Project-6

Hand-Written Digit Recognition through Convolutional Neural Network

Machine Learning

Spring 2018

Submitted to

Dr. Hamed Sari-Sarraf

Submitted by

Sabbir Hassan

Taslim Anupom

Shamsul Arefeen

May 14, 2018

Introduction:

The project deals with the task of implementing a Convolutional Neural Network for identifying hand-written digits. It demonstrates the formation and functioning of a CNN in an object classification problem scenario.

Database and Platform:

The MNIST data of hand-written images are used as the data to train the network whereas MATLAB 2018a Release is used as the implementation platform and programming environment. The scope of the project consciously refrained from using any pre-trained network or hence, transfer learning from a trained network. However, using the available library functions in MATLAB was within the implementation scope and a new convolutional neural network was configured using those tools.

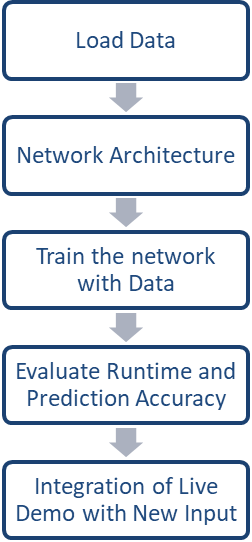
Objective:

Whereas learning through the implementation experience remains to be the high-level objective, the implementation goals may be summarized as below:

1. Through trial, design the architecture of the deep neural network in such a way that it trains from the data in a time-efficient manner and delivers predictions with reasonable accuracy.
2. Observe, manipulate and optimize the complexity of the network in terms of the number of earning parameters
3. Have the concepts of Early Stopping, Regularization integrated into the implementation to ensure best possible fitting of the learning model.

Implementation Strategy:

Following workflow was followed as the project implementation strategy.



Used the pre-defined functions for loading the MNIST data into the image dataset.

Started off with a 3-layer network and trialed with fewer and more layers

Used the MATLAB functions to train the network with loaded data with low initial Learning Rate

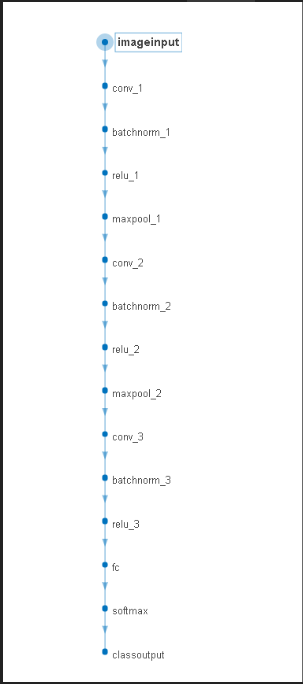
Runtime and Prediction Accuracy was observed for different architecture and complexity

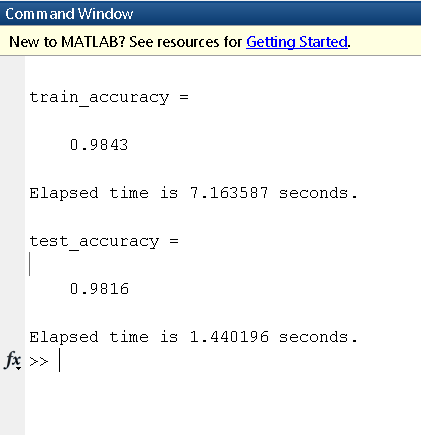
Integrate the module to capture camera image of hand-written digits as new input

Optimize Network Architecture through trial and error

Major Observations:

As we trained the network with different depths, the operational performance of timing, training and test accuracy obtained for the 3-layer network was as below:

Figure-1: 3-layer convolutional neural network architecture



As a very close accuracy level for both training and test set is obtained for a layer-3 architecture, it ensured avoidance of potential over-fitting and we decided to implement the deep learning convolutional neural network as a 3-layer architecture for our problem. It is to be noted that even for a 4-layer or a 1-layer architecture, the training accuracy and test accuracy did not show too far a difference between them but widened for a layer-4 architecture (indicating an overfitting) and a slight over-all decrease for both training and test accuracy in a layer-1 architecture.

The MATLAB generated runtime evaluation report for the 3-layer network is shown in Fig-2 below.

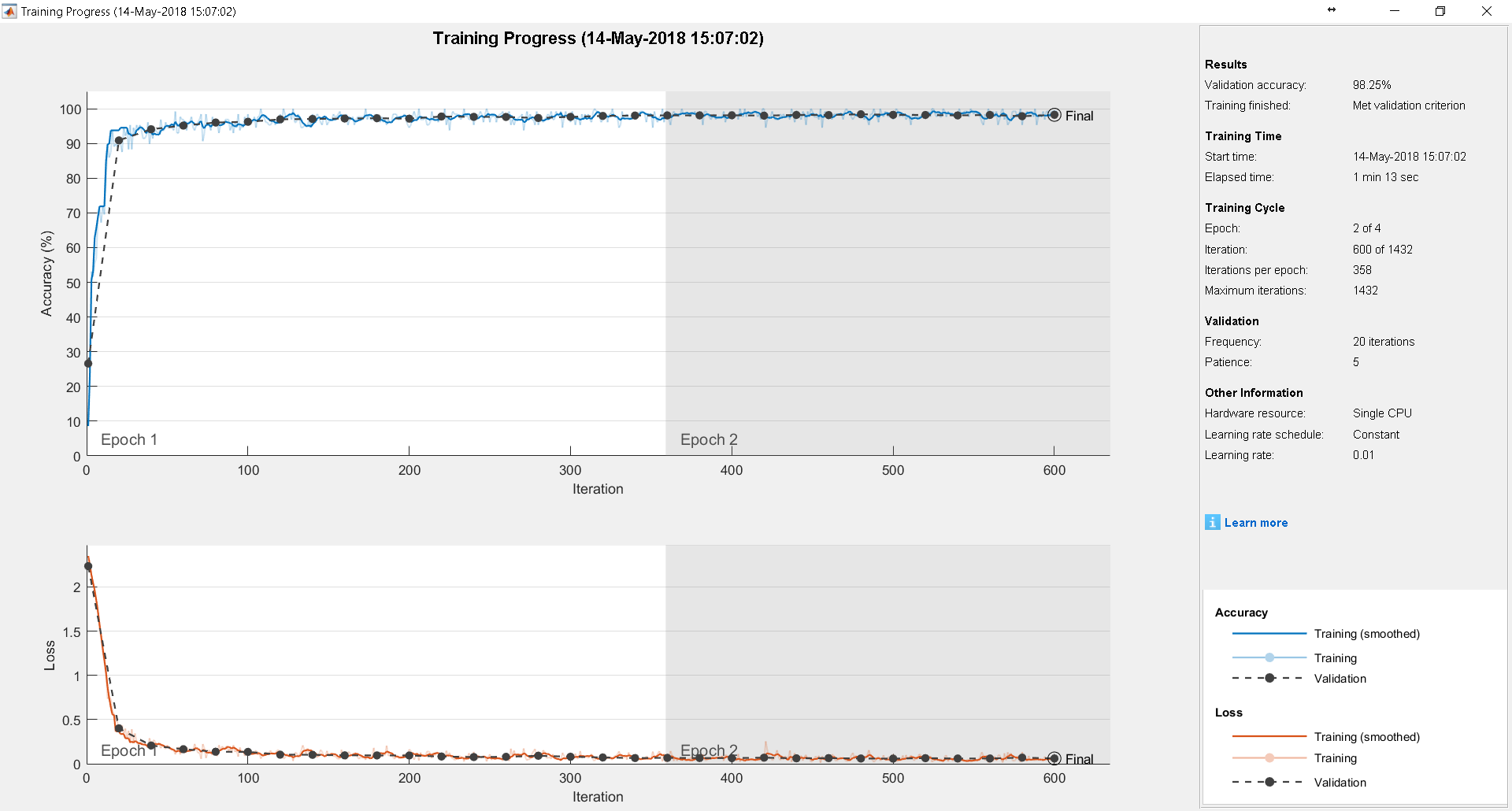


Figure-2: Training Progress Report for a 3-layer Convolutional Network

We also measured test sample accuracy for networks of different depth (keeping the convolution matrix/filters same and other architectural parameters same). We used new hand-written digits as test samples. It was observed that the depth of the network architecture didn’t quite demonstrate a clearly explainable trait. For instance, there was always the theoretical possibility that with more layers, the trained network might incline towards an overfitting problem and hence show more test errors; however it was observed that a single-convolution-layer network demonstrated a much better prediction accuracy compared to a 2-layer network whereas a 3-layer network offered better prediction than both a 2-layer and a 4-layer network. Most interestingly, a single-layer architecture got a sample (Image 11) classified correctly whereas it was misclassified by all other deeper networks.

Table-1: Evaluating different networks with real-time test inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Samples | 1-layer | 2-layer | 3-layer | 4-layer |
| Image 6 |  |  |  |  |
| 7 | X |  |  |  |
| 8 | X | X | X | X |
| 9 |  |  |  |  |
| 10 | X | X |  | X |
| 11 |  | X | X | X |
| 12 |  |  |  |  |
| 14 |  | X |  |  |
| 15 |  | X |  |  |
| 16 |  |  |  |  |

Note: ‘X’ marks wrong predictions by the learning model.

Network Complexity and Learnable Parameters:

Progressing with a 3-layer architecture we observed the complexity of the network with its learnable parameters as below.

Table-2: Learnable Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Conv1 | Conv2 | Conv3 | FC | Total |
| 370 | 1610 | 910 | 4910 | 7860 |

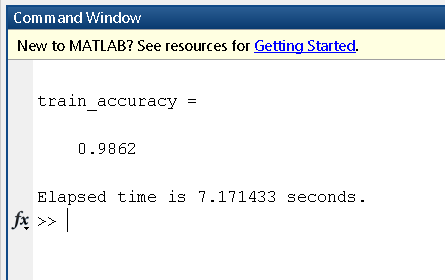
Figure-3 below shows the learnable parameters at different layers as evident in the MATLAB generated analysis report.



Figure-3: The learnable parameters with a 3-layer deep network from MATLAB generated Analysis

Additional Analysis:

For additional analysis, the 3-layer deep neural network was trained with all the data produced the results below:



We also observed the results with a view to identifying the major confusion pairs and ended up with the following confusion matrix for the whole data taken as training set.

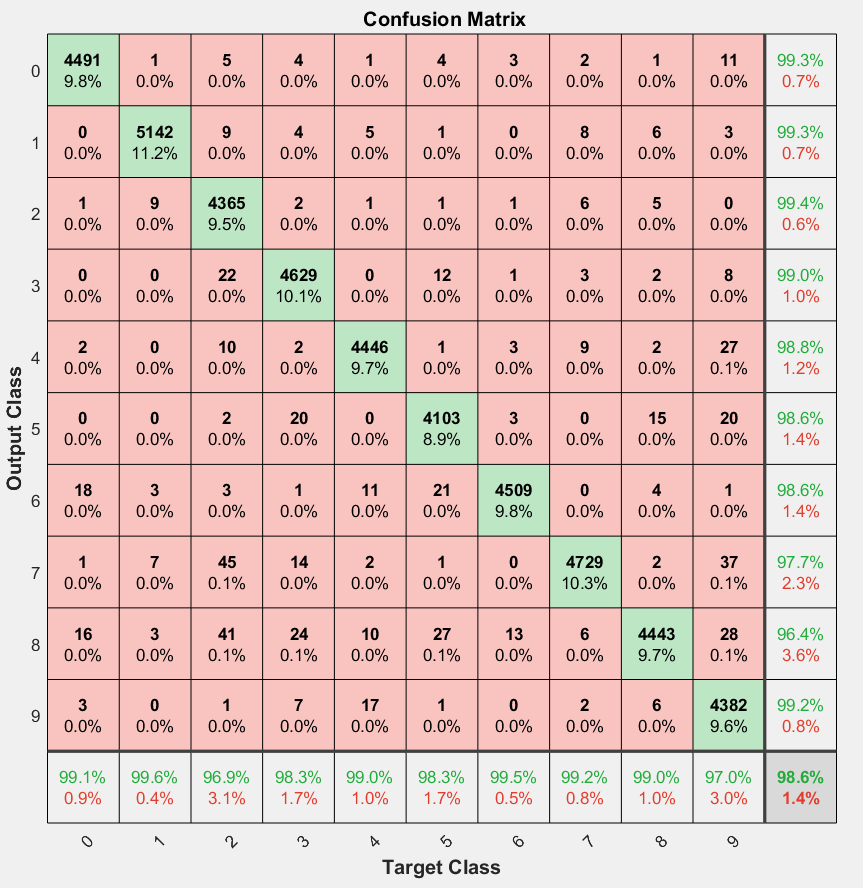


Figure-4: Confusion Matrix upon Training the network with entire dataset

We also went onto observe the activations at different layers by the convolution weights:

* Closer to the input layers, the learning happens with more salient features (such as the edges, intensities in the image).
* As the layers get deeper, the learning parameters get further complex and do not remain recognizable by observation for a human eye.

These observations are demonstrated through the figures below:

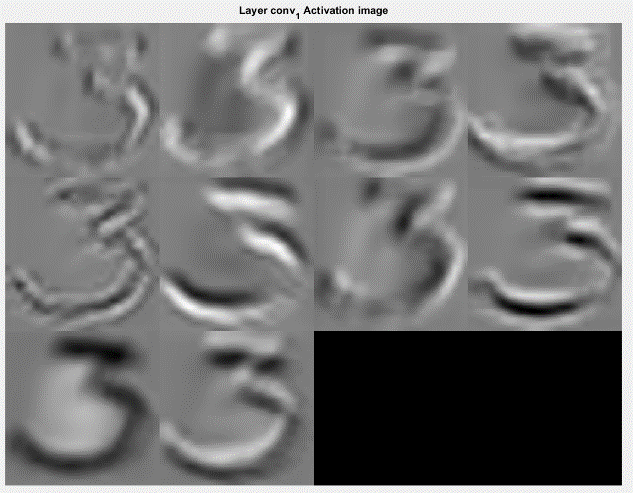
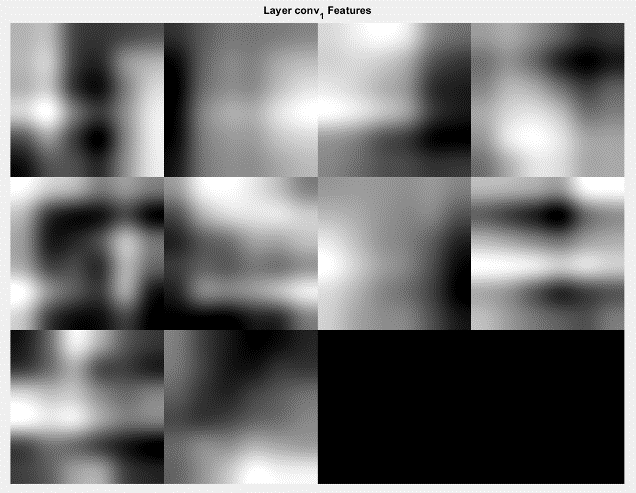


Figure-5: Convolution layer-1 Features (left) and activation (right)

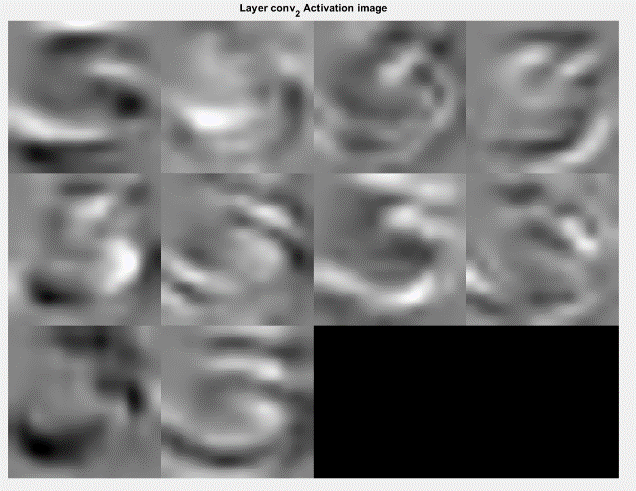
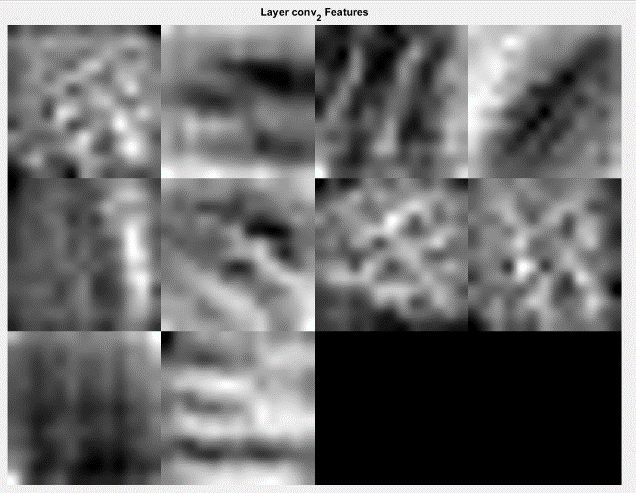


Figure-6: Figure-5: Convolution layer-2 Features (left) and activation (right)

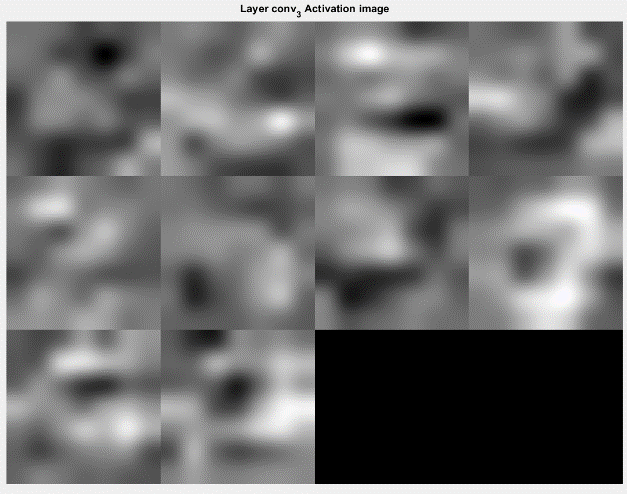
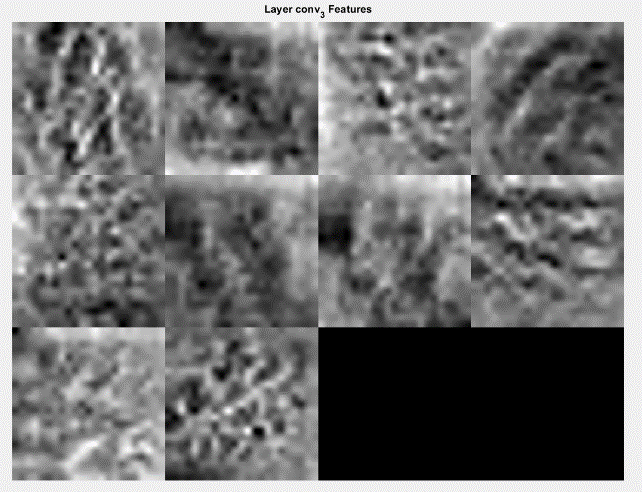


Figure-7: Figure-5: Convolution layer-3 Features (left) and activation (right)

It can easily be noticed that as the convolution layers got deeper, the features and the activation outputs became more unpredictable on our part; expectedly so, as the network learnt more complex features beyond the apparent parameters (ex: edges).

Live Demo Implementation:

We implemented a MATLAB module for capturing hand-written images through camera (laptop webcam) for live input and instant classification through the trained network. The demo leaves room for improvement as it is made on an idealistic manner able to read digits only from a relatively contrasting lighter background and there are no other image fractions in the vicinity. The distance from the camera and the size of the Handwritten digit also affected the correctness of classification of the images-probably due to change in resolution. However, with 36 hand-written images the network model was tested, and it showed consistent output as per the observations described in this report.

Conclusion:

It was overall an enthralling learning experience to implement a convolutional neural network for a very real-life problem of hand-written digit recognition. The code was well facilitated by the rich library functions of MATLAB, but it still needed a thorough understanding of how the theoretical techniques were put into play through those built-in functions. However, as we got deep into analysis and researching on the observations, it became evident that deep neural networks takes its own learning and feature extraction beyond our understanding with its increasing depth; however, there is no such rule that the deeper the better, finding the optimum depth is still a job left in the network designer’s hand.

**References:**

Create Simple Deep Learning Network for Classification

<https://www.mathworks.com/help/nnet/examples/create-simple-deep-learning-network-for-classification.html>

Specify Layers of Convolutional Neural Network

<https://www.mathworks.com/help/nnet/ug/layers-of-a-convolutional-neural-network.html>

Set Up Parameters and Train Convolutional Neural Network

<https://www.mathworks.com/help/nnet/ug/setting-up-parameters-and-training-of-a-convnet.html>

Train Network with Augmented images

<https://www.mathworks.com/help/nnet/ref/trainnetwork.html>

Fully Connected Network

<https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.fullyconnectedlayer.html#mw_bac20c29-f95b-423a-816a-428fc8e0463e>

Visualize Features of a Convolutional Neural Network

<https://www.mathworks.com/help/nnet/examples/visualize-features-of-a-convolutional-neural-network.html>

Visualize Activations of a Convolutional Neural Network

<https://www.mathworks.com/help/nnet/examples/visualize-activations-of-a-convolutional-neural-network.html>

Training Network Options

<https://www.mathworks.com/help/nnet/ref/trainingoptions.html#bu59f0q-2>