Foundations of Deep Learning **2019-2020 Spring Semester**

Assignment 1: Standard Practices in Supervised Deep Learning

Due: May 15th, 2020

Important Guidelines

1. Work should be done in pairs.
2. Solutions must be submitted in digital format (Word or LaTeX for your report, Python or Notebook files for your code).
3. You may use TensorFlow or PyTorch for your implementation. You may **not** use any high-level APIs such as Keras or TensorFlow Estimators.[1]
4. You are strongly encouraged to submit the assignment by May 1st. The long submission date is intended to allow those of you submitting a paper to NeurIPS to finish after the assignment after the deadline. It is likely that an additional assignment will be published prior to May 15th.
5. You may use `Nova` servers for the sake of this assignment. Use one of the following machines: `rack-gamir-g03`, `rack-gamir-g04`, `rack-gamir-g05`, `rack-gamir-g06` or `rack-gamir-g07`. Each submitting team should limit themselves to **use a single GPU** at any point in time. These machines are shared across all advanced machine learning courses in CS. If you prefer, you may use Google Colab or a personal laptop/PC (note that some experiments may require very long runtime on a personal laptop without GPU).

Part 1 - Setup and Baseline (20 points)

In this exercise you will experiment with different architectures to solve the image classification problem of CIFAR-10. The CIFAR-10 dataset consists of 60000 32x32 colour images from 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. In order to reduce computation time, sample at random a subset of 10% of the original data (i.e. you should have 5000 images for training and 1000 for test).[2]

1. Download and extract CIFAR-10. [3]
2. As a baseline, use `sklearn` python package to implement an SVM classifier. Report results using a linear kernel and an RBF kernel.
3. Report the accuracy obtained and plot the confusion matrices.

[1]If there is any doubt please post a question in the discussion forum on Moodle.

[2]Note that your performance need not be comparable to state of the art result on CIFAR-10 as you are using a small subset of the original data.

[3]https://www.cs.toronto.edu/ kriz/cifar.html

Part 2 - Feed Forward Neural Network (40 points)

In the following you will implement a simple fully connected neural network and experiment with the effects of different configurations on performance and runtime (i.e. time until convergence). For each item below, report the accuracy obtained and plot train/test losses as a function of the training epoch.

1. **Baseline** - Implement a fully connected neural network with 2 layers (i.e. 1 hidden layer), hidden layer width of 256, ReLU activation and cross-entropy loss. Normalize data to span the range [0,1] (i.e. minimal intensity value corresponds to 0, maximum to 1). Do not use any other form of regularization or preprocessing.
   (a) Use an SGD optimizer with momentum, use a constant learning rate. Perform grid search to obtain the best value.
   (b) Use Gaussian weight initialization with zero mean and a constant standard deviation. Perform grid search to obtain the best value.

   You may perform any number of steps/epochs you wish.
2. **Optimization** - Compare the best SGD configuration obtained to the usage of Adam optimizer. What are the effects of the different schemes on accuracy and convergence time? Explain your results.
3. **Initialization** - Use Xavier initialization.[4] How does this affect performance in terms of accuracy and convergence time?
4. **Regularization** - Experiment with weight decay and dropout. How do these affect accuracy and runtime?
5. **Preprocessing** - Perform PCA whitening prior to training. How does this affect results and convergence time?
6. **Network Width** - Experiment with the effect of varying width by training the network with one hidden layer using with a width of 2^i where $i = 6, 10, 12$. Explain your results and plot all the loss curves (on the same graph).
7. **Network Depth** - Fix the layer width to 64, and repeat a similar experiment with varying the depth of the network. Use depth values of $3, 4, 10$. Explain your results and plot all the loss curves (on the same graph).

Part 3 - Convolutional Neural Network (40 points + 5 point bonus)

1. Implement a CNN that receives as input 32×32 RGB images with the following architecture:
   - 3×3 convolution layer with 64 filters (use stride of 1).
   - ReLU activation.
   - 2×2 max-pooling layer (use SAME padding).
   - 3×3 convolution layer with 16 filters (use stride of 1).

[4]http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

- ReLU activation.
- 2×2 max-pooling layer (use SAME padding).
- Fully connected layer with dimension 784.
- Output layer with dimension 10.

Use cross-entropy loss.

2. **Optimization** - Compare the best momentum SGD configuration obtained to the usage of Adam optimizer. What are the effects of the different schemes on accuracy and convergence time? Explain your results.
3. **Initialization** - Use Xavier initialization. How does this affect performance in terms of accuracy and convergence time?
4. **Regularization** - Experiment with weight decay and dropout. How do these affect accuracy and runtime?
5. **Preprocessing** - Perform PCA and ZCA whitening prior to training. How does this affect results and convergence time? Compare the two whitening schemes and explain your results.
6. **Filter Kernel** - Change the network architecture to facilitate filters with kernel size 5×5. Report accuracy and loss and explain the changes in results.
7. **Network Width** - Experiment with increasing number of filters, the standard configuration consists of $(64, 16)$ for the first and second conv layers respectively. Run experiments with $(256, 64)$ and $(512, 256)$. Explain your results and plot the loss curves (on the same graph).
8. **Network Depth** - The convnet described above has 2 convolutional layers. Modify the network architecture to have k convolutional layers for $k = 3, 4, 5$. Explain your results and plot (on the same graph) the loss curves.
9. **(Bonus) Residual Connections** - Repeat (8) with skip connections.[5] Report your results along with a possible explanation to the changes.

(Bonus) Part 4 - Recurrent Neural Network (20 points)

In this part you are free to use any dataset of your choice. Compare two recurrent architectures, RNN and LSTM, with a single hidden layer of width 100.[6] Demonstrate the vanishing/exploding gradient problem, explaining your results along with empirical evidence supporting your claims.

[5] https://arxiv.org/pdf/1512.03385.pdf

[6] Note that the number of parameters is not the same.