Exercise Session
CIVIL-459
Spring 2020
Lab 3

EPFL
School of Civil Engineering
Prof. Alexandre Alahi
TA: George Adaimi

Problem Set 3 - Convolutional Neural Network

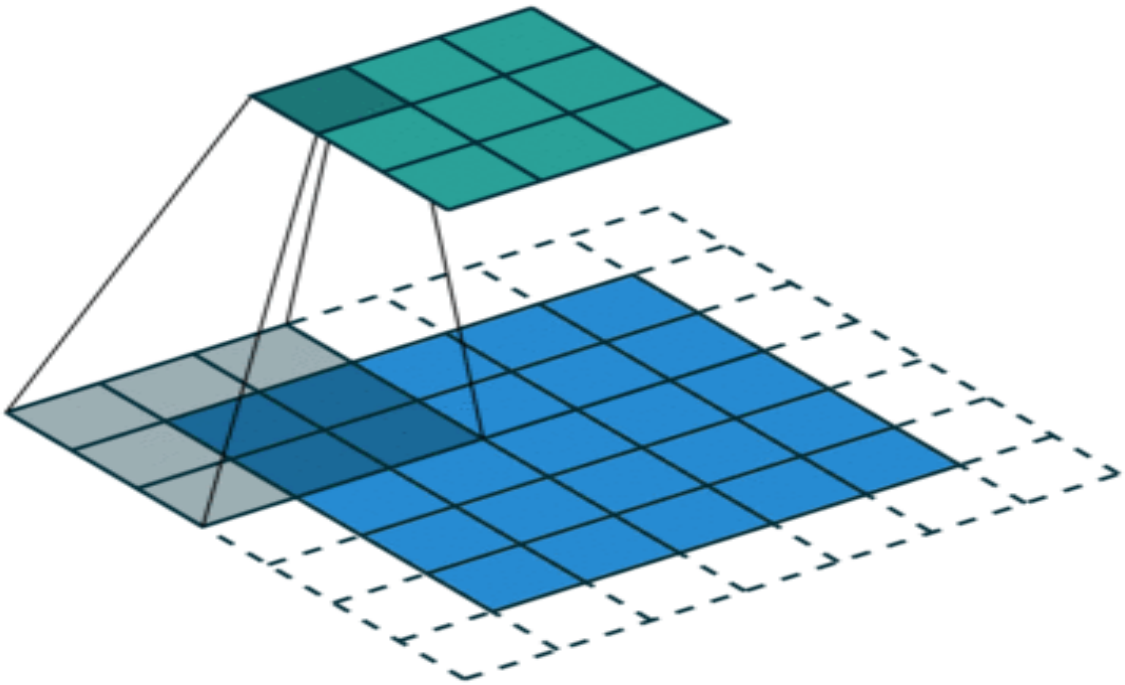

Figure 1: Convolution

This time we'll check out how to build a convolutional network to classify CIFAR10 images. By using weight sharing - multiple units with the same weights - convolutional layers are able to learn repeated patterns in your data. For example, a unit could learn the pattern for an eye, or a face, or lower level features like edges.

Traditionally, convolutional layers are followed by max-pooling, where values in the convolutional layer are aggregated into a smaller layer shown in Figure 2. These types of networks become really useful when you stack a bunch of layers, you make the network deeper. Typically you'll use a convolutional layer to change the depth, ReLU activation, then a max-pooling layer to reduce the height and width.

To finish off the network, you need to flatten the final convolutional layer into a normal fully-connected (dense) layer, then add more fully-connected layers as a classifier. The convolutional layers act as feature detectors that the classifier uses as inputs. Convolutional networks have been massively effective in image classification, object recognition, and even in natural language

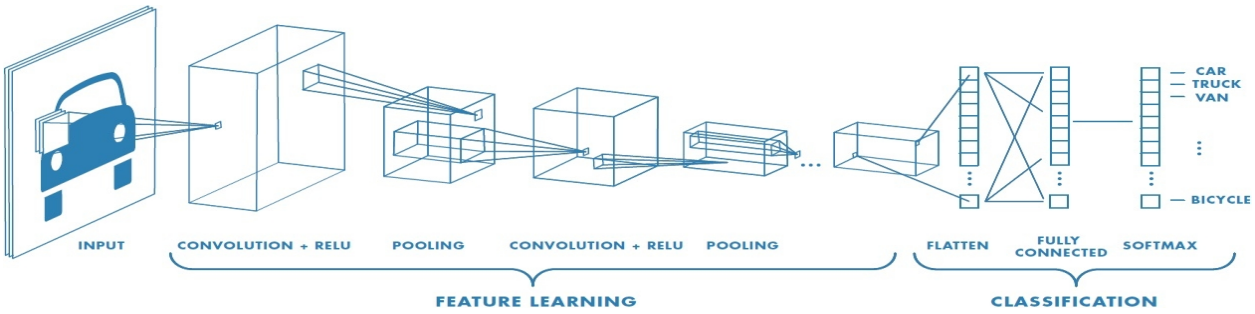

Figure 2: Convolutional Neural Network

processing applications like speech generation.

Building this new network is pretty much the same as before, but we use nn.Conv2d for the convolutional layers and nn.MaxPool2d for the max-pooling layer.

Note: Do not forget to submit, with your solution, the trained model of the notebook as well as filling the "run.py" file. This file will be used to run your model on our dataset and check your accuracy. Check this discussion to save your model.

What to do

Implement the convolutional neural network using pytorch in **CNN Exercise.ipynb**

1. Define the different layers of the neural network. The neural should have at least 2 layers but can be as large you want.
2. Set up the forward pass by connecting the different layers of the neural network. **Note:** Do not forget to use activation layers.
3. Choose a loss function and an optimizer that will be used.You can tune the parameters of the optimizer
4. Implement the training process and save your best model

   Finally, for us to test your code and get an accuracy, you should implement inside the **run.py**: **predict_usingCNN()**. You should load the saved model, input to it the data and get the prediction labels. We will be using our own test data.

Helpful References

- Pytorch Documentation: https://pytorch.org/docs/stable/
- Pytorch Tutorial: Official link, Collection
- Pytorch Convolution Layers: http://pytorch.org/docs/master/nn.html#convolution-layers