Foundations of Deep Learning **2022–2023 Spring Semester**

Assignment 1: Standard Practices in Supervised Deep Learning

Due: April 23, 2023

Important Guidelines

1. Work can be done in groups of up to three students.
2. Solutions are submitted in digital format through Moodle. Please submit a zip file containing the report in PDF format and the code implementation in Python/Notebooks.
3. Please include your ID numbers in the PDF report.
4. You may implement the exercise with either TensorFlow or PyTorch. **Do not** use any high-level APIs such as Keras or TensorFlow Estimators, *i.e.* you are required to implement your own training and evaluation loops, though using off the shelf layers such as PyTorch's `torch.nn.Conv2d` and `torch.nn.Linear` is permitted.
5. To speed up runtimes you may want to use Google Colab and run the experiments on a GPU. A personal laptop/PC should suffice but will take longer, especially the experiments in part 3.

Part 1 - Setup and Baseline (20 points)

In this exercise you will experiment with different architectures for image classification. We will use the CIFAR-10 dataset, which consists of 60000 32x32 color images from 10 classes (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).[1] Throughout the rest of the assignment you need only use the subsampled version.

1. Download and extract CIFAR-10.[2] Subsample 10% of the original data as explained above and normalize the inputs to span the range [0,1] (i.e. minimal intensity value corresponds to 0, maximal to 1).
2. As a baseline, use the `sklearn` python package to implement an SVM classifier. For both the linear and RBF kernels, report the train and test accuracies obtained.

Part 2 - Feed Forward Neural Network (40 points)

In the following you will implement a simple fully connected neural network and explore the effects of different configurations on performance and runtime (*i.e.* time until convergence). For each item below, plot the train and test losses (on the same graph), as well as the train and test accuracies (on the same graph), as a function of the training epoch. In total there should be two graphs, one

[1] 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.

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

with the losses and another one with the accuracies. Additionally, report the losses and accuracies obtained by the model at the end of optimization. In each item, please incorporate the changes over the baseline you find in item (1) below. For example, after experimenting with Xavier initialization in item (3), do not use it in the following items.

1. **Baseline** - Implement a fully connected neural network with 2 layers (*i.e.* a single hidden layer), hidden layer width 256, ReLU activation, and cross-entropy loss. Do not use any form of regularization at this point. Use an SGD optimizer with momentum and a constant learning rate, and initialize the parameters randomly by sampling from a zero-mean Gaussian distribution. Set the batch size to 64 and perform a grid search over the momentum coefficient, learning rate, and initialization standard deviation. Report the hyperparameters of the best configuration found, the values over which the grid search was performed, and the results for the best configuration found.

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.[3] 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? You are allowed to use the `sklearn` implementation of PCA.

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 accuracy and loss curves of the different configurations on the same graph (one plot for train/test loss and one for train/test accuracy).

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 accuracy and loss curves of the different configurations on the same graph (one plot for train/test loss and one for train/test accuracy).

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

In the following you will implement a simple CNN and explore the effects of different configurations on performance and runtime (*i.e.* time until convergence). For each item below, plot the train and test losses (on the same graph), as well as the train and test accuracies (on the same graph), as a function of the training epoch. In total there should be two graphs, one with the losses and another one with the accuracies. Additionally, report the losses and accuracies obtained by the models at the end of optimization. In each item, please incorporate the changes over the baseline you find in item (1) below. For example, after experimenting with Xavier initialization in item (3), do not use it in the following items

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

1. **Baseline** - 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 stride of 2).
   - 3×3 convolution layer with 16 filters (use stride of 1).
   - ReLU activation.
   - 2×2 max-pooling layer (use stride of 2).
   - Fully connected layer with dimension 784.
   - Output layer with dimension 10.

   Use the cross-entropy loss, SGD optimizer with momentum and a constant learning rate, and initialize the parameters randomly by sampling from a zero-mean Gaussian distribution. Set the batch size to 64 and perform a grid search over the momentum coefficient, learning rate, and initialization standard deviation. Report the hyperparameters of the best configuration found, the values over which the grid search was performed, and the results for the best configuration found.

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 whitening prior to training. How does this affect results and convergence time? You are allowed to use the `sklearn` implementation of PCA.

6. **Network Width** - The standard configuration has filter sizes $(64, 16)$ for the first and second convolutional layers, respectively. Run experiments with filter sizes $(256, 64)$ and $(512, 256)$. Explain your results and plot all accuracy and loss curves of the different configurations on the same graph (one plot for train/test loss and one for train/test accuracy).

7. **Network Depth** - The CNN 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 all accuracy and loss curves of the different configurations on the same graph (one plot for train/test loss and one for train/test accuracy).

8. **(Bonus) Residual Connections** - Repeat (7) after adding skip connections to the network.[4] Report your results along with a possible explanation to the changes.

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

(Bonus) Part 4 - Recurrent Neural Network (10 points)

In this part you are free to use any dataset of your choice. Compare two recurrent architectures, RNN and LSTM, with hidden state size 100.[5] Demonstrate the vanishing/exploding gradient problem, explaining your results along with empirical evidence supporting your claims.

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