# Optimasi Deep Belief Network Menggunakan Simulated Annealing

L. M. Rasdi Rere, Soegijanto dan Sudjiran
rasdi@jak-stik.ac.id
STMIK Jakarta STI&K
Jalan BRI No.17 Radio Dalam, Kebayoran Baru Jakarta Selatan

**ABSTRAK**

*Dalam beberapa tahun terakhir, deep learning (DL) merupakan area penelitian yang sangat penting dalam pemelajaran mesin. Metode ini dapat mempelajari beragam tingkat abstraksi dan representasi pada berbagai macam data seperti teks, suara dan citra. Meskipun metode DL telah sukses dipergunakan untuk aplikasi seperti pemrosesan suara, pengenalan fonetik, robotika, pencarian informasi, bahkan sampai analisa molekul, namun untuk melatih metode ini tidaklah mudah. Sejumlah teknik telah diusulkan untuk membuat pelatihan dalam DL menjadi lebih optimal, seperti menambahkan proses pra-training, mengganti fungsi aktivasi maupun metode gradien standar yang dipergunakan, ataupun memutuskan sebagian dari jaringan pada lapisan. Dalam penelitian ini, diusulkan optimasi deep belief network, yang merupakan salah satu metode popular dalam DL, dengan menambahkan simulated annealing pada lapisan terakhir. Hasil eksperimen yang dilakukan menggunakan data set MNIST menunjukkan bahwa, meskipun ada penambahan waktu komputasi, akan tetapi secara umum ada peningkatan akurasi dari DBN asli.*

**Kata Kunci:** *optimasi deep belief network dengan simulated annealing, deep learning menggunakan metaheuristik*

## PENDAHULUAN

*Deep learning* pada dasarnya merupakan area baru dalam pemelajaran mesin. Metode ini dimotivasi oleh penelitian kecerdasan buatan yang bertujuan meniru kemampuan manusia dalam mengamati, menganalisa, belajar dan mengambil keputusan, terutama sekali untuk mengatasi masalah yang sulit dan kompleks [1].

Secara prinsip metode DL dapat dikatakan berada di antara penelitian pada bidang pemrosesan sinyal, jaringan syaraf tiruan, pemodelan grafik, optimasi dan pengenalan pola. Reputasi penelitian dalam DL saat ini dipicu oleh meningkatnya kemampuan pemrosesan *chip* sebuah komputer secara signifikan, penurunan harga perangkat keras secara drastis, serta penelitian tingkat lanjut dalam bidang pemelajaran mesin [2].

Secara umum teknik yang ada dalam DL dapat diklasifikasikan dalam model diskriminatif, generatif dan hibrid. Contoh dari model diskriminatif adalah *deep neural network*, *convolutional neural network* (CNN) dan *recurrent neural network* (RNN). Model generatif misalnya *deep belief network* (DBN), *stacked autoencoder* (SAE) dan *deep* Boltzmann *machines* (DBM). Sedangkan model hibrid mengacu pada kombinasi dari arsitektur diskriminatif dan generatif, seperti model DBN untuk pre-training *deep* CNN [2].

Bentuk klasifikasi lain dari DL juga bisa dilakukan berdasarakan metode dasar yang membentuknya seperti metode berbasis CNN, berbasis *restricked* Boltzmann *machines* (RBM), berbasis *autoencoder* (AE) dan berbasis *sparse coding* (SC). DL berbasis CNN misalnya AlexNet, VGG, dan GoogLeNet. DL berbasis RMB adalah *deep energy models*, DBN dan DBM. DL berbasis AE contohnya SAE, *contractive* AE dan *denoising* AE. Sedangkan DL berbasis *sparse coding* adalah *laplacian* SC, *local coordinate coding* dan SC *spatial pyramid matching* [3].

Meskipun metode DL memiliki reputasi yang sangat baik dalam menyelesaikan beragam tugas pemelajaran, akan tetapi untuk melatih metode ini tidaklah mudah. Hal ini karena secara umum dibutuhkan jumlah data yang cukup besar pada DL, untuk dapat bekerja dengan optimal. Selain itu ketika beberapa jaringan ditambahkan, untuk dilatih secara berlapis sehingga dapat lebih meningkatkan kinerjanya, komplek-sitas solusi dapat terjadi, sehingga waktu pelatihan akan menjadi semakin lama.

Sejumlah cara dan metode yang ada telah diusulkan untuk mengatasi masalah ini. Salah

satu yang dianggap berhasil adalah teknik dengan *layered wise pre - training* seperti diajukan Hinton dan Salakhutdinov [4]. Usulan lainnya adalah dengan memutuskan sebagian jaringan, atau mengganti fungsi aktivasi maupun metode gradien standar, seperti menggunakan ReLu, *Hessian free optimization* [5], atau Krylov *Subspace Descent* [6].

Fakta yang ada saat ini, teknik optimasi modern yang banyak dikembangkan adalah heuristik atau metaheuristik. Teknik optimasi ini sangat handal serta telah banyak diaplikasikan untuk menyelesaikan beragam masalah dalam ilmu pengetahuan, rekayasa dan bahkan dunia industri [7].

Kombinasi beberapa aturan dan randomisasi untuk meniru fenomena di alam umumnya dipergunakan dalam metaheuristik. Misalnya fenomena etologi sebagai bentuk perilaku dari hewan, fenomena biologi dalam proses evolusi mahluk hidup, maupupun fisika dalam proses annealing pada logam.

Berdasarkan solusinya, algoritme metaheuristik dapat diklasifikasikan dalam metaheuristik berbasis solusi tunggal (*single solution based metaheuristic*) dan metaheuristik berbasis populasi (*population based metaheu-ristic*). Beberapa algoritme metaheuristik solusi tunggal misalnya adalah *simulated annealing*(SA),*macrocanonical annealing*(MA),dan*threshold accepting method*(TA). Sedangkan untuk metaheuristik berbasis populasi adalah algortime *evolution strategy*, *genetic algorithm*(GA),*particle swarm optimization*(PSO),*harmony search*(HS) dan *ant colony optimization* [8].

Selama beberapa tahun terakhir, algoritme metaheuristik mulai dipergunakan untuk optimasi dalam DL. Beberapa hasil penelitian [9-13], dengan berbagai usulan dan teknik yang berbeda, melaporkan bahwa secara umum, algoritme metaheuristik dapat lebih meningkatkan nilai akurasi dari DL.

Dalam penelitian ini diusulkan optimasi *deep belief network* (DBN), salah satu metode awal deep learning, menggunakan *simulated annealing* (SA). Strategi yang dipergunakan adalah dengan mencari nilai fungsi objektif terbaik pada lapisan terakhir dari DBN dengan algoritme SA, kemudian hasilnya akan dipergunakan kembali untuk menghitung nilai bobot dan bias dari lapisan sebelumnya.

Untuk menguji metode yang diusulkan, dipergunakan dataset MNIST (*Mixed National Institute of Standards and Technology*). Data set ini adalah data digital angka tulisan tangan, yang mengandung 60.000 data training dan 10.000 data testing. Semua citra pada data ini telah ditengahkan dan distandarkan dengan ukuran 28 x 28 piksel. Setiap piksel pada citra diwakili nilai 0 untuk warna hitam dan nilai 255 untuk warna putih, serta diantaranya adalah gradasi warna abu-abu.

## DEEP BELIEF NETWORK

Salah satu metode penting dalam sejarah DL adalah *deep belief networks*. Metode ini setidaknya mempunyai dua fitur penting yaitu *greedy learning strategies*, yang merupakan efisiensi setiap lapisan untuk menginisialisasi jaringan, dan *fine tuning* semua bobot yang diinginkan [3].

Banyak keuntungan yang diperoleh dari strategi pemelajaran *greedy* ini seperti yang dilaporkan dalam [14], akan tetapi metode ini juga mempunyai kelemahan yaitu tingginya biaya komputasi, dan upaya yang dilakukan untuk lebih meningkatkan metode ini melalui pelatihan dengan pendekatan *maximum - likelihood* belum menemukan titik terang [15].

Beragam tipe dari metode *deep belief networks* telah diusulkan dengan pendekatan yang berbeda-beda [3]. Dalam penelitian ini yang dipergunakan adalah *deep belief networks* berbasis *restricked Boltzmann machine* (RBM), yaitu dengan cara menumpuk sejumlah RBM yang kemudian dilatih dengan "*layer wise*", dalam prosedur pemelajaran tanpa pengawasan.

RBM merupakan varian dari metode *Boltzmann machine* dengan pembatasan, memiliki lapisan terlihat dan lapisan tersembunyi yang membentuk graf bipartit, sehingga memungkinkan pelatihan menjadi lebih efisien [16].

Karena tidak ada hubungan antara *neuron* dalam satu lapisan pada RMB, sebuah

$$p_i(h_i = 1/v) = \frac{e^{(b_j + w_j v)}}{1 + e^{(b_j + w_j v)}}$$

bagian yang tersembunyi biner diatur bernilai 1, untuk vektor masukan *v* dengan probabilitas *p* memenuhi persamaan (1) berikut ini.

Dengan cara yang sama untuk lapisan terlihat, probabilitas bersyarat dapat ditemukan sesuai dengan persamaan (2) berikut ini, dimana *a* adalah nilai bias untuk lapisan terlihat (*v*), *b* adalah bias untuk lapisan tersembunyi (*h*), dan *w* adalah parameter yang menghubungkan koneksi antara lapisan terlihat dan lapisan tersembunyi.

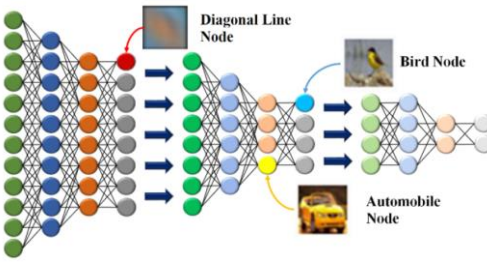
Tahapan pertama pada RBM dengan pemelajaran *greedy* yang efisien, lapisan demi lapisan adalah pelatihan menggunakan RBM pertama untuk data. Kemudian

$$p(v_i = 1/h) = \frac{e^{(a_i + w_i h)}}{1 + e^{(a_i + w_i h)}}$$

pelatihan menggunakan lapisan RBM pertama sebagai lapisan tersembunyi pada RBM kedua. Proses ini dapat di lanjutkan untuk beberapa RBM, dimana RBM berikutnya akan belajar fitur yang semakin kompleks.

Sebagai contoh untuk aplikasi dalam bidang *computer vision* seperti diperlihatkan pada gambar 1, lapisan pertama dapat mempelajari fitur sederhana untuk mengenali tepi. Lapisan selanjutnya mempelajari sekumpulan tepi seperti bentuk tertentu, dan pada akhirnya lapisan teratas dapat mengenali suatu objek seperti citra kucing, burung, mobil dan lainnya.

Tahapan kedua adalah *fine-tuning* yang umumnya menggunakan metode *back propagation*, untuk semua nilai bobot dengan keluaran yang diinginkan. Karena *pre-training* bobot menghasilkan inisialisasi yang bagus, proses *back propagation*akan cepat mengoptimalkan nilai pada bobot [17].




**Gambar-1 : Pembelajaran fitur pada DBN**

## SIMULATED ANNEALING

*Simulated annealing* (SA) merupakan salah satu algortime metaheuristik solusi tunggal yang diusulkan Kirkpatrick, Gelatt dan Vecchi pada tahun 1983 [18]. Algortime ini merupakan sebuah metode pencarian acak (*random search*) untuk masalah optimasi global yang didasarkan pada proses annealing pada logam.

Pada dasarnya algortime SA menggunakan pencarian acak yang tidak hanya mengijinkan perubahan nilai fungsi objektif baru yang lebih baik, akan tetapi juga memungkinkan untuk memilih fungsi objektif baru yang tidak ideal. Misalnya untuk kasus optimasi minimum, setiap perubahan yang menurunkan nilai fungsi objektif *f* (*x*) akan diterima, tetapi beberapa perubahan yang meningkatkan nilai *f* (*x*) juga dapat diterima dengan probabilitas transisi menurut persamaan (3) berikut ini:

$$p = e^{-\triangle E/kT}$$

di mana Δ*E* adalah perubahan tingkat energi, *k* adalah konstanta Boltzmann dan *T* adalah temperatur untuk mengatur proses annealing. Persamaan ini di dasarkan pada rumusan distribusi Boltzmann dalam ilmu fisika [7]. Prosedur standar SA untuk masalah optimasi adalah sebagai berikut [8]:

1. **Membangkitkan vektor solusi**
   Vektor solusi awal dipilih secara acak, untuk kemudian dilakukan perhitungan pada nilai fungsi objektif.

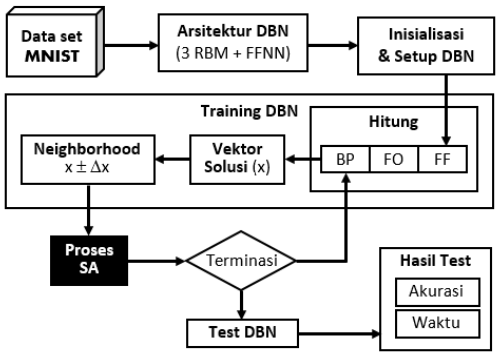
2. **Inisialisasi temperatur**
   Jika temperatur awal terlalu tinggi, untuk mencapai konvergensi akan memerlukan waktu yang cukup lama, sebaliknya jika terlalu kecil optimum global kemungkinan tidak akan ditemukan.
3. **Memilih solusi baru**
   Sebuah solusi baru dipilih secara acak yang diperoleh dari solusi terdekat yang ada (*neighborhood*).
4. **Evaluasi solusi baru**
   Sebuah kandidat solusi dipilih sebagai solusi baru, bergantung pada nilai fungsi objektif dan probabilitas transisi.
5. **Menurunkan temperatur**
   Secara periodik temperatur diturunkan selama proses pencarian solusi yang diinginkan.
6. **Kriteria berhenti**
   Ketika kriteria penghentian telah terpenuhi, maka proses komputasi dihentikan. Jika tidak maka langkah 2 sampai 6 diulang sampai terpenuhi solusi yang diharapkan.

## SKENARIO DESAIN

Secara umum skenario desain optimasi DBN menggunakan SA diperlihatkan seperti pada gambar 2. Pada awalnya diberikan dataset yang dipergunakan dalam proses klasifikasi, dalam hal ini dataset MNIST. Kemudian ditentukan struktur arsitektur, inisialisasi parameter, serta *set up* bobot dan bias pada DBN.

Arsitektur yang dipergunakan dalam skenario desain DBN ini terdiri dari tiga tumpukan RBM dan satu *feed forward neural network* (FFNN). Ukuran lapisan pada FFNN adalah 100-100-100-10, dimana lapisan pertama sampai tiga terdiri dari 100 masukan, sedangkan 10 lapisan terakhir merupakan label keluaran.

Tahapan selanjutnya adalah pelatihan DBN yang dilakukan dengan menentukan semua nilai bobot dan bias pada lapisan dengan perhitungan *feed forward* (FF), fungsi objektif (FO) serta *back propagation* (BP). Nilai bobot dan bias pada lapisan terakhir ($x$), kemudian dipergunakan sebagai vektor solusi oleh metode SA. Dengan cara menambahkan sejumlah nilai $\Delta x$ yang didapatkan secara acak, akan didapatkan sejumlah nilai ketetanggaan yang merupakan kandidat solusi ($x + \Delta x$).




**Gambar-2:** Skenario Desain optimasi DBN dengan algoritme SA

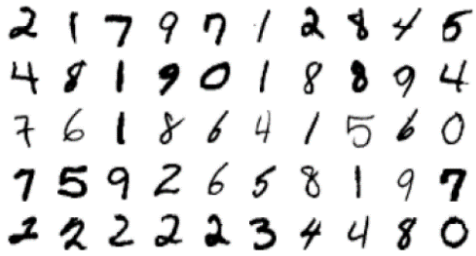
Penentuan nilai dari $\Delta x$ adalah sangat penting, karena pemilihan yang tepat akan dapat meningkatkan nilai akurasi pada DBN. Selanjutnya vektor solusi dari DBN akan selalu diperbaharui berdasarkan nilai ketetanggaan, melalui suatu mekanisme dalam proses pada metode SA.

Ketika kriteria terminasi tercapai, semua bobot dan bias diperbaharui untuk semua lapisan dalam sistem. Tahapan akhir dari skenario optimasi DBN dengan SA adalah pengujian dari hasil pelatihan DBN. Hasil akhir yang diperoleh adalah nilai akurasi dan waktu komputasi.

## HASIL EKSPERIMEN

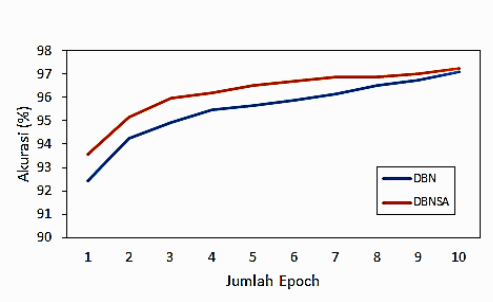
Dalam penelitian ini, klasifikasi pada citra dilakukan dengan menggunakan dataset MINIST. Beberapa contoh dari citra tulisan angka dengan tangan pada dataset MNIST diperlihatkan seperti gambar 3.

Eksperimen yang dilakukan pada dataset ini diimplementasikan dalam MatLab-R2011a, pada komputer dengan prosesor Intel Core i7-4500u, memori RAM 8 GB, dan Windows 10. Program asli pada eksperimen adalah DeepLearnToolbox dari Palm [19], yang dimodifikasi dengan menambahkan algoritme SA.



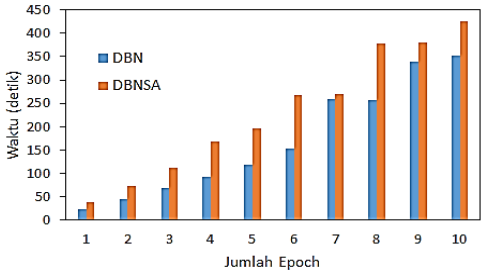
**Gambar-3:** Contoh citra dataset MNIST

Hasil rata-rata dari *deep belief network* yang dioptimasi dengan SA (DBNSA), dibandingkan dengan DBN asli, diberikan pada gambar 4 untuk nilai akurasi (%) dan gambar 5 untuk waktu komputasi (detik), pada 10 epoch pertama. Secara umum dari hasil ekperimen yang dilakukan menunjukkan bahwa metode yang diusulkan menghasilkan nilai akurasi yang lebih tinggi dari DBN asli untuk setiap epoch yang diberikan. Sebagai contoh untuk 2 epoch, nilai akurasi DBN asli adalah 94,26%, sedangkan DBNSA adalah 95,34%.




**Gambar-4:** Perbandingan Nilai Akurasi DBNdan DBNSA

Untuk waktu komputasi, nilai DBN asli secara umum lebih baik, karena metode DBNSA yang diusulkan pada dasarnya menambahkan proses algortime SA pada DBN asli. Pada 2 epoch misalnya, waktu komputasi DBN asli adalah 46 detik, sedangkan waktu komputasi DBNSA adalah 90 detik. Hasil lengkap nilai akurasi (A) dan waktu komputasi (T) untuk 10 epoch pertama diberikan pada tabel 1.




**Gambar-5**: Perbandingan Waktu Komputasi DBN dan DBNSA

Peningkatan nilai akurasi dari metode yang diusulkan, dibandingkan DBN asli bervariasi untuk setiap epoch dengan rentang nilai 0,37% sampai dengan 1,03%. Sedangkan waktu komputasi untuk DBNSA dan DBN asli pada kisaran 1,04× sampai dengan 1,82×. Dalam kasus 100 epoch, nilai akurasi DBN asli adalah 97,77%, dengan waktu komputasi 3.197 detik. Sedangkan nilai akurasi untuk DBNSA adalah 97,79% dengan waktu komputasi 4.916 detik.

**Tabel-1:** Hasil DBN dan DBNSA

| Epoch | DBN | | DBNSA | |
|---|---|---|---|---|
| | A | T | A | T |
| 1 | 92,44 | 38 | 93,58 | 38 |
| 2 | 94,26 | 46 | 95,17 | 73 |
| 3 | 94,94 | 69 | 95,97 | 113 |
| 4 | 95,48 | 93 | 96,19 | 169 |
| 5 | 95,68 | 118 | 96,52 | 196 |
| 6 | 95,90 | 153 | 96,72 | 267 |
| 7 | 96,14 | 256 | 96,88 | 378 |
| 8 | 96,52 | 259 | 96,89 | 379 |
| 9 | 96,74 | 340 | 97,00 | 379 |
| 10 | 97,12 | 351 | 97,26 | 425 |

## PENUTUP

Algoritme *simulated annealing* yang dipergunakan dalam penelitian ini terbukti dapat dipergunakan untuk optimasi metode *deep belief netwoks*. Hal ini dapat dilihat dari meningkatnya nilai akurasi dari metode yang diusulkan, untuk semua nilai epoch yang diberikan, dibandingkan dengan *deep belief netwoks* standar.

Adalah mungkin untuk memvalidasi metode yang diusulkan, dengan berbagai modifikasi pada program yang ada, menggunakan dataset lainnya seperti ImageNet, ORI maupun INRIA. Selain itu studi lanjutan dapat dilakukan dari strategi

metode yang diusulkan dengan mengganti SA memakai algoritme metaheuristik lainnya seperti MA, TA, GA, PSO atau HS.

**DAFTAR PUSTAKA**

[1] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya,R.Wald, and E.Muharemagic, "Deep learning applications andchallenges 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 Applica-tion*,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 dimensionalityof 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 deeplearning," in *Proceedings of the 15th International Conference onArtificial Intelligent and Statistics (AISTATS)*, La Palma, Spain, 2012.

[7] X.-S. Yang, *Engineering Optimi-zation: An Introduction withMetaheuristic Application*, John Wiley & Sons, Hoboken, NJ, USA, 2010.

[8] I. Boussa¨ıd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[9] L. M. Rasdi Rere, M. I. Fanany, and A. M. Arymurthy, "Simulatedannealing algorithm for deep learning," *Procedia Computer Science*, vol. 72, pp. 137–144, 2015.

[10] Z. You and Y. Pu, "The genetic convolutional neural networkmodel based on random sample," *International Journal of u- ande-Service, Science and Technology*, vol. 8, no. 11, pp. 317–326, 2015.

[11] G. Rosa, J. Papa, A. Marana, W. Scheire, and D. Cox, "Fine tuningconvolutional neural networks using harmony search,"in *Progress in Pattern Recognition, Image Analysis, ComputerVision, and Applications*, A. Pardo and J. Kittler, Eds., vol. 9423of *Lecture Notes in Computer Science*, pp. 683–690, 2015.

[12] Earnest Paul Ijjina and Krishna Mohan Chalavadi. Human action recognitionusing genetic algorithms and convolutional neural networks. Pattern Recognition, 59:199–212, 2016.

[13] 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 (ICACSIS), Malang, 15 - 16 Oktober 2016.

[14] I. Arel, D.C. Rose, and T.P. Karnowski. Deep machine learning - a new frontierin artificial intelligence research [research frontier]. Comput. Intell. Mag. IEEE, 5(4):13–18, 2010.

[15] Y. Bengio and A. Courville and P. Vincent. Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol.5, Issue: 8, pp. 1798 – 1828, Aug. 2013.

[16] M.A. Carreira-Perpinan and G.E Hinton. On constractive divergence learning.in: Proceedings of the tenth international workshop on artificial intelligenceand statistics. NP: Society for Artificial Intelligence and Statistics, pp. 33–40, 2005.

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

[18] S. Kirkpatrick, C.D. Gelatt, and M.P Vecchi. Optimization by simulated annealing. Science, New Series, 220 (4598): 671–680, 1983.

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