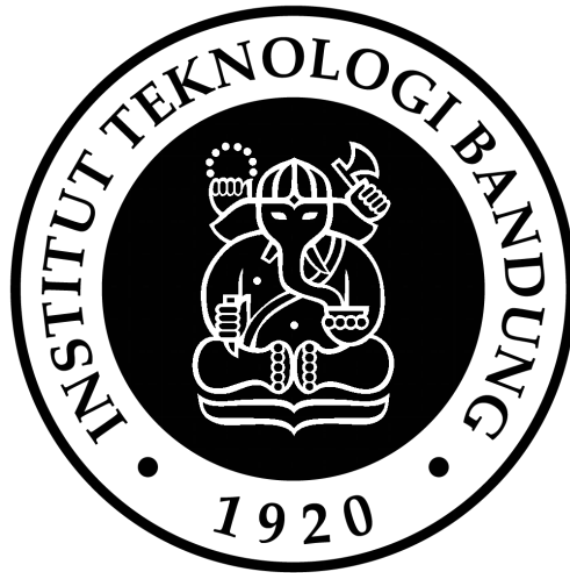


# **TUGAS BESAR**

**IF3270: Pembelajaran Mesin**

**Bagian A: Implementasi *Forward Propagation* untuk *Feed Forward*  
*Neural Network***



**Dibuat oleh:**

**13520060 - Rheza Rizqullah Ecaldy - K02**

**13520072 - Jova Andres Riski Sirait - K02**

**13520111 - Rizky Akbar Asmaran - K02**

**13520132 - Januar Budi Ghifari - K03**

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG**

**2022**

# 1. Penjelasan Implementasi

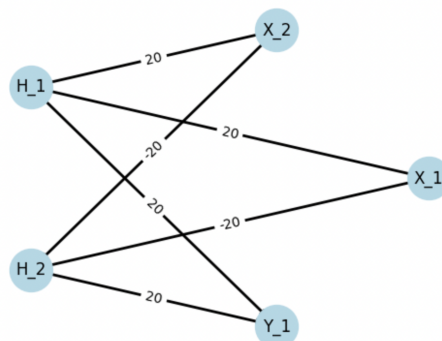
Program ini merupakan implementasi *forward propagation* untuk *Feed Forward Neural Network*. Sebelum menjalankan program ini, diperlukan instalasi python dan jupyter notebook. Program ini juga membutuhkan beberapa modul python, yaitu json, numpy, networkx, matplotlib.pyplot. Modul ini dapat di-*install* dengan menggunakan perintah “pip install” pada *command prompt*.

Program ini menerima input berupa model FFNN, nilai x yang akan diprediksi, serta nilai y sebenarnya dari nilai x yang ingin diprediksi. Model FFNN disimpan dalam bentuk JSON yang akan di-*load* ke dalam program. Berikut merupakan contoh file JSON dari model.

```
{
  "layers": [
    {
      "type": "input",
    },
    {
      "type": "hidden",
      "n_neuron": 2,
      "activation_function": "relu",
      "weight": [
        [
          1,
          1
        ],
        [
          1,
          1
        ]
      ],
      "bias": [
        0,
        -1
      ]
    },
    {
      "type": "output",
      "n_neuron": 1,
      "activation": "linear",
      "weight": [
        [
          1
        ],
        [
          -2
        ]
      ],
      "bias": [
        0
      ]
    }
  ]
}
```

Gambar 1.1 Contoh File JSON untuk Menyimpan Model FFNN

Model FFNN ini juga bisa divisualisasikan dalam bentuk *explicit model* seperti gambar di bawah.



Gambar 1.2 Bentuk Visualisasi Model FFNN

Dalam implementasi program ini, dibuat 2 kelas, yaitu LoadModel dan FFNN. Kelas LoadModel merupakan kelas yang merepresentasikan model yang di-load dari file JSON. Kelas ini memiliki beberapa atribut, yaitu weights yang merepresentasikan nilai *weight* untuk setiap input pada setiap layer, biases yang merepresentasikan nilai bias untuk setiap node pada setiap layer, dan activations yang merepresentasikan fungsi aktivasi setiap layer. Kelas ini juga memiliki fungsi get untuk setiap atributnya dan fungsi print\_model yang menampilkan atribut-atribut dari model.

Kelas FFNN merupakan kelas yang melaksanakan fungsi utama untuk *forward propagation*. Kelas ini memiliki beberapa atribut, yaitu model, weights, biases, dan activations. Kelas ini juga memiliki beberapa fungsi, yaitu forward\_propagation, predict, accuracy, print\_model, dan draw\_network. Fungsi forward\_propagation menerima input berupa nilai x kemudian melakukan perhitungan *forward propagation* untuk menghasilkan hasil akhir setelah output layer. Fungsi predict menerima input berupa nilai X, baik 1 *instance* maupun *batch*, kemudian fungsi ini akan memanggil fungsi forward\_propagation. Fungsi accuracy merupakan fungsi yang membandingkan nilai prediksi oleh program dengan nilai y sebenarnya. Kami menggunakan nilai threshold 0.5. Nilai perhitungan pada output layer yang lebih besar atau sama dengan dari nilai threshold dianggap 1 dan selebihnya 0. Fungsi print\_model merupakan fungsi yang menampilkan nilai atribut dari model. Fungsi draw\_network merupakan fungsi untuk memvisualisasikan model menjadi bentuk *explicit model*.

Secara umum, untuk melakukan prediksi terhadap suatu nilai X dengan menggunakan model tertentu, ikuti alur sebagai berikut.

```
# isi nama file json
filename = 'sigmoid'
model_path = filename + '.json'

# load model
model = FFNN(model_path)

# masukkan input nilai X
X = np.array([[0,0], [0,1], [1,0], [1,1]])

# predict output berdasarkan input layer X
y_pred = model.predict(X)

# menghitung accuracy
y_true = np.array([1, 0, 0, 1])
accuracy = model.acurracy(X, y_true)

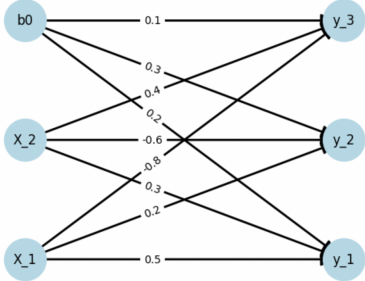
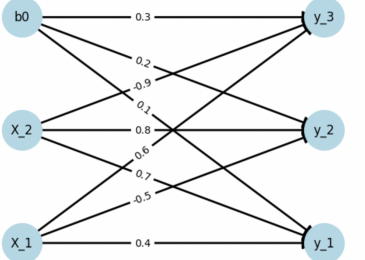
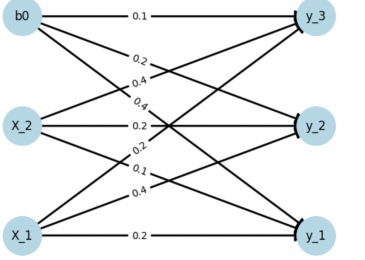
model.draw_network()

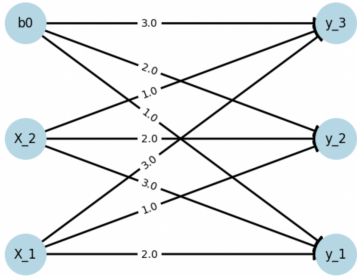
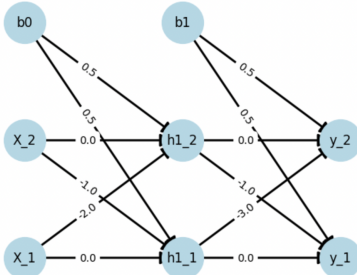
# print predictions and accuracy
print('Predictions: \n', y_pred)
print('Accuracy: {:.2f}%'.format(accuracy*100))
```

Gambar 1.3 Alur umum penggunaan program

## 2. Hasil Pengujian

Berikut adalah hasil pengujian berdasarkan kasus uji yang telah kami siapkan.

No.	Jenis Input	Fungsi Aktivasi	Input	Expected Output	Screenshot Hasil Pengujian
1.	1 instance	linear	[3.0, 1.0]	[ 2.0, 0.3, -1.9]	<p>Enter the filename: linear  Instance: [3. 1.]  Layer 1 Calculated: [ 2. 0.3 -1.9]  Predictions: [[1 0 0]]</p> 
2.	1 instance	relu	[-1.0, 0.5]	[0.05, 1.1, 0.0]	<p>Enter the filename: relu  Instance: [-1. 0.5]  Layer 1 Calculated: [0.05 1.1 0. ]  Predictions: [[0 1 0]]</p> 
3.	1 instance	sigmoid	[0.2, 0.4]	[0.617747], [0.589040], [0.574442]	<p>Enter the filename: sigmoid  Instance: [0.2 0.4]  Layer 1 Calculated: [0.61774787 0.58904043 0.57444252]  Predictions: [[1 1 1]]</p> 

4.	1 instance	softmax	[1.0, 2.0]	[0.665241], [0.090031], [0.244728]	<div>Enter the filename: softmax Instance: [1. 2.] Layer 1 Calculated: [0.66524096 0.09003057 0.24472847] Predictions: [[1 0 0]]</div> 
5.	Batch	linear + relu	[ [1.0, 0.0], [0.0, 1.0], [0.0, 0.0] ]	[ [2.0, 0.0], [0.0, 2.0], [0.0, 0.0] ]	<div>Enter the filename: multilayer Instance: [1. 0.] Layer 1 Calculated: [ 0.5 -1.5] Layer 2 Calculated: [2. 0.]  Instance: [0. 1.] Layer 1 Calculated: [-0.5 0.5] Layer 2 Calculated: [0. 2.]  Instance: [0. 0.] Layer 1 Calculated: [0.5 0.5] Layer 2 Calculated: [0. 0.]  Predictions: [[1 0] [0 1] [0 0]]</div> 

### 3. Perbandingan dengan Hasil Perhitungan Manual

Berikut adalah perbandingan hasil prediksi oleh program dengan hasil perhitungan manual.

No.	Fungsi Aktivas i	Hasil Program	Perhitungan Manual																																																				
1.	linear	Enter the filename: linear Instance: [3. 1.] Layer 1 Calculated: [ 2.    0.3 -1.9]  Predictions: [[1 0 0]]	<table><tr><td colspan="6">Linear</td></tr><tr><td>x0</td><td>x1</td><td>x2</td><td></td><td></td><td></td></tr><tr><td>1</td><td>3</td><td>1</td><td></td><td></td><td></td></tr><tr><td>Σy1</td><td>y1</td><td>Σy2</td><td>y2</td><td>Σy3</td><td>y3</td></tr><tr><td>2</td><td>2</td><td>0.3</td><td>0.3</td><td>-1.9</td><td>-1.9</td></tr></table>	Linear						x0	x1	x2				1	3	1				Σy1	y1	Σy2	y2	Σy3	y3	2	2	0.3	0.3	-1.9	-1.9																						
Linear																																																							
x0	x1	x2																																																					
1	3	1																																																					
Σy1	y1	Σy2	y2	Σy3	y3																																																		
2	2	0.3	0.3	-1.9	-1.9																																																		
2.	relu	Enter the filename: relu Instance: [-1.    0.5] Layer 1 Calculated: [0.05 1.1    0. ]  Predictions: [[0 1 0]]	<table><tr><td colspan="6">Relu</td></tr><tr><td>x0</td><td>x1</td><td>x2</td><td></td><td></td><td></td></tr><tr><td>1</td><td>-1</td><td>0.5</td><td></td><td></td><td></td></tr><tr><td>Σy1</td><td>y1</td><td>Σy2</td><td>y2</td><td>Σy3</td><td>y3</td></tr><tr><td>0.05</td><td>0.05</td><td>1.1</td><td>1.1</td><td>-0.75</td><td>0</td></tr></table>	Relu						x0	x1	x2				1	-1	0.5				Σy1	y1	Σy2	y2	Σy3	y3	0.05	0.05	1.1	1.1	-0.75	0																						
Relu																																																							
x0	x1	x2																																																					
1	-1	0.5																																																					
Σy1	y1	Σy2	y2	Σy3	y3																																																		
0.05	0.05	1.1	1.1	-0.75	0																																																		
3.	sigmoid	Enter the filename: sigmoid Instance: [0.2 0.4] Layer 1 Calculated: [0.61774787 0.58904043 0.57444252]  Predictions: [[1 1 1]]	<table><tr><td colspan="6">Sigmoid</td></tr><tr><td>x0</td><td>x1</td><td>x2</td><td></td><td></td><td></td></tr><tr><td>1</td><td>0.2</td><td>0.4</td><td></td><td></td><td></td></tr><tr><td>Σy1</td><td>y1</td><td>Σy2</td><td>y2</td><td>Σy3</td><td>y3</td></tr><tr><td>0.48</td><td>0.61774</td><td>0.36</td><td>0.58904</td><td>0.3</td><td>0.57444</td></tr></table>	Sigmoid						x0	x1	x2				1	0.2	0.4				Σy1	y1	Σy2	y2	Σy3	y3	0.48	0.61774	0.36	0.58904	0.3	0.57444																						
Sigmoid																																																							
x0	x1	x2																																																					
1	0.2	0.4																																																					
Σy1	y1	Σy2	y2	Σy3	y3																																																		
0.48	0.61774	0.36	0.58904	0.3	0.57444																																																		
4.	softmax	Enter the filename: softmax Instance: [1. 2.] Layer 1 Calculated: [0.66524096 0.09003057 0.24472847]  Predictions: [[1 0 0]]	<table><tr><td colspan="6">Softmax</td></tr><tr><td>x0</td><td>x1</td><td>x2</td><td></td><td></td><td></td></tr><tr><td>1</td><td>1</td><td>2</td><td></td><td></td><td></td></tr><tr><td>Σy1</td><td>y1</td><td>Σy2</td><td>y2</td><td>Σy3</td><td>y3</td></tr><tr><td>9</td><td>0.66524</td><td>7</td><td>0.09003</td><td>8</td><td>0.24472</td></tr></table>	Softmax						x0	x1	x2				1	1	2				Σy1	y1	Σy2	y2	Σy3	y3	9	0.66524	7	0.09003	8	0.24472																						
Softmax																																																							
x0	x1	x2																																																					
1	1	2																																																					
Σy1	y1	Σy2	y2	Σy3	y3																																																		
9	0.66524	7	0.09003	8	0.24472																																																		
5.	linear + relu	Enter the filename: multilayer Instance: [1. 0.] Layer 1 Calculated: [ 0.5 -1.5] Layer 2 Calculated: [2. 0.]  Instance: [0. 1.] Layer 1 Calculated: [-0.5    0.5] Layer 2 Calculated: [0. 2.]  Instance: [0. 0.] Layer 1 Calculated: [0.5 0.5] Layer 2 Calculated: [0. 0.]  Predictions: [[1 0] [0 1] [0 0]]	<table><tr><td colspan="4">Multilayer (Linear + Relu)</td></tr><tr><td>x0</td><td>x1</td><td>x2</td><td></td></tr><tr><td>1</td><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>0</td><td></td></tr><tr><td>Σh1</td><td>h1</td><td>Σh2</td><td>h2</td></tr><tr><td>0.5</td><td>0.5</td><td>-1.5</td><td>-1.5</td></tr><tr><td>-0.5</td><td>-0.5</td><td>0.5</td><td>0.5</td></tr><tr><td>0.5</td><td>0.5</td><td>0.5</td><td>0.5</td></tr><tr><td>Σy1</td><td>y1</td><td>Σy2</td><td>y2</td></tr><tr><td>2</td><td>2</td><td>-1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>2</td><td>2</td></tr><tr><td>0</td><td>0</td><td>-1</td><td>0</td></tr></table>	Multilayer (Linear + Relu)				x0	x1	x2		1	1	0		1	0	1		1	0	0		Σh1	h1	Σh2	h2	0.5	0.5	-1.5	-1.5	-0.5	-0.5	0.5	0.5	0.5	0.5	0.5	0.5	Σy1	y1	Σy2	y2	2	2	-1	0	0	0	2	2	0	0	-1	0
Multilayer (Linear + Relu)																																																							
x0	x1	x2																																																					
1	1	0																																																					
1	0	1																																																					
1	0	0																																																					
Σh1	h1	Σh2	h2																																																				
0.5	0.5	-1.5	-1.5																																																				
-0.5	-0.5	0.5	0.5																																																				
0.5	0.5	0.5	0.5																																																				
Σy1	y1	Σy2	y2																																																				
2	2	-1	0																																																				
0	0	2	2																																																				
0	0	-1	0																																																				

Dapat dilihat dari tabel di atas, Hasil perhitungan program dan perhitungan manual kurang lebih sama.

## 4. Pembagian Tugas

Berikut adalah pembagian tugas yang dilakukan oleh kelompok kami.

No.	NIM	Nama	Kontribusi
1.	13520060	Rheza Rizqullah Ecaldy	Model neural network, Input file, Fungsi FFNN, Visualisasi model, Prediksi output, Laporan
2.	13520072	Jova Andres Riski Sirait	Model neural network, Input file, Fungsi FFNN, Visualisasi model, Prediksi output, Laporan
3.	13520111	Rizky Akbar Asmaran	Model neural network, Input file, Fungsi FFNN, Visualisasi model, Prediksi output, Laporan
4.	13520132	Januar Budi Ghifari	Model neural network, Input file, Fungsi FFNN, Visualisasi model, Prediksi output, Laporan

