Deep Belief Network Sebagai Algoritma untuk Mendeteksi Penyakit Diabetes

Debora Sopiana Ikawahyuni
Universitas Kristen Immanuel; Jl. Solo Km. 11, 1 Kalasan, (0274) 496256
surel: deborasopiana47@student.ukrimuniversity.ac.id

Abstrak

Menurut data *International Diabetes Federation* (IDF) dalam IDF Diabetes Atlas 2019 angka kematian akibat diabetes mencapai 4.2 juta, dan sekitar 463 juta orang dewasa (20 – 79 tahun) menderita diabetes. IDF juga menyebutkan bahwa angka prevalensi diabetes pada orang dewasa di Indonesia adalah sebesar 6.2%, dengan total kasus diabetes pada orang dewasa sebesar 10.681.400. Oleh karena itu, masyarakat perlu waspada akan kondisi kesehatan mereka.

Deep Belief Network (DBN) merupakan algoritma dalam Jaringan Saraf Tiruan (JST) yang memiliki potensi dan dapat dimanfaatkan untuk melakukan deteksi dini penyakit diabetes dengan melakukan kalkulasi statistik. DBN dibentuk dengan menumpuk algoritma *Restricted Boltzmann Machine* (RBM). Algoritma ini merupakan *generative model* yang sifatnya hierarkis dan dapat menggambarkan fungsi yang memiliki variasi tinggi dan menemukan berbagai macam fitur. Proses pendeteksian diabetes menggunakan DBN ini terdiri dari beberapa langkah. Langkah pertama adalah persiapan data, di mana data yang digunakan merupakan dataset yang disediakan oleh Datahub. Kedua, yaitu *training* algoritma DBN. Langkah terakhir, yaitu evaluasi dengan menggunakan algoritma *fine-tuning*, yaitu *Backpropagation* (BP).

Hasil penelitian menunjukkan nilai rata-rata tingkat *error* yang sangat kecil dari DBN dengan 15 *layer*, yaitu sebesar 0.055%. DBN juga telah dibuktikan mampu bekerja sebagai tahap inisialisasi *weights* awal dalam BP. Hal ini ditunjukkan dengan peningkatan signifikan rata-rata akurasi evaluasi pada tahap *fine-tuning* sebesar 22.616%, dibandingkan dengan BP yang menggunakan *weights* bernilai 0 dan bilangan acak yang memiliki rata-rata akurasi 0.0%.

Kata kunci: *Deep Belief Network*, diabetes, *Restricted Boltzmann Machine, Backpropagation*, Jaringan Saraf Tiruan

Abstract

According to the International Diabetes Federation (IDF) on Diabetes Atlas 2019, the mortality number of diabetes reached 4.2 million, and approximately 463 million adults (20 – 79 years) were living with diabetes. IDF also mentioned the prevalence of diabetes in adults in Indonesia was 6.2% with the total case of diabetes amounting to 10.681.400. There was also a gradual increase in the total case of diabetes in a population from 1980 to 2014. Hence, people should be aware and cautious about their state of health.

Deep Belief Network (DBN) is one of the novel Artificial Neural Network algorithms that is potential for early detection in diabetes because it provides a comprehensive prediction by utilizing statistical calculations. DBN is a hierarchical generative model, that is arranged by stacking Restricted Boltzmann Machine (RBM) algorithm. Furthermore, its capability to map high-variety functions and to find features could help recognizing the diabetes data patterns. The processes done in this algorithm are as follows: data preparation, training, and fine-tuning. Datahub provides the data for this research, and the fine-tuning algorithm is Backpropagation (BP).

As a result, DBN is proven capable to act as a preprocesing step in BP. This is indicated by the significant increase in the average accuracy value at the fine-tuning phase by 22.616% compared to BP without preprocessing, using weights with 0 and random values that have average accuracy values of 0.0%. In addition, this research shows a small average error for a 15-layer DBN with a value of 0.055%.

Keywords: Deep Belief Network, diabetes, Restricted Boltzmann Machine, Backpropagation, Artificial Neural Network

1. PENDAHULUAN

Menurut data *International Diabetes Federation* (IDF) dalam IDF Diabetes Atlas 2019, angka kematian yang disebabkan oleh diabetes mencapai 4.2 juta, dan sekitar 463 juta orang dewasa (20 – 79 tahun) menderita diabetes. Grafik jumlah kasus diabetes per % dari total populasi sepanjang tahun mulai dari 1980 sampai 2014 juga terus mengalami peningkatan, meskipun tidak signifikan. Oleh karena itu, masyarakat perlu waspada akan risiko jangka panjang dari penyakit ini. Kewaspadaan tersebut dapat muncul dengan diberikannya akses berupa alat agar masyarakat bisa mengetahui kondisi kesehatannya masing-masing, di mana alat ini bekerja dengan memanfaatkan sistem Jaringan Saraf Tiruan (JST).

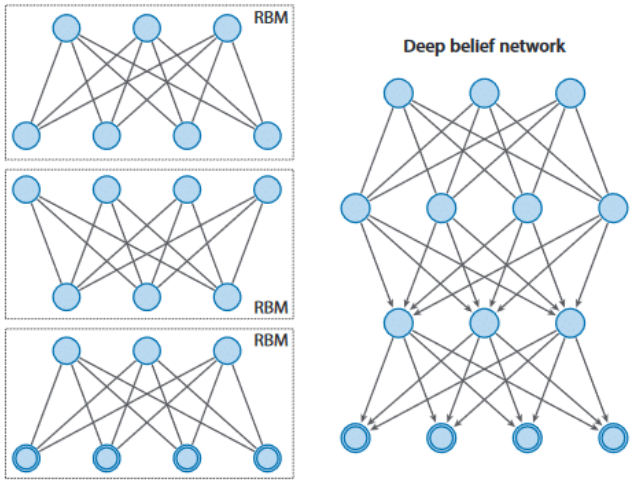
Diabetes di Indonesia juga dianggap sebagai masalah kesehatan yang besar dan telah menjadi perhatian sejak awal 1980-an [9]. Indonesia juga memiliki tingkat prevalensi 6,2% dengan lebih dari 10 juta orang yang mengidap diabetes [4]. Selain itu, Indonesia juga merupakan salah satu dari sepuluh besar negara secara global dengan jumlah penderita diabetes yang tinggi pada tahun 2013 [5]. Infodatin Pusat Data dan Informasi Kementerian Kesehatan RI dalam Simulasi dan Analisis Diabetes pada 14 November 2014 menyebutkan tentang pengendalian diabetes salah satunya adalah Pos Pembinaan Terpadu Penyakit Tidak Menular (Posbindu PTM). Salah satu kegiatan Posbindu PTM ini adalah meningkatkan kewaspadaan terhadap diabetes dengan melakukan deteksi dini dan konseling. Deteksi dini dapat dilakukan dengan menerapkan salah satu algoritma JST, yaitu *Deep Belief Network* (DBN). Algoritma *deep learning* ini juga telah sangat luas digunakan dalam berbagai macam bidang dan menunjukkan proses yang signifikan.

Alasan mengapa DBN ini digunakan adalah karena algoritma DBN dapat mengatasi kekurangan dari algoritma JST lain seperti *Backpropagation* yang dapat terjebak di lokal optimum dan memiliki *learning time* yang lama di jaringan yang memiliki banyak *hidden layers* [12]. DBN juga telah diimplementasikan dalam berbagai hal dan menghasilkan akurasi yang tinggi, salah satunya adalah untuk memprediksi arus lalu lintas [11] dan untuk mendeteksi gangguan, dengan tingkat akurasi deteksi sebesar 97.5% [1].

2. METODE PENELITIAN

2.1 *Deep Belief Network* (DBN)

Algoritma DBN diusulkan oleh Hinton, dkk. bersama dengan algoritma pembelajaran *greedy* yang *unsupervised* untuk membentuk satu *layer* dalam satu waktu di suatu jaringan [6]. Jumlah *layer* dalam DBN dapat ditambah dengan cara *greedy* [8]. Setiap *layer* baru yang ditumpuk di atas DBN akan memodelkan *output* dari *layer* sebelumnya dan bertujuan untuk mengekstraksi *higher-level dependencies* antara variabel input asli, sehingga dengan demikian dapat meningkatkan kemampuan jaringan tersebut untuk menangkap pola yang tidak berubah dalam data.



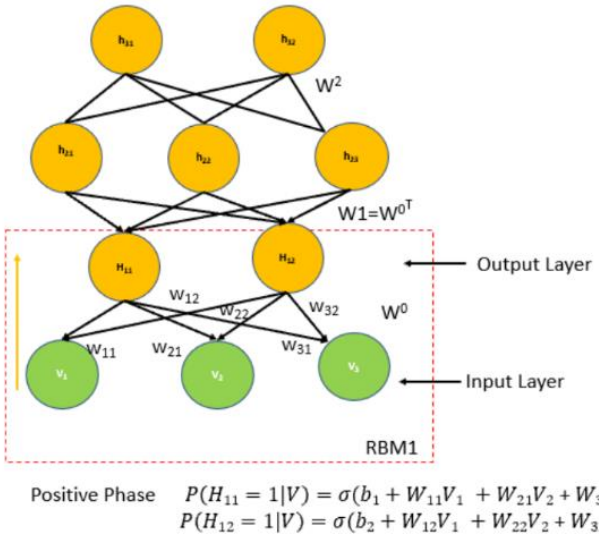
Gambar 1 Arsitektur DBN

Langkah-langkah *training* jaringan ini adalah sebagai berikut [2]:

a. *Set visible layer* dengan *nodes* sesuai dengan nilai atribut yang telah dibinerisasi.
b. Dilakukan *pre-training* per 1 RBM (*visible* dan *hidden layer*) dengan *Greedy learning algorithm* [7], di mana tahapan ini disebut *positive phase* dengan rumus

$$p(h_j = 1 \mid V) = \sigma(b + \sum_i v_i w_{ij}) \quad (1)$$

seperti pada Gambar 2.

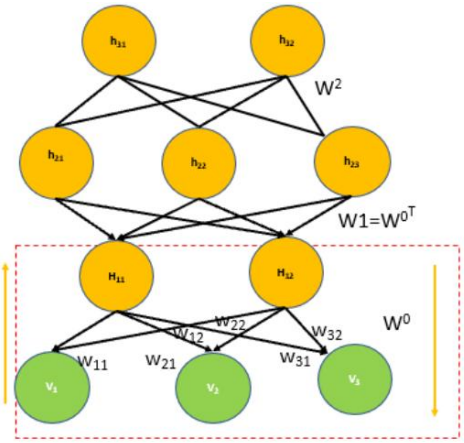


Gambar 2 *Positive Phase*

c. Rekonstruksi ulang unit *visible* dengan melakukan *negative phase* [7] dengan rumus

$$p(v_i = 1 \mid H) = \sigma(b + \sum_j h_j w_{ij}) \quad (2)$$

seperti pada Gambar 3.



Negative Phase $P(V_1 = 1|H_1) = \sigma(a_1 + W_{11}H_{11} + W_{12}H_{12})$
$P(V_2 = 1|H_1) = \sigma(a_2 + W_{21}H_{11} + W_{22}H_{12})$
$P(V_3 = 1|H_1) = \sigma(a_3 + W_{31}H_{11} + W_{32}H_{12})$

Gambar 3 *Negative Phase*

d. *Update weight* dengan rumus

$$w_{ij} = w_{ij} + l\left(p\left(h_j = 1 \mid V\right) - p(v_i = 1 \mid H)\right) \quad (3)$$

di mana l adalah nilai *learning rate.*

e. Proses diulangi sampai nilai *threshold* yang telah di-*set.*

f. Lakukan *fine-tuning*, di mana dalam penelitian ini, algoritma *fine-tuning* yang digunakan adalah Backpropagation.

2.2 *Backpropagation* (BP)

Backpropagation (BP) merupakan bagian dari JST, di mana arsitekturnya terdiri dari beberapa *layer* yang saling terhubung [3]. Ide dari algoritma ini cukup sederhana, yaitu *output* dari jaringannya akan dievaluasi dengan *output* yang diharapkan. Jika hasil belum memuaskan, maka *weights* dalam setiap *layer* akan diubah dan proses ini akan terus diulang sama nilai kesalahan yang didapat sudah cukup kecil atau batas maksimal iterasi sudah terpenuhi. Algoritma ini akan digunakan dalam tahap *fine-tuning* karena BP memberikan *output* yang merupakan hasil prediksi dari data input. Algoritma DBN yang telah dijelaskan sebelumnya berperan dalam melakukan pendekatan untuk mendapatkan *weights* yang akan digunakan pada BP.

Algoritma ini dapat dipecah menjadi empat tahapan utama [10], yaitu:

a. Feed-forward

Proses ini merupakan proses dua tahap. Bagian pertama adalah mendapatkan nilai untuk *hidden layer*, dan yang kedua adalah mendapatkan nilai untuk *output layer* dengan menggunakan nilai yang didapat dari *hidden layer* tadi. Setiap *node* input akan dikalikan dengan masing-masing *weights*-nya untuk menghitung nilai dari *node* pada *hidden layer*, lalu dilakukan aktivasi dengan menggunakan fungsi sigmoid.

b. Backpropagation output layer

Selanjutnya adalah menghitung *error* output *node*. Penghitungan *error* ini dimulai dari *output layer* ke *hidden layer* terlebih dahulu. Setelah nilai *error* didapatkan, nilai tersebut akan digunakan untuk proses backpropagation dan penyesuaian *weights*.

c. Backpropagation ke hidden layer

Tahap ini melakukan perhitungan *error* dari *hidden layer* ke *layer* input. Setelah *error* pada *hidden layer* didapat, *weights* antara *layer* input dan *hidden layer* dapat di-*update.* Sebelum melakukan proses *update weights*, tingkat perubahan perlu dilakukan terlebih dahulu, lalu *weights* baru dapat dihitung.

d. Update bobot

Hal yang penting dalam tahap ini adalah tidak melakukan *update weights* sampai semua *error* telah dihitung. Ketika *update weights* dilakukan saat semua *error* belum dihitung, dan nilai *weights* baru yang digunakan, maka hasilnya akan menjadi tidak valid. Tahap ini dilakukan untuk menyesuaikan nilai *weights* agar *output* yang dihasilkan mendekati *output* target.

2.3 Tahapan Proses

Langkah-langkah pengerjaan penelitian ini terdiri dari dua tahapan besar, yaitu algoritma DBN dan tahap *fine-tuning*. Langkah penyelesaian tahapan-tahapan tersebut adalah:

a) Algoritma DBN

1. Baca dataset: program membaca dataset dengan format .csv atau .xlsx dengan menggunakan fungsi yang telah disediakan oleh Python.
2. Binerisasi: bagian ini adalah proses mengubah setiap dataset yang masih berupa bilangan desimal menjadi bilangan biner dengan cara menyediakan tempat untuk masing-masing variabel sebanyak n digit, di mana n adalah jumlah digit dari hasil biner nilai maksimum dari setiap variabel ditambah suatu bilangan lalu dikurangi dengan nilai minimumnya. Hasil dari proses inilah yang akan dijadikan input pada program.
3. *Positive phase*: merupakan proses mendapatkan nilai *hidden units* (h) dengan menambah nilai bias (b) dengan jumlah hasil perkalian setiap input (x) dengan *weights*-nya (w) masing-masing pada setiap *layer*, dan kemudian diaktivasi dengan fungsi aktivasi sigmoid (σ).
4. *Negative phase*: merupakan proses untuk membarui nilai input atau *visible units* (v) yang dipakai untuk mendapatkan nilai *hidden units* sekali lagi.
5. *Update weights*: merupakan proses membarui *weights* (w) dengan menambah *weights* lama dengan *learning rate* (l) dikali selisih antara hasil *positive phase* dan *negative phase*.
6. Proses tersebut dijalankan sebanyak jumlah *layer* DBN yang telah di-*set* sebelumnya, jika proses belum berjalan sebanyak jumlah *layer*, maka proses akan kembali diulang dari tahap *positive phase* sampai *update weights.* Jika sudah berjalan sebanyak jumlah *layer*, maka tahap selanjutnya adalah melakukan *fine-tuning.*

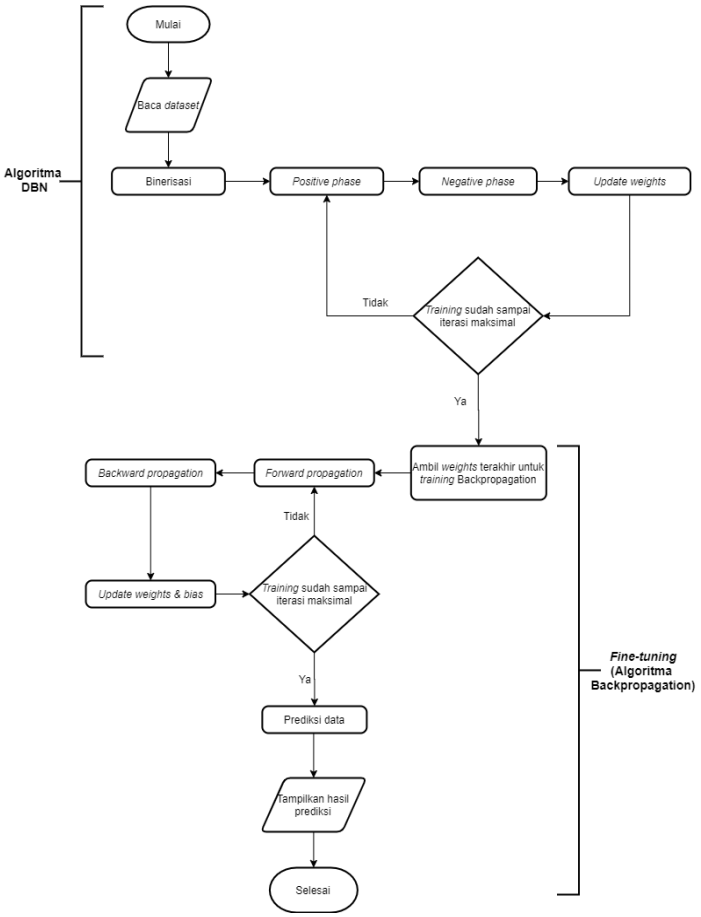
b) *Fine-tuning* (Algoritma *Backpropagation*)

Tahap ini berfungsi untuk memberikan *output* yang merupakan hasil prediksi dari data input.

1. Ambil *weights* terakhir untuk *training Backpropagation*: setiap *weights* (w) yang didapat dari setiap layer pada DBN yang telah dijalankan sebelumnya untuk dijadikan bobot pada Backpropagation.
2. *Forward propagation*: tahap ini adalah untuk mendapatkan nilai *hidden layer* dan output *layer* dengan menambah nilai bias (b) dengan jumlah hasil perkalian setiap input (x) dengan *weights*-nya (w) masing-masing pada setiap *layer*, dan kemudian diaktivasi dengan fungsi aktivasi sigmoid (σ).
3. *Backward propagation*: proses ini bertujuan untuk menghitung *error* (δ) dari setiap unit, yang di mana nilainya akan digunakan untuk mendapatkan nilai perubahan *weights* (Δw) dan bias (Δb).

4. *Update weights* & *bias:* proses ini bertujuan untuk membarui nilai *weights* (*w*) dan bias (*b*) dengan menambahkan nilai *weights* lama dan bias lama dengan masing-masing perubahan, yang telah didapatkan setelah menghitung nilai *error* (δ) setiap unit.
5. Proses tersebut diulangi sebanyak jumlah iterasi yang telah di-*set* sebelumnya, jika proses belum berjalan sebanyak jumlah iterasi, maka proses akan kembali diulang mulai dari tahap *forward propagation* sampai *update weights* & *bias.* Jika sudah berjalan sebanyak jumlah iterasi, maka tahap selanjutnya adalah melakukan prediksi data.
6. Prediksi data: tahapan ini merupakan tahap evaluasi di mana proses yang dijalankan sama seperti proses *forward propagation*, tetapi dengan nilai *weights* dan bias baru yang telah didapatkan dari tahapan *training.* Tahap ini mengevaluasi algoritma dengan menggunakan data *non-training*, yaitu data di luar yang dipakai pada tahapan *training*.
7. Tampilkan hasil prediksi: nilai prediksi yang dihasilkan akan berbentuk bilangan biner di mana 10 mengindikasikan positif diabetes, 01 mengindikasikan negatif diabetes, 00 mengindikasikan data asing, dan 11 untuk menyatakan keraguan.

Langkah-langkah penyelesaian algoritma ini dapat dilihat seperti pada Gambar 4.



Gambar 4 Diagram Alir DBN dalam Prediksi Diabetes

3. HASIL DAN PEMBAHASAN

3.1 Implementasi Program

a. *Positive phase*

Gambar 5 menunjukkan implementasi *positive phase* untuk meng-*update hidden node* pada *code* dengan menggunakan fungsi `np.dot` untuk melakukan perkalian antara *array* input dan masing-masing *weights*-nya dan ditambah dengan bias, kemudian dilakukan aktivasi dengan menggunakan fungsi `self.sigmoid()`.

```
pp_hidden_activations = np.dot(data, weights) + bias
pp_hidden_probs = self.sigmoid(pp_hidden_activations)
```

Gambar 5 *Positive phase*

b. *Negative phase*

Gambar 6 menunjukkan implementasi *negative phase* untuk meng-*update visible node* pada *code* dengan menggunakan fungsi `np.dot` untuk melakukan perkalian antara hasil dari *positive phase* dan masing-masing *weights*-nya dan ditambah dengan bias, kemudian dilakukan aktivasi dengan menggunakan fungsi `self.sigmoid()`.

```
np_visible_activations = np.dot(pp_hidden_states,
            weights.T) + bias
np_visible_probs = self.sigmoid(np_visible_activations)
```

Gambar 6 *Negative phase*

c. *Update weights*

Implementasi *update weights* berdasarkan rumus:

$$w_{ij} = w_{ij} + l\left(p\left(h_j = 1 \mid V\right) - p(v_i = 1 \mid H)\right) \quad (3)$$

dapat dilihat pada Gambar 7.

```
weights += self.learning_rate * ((pp_associations -
       np_associations) / n_examples)
```

Gambar 7 *Update weights*

3.2 Hasil Pengujian

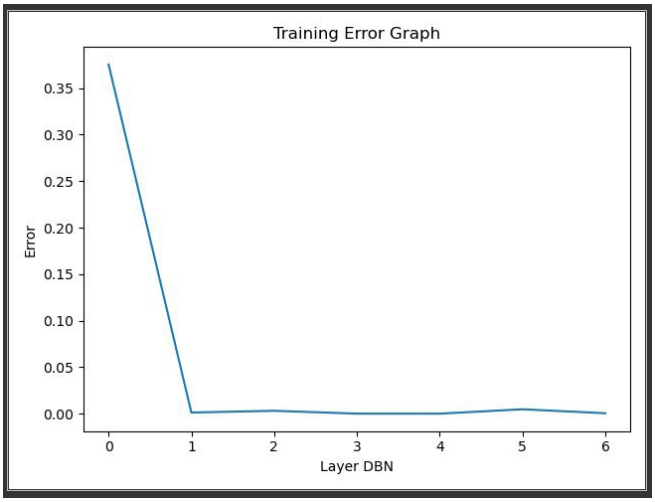
Pengujian dalam sistem ini dibagi menjadi dua, yaitu: *training* dan evaluasi. Bagian *training* untuk melakukan *training* algoritma, dan evaluasi pada tahap *fine-tuning.* Nilai *epoch* & *learning rate* RBM, *layer* DBN, dan *layer* Backpropagation untuk tahap *training* dapat dilihat pada Gambar 5.

```
rb = RBM(max_epoch=10000, l_rate=0.1)

dbnets = DBN(15)
self.n_layers_bp = 10
```

Gambar 8 *Epoch* dan *layer* tahap *training*

Hasil *error* dalam *training* DBN untuk setiap *layer* yang merepresentasikan *output* adalah: 0.3753481, 0.0012443, 0.0031215, 6.21e-05, 2.05e-05, 0.0047576, 0.0004884, dan dapat ditunjukkan melalui visualisasi grafik pada Gambar 9.



Gambar 9 Grafik tingkat *error* DBN

Setelah *training*¸ dilakukan tahap *fine-tuning* yang merupakan evaluasi akurasi yang didapat pada *Backpropagation* menggunakan *weights* bernilai 0, nilai acak, dan *weights* dari DBN. Tabel 1 menunjukkan perbandingan akurasi prediksi pada Backpropagation berdasarkan masing-masing *weights*-nya.

Run	Akurasi dengan inisialisasi *weights* dari DBN	Akurasi dengan *weights* = 0	Akurasi dengan *weights* bilangan acak
1	18.994 %	0.0 %	0.0 %
2	25.698 %	0.0 %	0.0 %
3	18.063 %	0.0 %	0.0 %
4	17.877 %	0.0 %	0.0 %
5	22.905 %	0.0 %	0.0 %
6	28.678 %	0.0 %	0.0 %
7	17.877 %	0.0 %	0.0 %
8	26.071 %	0.0 %	0.0 %
9	19.367 %	0.0 %	0.0 %
10	19.367 %	0.0 %	0.0 %
11	20.484 %	0.0 %	0.0 %

12	28.119 %	0.0 %	0.0 %
13	18.063 %	0.0 %	0.0 %
14	34.823 %	0.0 %	0.0 %
15	20.298 %	0.0 %	0.0 %
16	25.698 %	0.0 %	0.0 %
17	27.002 %	0.0 %	0.0 %
18	19.367 %	0.0 %	0.0 %
19	21.601 %	0.0 %	0.0 %
20	21.974 %	0.0 %	0.0 %
Rata-rata	22.616 %	0.0 %	0.0 %

Tabel 1 Hasil akurasi tahap *fine-tuning*

4. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem ini, maka dapat diambil beberapa kesimpulan tentang bagaimana menggunakan algoritma *Deep Belief Network* (DBN) dalam mendeteksi penyakit diabetes, yaitu:

a. Proses pendeteksian diabetes menggunakan DBN dilakukan dengan beberapa langkah, yaitu persiapan data, *training* DBN, dan evaluasi dengan *Backpropagation* (BP).
b. DBN menunjukkan performa yang baik dalam melakukan klasterisasi dengan rata-rata nilai *error* 0.055%.
c. DBN dapat digunakan untuk mendeteksi penyakit diabetes dengan bekerja sebagai proses inisialisasi *weights* awal (*preprocessing*) untuk BP. Hal ini ditunjukkan pada hasil evaluasi BP menggunakan inisialisasi *weights* dari *training* DBN, di mana terdapat peningkatan nilai akurasi sebesar 22.616%, dibandingkan dengan BP tanpa *preprocessing* menggunakan *weights* 0 dan nilai acak yang memiliki tingkat akurasi 0.0%.

5. SARAN

Adapun saran yang dapat membantu pengembangan penelitian DBN untuk mendeteksi diabetes ini adalah perlunya dilakukan tahapan *training* pada *Backpropagation*

DAFTAR PUSTAKA

[1] Alom, Z., Bontupalli, V., & Taha, Tarek M. 2015. Intrusion Detection using Deep Belief Networks. *IEEE,* Vol. 15, 339 – 344

[2] Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, Hugo. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing system,* 153 – 160.

[3] Buscema, Massimo. 1998. Back Propagation Neural Networks. *Substance Use & Misuse,* Vol. 33, 233 – 270.

[4] International Diabetes Federation. 2019. *IDF Diabetes Atlas.* Brussels: International Diabetes Federation.

[5] Guariguata, L., Whiting DR, Hambleton, I., dkk. 2014. Diabetes Research and Clinical Practice. *Global estimates of diabetes prevalence for 2013 and projections for 2035*, 137 – 149.
[6] Hinton, G.E., Osindero, S., & Teh, Y.W. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*, Vol. 18, 1527 – 1554.
[7] Khandelwal, Renu. 2018. Deep Learning – Deep Belief Network (DBN). *Data Driven Investor*. Diakses melalui https://medium.com/datadriveninvestor/deep-learning-deep-belief-network-dbn-ab715b5b8afc. Diakses pada 8 April 2020.
[8] Larochelle, H., Erhan, D., Courville, A., dkk. 2007. An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, 473 – 480.
[9] Ligita, T., Wicking, K., Francis, K., dkk. 2009. PLOS ONE. *How people living with diabetes in Indonesia learn about their disease: A grounded theory study*, 1 – 9.
[10] Rojas, Raul, 1996, *Neural Networks: A Systematic Introduction*, Berlin, Springer.
[11] Wenhao, H., Song, G., dkk. 2014. Deep Architecture for Traffic Flow Prediction: Deep Belief Networks with Multitask Learning. *IEEE Transactions on Intelligent Transportation* Systems, 1 – 11.
[12] Xiao-Lei, Zhang & Ji, Wu. 2013. Deep Belief Networks Based Voice Activity Detection. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 21, 697 – 710.