CERI - 8104 – HW #4

ML Seminar – Convolutional Neutral Networks

Reading: "Python Machine Learning", Raschka, Chapter 15
Submit a short report with figures that illustrate your results! Explain your observations and respond to all questions below!

Part V: Comparing MLP and CNN performance and speed

You will compare training behavior and performance of the MLP and CNN.

1. We will start by comparing the shape and total number of weights and biases in convolutional and fully connected layers. Remember that for fully connected layers the number of weights is simply the product of neurons in current and previous layers. The number of biases is equal to the number of neurons in the current layer. For convolutional layers the number of weights can be computed using:
   $n_w = K^2 * n_c * n_{filt}$ and $n_b^{[l]} = n_{filt}^{[l-1]}$

Compute the number of parameters (weights+biases) for each layer in the following network, with filter heigh = 5, batch size = 16 and original image size 28x28:

Layer	Shape	#Parameters
conv_1 (Conv2D)	(16, 28, 28, 32)	___
pool_1 (MaxPooling2D)	(16, 14, 14, 32)	___
conv_2 (Conv2D)	(16, 14, 14, 64)	___
pool_2 (MaxPooling2D)	(16, 7, 7, 64)	___
flatten (Flatten)	(16, 3136)	___
fc_1 (Dense)	(16, 1024)	___
dropout (Dropout)	(16, 1024)	___
fc_2 (Dense)	(16, 10)	___

2. Write a script to perform the image classification tasks using the neural network model structure and the predefined layers in Keras!
   a. You can copy/paste data import and preprocessing step from your MLP solution.
   b. Construct a CNN with the following architecture:

  i. 9 Layers total: 1 Input, 1 Conv2D, 1 MaxPool, 1 Conv2D, 1 MaxPool, 1 Conv2D, 1 MaxPool, 1 FC (n-hidden = 20), 1 FC (n-hidden=number of unique class labels). Note that the last fully-connected layer should be a softmax layer which will return the class label probabilities for each unique target and for all input data samples (shape=(nSample,nLabel)).
  ii. Use pool_size=(2,2) for the MaxPool layers
  iii. Increase the number of filters by a factor of two between each subsequent convolutional layer, starting with 8 filters. Use a kernel size of 3x3, stride=1 and padding = 'same'.
  iv. Be careful with the shape of the input data and change in shape when transitioning from Conv2D to fully connected layers.

3. Create plots of the loss and learning curves as a function of epochs over the training and validation data sets.

4. Plot a few examples of correct and false predictions. The keras model object has a convenient function: .predict() which will allow you to compare observed and predicted labels in the test data set. Note that .predict() returns the relative probabilities for each class label from your *softmax* activation function within the last layer of your NN. You can use these probabilities to create predicted class labels.

5. Bonus: Convert the numpy arrays to an iterable tensorflow dataset object and construct the computational graph including shuffle(), batch() and repeat().

Bonus: Compare your result to a simple base model

One common way to classify seismic event types e.g. earthquakes vs. quarry blasts is to look at amplitude ratios across different frequency bands (or by comparing P and S wave amplitude rations).

1. Construct a simple classifier based on the average amplitude ratios:
    a. Compute average amplitude rations between 0.1 - 2.5 Hz and 2.5 – 5 Hz.
    b. Plot the class labels as a function of amplitude ratios.
    c. Formulate a classifier based on your observation.
    d. How well would the simple base model perform compare to your Neural Nets?