Comparison Of CNN Built from scratch with Tensorflow Implementation

Sujit Rai : 2017CSM1006, Manish Singh : 2017CSM1003

Computer Science & Engineering, IIT Ropar

Abstract

Deep learning methods have resulted in significant performance improvements in several application domains and as such several software frameworks have been developed to facilitate their implementation. Convolutional neural network are among the popular deep learning methods.MNIST is well-known dataset of handwritten digits. We have used LeNet-5 architecture for MNIST digits recognition task. LeNet-5 is proposed by Y.LeCun, which is known to work well on handwritten digit recognition. In this project we have done a comparative study of the Lenet-5 and the Tensor Flow Implementation on the MNIST database of hand written digits based on accuracy and efficiency. Tensor flow built in implementation is used while we have developed the Lenet-5 model from scratch. All hyper parameters are kept the same in both the implementation.

Introduction

Convolutional Neural Networks are are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

The MNIST is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem.The dataset was constructed from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST). This is where the name for the dataset comes from, as the Modified NIST or MNIST dataset.

The images were taken from a variety of dataset and normalized to center so that there is no need for preprocessing and the images can be used directly. Each image is a 28 by 28 pixel square (784 pixels total). A standard spit of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it. It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy.

Related Work

LeCun, Yann, et al. in his paper "Gradient-based learning applied to document recognition." has applied several methods to handwritten character recognition and compared them on a standard handwritten digit recognition task. He found out that convolutional neural network specifically designed to deal with the variability of the two dimensional shapes outperforms all other techniques.

Using character recognition as a case study, the paper showed that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images.

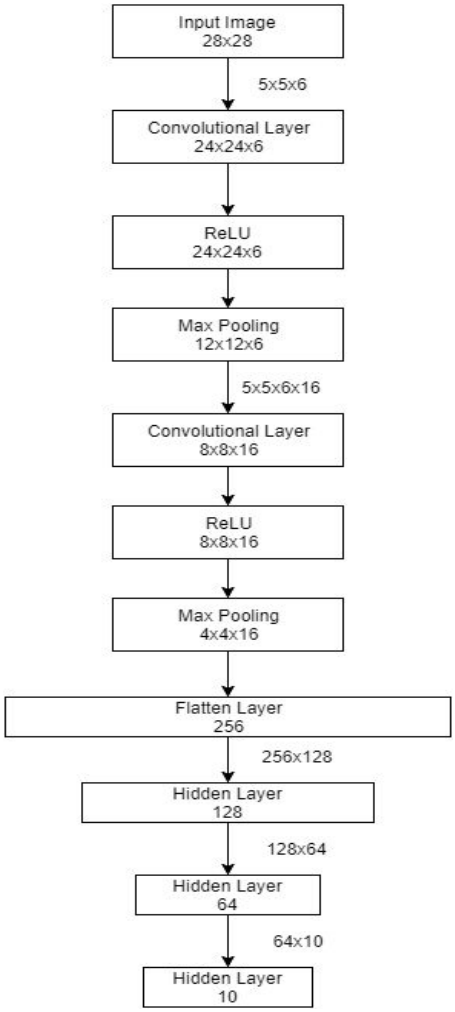
The systems described in this paper shows that increasing the role of learning seems to have improved the overall performance of recognition systems. Convolutional Neural Networks have been shown to eliminate the need for hand-crafted feature extractors.

More specifically, the methods and architectures presented in this paper offer generic solutions to a large number of problems encountered in pattern recognition systems.

For testing the architecture the author used Neural Network classifier initially trained on 500,000 of character images from various origins spanning the entire printable ASCII set. This contained both handwritten and
machine-printed characters that had been previously size normalized at the string level. Additional images were generated by randomly distorting the original images using simple affine transformations of the images. The network was then further trained on character images that had been automatically segmented from check images and manually truthed . The network was also initially trained to reject non-characters that resulted from segmentation errors. The recognizer was then inserted in the check-reading system and a small subset of the parameters were trained globally (at the field level) on whole check images. On 646 business checks that were automatically categorized as machine printed, the performance was 82%
correctly recognized checks, 1% errors, and 17% rejects. Inspired by the success of the Neural Network in hand written character recognition we decided to build specific CNN LeNet-5 from scratch and compare its performance to the Tensor Flow implementation of the CNN on the MNIST digit images.

Methodology

Convolutional Neural Network is considered to perform better for the problems involving images. For predicting the hand-written digits classification we used a 2 layer convolutional neural network layer and 2 layer fully connected neural network layer.



Architecture :

- The input layer is 32x32 grayscale image of the handwritten digits.
- 6 filters of dimensions 5x5 are applied to this image with stride of 1.
- Then we added non-linearity as ReLU (Rectified Linear Unit).
- Max polling layer with stride 2 and window size 2x2 was applied which summaries the convolution outputs.
- This output of max pooling is then used as an input to the next convolutional layer.
- 16 Filters of dimensions 5x5x6 is used for convolution with stride 1.
- ReLU and Max Polling Layer is applied with same parameters
- The Output is then of dimensions 5x5x16, which is then flattened into a vector of dimensions 400x1

- Then 2 Fully connected layers are applied with hidden layer inputs of size 128 & 64

After performing the derivations of the forward and backward propagation it was found that there is not much difference in the formulas of the Forward and Backward Propagation of the CNN and ANN. The matrix multiplications in the ANN are replaced by convolution operation in CNN.

Forward Propagation

- Convolution Layer : For performing convolution with the input. matlab's built-in function 'convn' was used with convolution type as 'valid'.
  C = convn(Image,Filter,'valid');
- ReLU Layer : For performing ReLU layer the weights with negative values were made zero.
  R = C; R(find(C<0) = 0;
- Max Pooling Layer : For performing max pooling, windows of size 2x2 were choosen and only those weights were kept non zero which had maximum value in the current window.
  P = Max(R(2i - 1 : 2i , 2j - 1 : 2j)), i,j = 1,2, …

Backward Propagation

- Convolution Layer : The back propagation for convolution layer can be divided into 2 parts.
  - Gradient of weights
  - Gradient of input

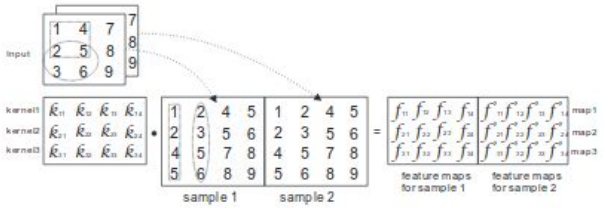
  Gradient of weights can be calculated by applying convolution operation of gradient of output with the input.
  delW = convn(input,del(C),'valid')
  Gradient of input can be calculated by first rotating the weights by 180 degrees and then performing convolution operation of this filter with the gradient of the output.
  delInput = convn(del(C),rotate(W,180),'full')
- Max Pooling : For the places which had max values in the window will be assigned the same gradients as that of the output gradients.
- ReLU : The gradients of the ReLU layer will be same as that of the gradients of the max pooling layer.

Time Complexity : The time taken to execute 1 epoch is around 15 minutes.

Since time taken for execution of 1 epoch is very high. Therefore there is a need for implementing some mechanism for faster convolution operations.

Vectorization of Convolutional Neural Network

Since the matrix multiplications can be easily performed on CPU and GPU therefore the convolution operations can be converted into matrix multiplication. This conversion of convolution to matrix multiplication can be done by unrolling the windows which are involved in the convolution into a single vector. Thus each convolution window will be converted to a single vector which can be concatenated vertically to form a matrix. The filters can also be unrolled to form a vector. Then the filter vector and the input matrix can be multiplied to obtain the same output as that of convolution.

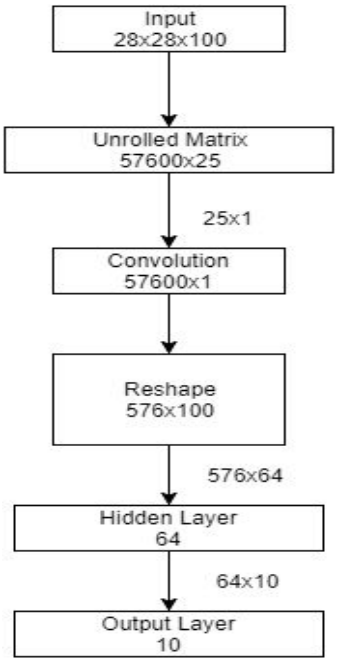


The vectorization approach can also be used in the mini-batch gradient descent approach, Wherein the matrix formed after unrolling of each image can be concatenated vertically. The forward and backward propagation using vectorization becomes the same as in the ANN.

Architecture :

For experimental purposes we implemented vectorization on a model consisting of 1 convolutional layer and 2 fully connected layers.

- The input images were vectorized and concatenated to form a matrix of size 57600x25
- This was then multiplied with the filter of dimensions 25x1
- The resultant is passed through a ReLU layer
- The result is then reshaped to 576x100
- Then 2 fully connected layers were applied.



Optimizers :

Momentum : SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another. Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

Here the previous gradients are multiplied with the momentum and added with the current gradients.

RMSProp : RMSprop is an unpublished, adaptive learning rate method proposed by Geoff Hinton.RMSprop as well divides the learning rate by an exponentially decaying average of squared gradients. Hinton suggests γ to be set to 0.9, while a good default value for the learning rate η is 0.001.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

Results & Discussions

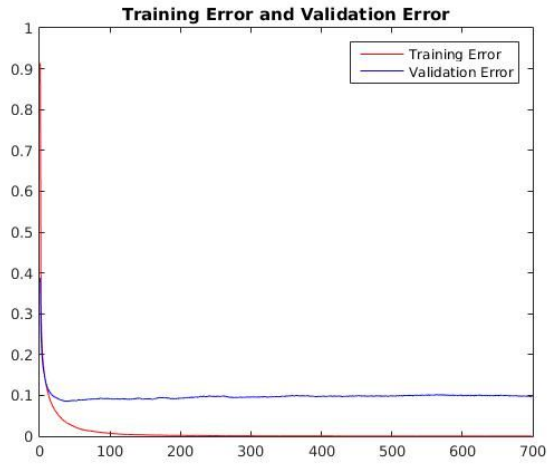
Accuracy of CNN with RMS prop optimizer : 0.60

Accuracy of Tensorflow CNN : 0.86
Accuracy of CNN without RMS prop optimizer : 0.106

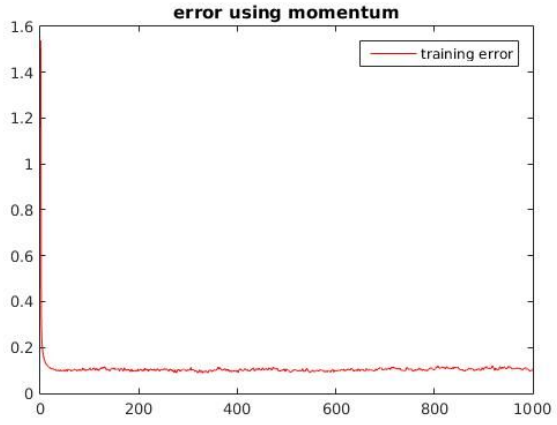
Execution time per epoch with vectorization : 78sec
Execution time per epoch without vectorization : 15mins
Execution time per epoch on tensorflow : 29sec

Graphs :

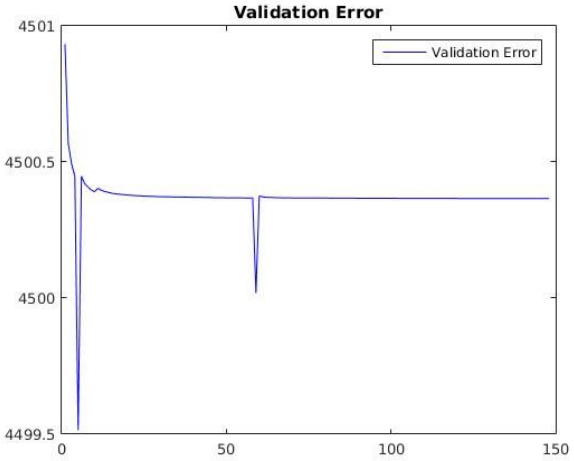
- Graphs containing training error vs validation error for vectorized implementation of RMS prop on 700 epochs, 0.0001 learning rate and 1-10^6 weight decay
- As can be seen from the graphs the error reduces to a very low value.



- Graphs of error plotted using momentum as an optimizer and learning rate of 0.01 and momentum of 0.9
- As can be seen from the graph the error fluctuates very much.



- Graph containing validation error of unvectorized CNN when run on 100 epochs using momemtum.



- As can be seen from the above graph there are fluctuations in the validation error as well because of the momemtum.

Summary

This report describes the 2 implementations of using Convolutional neural network for deep learning applications. The time taken by the un-vectorized code was 15mins per epoch while the time taken by the vectorized code was only 79sec. We then used different optimizers for improving our accuracy. It was seen that normal CNN achieved an accuracy of 10% on 100 epochs while the CNN with RMSProp as an optimizer achieved around 60% accuracy.

Future Work

One of the disadvantages of CNN is that it is variant to rotational changes and invariant to translational changes. Therefore we can use capsule networks which are invariant to rotation transformations which can further be used in the classification, detection and localization of hand-written character recognition.

References

LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

Ruder, Sebastian. “An Overview of Gradient Descent Optimization Algorithms.” Sebastian Ruder, Sebastian Ruder, 23 Nov. 2017, ruder.io/optimizing-gradient-descent/.