build_model builds the model whereas the function where we fit the train data inside the training model and model.save() which saves the currently built model so as to avoid training datasets every now and then. So now, if the model is already existing, we can simply conduct the validation of the resultant data. We used if then statement for the user to decide weather or not they want to re-build the model. So, if the user wants to rebuild the model and they can choose the cnn model and use the index as 1 i.e. **cnn.build_model(1)** else if they want to check out the existing model, they can use index as 2 as **cnn.build_model(2)**. Similarly, we can also

Model Architecture. What is the result?

We used 4 labels in order to conduct observations for which we got an accuracy of 0.50 which was less than the observation done on 3 labels. Initially, we build the model with 3 labels giving accuracy of 0.6666 within the training folder and we tested, when dropout was added to be 0.25. With the same dropout value at 4 labels, the model gives an accuracy of 0.50



```
nter op setting: 4. Tune using inter_op_parallelism_threads for best performance.
 - 15s - loss: 0.5901 - acc: 0.8059 - val_loss: 3.6659 - val_acc: 0.4717
Epoch 2/10
 - 14s - loss: 0.0582 - acc: 0.9853 - val_loss: 5.6310 - val_acc: 0.5023
Epoch 3/10
 - 14s - loss: 0.0134 - acc: 0.9979 - val_loss: 7.3474 - val_acc: 0.5023
Epoch 4/10
 - 14s - loss: 0.0041 - acc: 0.9989 - val_loss: 7.9714 - val_acc: 0.5023
Epoch 5/10
 - 14s - loss: 0.0022 - acc: 0.9995 - val_loss: 7.6607 - val_acc: 0.5023
Epoch 6/10
 - 14s - loss: 0.0035 - acc: 0.9989 - val_loss: 7.9003 - val_acc: 0.5023
Epoch 7/10
 - 14s - loss: 5.5056e-04 - acc: 1.0000 - val_loss: 7.8976 - val_acc: 0.5023
Epoch 8/10
 - 14s - loss: 0.0016 - acc: 0.9995 - val_loss: 8.0233 - val_acc: 0.5023
Epoch 9/10
 - 14s - loss: 2.3791e-04 - acc: 1.0000 - val_loss: 8.0243 - val_acc: 0.5023
Epoch 10/10
 - 14s - loss: 1.0472e-04 - acc: 1.0000 - val_loss: 8.0221 - val_acc: 0.5023
653/653 [==============================] - 2s 3ms/step
Test loss: 8.022050411795268
Test accuracy: 0.5022970903522205
```

Fig 2: Final Output result

Then again, we tried building the model based on 4 labels (i.e. Read images from 5 folders of training dataset). The **val_acc**(validation accuracy) is 0.50. Considering it as a multi-label problem, the result was satisfying. There were 10 epochs in the network and the graph plotted by using matplotlib on the basis of each epoch is as below:

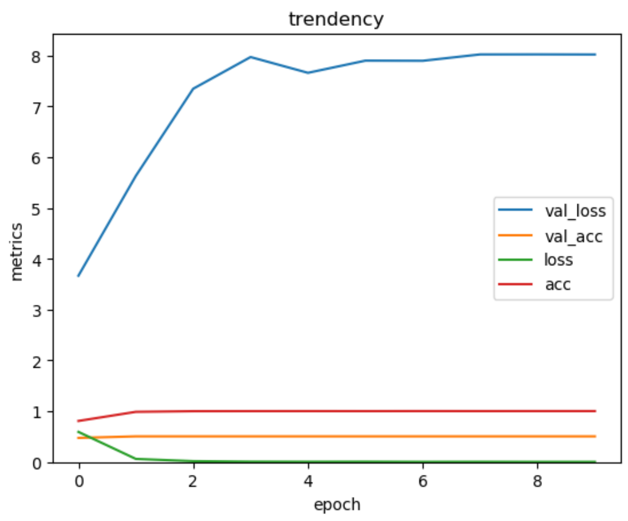


Fig 1: Tendency of different metrics relative to epoch using 4 labels

Interesting Discoveries

We found that if you get your hands dirty, you can encounter many issues unexpected, and you will learn a lot during the progress of solving it. For example :

1. We learned the concept of train and test dataset concept in the class, but in the deep learning, we still need to use validation dataset. This is a new concept and we searched a lot of materials to figure it out.
2. To reduce the waiting time, initially we only used equal number of validation folders in test, and found that the **val_acc** can easily reach up to 1.0 which is possible but makes no sense. It is mainly because of overfitting or the images being of similar size and positions; the solution is to add more dropout or training and validating on a larger data set. It can also be done by testing on a variety of validation data that differ in size and formation.
3. The difficult part was still about how to best tune the parameter. Benefited from open source, we now don't need to write the neural network code from scratch, but the

challenging part is how to improve the performance based on different situations. You need to apply the complex mathematical principle into codes and know which parameter you should modify and whether it should be increased or decreased. Overall a 4-month class can’t cover all those areas, if we still want to become a MLE(Machine Learning Engineer), much new knowledge is waiting for us.

4. Based on this answer : *https://stackoverflow.com/questions/47843265/how-can-i-get-the-a-keras-models-history-after-loading-it-from-a-file-in-python,* We can't draw the plots if we get the model from saved **h5** file. There will be no history object.