HU Extension Assignment 05 E89 Deep Learning

Handed out: 03/07/2020 Due by 11:59 PM EST on Saturday, 03/14/2020

If you are using AWS, please note that AWS charges per second of use of EC2 instance, not per hour as is usually stated. Here is the link that provides the clarification: https://aws.amazon.com/blogs/aws/new-per-second-billing-for-ec2-instances-and-ebs-volumes/. This is very important. You can perform brief experiments and then stop your instance while analyzing results or contemplating code change. If and when selecting an AWS instance, try offering 1/3 or ½ normal price by choosing a spot instance. Spot instances are not always available.
Compile your network and then train the network for two or three epochs on your local machine. Only when you are sure that your code works, transfer the code to the AWS instance of your choice. Pay attention to the amount of time your instance is running. Once you are done with a particular test or a training, stop your AWS instance. Stopped instances are not expensive. If you have a laptop or PC with working GPU card, you can safely do all problems locally and are not obliged to use AWS or any other Cloud.

Using Google's Colaboratory is a safe bet.

You are provided with 2 Jupyter notebook created by Francois Chollet. Please save the unmodified copy of each of those Jupyter notebooks so that you could compare the original results with your own.

Before you start running your Jupyter notebooks, please install Python packages: `pillow` and `h5py`.

Problem 1. Please explain where are the numbers of tunable parameter 73856 on the summary display for block2_conv1 layers on slide 31 of notes for lecture 5 is coming from. Please do not just give us a product of two numbers. Explain where those are coming from
(5%)

Problem 2. We want to verify that the code for the basic binary classification on Kaggle.com `dog_vs_cats` dataset, as provided in the Jupyter Notebook `5.2-using-convnets-with-small-dateset.ipynb`, works. We refer to the convnet with 4 `convolutional layers`, 4 `max_pooling` layers, 1 `flatten` layer and 2 `dense` layers. Modify your model by adding plain vanilla regularization. Keras has a specific syntax for adding regularization. Please examine: https://keras.io/regularizers/. Add an L2 regulizer on next-to-the last `Dense` layer. That is the one with (512,1) output tensor. As the regularization parameter `lambda`, use the value of 0.0001. Run your model for 15 epochs and compare it with the original results on the model without regularization. Present the plots of training and validation accuracies and let us know whether regularization alleviate the overfitting and to what extent. Save your model as an H5 file. Keep H5 file for future reference. Do not submit H5 file as part of your solution. Run this

code for 2 or 3 epoch on your local machine to determine the duration of your run. Only then decide whether you need to run it in an AWS instance or Google Colab. If the total run of all 15 epochs is of a bearable duration, you do not have to move the code to AWS.
(25%)

Problem 3. Demonstrate image augmentation for a breed of cats of your choice. Produce one modified image for every one of listed options: `rotation_range`, `width_shift`, `shear_range`, `zoom_range`, and `horizontal_flip`.
(15%)

Problem 4. Add image augmentation to the network in Problem 2 and report whether training and validation accuracy with the combined regularization and data augmentation differ significantly from the original results obtained on the network with augmentation only. Train your network on half the number of epochs as in the original notebook. Save your model as an H5 file. Keep H5 file for future reference. Do not submit H5 file as part of your solution.
(30%)

Problem 5. In Jupyter notebook `5.3-using-a-pretrained-convnet00.ipynb` we have training and validation accuracy for fine-tuned VGG16 network. You are given both `ipynb` and the `HTML` file. We started with conv-base tuned on large data set. One cell of the notebook attempts to unfreeze portion of the network. Experiment with that cell. Keep the network frozen and unfreeze one layer in the `block5` at a time. Report the number of tunable parameters for every layer separately. Unfreeze the entire block5. Report the number of trainable parameters. Please consult Keras documentation on how to freeze and unfreeze individual layers. For example, if your convolutional base is called `conv_base`, you can examine its layers for trainability by issuing the command:

```
for layer in conv_base.layers:
    print(layer, layer.trainable)
```

You can freeze all the layers of the `conv_base` except the last 4 layers by issuing the following commands:

```
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```

Examine whether fine-tuning layer `block5_conv3` only and freezing layers: `block5_conv1`, and `block5_conv2` will result in significant decrease in training accuracy compared to the results given in cell 24 at the end of the Challet notebook. Do not rerun the example from Chollet book. Keep it for reference. Add new cell with modified code and run only the network with trainable `block5_conv3` layer and compare. Please be warned that this training will take very long time on a machine with CPU only. Run the test with 2 epochs on your CPU machine first to make sure everything works and then run the training on a machine with GPU.
(25%)

Your main submission should be a Jupyter notebook and its HTML image. Your notebook should include all results, images and comments.
Please add your name to every document you submit.