Assignment 4

Due No Due Date **Points** 2,000 **Submitting** a website url **Available** Mar 14, 2019 at 8:30am - Apr 11, 2019 at 5:30pm 28 days

This assignment was locked Apr 11, 2019 at 5:30pm.

Link to Session 4 content is here: **SESSION 4** (https://drive.google.com/file/d/1mTqa6BSDCE_el2LG1uMefxDzq5qZJVKZ/view?usp=sharing)

ASSIGNMENT 4

Refer to this code: **https://github.com/raghakot/keras-resnet** (https://github.com/raghakot/keras-resnet)

1. This is the starting code. You are training ResNet **50**
2. Sticking to ResNet architecture as defined in the code above will give you 500 points (if you clear the cut-off), else you are free to define any architecture of your choice.
3. You need to change the dataset from Cifar10 to Tiny-ImageNet **https://www.kaggle.com/c/tiny-imagenet/data** (https://www.kaggle.com/c/tiny-imagenet/data)
    1. in the case above link is not working for you use this direct dataset link: **http://cs231n.stanford.edu/tiny-imagenet-200.zip** (http://cs231n.stanford.edu/tiny-imagenet-200.zip)
    2. or **GOOGLEDRIVELINK** (https://drive.google.com/file/d/18m4-W4z8zkxxRbOsEgrLBWHzTiRvT7z_/view?usp=sharing)
4. You are free to use any optimizer.
5. You MUST perform image augmentation (at least 5 out of 10 variants).
6. You need to get a Validation Score of +70%
7. You cannot use more than 500 Epochs

8. You cannot have more than 26M parameters
9. The assignment default weightage is 2000 pts (if you beat 70% target). For each 0.1% improvement, you get 100 point (i.e. 72% = 4000 pts).
10. You cannot use:
    1. 1x1 for an increasing number of channels
    2. dropout
    3. fully connected layers
    4. test images in training the data (this will give you -7000 points)
    5. anything else than Google Colab (using anything else would result in -7000 points)
    6. already trained model, (this will give you -7000 points)
    7. someone else's code. We'll know you have copied someone else's code if we know you followed same augmentation strategy, similar accuracy for same epochs, same variable names, etc. This would lead to straight disqualifications. There are friends, couples and colleagues in EIP 3.0, please consider disqualification strongly before sharing your code.
11. Additional points for:
    1. using more than 10 different kinds of image augmentation variations (100 additional points for additional variant) (clearly define variants using your code comments)
    2. using Separable Convolution (the second one we covered) (500 points)
    3. using cyclic learning rate (+2000 points)
    4. reduction of 1M parameters give you an additional 500 points
12. Submit your best score. Upload the link. It must have your **logs**. You are sharing colab ipynb file link. It is Ok to write the utility code in the same file, else you can refer it as an external file, both approaches are fine.
13. Refer this link to get more insight: **https://www.fast.ai/2018/08/10/fastai-diu-imagenet/ (https://www.fast.ai/2018/08/10/fastai-diu-imagenet/)**
14. Deadline is 14 days from today (5:30 **PM** sharp). Deadline is **HARD (please check LMS for final deadline for your batch)**

Hints:

1. This is a **hard** assignment and would take more than 10 days of training
2. Make sure you have added enough checkpoints
3. Make sure you are following recommendations mentioned in "Architectural Basics", else you will not be able to train this in time

Notes:

Please refer to this link to get some help on how to handle tiny-imagenet data: **https://github.com/seshuad/IMagenet (https://github.com/seshuad/IMagenet)**

You can also use this link to download the dataset: **LINK (http://cs231n.stanford.edu/tiny-imagenet-200.zip)**

Please note that that ZIP folder should be around ~240MB.

You need to figure out a way to load this dataset to Colab somehow. This **LINK (https://stackoverflow.com/questions/46986398/import-data-into-google-colaboratory) (https://stackoverflow.com/questions/46986398/import-data-into-google-colaboratory)** should be able to help you do that.

1. Compared to original dataset, tiny-imagenet images are only 64x64 pixels. Please look at the images before you start training your model.
2. ResNet50 is starting points and based on your "intuition/calculations" you can decide to use less number of ResNet layers. Personally, I would make sure I am definitely more than the receptive field requirements, and since the images are small, and from the images, I can see there is some background context, I would go beyond Receptive Field to learn the background as well.
3. Do not forget to train the network first on small images. This will help you go through a lot of epochs faster. To save time, I will create another dataset with 32x32 resolution (maybe 48x48 as well if time permits) and use that, instead of scaling images every time my network runs.
4. Look at the color composition of the images, try and see what image augmentations you need.
5. There is also a large variance in the sizes of the objects in the images. I would want to scale the images as one of my augmentation strategies. 32x32 does not mean I cannot scale a few images and send say 24x24 or 48x48
6. Please try and answer these questions
    1. "Does a large number of classes demand a large number of kernels, especially at the end?".
    2. "Do I need a longer network or wider network" (linked to the answer of 1 above)
7. I would also try an experiment first with say 10 classes, where I am handling smaller dataset, to make sure network is functional and things are not broken, instead of directly working with a very large dataset. Of course, I have to change my test/validation data to include only these 10 same classes

8. Once I have a model with good-ish accuracy (~50-60%), I would like to create a list of images where I am not performing well and more of these images.
9. My first target would be to get to 50% without focusing on the number of parameters or a lot of augmentations. **Since I am not using augmentation for this experiment, I WILL add dropout, reminding myself that this is a test, and I NEED to remove it from my final architecture.** I will think about image augmentation after I have a base network with 50%+ accuracy trained on low res images as well as on high res, then focus on the number of parameters (not to get a high score, but to get a fast trainable model with low parameters). By this time I should have integrated Batch Normalization, preferred optimizer (SGD vs Adam) faster convolution types, as well as Cyclic learning rate (if possible). Then I will remove dropout (if I still have them), and then proceed with image augmentations (IA). For these experiments, I will not wait until the model hits 70%, but see the total epochs taken to reach 60%. Models reaching 60% faster most probably will cross 70% for more number of epochs. From my list, I will select 2-3 different models/strategies and then run them for a full 500 epochs.
10. I will also make sure that I am not **increasing the number of samples** by using IA. Rather I will replace the existing images with these augmentations to make sure my per-epoch time is not long.
11. I will also have multiple colab accounts where I can run multiple instances of different experiments.

Please discuss assignment 4 questions if you have any in session 5.