Image Compression with Auto-encoder Algorithm using Deep Neural Network (DNN)

Adna Sento
Faculty of Engineering
Thai-Nichi Institute of Technology, Bangkok, 10520, Thailand
Corresponding Author E-mail addresses: adna@tni.ac.th

***Abstract*—An image compression is necessary for the image processing applications such as data storing, image classification, image recognition etc. Then, several research articles have been proposed to reserve for these topics. However, the image compression with auto encoder has been found for a small number of the improvements. Therefore, this paper presents a detailed study to demonstrate the image compression algorithm using the deep neural network (DNN). The proposed algorithm consists of 1) compressing image with auto encoder, and 2) decoding image. The proposed compressing image with auto encoder algorithm uses non-recurrent three-layer neural networks (NRTNNs) which use an extended Kalman filter (EKF) to update the weights of the networks. To evaluate the proposed algorithm performances, the Matlab program is used for implementations of the overall testing algorithm. From our simulation results, it shows that the proposed image compression algorithm is able to reduce the image dimensionality and is able to recall the compressed image with low loss.**

Keywords—Image compression algorithm, Deep neural network, Image processing, extended Kalman filter.

I. INTRODUCTION

Nowadays, the image compression algorithms for dimensional reductions play an important role in an image processing especially in the image classifications, visualizations, communications, and storage of the high-dimensional data. Generally, the principal component analysis (PCA) is one of the choice solution to handle with these problems. The PCA algorithm finds the best variance in the data which are represented by their coordinating. Then, the PCA algorithm widely spreads out to many real-world applications such as [1], [2], [3] etc. However, in the case of the nonlinear PCA with auto encoder data has been found for a small number of the improvements.

Consequently, several types of the algorithms using various kinds of the techniques have been established for example, G. Hinton and R.R. Salakhutdinov proposed the dimensionality reduction of data with the neural network [4]. They used a restricted Boltzmann machine (RBM) to create a reduction of the binary stochastic information. Next, T. Orłowski presented an image compression and an extraction using a deep belief network (DBN) [5]. From his demonstration, it found that this algorithm is able to potentially compress the image with less loss extraction. Furthermore, C. Dong et al. provided the other approach to compress the image with low loss, called deep convolutional networks (DCN) [6]. They used DCN algorithm to train a large of the image data sets to a low dimension compression. Recently, S. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie also proposed the image data sets dimensionality reduction using the deep belief neural network (DBN) [7]. From their experimental results, it shows that this algorithm efficiently compresses the data on the tasks without loss of the accuracy.

Although several research articles were successful in gaining the effective performances on the image data set dimensionality reduction, it still has the rooms to be developed including the easy to use, head cost effective and uncomplicated calculations. Therefore, this paper intends to develop a new image compression with auto encoder using a deep neural network (DNN). The aim of the paper is to obtain the low dimensional image for the applications such as image classifications, visualizations, communications, and storage of the high-dimensional data. This paper is organized as follows. A new image compression with auto encoder using a deep neural network (DNN) is first described in section II. Then, in section III, the experiment setup and results of the image compression algorithm performances are discussed. Finally, the conclusions are given in section IV.

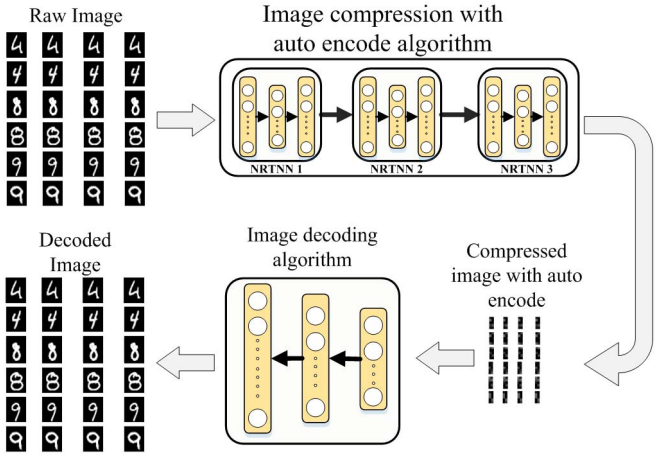

Figure 1. The proposed image compression with auto encoder algorithm using a deep neural network (DNN)

II. THE PROPOSED IMAGE COMPRESSION WITH AUTO ENCODER ALGORITHM USING DEEP NEURAL NETWORK (DNN)

The overall of the proposed image compression with auto encoder algorithm using a deep neural network (DNN) is shown in Fig. 1. The proposed algorithm consists of five parts; 1) raw image data sets, 2) image compression with auto encoder algorithm, 3) compressed image output, 4) image decoding and 5) the decoded image. The details of the proposed algorithm will be described as follows.

A. Raw Image Data Set

The training image data sets consist of 60,000 samples of digits as shown in Fig. 2. Each image consists of 28x28 gray

978-1-5090-4105-3/16/$31.00 ©2016 IEEE

scale pixels, which are binarized with a threshold value of 127. All of the training sets is the MNIST hand written digit recognition benchmark [8].

Figure 2. The 28x28 gray scale of the handwritten image.

B. The proposed Image Compression Algorithm

Several image compression algorithms, also known as image dimensional reduction algorithms, have been proposed for some applications such as image classifications, image visualizations, image communications etc. In this paper, we present the image compression that is not only reduce the image dimensionality but also automatically encode the image by using the deep neural network (DNN) algorithm. The construction of the deep neural network (DNN) is a non-recurrent three-layer neural networks (NRTNNs) that trains the weights until reaching the optimal hidden neurons. The non-recurrent three-layer neural network structure is used for the data training as shown in Fig. 3. It consists of three-layer without the recurrent node; 1) observable data layer with m visible nodes $\mathbf{v} = (v_1, v_2, \ldots\ldots, v_m)$, 2) hidden variable layer with n hidden nodes $\mathbf{h} = (h_1, h_2, \ldots\ldots, h_n)$ with the biases node $\mathbf{b} = (b_1, b_2, \ldots\ldots, b_n)$ and 3) predicted variable layer with m nodes $\mathbf{t} = (t_1, t_2, \ldots\ldots, t_m)$ with the bias nodes $\mathbf{c} = (c_1, c_2, \ldots\ldots, c_m)$. There are several links associated between visible neurons and hidden neurons, known as weights ($\mathbf{w}$). Similarly, the links that associate between hidden neurons and predicted neurons are defined as the transpose of weights ($\mathbf{w}^T$). The non-recurrent three-layer neural networks (NRTNNs) use the weights of the neural network relation function to map the raw image data set from an input space $\mathbf{v}$ of dimension m to a hidden space $\mathbf{h}$ of dimension n where $n < m$. Then, it clearly sees that the features of the image data are reduced automatically.

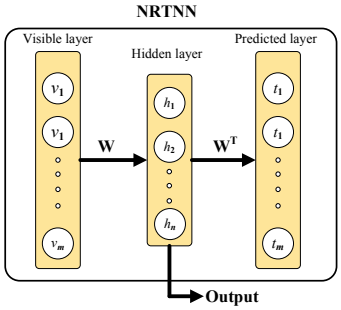

Figure 3. The non-recurrent three-layer neural network (NRTNN) structure.

The basic idea of the non-recurrent three-layer neural network (NRTNN) is to find the joint distributions over the hidden and visible neurons of the network which can be expressed as

$$h_i = sig\left(b_i + \sum_j^n w_{ij} v_j\right) \tag{1}$$

$$t_j = sig\left(c_j + \sum_i^m w_{ji} h_i\right) \tag{2}$$

where sig(•) is the sigmoid function, $\mathbf{v}$ is the input space vectors and also defines for a desired target vector, and $\mathbf{t}$ is the predicted output space vectors of the neural network. The aim of the algorithm is to minimize the different values between the visible neurons and the predicted neurons, called the errors. Therefore, we provide the effective algorithm to update the weights of the network with fast convergent property, called extended Kalman filter (EKF) learning algorithm which is shown in the Fig. 4.

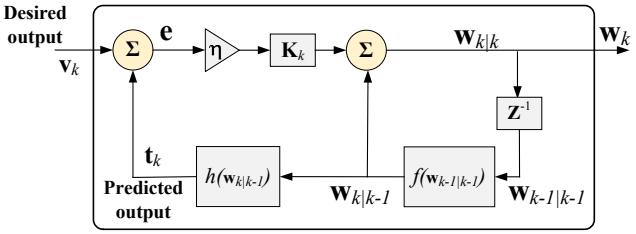

Figure 4. The block diagram of weight updated rule of the NRTNN using an extended Kalman filter (EKF) algorithm.

Generally, we form the state vector of the Kalman model to create the Kalman functions that can be expressed as

$$\mathbf{w}_k = \left[w_{11,k}^h \quad \cdots \quad b_{3,k}^h \quad w_{1,k}^y \quad \cdots \quad w_{3,k}^y \quad b_k^y\right]^T \tag{3}$$

where $\mathbf{w}^y$ and $\mathbf{w}^h$ are the state of the weighted network in the output and the hidden layer, respectively. According to the non-recurrent three-layer neural network (NRTNN), the process model function and measurement model function in according to [9] are given as:

$$\mathbf{w}_k = f\left(\mathbf{w}_{k-1}\right) \tag{4}$$

$$\mathbf{t}_k = sig\left(\mathbf{w}_k^T\left[sig\left(\mathbf{w}_k \mathbf{v}\right) + \mathbf{b}\right] + \mathbf{c}\right) \tag{5}$$

The procedure of the proposed weight update rule, as shown in Fig. 4, are described as follows:

1) The parameters including $\mathbf{w}_0$, $\mathbf{K}_0$, $\mathbf{P}_0$, and $\mathbf{R}_0$ will be initiated by random to generate the predicted output $\mathbf{t}$ at the time k. Conseqeuntly, the neural network will give an error which is the difference between predicted values and visible values, called disired values. If error is not equal to zero, then the weight update rule begins to calculate by using the extended Kalman filter algorihtm, which has 2 steps as follows.

a) The model prediction step: The predicted state ($\mathbf{w}_{k|k-1}$) and the predicted error covariance ($\mathbf{P}_{k|k-1}$) are first given by [10]

$$\mathbf{w}_{k|k-1} = \mathbf{w}_{k-1|k-1} + \alpha \frac{\partial \mathbf{E}_{k-1}}{\partial \mathbf{w}_{k-1|k-1}} \tag{6}$$

$$\mathbf{P}_{k|k-1} = \mathbf{P}_{k-1|k-1} + \mathbf{Q}_{k-1} \tag{7}$$

where $\mathbf{E}$ is an error cost function that is a sum square error function, α is a learning rate which is set to 0.01, $\mathbf{Q}_k$ is the process noise error covariance matrix assumed to be zero, and $\mathbf{P}_{k|k-1}$ is the latest predicted error covariance.

b) The measurement prediction step: To optimize the predicted state ($\mathbf{w}_{k|k-1}$), we must determine the Kalman gain expressed as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left[\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\right]^{-1} \tag{8}$$

where $\mathbf{R}_k$ is the unknown prior covariance that is given by

$$\mathbf{R}_k = \mathbf{R}_{k-1} + \frac{1}{k}\left((\mathbf{v}-\mathbf{t}_k)(\mathbf{v}-\mathbf{t}_k)^T - \mathbf{R}_{k-1}\right) \quad (9)$$

The parameter $\mathbf{H}_k$ can be described as follows:

$$\mathbf{H}_k = \frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k} = \left[\frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k^h} \quad \frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k^y}\right] \quad (10)$$

In order to simplify the matrix $\mathbf{H}$, the element of equation (10) is divided into 2-part that will be expressed as

$$\mathbf{H}_k = \left[\mathbf{H}_k^y \quad \mathbf{H}_k^h\right] = \left[\frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k^h} \quad \frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k^y}\right] \quad (11)$$

$$\mathbf{H}_k^h = \frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k^h} = \left[\frac{\partial \mathbf{t}_k}{\partial \mathbf{h}_k}\frac{\partial \mathbf{h}_k}{\partial \mathbf{a}_k}\frac{\partial \mathbf{a}_k}{\partial \mathbf{w}_k^h}\right] \quad (12)$$

Each of the elements of the matrix $\mathbf{H}^h$ are given as follows:

$$\mathbf{H}_k^h = \begin{bmatrix} \frac{\partial t_{1,k}}{\partial h_{1,k}} & \cdots & \frac{\partial t_{1,k}}{\partial h_{n,k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial t_{m,k}}{\partial h_{1,k}} & \cdots & \frac{\partial t_{m,k}}{\partial h_{n,k}} \end{bmatrix}_{m\times n} \begin{bmatrix} \frac{\partial h_{1,k}}{\partial a_{1,k}} & \cdots & \frac{\partial h_{1,k}}{\partial a_{n,k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{n,k}}{\partial a_{1,k}} & \cdots & \frac{\partial h_{n,k}}{\partial a_{n,k}} \end{bmatrix}_{n\times n} \times \begin{bmatrix} \frac{\partial a_{1,k}}{\partial w_{11,k}^h} & \cdots & \frac{\partial a_{1,k}}{\partial w_{nm,k}^h} & \frac{\partial a_{1,k}}{\partial b_{n,k}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial a_{n,k}}{\partial w_{11,k}^h} & \cdots & \frac{\partial a_{n,k}}{\partial w_{nm,k}^h} & \frac{\partial a_{n,k}}{\partial b_{n,k}} \end{bmatrix}_{n\times(nm+1)} \quad (13)$$

where a_k is the linear combination of each of the hidden node $\mathbf{h}$. In case of the $\mathbf{H}^y$ matrix can be expressed as

$$\mathbf{H}_k^y = \frac{\partial \mathbf{t}_k}{\partial \mathbf{w}_k^y} = \left[\frac{\partial \mathbf{t}_k}{\partial s_k}\frac{\partial s_k}{\partial \mathbf{w}_k^y}\right] = \begin{bmatrix} \frac{\partial t_{1,k}}{\partial s_{1,k}} & \cdots & \frac{\partial t_{1,k}}{\partial s_{m,k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial t_{m,k}}{\partial s_{1,k}} & \cdots & \frac{\partial t_{m,k}}{\partial s_{m,k}} \end{bmatrix}_{m\times m} \times \begin{bmatrix} \frac{\partial s_{1,k}}{\partial w_{11,k}} & \cdots & \frac{\partial s_{1,k}}{\partial w_{mn,k}} & \frac{\partial s_{1,k}}{\partial c_{m,k}} \\ \vdots & \ddots & \vdots & \vdots \\ \frac{\partial s_{n,k}}{\partial w_{11,k}} & \cdots & \frac{\partial s_{n,k}}{\partial w_{mn,k}} & \frac{\partial s_{n,k}}{\partial c_{m,k}} \end{bmatrix}_{m\times(nm+1)} \quad (14)$$

where s_k is the linear combination of each of the output node. Then, the new update of the weights ($\mathbf{w}_{k|k}$) and the updated corresponding error covariance ($\mathbf{P}_{k|k}$) are expressed as follows:

$$\mathbf{w}_{k|k} = \mathbf{w}_{k|k-1} + \eta \mathbf{K}_k\left(\mathbf{v}-\mathbf{t}_{k-1}\right) \quad (15)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k^T \mathbf{P}_{k|k-1} \quad (16)$$

where η is the Kalman scale rate that set to 0.1. From our experiments, the Kalman gain gives a large scale so that it must be multiplied by the Kalman scale rate (η) to normalize and to avoid the over fitting in the increment of the updated weights.

2) The recursive calculation of each iterations will continually update the weights of the network until reaching the setting criteria that is set to 0.001 of the error between desired output and predicted output or reaching the setting epochs.

C. Image Decoding

After the proposed image compression algorithm is applied, we obtain the encoded image outputs by using the value of the hidden neurons. Furthermore, the dimensionality of the image has been reduced corresponding to the number of the hidden neurons. From our experiments, the compressed images with good quality depend on the quantity of the non-recurrent three-layer neural networks (NRTNNs) so the algorithm might be used many units of the NRTNNs to acquire the lower image dimensionality as shown in Fig. 5. Each of the NRTNNs must be trained by the weight updated rule that already discussed in prior section until reaching the criteria. As a result, the hidden neurons of the NRTNN unit in the latest iteration of the neural network are defined as the output, called output1. Then, this output will be fed into the next NRTNN unit to reduce the lower image dimension. The latest output of the NRTNNs unit is the image compression with auto encoder.

Finally, the compressed images with auto encoder will be decoded by the multi-layer feed forward neural network. The construction of this neural network is shown in Fig. 6.

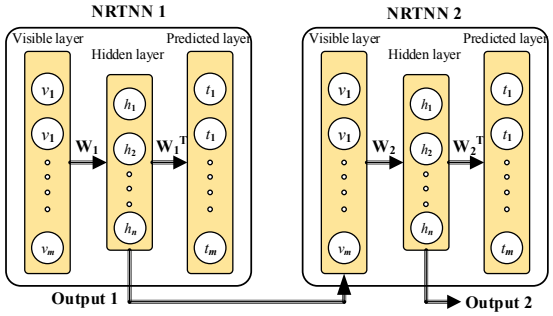

Figure 5. The applications of the non-recurrent three-layer neural networks (NRTNNs) in the multi-compression operations.

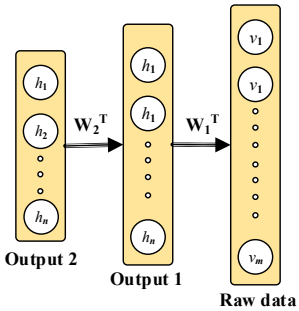

Figure 6. The block diagram of the decoded neural network.

III. EXPERIMENTS AND RESULTS

A. Experiment Setup

The aim of the experiments is to reduce a large size of the image dimensionality. Then, we use the MNIST data that are the handwritten digit recognition benchmark to test the proposed algorithm. It consists of the 60,000 samples which are 28x28 gray scale pixels with binary threshold of 127. The experiments are simulated by the Matlab program. In the simulations, we use three units of the non-recurrent three-layer

neural networks (NRTNNs) to reduce the dimension of the image. Then, the pixels of the 28x28 of the raw image data is prepared for the visible layer which are set to 784 neurons. These visible neurons are mapped to the hidden layer using extended Kalman filter (EKF) training algorithm that minimize the errors between the predicted layer and the visible layer. The quantity of the hidden neurons for the first NRTNN, second NTRNN and third NRTNN are set to 196 neurons, 49 neurons, and 10 neurons, respectively, as shown in Fig. 7. Then, each of the NRTNN outputs will be trained until reaching the optimal hidden neurons. After that, these optimal hidden neurons are fed into the next NRTNN unit to secondly reduce the image dimensionality. Finally, the optimal hidden neurons at the latest NRTNN will be used for next applications such as the image compression, the image communication or the image visualization etc.

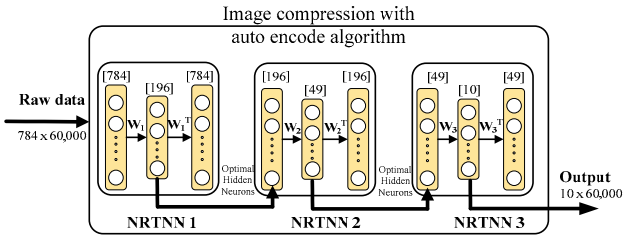

Figure 7. The block diagram of simulation for image compression with auto encoder algorithm using three units of the NRTNNs.

In case of the image recall, the compressed images will be converted back to the original images by using the image decoding method that already discussed in the previous section. The decoding method uses a multi-layer neural network. Since, the neural network layer depends on the number of the NRTNN unit so the construction of the image decoding neural network is shown in Fig. 8.

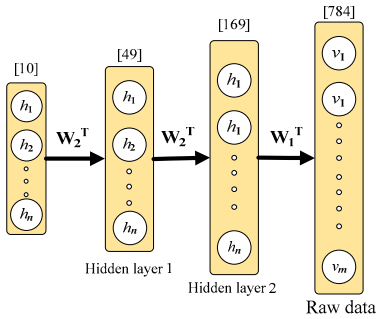

Figure 8. The multi-layer feed forward neural network for an image decoding algorithm.

B. *Experimental Results*

The experimental results of the image compression with auto encoder are shown in Fig. 9. The raw data of the handwritten digit recognition image of the Arabic number "7" is compressed by the proposed algorithm. As a result, it clearly sees that the data dimensionality of the output images is reduced from 784 points to 10 points (or 163,618 points including the weights). Although the proposed algorithm must keep a large points of the weights of the NRTNNs, but it satisfies for a large data samples. For example, in this experiment, it use the 60,000 image data sets. Therefore, the dimension of the output compressed images is 763,111 points while the raw image data is 47,040,000 points. In other word, the proposed algorithm can reduce around 60 times of the image dimensionality. Furthermore, as the proposed algorithm uses an extended Kalman filter algorithm to update the weights of the NRTNNs, so each of the NRTNN units will speedily converges to minimum error as shown in Fig. 10. It clearly sees that the algorithm uses only around 50 iterations to reach the optimal weights. In the case of the image decoding, the algorithm which uses a multi-layer neural network to obtain the raw image data. The result of the experiment is shown in Fig. 11.

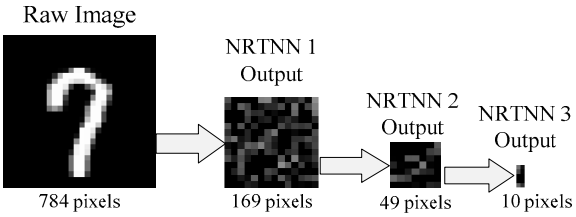

Figure 9. Results of the image compression with auto encoder.

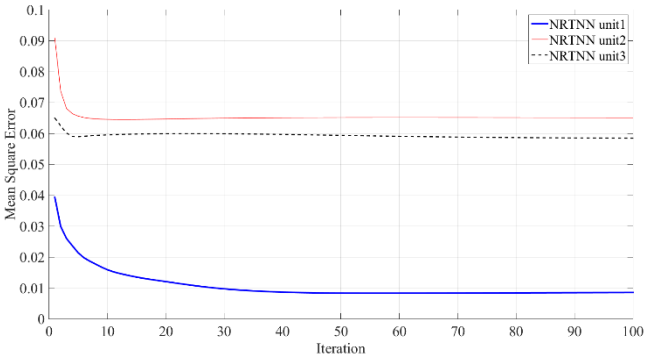

Figure 10. Mean square error of the compression algorithm.

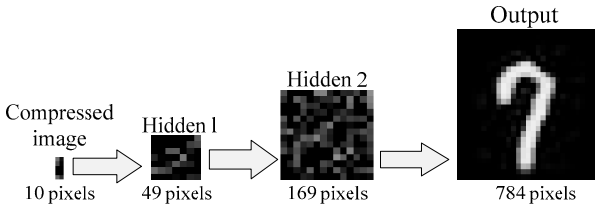

Figure 11. Results of the image decoding method.

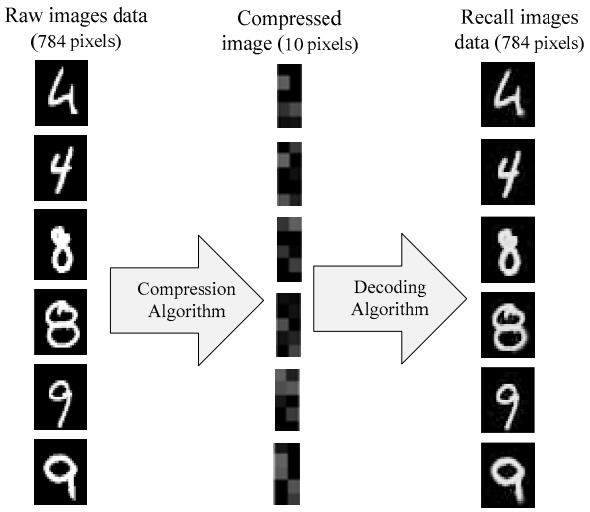

Figure 12. Practical example of the mapping between raw image data and decoded image by using the proposed algorithm.

In addition, we demonstrate the proposed image compression algorithm in the others handwritten digital number as shown in Fig. 12.

IV. CONCLUSIONS

The image compression algorithm is proposed by using the deep neural network (DNN) algorithm. The proposed algorithm does not only reduce the image dimension but also automatically encode the image. The image compression with auto encoder algorithm consists of two-part; image compression and image decoder. The image compression uses the non-recurrent three-layer neural networks (NRTNNs) to train the data sets. The NRTNN will update the weights to minimize the error of the proposed algorithm by using an extended Kalman filter (EKF). The learning EKF algorithm can optimize the updated weights to improve the speed of the error convergence. From the experimental results, it clearly sees that proposed image compression with auto encoder algorithm is able to reduce the image dimensionality and is able to recall back encoded image to original image with low loss.

In the future work, we will apply the proposed algorithm to the other image patterns or other image scenes.

REFERENCES

[1] Chien-I Chang and Q. Du, "Interference and noise-adjusted principal components analysis," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 5, pp. 2387-2396, Sept. 1999.

[2] R. C. Runkle, M. F. Tardiff, K. K. Anderson, D. K. Carlson and L. E. Smith, "Analysis of spectroscopic radiation portal monitor data using principal components analysis," *in IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 1418-1423, June 2006.

[3] S. Shokralla, J. E. Morelli and T. W. Krause, "Principal Components Analysis of Multifrequency Eddy Current Data Used to Measure Pressure Tube to Calandria Tube Gap," *in IEEE Sensors Journal*, vol. 16, no. 9, pp. 3147-3154, May1, 2016.

[4] G.E. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. ,504-507, 2006.

[5] T. Orłowski, "Application of deep belief networks in image semantic analysis and lossy compression for transmission," *Signal Processing Symposium (SPS)*, 2013, Serock, 2013, pp. 1-5.

[6] C. Dong, Y. Deng, C. C. Loy and X. Tang, "Compression Artifacts Reduction by a Deep Convolutional Network," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 576-584.

[7] S. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning", *Pattern Recognition*, in press, vol. 58, pp. 121-134, 2016.

[8] Y. LeCun, L.Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE 86 (11)* (1998), pp. 2278–2324.

[9] S. Haykin, Kalman filtering and neural networks. New York: Wiley, 2001.

[10] D. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.