Comparing Multi CNN and MAX_POOL CNN in size invariant object recognition

Srinivasa Reddy Duggempudi,
Deep Learning,
Department of Computer Science,
Wright State Univesity,
Dayton ,OH,45324,USA,
Email:{duggempudi.2@wright.edu}

A lot of research has been done in the field of object recognition in images. Convolutional neural networks have achieved state of art performance in object recognition. Classifying size invariant objects in images is still a very hard task. Multi CNN are used to classify size invariant images. In this project a MAX_POOL network is designed to classify size invariant objects in images and its performance is compared with MLP(Multi Layer Perceptron),CNN(Convolutional Neural Networks) and Multi CNN. A new dataset is designed by modifying each image in MNIST dataset and used as training images in this project.

1.INTRODUCTION

Classifying size invariant objects in images is a challenging task. CNN performs very good in classifying objects in images in which objects vary in position. But they perform very poorly in classifying objects in images in which objects vary in size. Multi CNN tackle this problem by training each CNN in Multi CNN architecture with resized copies of input images. In this project a MAX_POOL network is created by replacing a pooling layer with MAX_POOL layer to improve the performance of CNN in size invariant object recognition in images. The performance of Multi CNN and MAX_POOL CNN are compared on a modified MNIST dataset.

(a)

(b)

Fig. 1. Illustration of size variance of objects in images .
(a) large football object (b)small football object in images

2. MULTICNN

Multi CNN contains multiple CNNs and their softmax output is added. The Multi CNN used in this project consists of 2 CNNs . CNN1 is trained with [28,28] size images and CNN2 is trained with the resized[20,20] images. Each CNN consists of two convolutional layers each followed by a polling and a ReLU layer. Each ReLU layer after second polling layer is connected to two fully connected layers.The strides,padding, filter shapes and the fully connected layer shape of CNN is described in the Table 1.

Abbreviations used in the table:

- Conv : convolution
- Pool : Pooling
- padd : padding
- ReLU : Rectified Linear Unit
- conn : Connected

Table 1. Architectur of CNN1 in the Multi CNN

Layer	Strides	Zero Padd	Filter shape	Size before	Size after	Hidden Layer shape
Input	-	-	-	[1,784]	[1,28,28]	-
Conv	[1,1]	-	[32,1,3,3]	[1,28,28]	[32,26,26]	-
Pool	[2,2]	-	-	[32,26,26]	[32,13,13]	-
ReLU	-	-	-	[32,13,13]	[32,13,13]	-
Conv	[1,1]	-	[64,32,3,3]	[32,13,13]	[64,11,11]	-
Pool	[2,2]	-	-	[64,11,11]	[64,6,6]	-
ReLU	-	-	-	[64,6,6]	[64,6,6]	-
Flat	-	-	-	[64,6,6]	[1,2304]	-
Fully conn layer	-	-	-	[1,2304]	[1,625]	[2304,625]
Fully conn layer	-	-	-	[1,625]	[1,10]	[625,10]
s_max	-	-	-	[1,10]	[1,10]	-

The architecture of the CNN2 used in the Multi CNN used is described in the Table 2. The softmax output of CNN1 and CNN2 is added is used for the for the computation of the cost.

Table 2. Architecture of CNN2 used in Multi CNN

Layer	Strides	Zero Padding	Filter Shape	Size before	Size after	Shape
Input	-	-	-	[1,784]	[1,28,28]	-
Resize	-	-	-	[1,28,28]	[1,20,20]	`
Convolution	[1,1]	-	[32,1,3,3]	[1,28,28]	[32,18,18]	-
Pooling	[2,2]	-	-	[32,18,18]	[32,9,9]	-
ReLU	-	-	-	[32,9,9]	[32,9,9]	-
Conclusion	[1,1]	-	[64,32,2,2]	[32,9,9]	[64,8,8]	-
Pooling	[2,2]	-	-	[64,8,8]	[64,4,4]	-
ReLU	-	-	-	[64,4,4]	[64,4,4]	-
Flatten	-	-	-	[64,4,4]	[1,1024]	-
Fully connected	-	-	-	[1,1024]	[1,625]	[1024,625]
Fully Connected	-	-	-	[1,625]	[1,10]	[625,10]
Softmax	-	-	-	[1,10]	[1,10]	-

3.MAX_POOL LAYER

The MAX_POOL layer used in the project is described using the values in Table 3 .The values in the table describe a The shape of the image in the below figure is [1,6,6].The element in the 0 th position suggests image contains 1 channel. The value 6 in 1 th and 2 nd position suggests that the image is 6X6 pixel wide. After applying ReLU activation to a channel the values in each pixel of the channel will be greater than zero.

Filter_Content : The sum of the value of each pixel in the channel divided by the total size of the channel.

Filter_Content[a,b]: The filter content in the subarray of shape [a,b] in the channel.

Max_Array[a,b]: Of all the subarrays of shape [a,b] in channel the the subarray which contains maximum Filter_content[a,b].

Table 3.

0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.2	0.2	0.1	0.1
0.1	0.3	0.4	0.4	0.3	0.1
0.1	0.3	0.4	0.4	0.3	0.1
0.1	0.1	0.2	0.2	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1

Filter_Content of the image described in the Table 3. is 0.16.

The array describing the Filter_Content[3,3] of the image in Table 3. is shown in Table 4.

Table 4.

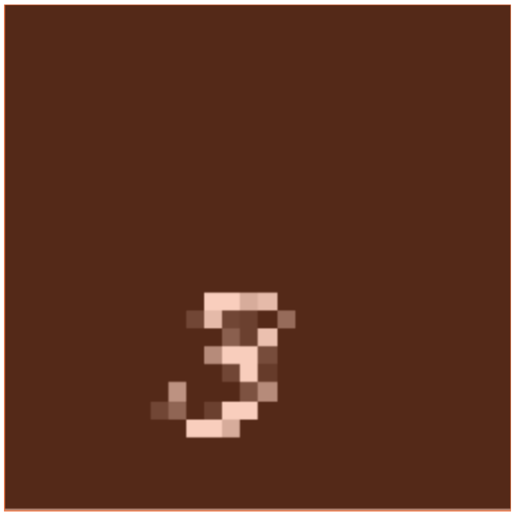
0.16	0.21	0.21	0.16
0.22	0.30	0.30	0.22
0.22	0.30	0.30	0.22
0.16	0.21	0.21	0.16

Max_Array[3,3] of the values in the table 3 is shown in Table 5.

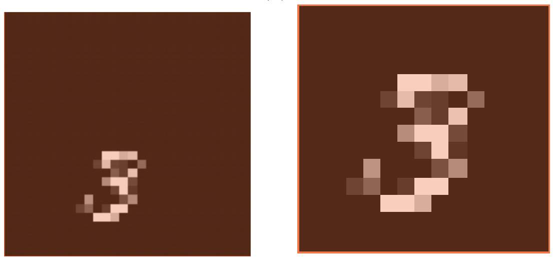
Table 5.

0.4	0.4	0.3
0.4	0.4	0.3
0.2	0.2	0.1

MAX_POOL layer[image,threshold]: (2,2) pooling of a layer will reduce the size of the image of layer of [M,N] to size [M/2,N/2]. In this max_or_pool layer first the filter_content of a layer is calculated if max filter content is grater that a certain threshold pooling of a channel is conducted or else Max_Array[M/2,N/2] of the channel is calculated. The inputs to the max pool layer image and threshold.



(a)



(b) (c)

Figure 2. (a) The input to the MAX_POOL layer. (b) The output if the Filter_Content is greater than threshold (b) output if the Filter_Content is less than or equal to threshold.

4.PROJECT

4.1 BASELINE

4.1.1 MULTI LAYER PERCEPTRON(MLP)

The multilayer perceptron contains one hidden layers and a softmax output layer. The input to the MLP is an array of 784 pixels . Each 28*28 size image in modified MNIST data is flattened to 784 pixel vector. After each hidden layer a sigmoid function is applied . The output layer consists of 10 neurons . A softmax layer is applied after the output layer. The architecture of the MLP is shown in the Table 6.

Table 6.

Layer	Shape	Image shape before	Image shape after
Input	-	[1,784]	[1,784]
Hidden	[784,1024]	[1,784]	[1,1024]
Output	[1024,10]	[1,1024]	[1,10]
Softmax	-	[1,10]	[1,10]

4.1.2 CONVOLUTIONAL NEURAL NETWORK(CNN)

The architecture of the convolutional neural network used for the baseline is as follows. The network consists of two convolutional layers and two fully connected layers. Each convolutional layer is followed by a pooling layer and a ReLU layer . The architecture of the convolutional neural network used for the baseline is shown in the below figure.

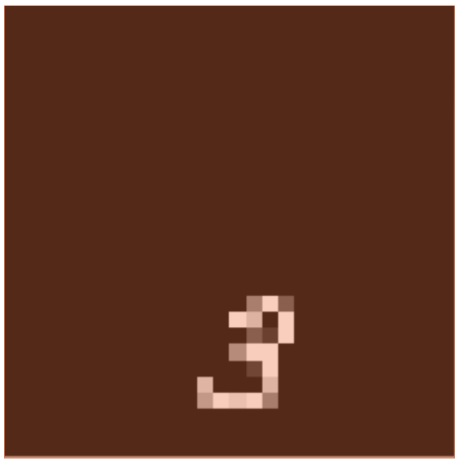
Table 7. Architecture of CNN

Layer	Strides	Zero Padd	Filter shape	Size before	Size after	Layer shape
Input	-	-	-	[1,784]	[1,28*28]	-
Conv	[1,1]	-	[32,1,3,3]	[1,28,28]	[32,26,26]	-
Pool	[2,2]	-	-	[32,26,26]	[32,13,13]	-
ReLU	-	-	-	[32,13,13]	[32,13,13]	-
Conv	[1,1]	-	[64,32,3,3]	[32,13,13]	[64,11,11]	-
Pool	[2,2]	-	-	[64,11,11]	[64,6,6]	-
ReLU	-	-	-	[64,6,6]	[64,6,6]	-
Flattening	-	-	-	[64,6,6]	[1,2304]	-
Fully Conn layer	-	-	-	[1,2304]	[1,625]	[2304,625]
Fully conn layer	-	-	-	[1,625]	[1,10]	[625,10]
s_max	-	-	-	[1,10]	[1,10]	-

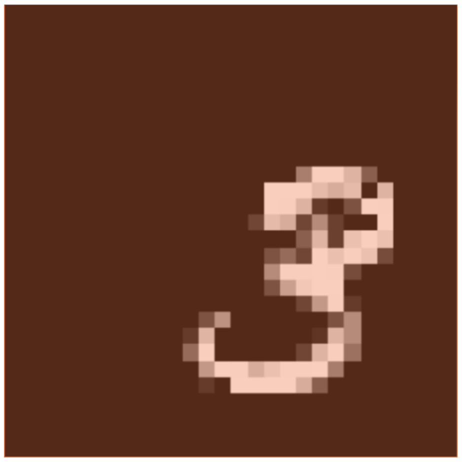
4.2 DATASET

MNIST dataset is used for training and testing the networks in the project. Each Image in MNIST dataset is represented as an array of 28*28 array. A random number is selected between 6 and 28. Suppose if the selected random number is M ,the 28*28 image array is resized to M*M. After resizing the image to a M*M array the image is positioned at a random position in a 28*28 array and padded with zeros around it . The same resizing is applied to every image in training set and testset in the MNIST data. This resizing and padding causes the creating of new data set in each image is a 28*28 array but each object in the image is of random size between 6*6 and 28*28.

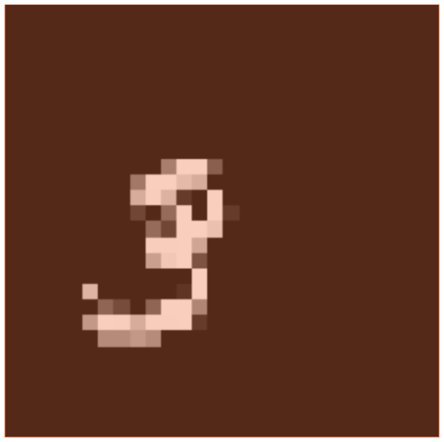
(a)



(b)



(c)



(d)

Figure 3. The images in the dataset. (a) Original image (b),(c),(d) modified images.

4.3 LIBRARIES

The programming language python is used for designing and implementing the networks in this project. In python the library Theano is used for designing the networks and the library Tensorflow is used for importing the MNIST dataset. PIL library in python is used for resizing the images import from Tensorflow.

4.4 MAX_POOL CNN

The architecture of MAX_POOL CNN is similar to conventional CNN except that pooling layer is replaced by a MAX_POOL layer. MAX_POOL CNN consists of 2 convolutional layers. The first convolutional is followed by a MAX_POOL layer. The second convolutional layer is followed by a normal (2,2) max pool layer. MAX_POOL layer and pooling layer are followed by ReLU layer. The output of the second convolutional layer is passed to 2 fully connected layers. The output of the network is passed to a softmax layer. Drop out is introduced into the network to minimize overfitting. Drop out used in the convolutional layer is 0.2 and dropout used for the fully connected layers is 0.5. The strides,zero padding and the filter shapes used in the network are described in the following table. The shape of the fully connected layers is also described in the Table 8.To make things very clear the shape of the tensor before and after each layer is also included in the table.

Table 8. Architecute of MAX_POOL CNN

Layer	Strides	Zero Padd	Filter shape	Size before	Size after	Layer shape
Input	-	-	-	[1,784]	[1,28,28]	-
Conv	[1,1]	-	[32,1,3,3]	[1,28,28]	[32,26,26]	-
MAX_POOL	[2,2]	-	-	[32,26,26]	[32,13,13]	-
ReLU	-	-	-	[32,13,13]	[32,13,13]	-
Conv	[1,1]	-	[64,32,3,3]	[32,13,13]	[64,11,11]	-
Pool	[2,2]	-	-	[64,11,11]	[64,6,6]	-
ReLU	-	-	-	[64,6,6]	[64,6,6]	-
Flattening	-	-	-	[64,6,6]	[1,2304]	-
Fully Conn layer	-	-	-	[1,2304]	[1,625]	[2304,625]
Fully conn layer	-	-	-	[1,625]	[1,10]	[625,10]
s_max	-	-	-	[1,10]	[1,10]	-

4.5 TRAINING and TESTING

Training and testing of the network are conducted using

modified MNIST dataset. In each batch of training the average filter content of the each batch is calculated and is passed as the threshold for the MAX_POOL layer. Ten epochs of training are conducted MNIST dataset. In each EPOCH a new dataset is created from the MNIST dataset and is used for the training. MNIST dataset contains 55000 training images, so after 10 epochs of training the network will be trained with 550000 images. MNIST dataset contains 10000 test imges. Testing of the images is conducted on those test images. The training parameters used in the network are tabulated in below table.

Table 9. Training parameters

Learning Rate	0.01
Batch Size	100
Drop out Conv	0.2
Drop out Fully Connected layers	0.5

5.RESULTS

The modified MNIST dataset is trained on MLP ,CNN,

Multi CNN and MAX_POOL CNN and their classification accuracy is tabulated in Table 9.

Table 10. Results.

Architecture	Accuracies	Average
MLP	• 33.1 • 32.7 • 34.2 • 33.5	33.3
CNN	• 79.2 • 78.2 • 77.6 • 80.3	78.8
Multi CNN	• 81.9 • 81.1 • 80.7 • 82.1	81.4
MAX_POOL CNN	• 83.4 • 84.3 • 83.1 • 82.7	83.3

6.CONCLUSION

The results obtained are encouraging and gives interest for future research. MAX_POOL CNN performs better than Multi CNN in classifying modified MNIST dataset. In the future the MAX_POOL CNN has to be tested with ImageNet and CIFAR dataset.

7.REFERENCES

- Learning scale-variant and scale-invariant features for deep image classification.

  [https://arxiv.org/pdf/1602.01255.pdf]

- Implementing CNN in Theano.

  [https://github.com/Newmu/Theano-Tutorials]