Comparative Analysis of Restricted Boltzmann Machine Models for Image Classification

Chapter · March 2020

DOI: 10.1007/978-3-030-42058-1_24

4 authors, including:

Christine Dewi
Universitas Kristen Satya Wacana

Rung-Ching Chen
Chaoyang University of Technology

Hendry Hendry
Universitas Kristen Satya Wacana

Comparative Analysis of Restricted Boltzmann Machine Models for Image Classification

Christine Dewi[12], Rung-Ching Chen[*], Hendry[2] and Hsiu-Te Hung[1]

[1*]Department of Information Management, Chaoyang University of Technology
Taichung, Taiwan, R.O.C.
[2]Faculty of Information Technology, Satya Wacana Christian University
Salatiga, Indonesia
{s10714904, [*]crching}@cyut.edu.tw
hendry@uksw.edu
a0970783520@gmail.com

Abstract. Many applications for Restricted Boltzmann Machines (RBM) have been developed for a large variety of learning problems. Recent developments have demonstrated the capacity of RBM to be powerful generative models, able to extract useful features from input data or construct deep artificial neural networks. In this work, we propose a learning algorithm to find the optimal model complexity for the RBM by improving the hidden layer. We compare the classification performance of regular RBM use *RBM()* function, classification RBM use *stackRBM()* function and Deep Belief Network (DBN) use *DBN()* function with different hidden layer. As a result, Stacking RBM and DBN could improve our classification performance compare to regular RBM.

Keywords: Classification Comparison, DBN, RBM, stack-RBM.

1 Introduction

Deep learning has gained its popularity recently as a huge probabilistic models and way of learning complex. Deep neural networks are characterized by the large number of layers of neurons and by using layer-wise unsupervised pre-training to learn a probabilistic model for the data. A deep neural network is typically constructed by stacking multiple Restricted Boltzmann Machines (RBM) so that the hidden layer of one RBM becomes the visible layer of another RBM. Layer-wise pre-training of RBM then facilitates finding a more accurate model for the data. RBM have been particularly successful in classification problems either as feature extractors for text and image data [1] or as a good initial training phase for deep neural network classifiers [2]. However, in both cases, the RBMs are merely the first step of another learning algorithm, either providing a preprocessing of the data or an initialization for the parameters of a neural network.

The main contributions of this work can be summarized as follows: First, we propose a learning algorithm to find the optimal model complexity for the RBM by improving the hidden layer. Second, we will compare the classification performance of regular RBM use *RBM()* function, classification RBM use *stackRBM()* function and

Deep Belief Network (DBN) use *DBN()* function with different hidden layer. The rest of the paper is organized as follows. In Section 2, we describe brief explanation about the RBM. Section 3 describes the proposed experimental improvement for RBM. In Section 4, we present experimental results, and finally, Section 5 concludes this paper and suggests a future work.

2 Related Work

2.1 Restricted Boltzmann Machines (RBM)

RBM are undirected graphs and graphical models belonging to the family of Boltzmann machines, they are used as generative data models [3]. RBM can be used for data reduction and can also be adjusted for classification purposes [4]. They consist of only two layers of nodes, namely, a hidden layer with hidden nodes and a visible layer consisting of nodes that represent the data. The discriminate RBM was proposed by Larochelle [4], [5], which uses class information as visible input, so that RBM can provide a self-contained framework for deriving a non-liner classifier. The discriminate RBM model the joint distribution of the inputs and associated target classes, whose graphical model is illustrated in Figure 1 [5].

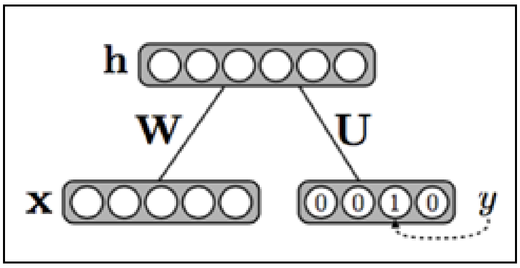


Fig. 1. Discriminative RBM [4] [5]

RBM consists of visible units *v*, binary hidden unit's *h* and symmetric connections between visible units and hidden units. The connections are represented by a weight matrix *W*. RBM uses the energy function for the probabilistic semantics. The energy function is described as follow: [6], [7], [12].

$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_i \sum_j v_i W_{ij} h_j - \sum_j b_j h_j - \sum_j c_i v_i \quad (1)$$

where b_j are biases of hidden units and c_i are biases of visible units. This energy function is used to configure a probability model for RBM. *W* is the weight matrix, *v* and *h* represent the visible and hidden layers. a and b are the bias of the visible and hidden layers. When the visible unit state is determined, each hidden element activation state is conditional independent to others. The j^{th} hidden element activation probability is denied as following: [7], [13].

$$P(h_j = 1|v,\theta) = \sigma(b_j + \sum_i v_i w_{ij}) \quad (2)$$

When the hidden element state is determined, the activation state of each visible element is also independent of each other. The probability of i^{th} visible unit is defined as following: [7], [12], [13].

$$P\left(v_i = 1 \middle| h, \theta\right) = \sigma(a_i + \textstyle\sum_i w_{ij} h_j) \tag{3}$$

2.2 *Stack RBM*

In general, stacking RBM is only used as a greedy pre-training method for training a Deep Belief Network as the top layers of a stacked RBM have no influence on the lower level model weights. However, this model should still learn more complex features than a regular RBM. We stack some layers of RBM with the stackRBM function, this function calls the RBM function for training each layer and so the arguments are not much different, except for the added layers argument. With the layers' argument we can define how many RBM you want to stack and how many hidden nodes each hidden layer should have. The stack RBM architecture is showed in Figure 2.

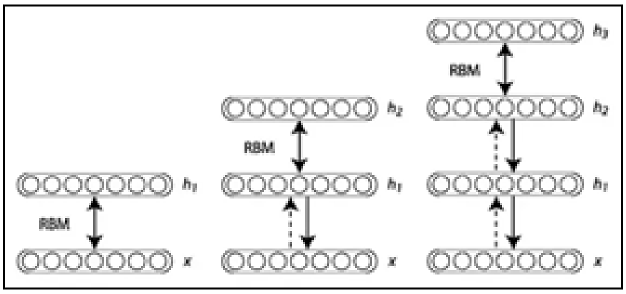


Fig. 2. The Stack RBM Architecture

2.3 *Deep Belief Network (DBN)*

Deep Belief Network (DBN), as shown in Figure 3, is a deep architecture built upon RBM to increase its representation power by increasing depth. In a DBN, two adjacent layers are connected in the same way as in RBM. The network is trained in a greedy, layer-by-layer manner [6], where the bottom layer is trained alone as an RBM, and then fixed to train the next layer. DBN was originally developed by Hinton et al. (1995) [8] and was originally trained with the sleep-wake algorithm, without pre-training. However, in 2006 Hinton et al. found a method that is more efficient at training DBNs by first training a stacked RBM and then use these parameters as good starting parameters for training the DBN [9]. The DBN then adds a layer of labels at the end of the model and uses either back propagation or the sleep-wake algorithm to fine tune the system with the labels as the criterion. The *DBN()* function in the RBM package uses the backpropagation algorithm. The backpropagation algorithm works as follows: (1) first a feedforward pass is made through all the hidden layers ending at the output layer (2) then the output is compared to the actual label and (3) the error is used to adjust the weights in all the layers by going back through the whole system.

This process is repeated until some stopping criterion is reached, in the *DBN()* function that is the maximum number of epochs but it could also be the prediction error on a validation set.

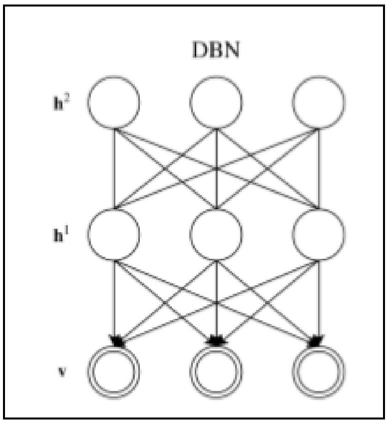


Fig. 3. The DBN Architecture.

3 Methodology

This paper use Modified National Institute of Standards and Technology database (MNIST dataset) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning [10][14][15]. The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from National Institute of Standards and Technology (NIST). The digits have been size-normalized and centered in a fixed-size image.

In this work we use various type of hidden layer. We raise the nodes in the hidden layer for each model. The configurations of nodes in hidden layer are 50, 100, 150, 200, 250, 300, 350, and 400. We also combine different layer to improve the classification performance of RBM. Moreover, we use 2 and 3 layers for stack RBM and DBN. We will compare the classification performance of regular RBM using RBM function, classification RBM using *stackRBM* function and DBN function with different hidden layer. The *n.hidden* argument defines how many hidden nodes the RBM will have and *size.minibatch* is the number of training samples that will be used at every epoch. For each model we use 1000 as the number of iterations and 10 for the minibatch. The workflow of this research could be seen on Figure 4.

Furthermore, after training the RBM model, *stackRBM* model and DBN model we can check how well it reconstructs the data with the *ReconstructRBM* function. The function will then output the original image with the reconstructed image next to it. If the model is good, the reconstructed image should look similar or even better than the original. RBM not only good at reconstructing data but can actually make predictions on new data with the classification RBM. So, after we trained our regular RBM, classification RBM and DBN, we can use it to predict the labels on some unseen test data with the *PredictRBM* function. This function will output a confusion matrix and the accuracy score on the test set.

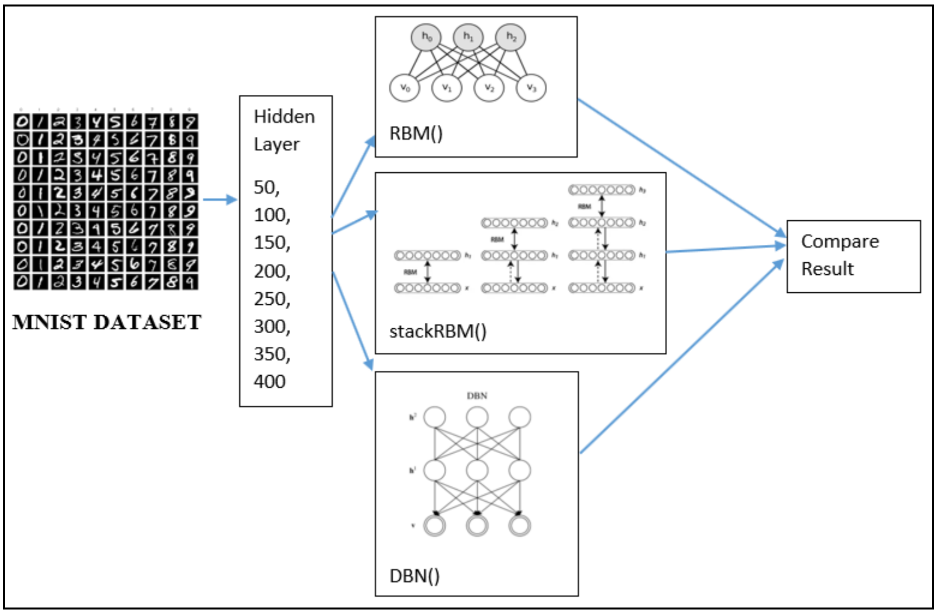


Fig. 4. The Workflow of the Research.

4 Experiment and Result

We evaluate the performance of the proposed learning algorithm using the MNIST dataset [10]. In the classification results, we focused on whether the experiment improvement RBM obtained the best classification accuracy performance. Also, we compared the number of hidden neurons RBM. The classifier used in all the experiment is the Back-Propagation Network (BPN) [11][15].

Table 1. Classification Accuracy of MNIST Dataset with Various Type of Hidden Layer using RBM Function.

n.iter	n.hidden	Accuracy
1000	50	0.8245
1000	100	0.846
1000	150	0.851
1000	200	0.8585
1000	250	0.859
1000	300	0.86
1000	350	0.86
1000	400	0.815

Table I shows the classification accuracy of MNIST dataset with various type of hidden layer using RBM function. In this experiment we use 50, 100, 150, 200, 250, 300, 350, and 400 nodes in hidden layer. In addition, to train a RBM we need to provide the function with train data, which should be a matrix of the shape *(samples * features)* other parameters have default settings. The number of iterations defines the number of training epochs, at each epoch RBM will sample a new minibatch. When we have enough data it is recommended to set the number of iterations to a high value as this will improve our model and the downside is that the function will also take longer to train. The *n.hidden* argument defines how many hidden nodes the RBM will have and *size.minibatch* is the number of training samples that will be used at every epoch. We use 1000 as the number of iterations and 10 for the minibatch. Moreover, the highest accuracy is 86% with 350 nodes in hidden layer.

After training the RBM model we can check how well it reconstructs the data with the *ReconstructRBM* function. The function will then output the original image with the reconstructed image next to it. If the model is any good the reconstructed image should look similar or even better than the original. The reconstruction model for digit ”0” and digit ”3” using RBM function could be seen on Figure 5 and Figure 6 The model reconstruction looks even more like a three and zero than the original image. Furthermore, RBM not only good at reconstructing data but can actually make predictions on new data with the classification RBM. After we trained our classification RBM we can use it to predict the labels on some unseen test data with the *PredictRBM* function. Which should output a confusion matrix and the accuracy score on the test set that could be seen on Figure 7.

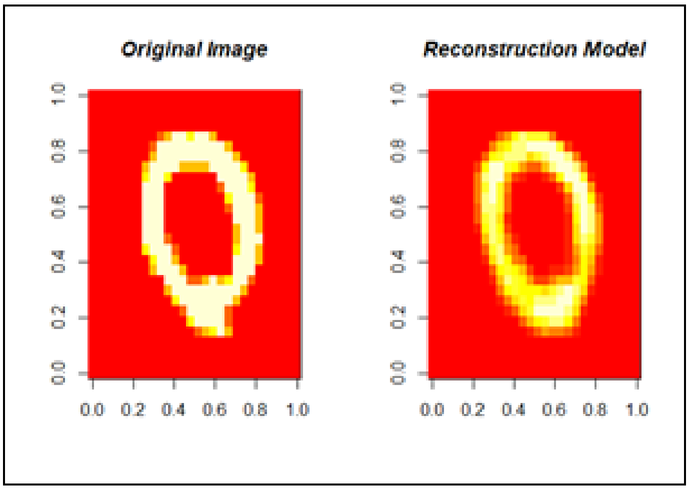


Fig. 5. Reconstruction Model Digit ”0” using RBM Model

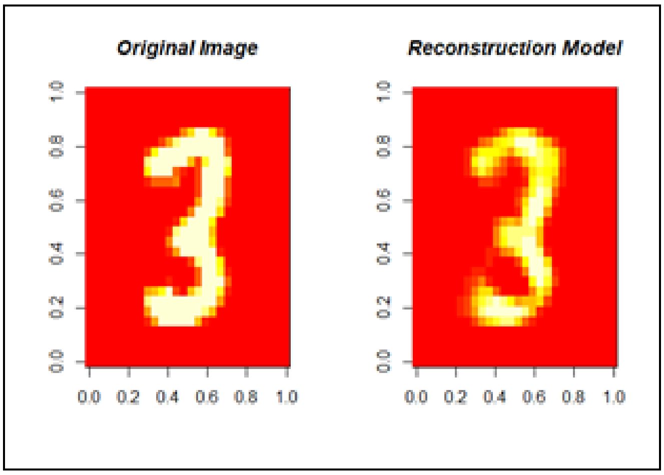
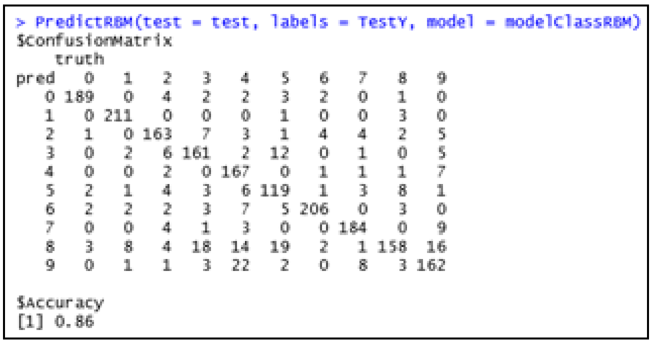


Fig. 6. Reconstruction Model Digit ”3” using RBM Model



```
> PredictRBM(test = test, labels = TestY, model = modelClassRBM)
$ConfusionMatrix
    truth
pred   0   1   2   3   4   5   6   7   8   9
   0 189   0   4   2   2   3   2   0   1   0
   1   0 211   0   0   0   1   0   0   3   0
   2   1   0 163   7   3   1   4   4   2   5
   3   0   2   6 161   2  12   0   1   0   5
   4   0   0   2   0 167   0   1   1   1   7
   5   2   1   4   3   6 119   1   3   8   1
   6   2   2   2   3   7   5 206   0   3   0
   7   0   0   4   1   3   0   0 184   0   9
   8   3   8   4  18  14  19   2   1 158  16
   9   0   1   1   3  22   2   0   8   3 162

$Accuracy
[1] 0.86
```

Fig. 7. Confusion Matrix MNIST dataset using RBM Model

Table 2. Classification Accuracy of MNIST Dataset with Various Type of Hidden Layer using stackRBM Function.

n.iter	n.hidden	layers	Accuracy	layers	Accuracy
1000	50	2	0.846	3	0.827
1000	100	2	0.8695	3	0.868
1000	150	2	0.889	3	0.901
1000	200	2	0.8885	3	0.897
1000	250	2	0.9015	3	0.899
1000	300	2	0.9015	3	0.8975
1000	350	2	0.909	3	0.9165
1000	400	2	0.907	3	0.906

Table II shows classification accuracy of MNIST dataset with various type of hidden layer using stackRBM function. In this experiment we use various type of hidden layer consists of 50, 100, 150, 200, 250, 300, 350, and 400 nodes for each layer (2 and

3). In this work the highest accuracy for 2 layers is 90.9% use 350 nodes in hidden layer and for 3 layers is 91.65% use 350 nodes in hidden layers. As we can see on Table II stacking RBM use 350 nodes in hidden layer receive 90.9% accuracy and it is higher than on Table I normal RBM receive 86% for 350 nodes in hidden layer. We can conclude from this result stacking RBM improves our classification performance. However, stackRBM is not a very elegant method though as each RBM layer is trained on the output of the last layer and all the other RBM weights are frozen. It is a greedy method that will not give us the most optimal results for classification. After training the stackRBM model we can check how well it reconstructs the data with the *ReconstructRBM* function. The reconstruction model for digit ”0” and digit ”3” could be seen on Figure 8 and Figure 9. Figure 10 explain about confusion matrix MNIST dataset using stackRBM function with 350 nodes in hidden layer using 2 hidden layers, which got 90.9% accuracy.

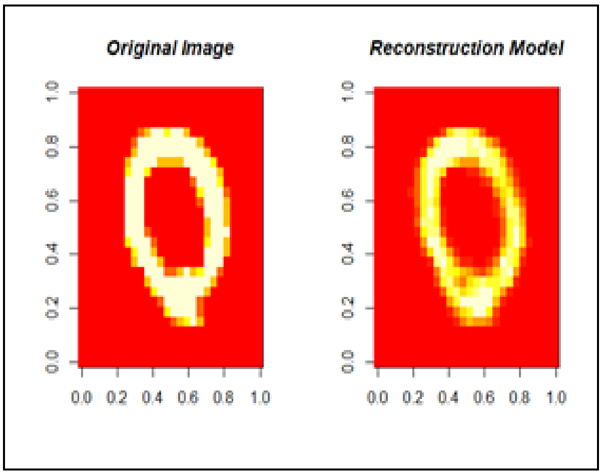


Fig. 8. Reconstruction Model Digit ”0”, 2 Layers using stackRBM Model

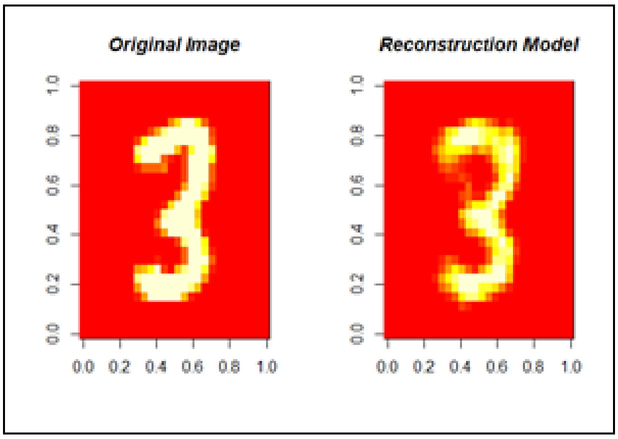


Fig. 9. Reconstruction Model Digit ”3”, 2 Layers using stackRBM Model

```
> PredictRBM(test = test, labels = TestY, model = modStackSup, layers =
 2)
$ConfusionMatrix
    truth
pred   0   1   2   3   4   5   6   7   8   9
   0 189   0   3   0   0   2   0   0   0   2
   1   0 222   0   5   2   3   0   5  10   6
   2   0   0 163   4   0   0   1   2   1   1
   3   0   1   4 168   0   0   0   0   2   2
   4   0   0   2   0 212   2   1   1   2   7
   5   1   0   1  12   0 142   3   0   5   1
   6   3   1   3   4   1   9 211   0   4   0
   7   0   0   5   1   1   1   0 183   2   3
   8   3   1   7   1   0   2   0   0 146   1
   9   1   0   2   3  10   1   0  11   7 182

$Accuracy
[1] 0.909
```

Fig. 10. Confusion Matrix MNIST dataset using stackRBM Model

```
> PredictDBN(test = test, labels = TestY, model = modDBN, layers = 3)
$ConfusionMatrix
     truth
Preds   0   1   2   3   4   5   6   7   8   9
    0 191   0   4   1   1   2   3   0   3   1
    1   0 206   0   0   0   1   0   0   3   0
    2   0   0 169   2   5   0   3   1   3   1
    3   0   1   2 173   0   5   0   1   4   3
    4   0   0   3   0 199   1   1   0   0   5
    5   2   5   4  10   1 140   5   1   4   4
    6   2   1   0   2   2   3 202   0   1   0
    7   0   3   4   3   2   3   1 198   2  24
    8   2   9   3   6   7   7   1   0 159   3
    9   0   0   1   1   9   0   0   1   0 164

$Accuracy
[1] 0.9015
```

Fig. 11. Confusion Matrix MNIST dataset using DBN Model

Table 3. Classification Accuracy of MNIST Dataset with Various Type of Hidden Layer using DBN Function.

n.iter	n.hidden	layers	Accuracy	layers	Accuracy
1000	50	2	0.8485	3	0.8215
1000	100	2	0.8845	3	0.8665
1000	150	2	0.8905	3	0.8865
1000	200	2	0.8895	3	0.899
1000	250	2	0.8675	3	0.889
1000	300	2	0.902	3	0.8985
1000	350	2	0.9015	3	0.8945
1000	400	2	0.89	3	0.90

Table III shows classification accuracy of MNIST dataset with various type of hidden layer using DBN function. In this experiment we use various type of hidden layer consists of 50, 100, 150, 200, 250, 300, 350, and 400 nodes for each layer (2 and 3). In this work the highest accuracy for 2 layers is 90.15% use 350 hidden layer and for 3 layers is 90% use 400 nodes in hidden layers.

Figure 11 explain about confusion matrix MNIST dataset using DBN function with 350 nodes in hidden layers, for 2 layers and 1 label layers, which got 90.15% accuracy. Based on experiment result on the Figure 12 the trends of accuracy increases. When we use more nodes in hidden layer, we can get higher accuracy performance. In this work the highest accuracy is obtained when using 350 nodes in hidden layer. After that the accuracy performance relatively decrease.

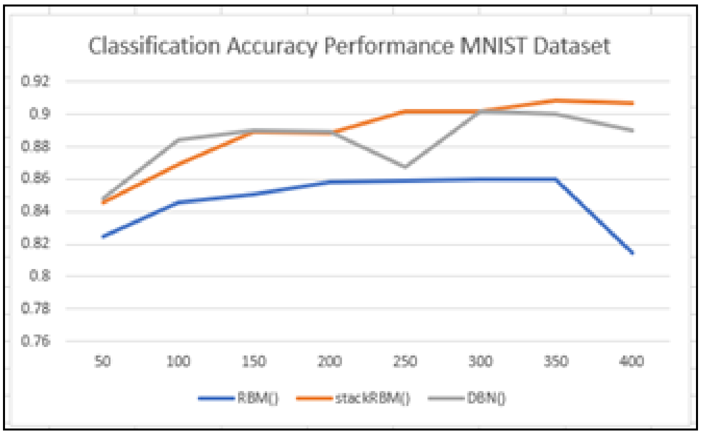


Fig. 12. Classification Accuracy Performance MNIST Dataset with Various Type of Hidden Layer.

5 Conclusions

Based on all experiment result on Table I, Table II, and Table III the number of hidden units and the key parameter of restricted Boltzmann machine play an important role in the modeling capability. Too many hidden units lead to a large model size and slow convergence speed, even overfitting results in poor generalization ability. And too few hidden units result in low accuracy and bad performance of feature extraction. Stacking RBM and DBN could improve our classification performance compare to regular RBM. Our experiment was focused on comparing the number of hidden neurons use RBM function, stackRBM function, and DBN function. Our future work includes to design a fully automated incremental learning algorithm that can be used in the deep architecture and we will use other advanced types of RBM like Gaussian RBM and Deep Boltzmann Machines.

ACKNOWLEDGMENT

This paper is supported by Ministry of Science and Technology, Taiwan. The Nos are MOST-107-2221-E-324 -018 -MY2 and MOST-106-2218-E-324 -002, Taiwan.

References

1. Gehler, P. V., Holub, A. D., and Welling, M. "The rate adapting Poisson model for information retrieval and object recognition," *in Proceedings of the 23rd International Conference on Machine Learning, ser.* ICML'06. New York, NY, USA: ACM, pp. 337–344. (2006). [Online]. Available: http://doi.acm.org/10.1145/1143844.1143887
2. Hinton, G. E.: "To recognize shapes, first learn to generate images." *Progress in brain research*, vol. 165, pp. 535–547, (2007).
3. Hinton, G. E.: *A Practical Guide to Training Restricted Boltzmann Machines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8 32
4. Larochelle, H., Mandel, M., Pascanu, R., and Bengio, Y. "Learning algorithms for the classification restricted boltzmann machine," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 643–669, (2012)
5. Larochelle, H., and Bengio, Y.: "Classification using discriminative restricted boltzmann machines," *in Proceedings of the 25th International Conference on Machine Learning, ser. ICML '08.* New York, NY, USA: ACM, 2008, pp. 536–543. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390224, (2008).
6. Jongmin, Y., Jeonghwan, G., Sejeong, L., and Moongu, J., "An incremental learning approach for restricted boltzmann machines," *in 2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 113–117, (2015).
7. Jiang, Y., Xiao, J., Liu, X., and Hou, J.: "A removing redundancy restricted boltzmann machine," *in 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pp. 57–62, (2018).
8. Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M.: "The wake-sleep algorithm for unsupervised neural networks*." Science*, vol. 268, p. 1158, (1995).
9. Hinton, G. E., Osindero, S., and Teh, Y.W.: "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1162/neco.2006.18.7.1527, (2006).
10. Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P.: "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, (1998).
11. Cun, Y. L., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., and Henderson, D.: "Advances in neural information processing systems 2," D. S. Touretzky, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Handwritten Digit Recognition with a Back-propagation Network, pp. 396–404. [Online]. Available: http://dl.acm.org/citation.cfm?id=109230.109279
12. Han-Gyu, K., Seung-Ho, H., Ho-Jin, C.: "Discriminative restricted Boltzmann machine for emergency detection on healthcare robot", *in 2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, pp. 407-409. (2017).
13. Jinyong, Y., Tao, S., Jiangyun, Z., Xiaolon, B.: "Fault Prognosis Based on Restricted Boltzmann Machine and Data label for Switching Power Amplifiers", *in 2018 12th International Conference on Reliability, Maintainability, and Safety (ICRMS),* IEEE, pp. 287-291. (2018).
14. Renshu, W., Bin, C., Jingdong, G., Jing, Z.: "The Image Recognition Based on Restricted Boltzmann Machine and Deep Learning Framework", *in 2019 4th International Conference on Control and Robotics Engineering (ICCRE),* IEEE, pp. 161-164. (2019).
15. Shamma, N., Justine, L. D., Supriyo, B., Amit, R. T.,: "Low Power Restricted Boltzmann Machine Using Mixed-Mode Magneto-Tunneling Junctions", *IEEE Electron Device Letters*, vol. 40, no. 2, pp. 345-384, (2019).