EE416 Final Project

Il Yong Chun
Department of Electrical and Computer Engineering, the University of Hawai'i, Mānoa

Novermber 2, 2021

Instructions

- The submission is **due 11:55 PM on December 7, 2021** via Laulima.
- A **single** pdf file as your write-up, including your plots and answers to all the questions and key choices you made.
  The write-up must be an electronic version. **No handwriting, including plotting questions.** LaTeXis recommended but not mandatory. You might like to combine several files to make a submission. Here is an example online link for combining multiple PDF files: `https://combinepdf.com/`.
- A zip file including all your codes, and files specified in questions with **Submit**, all under the same directory. You can submit Python code in either .py or .ipynb format.

Python Environment

We are using Python 3.7 for the final project. You can find references for the Python standard library here. To make your life easier, I recommend installing Anaconda 5.2 for Python 3.7.x. This is a Python package manager that includes most of the modules you need for this project.

You are expected to use the following packages extensively:

- Numpy
- OpenCV
- PyTorch
- Matplotlib

You are required to use PyTorch for building and training neural networks. Install PyTorch as torch and torchvision for datasets. You may also need matplotlib.pyplot to visualize results and tqdm to display a progress bar.

As a deep learning library, PyTorch performs backpropagation automatically for you and trains your network faster. For this homework we have also provided some starter code with comments.

Transfer learning with designed fully-connected network classifier for COVID-19 detection

You will implement **transfer learning** in **PyTorch** to detect COVID-19 infection from a 2D chest CT image. Transfer learning stores knowledge gained while solving one problem and apply to a different but related problem. In particular, you will design your own fully-connected network (FCN) classifier and train/test it using a COVID-19 lung chest CT dataset, whereas you will use the convolutional neural network (CNN) feature extractor of pre-trained ResNet-50 [1] with the conventional large dataset.[1]

[1] In general, one first determines how many first layers of the pre-trained ResNet-50 to "freeze" before designing a FCN classifier. Shallow and deeper layers are responsible for low-level and high-level vision tasks, respectively; in general, one only refine the last few layers of CNN feature extractors. For simplicity we omit this task in the project.

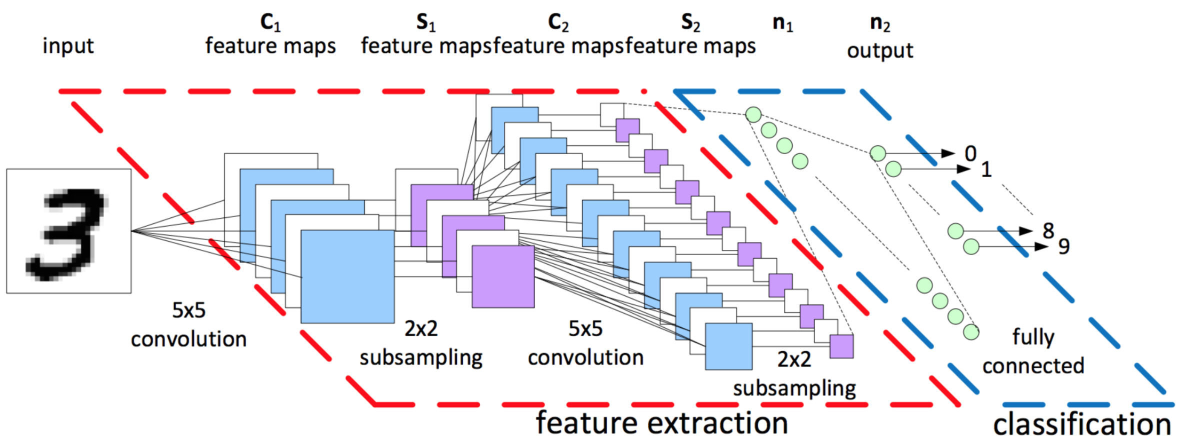


Figure 1: An example for CNN+FCN image classifier. (Image is from this link.)

Back-propagation is automatically inferred by PyTorch (autograd), so you only need to write code for the forward pass.

You will use the provided COVID-19 chest CT dataset consisting of $\approx 4,400$ cropped 2D lung CT images with three classes, COVID-19 pneumonia, other pneumonia, and healthy condition. Noting that data preparation is an important part of all data science research, you will first write a code that constructs the dataset structure as follows:

- ./Dataset/Train/image/
- ./Dataset/Train/label/
- ./Dataset/Test/image/
- ./Dataset/Test/label/

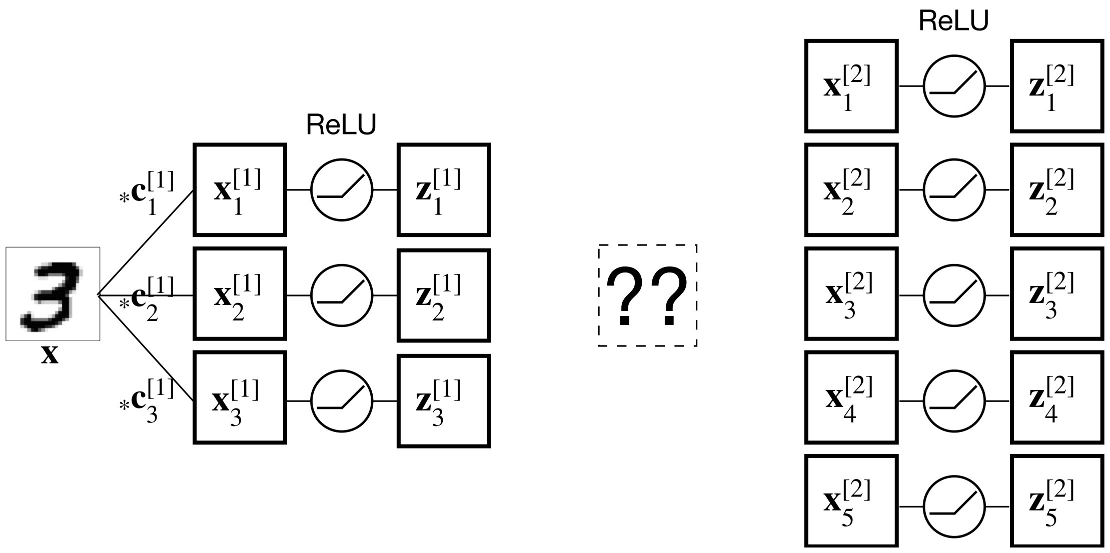
In particular, you will use five-fold cross validation, splitting the dataset to training and testing sets containing 80% and 20% images, respectively. You may apply *data augmentation*, e.g., rotating images by 90°, 180°, and 270°, and increase the number of images from $4,400$ to $17,600$. The augmented training and testing data should maintain the sample size ratio 4:1.

You will design your own FCN classifier that achieves the "best" classification accuracy. I have provided some start code in eg.py where you need to modify and experiment with the following:

- The architecture of the FCN classifier (define architectures and implement forward pass).
- The training loss function. (*Default:* CrossEntropyLoss; see the Loss functions shortcut in this link)
- The optimizer (stochastic gradient descent, RMSProp, Adam, etc.) and its parameters, learning rate and weight_decay. (Note that weight_decay is the ℓ_2 regularization strength.)
- Training parameters (batch size, number of epochs, learning rate decay scheme, etc).

Complete the following:

(a) Suppose that you construct conventional two-layer CNNs with 5×5 kernels and 3 features in the first convolutional layer and 5 features in the second convolution layer. The first convolution layer is parameterized with the first layer convolutional kernels $\{\mathbf{c}_k^{[1]} \in \mathbb{R}^{25} : k = 1, \ldots, 3\}$; its operators can be specified as follows:



Fill the question-box above with the second layer convolution kernels $\{\mathbf{c}_?^{[2]}\}$, convolution operator $*$, and summation $\sum_?$. **Determine** the total number of parameters of the CNN architecture. Justify your answer. (20 pts) (Note that this architecture will not apply to answering the questions below.)

(b) **Submit** a program which trains with your best combination of model architecture, training loss function, optimizer and training parameters, and evaluates on the test set to report an accuracy at the end. (20 pts)

(c) **Report** the detailed architecture of your best model. Include information on hyperparameters chosen for training and a plot showing both training and test loss across iterations. (30 pts)
Report the accuracy of your best model on the test set. We expect you to achieve over 85%. (30 pts)

Hints for designing your FCN classifier: Read the PyTorch documentation for torch.nn and pick layers for your network. Some common choices are

- nn.Linear
- nn.ReLU, which provides non-linearity between layers
- nn.Dropout, which helps reduce overfitting.

You will get partial credits for any accuracy over 75%, so do not worry too much and spend your time wisely.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Las Vegas, NV, June 2016.