INTI NUSA MANDIRI
DOI: https://doi.org/10.33480/inti.v14i2.1179

VOL. 14. NO. 2 FEBRUARI 2020
P-ISSN: 0216-6933 | E-ISSN: 2685-807X

NEURAL NETWORK FUNDAMENTAL DAN IMPLEMENTASI DALAM PEMROGRAMAN

Ibnu Akil

Sistem Informasi
Universitas Bina Sarana Informatika
Ibnu.ial@bsi.ac.id

Abstract—Nowadays machine learning and deep learning are becoming a trend in the world of information system. They are actually is part of artificial intelligence domain. However, so many people don't understand that machine learning and deep learning is built using neural network. Therefore, in order to understand how machine learning and deep learning works, we must understand the basic concept of neural network first. In this article the writer describes the basic theory, math function of neural network, and the example of implementation into java programming language. The writer hope that this article may help to understand neural network which is the core of machine learning and deep learning.

Keyword: *Neural Network, Machine Learning, Deep Learning, Java.*

Abstrak—Dewasa ini machine learning dan deep learning menjadi tren di dunia system informasi. Keduanya merupakan bagian dari domain artificial intelligence. Namun banyak yang belum memahami bahwa machine learning dan deep learning dibangun dengan menggunakan Neural Network. Karenanya untuk memahami bagaimana deep learning dan machine learning bekerja, terlebih dahulu kita harus memahami konsep dasar dari neural network. Dalam artikel ini penulis uraikan teori dasar, fungsi matematika dari neural network dan implementasi ke dalam Bahasa pemrograman java. Diharapkan artikel ini dapat membantu untuk memahami neural network yang menjadi inti dari machine learning dan deep learning.

Kata Kunci: Neural Network, Machine Learning, Deep Learning, Java.

PENDAHULUAN

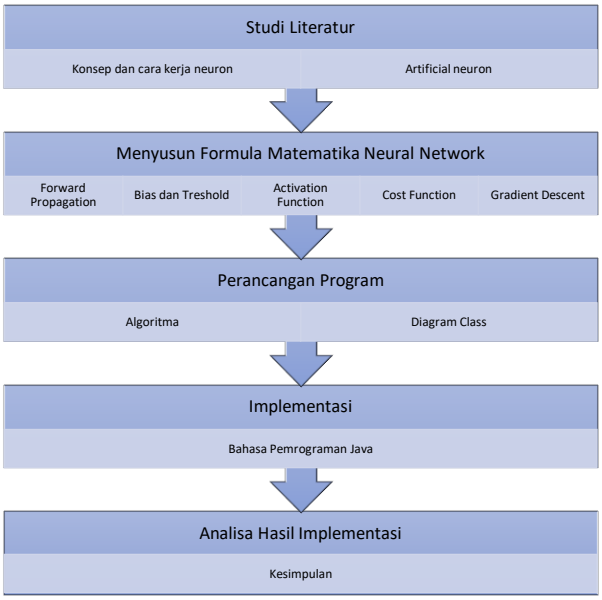
Dewasa ini Deep Learning (DL) merupakan topik yang sedang hangat dibicarakan dikalangan akademisi dan professional. Deep Learning merupakan salah satu cabang Machine Learning (ML) yang masuk dalam domain Artificial Intelligence(AI) (Sena, 2017). AI yang merupakan aplikasi dari ML, telah menyebar luas. Mulai dari mobil otomatis sampai ke penyeteman database mandiri, AI dan ML ditemukan dimana-mana. Semua factor-faktor diatas telah menempatkan pengembang dengan tingkat kemampuan rata-rata bekerja dalam tekanan untuk dapat menguasai ML. Terlepas dari minat dan rasa urgensi, pengembang berjuang untuk mempelajari keterampilan penting yang diperlukan untuk menguasai ML. (Janakiram, 2016)

Sebelum dapat menguasai DL dan ML terlebih dahulu kita harus memahami tentang Neural Network, yang merupakan dasar-dasar dari DL dan ML. Pemahaman terhadap Neural Network yang baik akan memberikan jalan bagi pengembang dan juga mahasiswa kearah yang tepat dalam memahami DL dan ML. Disini penulis mencoba untuk menerapkan Neural Network yang merupakan fondasi dari DL dan ML dalam Bahasa pemrograman Java mulai dari dasar, tanpa menggunakan framework atau library AI.

Tujuan dari penelitian ini adalah bagaimana mengimplementasikan konsep dasar neural network berdasarkan formula matematika kedalam Bahasa pemrograman untuk dapat dipahami cara kerjanya sehingga dapat diimplementasikan dalam aplikasi ML ataupun DL. Teori dasar, rumusan formula matematika, dan implementasi dalam bahasa pemrograman yang cukup lengkap mengenai Neural Network diharapkan dapat menjadi referensi bagi mahasiswa ataupun professional untuk bisa menguasai DL dan ML.

BAHAN DAN METODE

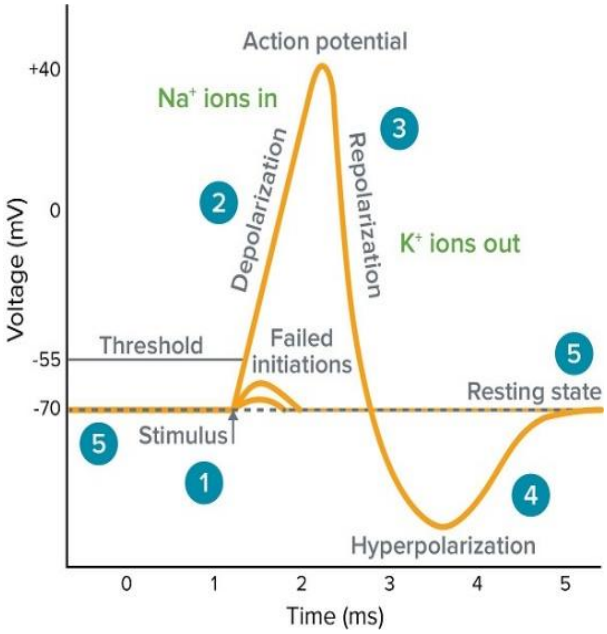
Penelitian ini termasuk dalam jenis penelitian terapan, dimana penulis mencoba untuk menerapkan model neural network dalam pemrograman yang sederhana. Berikut adalah kerangka kerja penelitian yang dilakukan:



Sumber: (Akil, 2019)

Gambar 1. Kerangka Kerja Penelitian

Arsitektur Neural Network



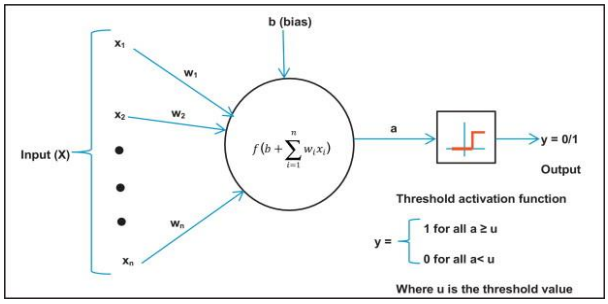
Sumber: (Molecular Devices, 2012)

Gambar 1. Model Artificial Neural Network.

Sebuah Artificial Neural Network (ANN), adalah termasuk algoritma pembelajaran terawasi (*supervised learning algorithm*) yang berarti bahwa kita harus menyediakan data masukan yang berisi variabel-variabel *independent* dan data output yang berisi variabel *dependent*. Pada awalnya ANN membuat prediksi-prediksi acak, prediksi-prediksi ini dibandingkan dengan output yang benar dan *error* (perbedaan antara nilai yang diprediksi dan nilai yang sesungguhnya) dikalkulasi. Fungsi yang mencari perbedaan antara nilai yang sesungguhnya dan nilai yang diperbanyak disebut *cost function*. Cost disini merujuk pada error. Tujuan kita adalah untuk meminimalisasi fungsi *cost*. Melatih neural network pada dasarnya adalah untuk meminimalkan fungsi *cost* (Chauhan, 2019).

1) Forward Propagation

Sebuah neural network dieksekusi dalam dua tahapan. Tahap *forward propagation* dan *back propagation*. Proses melewatkan data melalui ANN disebut sebagai *forward propagation*. Di dalam tahap *forward propagation*, prediksi-prediksi dubuat berdasarkan nilai-nilai di node-node masukan dan di bobot-bobot.



Sumber: (Deka & Quddus, 2014)

Gambar 2. Neuron Linear Tunggal.

Formula:

$$u = \sum_{i=1}^{n} w_i x_i \quad \text{.....(1)}$$

Dimana, u merujuk pada semua kasus, $\boldsymbol{w}$ adalah bobot, $\boldsymbol{x}$ adalah vektor input, ***Sigma*** adalah akumulasi penjumlahan.

2) Bias dan Threshold

Bias ditambahkan kedalam penjumlahan dari sigma, bias digunakan untuk memindahkan seluruh fungsi aktivasi ke kiri atau ke kanan untuk menhasilkan output yang diinginkan. Beberapa praktisi tidak menggunakan bobot bias, akan tetapi menggunakan *threshold* yang fix yaitu: 0 untuk fungsi aktivasi. Memasukan entah bias atau threshold pada dasarnya sama saja, untuk membentuk garis yang terpisah untuk melewati garis yang sebenarnya dalam fungsi aktivasi.

Dengan bias:

$$b + x_1 w_1 + x_2 w_2 = 0,$$
$$x_2 = \frac{w_1}{w_2} x_1 - \frac{b}{w_2} \quad \text{.....(2)}$$

Dengan threshold:

$$x_1 w_1 + x_2 w_2 = \emptyset,$$
$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\emptyset}{w_2} \quad \text{.....(3)}$$

3) Activation Function

Sigmoid function adalah salah satu fungsi yang biasa digunakan sebagai fungsi aktivasi dalam neural network. Sigmoid termasuk fungsi non-linear yang memiliki karakteristik berbentuk "S". pilihan standar untuk sigmoid adalah *logistic function* dengan formula sebagai berikut:

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}} \quad \text{.....(4)}$$

Dimana, σ adalah sigmoid activation function dan output yang kita dapatkan setelah forward propagation adalah nilai yang diprediksi disimbolkan dengan $\hat{y}$.

4) Back Propagation

Back propagation merujuk pada algoritma untuk menghitung gradien dari cost function dari bobot. Pada awalnya ANN membuat prediksi-prediksi yang acak yang tentu saja salah. Kemudian ANN membandingkan output yang terprediksi dengan output yang sebenarnya. Kemudian ANN memperbarui bobot-bobot dan bias sedemikian rupa sehingga output yang diprediksi mendekati output yang sebenarnya. Proses ini disebut back propagation atau secara umum disebut sebagai *training* atau *learning algorithm*.

5) Cost Function

Langkah pertama dalam tahapan ini adalah untuk menentukan cost dari prediksi-prediksi. Biasanya untuk mencari cost function digunakan *mean squared error* atau MSE.

Formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \quad \text{.....(5)}$$

6) Gradient Descent atau Cross-Entropy

Untuk dapat meminimalisasi cost umumnya dalam ANN digunakan *gradient descent* dengan fungsi cross-entropy. *Gradient descent* mencari nilai dari bobot-bobot dan bias yang terkecil.

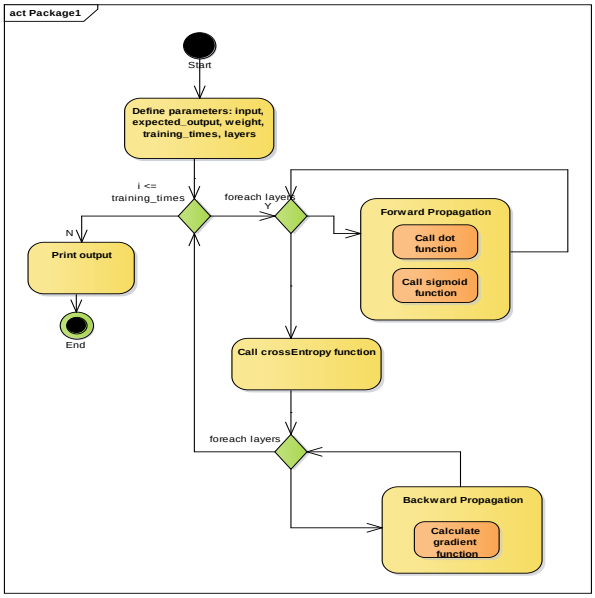
Formula:

$$J(a,y) = \sum_{i=1}^{m} L(a^i, y^i)$$

$$J(a,y) = -\sum_{i=1}^{m}(y^i \log(a^i) + (1 - y^i)\log(1 - a^i)) \quad \text{.....(6)}$$

Untuk menghitung turunan dari berkurangnya cost function terhadapt bobot dapat dilakukan dengan menggunakan chain rule (Deus 2018). Persamaan diatas memberitahu kita turunan parsial dari cost function terhadap masing-masing bobot dan bias dan mengurangi hasilnya dari bobot yang ada untuk mendapatkan bobot baru. Jika cost menaik dengan menaiknya bobot, nilai yang negative akan dikembalikan yang akan ditambahkan ke nilai bobot yang ada (Chauhan 2019).

7) Algoritma



Sumber: (Akil, 2019)

Gambar 3. Algoritma

HASIL DAN PEMBAHASAN

Untuk lebih memahami bagaimana ANN bekerja tentunya harus di-implementasikan didalam pemrograman, dalam hal ini akan kita coba menggunakan bahasa Java, penulis memilih java sebagai alternatif karena umumnya diluar sana banyak yang mengembangkan neural network dengan bahasa python. Bagaimana formula-formula tersebut diatas kita terjemahkan ke dalam pemrograman? Untuk lebih detailnya akan penulis bahas langkah demi langkah berurutan sesuai dengan tahap-tahap dalam ANN yang sudah dijelaskan diatas.

1) Dataset

Untuk uji coba neural network berikut adalah sample dataset yang digunakan.

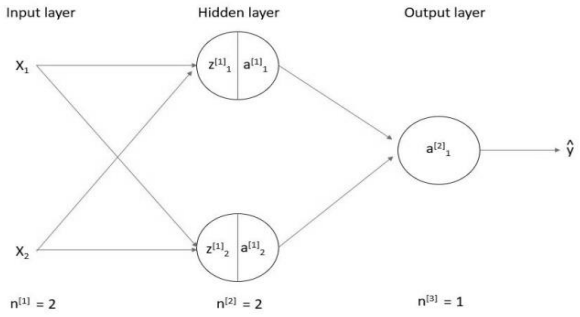
Table 1. Dataset.

Person	X1	X2	Y
Person-1	0	1	1
Person-2	0	0	0
Person-3	1	0	0
Person-4	1	1	1
Person-5	1	1	1
Person-6	0	1	0
Person-7	0	1	1

Sumber: (Akil, 2019)

Pada table diatas ada lima kolom yaitu: person, X1, X2, dan Y. Disini angka 1 berarti true, dan 0 berarti false. Kita akan mencoba ANN yang mampu memprediksikan nilai dari Y berdasarkan nilai dari X1 dan X2. Arsitekturnya terdiri dari satu layer

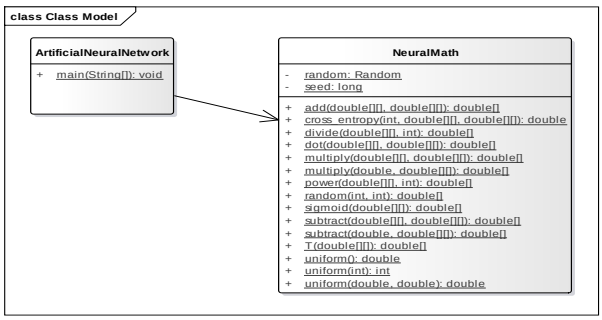
input, satu layer tersembunyi, dan satu layer output. Seperti gambar berikut:



Sumber: (Deus, 2018)
Gambar 3. Neural Network dengan 1 Hidden layer.

2) Arsitektur Sistem

Dalam uji coba ini kita hanya menggunakan dua kelas di dalam aplikasi. Satu kelas berisi fungsi-fungsi untuk operasi matematika yang digunakan dalam neural network yaitu NeuralMath dan satu kelas testing yaitu ArtificialNeuralNetwork.



Sumber: (Akil, 2019)
Gambar 4. Class Diagram.

Kode sumber kelas NeuralMath:

```
package myneural;
import java.util.Arrays;
import java.util.Random;

public class NeuralMath {
private static Random random;
private static long seed;

static {
seed = System.currentTimeMillis();
random = new Random(seed);
}

public static double uniform() {
return random.nextDouble();
}

public static int uniform(int n) {
if (n <= 0) {
throw new IllegalArgumentException("argument
must be positive: " + n);}
return random.nextInt(n);

public static double uniform(double a, double
b) {
if (!(a < b)) {
throw new IllegalArgumentException("invalid
range: [" + a + ", " + b + ")");}
return a + uniform() * (b - a);
}

public static double[][] random(int m, int n)
{
double[][] a = new double[m][n];
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
a[i][j] = uniform(0.0, 1.0);}}
return a;
}

public static double[][] T(double[][] a) {
int m = a.length;
int n = a[0].length;
double[][] b = new double[n][m];
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
b[j][i] = a[i][j]; }}
return b;
}

public static double[][] add(double[][] a,
double[][] b) {
int m = a.length;
int n = a[0].length;
double[][] c = new double[m][n];
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
c[i][j] = a[i][j] + b[i][j];
}}
return c;
}

public static double[][] subtract(double[][]
a, double[][] b) {
int m = a.length;
int n = a[0].length;
double[][] c = new double[m][n];
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
c[i][j] = a[i][j] - b[i][j]; }}
return c;
}

public static double[][] subtract(double a,
double[][] b) {
int m = b.length;
int n = b[0].length;
double[][] c = new double[m][n];
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
c[i][j] = a - b[i][j]; }}
return c;
}

public static double[][] dot(double[][] a,
double[][] b) {
int m1 = a.length;
int n1 = a[0].length;
int m2 = b.length;
int n2 = b[0].length;
if (n1 != m2) {
throw new RuntimeException("Illegal matrix
dimensions.");
}
double[][] c = new double[m1][n2];
for (int i = 0; i < m1; i++) {
for (int j = 0; j < n2; j++) {
for (int k = 0; k < n1; k++) {
```

VOL. 14. NO. 2 FEBRUARI 2020
P-ISSN: 0216-6933 | E-ISSN: 2685-807X

```
c[i][j] += a[i][k] * b[k][j]; }}}
return c;
}

public static double[][] multiply(double[][]
x, double[][] a) {
int m = a.length;
int n = a[0].length;

if (x.length != m || x[0].length != n) {
throw new RuntimeException("Illegal matrix
dimensions.");
}
double[][] y = new double[m][n];
for (int j = 0; j < m; j++) {
for (int i = 0; i < n; i++) {
y[j][i] = a[j][i] * x[j][i]; }}
return y;
}

public static double[][] multiply(double x,
double[][] a) {
int m = a.length;
int n = a[0].length;

double[][] y = new double[m][n];
for (int j = 0; j < m; j++) {
for (int i = 0; i < n; i++) {
y[j][i] = a[j][i] * x; }}
return y;
}

public static double[][] power(double[][] x,
int a) {
int m = x.length;
int n = x[0].length;

double[][] y = new double[m][n];
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
y[i][j] = Math.pow(x[i][j], a);
}}
return y;
}

public static double[][] sigmoid(double[][]
a) {
int m = a.length;
int n = a[0].length;
double[][] z = new double[m][n];

for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
z[i][j] = (1.0 / (1 + Math.exp(-a[i][j])));
}}
return z;
}

public static double[][] divide(double[][] x,
int a) {
int m = x.length;
int n = x[0].length;

double[][] z = new double[m][n];

for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
z[i][j] = (x[i][j] / a);
}}
return z;
}

public static double cross_entropy(int
batch_size, double[][] Y, double[][] A) {
int m = A.length;
int n = A[0].length;
double[][] z = new double[m][n];

for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
z[i][j] = (Y[i][j] * Math.log(A[i][j])) + ((1
- Y[i][j]) * Math.log(1 - A[i][j]));
}}

double sum = 0;
for (int i = 0; i < m; i++) {
for (int j = 0; j < n; j++) {
sum += z[i][j];
}}
return -sum / batch_size;
}}
```

Kode sumber kelas ArtificialNeuralNetwork:

```
package myneural;
import java.util.Arrays;

public class ArtificialNeuralNetwork {
public static void main(String[] args) {

double[][] X = {{0,1}, {0,0}, {1,0}, {1,1},
{1,1}, {0,1}, {0,1}};
double[][] Y = {{1},{0},{0},{1},{1},{0},{1}};
int m = 7;
int nodes = 400;
NeuralMath np = new NeuralMath();
X = np.T(X);
Y = np.T(Y);

double[][] W1 = np.random(nodes, 2);
double[][] b1 = new double[nodes][m];
double[][] W2 = np.random(1, nodes);
double[][] b2 = new double[1][m];

for (int i = 0; i < 4000; i++) {
// Foward Prop
// LAYER 1
double[][] Z1 = np.add(np.dot(W1, X), b1);
double[][] A1 = np.sigmoid(Z1);

//LAYER 2
double[][] Z2 = np.add(np.dot(W2, A1), b2);
double[][] A2 = np.sigmoid(Z2);

double cost = np.cross_entropy(m, Y, A2);
//costs.getData().add(new XYChart.Data(i,
cost));
// Back Prop
//LAYER 2
double[][] dZ2 = np.subtract(A2, Y);
double[][] dW2 = np.divide(np.dot(dZ2,
np.T(A1)), m);
double[][] db2 = np.divide(dZ2, m);

//LAYER 1
double[][] dZ1 = np.multiply(np.dot(np.T(W2),
dZ2), np.subtract(1.0, np.power(A1, 2)));
double[][] dW1 = np.divide(np.dot(dZ1,
np.T(X)), m);
double[][] db1 = np.divide(dZ1, m);

// G.D
W1 = np.subtract(W1, np.multiply(0.01, dW1));
b1 = np.subtract(b1, np.multiply(0.01, db1));
W2 = np.subtract(W2, np.multiply(0.01, dW2));
b2 = np.subtract(b2, np.multiply(0.01, db2));

if (i % 400 == 0) {
System.out.println("==============");
```

```
System.out.println("Cost = " + cost);
System.out.println("Predictions = " +
Arrays.deepToString(A2));
}}}}
```

Sumber: (Deus 2018)

3) Hasil Training

Berikut adalah hasil dari training neural network untuk 4000 kali perulangan. Output program:

```
==============
Cost = 0.23478588766730996
Predictions = [[0.8846351991050798,
0.2769768434018224, 0.21722001635294497,
0.7888504151641689, 0.7888504151641689,
0.29866288111016814, 0.8846351991050798]]
…
…
==============
Cost = 0.019242272299291163
Predictions = [[0.9864665086989707,
0.014703053989306273, 0.02152316653418999,
0.9827324679847836, 0.9827324679847836,
0.035395403610008314, 0.9864665086989707]]
```

Kita bisa melihat dengan jelas bahwa neural network telah berhasil memprediksi output yang mendekati nilai Y pada dataset sebelumnya. Jika nilai-nilai tersebut dibulatkan maka akan mendekati 1 atau 0, sesuai dengan nilai Y (1,0,0,1,1,0,1). Kita juga bisa melihat penurunan dari cost function seperti contoh diputaran pertama nilai prediksi sebesar: 0.8846351991050798 sedangkan pada putaran terkahir nilainya semakin mendekatai 1 yaitu: 0.9864665086989707.

KESIMPULAN

Artificial neural network meski tidak bekerja seratus persen seperti neuron pada otak manusia, namun dari hasil implementasi pada program diatas dapat kita lihat, bahwa ANN dapat memprediksi nilai dari output yang diharapkan, setelah ditraining sebanyak 4000 perulangan (learning time) dan dapat disimpulkan bahwa system dengan menggunakan konsep ANN dapat belajar layaknya otak manusia, yang mana neural network ini menjadi landasan bagi pengembangan sistem-sistem *machine learning* dan *deep learning*. Dalam uji coba disini penulis hanya menggunakan dua layer, untuk pengembangan riset lebih lanjut bisa menggunakan lebih dari dua layer dengan merubah sedikit pada fungsi-fungsi matematika pada kelas NeuralMath dan dengan studi kasus yang lebih nyata.

REFERENSI

Chauhan, N. S. (2019). *Build an Artificial Neural Network From Scratch: Part 1*. https://www.kdnuggets.com/2019/11/build-artificial-neural-network-scratch-part-1.html

Deka, L., & Quddus, M. (2014). Network-level accident-mapping: Distance based pattern matching using artificial neural network. *Accident Analysis and Prevention*, *65*, 105–113. https://doi.org/10.1016/j.aap.2013.12.001

Deus, J. (2018). *Implementing an Artificial Neural Network in Pure Java (No external dependencies)*. https://medium.com/coinmonks/implementing-an-artificial-neural-network-in-pure-java-no-external-dependencies-975749a38114

Janakiram. (2016). *Why Do Developers Find It Hard To Learn Machine Learning?* https://www.forbes.com/sites/janakiramsmsv/2018/01/01/why-do-developers-find-it-hard-to-learn-machine-learning/#74e353dd6bf6

Molecular Devices. (2012). *What is an Action Potential?* Molecular Devices. https://www.moleculardevices.com/applications/patch-clamp-electrophysiology/what-action-potential#gref

Sena, S. (2017). *Pengenalan Deep Learning Part 2 : Multilayer Perceptron*. https://medium.com/@samuelsena/pengenalan-deep-learning-part-2-multilayer-perceptron-e8f98d625b09