Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 72 (2015) 137 – 144

Procedia Computer Science

The Third Information Systems International Conference

Simulated Annealing Algorithm for Deep Learning

L.M. Rasdi Rere, Mohamad Ivan Fanany, Aniati Murni Arymurthy

[a]*Faculty of Computer Science, Universitas Indonesia, Depok 16424, Indonesia*

Abstract

Deep learning (DL) is a new area of research in machine learning, in which the objective is moving us closer to the goal of artificial intelligent. This method can learn many levels of abstraction and representation to create a common sense of data such as text, sound and image. Although DL is useful for a variety of tasks, it's hard to train. Some methods in training deep learning to make it optimal have been proposed, including Stochastic Gradient Descent, Conjugate Gradient, Hessian-free optimization, and Krylov Subspace Descent. In this paper, we proposed Simulated Annealing (SA) to improve the performance of Convolution Neural Network (CNN), as an alternative approach for optimal DL using modern optimization technique, i.e. metaheuristic algorithm. MNIST dataset is used to ensure the accuracy and efficiency of the proposed method. Moreover, we also compare our proposed method with the original of CNN. Although there is an increase in computation time, the experiment results show that the proposed method can improve the performance of original CNN.

Peer-review under responsibility of organizing committee of Information Systems International Conference (ISICO2015)

Keywords: Simulated Annealing algorithm; Deep Learning, Optimization

1. Introduction

Deep learning refers to a new class of machine learning methods, where the process of information from many layers in hierarchical architectures can be used to classify pattern and learning of feature. This technique is in the intersections between the areas of research in the graphical model, optimization, signal processing, pattern recognition and neural network. Three principal reasons for the current popularity of deep learning are it drastically increases the capabilities of chip processing, significantly lowers the cost of computing hardware and recent advances in research of machine learning [1].

In general, techniques of deep learning can be classified into deep discriminative models and generative models [2]. Examples of discriminative models are deep neural networks (DNNs), recurrent neural network (RNNs), and convolutional neural networks (CNNs). On the other hand, generative models, for instance, are restricted Boltzmann machine (RBMs), deep belief networks (DBNs),

Peer-review under responsibility of organizing committee of Information Systems International Conference (ISICO2015)
doi:10.1016/j.procs.2015.12.114

regularized autoencoders, and deep Boltzmann machines (DBMs). Among those techniques, CNNs is the focus of this paper. CNNs is a deep supervised-learning while this model is usually efficient to train and test, flexible to construct, and suitable for end-to-end learning of complex system [2].

Although DL has good reputation for solving a variation of learning task, it is not easy to train [3]. The Recent proposal of optimization techniques for training DL used layered wise pre-training [4]. Some examples of the successful methods for training of this technique are Stochastic Gradient Descent, Conjugate gradient, Hessian-free Optimization and Krylov Subspace Descent.

Stochastic Gradient Descent (SGD) is simple to implement and also fast in the process for problems that have many training examples. However, this method needs a lot of manual tunings to optimize its parameters and essential sequential, so that to distribute them using computer clusters or to parallelize them using GPUs is difficult. In contrast, Conjugate Gradient (CG) can distribute the computation across machines and parallelize for computing the gradient on GPU as well as more stable to train and easier to check for convergence. But this method usually is slow, and it can be fast due to the availability of large amounts of RAMs, multicore CPUs, GPU and computer cluster with fast network hardware [5].

Hessian-free optimization (HFO), proposed by J. Marten [6], uses the basis of 2^{nd} order optimization approach, the truncated-Newton. HFO has been applied to train deep auto-encoders neural network and able to overcome the under-fitting problem as well as more efficient than pre-training + fine-tuning approach. Krylov Subspace Descent (KSD), proposed by Vinyals [7], is another second order optimization method. Compare to HFO, optimization using KSD has some advantages. First, KSD has greater simplicity and robustness, in which it does not need heuristic to initialize and update the smoothing value. Second, KSD can be applied even if H is not positive semi-definite, and third, KSD seems to work better than HFO in both optimization speed and classification performance. However, the weakness of optimization using SGD is more memory required.

In fact, a vast majority of modern optimization techniques are usually heuristic and metaheuristic [8]. These optimization methods are very powerful in solving hard optimization problems, and they have been applied in almost all the main areas of science and engineering as well as industrial application. However, the research of metaheuristic algorithms to optimize DL is rarely conducted. Because of that in this worked, we used metaheuristic algorithm as an alternative approach for optimal performance of DL.

Metaheuristic work for three main purposes: solving the problem faster, solving large problems and obtaining robust algorithms. Moreover, they are flexible, simple to design and also not difficult to implement [9]. The combination of rules and randomness to duplicate natural phenomena usually is employed in metaheuristic algorithms. The phenomena may include biological evolutionary process like Genetic Algorithms (GA), Genetic Programming (GP), Evolution Strategy (ES), and Differential Evolution (DE). Animal behaviors, as an ethology phenomenon, for instance, are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Bacterial Foraging Optimization Algorithms (BFOA) and Bee Colony Optimization (BCO). Examples of physics phenomena are Simulated Annealing (SA), Threshold Accepting method (TA) and Micro canonic Annealing (MA). TA and MA are variants of SA [10]. Another form of metaheuristic, for instance, is Harmony Search method, is inspired by musical phenomena [11].

Metaheuristic algorithms can also be classified into trajectory-based (single-solution) and population-based [10]. Some of single solution based metaheuristic are SA, TA, MA, Tabu Search (TS), Noising Method (NM), and Guided Local Search (GLS). In case of Population-based metaheuristic, it can be separated into Evolutionary Computation and Swarm Intelligent. The general term of Evolutionary Computation is inspired by the Darwinian principles of nature's capability to evolve living beings well adapted to their environment. Examples of these algorithms are GA, ES, GP, and DE. Furthermore, Swarm Intelligent takes inspiration from the collective behavior of a group of social insect colonies and of other animal societies. Including these methods are ACO, PSO, BFOA, and BCO. Among these algorithms, SA is used in our study.

SA is a compact and robust technique, which provides excellent solutions to single and multiple-objective optimization problems with a substantial reduction in computation time. The origin of this method is Metropolis Algorithm, in statistical mechanics [9].

Moreover to test the performance of DL based on SA, we used MNIST dataset. This dataset consists of digital images of handwritten and it is separated from the 60,000 training data and the 10,000 testing data. All of the image data have been standardized and centered in the image of fixed size of 28 x 28 pixels. Each of images on the original of this dataset is represented by a value between 0 for black and 255 for white, and anything in between is a different shade of gray [11].

This paper is organized as follows. Section 1 is introduction. Section 2 gives description of simulated annealing algorithm. Section 3 explains about convolution neural networks. Section 4 describes the proposed methods. Section 5 presents simulation result. Finaly, Section 6 concludes this paper.

2. Simulated Annealing algorithm

Simulated Annealing (SA) was first proposed by Kirkpatrick et al. [13]. This method is based on the annealing technique to get the ground state of matter, which is the minimal energy of the solid state. In case of growing a single crystal from the melt, the low temperature is not a suitable condition to obtain the ground state of matter. Annealing techniques is an attempt to achieve a low-temperature state by way of melting the substance, and then lowering the temperature slowly; this can spend long times at temperatures around the freezing point. If this is not done, the substance can be outside the limits of equilibrium, and the crystals obtained will have a lot of defects. For the case of glass, crystals will not form a crystalline order, because the condition is metastable, in which its structure is locally optimal.

Boltzmann distribution is a quantitative key of SA method. The following equation gives the probability of any actual state of x

$$p(x) = e^{\frac{-\Delta f(x)}{kT}} \tag{1}$$

where $f(x)$ is the configuration of energy, k is Boltzmann's constant, and T is temperature [14]. In case of optimization problem, the standard optimization procedure of SA is given by the following steps:

1. *Generate initial solution vector*: Choose at a random initial solution x_0 for the system to be optimized, and then calculate the objective function.
2. *Initialize the temperature*: The initial value of temperature T is an important parameter for successful implementation of SA. If the value is too high, then it takes more reduction to converge. If too small, the search process may less than perfect so that the points could potentially global optimum be exceeded.
3. *Select a new solution in the neighborhood of the current solution*: A new solution $x_0+\Delta x$ is accepted as a new recent solution depending on T. The objective function for $x_0+\Delta x$ and x_0 are represented by $f(x_0+\Delta x)$ and $f(x_0)$, respectively.
4. *Evaluation a new solution*: If $f(x_0+\Delta x) \leq f(x_0)$, then $x_0+\Delta x$ is accepted and it replaces x_0, update the existing optimal solution and go to step 6. On the other hand, if $f(x_0+\Delta x) > f(x_0)$, $x_0+\Delta x$ can also be accepted with a probability base on equation (4).
5. *Decrease the temperature periodically*: For the duration of the search of process, temperature T decreases, as a result at the beginning of search, the probability of accepting deteriorating moves is high, and it gradually decreases.

6. *Repeat step 2 – 6 until a stopping criterion is met*: The computation is terminated when the termination criterion is satisfied. Otherwise, step 2 until 6 are repeated. The algorithm of SA is presented in Fig. 1.

3. Convolution Neural Network

Convolution Neural Network (CNN) is a variant of Multi-Layer Perceptron (MLP) which is inspired from cat's visual cortex, in particular by the models proposed by Hubel and Wiesel in 1962 [12]. The first computation of this model is found by Fukushima's Neocognitron in 1980, Later in 1989, LeCun *et al* following up on this idea, designed and trained convolution networks using error gradient, obtaining state-of-the-art performance on several pattern recognition tasks [15].

The architecture of original CNN is usually organized in layers of two types: convolution layers and subsampling layers as is shown in Fig. 2 [12]. In this technique, a model trained on a multilevel architecture that consists of several stages. Each stage has input and output in the caller of feature map, which are set in array. Each feature map on the output represents a certain feature extracted at all locations on the input. Where each stage consists of three layers: a layer of the *filter bank*, a layer of *non-linearity*, and a layer of *feature pooling* [16].

Conceptually, a feature map is found by convolving k-th input image with a linier filter. Values of bias term are added and then apply it to the non-linear function. If k-th feature map at a given layer as h^k, whose filters are determined by the weights W^k and bias b_k, then the feature map h^k is obtained as follow:

$$h_{ij}^{k} = \tanh((W^{k} * x)_{ij} + b_{k}) \tag{2}$$

Simulated annealing algorithm

```
1  Select the best solution vector x0 to be optimized
2  Initialize the parameters: temperature T, Boltzmann's constant k, reduction factor c
3  while termination criterion is not satisfied do
4      for number of new solution
6          Select a new solution: x0+Δx
7              if f(x0+Δx) > f(x0) then
8                  fnew = f(x0 + Δx);  x0 = x0 + Δx
9                  else
10                     Δf = f(x0 + Δx) – f(x0)
11                     random r(0, 1)
12                         if r > exp(–Δf/kT) then
13                             fnew = f(x0 + Δx),  x0 = x0 + Δx
14                             else
15                             fnew = f(x0),
16                     end if
17             end if
18         f = fnew
19         Decrease the temperature periodically: T = c x T
20     end for
21 end while
```

Fig.1. Pseudo-code for Simulated Annealing algorithm

Another important concept of this CNN is max-pooling, which is a form of non-linear down sampling. The image of input is set into non-overlapping rectangles on max-pooling, and each of subregions like that, its output value is maximum.

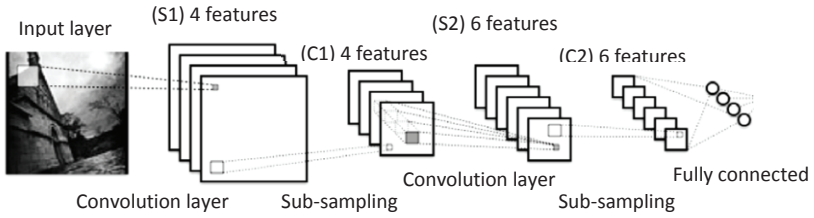

Fig. 2 The architecture of convolution neural network

Max-pooling is convenient in vision for two reasons: it reduces the computational complexity for upper layers, and it provides a form of translation invariance. As max-pooling gives additional robustness to the position, therefore it is a "smart" way to reduce the dimensionality of intermediate representation [11].

4. Design of proposed method

The design of proposed method uses CNN: 6c – 2s – 12c – 2s. The architecture of this model is shown in Fig.3. The kernel size of 1st and 2nd convolution layer is 5x5. On the other hand, the scale of 1st and 2nd subsampling layer is 2. This architecture is developed for MNIST dataset, in which the image size is 28 x 28 with the similar position set in [12].

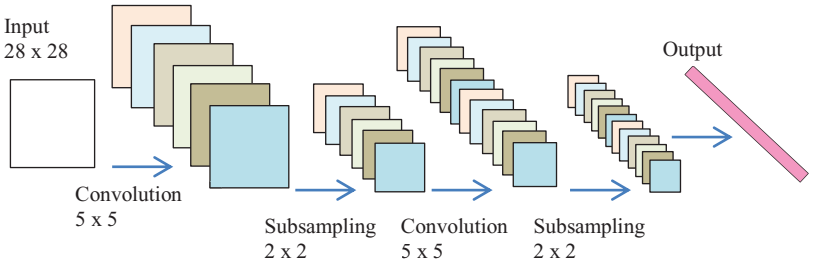

Fig. 3 Architecture for proposed method

The purpose of using SA to train CNN is to meet certain accuracy requirement and to make the approximation error accuracy and network complexity indicators minimum. These can be done by calculating standard error on the training set and fitness function of vector solution. One of the fitness functions used in this work as follow:

$$f = \frac{1}{2}\sqrt{\frac{\sum_{i=1}^{R}(o-y)^2}{R}} \tag{3}$$

Where o is the desired output, y is the actual output, and R is the number of training samples. In case of termination criterion, there are two conditions. First is the iteration reaches the specified maximum iteration, and second is the fitness function is less than a particular constant. At this moment, the approximate error accuracy and minimum network complexity indicators have achieved the most optimal state.

Essentially in this proposed method, algorithm of CNN calculates the weights and the bias of the system, and then the results on the last layers are used to compute the lost function, as the best solution vector x_0 to be optimized in SA. Selecting a new solution vector from the neighbourhood of the recent solution is conducted by adding Δx randomly. It is updated depending on the objective function $f(x_0+\Delta x) < f(x_0)$, or by Boltzmann distribution on equation (1). When SA algorithm has achieved the best objective function, the value of weights and the bias are updated for all layers of the system. Flowchart of this proposed method is shown in Fig. 4

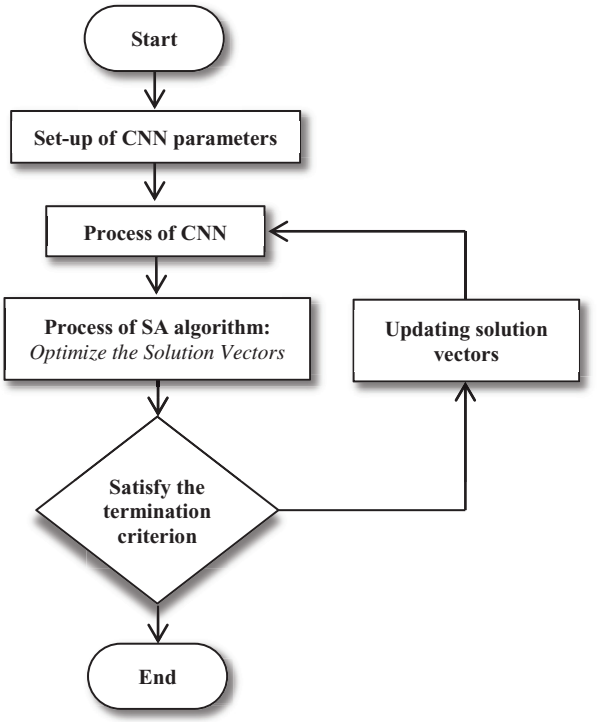

Fig. 4 Flowchart of SA algorithm for optimal CNN

5. Simulation and results

The primary concern of simulation in this paper is to improve the performance of original CNN by using SA. The objective is to minimize the error of classification task on MNIST dataset. This dataset is used to test the performance of the methods. Some of the MNIST hands written image datasets are shown in Fig. 5.

Fig. 5 Example of images on MNIST dataset

There are four strategies for implementing CNN and CNN by SA on the MNIST datasets. The first is the original CNN method (cnn). The second is CNN based on SA by neighborhood size 10 (cnnsa10), the

third is CNN based on SA by neighborhood size 20 (cnnsa20), and the fourth is CNN based on SA by neighborhood size 50 (cnnsa50). All of the implementation results are summarized in Table 1.

Table 1. Performance comparison of cnn, cnnSA10, cnnSA20 and cnnSA50 in terms of accuracy

Methods	Number of epochs									
	1	2	3	4	5	6	7	8	9	10
cnn	88.87	92.25	93.90	94.81	95.44	96.18	96.59	96.92	97.22	97.27
cnnSA10	89.18	92.38	94.20	95.19	95.81	96.50	96.77	97.04	97.27	97.41
cnnSA20	90.50	92.77	94.68	95.45	96.66	96.87	97.08	97.26	97.30	97.61
cnnSA50	91.10	94.16	95.49	96.20	96.91	96.99	97.33	97.42	97.40	97.71

Demonstration of the performance of implementation CNN and CNN by SA on MNIST dataset are given in Fig. 6. In general, the simulation results show that percentage accuracy or error of CNN by SA is better than the original CNN. The CNN by SA with neighborhood size 50 (cnnSA50) is better than cnnSA10 as well as cnnSA20.

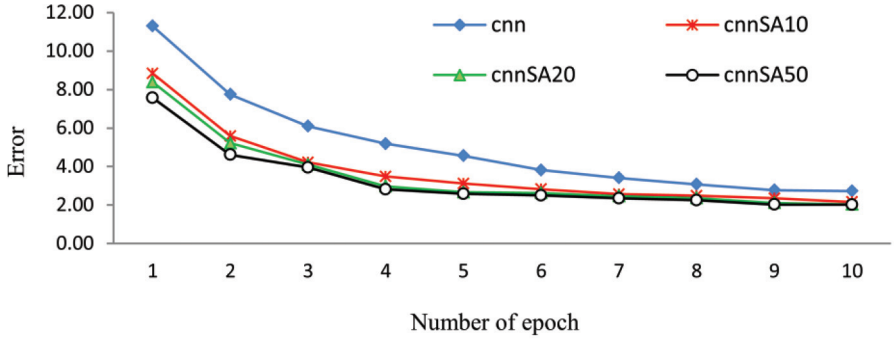

Fig. 6 Simulation results of cnn, cnnSA10, cnnSA20 and cnnSA50

In case of computation time for all methods are shown in Fig. 7. Illustration from this chart shows that the original CNN (cnn) is better that CNN by SA and the time will be increased by increasing the size of the neighborhood on SA.

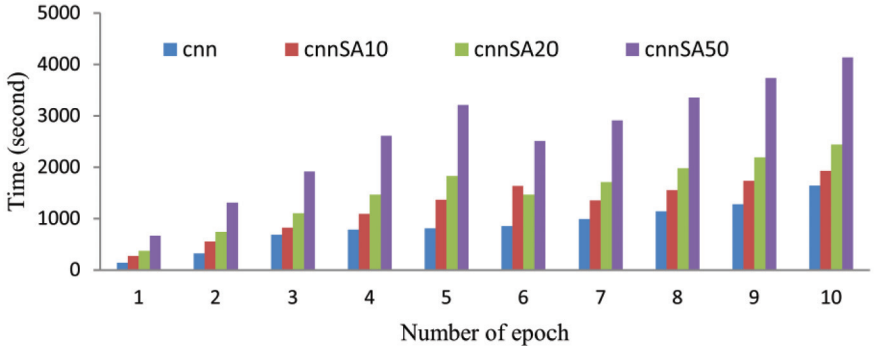

Fig. 7 Computation time for cnn, cnnSA10, cnnSA20 and cnnSA50

6. Conclusion

The objective of our proposed method, to optimize Convolution Neural Networks (CNN) using Simulated Annealing (SA), has been achieved. Even though there is an increase in computation time, the classification error from the proposed method is lower than the original of CNN for all of variation of epoch. This proposed method could potentially be employed and tried for another benchmark dataset, such as ImageNet, INRIA datasets, and Hollywood II dataset. The strategy can also be tested to other metaheuristic algorithms for deep learning, such as harmony search or differential evolution for more optimal CNN, Deep Belief Network (DBN) or Recurrent Neural Network (RNN). For future study, we plan to investigate for more efficient metaheristics optimization applied on deep learning.

Acknowledgements

This work is supported by Higher Education Center of Excellence Research Grant funded by Indonesia Ministry of Research, Technology and Higher Education. Contract No. 0475/UN2.R12/HKP.05.00/2015.

References

[1] L. Deng, ”Three Classes of Deep Learning Architecture and their application: A Tutorial Survey,” Proc. APSIPA 2011

[2] L. Deng and D. Yu, Deep Learning: Methods and Application, NOW Publisers 2014.

[3] J.D. Lamos Sweeney, Deep Learning using Genetic Algorithms, master thesis, Dept. of Computer Science, Rochester Institute of Technology, 2012.

[4] G. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural network”, Science, 313:504-507, 2006.

[5] Q.V. Lee, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A.Y. Ng,”On Optimization methods for deep learning,,” in Proc. The 28th International Conference on Machine Learning, Bellevue, WA, USA, 2011.

[6] J. Martens,”Deep learning via Hessian-free optimization,” in Proc. The 27th International Conference on Machine Learning, Haifa, Israel 2010.

[7] O. Vinyal, and D. Poyey,” Krylov Subspace Descent for Deep Learning,” ,” in Proc. The 15th International Conference on Artificial Intelligence and Statistics (AISTATS), La Palma, Canada Island, 2012.

[8] Xin-She Yang, Engineering Optimization: an introduction with metaheuristic applications, John Wiley & Sons, Hoboken, New Jersey, 2010.

[9] El-Ghazali Talbi, Metaheuristics from Design to Implementation, John Wiley & Sons, Hoboken, New Jersey, 2009.

[10] I. Boussaid, J. Lepagnot, P.Siarry,”A Survey on optimization metaheuristics,” Information Science. 237, 2013, pp. 82 – 117.

[11] K.S. Lee and Z.W. Geem,”A new meta-heuristics algorithm for continuous engineering optimization: harmony search theory and practice,” Comput. Methods Appl. Mech. Engrg. 194, 2005, pp. 3902 – 3933.

[12] LISA lab, Deep Learning Tutorial Release 0.1, University of Montreal, 2014.

[13] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi,”Optimization by Simulated Annealing,” Science, new series, 220.4598, 1983, pp. 671 – 680.

[14] L.M. Rasdi Rere, M. Ivan Fanany and A. Murni,”Application of metaheuristic algorithms for optimal smartphone-photo enhancement,” In: The 3th Global Conference of Consumer Electronic (GCCE), Japan, 2014, pp.542 – 546.

[15] Yoshua Bengio, Learning Deep Architecture for AI, Foundation and Trends® in Machine Learning: Vol.2: No.1, 2009, pp.1 – 127.

[16] Yann LeCun, K. Kavukcuoglu and C. Farabet, “Convolution Networks and Application in Vision,” in: Proceedings of 2010 IEEE International Symposium on Circuit and Systems,” Paris, 2010, pp.253 – 256.