华南理工大学

South China University of Technology

The Experiment Report of Deep Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: Deep Learning

Author:

Muhammad Nabeel

Malik Fayaz Ahmed

Ana Madeleyn Oporto Guzman

Student ID： 201722800002

Student ID: 201722800095

Student ID: 201722800070

Supervisor:

Mingkui Tan

Grade:

Graduate

December 29, 2017

Abstract

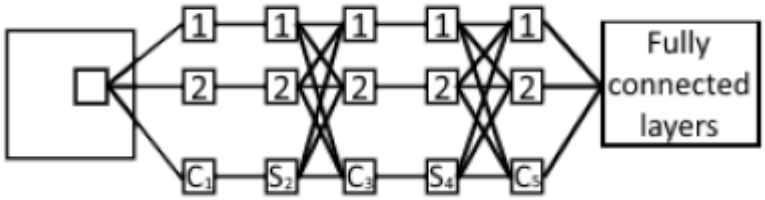
An artificial neural network is powerful tool in different domains . Over at present the deep neural network is the hottest topic in the domain of machine learning and can accomplish a deep hierarchical representation of the input data. Due to deep architecture the large convolutional neural networks can reach very small test error rates be-low 0.4% using the MNIST database. In this work we have shown, that high accuracy can be achieved using reduced shallow convolutional neural network without adding distortions for digits we have in This experiment uses the MNIST data-set. The data-set contains 70,000 hand-written digital pictures (training set 60000, verification set 10000), a size of 28 * 28 pixels. Data set download, training set, validation set of operations such as reading can be done using pytorch.. The main contribution of this paper is to point out how using simplified convolutional neural network is to obtain test error rate 0.71% on the MNIST handwritten digit benchmark. It permits to reduce computational resources in order to model convolutional neural network.

Introduction:

last decade the machine learning techniques has the leading role in domain of artificial intelligence. This is confirmed by recent qualitative achievements in images, video, speech recognition, natural language processing, big data processing and visualization, etc.. These achievements are primarily associated with new paradigm in machine learning, namely deep neural networks and deep learning . However in many real world applications the important problem is limited computational resources, which does not permit to use deep neural networks. Therefore the further development of shallow architecture is an important task. It should be noted especially that for many real applications, the shallow architecture can show the comparable accuracy in comparison with deep neural networks. This paper deals with a convolutional neural network for handwritten digits classification. We propose a simplified architecture of convolutional neural net-works, which permits to classify handwritten digits with more precision than a conventional convolution neural network LeNet -5.

A simplest convolutional neural network can be obtained the better classification results. introduces the standard convolution neural networks. Convolutional neural network is a further development of a multilayer perceptron and neocognitron and is widely used for image processing. This kind of neural network is invariant to shifts and distortions of the

input. Convolutional neural network integrates three approaches, namely local receptive field, shared weights and spatial sub-sampling. Using local receptive areas the neural units of the worst convolutional layer can extract primitive features such as edges, corners etc. The general structure of convolutional neural network is shown in the Fig.



A convolutional layer consists of set of a feature maps and the neural units of each map contain the same set of weights and thresholds. As a result each neuron in a feature map performs the same operations on different parts of image. The sliding windows technique is used for image scanning. Therefore if size of window is pxp (receptive field) then each unit in a convolutional layer is connected with p2 units of the corresponding receptive field. Each receptive field in input space is mapped into special neuron in each feature map. Then if the stride of sliding window is one that the numbers of neurons in each feature map is given by

$$D(C_1) = (n - p + 1)(n - p + 1) \quad (1)$$

where nxn is size of image. If the stride of sliding window is S that the numbers of neurons in each feature map is defined by the following way:

$$D(C_1) = (\frac{n - p}{s} + 1)(\frac{n - p}{s} + 1) \quad (2)$$

Accordingly, the common number of synaptic weights in convolutional layer is defined by

$$V(C_1) = M(p^2 + 1) \quad (3)$$

where M – the number of feature maps in convolutional layer. Let's represent the pixels of the input image in one-dimensional space. Then the ij-th output unit for k-th feature map in convolutional layer is given by

$$y_{ij}^k = F(S_{ij}^k) \quad (4)$$

$$S_{ij}^k = \sum_c w_{cij}^k x_c - T_{ij}^k \quad (5)$$

where c = 1, p2, F { the activation function, Skij | the weighted sum of the ij-th unit in k-th feature map, wkij | the weight from the c-th unit of the input layer to the ij-th unit of the k-th feature map, Tkij { the threshold of the ij-th unit of the k-th feature map.As already said the neural units of each feature map contain the same set of weights and thresholds. As a result the multiple features can be extracted at the same location. These features are then combined by the higher layer using pooling in order to reduce the resolution of feature maps . This layer is called sub sampling or pooling layer and performs a local averaging or maximization different regions of image. To this end, each map of convolutional layer is divided into non-overlapping areas with size of kxk and each area is mapped into

one unit of corresponding map in pooling layer. It should be noted that each map of convolutional layer is connected only with corresponding map in pooling layer. Each unit of pooling layer computes the average or maximum of k2 neurons in a convolutional layer:

$$z_j = \frac{1}{k^2} \sum_{j=1}^{k^2} y_j \qquad (6)$$
$$z_j = max(y_j)$$

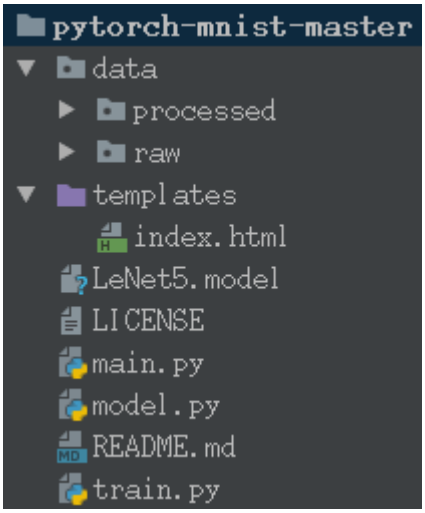
The number the of neurons in each pooling map is given by

$$D(S_2) = \frac{D(C_1)}{k^2} \qquad (7)$$

The number of feature maps in pooling layer will be the same like in convolutional layer and equal M. Thus convolutional neural network represents combination of convolutional and pooling layers, which perform nonlinear hierarchical transformation of input data space. The last block of the convolutional neural network is a multilayer perceptron, SVM or other classifier. Lets consider the conventional convolutional neural network (LeNet-5) for handwritten digits classification . The input image has size 32x32. The sliding window with size 5x5 scans the image and the segments of images enter to the layer C1 of neural network. Layer C1 is a convolution layer with 6 feature maps and each feature map contains 28x28 neurons. Layer S2 is a sub sampling layer with 6 feature maps and a 2x2 kernel for each feature map. As a result each feature map of this layer contains 14x14 units. Layer C3 is

Methods and Theory:

We use model.py to create the module LeNet5 which is inherits from nn.Module, the module LeNet5 is to train the network. We use train.py to load the module LeNet5 and generate a instance “LeNet5.model” to work. After that, we use main.py to load “LeNet5.model” and use it to predict.



In train.py, we have done:

2.1 Load data using torch.utils.data.Data Loader, torchvision.datasets.MNIST and torchvision.transforms.

2.2 instantiation torch.optim.SGD optimizer optimizer.

2.3 instantiation LeNet5 to get network net.

2.4 put the size of the batch size data into the network for forward propagation predict target.

2.5 Use torch.nn.CrossEntropyLoss to calculate the loss of predict target and ground truth target.

2.6 Calculate the gradient using loss.backward and optimize with optimizer.step.

2.7 Calculation of recognition accuracy.

2.8 Repeat steps 2.4-2.7 until all data is placed on the network.

2.9 The number of selected training epoch, repeat the training process 2.5--2.9

2.10 Save the best model as a .pkl file

Experiment:

The code is in the .tar.gz file.

When I used python train.py, it works as follows, because my python is 3.5, but the operation f"" is used only in 3.6+,so I move the f, If you want to see it, using the .format to see in print().

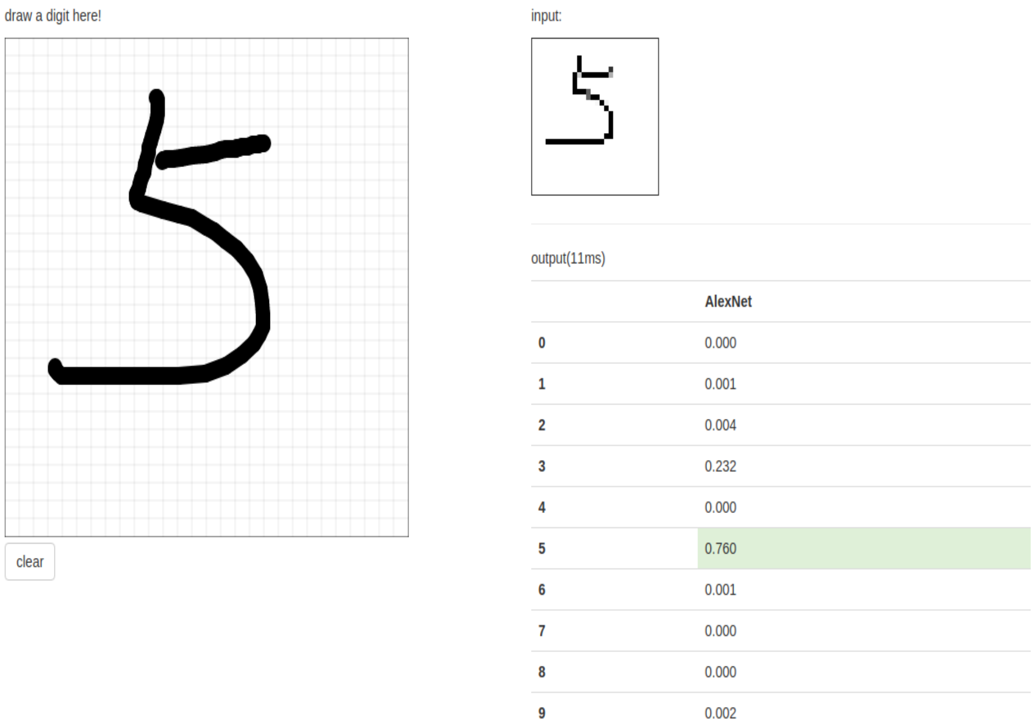
```
(py3.5Torch) lei@lei-HP-Zhan-86-Pro-G1-MT:~/桌面/pytorch-mnist-master$ python /home/lei/桌面/pytorch-mnist-master/trai
n.py
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Processing...
Done!
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TEST  PHASE epoch={epoch:03d} acc={correct/len(test_loader.dataset)*100:.2f}% loss={loss.data[0]:.4f}

TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TEST  PHASE epoch={epoch:03d} acc={correct/len(test_loader.dataset)*100:.2f}% loss={loss.data[0]:.4f}

TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TRAIN PHASE epoch={epoch:03d} iter={batch_index:03d} loss={loss.data[0]:.4f}
TEST  PHASE epoch={epoch:03d} acc={correct/len(test_loader.dataset)*100:.2f}% loss={loss.data[0]:.4f}

done.
```

Here is the work it does, but it doesn't perform very well.



Conclusion:

This experiment achieved handwritten numeral recognition through the framework of Pytorch, but the test result is not good, the program is poor in robustness. Perhaps add a few hidden layers or the epoch tune larger, the effect will be better, but also prone to over-fitting. The MNIST data-set has only 70,000 samples, which is too small for analysis. Convolutional neural network for handwritten digits classification. We performed a simplified architecture of convolutional neural net-works, which permits to classify handwritten digits with more precision than a conventional convolution neural network LeNet-5. The main differences from the conventional LeNet-5 are the following: we removed two last layers in LeNet-5; the layers S2 and C3 are fully connected; the sigmoid transfer function is used in all convolutional and output layers. We have shown that simple neural network is capable of achieving test error rate 0.71% on the MNIST hand written digits classification.