

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Digit Recognition with CNN | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Utsav Anantbhat |  | | | |
|  | | | | 1/19/2023 |  | | | |
|  | | |  | | |  | | |

Introduction

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that takes images as inputs, assigns weights and biases to them, and distinguishes one image from another. In this assignment, we were asked to implement a CNN in MATLAB and accomplish certain tasks. The primary goal of the CNN that was implemented was to recognize numbers or digits created in a myriad of styles and be able to differentiate them from one another.

Overview

As described in the handout, a typical CNN has five layers: Fully Connected/Inner Product (IP) layer, Convolutional layer, Pooling layer, Activation layer (ReLU – Rectified Linear Unit), and Loss layer.

Fully Connected Layer/Inner Product Layer

This layer is the simplest of the bunch and makes up neural networks. Each neuron of the layer is connected to all the neurons of the previous layer (See Fig 1). Mathematically it is modeled by a matrix multiplication and the addition of a bias term. For a given input x the output of the fully connected layer is given by the following equation,

f (x) = W x + b

W, b are the weights and biases of the layer. W is a two-dimensional matrix of m × n size where n is the dimensionality of the previous layer and m is the number of neurons in this layer. b is a vector with size m × 1 [1].

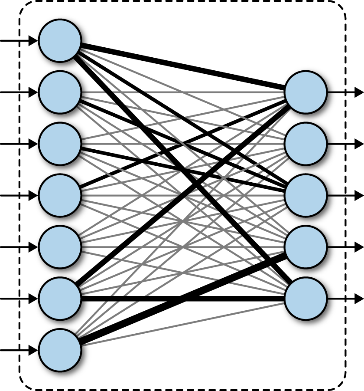


Figure 1: Fully Connected Layer

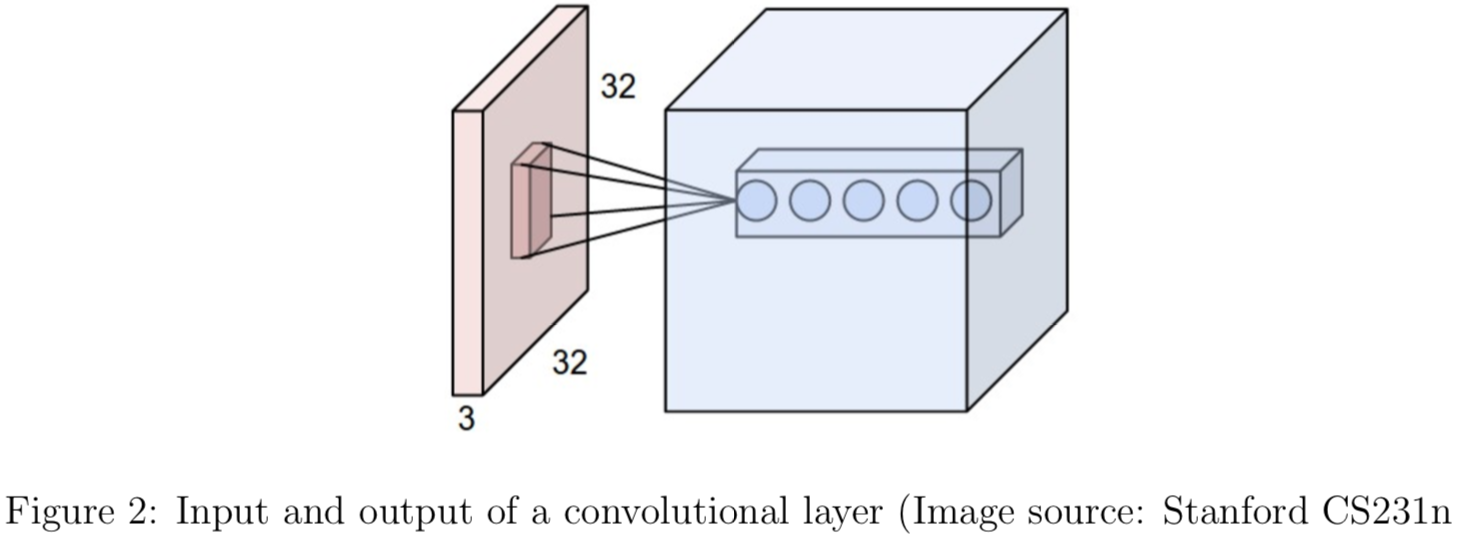
Convolutional Layer

This layer is the building block of the CNN. It contains kernels which are parameters that are learned by the network as it is trained. Convolution is performed using a k × k filter/kernel and a W × H image. The output of the convolution operation is a feature map. This feature map can bear different meanings according to the filters being used - for example, using a Gaussian filter will lead to a blurred version of the image. Using the Sobel filters in the x and y direction give us the corresponding edge maps as outputs [1].

The general convolution operation can be represented by the following equation:

f (X, W, b) = X ∗ W + b

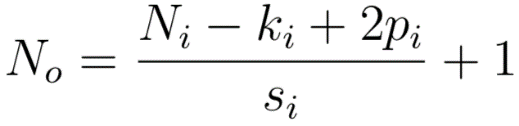
where W is a filter of size k×k×C\_i, X is an input volume of size Ni ×Ni ×C\_i and b is 1×1 element.



In the following example the subscript i refers to the input to the layer and the subscript o refers to the output of the layer.

* Ni - width of the input image
* Ni - height of the input image (image has a square shape)
* Ci - number of channels in the input image
* ki - width of the filter
* si - stride of the convolution
* pi - number of padding pixels for the input image
* num - number of convolution filters to be learnt

A grayscale image has 1 channel, which is the depth of the image volume. For an image with Ci channels - we will learn num filters of size ki × ki × Ci. The output of convolving with each filter is a feature map with height and width No, where

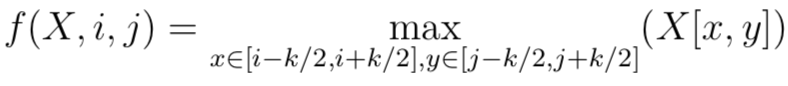


If we stack the num feature maps, we can treat the output of the convolution as another 3D volume/ image with Co = num channels [1].

In summary, the input to the convolutional layer is a volume with dimensions Ni × Ni × Ci and the output is a volume of size No × No × num. Figure 2 shows a graphical picture [1].

Pooling Layer

The pooling layer combines the outputs of neurons at the previous layer into a single neuron in the next layer. This in turn reduces the dimensions of the hidden layer. As described in the handout, the pooling layer operates on each feature map separately and replaces a local region of the feature map with some aggregating statistics like max or average. In addition to reducing the size of the feature maps, it also makes the network invariant to small translations. This means that the output of the layer doesn’t change when the object moves a little [1]. We are given the following function which is applied to a padded feature map X:



* k = kernel size
* s = stride
* p = padding

Activation Layer (ReLU – Rectified Linear Unit)

These layers introduce the non-linearity in the network and give the power to learn complex functions. The most commonly used non-linear function is the ReLU function defined as follows:

f (x) = max(x, 0)

The ReLU function operates on each output of the previous layer.

Loss Layer

The loss layer “continually checks the inner product layer’s guesses against the actual values with the goal of minimizing the difference between the guess and the real value as much as possible” [2]. It does so by adjusting the weights in both the convolution and inner product layers. And then to convert the output to a probability score, a softmax function is used. This operation is given by,

p = softmax(W x + b)

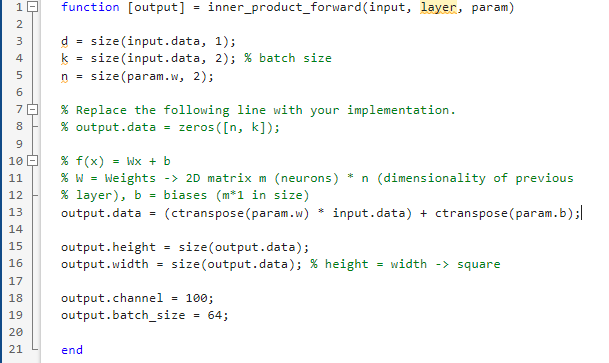
where, W is of size C × n where n is the dimensionality of the previous layer and C is the number of classes in the problem.

This layer also computes a loss function which is to be minimized in the training process. The most common loss functions used in practice are cross entropy and negative log-likelihood.

Programming

Part 1: Forward Pass

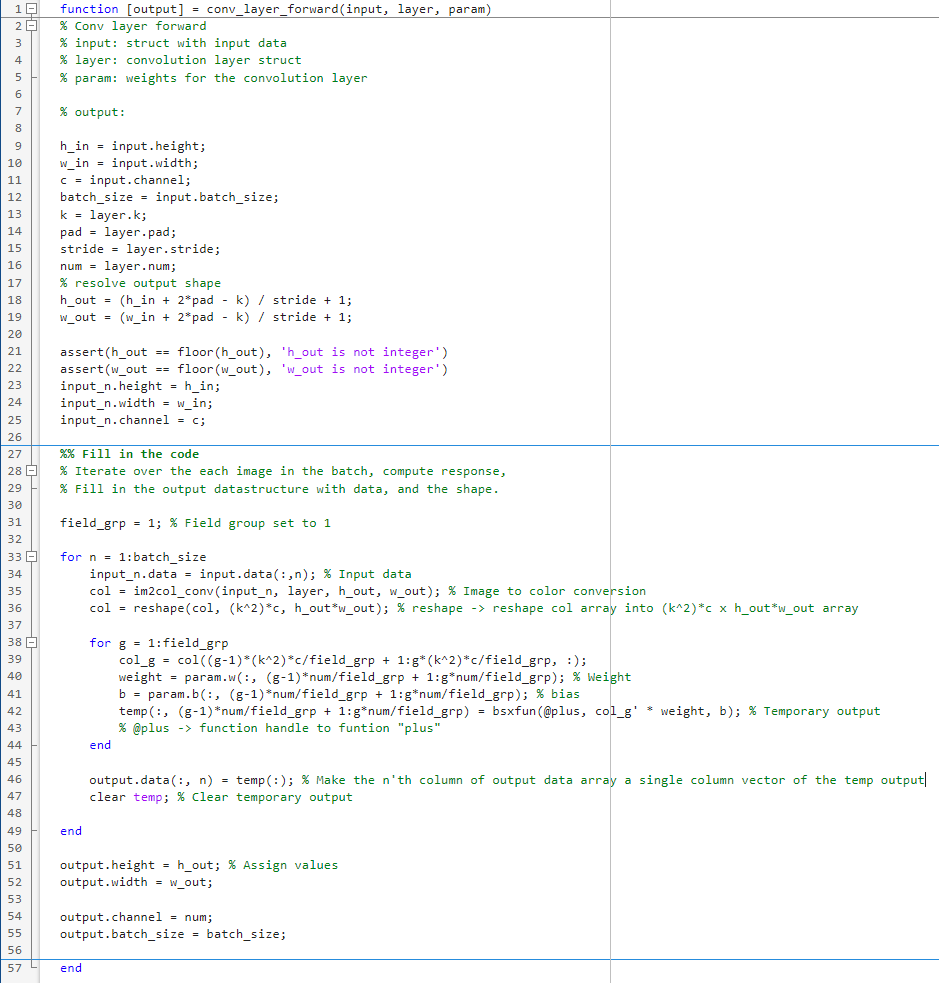
Q1.1 – Inner Product Layer:



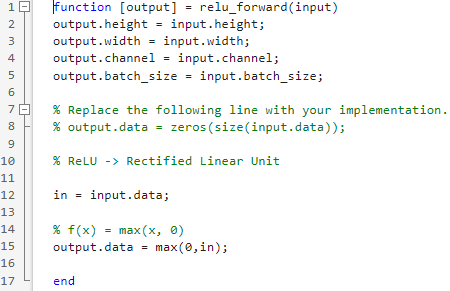
Q1.2 – Pooling Layer:



Q1.3 – Convolutional Layer:



Q1.4 – ReLU:

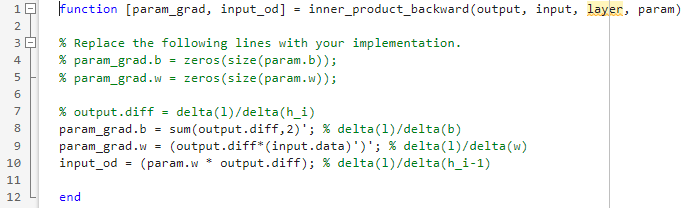


Part 2: Back Propagation

2.1 – ReLU (backward):



2.2 – Inner Product Layer (backward):



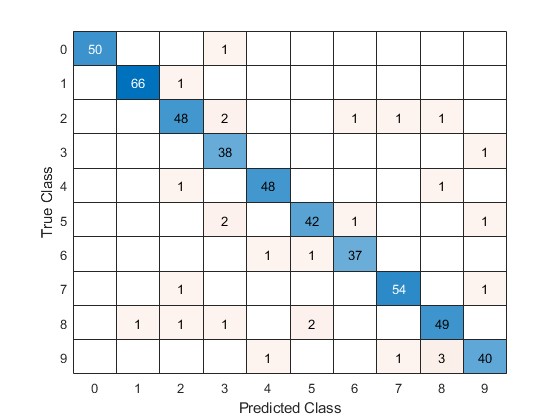
Part 3: Training

3.1 – Training:



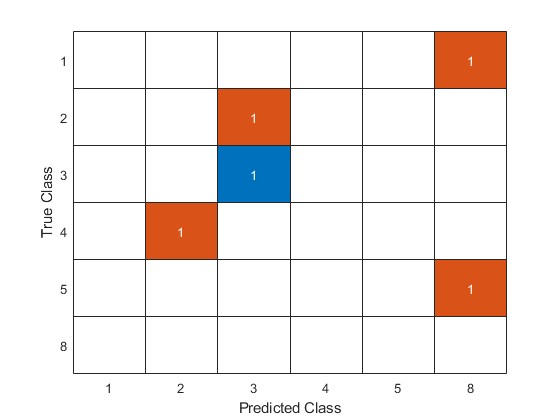
According to these results, the accuracy averages to 95% after 3000 iterations.

3.2 – Testing the Network:



From the data above, we can see that some predictions were inaccurate. Many ‘9’ digits were predicted as ‘8’ digits. ‘8’ digits were predicted as ‘5’ or ‘2’. Type of font, style, and size of alphanumeric characters can have an impact on visualization/perception.

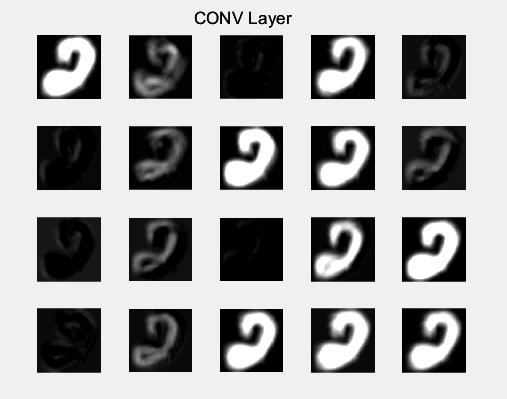
3.3 – Real World Testing:

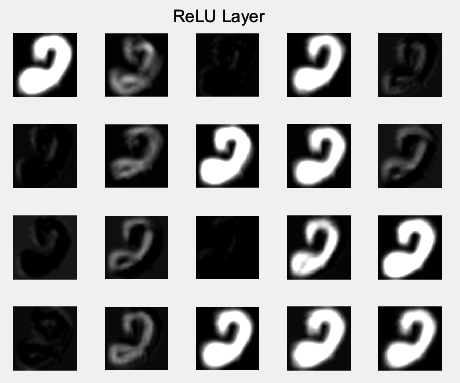


Here we created another confusion matrix using real word digit examples. Based on the results, the accuracy of this model is a rather unsatisfactory 20%. The images used for this model are contained within the project in a folder called “real\_numbers” for further review.

Part 4: Visualization

4.1 – Feature Maps:





4.2 – Original Image vs. Feature Maps:

The original image is as follows:



The original image appears to be bounded and the feature maps essentially apply different filters such as sharpening, blurring, etc. and detect edges, whitespaces, and other features necessary to classify the image.

Challenges Faced

Due to time limitations, part 5 of the project was not implemented. Other minor runtime errors were faced during initial stages of the project but were ironed out.

References

[2] “A friendly introduction to Convolutional neural networks,” Hashrocket.com. [Online]. Available: https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks. [Accessed: 19-Jan-2023].