17.10.2022

Department of Computer Engineering
Prof. Dr. Hazım Kemal EKENEL

BLG 506E COMPUTER VISION
ASSIGNMENT 4
Fully Connected Nets, Batch Normalization, Dropout and More

-1. This assignment is veeeery long and tough. I suggest you to start early. Fasten your seatbelts, we are taking off.

0. For this assignment and the others you will be given Stanford University CS231n course (http://cs231n.stanford.edu/) assignments. As stated in CS231n, you should be good at *Python*. Please have a look at *Python/NumPy/IPython* tutorials at http://cs231n.github.io/ .
Also we recommend you to have a *Linux OS* either locally (on your machine) or virtually (on your machine or Cloud services). Similarly, it is recommended to build a *Python* environment preferably by one of the methods below.
* *Anaconda* (https://www.anaconda.com/)
* *Miniconda* (https://docs.conda.io/en/latest/miniconda.html),
* *virtualenv* (https://virtualenv.pypa.io/en/latest/)
Check setup instructions page of CS231n (http://cs231n.github.io/setup-instructions/)

All works must be your own! Solutions from different GitHub repositories will get 0 points.
In your submission (.zip), provide all source files (.py, .ipynb etc.) that you used.
You should have comments in your code.
You are responsible to write a report that includes motivations behind your decisions (eg: hyperparameter tuning) related to experiments and your observations/thoughts about the results.
The report should be in .pdf, .txt or .doc(x) format.

1. Download assignment 2 from Stanford's CS231n:
http://cs231n.github.io/assignments2019/assignment2/
You will be following FullyConnectedNets.ipnyb, BatchNormalization.ipynb and Dropout.ipynb notebooks. You will write a better version of neural layers to construct deeper models. Batch normalization and dropout layers will be added to this modular version in the second part.

2. Start with FullyConnectedNets.ipynb. Implement affine_forward function in cs231n/layers.py .

3. Implement affine_backward function.

4. Implement relu_forward and relu_backward funcions. Answer the inline question 1.

5. For affine_relu_forward and affine_relu_backward functions in cs231n/layer_utils.py and for the loss layers **do not implement** anything. Just run the cells. They come free. Enjoy.

6. Now you will redo two layer implementation as you did in the previous assignment. Go to cs231n/classifiers/fc_net.py and complete TwoLayerNet class (__init__ and loss methods).

7. Go to cs231n/solver.py . Use it to train with different hyper-parameters and setup. You can achieve around 50% accuracy. Just do as best as you can. And plot your training curves.

8. Overfit the small dataset of 50 images with two different networks. Play with learning late and weight scale hyper-parameters to get around 100% accuracy on training set (overfitting). Answer inline question 2.

9. Go cs231n/optim.py . Implement sgd_momentum, rmsprop and adam update rules. Answer inline question 3. (I know this step contains a lot. Do your best.)

10. Now using solver, train a better model. You will come back (or maybe not) this step after implementing batch normalization and dropout layers.

11. Now open BatchNormalization.ipynb and go to cs231n/layer.py again. Implement batchnorm_forward, batchnorm_backward and batchnorm_backward_alt functions. (Yeah yeah, getting tough, I'm aware. Stay with me.)

12. Go to cs231n/classifiers/fc_net.py. Add your newly implemented batchnorm layers to your model. You can create new function that combines affine layers and batchnorm layers.

13. Try sample nets with batchnorm and without batchnorm, plot curves.

14. Run batchnorm-weight initialization interaction test, plot curves. Answer inline question 1.

15. Run batchnorm-batch size interaction test, plot curves. Answer inline question 2.

16. Surprise! A gift for you. Totally skip layer normalization. This is not covered in the course.

17. Finally(!), now go Dropout.ipynb (I promise this is easy and will be the last). Implement dropout_forward and dropout_backward functions in cs231n/layer.py. Answer inline question 1.

18. Go cs231n/classifiers/fc_net.py. Add your newly implemented dropout layers to your model. Combine affine layers, batchnorm layers and dropout layers.

19. Do dropout-regularization experiment. Answer inline question 2 and 3.

20. Bonus: Since you come this far and done batchnorm and dropout, just go back to step 10 and improve your accuracy with batchnorm and dropout layers. But as I already said "finally" a couple of lines ago, this step is totally bonus. You have my word.

21. Not a bonus: Do not go back to any step more. Have a smile. You are done. This was very hard. I know. Next homework will be easier. Celebrate it. But, you know. Epidemic is going worse, so do not go outside. Celebrate it at home.

22. Definitely not a bonus: Be happy, safe and healthy!

For any question or discussion, you can e-mail to saritas21@itü.edu.tr or you can (before informing is preferable) and come to the SiMiT Lab (office no: 4105).

For Colab users, you can insert the code below (do not forget changing the "FOLDERNAME") to the beginning of the knn.ipynp file to mount Google Drive;

```
# This mounts your Google Drive to the Colab VM.
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Enter the foldername in your Drive where you have saved the unzipped
# assignment folder, e.g. 'cs231n/assignments/assignment1/'
FOLDERNAME = None
assert FOLDERNAME is not None, "[!] Enter the foldername."

# Now that we've mounted your Drive, this ensures that
# the Python interpreter of the Colab VM can load
# python files from within it.
import sys
sys.path.append('/content/drive/My Drive/{}'.format(FOLDERNAME))

# This downloads the CIFAR-10 dataset to your Drive
# if it doesn't already exist.
%cd drive/My\ Drive/$FOLDERNAME/cs231n/datasets/
!bash get_datasets.sh
%cd /content/drive/My\ Drive/$FOLDERNAME
```