2017

The Evaluation of Training A Siamese Network on Different Samples: An Experimental Report

SIAMESE NETWORK

GROUP 9: LINNI QIN N9632981 JIUZENG RONG N9814434

Table of Contents

I. Introduction

1.1 Experiment Context

Convolutional neural networks (CNN) have become an important tool to tackle the problem of object recognition in the field of machine learning (Eindhoven University of Technology, n.d.). As claimed by Dr Frederic Maire, he and his research students have identified that a Siamese system based on hierarchical convolutional neural networks is capable to perform ideally while its discriminating homographic images with a wide range of transformations (2017, p. 2). Their finding is based on the development of an automatic verification system for marine biologist to monitor marine life such as manta rays, with attempting to recognize the pose patterns of the same manta ray as per *Figure 1*.

Figure 1: Two Poses of One Manta Ray

In the context of experiment in this report, a similar Siamese neural network is built to predict whether two input data belong to one equivalence class. The basic subnetwork is a CNN with three conventional layers and two fully connected layers. The architecture of this basic system will be explained in detail in section 2.2.1. In the meantime, the artificial database for the experiment is a dataset named MNIST with containing grey scale images of handwritten digit ranging from 0 to 9 with size 28 pixels by 28s pixel as displayed in *Figure 2*. MNIST consists of one training set with 60,000 images and a test set of 10,000 images (TensorFlow, 2017).

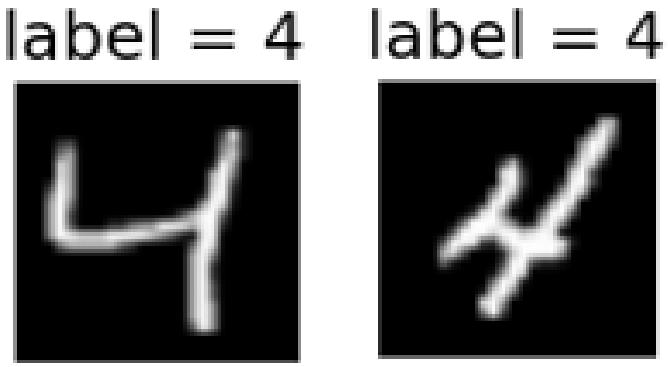


Figure 2 One Digit Image in MNIST

1.2 Experiment Objectives

By training the customized Siamese network with three different groups of samples, the experiment is attempting to examine the executing time, the converge progress and together with the accuracy rate so that the performance can be examined in a whole. The reasons to choose these three metrics are as following.

- Computing time is able to reflect the efficiency of the model;
- Model's loss denotes the learning progress of the model;
- Accuracy rate represents the effectiveness of the model.

The guidelines of experiment will be discussed in section two *Experimental Method* for conducting three times of model training and testing. Section three *Experimental Results* will demonstrate and discuss the related evaluation metric values. The achievement of the well-trained classifier will be claimed in the end.

1.3 Brief Description of Siamese and Convolutional Neural Network

Siamese Network

A Siamese neural network is applied to solve image matching problems (Gregory, Richard & Ruslan, 2017, p.3). As Figure 3, it contains twin networks which proceed distinct two inputs. The twin subnetworks within the experimental context is based on a 5-layer CNN which be explained in the latter sections. The contrastive loss is a loss function to ensure the pair images from a same class equals 1, otherwise is 0 (Raia, Sumit & Yann, 2006).

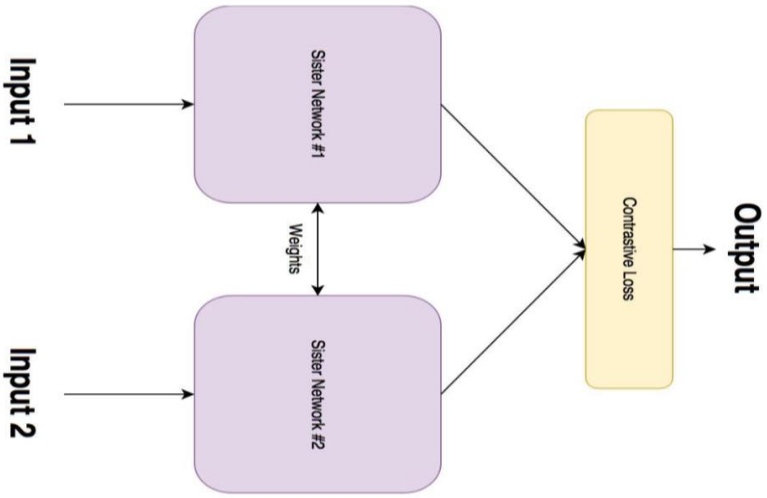


Figure 3 Siamese Network

Convolutional Neural Network

The architecture of a convolutional neural network is a list of layers neural networks as shown in Figure 4. It consists of certain distinct categories of layers, for instance, the conventional layers, subsampling layers and fully-connected layers in the end. The working operation of CNN is to convert the input image volume into an output volume by extracting features from the input image with sliding the filter over the input image, then learning the values of those filters for model training (UFLDL Tutorial, n.d.). Therefore, the more number of filters will enable CNN to extract more image features, hence, the better performance of the network at verifying the object. CNN has been proven that it is an effective neural network in the field of image classification and recognition (Eindhoven University of Technology, n.d.).

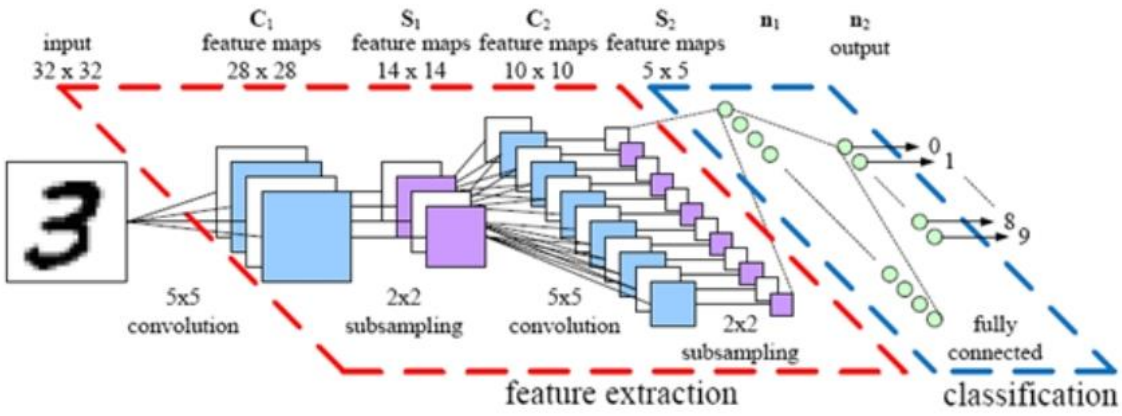


Figure 4 from Eindhoven University of Technology

II. Experimental Methodology

2.1 Computing Environment

The following computing environment is used for processing and classifying the database images, training the Siamese network, as well as, evaluating the network.

Operation Systems: Windows 7 Enterprise System x64-based
CPU: Intel® Xeon® CPU E5-2687W v4 @ 3.00GHz
Memory(RAM): 4.00 GB
Programming Tools: TensorFlow, Python 3.6.0, Spyder 3.1.2

2.2 Experiment Approach

2.2.1 Create CNN and Other Necessary Functions

Beside the original functions of *euclidean_distance(), contrastive_loss(), compute_accuracy()* and *create_pairs()*, another six self-defined functions have been added into the architecture of Siamese network based on the scaffolding code in *my_submission.py*. The six self-defined functions include:

- *intitial():*
  It is used to load and save the original MNIST dataset

- *warped_data():*
  It is used to create a dataset that can store 100,000 warped examples and define the template to warp images with filling in relative parameters.

- *convolutional_neural_networks():*
  Conventional layer is the core layer with loading most of the computational tasks. Because of the time limits for the whole testing task, the basic CNN network in this experiment meets the minimum requirement of assignment two. It is a five-layer CNN. For more training challenges, more conventional layers and fully connected layers can be added.

  Three conventional layers:
  The first conv layer consists of 16 filters, the second conv layer has 32 filters and the third one owns 64 filters. The kernel size for these three conv layers are 3 pixels by 3 pixels.

  Two fully-connected layers:
  The first fully connected layer is expected to have 128 output neurons and the output size of the last layer is corresponding to the 10 classes of digits ranging from 0 to 9.

  For these five network layers, ‘relu’ activation function is used to achieve better performance

- *exp1():*
  It will be called in Main method to train and evaluate the model on original MNIST dataset.

- *exp2():*
  It will be called in Main method to create the warped sample named Hard set with a rotation of at 45-degree and a projective transformation with a strength of 0.3, then to fit and evaluate the model with the warped samples.

- *exp3():*
  It will be called in Main method to create the warped sample named Easy set with a rotation of at 15-degree and a projective transformation with a strength of 0.1, then to create Hard sample set as exp2(). The model will be fitted and evaluated with the hybrid samples.

- *Main method:*
  Several essential variables are assigned in the Main method, including the size of warped image dataset, the epoch number, the batch size and the loops parameter. The precise of the prediction can be judged by average accuracy rate with repeating suitably a same experiment.
  Three relevant experiment functions listed above will be called from Main method to implement the test.

2.2.2 Train the Network on Original Dataset

After building the whole Siamese network, it is ready to implement the experiments. The following operation guideline conduct main actions of the classifier step-by-step to identify the class of the paired inputs. A couple of system actions will be adjusted accordingly in the other two experiments. The first training task is implemented with four literation

- Load the original data
- Create paired images
- Define Model
- Input two images
- Proceeding the input with Siamese Network (two mirrored CNN)

- Compute the distance between the two vectors
- Compile Model
- Fit Model
- Evaluate Model
- Plot the loss value

2.2.3 Train the Network on Warped Dataset

Referring to function description of exp2() in section 2.2.1, to load dataset of 100,000 images and create Hard sample set. The other operations follows the step-by-step list in last section 2.2.2.

2.2.4 Train the Network on Hybrid Dataset

Referring to function description of exp3() in section 2.2.1, to load the dataset of 100,000 images and create two sample sets: Easy and Hard. Then, fit the model twice with training Easy dataset firstly and Hard dataset afterward. Lastly, evaluate the model and plot the loss. Since the test of convolutional neural network is time thirsty based on this large scale of images, exp3() was implemented two times for classifier evaluation.

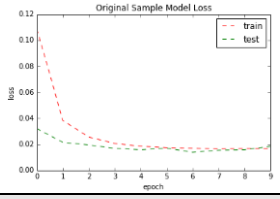
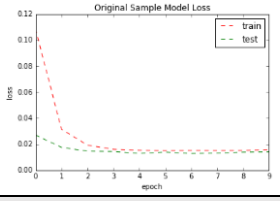
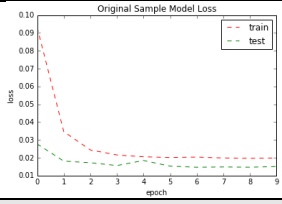
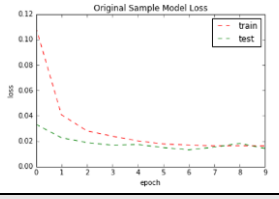
III. Experimental Results

The experimental results are demonstrated in two big tables. One of the tables contains the value covering the execution time for every epoch and the development of the weight loss. Referring to the data stated in Table 1, it can be noticed that the computing time for each epoch lasts early the same and it runs around 429 seconds to train the whole samples once no matter the example dataset is 60,000 or 100,000. Meanwhile, the loss data of the model that be trained on the original MNIST and the Easy dataset converges apparently faster than the model training on the Hard dataset. Given the training on original and easy set, the loss converges to the minimum around epoch five commonly. The loss value for the model that proceeds training on Hard set and the hybrid dataset drops significantly before the first three epoch, however, it converges slowly afterward. The other table includes the individual accuracy rate for ten tests and also the average rate for three main experiments.

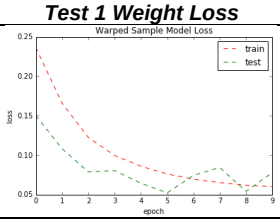
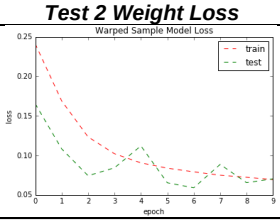
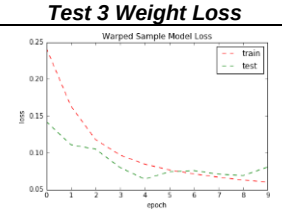
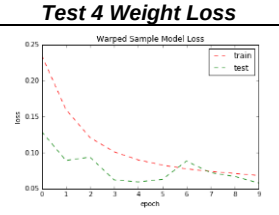
3.1 Execution Time and Weight Loss

Table 1

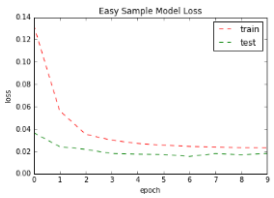
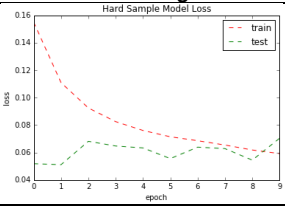
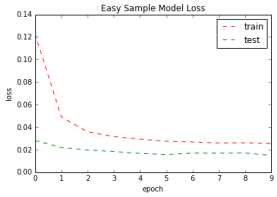
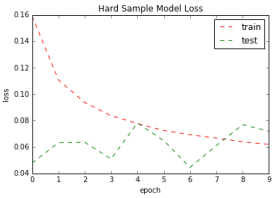
exp1() @Original Samples

Test1 Epoch	Time (s)	Loss	Test2 Epoch	Time (s)	Loss	Test3 Epoch	Time (s)	Loss	Test4 Epoch	Time (s)	Loss
1	427	0.1085	1	434	0.1085	1	430	0.0931	1	431	0.1095
2	426	0.0383	2	428	0.0315	2	428	0.0347	2	429	0.0408
3	426	0.0255	3	427	0.0192	3	428	0.0244	3	430	0.0284
4	426	0.0206	4	427	0.0161	4	427	0.0216	4	428	0.0238
5	426	0.0185	5	427	0.0154	5	428	0.0205	5	429	0.0202
6	426	0.0174	6	427	0.0150	6	428	0.0201	6	428	0.0178
7	426	0.0170	7	427	0.0152	7	427	0.0204	7	428	0.0169
8	426	0.0164	8	427	0.0151	8	427	0.0199	8	428	0.0163
9	426	0.0167	9	427	0.0152	9	427	0.0197	9	428	0.0165
10	429	0.0165	10	427	0.0158	10	428	0.0199	10	428	0.0162
Test 1 Weight Loss			***Test 2 Weight Loss***			***Test 3 Weight Loss***			***Test 4 Weight Loss***		
											

exp2() @Warped Samples (Hard)

Test1 Epoch	Time (s)	Loss	Test2 Epoch	Time (s)	Loss	Test3 Epoch	Time (s)	Loss	Test4 Epoch	Time (s)	Loss
1	428	0.2364	1	434	0.2410	1	430	0.2419	1	432	0.2349
2	428	0.1658	2	429	0.1689	2	429	0.1633	2	430	0.1600
3	427	0.1223	3	429	0.1233	3	430	0.1178	3	429	0.1210
4	427	0.0996	4	430	0.1020	4	429	0.0968	4	49	0.1012
5	427	0.0860	5	429	0.0905	5	429	0.0845	5	429	0.0898
6	427	0.0762	6	429	0.0838	6	429	0.0766	6	429	0.0828
7	427	0.0695	7	429	0.0792	7	429	0.0712	7	429	0.0774
8	430	0.0653	8	429	0.0749	8	429	0.0668	8	430	0.0740
9	427	0.0620	9	429	0.0723	9	429	0.0628	9	429	0.0711
10	427	0.0601	10	429	0.0694	10	430	0.0602	10	430	0.0682
Test 1 Weight Loss			***Test 2 Weight Loss***			***Test 3 Weight Loss***			***Test 4 Weight Loss***		
											

exp3() @Easy + Hard

Test1_E Epoch	Time (s)	Loss	Test1_H Epoch	Time (s)	Loss	Test2_E Epoch	Time (s)	Loss	Test2_H Epoch	Time (s)	Loss
1	433	0.1327	1	430	0.1549	1	431	0.1227	1	429	0.1591
2	428	0.0562	2	428	0.1109	2	429	0.0492	2	428	0.1107
3	428	0.0353	3	428	0.0925	3	429	0.0359	3	428	0.0937
4	428	0.0300	4	428	0.0825	4	429	0.0316	4	429	0.0837
5	428	0.0270	5	428	0.0761	5	428	0.0293	5	428	0.0778
6	428	0.0255	6	428	0.0715	6	428	0.0275	6	428	0.0725
7	428	0.0244	7	428	0.0686	7	428	0.0269	7	428	0.0693
8	428	0.0239	8	428	0.0655	8	428	0.0259	8	429	0.0667
9	428	0.0232	9	428	0.0618	9	429	0.0261	9	437	0.0639
10	431	0.0231	10	428	0.0592	10	428	0.0255	10	438	0.0620
Test 1 Easy Weight Loss			***Test 1 Hard Weight Loss***			***Test 2 Easy Weight Loss***			***Test 2 Hard Weight Loss***		
											

3.2 Accuracy Rate

Table 2

exp1() @Original Samples	Test 1	Test 2	Test 3	Test 4	Average
Accuracy on Training set	*99.38%*	*99.38%*	*99.35%*	*99.34%*	*99.36%*
Accuracy on Test Set	*98.27%*	*98.59%*	*98.51%*	*98.50%*	*98.50%*
exp2() @Warped Samples (Hard)	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Average**
Accuracy on Training Set	*95.76%*	*92.88%*	*93.95%*	*94.37%*	*94.24%*
Accuracy on Test Set	*90.54%*	*90.07%*	*89.64%*	*93.45%*	*90.93%*
exp3() @Easy + Hard	**Test 1**	**Test 2**	**--**	**--**	**Average**
Accuracy on Training Set	*96.20%*	*95.81%*	**--**	**--**	*96.00%*
Accuracy on Test Set	*91.49%*	*90.70%*	**--**	**--**	*91.10%*

3 Conclusion

To sum up, by evaluating the metrics of execution time, the weight loss and the accuracy rate, the training model on the original MNIST samples achieves the highest accuracy rate when identifying the homographic digit images. The performance of the model with training the easy dataset first then training the hard dataset appears better than the model with training on the hard dataset. Furthermore, the common average accuracy rates stay higher than 90 percent while training the Siamese network on three different groups of samples. Thus, it turns out that Siamese network with CNN subnetworks is not only capable to recognise precisely the handwritten digit but also able to classify the same digit from different observation points in a same class.

For further experiment, the training of the Siamese network can be implemented with more challenges by changing the size of epoch or batch, increasing or reducing the layer of CNN, changing the filter number for each CNN layer, kernel size or editing the rotating degree and transformation strength. The code in the attached *my_submission.py* contributes majority to those future experiments with providing clear network structure, functions with purposes and convenient portals to change the mentioned parameters without adding code. Furthermore, to reduce the execution time, it would be better to execute the experiment with GPU-based computers.

Reference

Eindhoven University of Technology (n.d.). Conventional Neural Networks. Retrieved from http://parse.ele.tue.nl/education/cluster2

Frederic, M., (2017). 2017_IFN680_assignment_2. Retrieved from https://blackboard.qut.edu.au/bbcswebdav/pid-7008990-dt-content-rid-9551981_1/courses/IFN680_17se2/2017_IFN680_assignment_2%281%29.pdf

Gregory, K., Richard, Z. & Ruslan, S. (2017). Siamese Neural Networks for One-shot Image Recognition. Retrieved from https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

Raia, H., Sumit, C. & Yann, L (2006). Dimensionality Reduction by Learning an Invariant Mapping. Retrieved from http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf

TensorFlow (2017). MNIST For ML Beginners. Retrieved from https://www.tensorflow.org/get_started/mnist/beginners

UFLDL Tutorial (n.d.). Conventional Neural Network. Retrieved from http://ufldl.stanford.edu/tutorial/