Experiments on Flower Classification with ResNet50

**Dataset Link:** <https://www.kaggle.com/alxmamaev/flowers-recognition>

**Team Number:** 2

**Team Member:**Jirom Mebrahtu

**Experiment 1**

**Data Augmentation:**

Images are resized to 150 \* 150.

One-Hot encoding is applied for the class labeling.

Images are divided by 25.

95% of the data is taken for train set.

5% of the data is taken for validation set.

**Transfer Learning:**

ResNet:

150 \* 150 input shape and 3 channels.

Pooling is set to average.

Fully connected layers are discarded.

Default weights are used.

All layers are disabled to learn.

**Fine Tuning:**

New fully connected layer with 256 neurons and relu activation.

New fully connected layer with 5 neurons and softmax activation.

**Optimizer:**

**adam as our optimizer and 'accuracy' as our accuracy measure**

I used Adam as our optimizer and accuracy as our accuracy .

Summary

I used the pretrained model of ResNet50 from imagenet. First, I got this pre-trained model, but then I have to execlude the top layer('include\_top=False'), since we want to append our own model with it. When I started to train my model. It was taken really long time even though I tried with a smaller number of epoch (which is =5). Finally, I got the accuracy=0.9862 and loss= 0.048. when I came to my transfer curving in the graph of model accuracy, it shows that my model was effective in the training process but not good on testing process which mean to test or to know for unseen flowers. Generally, in the accuracy model what I can able to understand is it was extremely overfitting, almost memorizing, while on the testing it score approximately 29%. Which mean not effective for testing a new flower. What makes me happy is I know many thing about ResNet and I got it it was interesting model very deep model from all of the others but not working as we expected, I think it is because of the optimization problem I mean when we go to deeper layer it is hard to optimize. I will reduce the overfitting on my model by using fewer parameters, fewer layers, using different optimizers , changing the batch size, changing the size of the image…etc.

Thank you so much

So the rule of thumb is use images about 256x256 for ImageNet-scale networks and about 96x96 for something smaller and easier. I have heard that in kaggle people train on 512x512 sometimes, but you will need to compromise on something. Or just buy gpu cluster