Programming Project I – Neural Network Architectures, Optimization and Regularization Techniques
CAP 5619, Deep & Reinforcement Learning (Spring 2024), Department of Computer Science, Florida State University

---

Points: 100
Maximum Team Size: 2
Due: 11:59pm on Tuesday, March 19th, 2024

Submission: You need to submit electronically via Canvas by uploading a) a pdf file (named "**lab1Lastnames.pdf**") as your report (including analysis and experimental results), and b) the program(s) you have created (named as "**lab1-prog-Lastnames.???**"); if there are multiple program files, please zip them as a single archive. Here replace "Lastnames" using your last names of the group members in the file names. Only one submission is required for each group.

The main purpose of this assignment is to design (deep) neural networks and use the regularization and optimization techniques we have covered in class to improve optimization and generalization performance on a dataset. Note that the focus of this programming project is to gain a deeper understanding of these techniques and having good recognition performance itself is not sufficient.

For this assignment, we use a handwritten digit dataset, consisting of a training set (http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/zip_train.txt, 7291 samples) and a test set (http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/zip_test.txt, 2007 samples), which will be used as the validation set and there is no additional test set. In other words, the generalization performance here is the performance on the 2007 samples. A description of the entire dataset is available at http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/zip_info.txt. Very briefly, each row in the files is a sample, starting with the class name (0 to 9) and then 256 floating point numbers between -1 and 1. See the additional information near the end on how to visualize the digits.

Task I – Neural Network Design

In the deep learning framework you have set up, design three different neural networks. Each one must have at last four layers; rectified activation functions must be used at least in one of the layers (among all the three networks) and sigmoid/tanh activation functions must be used in the one of the layers as well (among all the three networks). The overall connection requirements are as follows:

(1) **Fully** connected, where each input/neuron is connected to all the neurons in the next layer
(2) **Locally** connected with no weights shared in the first three layers, where each input/neuron is connected to the neurons in a local neighbor in the next layer
(3) Locally connected with weights shared in the first three layers (i.e., a **convolutional** neural network)
In your report, you need to provide explanations of your design choices and describe your neural networks clearly.

Task II - Techniques for Optimization

You need to do the required analysis and then perform experiments.

(1) Parameter initialization strategies. For each of the three networks, analyze how the parameters should be initialized. Then demonstrate three cases based on your analysis: 1) learning is very slow; 2) learning is effective (i.e., fast with accurate results); and 3) the learning is too fast (i.e., the network does not give good performance).
(2) Learning rate. Estimate a good learning rate for each of the three networks. Then demonstrate three cases based on your analysis: 1) learning is very slow; 2) learning is effective; and 3) learning is too fast.

(3) Explain how the batch size would impact the **batch normalization** for your **convolutional neural network**. Then demonstrate an effective batch size and an ineffective batch size on the **convolutional neural network** you have.
(4) Momentum. Commonly used momentum coefficient values are 0.5, 0.9, and 0.99. Using the best parameter initialization strategy, the best learning rate, and the best batch size you have found so far, experiment with the three different momentum values on the **convolutional neural network** you have and document the results. Explain the differences you have observed on the convolutional neural network you have.

Task III - Techniques for Improving Generalization

For this task, you need to do the required analysis and apply the following regularization techniques with the goal to improve the performance on the 2007 samples in zip_test.txt.

(1) Use an ensemble to improve the generalization performance. Here you need to use at least **three** neural networks to improve the performance of the individual neural networks. You need to analyze your results. You can use the neural networks from Task II.
(2) Dropout. Explain the effects of the dropout parameter (probability of keeping a neuron) on the **fully connected neural network** you have. Then demonstrate an effective case and an ineffective case on your fully connected neural network.

Extra Credit Options

(1) **Adversarial training**. Generate adversarial examples and then use them as additional training samples to improve the generalization performance. You need to improve the performance on at least one network to receive the full credit for this option.
(2) **Effective hyperparameter optimization algorithm**. You need to develop an effective algorithm to find the optimal values automatically for the hyperparameters used in Tasks II and III. At the minimum, your algorithm needs to use a coarse-to-fine strategy. (Hint: see https://www.automl.org and chapters in the AutoML book (https://www.automl.org/book/) can be helpful.)
(3) **Architecture optimization algorithm**. You need to develop an effective algorithm to search through different neural network architectures to identify an optimal one for the given task. (Hint: see https://www.automl.org and chapters in the AutoML book (https://www.automl.org/book/) can be helpful.)

Grading

- **Report and analysis** – 30 points
  - Note that the focus is on understanding and you have to provide meaningful/insightful analysis for what you expect from your experiments before doing them and explain what you have observed in your experiments for each of the cases for each of the networks as explained in the tasks.
- **Correct implementation and experimental results** – 70 points
  - **Task I – 10 points**
    - **Correct neural network architectures** – 10 points
  - **Task II – 40 points**
    - **Parameter initialization strategies –** 15 points
    - **Learning rate** – 15 points
    - **Batch normalization** – 5 points
    - **Momentum** – 5 points
  - **Task III – 20 points**
    - **Ensemble –** 10 points
    - **Dropout –** 10 points

- **Adversarial training** – 10 points
- **Hyperparameter optimization** – 10 points
- **Architecture optimization** – 10 points

Additional information:

Regularization techniques are covered in Chapter 7 of the Deep Learning textbook and optimization techniques in Chapter 8. For adversarial training, you may find the paper “Explaining and Harnessing Adversarial Examples,” (available from https://arxiv.org/pdf/1412.6572.pdf) and " Towards Deep Learning Models Resistant to Adversarial Attacks" (available from https://arxiv.org/abs/1706.06083) helpful. Various hyperparameter optimization strategies are discussed in 11.4 of the Deep Learning textbook. For architecture optimization, you may find the relevant references in “Neural Architecture Search: A Survey” (available from https://arxiv.org/abs/1808.05377) and "A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions" (ACM Computing Surveys Vol. 54, No. 4) helpful.

The following shows the hand-written digits in the training set so that you know what the digits look like. If you like to show a digit, you need to scale the values from [-1 1] to [0 255] and reorganize the pixels from a 256-element vector to a 16×16 image; you may need to rotate the image.