CSCI 4146/6409 - Process of Data Science (Summer 2023)

Assignment 4: Neural Network and Deep Learning

The objective of this assignment is to develop an understanding and hands-on experience with various deep learning techniques, including fully-connected networks, batch normalization, dropout, and convolutional networks using PyTorch. You will apply these techniques on the MNIST dataset, analyze results, and draw meaningful conclusions.

Due Tuesday July 04, by 11:59 pm - see submission details below.

Dataset: MNIST- http://yann.lecun.com/exdb/mnist/ || https://www.kaggle.com/datasets/hojjatk/mnist-dataset

Q1: Fully-connected Neural Network [1]

In the 'fc_network.ipynb', a basic fully connected network implementation is provided. Using this as your starting point, apply and compare at least three popular update rules, such as Stochastic Gradient Descent (SGD), Momentum SGD, RMSprop, and Adam. Evaluate the performance differences introduced by selected optimization methods and discuss your findings.
You can refer to the PyTorch Optimizer Documentation for additional information on these methods.

Q2: Batch Normalization [1]

Modify the fully-connected network from Q1 to include batch normalization. Compare the performance of your network with and without batch normalization.
PyTorch BatchNorm Documentation for details on how to implement batch normalization in PyTorch.

Q3: Dropout [1]

Modify the network from Q1 or Q2 to include dropout. Implement dropout and explore its effects on your model. Compare the performance of your network with and without dropout.
The PyTorch Dropout Documentation can provide guidance on how to add dropout to your model.

Q4: Convolutional Networks [1]

Implement a convolutional neural network (CNN) using PyTorch and MNIST dataset. Your CNN should include at least one convolutional layer, one pooling layer, and one fully-connected layer. For additional help on building CNNs, refer to the PyTorch CNN Tutorial.

Q5: Model Training [1]
Finally, with PyTorch, train a model on the MNIST dataset. You may use any architecture of your choice but must justify your choice. Also, discuss what you did to optimize your model's performance.
The PyTorch Training a Classifier Tutorial may provide a good reference.

For each question, you should include the following deliverables:

- A detailed description of your model architecture and the rationale behind your design decisions.
- A graph of loss vs. epochs for your model.
- A comparison of performance for different variations of your model (e.g., with and without batch normalization/dropout, using different update rules, etc.).
- A discussion of your results. What insights can you draw from the performance of your model?

Performance Expectations

Before you start implementing, here are some guidelines on what you should expect your models to achieve:

- Fully connected networks can reach an accuracy of about 97-98% on the MNIST dataset.
- Introducing batch normalization should speed up training as well as a slight improvement in accuracy.
- Implementing dropout can help to prevent overfitting. The accuracy might not significantly improve. But the model should generalize better to unseen data.
- With a well-implemented convolutional network, you should aim for an accuracy of around 99% on the MNIST dataset.
- The model's performance will greatly depend on the chosen architecture and training procedure. As a benchmark, note that state-of-the-art models can achieve about 99.7% accuracy on MNIST.

Submission Details

Submissions should be made through Brightspace, adhering to the due date and time specified under the Assignments section. To prepare your assignment solution, make use of the provided assignment template notebook on Brightspace. The detailed requirements for writing and coding can be found within the evaluation rubric document available on the platform. Keep in mind that questions will be graded individually using letter grades, with their respective weights indicated in parentheses.

You may complete the assignment individually or with another individual. In the case of a pair submission, only one student should submit the assignment on Brightspace. All analyses and discussions should be based on your own insights and understanding. While collaboration and discussion among classmates is encouraged, all written, and code work must be your own. Any sources consulted, including websites, textbooks, papers, etc., should be properly cited. Any evidence of plagiarism or academic dishonesty will result in a zero for the assignment.

Your submission should consist of a single Jupyter notebook as well as a generated PDF that contains the compiled results generated by the notebook. This PDF should include both the code and the results as part of the final printout. Name your files as follows:

- A4-<your_name1>-<your_name2>.ipynb and
- A4-<your_name1>-<your_name2>.pdf.

Failing to submit both files will result in a zero mark for both students.

References

Byerly, A., Kalganova, T., & Dear, I. (2021). No routing needed between capsules. Neurocomputing, 463, 545-553.

Kowsari, K., Heidarysafa, M., Brown, D. E., Meimandi, K. J., & Barnes, L. E. (2018, April). Rmdl: Random multimodel deep learning for classification. In Proceedings of the 2nd international conference on information system and data mining (pp. 19-28).

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013, May). Regularization of neural networks using dropconnect. In International conference on machine learning (pp. 1058-1066). PMLR.

Ciregan, D., Meier, U., & Schmidhuber, J. (2012, June). Multi-column deep neural networks for image classification. In 2012 IEEE conference on computer vision and pattern recognition (pp. 3642-3649). IEEE.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. Advances in neural information processing systems, 30.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.