A comparison study of three single-solution based metaheuristic optimisation for stacked auto encoder

L M Rasdi Rere¹, Bheta Agus Wardijono², Yudi Irawan Chandra³ and Latifah⁴
STMIK Jakarta STI&K, Jakarta 12140, Indonesia

Email: rasdirere267@gmail.com¹

Abstract. Deep learning has been effectively used in a variety of application such as audio processing, phonetic recognition, robotic, information retrieval and even analysis of molecules. However to train deep learning is interesting yet challenging. A layer wise pre-training, drop-connect, Hessian-free optimization, and Krylov suspense descent are amongst the successful technique or methods proposed in training it. Recently, some of metaheuristic algorithms have been used to optimize Deep learning, especially Convolutional neural network using Genetic algorithm, Particle swarm optimization, Harmony search, and Simulated annealing. In this paper, three type of Single-solution metaheuristic have been proposed, i.e. Simulated annealing, Macrocanonical annealing or Threshold accepting method to optimize Stacked autoencoder, one of the famous Deep learning. The result of experiment conducted to MNIST dataset show that the proposed method can improve accuracy at the rank of 0.07% up to 12.13% for the first time epoch, although there is an increase in computation time.

1. Introduction
Deep learning is a new research area in machine learning, motivated by the field of artificial intelligence, which aims to mimic the human brain’s ability to observe, study, analyze and make a particular decision [1]. Basically this method is about learning a hierarchical representation of the feature, which can consist of a variety of methods such as neural network, hierarchical probabilistic models, as well as learning algorithms. A significant development at this time on deep learning is due to increase the processing of computational abilities on a chip, decrease in the price of computer hardware, and fairly rapid progress in the machine learning algorithms [2].

The method of Deep learning is usually categorized into the methods of generative, discriminative and hybrid. Some examples of generative models are Deep belief network (DBN), Deep Boltzmann machine (DBM) and Stacked autoencoder (SAE). Discriminative model for instance is Deep neural network, Convolutional neural network (CNN) and Recurrent neural network. In case of a hybrid model, it is the combination of generative and discriminative. An example of this model is DBN that is used to pre-train deep CNN [2].

Other forms of Deep learning classification can also be based on the methods that compose them, such as GoogLeNet, VGG, and AlexNet which are types of DL based on the CNN method. Deep energy model, DBN and DBM based on the method of Restricted boltzmann machines. SAE, Contractive auto encoder, and Stacked denoising autoencoder based on AE method, as well as Laplacian sparse coding, Local coordinate coding and Sparse coding spatial pyramid matching based on Sparse coding method [3].
Although the Deep learning method has a good reputation for solving various problems of learning, to train this method is not easy. A successful technique to train Deep learning is pre-training as proposed by Hinton and Salakhutdinov [4]. Some other technique used a Hessian-free optimization proposed by Marten [5] and Krylov Suspenion Descent by Vinyal et.al. [6].

In recent years, a number of metaheuristic algorithms have been used to further improve the accuracy of Deep learning, especially CNN. You and Pu [7] used Genetic algorithm (GA), the first generation of metaheuristic, to optimize CNN based on a random sample. Their strategy is to select the characteristic of CNN using the process of recombination and mutation, where a number of CNN is as individual in the GA algorithms. Besides, in the process of recombination, only threshold values and weights layers on the first convolution layer and third convolution layer that are used in this model.

Another work is conducted by Rosa et al [8] using Harmony search to fine-tune CNN, Vina et al [9] used Macrocanonical annealing to optimize CNN, and Fedorovici et al [10] used Particle swarm optimization (PSO) and Gravitational search algorithm (GSA) to select the best parameters on CNN, which is used in optical character recognition applications.

In this paper, the performance of three single-solution based metaheuristic algorithms, i.e. Simulated annealing (SA), Macrocanonical annealing (MA), and Threshold accepting method (TA) are compared, for optimizing Stacked autoencoder (SAE). They were selected amongst available techniques because SA is commonly used to optimize some variation of problems, while MA and TA are its variant. However, to the best of our knowledge, are not yet used to optimize SAE. The strategy employed in this technique is looking for the best value of the objective function on the last layer using SA, MA, or TA, then the results will be used again to calculate the weights and biases in the previous layer. To test the performance of the proposed method, the MNIST dataset (Mixed National Institute of Standards and Technology) was used. This dataset consists of digital handwritten images, containing 60,000 training data and 10,000 test data. All images are centered and standardized, with a size of 28 × 28 pixels. Each pixel image represented by 0 for black and 255 for white, and among them have different gray shades [11].

This paper is organized as follows: Section 1 is an introduction, section 2 gives description of three single-solution based metaheuristic, section 3 explains the method of stacked autoencoder, section 4 describes the proposed methods, section 5 presents the result of simulation, and Section 6 is the conclusion.

2. Single-solution based metaheuristic

Metaheuristic is a reliable method to solve difficult optimization problems, while it has been widely used in the field of science, engineering and even industry [12]. This method can solve a big problem with an effective algorithm, as well as relatively fast time. In addition, it is flexible, simple to design it and also reasonably easy to use in various fields of application [13].

In general, almost all metaheuristic algorithms inspired by natural phenomena, such as physics, biology and ethology. Metaheuristic algorithms based on physics phenomena for example are Simulated annealing (SA), Macrocanonical annealing (MA), and Threshold accepting method (TA). Based on biological phenomena are Genetic algorithm (GA), Evolution strategy, and Differential evolution (DE). As for the phenomena of ethology are Particle swarm optimization (PSO), Artificial immune systems, and ant Colony optimization [14]. Other phenomena is inspired by music, for example the algorithm of Harmony search [15].

Another classification on metaheuristic algorithms are Single-solution based metaheuristic (S-methheuristic), which is the focus of this research, and Population based metaheuristic (P-metaheuristic). Some examples of S-metaheuristic are SA, Guided local search, MA, TA, and Tabu search. While the P-metaheuristic are GA, DE, Bee colony optimization, and PSO [14]. The algorithms that are used in this paper S-metaheuristic algorithms that are used in this paper, i.e. SA, MA, and TA.
2.1. Simulated annealing

The algorithm of SA is a random search technique for the problem of global optimization. It mimics the process of annealing in material processing. This technique was firstly proposed in 1983 by Kirkpatriek, Gelatt and Vecchi [16].

The principle of idea of SA is using random search, which not only allows changes that improve the fitness function but also maintain some changes that are not ideal. As an example in minimum optimization problem, any better change that decrease the fitness function value \( f(x) \) will be accepted, but some change that increase \( f(x) \) will also be accepted with a transition probability \( p \), as equation (1) follow:

\[
p = e^{-\Delta E / kT}
\]

Where \( \Delta E \) is the energy level changes, \( k \) is the Boltzmann’s constant, and \( T \) is temperature for controlling the process of annealing. This equation is based on the Boltzmann distribution in physics [12]. The following is the standard procedure of SA for optimization problems:

1. Generate the solution vector: the initial solution vector is selected randomly, followed by computing the objective function
2. Initialize the temperature: if the temperature value is too high, it needs a long time to achieve a convergence, while too small value will cause the system missing the global optimum.
3. Select a new solution: a new solution is randomly selected from the neighborhood of the current solution
4. Evaluate a new solution: depending on its fitness function, a new solution is accepted as a new current solution.
5. Decrease the temperature: the temperature is periodically decreased, during the search process of solution.
6. Stop or repeat: when the termination criterion is satisfied, the computation is stopped, otherwise, step 2 until 6 are repeated until it reaches satisfactory.

2.2. Macro canonical annealing

MA algorithm is one of the well-known variant of SA. However unlike SA, this method is based on Creutz algorithm, known as Macrocanonical monte carlo simulation or “Demon” algorithm. This algorithm is simulation of isolated system at the equilibrium thermodynamic and in this condition, the total energy of the system is constant [14]. The total energy of the system \( (E_{\text{total}}) \) as is shown in the following equation (2), is the sum of potential energy \( (E_p) \) and kinetic energy \( (E_k) \):

\[
E_{\text{total}} = E_p + E_k
\]

In case of a minimum optimization problem, the potential energy is a function of the objective to be minimized value, while the kinetic energy is the function like the temperature in the SA algorithm, which was forced to keep a positive value [14]. If the energy change in the system is negative \((-\Delta E)\), then the kinetic energy will increase \((E_k \leftarrow E_k - \Delta E)\), and a new situation is accepted. However, if the energy change in the system is positive \((+\Delta E)\), then a new situation is acceptable depending on the kinetic energy. If the change in energy is smaller than the kinetic energy \((\Delta E < E_k)\), a new situation is also acceptable, but if not, then rejected.

2.3. Threshold accepting method

TA algorithm is proposed by Dueck and Scheuer in 1990 [17], as a deterministic algorithm, while it does not depend on some probability. The number of steps in which the solution will improve by neighborhood is fixed. The threshold will be decrease gradually up to a value of zero in accordance with the number steps provided.
Acceptance criteria for candidate of solutions is the main different between TA and SA. The candidate of solution is accepted by TA, if degradation of objective function does not exceed a certain threshold value. On the other hand, a solution candidate can acceptable by SA, although the value of the objective function does not qualify, with a certain probability.

3. Stacked autoencoder

Autoencoder is one of the special artificial neural network. This network is a discriminative graphical model that attempts to reconstruct its input signal. Suppose there is a set of data points \{x^{(1)}, x^{(2)} \ldots x^{(m)}\} as an input vector \(x\), where each data point has many dimensions. Autoencoder will take this vector \(x\), encodes it to a hidden layer \(h\), and decodes it to a reconstruction \(\hat{x}\), as is shown in figure 1. Systematically to map the data back and forth, \(h\) and \(\hat{x}\) are function of their inputs that are given by the following equations:

\[
\begin{align*}
  h^{(i)} &= W_1 x^{(i)} + b_1 \\
  \hat{x}^{(i)} &= W_2 h^{(i)} + b_2
\end{align*}
\]

As the goal is to have \(\hat{x}^{(i)}\) to approximate \(x^{(i)}\), the objective function can be set up by the following equation, which is the sum of squared differences between have \(\hat{x}^{(i)}\) and \(x^{(i)}\):

\[
f(W_1, b_1, W_2, b_2) = \sum_{i=1}^{m} (\hat{x}^{(i)} - x^{(i)})^2
\]

![Figure 1](image1.png)  
**Figure 1.** The basic principle of Autoencoder.

Objective function of this autoencoder can be minimized using stochastic gradient descent. Besides, as the activation function of hidden layer is linear, this network architecture is called linear autoencoder, which can be showed in figure 2. This architecture will map data from 4 dimension into 2 dimension using neural network with one hidden layer. In case of using nonlinear activation function like sigmoid, it is called nonlinear autoencoder, and if the data is highly nonlinear, more hidden layers can be added to a network to have a deep autoencoder [18].

![Figure 2](image2.png)  
**Figure 2.** Network architecture of Autoencoder.

One of important variants on autoencoder is Stacked autoencoder (SAE), that consist of some autoencoder stacked on top of each other, with some of the hidden neurons and each trained in a greedy-layer wise approach. The following is training procedure of SAE [19]:

1. The first layer is trained as autoencoder (AE), with aim to minimize the reconstruction error from input. This step is unsupervised learning
2. The outputs of hidden unit from AE are used as input for next layer, and also be trained to be AE, where unlabeled data are still needed.
3. Iterate as step 2 to initialize the desired number of additional layers.
4. Take the last hidden layer output as input for a supervised layer, and initialize its parameter.
5. Fine-tune all the parameters of this deep architecture with respect to supervised criterion. Alternately, unfold all AE into a very deep AE and fine-tune the global reconstruction error.

4. Design of proposed method

The scenario of design in this proposed method, is shown in figure 3. At the first time, the dataset used in the classification process is selected, in this case the MNIST dataset. Then the architectural structure of the SAE network is determined, including the initialization and set-up of some parameters. Architecture of SAE in this research, consist of three AE and one feed forward neural network.

The next stage is the SAE training which is carried out by determining all the weight values and biases in the layer with feed forward (FF), objective function (FO) and back propagation (BP) calculations. Weight and bias values in the last layer (x), then used as vector solutions by the SA method. By adding a number of Δx values that are obtained randomly, a number of neighboring values will be obtained which are candidate solutions (x + Δx).

The value of Δx is an important aspect, while selection the proper of this value, will significantly increase an accuracy of system. In case of one epoch as an example, if Δx = 0.0008 × rand, then the accuracy is 91.55%, in which this value is 12.34% greater than the original SAE (79.21%). However, if Δx = 0.00001 × rand, its accuracy only 85.51%. Furthermore, the solution vector from SAE will always be updated based on neighboring values, through a mechanism in the process of the SA, MA or TA methods.

In case of an objective function, the algorithms of SA, MA, or TA is used to train SAE to find the best accuracy condition, as well as to minimize estimated error and indicator of network complexity. This aim can realized by computing the objective function of vector solution or standard error on the training set. The following is the objective function used in this paper:

\[
f = \frac{1}{2} \left( \frac{\sum_{i=1}^{N} (o - y)^2}{N} \right)^{0.5}
\]

Where \( o \) is the expected output, \( y \) is the real output, and \( N \) is some training samples. For termination criterion, two situations are used in this method. The first is when the maximum iteration has been reached, and the second is when the objective function is less than a certain constant. Both condition

![Figure 3. Design scenario of proposed method.](image)
mean that the most optimal state has been achieved. When the termination criteria are reached, all weights and biases can be updated for all layers in the system. The final stage of the SAE optimization scenario with SA, MA or TA is testing the results of SAE training. The final result is the value of accuracy and computational time.

5. Experiment results
In this research, the classification of images is done using the MNIST dataset. Some examples of numerical handwritten images on MNIST datasets are shown in figure 4. The experiment was implemented in Matlab-R2010a, Windows 10, on a PC with processor Intel Core i7-4500u, and 8 GB Ram. The original program of this experiment is Deep Learn Toolbox by Palm [20].

![Figure 4. Example of some images from MNIST dataset.](image)

In this research, the SAE program is modified with algorithm SA, MA, or TA. The size of neighborhood in all experiment was set = 10, a maximum of iteration =10, and threshold value was set = 100. In addition, the learning rate $\alpha = 1$ and the batch size = 100.

![Figure 5. Accuracy vs epoch.](image)
Figure 6. Computation time vs epoch.

The first 10 epochs. The average results from SAE optimized by SA, MA, or TA, compared to the original SAE, are given in figure 5 for accuracy values (%) and figure 6 for computation time (second), for the first 10 epochs. In general, the experimental results show that the proposed method produces higher accuracy values than the original SAE for each given epoch. As an example for two epochs, the accuracy value of original SAE is 92.73%, while SAESA, SAEMA and SAETA are 93.41%, 93.33%, and 93.57% respectively. In case of computation time, the original SAE is generally better, because the proposed method basically adds the SA, MA, or TA algorithm to the original SAE. At two epochs, for example, the computation time of original SAE is 71 second, while the computation time of SAESA, SAEMA and SAETA are 99 second, 120 second, and 115 second respectively. Overall, the increase in accuracy given by the three proposed methods is similar. This is because the three methods, in principle, have the same way of working, single-solution.

The complete result of the accuracy value (A) and computation time (T) for the first 10 epochs is given in table 1. Increasing the accuracy value of the proposed method, compared to the original SAE varies for each epoch with a range of 0.07% (SAETA, 10 epoch) up to 12.13% (SAESA, 1 epoch). While the computation time for the proposed methods are compared with the original SAE in the range of 0.94× (SAESA, 9 epoch) up to 2.43× (SAEMA, 1 epoch).

Table 1. Accuracy and computation time for optimal SAE.

| Epoch | SAE     | SAESA  | SAEMA  | SAETA  |
|-------|---------|--------|--------|--------|
|       | A (%)   | T (s)  | A (%)  | T (s)  | A (%)  | T (s)  | A (%)  | T (s)  |
| 1     | 79.42   | 37     | 91.55  | 75     | 87.21  | 90     | 90.90  | 79     |
| 2     | 92.73   | 71     | 93.41  | 99     | 93.33  | 120    | 93.57  | 115    |
| 3     | 93.65   | 99     | 94.34  | 109    | 94.49  | 107    | 94.57  | 106    |
| 4     | 94.68   | 123    | 94.99  | 211    | 95.08  | 172    | 95.04  | 155    |
| 5     | 94.55   | 132    | 95.32  | 235    | 95.46  | 174    | 95.37  | 140    |
| 6     | 95.13   | 156    | 95.63  | 266    | 95.85  | 204    | 95.71  | 195    |
| 7     | 95.31   | 183    | 95.75  | 176    | 95.90  | 191    | 95.95  | 195    |
| 8     | 95.81   | 225    | 95.88  | 242    | 96.05  | 247    | 95.99  | 241    |
| 9     | 95.97   | 232    | 96.07  | 218    | 96.14  | 239    | 96.20  | 234    |
| 10    | 96.10   | 285    | 96.23  | 372    | 96.24  | 303    | 96.17  | 281    |

| 100 epoch | 97.71   | 2,148  | 97.73  | 2,790  | 97.72  | 3,234  | 97.72  | 2,564  |

100 epoch. In case of 100 epoch, the accuracy value of original SAE is already at 97.71%, with a computation time of 2,148 second. Thus, as expected, the accuracy value for SAESA, SAEMA and SAETA are slightly increased (97.73%, 97.72%, and 97.72%, respectively with longer computation time, which are 2,790 seconds, 3,234 seconds, and 2,564 seconds respectively. In our previous work, using the same MNIST data set, CNN was optimized by employing SA [9] and MA [21]. In fact, the
accuracy of CNN-SA and CNN-MA are higher than SAE coupled with SA, MA or TA. However, SAE technique has advantage in computation time. The calculation process takes shorter time than CNN.

This result shows that the three single-solution SAE techniques are potential to be used to increase accuracy (%A) when the original value is not that high (<90%). In terms of computing time, the results obtained are not very different from the original SAE. This is because the proposed method only optimizes the last layer of the SAE architecture, not for all layers.

6. Conclusion
This paper shows that the Single-solution based metaheuristic, i.e. SA, MA, and TA can improve the accuracy of SAE (0.007% - 12.13%) for MNIST dataset, compared to the original SAE. This result proves that the proposed method, with the strategy of optimizing the last layer of the SAE architecture for classification, can work well, as expected. To ensure the proposed method is powerful, in the future studies, application of these methods in face recognition, object detection, classification of character and others need to be explored.

References
[1] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R.Wald, and E.Muharemagic, “Deep learning applications and challenges in big data analytics,” Journal of Big Data, vol. 2, no. 1, pp. 1–21, 2015.
[2] L. Deng and D. Yu. Deep Learning: Methods and Application, Foundation and Trends in Signal Processing, Redmond, Wash, USA, 2013.
[3] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew, “Deep learning for visual understanding: A review”. Neurocomputing, 187:27–48, 2016.
[4] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no.5786, pp. 504–507, 2006.
[5] J. Martens, “Deep learning via hessian-free optimization,” in Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 2010.
[6] O. Vinyal and D. Poyey, “Krylov subspace descent for deep learning,” in Proceedings of the 15th International Conference on Artificial Intelligent and Statistics, La Palma, Spain, 2012.
[7] Z. You and Y. Pu, “The genetic convolutional neural network model based on random sample,” International Journal of u- and e-Service, Science and Technology, vol. 8, no. 11, pp. 317–326, 2015.
[8] G. Rosa, J. Papa, A. Marana, W. Scheire, and D. Cox, “Fine tuning convolutional neural networks using harmony search,” in Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, A. Pardo and J. Kittler, Eds., vol. 9423 of Lecture Notes in Computer Science, pp. 683–690, 2015.
[9] Vina Ayumi, L.M. Rasdi Rere, M. Ivan Fanany, and A. Murni Arymurthy. “Optimization of Convolutional Neural Network using Microcanonical Annealing Algorithm”, The 8th International Conference on Advanced Computer Science and Information Systems (ICACISIS), Malang, 15 - 16 Oktober 2016.
[10] L.-O. fedorovici, R.-E. Precup, F. Dragan, C. Purearu, Evolutionary optimization-based training of convolutional neural networks for ocr application, in Proc. 17th International Conference on System Theory, Control and Computing, pp. 207 – 212, 2013.
[11] LiSA-Lab, Deep Learning Tutorial Release 0.1, University of Montreal, Montreal, Canada, 2014.
[12] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Application, John Wiley & Sons, Hoboken, NJ, USA, 2010.
[13] E.-G. Talbi, Metaheuristics: From Design to Implementation, JohnWiley & Sons, Hoboken, NJ, USA, 2009.
[14] I. Boussa’id, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” Information Sciences, vol. 237, pp. 82–117, 2013.
[15] K. S. Lee and Z.W. Geem, “A new meta-heuristic algorithm for continuous engineering
optimization: harmony search theory and practice,” Computer Methods in Applied Mechanics and Engineering, vol. 194, no. 36–38, pp. 3902–3933, 2005.

[16] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, “Optimization by simulated annealing,” Science, vol. 220, no. 4598, pp. 671–680, 1983.

[17] G. Dueck, T. Scheuer, Threshold Accepting: a general purpose optimization algorithm appearing superior to simulated annealing, Journal of Computational Physics, 90: 161 – 175, 1990.

[18] Quoc V. Lee. A tutorial on deep learning. Google Brain, Google Inc., 2, 2015.

[19] Yoshua Bengio. Learning Deep Architecture for AI, volume 2:No.1. Foundation and Trends in Machine Learning, 2009.

[20] R.B. Palm. Prediction as a candidate for learning deep hierarchical model of data. (Master thesis), Technical University of Denmark, Denmark, 2012.

[21] L.M. Rasdi Rere, M. Ivan Fanany, and A. Murni Arymurthy, Computational Intelligence and Neuroscience, Volume 2016, Article ID 1537325, 13 pages