2019SP_MSDS_458-DL_SEC56 Week5- Assignment2

Prabhat Thakur

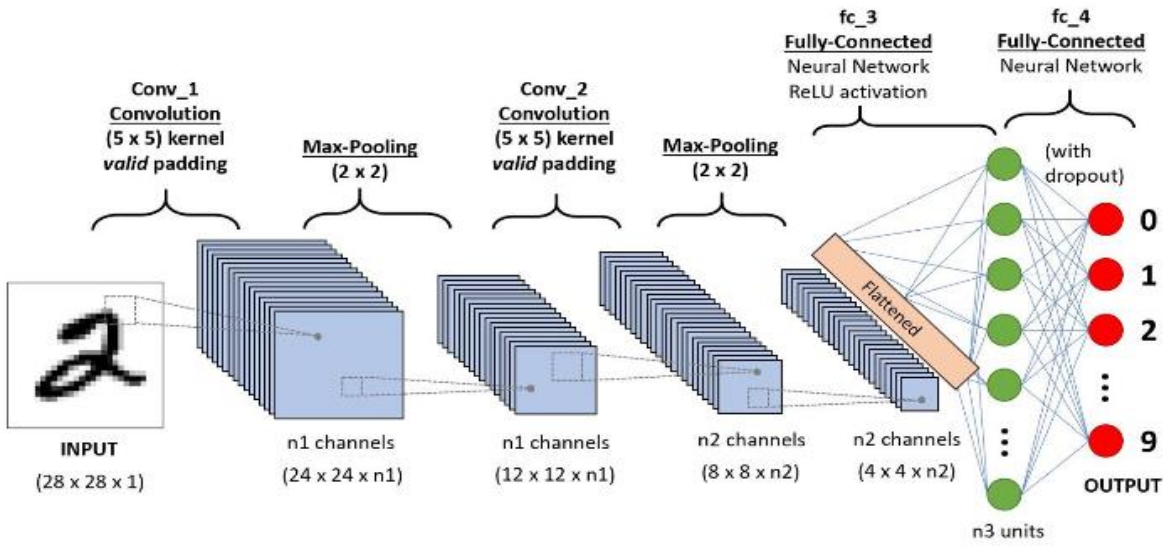
Deep Learning - Convolutional Neural Network (CNN) Study

Abstract: The goal of this assignment is to transition from simple (single hidden layer) to deep (multiple hidden layers) networks and focus on different convolutional neural network (CNNs) structures to understand the impact of various (convolution, pooling and dropout) layers on input data feature extraction, interpretation and model performance. The challenge of this assignment is less about training a successful neural network for deployment but to understand and interpret the hidden node activations as features.

Introduction: The data used in the analysis is the MNIST database of handwritten digits. The database consists of single digits displayed in grayscale. This data is quite different from the alphabet data used in earlier assignment. It has ten classes of output with sixty thousand training observations and ten thousand test observations. Also there are multiple representations of each digit. At a high level, this similar to the perturbed alphabet data but greater importance is given to the model's ability to generalize to new data. Before we look into different models let's see an example of a single digit from the training dataset (Appendix –Figure 1). All handwritten digit images are grayscale and of 28 by 28 pixels.

Literature: A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.



A CNN sequence to classify handwritten digits

Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting

dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. In this research max pooling is used.

The high-level features as represented by the output of the convolutional layer is then passed to Fully-Connected layer for learning non-linear combinations. The Fully-Connected layer is learning a possibly non-linear function in that space. Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, they are than flatten into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

Methods: As part of this exercise performance of 5 different CNN models are analyzed. The first model is made up of one 3x3 (kernel) convolutional layers with 32 feature maps respectively. After this layer a 2x2 max-pooling layer is added, followed by fattened layer, a fully connected hidden layer with 64 nodes. The last fully connected layer generates softmax class outputs. The second model has a 0.25 probability dropout layer between pooling and flatten layer. In third model we used 128 nodes in fully connected hidden layer instead 64 nodes. The fourth model is made up of one 3x3 (kernel) convolutional layers with 32 feature maps. After this layer a 2x2 max-pooling layer is added, followed by 2nd convolutional layer of 3x3 kernel size with 64 feature maps, 2x2 max-pooling layer, fattened layer, a fully connected hidden layer

with 128 nodes. In the 5th and final model we added 0.25 probability dropout layer after 1st convolutional layer and 0.5 probability dropout layer after the 2nd convolutional layer.

Results: All of these models are trained using 12 epochs with 128 images per batch. Each model is compared based on the time it takes to evaluate an epoch, the performance on the testing data. It is important to understand the impact and function of the different layers in a deep learning application. While the addition of multiple layers makes that interpretation more difficult, it is still critical to understand the elements of the input data the model is responding to when making a prediction. It is possible to visualize the output from each of the convolutional layers as shown in Figures 2 and 3. The features tend to activate in response to the various edges of the digits. For the second layer of features, specific edges activate them which intuitively would lead to better segmentation between classes.

Below table shows a comparison of performance between the five different models. The accuracy for all five is fairly high due to the simplicity of the data but worth observing the relative differences. The presence of the dropout layers, used to prevent overfitting, do in fact improve accuracy with the addition of some computational time. The biggest difference in accuracy and time is the result of adding the second convolutional layer as well. It's clear that having multiple convolutional layers is beneficial but comes at a cost in terms of computation time. Reducing the number of hidden nodes also reduces accuracy but had a minor impact on timing.

Models	Structure	Training Time/Epoch	Test Accuracy
CNNModel1	ConLayer (3x3 filter, 32 feature maps) + 2x2 max pooling + flatten layer + fully connected hidden layer (64 nodes) + Softamx class output layer	50 Sec	0.9855
CNNModel2	ConLayer (3x3 filter, 32 feature maps) + 2x2 max pooling + flatten layer + Dropout layer (0.25) + fully connected hidden layer (64 nodes) + Softamx class output layer	58 Sec	0.9879
CNNModel3	ConLayer (3x3 filter, 32 feature maps) + 2x2 max pooling + flatten layer + Dropout layer (0.25) + fully connected hidden layer (128 nodes) + Softamx class output layer	65 Sec	0.988
CNNModel4	ConLayer (3x3 filter, 32 feature maps) + 2x2 max pooling + Dropout layer (0.25) + ConLayer (3x3 filter, 64 feature maps) + 2x2 max pooling + Dropout layer (0.5) + flatten layer + fully connected hidden layer (128 nodes) + Softamx class output layer	90 Sec	0.9906
CNNModel5	ConLayer (3x3 filter, 32 feature maps) + 2x2 max pooling + ConLayer (3x3 filter, 64 feature maps) + 2x2 max pooling + flatten layer + fully connected hidden layer (128 nodes) + Softamx class output layer	98 Sec	0.9905

Conclusion: After fitting a series of convolutional neural networks to the MNIST data, it is clear that incorporating convolutional layers adds computation time but also significantly increases accuracy. Too few nodes and layers lead to higher error, where too many nodes and layers tend to increase the time it takes to train a network. In addition, too many nodes and layers lead to a risk of overfitting the training data and obtaining a model that doesn't generalize well.

References

Chollet, F. (2018). Deep learning with Python. Shelter Island, NY: Manning Publications.

Jumpstart Code: Dr. Miller and Dr. Maren

Keras. Mnist_cnn.py. Retrieved May 02, 2018, from https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

A Comprehensive Guide to Convolutional Neural Networks, Retrieved May 02, 2018, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Appendix:

Figure 1 - Handwritten Digit 6 from training data set.

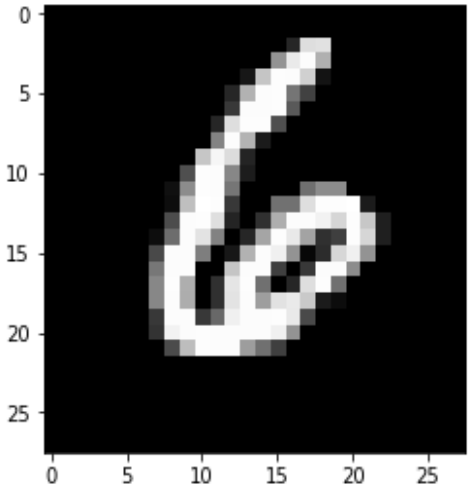


Figure 2 – Activation map or feature map output from Convolutional Layer 1.

Figure 3 - Activation map or feature map output from Convolutional Layer 2.