Discussion of the comparison with Auto-Encoder based on Artificial Neural Network (ANN) and Convolution Neural Network (CNN) for the Semeion Hand-written Digit Data Set

Boya Na
College of Engineering and Computer Science, Australian National University

The Australian Nation University, Canberra
Email: U5918786@anu.edu.au

Abstract. This report introduced the comparison between Auto-Encoder with Artificial Neural Network (ANN) and Convolution Neural Network (CNN) both using the Semeion Hand-written Digit Data Set. Both Auto-Encode plus ANN and CNN are implemented for classification of data set on Python and Pytorch. The Semeion Hand-written Digit Data Set records the various hand-written data referenced from UCI Machine Learning Repository. Finally, the report provides some discussion based on the comparison of two neural network structures

Keywords: Auto-Encoder, Artificial Neural Network, ANN, Convolution Neural Network, CNN, Semeion Hand-written Digits, Classification

1 Introduction

As one the major machine learning package in Python, Pytorch is used to implement the establishment of the structures for Auto-Encoder, Artificial Neural Network (ANN) and Convolution Neural Network (CNN). Pytorch applies some acceleration libraries like Intel MKL or NVIDA (CuDNN, NCCL) which can enhance the running of the neural networks (Pytorch, 2017). The used data set is Semeion hand-written digits dataset referenced from UCI Machines Learning Repository which contains 1593 instances with 256-dimension binary values (Semeion Research Center of Sciences of Communication, 2008). It will be introduced in detail in Methods part. Both Auto-Encoder and CNN are built depending on the ANN inspired and researched by the biological human brain structure (Zankinski, Barova, & Tomov, 2017). The regular ANN contains the input layer, the hidden layer and output layer, which functions are input information, processing information and return the result (Logan, 2017). Auto-encoder as a method of feature extraction of the data can be applied to improve the code performance (Cohen, 2017) and CNN can process images such as classification or recognition and provides the partial invariance for some operations including translation, rotation and more (Lawrence, Giles, Tsoi, & Back, 1997). The comparison of two methods will be discussion in later.

2 Methods

2.1 Motivation

The targets of implementation, especially to applying this data set, are to recognise the handwritten digital result automatically and accurately through the classification completed by machines. It need to apply the machine learning methods, such as ANN or Deep learning algorithms. Actually, the implementation of algorithms can be applied to reduce the working cost of the handwritten materials manually such as the electronical recording of hand-written materials or the recognition of the ancient materials. In addition, Auto-Encoder can be applied to extract the features and CNN is one of suitable image processing method. Thus, it is meaningful to compare and discuss two methods.

2.2 Dataset

Semeion hand-written digits dataset, which is found and references from UCI Machine Learning Repository, contains 1593 pieces of recording and each instance is made up by a 256-dimension binary vector with a 10-dimension label. Actually, the original handwritten digit images were scanned from about 80 people and each person wrote down the digit twice from 0 to 9 on the paper; the scanning result was recorded by resizing to 16x16 and transferred first to the grayscale-

type images, of which all pixel channel are 1 and values are float between 0 to 1; then the grayscale image matrix was converted to the binary matrixes that only contains 0 or 1 value by applying a single threshold; finally, all recorded images matrixes were reshaped form 16x16 to 256-dimension vectors and recording instances were generated by both data vectors and labels (Semeion Research Center of Sciences of Communication, 2008).

All labels recording in data set are represented by a 10-dimension binary vector. The label binary vector only contains one 1 value and other values are all 0. Thus, in this allocation, the position of value 1 is representing the actual value of the label. For example, if a label of a single instance is [0 1 0 0 0 0 0 0 0 0], it means that the original image converted to the data of this instance should show a digit 1.

The reason why this data set was selected was that the dimension of the data in this set is big enough which can be applied the data compression algorithms to save the running space. Correspondingly, some problem for the application of this data set will be discussed in the later section.

2.3 Data file type conversion and reading

The data file downloaded from UCI ML repository is the .data type and all the data was separated by blank space rather than comma. It means that the data can be only read normally through the line-by-line method, if the file type is not changed. It is less convenient than using pandas reading csv file because reading line-by-line need to combine the data into a matrix but reading csv file by pandas can generate and return a matrix directly. Thus, the black spaces between data values in the original file are replaced by commas which can help to separate data when the file type is converted to csv. After the conversion of the data file type, the data can be read directly by the in-built function provided by pandas package in Python and the function will return a matrix recording all data values and label values. Additionally, all the data type is transformed from float to integer which can be helpful for memory management.

2.4 Data set pre-process

Comparing with other handwritten datasets, Semeion hand-written digit dataset has completed some data pre-processing such as compressing data by resizing of the original images, changing images from three channels such as RGB, HVC or other types to only one channel (grayscale even binary type) or converting the matrix to vector by reshaping. However, some pre-processing operations containing label encoding, data reshaping, data compression, data loading and batching separation, still need to be applied to prove the classification is able to complete.

2.4.1 Label Encoding

As pervious description, the label of the data set is a 10-dimension vector. It means that it cannot be compared by the prediction result directly under ANN linear prediction. Therefore, before loading and using data in training or testing process, the label vectors has been encoded manually to their actual meaning digit, for example, encoding the vector [1 0 0 0 0 0 0 0 0 0] as digit 0, [0 1 0 0 0 0 0 0 0 0] as digit 1 and [0 0 1 0 0 0 0 0 0 0] as digit 2.

2.4.2 Data Reshaping

An instance of data reading from dataset a 256-dimension which is convenient to process for ANN especially its hidden layer is linear structure. However, it has some problems because the convolution layers and pooling layers cannot be applied on the vectors. Thus, the vector of each instance data should be reshaped back to 16-by-16 matrix from 256-dimension and a loop can be used to finish this process.

2.4.3 Data compression

In this report, two methods, Auto-Encoders and CNN, are implemented and introduced latter.

2.4.4 Data loading and batch separating

In the data training and testing of ANN, data can be called directly just after processing by some in-built functions. However, the loading data directly would cause the type errors especially the data value has been converted to the matrix with 16-by-16 size. Thus, the costume dataset class inherited from the dataset class and data loader of Data of Pytorch should be applied which can avoid the problems of types in training and testing, especially for the calculation of the loss functions. The costume dataset class should be rebuilt the functions *init()* which initializes the parameters of an class object, *getitem()* which return the a data-label pair by provided index and *len()* which returen the length of the dataset. In

initialization function, the data set loaded is separated to data part and label part and the data part need to conduct data reshaping processing introduced in pervious part before storing. After the generation of the datasets for both training and testing, the training data set was sent into Pytorch data loader for training. In addition, the data loader can separate automatically dataset by setting batch size.

2.5 Artificial Neural Network

ANN used in this working has the standard three-layer neural network which contains an input layer, a hidden layer and an output layer. The number of nodes in output layer are setting to ten based on the requirement of Pytorch, otherwise it would produce the runtime error in calculating loss function. The actual prediction with a single label value is selected from the output values. The number of nodes in hidden layer is based on the setting manually but if it is too large, the computing time and needed memory will also rise up so much, although the prediction may also increase. It means that high number hidden nodes will increase both accuracy and time and space complexity. The number of nodes in input layer is based on the input data because this number should be same with the dimension of the input data at least. In this data set, if no any application of dimension reducing technique, the number of input nodes should be 256 which is same with dimension of the data vector. The ANN is defined as a class and it need be initial by the node number of three layers. The hidden layer and output layer are both linear layer defined in the initial function and the activation function is defined with the sigmoid function, which will be applied to process the hidden layer input data to get the output data in the normal forward order. In addition, the loss is calculated by cross entropy loss method, which is similar to calculate the negative log likelihood (DiPietro, 2016) and stochastic gradient descent (SGD) is used for training process, which can update the parameter by decline the gradient (Christopher, 2016). For each batch, the gradient will be cleared, and the backward propagation will be applied to adjust the model parameters. Additionally, the setting of ANN part is same for both application in Auto-Encoder and CNN.

2.6 Auto-Encoder

The Auto-Encoder which is a technique from the paper Stochastic bidirectional training in the week 7 Compression part (Gedeon, 1998) is applied to extract the features and reduce input data although it cannot reduce the dimension of data and need the external running time and space. Auto-encoder neural network is an unsupervised learning algorithm (UFLDL Tutorial, n.d.) . The encoder and decoder which are the major feature extracting steps in Auto-Encoder are defined with the linear model. Through the encoder, the major valuable features with the target size is kept, then the decoder recovers the dimension of the input data. Although the dimension of data is not changed, the features which can affect the performance of the training model are extracted and reduce the difficulty of training. For example, the assumption is that the input data is ten-dimensional and the data in 3rd dimension is extracted out as the only one feature, in the training process of later classification, the linear classification model will be adjusted to which the parameters of 3rd dimension can affect the training result more. Actually, the feature extracting process is an external learning of which parameters are not related to the classification of the ANN. It means that the whole classification need twice training and test to get the prediction. The loss function of the implementation in auto-encoder training is the mean square error loss. Additionally, batch separating, SGD and backward propagation are also applied in the training process to get a better performance for model. The structure of Auto-Encoder is showing in Fig. 1.

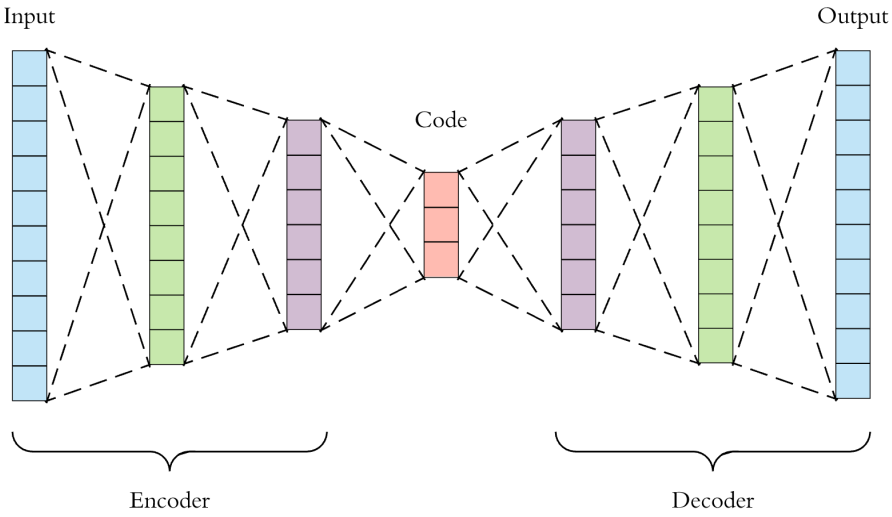

Fig. 1. Auto-Encoder Structure (Dertat, 2017)

2.7 Convolution Neural Network

Convolution Neural Network is a deep learning algorithm which can help machine to learn the processing of images and keep the partial invariance for some operations including translation, rotation and more (Lawrence, Giles, Tsoi, & Back, 1997). The regular structure of CNN consists of the convolution layer(s), the corresponding ReLU layer(s), which is a kind of activation functions, and pooling layers, and a full connection ANN. In the convolution layer, the image matrix is processed with the convolution calculation between the kernel matrix and same size area in image matrix step by step, that is related to the setting of the filter movement. Then the calculation results will be processed by ReLU layers and extracted the features by Pooling layers. After that, the data processed will be sent into the later full connection ANN. Thus, both dimension reduction and feature extraction for the input data can be implemented after the processing of the convolution layers, ReLU layers and pooling layers. The regular classification of CNN only contains once training and test. Similarly, the CNN is implemented by a class and the initialization function set the parameters for both convolution-pooling layers and full connection ANN based on the neural network model provided by Pytorch. The setting of full connection ANN has been introduced before.

3 Results and Discussion

3.1 Fixed parameters

Some parameters are set fixed for both CNN and Auto Encoder, for instance, the batch size is 50 temporarily and all pooling layers are average pooling layer with the kernel size = 2. Although the pool layer can also be setting to maximum pooling layer, it will not affect the comparison seriously. Both the full connection ANN whether in CNN or Auto-Encoder has 20 nodes in hidden layers and 10 nodes in output layers under the 500 epochs.

3.2 Results of CNN

The parameters of CNN are setting differently for comparison of CNN result. First result showing in Table 1. is based on the one convolution-pooling layer CNN which consists of a single convolution layer with kernel size = 16, a ReLU layer and a single corresponding pooling layer under the learning rate 0.01. It causes that the size of input nodes of ANN is 784. The second result showing in Table 2. is similar but based on the two convolution-pooling layers. Its kernel size of first convolution layer is 16 and second one is 32 thus the size of input nodes is 128. It can be observed from the result tables that the application of CNN can generate a satisfactory result and reduce the dimension of input data with more than one convolution-pooling layers, but it will also cost more running time.

Table 1. Result of CNN with 1 convolution-pooling layer

Learning rate	0.01		
Accuracy	99.54%	97.25%	99.03%
Running time (s)	54.568	54.346	52.9502

Table 2. Result of CNN with 2 convolution-pooling layers

Learning rate	0.01			0.02		
Accuracy	89.59%	98.25%	90.59%	96.18%	96.44%	97.94%
Running time (s)	88.782	88.754	90.492	117.085	91.706	92.083

3.3 Result of Auto-Encoder

The Auto-Encoder is trained with its own neural network with the code size 100, batch size 50, epoch number 2000 and learning rate 0.02. The result is recording in Table 3 with the different learning rate of ANN. After the application of feature extraction of data, it can produce an acceptable result with much external time.

Table 3. Result of Auto-Encoder

Learning rate	0.01			0.02		
Accuracy	92.12%	89.47%	93.88%	93.98%	95.11%	94.19%
Running time (s)	117.060	121.839	115.474	118.977	116.729	162.497

3.4 Comparison of results of CNN and Auto-Encoder

The comparison is between the CNN with two convolution-pooling layer and Auto-Encoder in average of pervious result. The result is recording in Table 4 under the condition of controlling variable same. It is obvious that in total, the performance under learning rate 0.02 is better than learning rate 0.01 whether CNN or Auto-Encoder because of higher accuracy but it also cost more time. However, the enhancing of CNN is more significant than Auto-Encoder, but the time cost of CNN is also increasing more. In addition, CNN performs better than Auto-Encoder whether in the accuracy or the cost of code running time.

Table 4. Result of Comparison

Learning rate	0.01		0.02	
Type	CNN	Auto-Encoder	CNN	Auto-Encoder
Average Accuracy	92.81%	91.82%	96.85%	94.43%
Average Running time (s)	89.343	118.124	100.291	132.734

3.5 Discussion of comparison

Compared with Auto-Encoder, CNN performs better whether in time saving or prediction accuracy increasing. However, both results are based on the original dataset which has been pre-processed before the its generation. Actually, the results of prediction by ANN directly, which is showing in Table 5, are so much better than the performances of both CNN and Auto-Encoder because it only has binary data value. Although CNN with 1 convolution-pooling layer provided a similar performance on prediction accuracy, it cost much more time than the classification of ANN directly.

4 Conclusion and Future Work

4.1 Conclusion

In total, both CNN and Auto-Encoders can be applied for the reduce the measure of the input data size. However, CNN performs overall better than Auto-Encoders whether on accuracy or time cost.

4.2 Future Work

Actually, in the exploring process, when the Auto-Encoder with same codes was applied on some other hand-written datasets which is not special with Semeion hand-written digit dataset, the result performed terrible because the loss in Auto-Encoder training process was showing as nan which means too large. Similarly, the CNN also shows its unstable ability in the classification of MNIST data set. Therefore, the future work may be developed on the experiments for the

CNN and Auto-Encoder running on other hand-written datasets which are more general than Semeion hand-written dataset or include letters with some necessary pre-processing.

References

Christopher, M. B. (2016). *PATTERN RECOGNITION AND MACHINE LEARNING.* New York: Springer-Verlag.

Cohen, E. (2017, July 2). *Reducing Dimensionality from Dimensionality Reduction Techniques.* Retrieved from https://towardsdatascience.com/reducing-dimensionality-from-dimensionality-reduction-techniques-f658aec24dfe

Dertat, A. (2017, October 3). *Applied Deep Learning - Part 3: Autoencoders*. Retrieved from https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

DiPietro, R. (2016, May 2). *A Friendly Introduction to Cross-Entropy Loss.* Retrieved from https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/#introduction

Gedeon, T. (1998). Stochastic bidirectional training. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on. 2*, pp. 1968-1971. IEEE.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997, January). Face Recognition: A Convolutional Neural-Network Approach. *IEEE TRANSACTIONS ON NEURAL NETWORKS, 8*(1).

Logan, S. (2017, November 27). *Understanding the Structure of Neural Networks.* Retrieved from https://becominghuman.ai/understanding-the-structure-of-neural-networks-1fa5bd17fef0

Pytorch. (2017). *About Pytorch*. Retrieved from http://pytorch.org/about/

Semeion Research Center of Sciences of Communication. (2008). *Semeion Handwritten Digit Data Set.* Retrieved from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit

UFLDL Tutorial. (n.d.). *Autoencoders.* Retrieved from http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

Zankinski, I., Barova, M., & Tomov, P. (2017). Hybrid Approach Based on Combination of Backpropagation and Evolutionary Algorithms for Artificial Neural Networks Training by Using Mobile Devices in Distributed Computing Environment. In L. I., & M. S., *Large-Scale Scientific Computing* (Vol. 10665). Springer, Cham.