Kaggle - Digit Recognizer

Competition link: https://www.kaggle.com/c/digit-recognizer/

Dataset: MNIST (Mixed National Institute of Standards and Technology database)

The goal in this competition is to take an image of a handwritten single digit, and determine what that digit is.

Our best submission

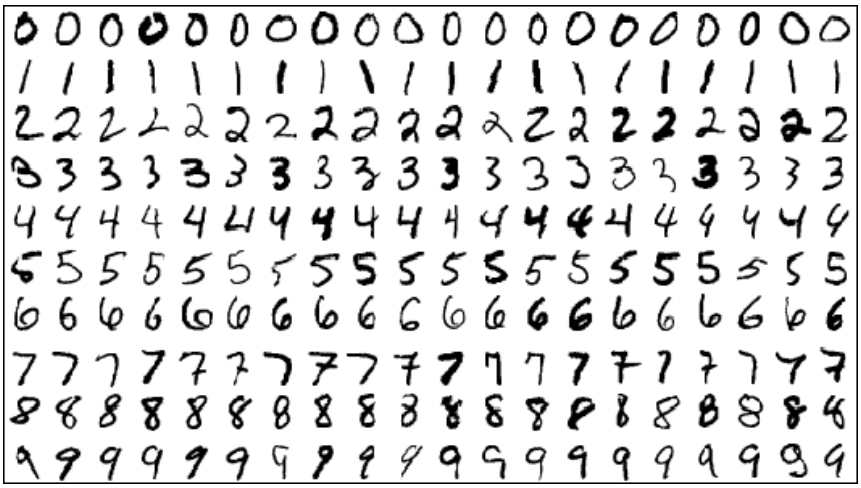
Leaderboard: 65th / 962 total

Error rate: 0.757%

63	↓5	andrei	0.99257	16	Wed, 09 Dec 2015 09:31:23 (-7.5h)
64	↓5	indy256	0.99257	18	Sun, 03 Jan 2016 07:46:36
65	new	**NTUST 1041 ML - Group 8**	**0.99243**	**10**	**Tue, 12 Jan 2016 10:00:52**
66	↓6	pklfz	0.99229	5	Wed, 09 Dec 2015 14:28:48
67	↓6	wanglingdeemo	0.99229	20	Mon, 14 Dec 2015 02:45:41

Our project source code and submission data:

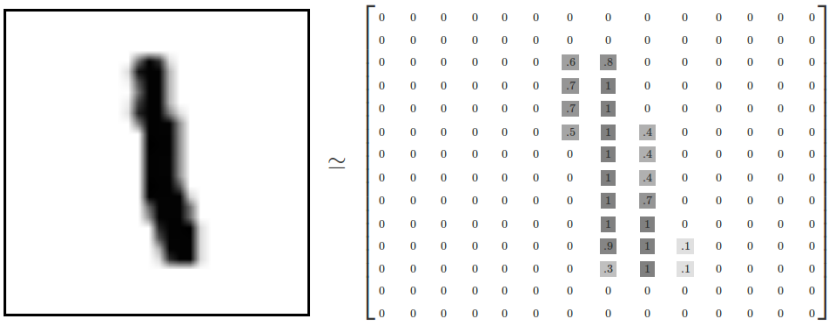
About MNIST Dataset



Pic-1. MNIST Examples. (source: myselph.de)

Each image is 28 * 28 pixels. We can interpret this image as a big array of numbers like Pic-1. We can flatten this array into 28 * 28 = 784 numbers, this MNIST images are a bunch of points in a 784-dimensional vector space.

The data files `train.csv` and `test.csv` contain gray-scale images of hand-drawn digits, from zero through nine. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.



Pic-2. MNIST matrix. (source: tensorflow.org)

The training data set, `train.csv` , has 785 columns. The first column, called **"label"**, is the digit that was drawn by the user. The rest of the

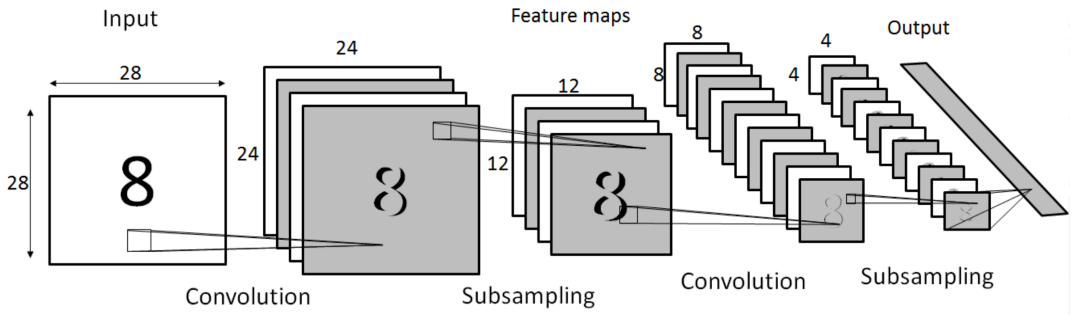
columns contain the pixel-values of the associated image.

The test data set, `test.csv`, is the same as the training set, except that it does not contain the **"label"** column.

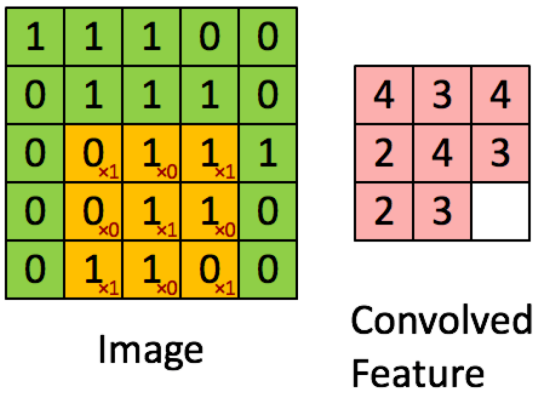
Related work

Type	Classifier	Distortion	Preprocessing	Error rate (%)
Linear classifier	Pairwise linear classifier	None	Deskewing	7.6[9]
K-Nearest Neighbors	K-NN with non-linear deformation (P2DHMDM)	None	Shiftable edges	0.52[14]
Boosted Stumps	Product of stumps on Haar features	None	Haar features	0.87[15]
Non-Linear Classifier	40 PCA + quadratic classifier	None	None	3.3[9]
Support vector machine	Virtual SVM, deg-9 poly, 2-pixel jittered	None	Deskewing	0.56[16]
Neural network	2-layer 784-800-10	None	None	1.6[17]
Neural network	2-layer 784-800-10	elastic distortions	None	0.7[17]
Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	elastic distortions	None	0.35[18]
Convolutional neural network	Committee of 35 conv. net, 1-20-P-40-P-150-10	elastic distortions	Width normalizations	0.23[8]

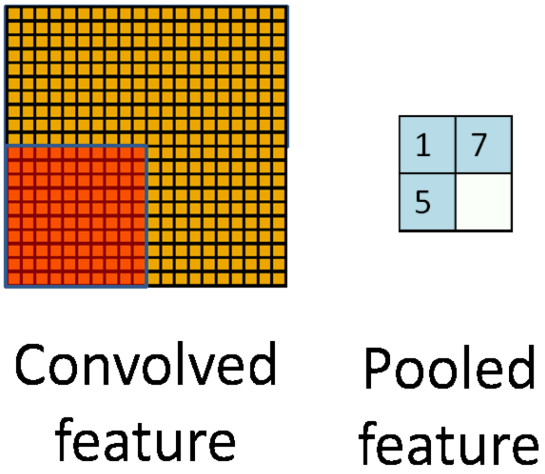
Pic-. Classifiers performence in MNIST. (source: wikipedia.org)



Pic-. LeNet workflow. (source: developer.nvidia.com)



Pic-. Convolutional schematic. (source: deeplearning.stanford.edu)



Pic-. Pooling schematic. (source: deeplearning.stanford.edu)

Experiment

Environment: **Mac OS X** El Capitan 10.11.1

Language: **R** version 3.2.3

Toolkit:

- MXNet 0.5.0 (http://mxnet.rtfd.org)

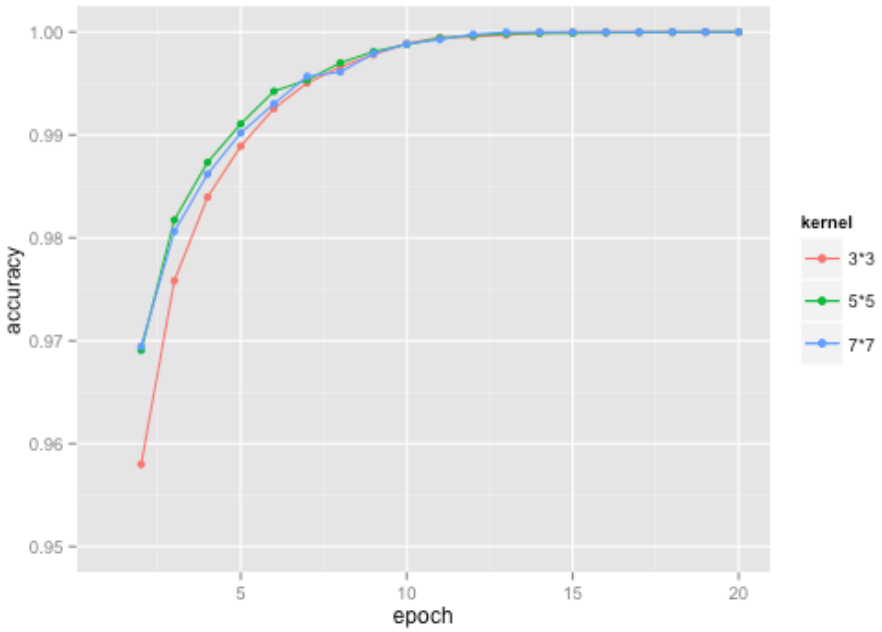
  MXNet is a deep learning framework designed for both efficiency

and flexibility. It allows you to mix the flavours of deep learning programs together to maximize the efficiency and your productivity.

Filter Shape

At first, we try to tune the convolution kernel size. Size of 3x3 is the most ofter used. But in this case, size of 5x5 is better than others in the MNIST data. [1]

- The same conditions: activation function (tanh)



Pic-. Compare different kernel size.

When kernel = 3x3 matrix, **Kaggle submission scored 0.99029**.

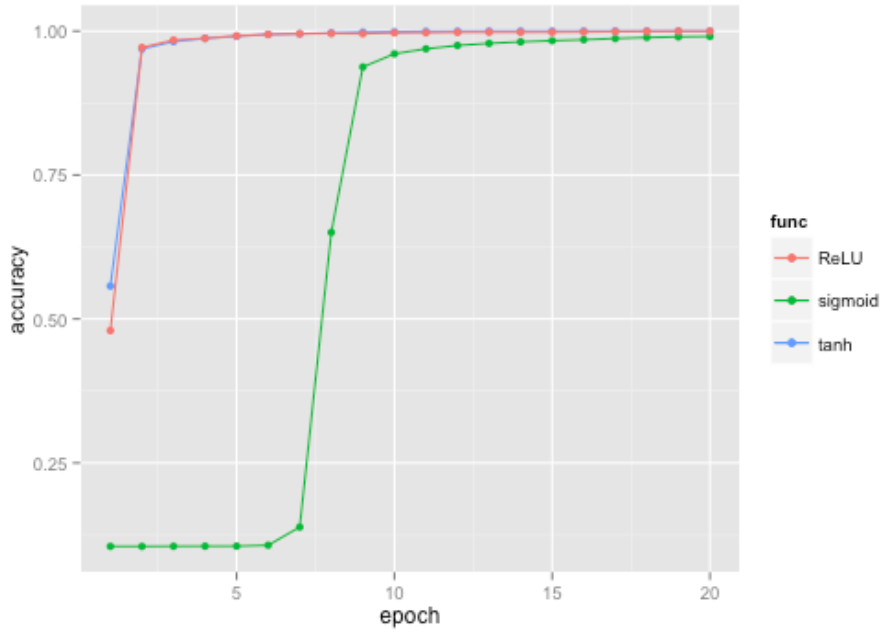
When kernel = 5x5 matrix, **Kaggle submission scored 0.99071 (Better)**.

When kernel = 7x7 matrix, **Kaggle submission scored 0.98886**.

Activation functions

Every activation function (or non-linearity) takes a single number and performs a certain fixed mathematical operation on it. [5] In this step, we try to chagne the activation function to get different outputs. Specifically, ReLU computes the function `f(x) = max(0,x)`. In other words, the activation is simply thresholded at zero. [5]

- The same conditions: convolution kernel size (5x5)



Pic-. Compare different activation function.

When using tanh function, **Kaggle submission scored 0.99071**.

When using sigmoid function, **Kaggle submission scored 0.98500**.

When using ReLU function, **Kaggle submission scored 0.99171 (Better)**

But using 20 epochs still not reaches 1.0 accuracy in ReLU function, we increase the epoch to 30 rounds. Finally, we get **Kaggle**

submission scored 0.99243 (Best).

Max Pooling Shape

Typical values are 2x2 or no max-pooling. [1] But we can't use 1x1 to do our pooling kernel, it costs too large time on iterating.

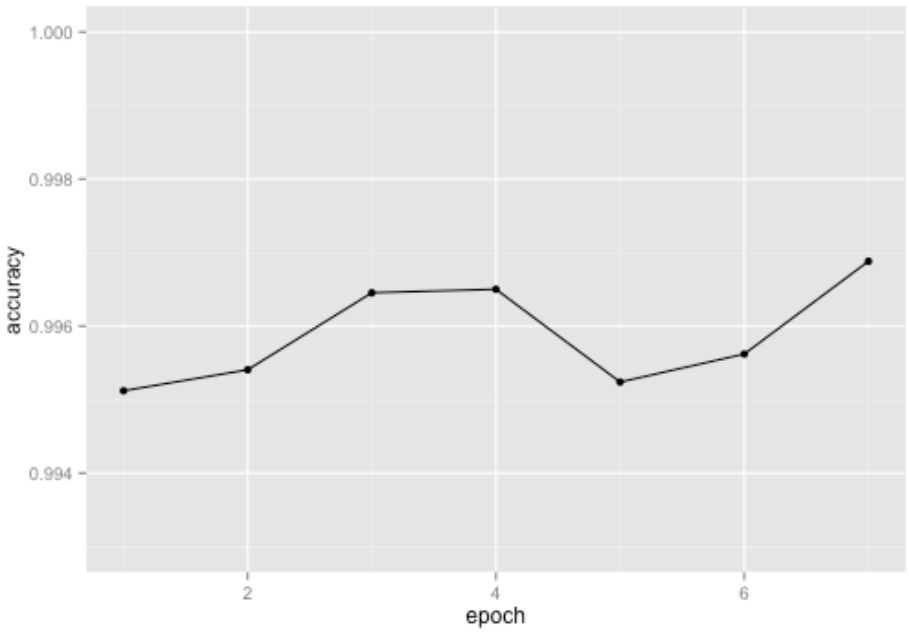
Conclusion

In this final project, we try to use the CNN algorithm (Convolutional Neural Network) that didn't learn in class. The image recognition domain. Researchers optimize the algorithms about image recognition in the past years. There are exist more and more packages, libraries and framework to apply on Machine Learning.

Overfitting

About overfitting problem. Below output is one case in overfitting, we can see the results not all get a better model by increasing epochs. Epoch that 14 to 17 increases, but then decreases.

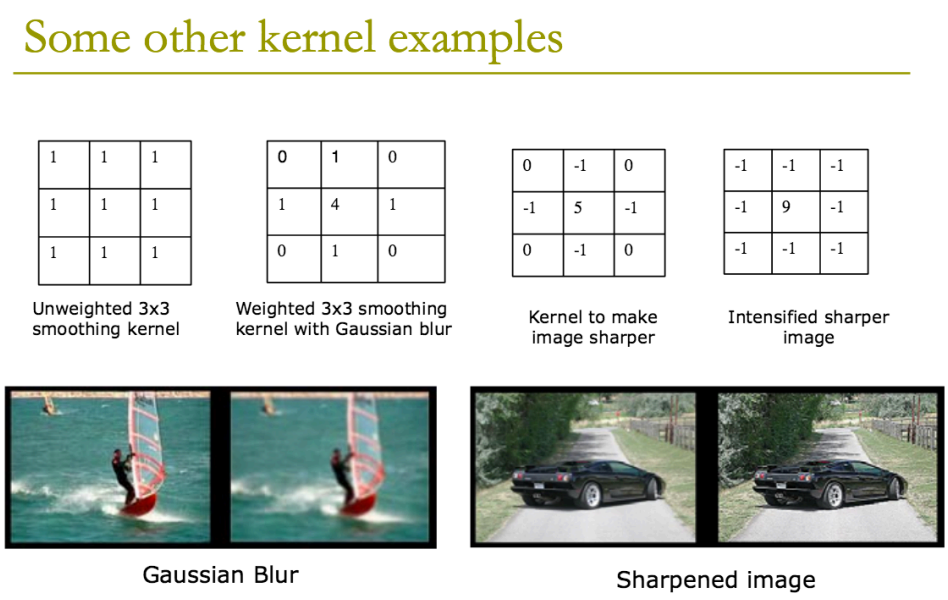
```
[14] Train-accuracy=0.995119047619049
[15] Train-accuracy=0.995404761904763
[16] Train-accuracy=0.996452380952383
[17] Train-accuracy=0.996500000000002
[18] Train-accuracy=0.995238095238097
[19] Train-accuracy=0.995619047619049
[20] Train-accuracy=0.996880952380954
```



Pic-. Overfitting problem.

Smoothing kernel and others

Look the image below. We can see some different desired results for images that based on choosed kernel (smooth, blur, enhance or sharpen). [7] Regrettably, **MXNet** doesn't support assign the kernel by user in current version.



Pic-. Smoothing kernel.

Speed up on computing

Computing is too slow in this project. Average computing time around 4500 - 5000 seconds. We can try to use GPU computing to speed up.

References

[1] Convolutional Neural Networks (LeNet), 2015/12, http://deeplearning.net/tutorial/lenet.html

[2] *Andrew Gibiansky*, Convolutional Neural Networks, 2015/12, http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/

[3] *Michael Nielsen*, Deep learning, 2015/12, http://neuralnetworksanddeeplearning.com/chap6.html

[4] Feature extraction using convolution, 2015/12, http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_u

sing_convolution

[5] Convolutional Neural Networks for Visual Recognition, 2016/01, http://cs231n.github.io/neural-networks-1/

[6] *Chen-Yu Lee, Patrick W. Gallagher, Zhuowen Tu*, Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree. 2015.

[7] *Jamie Ludwig*, Image Convolution, 2016/01 http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf

Special Thanks

Kai-Hsiang Hsu, NTU. Thanks for giving us information about the better algorithms in image recognition.