CMU 10-715: Homework 7

Implementation of Convolutional Neural Networks

DUE: Nov. 7, 2020, 11:59 PM.

Instructions:

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books, again after you have thought about the problems on your own. Please don't search for answers on the web, previous years' homeworks, etc. (please ask the TAs if you are not sure if you can use a particular reference). There are two requirements: first, cite your collaborators fully and completely (e.g., "Alice explained to me what is asked in Question 4.3"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Submitting your work:** Assignments should be submitted as PDFs using Gradescope unless explicitly stated otherwise. Each derivation/proof should be completed on a separate page. Submissions can be handwritten, but should be labeled and clearly legible. Else, submission can be written in LaTeX.
- **Late days:** For each homework you get three late days to be used only when anything urgent comes up. No points will be deducted for using these late days. We will consider an honor system where we will rely on you to use the late days appropriately.
- **Training NNs**: It is feasible to train the network in this homework on a CPU. However, to speed up the training process, you may use Google colab[1], a free GPU service provided by Google. To enable GPU, under the Runtime tab, select Change runtime type and choose GPU. To mount a colab notebook to your local file systems or google drive, refer to the link here[2].

[1]`https://colab.research.google.com/notebooks/intro.ipynb`
[2]`https://colab.research.google.com/notebooks/io.ipynb`

1 Implementation of Convolutional Neural Networks [100 points]

In this homework you are going to code a popular CNN called LeNet5[3] with **PyTorch**. You can use a different deep learning framework if you want to.

1.1 LeNet5

As we have discussed in class, Convolutional Neural Networks (CNNs) are a powerful type of networks which are good at classifying images such as handwritten digits. LeNet5 is a classic CNN model proposed back in 1998. We will implement LeNet5 to classify MINIST data with slight modifications:

1. The input image is 28×28 which is different from that of the original paper.

2. We will use ReLU instead of sigmoid activations.

3. We will not include the Gaussian activation in the final layer for the similar reason as in HW6.

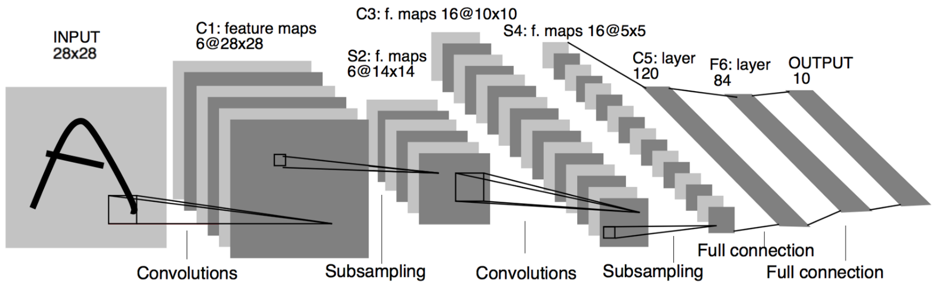

Figure 1: Architecture of LeNet 5. Adapted from the original paper of LeCun et al.

1.2 Instructions

- Read **PyTorch**'s documentation on convolution and pooling.

- You will need to complete the missing code (marked with #TODO) in the script provided, ***lenet.py***. Most of the code will be the same as that in HW6 including #TODOs of the same functions. Feel free to reuse

[3]Y. Lecun, L. Bottou, Y. Bengio and P. Haffner,"Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791. Avaliable at `http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf`

whatever you wrote of HW6, or the code in the solution of HW6. Also feel free to modify/add any functions when necessary. **Please append your code at the end of your pdf submission.**

- Complete the function ***_initialize_network*** method on *lenet.py* that defines the architecture of the neural network. Hint: use nn.Conv2d(), nn.AvgPool2d(), nn.MaxPool2d(), Reshape() and nn.Flatten() modules.
- Complete the function ***get_intermediate*** method on *lenet.py* that returns the intermediate outputs requried in the questions below. This will be analogous to the ***layer_features*** function of ***_DropoutClassifier*** in HW6.
- Train your classifiers, we provided you with beginning code in the ***main*** function. (Hint: trajectories will be available in the ***trajectories*** attribute of the classifier, save the weights of your classifiers so that your progress is not lost.)
- You should be able to reach $> 98\%$ test accuracy with max pooling after 30000 steps. Keep in mind that you should be able to almost perfectly fit the training set (close to 100% accuracy).

1.3 Expected Results

The architecture of the version of LeNet5 we are going to implement is:

Input image $\rightarrow$ 2D convolution with kernel size 5, stride 1 and padding 2 (all w.r.t. each dim) $\rightarrow$ ReLU $\rightarrow$ Pooling with kernel size 2, stride 2 and no padding $\rightarrow$ 2D convolution with kernel size 5, stride 1 and no padding $\rightarrow$ ReLU $\rightarrow$ Pooling with kernel size 2, stride 2 and no padding $\rightarrow$ Fully connected linear layer $\rightarrow$ ReLU $\rightarrow$ Fully connected linear layer $\rightarrow$ ReLU $\rightarrow$ Fully connected linear layer $\rightarrow$ Output

Train three CNN models: $\boldsymbol{M_{avg}}$: LeNet5 with average pooling , $\boldsymbol{M_{max}}$: LeNet5 with max pooling, $\boldsymbol{M_{no}}$: LeNet5 with no pooling. Note that even if you should use the same convolution settings as the models with poolings, you may have to change the in_features of the first fully connected layer when you define the architecture for $\boldsymbol{M_{no}}$ because the shape of the tensors will be changed. You may also have to insert operations to reshape the input tensor from 784 to 28×28 (use ***Reshape()*** given in the code) and flatten the tensors before the fully connected linear layer (use ***nn.Flatten()***).

a (5 points) Train your classifiers (using GPU) for **30000** steps. Alternatively, you may run your algorithm for **5000** steps (using CPU), in which case you will be awarded full points for the next results, but won't receive this 5 points. For your reference: it takes around 10 mins to train $\boldsymbol{M_{max}}$ for 30000 steps on Colab with a GPU.

b (25 points) Show the final training and testing accuracy and training time for the three models. Specify the number of steps you used for training (5000 or 30000). You can time the entire training process including steps for computing test accuracy. Discuss the reason for the difference of training time between $\boldsymbol{M_{no}}$ and the two models with pooling.

c (20 points) For each model, display the train and test loss vs the gradient steps (only for the given display steps specified in the hyper-parameters) in the same plot. You will show three subplots, each corresponding to a model.

d (20 points) For each model, display the train and test accuracy vs the gradient steps (only for the given display steps specified in the hyper-parameters) in the same plot. You will show three subplots, each corresponding to a model.

e (30 points) (For $\boldsymbol{M_{avg}}$ only) Pick a sample from the MINIST dataset, plot 1) The original image. 2) Output of the first layer (after the first convolutional layer and before ReLU). 3) Output of the second layer (after the first pooling layer and before the second convolutional layer). In particular, arrange the three plots as follows: 1)one 28×28 image. 2) six 28×28 images. 3) six 14×14 images. Compare 2) with 1) and 3) with 2). Briefly explain your observations.