# MNIST handwritten digit recognition system

A Preprint

**Yong Hu**
Department of Computer Science
Beijing Institute of Technology
Beijing, China

**Tan Yan**
Department of Computer Science
Beijing Institute of Technology
Beijing, China

**YaNan Wang**
Department of Computer Science
Beijing Institute of Technology
Beijing, China

**Dan Ma**
Department of Computer Science
Beijing Institute of Technology
Beijing, China

January 20, 2019

## Abstract

The task aims to implement a handwritten digit recognition system based on artificial neural network. The system consists of four sections. In the first section,we propose two kinds of neural network with supervised learning. In the second section, we trained an Autoencoder to improve the performance of neural network used in section one through semi-supervised learning. In the third section, we use Neural Architecture Search (NAS) with Reinforcement Learning to find optimal to improve upon the brute force method the goal of achieving optimal hyperparameters in a fraction the time of a brute force search. In the fourth section, we use evolutionary computation to further improve the neural network, the optimized network object is a fully connected network. Our code is open source on GitHub, the address is https://github.com/nghuyong/MinistRecognition

## 1 Introduction

Firstly the handwritten digit recognition system we implemented identified the handwritten digits using a fully connected network and a convolutional neural network(CNN) combined with a fully connected network. On the basis of this, a semi-supervised learning method is applied to realize the recognition of handwritten digits on a training data set with only a small number of tags. Then, using the methods of reinforcement learning and evolutionary computation to search the optimal network structure for the previously applied neural network, and improve the performance of the recognition neural network.

## 2 Dataset

The MNIST data set contains a training set of 55,000 samples, a validation set of 5000 samples, and a test set of 10,000 samples. Input-data.py has decompressed the downloaded data set, reconstructed the image and tag data to form a new data set object. The image is a grayscale image of 28 pixels x 28 pixels. The blank parts are all 0, and the places with handwriting have a value of 0 1 according to the color depth. Therefore, each sample has a feature of 28*28=784 dimensions, which is equivalent to expanding to 1 dimension. The training set is characterized by a 55000*784 Tensor, the first latitude is the picture number, and the second dimension is the image pixel point number. The Label of the training set (the picture represents which number from 0 to 9) is a 55000*10 Tensor, and 10 is the meaning of 10 types. For one-hot encoding, only one value is 1, and the rest is 0, such as the number 0, for label is [1,0,0,0,0,0,0,0,0,0].

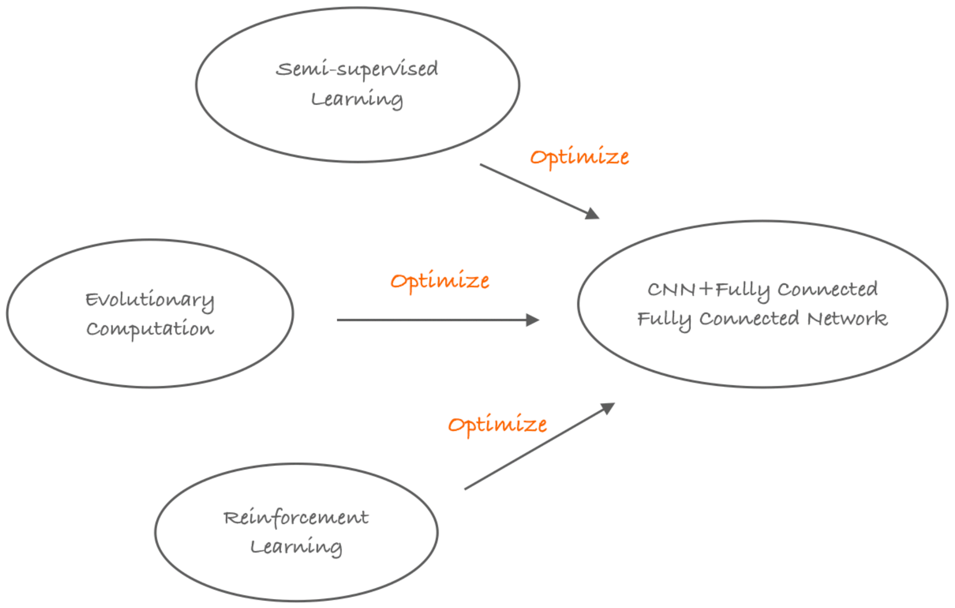

Figure 1: Overall of system model

## 3 Evaluation

For each question in the test set, the model is required to predict as much as five relevant tags, and the tags are sorted by their predicted probabilities. Specifically, the number of predicted tags for a given question can be less than 5 or even be 0 if the model can’t find enough relevant tags to the question.

The results are evaluated on the $F_1$ measure.We compute the positional weighted precision.Let $correct\_num_{p_i}$ denotes the correct count of predicted tags at position $i$, and $predict\_num_{p_i}$ denotes the count of predicted tags at position $i$.The precision, recall and $F_1$ measure are computed as following formulas:

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{1}$$

$$P = \frac{\sum_{i=1}^{5} correct\_num_{p_i}/log(i+2)}{\sum_{i=1}^{5} predict\_num_{p_i}/log(i+2)} \tag{2}$$

$$P = \frac{\sum_{i=1}^{5} correct\_num_{p_i}}{ground\_truth\_num} \tag{3}$$

## 4 Model

### 4.1 Supervised Learning

#### CNN

CNN(Convolutional Neural Networks) is a classical model for document recognition, which is proposed by Yann LeCun and is employed to perform hand-written recognition and image classification.This CNN-Encoder consists of two convolutional layers and a fully connected layer. We first feed our input sentence into convolutional layer to extract features of input sentence, the advantage of stacked convolutional layer is that we can learn more abstract relationship of features.After convolutional layer, we pass the output of convolutional layer to max-pooling layer and finally get the top k tag of this question.

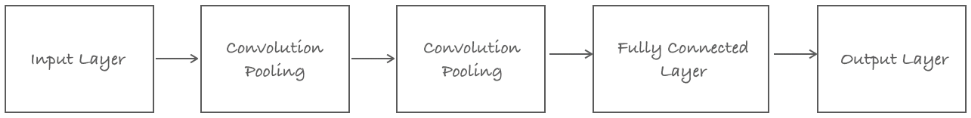

Figure 2: Structure schematic diagrams of CNN+FC

1. Input layer: First, each handwritten digital picture to be recognized is processed. We divide the picture into 28*28 size, totaling 784 pixels. These 784 features will enter the input layer as input, the input layer should contain 784 neurons.
2. Initialization weight and offset term: In the process of weight initialization, a small amount of noise needs to be added to break the symmetry while avoiding the gradient disappearing. We set the standard deviation of the weights to b 0.1. We use the ReLU activation function. To avoid the constant zero output of the neuron node, we apply a smaller positive number to initialize the bias term.
3. Convolution and pooling: We convolve using a method with a step size of 1. and the same fill. First, in each 5*5 grid, 32 feature maps are extracted. The number of input channels is 1, and the number of output channels is 32. For each output channel, an offset term is added, so there are 32 offset items. The first convolution is then performed using the ReLU activation function. After that, pooling is performed by max pooling using a 2*2 grid. The two-layer convolution is similar to the pooling process.
4. Fully connected layer: After convolution and pooling, the size of the image is 7*7. We add a fully connected layer with 1024 neurons here, and convert the pooled output into a one-dimensional vector. The weights are multiplied, plus the offset term, and finally the function is activated by ReLU function.
5. Output layer: The output layer should have 10 neurons, representing 10 numbers from 0 to 9.

Table 1: Comparison of different models.

| model | Accuracy |
|---|---|
| two layer full connected network | 97.546 |
| two layer cnn and a full connected layer | 99.219 |

**learning steps**

1. Softmax function: In the Logistic Regression two-class problem, the sigmoid function is used to map the input to the (0,1) interval, thereby obtaining the probability of belonging to a certain category. Then we implemented a handwritten digit recognition system involving 10 outputs. To generalize the two-category problem to the multi-class problem, we use the softmax function to normalize the output value to the probability value. It is assumed that before entering the softmax function, there is already a model output C , C represents the number of categories to be predicted, that is, the number of output layer neurons of the fully connected network designed above, C=10. The output is a1, a2,..., a10. Then calculate the probability of belonging to category i for each sample.
2. Cross entropy: Firstly, the relative entropy, also known as KL divergence, is a measure of the distance between two random distributions, denoted as DKL(p||q), and the cross entropy adds H(p) to reflect the distribution p. The degree of similarity between q.
3. The definition of cross entropy loss function: Define the cross entropy loss function by calculating the cross entropy between softmax(logits) and labels, where logits presents the category predictive output of the last layer of the neural network, and labels is the category label of the real data.
4. Optimization: The cross entropy loss function is optimized using the Adam method. The Adam optimization algorithm dynamically adjusts the learning rate of each parameter by using the first

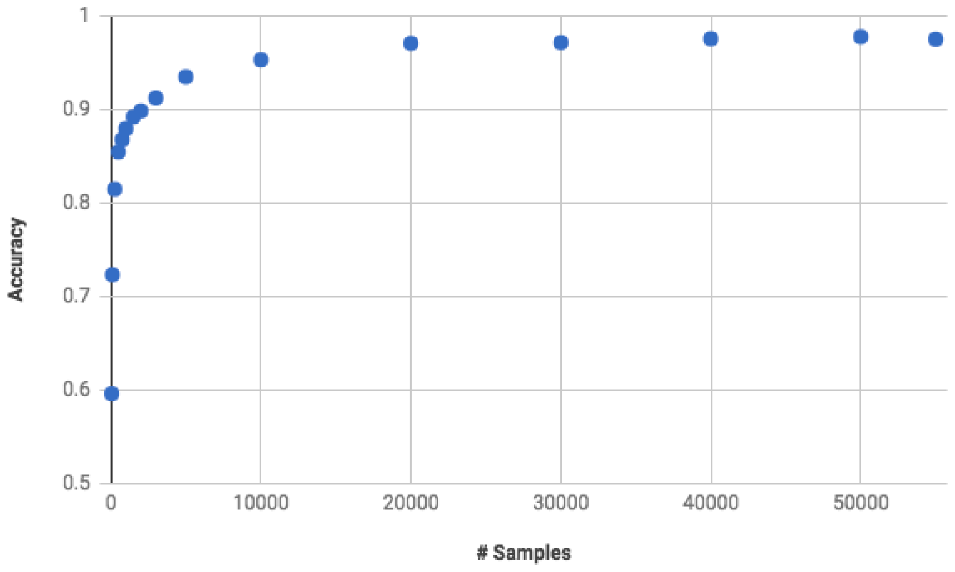

Figure 3: Accuracy-vs-samples

moment estimation and the second moment estimation of the gradient. The advantage of Adam is that after each iteration, the learning rate has a certain range, making the parameters more stable.

**Fully Connected Network**

We also trained a fully connected network classifier with a network structure consisting of an input layer, an intermediate layer, and an output layer. Similarly, the input layer 784 neurons represent 784 features (pixels) of the input image, the middle layer contains 100 neurons, and the output layer contains 10 neurons representing 10 numbers of 0-9. Its learning process is the same as CNN.

**Experiment**

We show the experimental accuracy of the two models we constructed in table 4. It can be seen that the accuracy of the model using two layers of cnn and a full connection layer has reached 99%. We use two layer full connected network to study accuracy in different training samples. You can see that 81% is achieved with only 250 samples and 90% with 2000. You need 27x the data to go from 90% to 97.5% with the simple network.

### 4.2 Semi-Supervised Learning

We only have 2000 labeled samples but have another 53000 unlabeled samples. So we use semi-supervised learning to help supervised learning.Let the learning process not rely on external consulting interactions, automatically use the distribution information contained in the unlabeled data samples.First we learn an AutoEncoder,that encoder the input image to a embedding and then decode the embedding to reconstruct the image.

Table 2: Comparison between supervised and semi-supervise model

| model | Accuracy |
|---|---|
| 2000 labeled samples,supervised | 88.030 |
| 2000 labeled samples,53000 unlabeled samples,semi-supervised | 92.176 |

1. AutoEncoder:The AutoEncoder consists of two processes, encode and decode. These two process functions are data-dependent, lossy, and automatically learn from the samples. The input image is calculated and processed by encode, and then decoded to obtain the output. The two processes

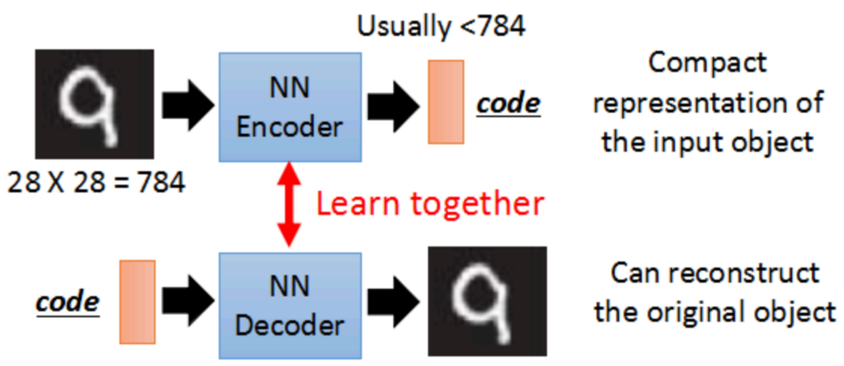
Usually <784
NN Encoder
code
Compact representation of the input object
28 X 28 = 784
Learn together
code
NN Decoder
Can reconstruct the original object

Figure 4: Sketch of AutoEncoder

of encode and decode can be understood as inverse functions. The encode process continues to reduce dimensions, and the decode process is improving the dimensions. The encode process is a deep convolutional network, multi-level convolution and pooling, then the decoding process requires corresponding deconvolution and unpooling.

2. The network structure includes an input layer, two convolutional layers, a fully connected layer, an embedded layer, two layers of fully connected layers, two layers of deconvolution layers, and an output layer in order.

**Experiment**

In figure 4 show a scatter plot of the 10,000 training samples from MNIST embedded in a 2-D space.We send the embedding(learned from autoencoder) of the image into a fully connected neural network. The result of experiment is shown in Table 5. After adding embedding the accuracy of fully connected neural network increases by 4%.

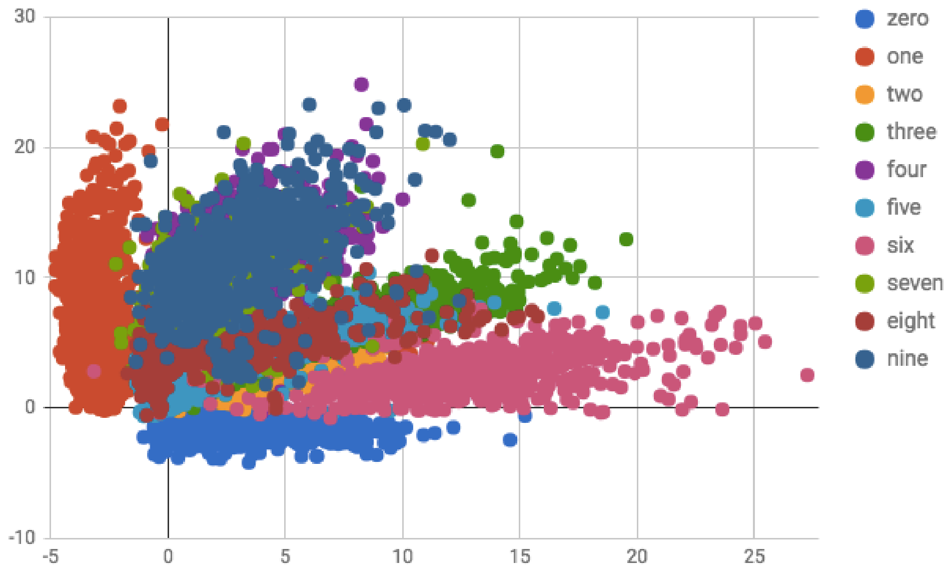

Figure 5: Scatter plot of the 10,000 training samples from MNIST embedded in a 2-D space

### 4.3 Reinforcement Learning

Neural Architecture Search (NAS) with Reinforcement Learning is a method for finding good neural networks architecture. For this part, we will try to find optimal architecture for Convolutional Neural Network (CNN) which recognizes handwritten digits. It is therefore possible to use a recurrent network –the controller –to generate such string. Training the network specified by the string –the "child network" –on the real data will result in an accuracy on a validation set. Using this accuracy as the reward signal, we can compute the policy gradient to update the controller. As a result, in the next iteration, the controller will give higher probabilities to architectures that receive high accuracies. In other words, the controller will learn to improve its search over time.

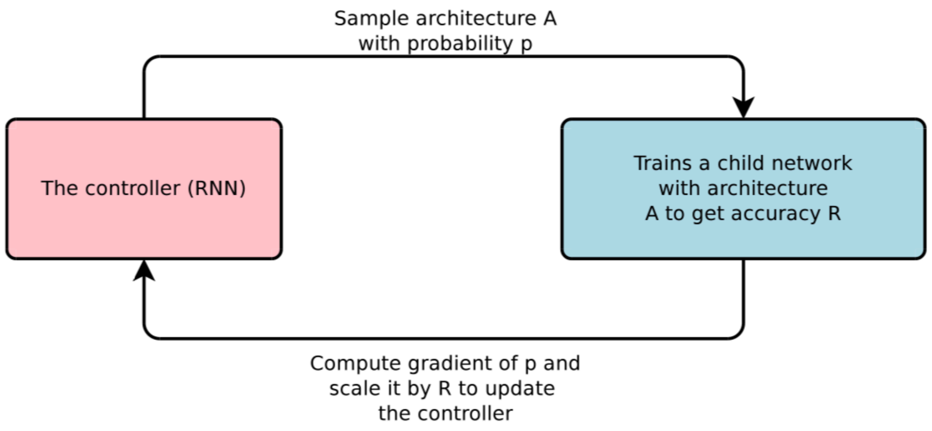

Figure 6: An overview of Neural Architecture Search

**Experiment**

Neural Architecture Search (NAS) with Reinforcement Learning is a method for finding good neural networks architecture. For this part, we will try to find optimal architecture for Convolutional Neural Network (CNN) which recognizes handwritten digits. For a signal layer network,we search the hidden layer size from 10 to 800 and step is 20.The result is showed in table 3. For a signal layer network,we search the two layers cnn parameters.The result is showed in table 4. And int table 5 we compared the optimized network after adding reinforcement learning with the unoptimized. From the table, we can see that the network accuracy has been improved significantly after optimization.

### 4.4 Evolutionary Computation

We try to improve upon the brute force method by applying a genetic algorithm to evolve a network with the goal of achieving optimal hyperparameters in a fraction the time of a brute force search. Let' s say it takes five minutes to train and evaluate a network on your dataset. And let' s say we have four parameters with five possible settings each. To try them all would take (5**4) * 5 minutes, or 3,125 minutes, or about 52 hours.

Now let' s say we use a genetic algorithm to evolve 10 generations with a population of 20 (more on what this means below), with a plan to keep the top 25 percentage plus a few more, so 8 per generation. This means that in our first generation we score 20 networks ($20 * 5 = 100$ minutes). Every generation after that only requires around 12 runs, since we don' t have the score the ones we keep. That' s 100 + (9 generations * 5 minutes * 12 networks) = 640 minutes, or 11 hours.We' ve just reduced our parameter tuning time by almost 80 percentage That is, assuming it finds the best parameters. Genetic Algorithms：

1. Creates a population of (randomly generated) members
2. Scores each member of the population based on some goal. This score is called a fitness function.
3. Selects and breeds the best members of the population to produce more like them
4. Mutates some members randomly to attempt to find even better candidates

Table 3: Neural Architecture Search in hidden layer by Reinforcement Learning

| HiddenSize | Accuracy | Reward | totalReward |
|---|---|---|---|
| 160 | 0.9781 | 0.1 | 0.1 |
| 450 | 0.9816 | 0.0035 | 0.1035 |
| 40 | 0.9687 | 0.0101 | 0.0934 |
| 90 | 0.9765 | -0.00028 | 0.09312 |
| 220 | 0.9797 | 0.00298 | 0.0961 |
| 30 | 0.9631 | -0.01422 | 0.08188 |
| 580 | 0.9817 | 0.00722 | 0.0891 |
| 470 | 0.982 | 0.00608 | 0.09518 |
| 410 | 0.9813 | 0.00416 | 0.09934 |
| 160 | 0.9779 | -7e-05 | 0.09928 |
| 710 | 0.983 | 0.00504 | 0.10432 |
| 420 | 0.9813 | 0.00234 | 0.10666 |
| 50 | 0.9689 | -0.01053 | 0.09613 |
| 610 | 0.9821 | 0.00477 | 0.1009 |
| 160 | 0.9798 | 0.00152 | 0.10242 |
| 280 | 0.9805 | 0.00192 | 0.10434 |
| 140 | 0.9795 | 0.00053 | 0.10487 |
| 740 | 0.9837 | 0.00463 | 0.1095 |
| 380 | 0.9814 | 0.0014 | 0.1109 |
| 360 | 0.9816 | 0.00132 | 0.11222 |
| 530 | 0.9831 | 0.00256 | 0.11477 |
| 160 | 0.9804 | -0.00065 | 0.11412 |
| 100 | 0.9769 | -0.00402 | 0.1101 |
| 660 | 0.9829 | 0.00278 | 0.11288 |
| 600 | 0.9824 | 0.00172 | 0.1146 |
| 60 | 0.9747 | -0.00632 | 0.10828 |
| 230 | 0.9798 | 4e-05 | 0.10832 |
| 190 | 0.9809 | 0.00114 | 0.10946 |

Table 4: Neural Architecture Search in two layers CNNs by Reinforcement Learning

| CNN-1-kernel | CNN-1-kernel | CNN-2-kernel | CNN-2-kernel | Accuracy | Reward | totalReward |
|---|---|---|---|---|---|---|
| 23 | 32 | 2 | 16 | 0.9904 | 0.1 | 0.1 |
| 3 | 64 | 5 | 64 | 0.9929 | 0.0025 | 0.1025 |
| 5 | 64 | 3 | 32 | 0.9923 | 0.0014 | 0.1039 |
| 5 | 64 | 4 | 32 | 0.9923 | 0.00112 | 0.10502 |
| 3 | 64 | 4 | 16 | 0.9917 | 0.0003 | 0.10532 |
| 4 | 64 | 5 | 16 | 0.9915 | 4e-05 | 0.10535 |

Table 5: Neural Architecture Search Overview by Reinforcement Learning

| Model | Accuracy |
|---|---|
| Supervised learning fc | 97.546 |
| Using RL,Supervised learning fc | 98.300 |
| Supervised learning cnn fc | 99.219 |
| Using RL,Supervised learning cnn fc | 99.240 |

5. Kills off the rest (survival of the fittest and all), and

6. Repeats from step 2. Each iteration through these steps is called a generation.

7. Repeat this process enough times and you should be left with the very best possible members of a population. Sounds like a lot evolution, right? Same deal.

Applying genetic algorithms to Neural Networks: We' ll attempt to evolve a fully connected network (MLP). Our goal is to find the best parameters for an image classification task. We' ll tune four parameters:

- Number of layers (or the network depth)
- Neurons per layer (or the network width)
- Dense layer activation function
- Network optimizer

The steps we' ll take to evolve the network, similar to those described above, are:

1. Initialize N random networks to create our population.
2. Score each network. This takes some time: We have to train the weights of each network and then see how well it performs at classifying the test set. Since this will be an image classification task, we' ll use classification accuracy as our fitness function.
3. Sort all the networks in our population by score (accuracy). We' ll keep some percentage of the top networks to become part of the next generation and to breed children.
4. We' ll also randomly keep a few of the non-top networks. This helps find potentially lucky combinations between worse-performers and top performers, and also helps keep us from getting stuck in a local maximum.
5. Now that we' ve decided which networks to keep, we randomly mutate some of the parameters on some of the networks.
6. Here comes the fun part: Let' s say we started with a population of 20 networks, we kept the top 25% (5 nets), randomly kept 3 more loser networks, and mutated a few of them. We let the other 12 networks die. In an effort to keep our population at 20 networks, we need to fill 12 open spots.
7. Breeding:breeding is where we take two members of a population and generate one or more child, where that child represents a combination of its parents.In our neural network case, each child is a combination of a random assortment of parameters from its parents. For instance, one child might have the same number of layers as its mother and the rest of its parameters from its father. A second child of the same parents may have the opposite

**Experiment**

In the two layer full connected network and the two layer CNN and a full connected network, the hyper-parametric search time was shortened by 14.9 times and 9.5 times respectively by using genetic algorithm optimization.

Table 6: Search time of FC (origin Accuracy 97.546) by Evolutionary Computation

| | Model | Accuracy |
|---|---|---|
| Time | 64.1 | 4.3 |
| FC Accuracy | 98.130 | 98.290 |
| Supervised learning cnn fc | 99.219 | |

Table 7: Search time of CNN+FC (origin Accuracy 99.219) by Evolutionary Computation

| | Model | Accuracy |
|---|---|---|
| Time | 64.1 | 4.3 |
| FC Accuracy | 98.130 | 98.290 |
| Supervised learning cnn fc | 99.219 | |

## 5 Conclusion

We can see from the experimental results that the CNN has an advantage over traditional MLP in image recognition. Semi-supervised learning can achieve predictive performance beyond supervised learning by using non-labeled samples . Reinforced Learning Neural Architecture Searching NAS can effectively optimize neural networks and obtain high-precision models. The evolutionary algorithm is a gradient-free optimization algorithm, which may obtain a global optimal solution. Using genetic algorithm to optimizing the neural network hyperparameters can effectively shorten the search time.

## 6 Reference

1. Dufourq E , Bassett B A . [IEEE 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech) - Bloemfontein, South Africa (2017.11.30-2017.12.1)] 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech) - EDEN: Evolutionary deep networks for efficient machine learning[J]. 2017:110-115.

2. Kingma D P , Welling M . Auto-Encoding Variational Bayes[J]. 2013.

3. Abbasnejad M E , Dick A , Hengel A V D . Infinite Variational Autoencoder for Semi-Supervised Learning[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

4. Wang B , Huang J , Zheng H , et al. Semi-Supervised Recursive Autoencoders for Social Review Spam Detection[C]// International Conference on Computational Intelligence  Security. IEEE, 2017.

5. Li Y , Pan Q , Wang S , et al. Disentangled Variational Auto-Encoder for Semi-supervised Learning[J]. 2017.

6. Zoph B , Le Q V . Neural Architecture Search with Reinforcement Learning[J]. 2016.

7. Liu C , Zoph B , Neumann M , et al. Progressive Neural Architecture Search[J]. 2017.

8. Kandasamy K , Neiswanger W , Schneider J , et al. Neural Architecture Search with Bayesian Optimisation and Optimal Transport[J]. 2018.

9. Elsken T , Metzen J H , Hutter F . Simple And Efficient Architecture Search for Convolutional Neural Networks[J]. 2017.

10.Suganuma M , Shirakawa S , Nagao T . A Genetic Programming Approach to Designing Convolutional Neural Network Architectures[J]. 2017.

11. Pham H , Guan M Y , Zoph B , et al. Efficient Neural Architecture Search via Parameters Sharing[J]. 2018.

12. Haeffele, Benjamin D. , and R. Vidal . "Global Optimality in Neural Network Training." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) IEEE, 2017.

13. Tian, Hui , X. Zhou , and J. Liu . "A Hybrid Network Traffic Prediction Model Based on Optimized Neural Network." International Conference on Parallel  Distributed Computing IEEE Computer Society, 2017.

14. Shen, Yiran , et al. "CS-CNN: Enabling Robust and Efficient Convolutional Neural Networks Inference for Internet-of-Things Applications." IEEE Access (2018):1-1.

15. Serrano, Will , and E. Gelenbe . "The Deep Learning Random Neural Network with a Management Cluster." Kes-idt 2017.

16. Delahunt, Charles B. , and J. N. Kutz . "Putting a bug in ML: The moth olfactory network learns to read MNIST." (2018).

17. Lacey, Griffin , G. W. Taylor , and S. Areibi . "Stochastic Layer-Wise Precision in Deep Neural Networks." (2018).

18. Chen, Huili , B. D. Rohani , and F. Koushanfar . "DeepMarks: A Digital Fingerprinting Framework for Deep Neural Networks." (2018).

19. Limonova, E. , et al. "Convolutional Neural Network Structure Transformations for Complexity Reduction and Speed Improvement." Pattern Recognition and Image Analysis 28.1(2018):24-33.

20. Shirakawa, S. , Iwata, Y. ,  Akimoto, Y. . (2018). Dynamic optimization of neural network structures using probabilistic modeling.