Spring 2019 Energy-efficient Machine Learning System
Final Project Report

Team Members: Yiyuan Fu (yf199), Song Yang(sy540)

Project Topic: Manually implement a LeNet-5 convolutional NN using NumPy (no automatic gradient). Train and test on MNIST dataset. The high test accuracy should be achieved.

Abstract

LeNet-5 is a classic CNN architecture. This neural network is originally proposed by Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner for handwritten and machine-printed character recognition in 1990's. The architecture is straightforward and simple to understand that's why it is mostly used as a first step for learning.

We implement the LeNet-5 with NumPy to gain a deeper understanding of neural network.

Introduction

The LeNet-5 architecture was first introduced by LeCun et al. in their 1998 paper, Gradient-Based Learning Applied to Document Recognition. As the name of the paper suggests, the authors' implementation of LeNet was used primarily for OCR and character recognition in documents.

The LeNet-5 architecture is straightforward and small, making it perfect for teaching the basics of CNNs — it can even run on the CPU, making it a great "first CNN".

We will write LeNet-5 in Numpy and using LeNet-5 that we written by numpy to the MNIST dataset to classify the handwriiten digit 0-9. The accuarcy of our LeNet will be test.

Related work

There are many deep learning library like Tensorflow, pythorch ,or Theano, and use the prebuilt module to rapidly build the model.

For example, in Keras We can Create a new instance of a model object using sequential model API. Then add layers to the neural network as per LeNet-5 architecture. Finally, compile the model with the 'categorical_crossentropy' loss function and 'SGD' cost optimization algorithm. When compiling the model, add metrics=['accuracy'] as one of the parameters to calculate the accuracy of the model.

However, to understand the convnet better, we implement LeNet-5 only using Numpy!

Data description

MNIST is an abbreviation of Modified National Institute of Standards and Technology, which is a large database of handwritten digits. The MNIST database contains 60,000 training images and 10,000 testing images and is commonly used for training various image processing systems.

Each digit is represented as a 28 x 28 grayscale image (examples from the MNIST dataset can be seen in the figure above). These grayscale pixel intensities are unsigned integers, with the values of the pixels falling in the range [0, 255]. All digits are placed on a black background with a light foreground (

The MNIST data is also used in this project. One paper, using a hierarchical system of convolutional neural networks, manages to get an error rate of 0.23% [2].

Method description

As we've already known, every layer in a neural net consists of forward and backward computation, because of the backpropagation. Conv layer is no different, as essentially, it's just another neural net layer.

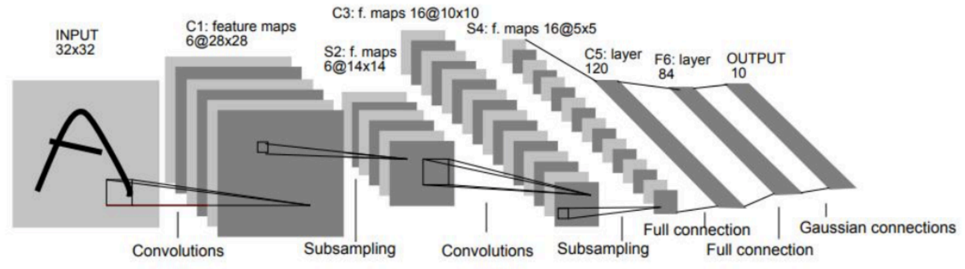
MNIST data is set as training data and test data, In training data we set 30% as validation data.

When in training LeNet-5, we have ***feedforward*** function to implement feed-ward and ***backpropagation*** function to compute feed forward layer's backward. After backpropagation we use ***update_parameters*** function to update parameters. The ***loss_function*** is used to compute loss using cross-entropy method. In each batch, ***feedforward-> loss_function-> backpropagation-> update_parameters*** repeat to all batches in all epochs. And then our training is complete.

When we test our model, we use ***feedforward*** function to our test data and ***accuracy_score*** function to get the accuracy.

Model description

The network model is a classic LeNet-5 neural network. It contains a convolutional Relu layer, a max pool layer, a convolution Relu layer, a max pool layer, two full connected Sigmoid layers, and a full connected SoftMax layer.



First Layer:

The input for LeNet-5 is a 32×32 grayscale image which passes through the first convolutional layer with 6 feature maps or filters having size 5×5 and a stride of one. The image dimensions changes from 32x32x1 to 28x28x6.

Second Layer:

Then the LeNet-5 applies max pooling laye with a filter size 2×2 and a stride of two. The resulting image dimensions will be reduced to 14x14x6.

Third Layer:

Next, there is a second convolutional layer with 16 feature maps having size 5×5 and a stride of 1. In this layer, only 10 out of 16 feature maps are connected to 6 feature maps of the previous layer

Fourth Layer:

The fourth layer (S4) is again an max pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer (S2) except it has 16 feature maps so the output will be reduced to 5x5x16.

Fifth Layer:

The fifth layer (C5) is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the 120 units in C5 is connected to all the 400 nodes (5x5x16) in the fourth layer S4.

Sixth Layer:

The sixth layer is a fully connected layer (F6) with 84 units.

Output Layer:

Finally, there is a fully connected softmax output layer ŷ with 10 possible values corresponding to the digits from 0 to 9.

Experimental procedure

The files: conv.py, maxpool.py, relu.py, fc.py, sigmoid.py, softmax.py are related to convolution layer, maxpooling layer, relu activation layer, fully connected layer, sigmoid activation layer and softmax layer. In each file we have forward and backward function to compute each layer's feedforward and backpropogation.In fully connection layer we have adagrad, gradient descent and adam as optimizer in the backpropogation.

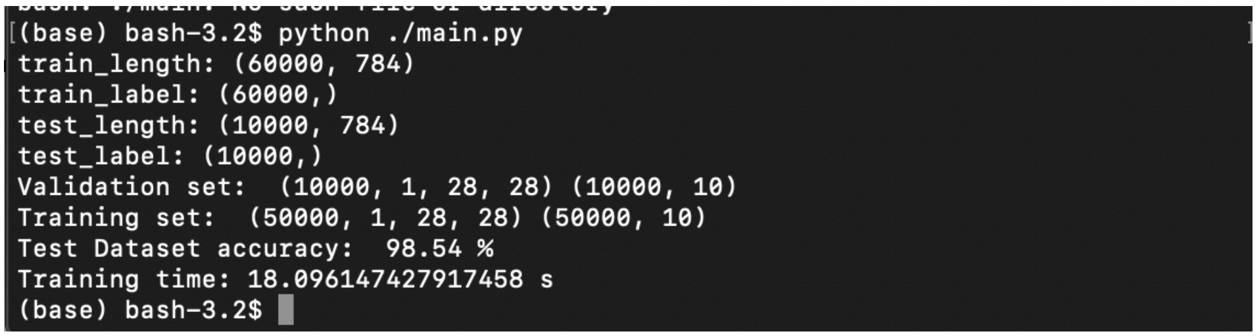
The Lenet5.py file set the architecture of LeNet-5 and the whole LeNet-5's feedforward and backpropogation. ***lenet_train*** function is to train the net and ***lenet_predictions*** function is to test our net.

The main.py file is first load data, we separate 60000 training dataset in to 2 part: 50000 for training an10000 for validation. Then crate and train Lenet5 and last get test set accuracy.

Our batch size is as50, using adam optimizer in fully connection layer and convolution layer , we set 4 epoches and the learinig rate is 0.01.

Results

With the batch size of 16, the learning rate as 0.01, the epoch as 4, We get the following result.



```
[(base) bash-3.2$ python ./main.py
 train_length: (60000, 784)
 train_label: (60000,)
 test_length: (10000, 784)
 test_label: (10000,)
 Validation set:  (10000, 1, 28, 28) (10000, 10)
 Training set:  (50000, 1, 28, 28) (50000, 10)
 Test Dataset accuracy:  98.54 %
 Training time: 18.096147427917458 s
 (base) bash-3.2$
```

Conclusion

We use Numpy to implement LecNet-5 which is a classic example of convolutional neural network to sucessfully predict handwriiten digits. It contains convolution layer, relu layer, max pooling layer, fully connected layer, sigmoid layer and sofmax layer. We compute feed forward and back forward in each layer and then update our parameters.

The accuracy of test data in our Lenet-5 is 98.54%, which is pretty good.

Reference

[1] NumPy User Guide https://www.numpy.org/devdocs/user/index.html
[2] Cires¸an, Dan; Ueli Meier; Jürgen Schmidhuber (2012). Multi-column deep neural networks for image classification(PDF). 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp.