

## **Tugas Besar IF3270 Pembelajaran Mesin**

Implementasi Forward Propagation untuk Feed Forward Neural Network

Tugas Besar Bagian A



Samuel Christopher Swandi 13520075

Grace Claudia 13520078

Ubaidillah Ariq Prathama 13520085

Patrick Amadeus Irawan 13520109

**PROGRAM STUDI SARJANA TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2023**

# Daftar Isi

<b>Daftar Isi.....</b>	<b>1</b>
<b>I. Penjelasan Implementasi.....</b>	<b>2</b>
Penjelasan struktur folder.....	3
Folder test.....	4
- Folder input.....	4
- Folder model.....	4
Folder src.....	7
- Library and dependencies.....	7
- Class.....	8
- Model Load Section.....	9
- Loading Weights.....	10
- Predict Output.....	10
- Neural Network Visualization.....	10
<b>II. Hasil Pengujian.....</b>	<b>11</b>
1. Pengujian test case buatan sendiri Linear - Sigmoid (xor_3.txt).....	11
2. Pengujian test case buatan sendiri ReLU - Softmax (xor_2.txt).....	14
3. Pengujian test case buatan sendiri Softmax - ReLU (xor_4.txt).....	17
4. Pengujian test case asisten: linear_tc.txt.....	20
5. Pengujian test case asisten: relu_tc.txt.....	21
6. Pengujian test case asisten: sigmoid_tc.txt.....	22
7. Pengujian test case asisten: softmax_tc.txt.....	23
8. Pengujian test case asisten: multilayer_tc.txt.....	24
<b>III. Perbandingan Hasil Perhitungan Manual.....</b>	<b>25</b>
1. Pengujian test case buatan sendiri: Linear - Sigmoid (xor_3.txt).....	25
2. Pengujian test case buatan sendiri: ReLU - Softmax (xor_.txt).....	27
3. Pengujian test case buatan sendiri: Softmax - ReLU (xor_4.txt).....	29
<b>IV. Pembagian Tugas.....</b>	<b>31</b>

## I. Penjelasan Implementasi

Program Feed Forward Neural Network ini dibuat dalam sebuah jupyter notebook. Input yang diberikan merupakan dua buah file, yaitu file untuk konfigurasi model neural network dan file untuk input data yang akan diprediksi. Sedangkan, output yang dihasilkan adalah sebuah file html yang memvisualisasikan neural network. Jika node (neuron) *dihover* akan muncul nilai prediksi dari node tersebut. Jika edge *dihover* akan muncul weight dari node tersebut.

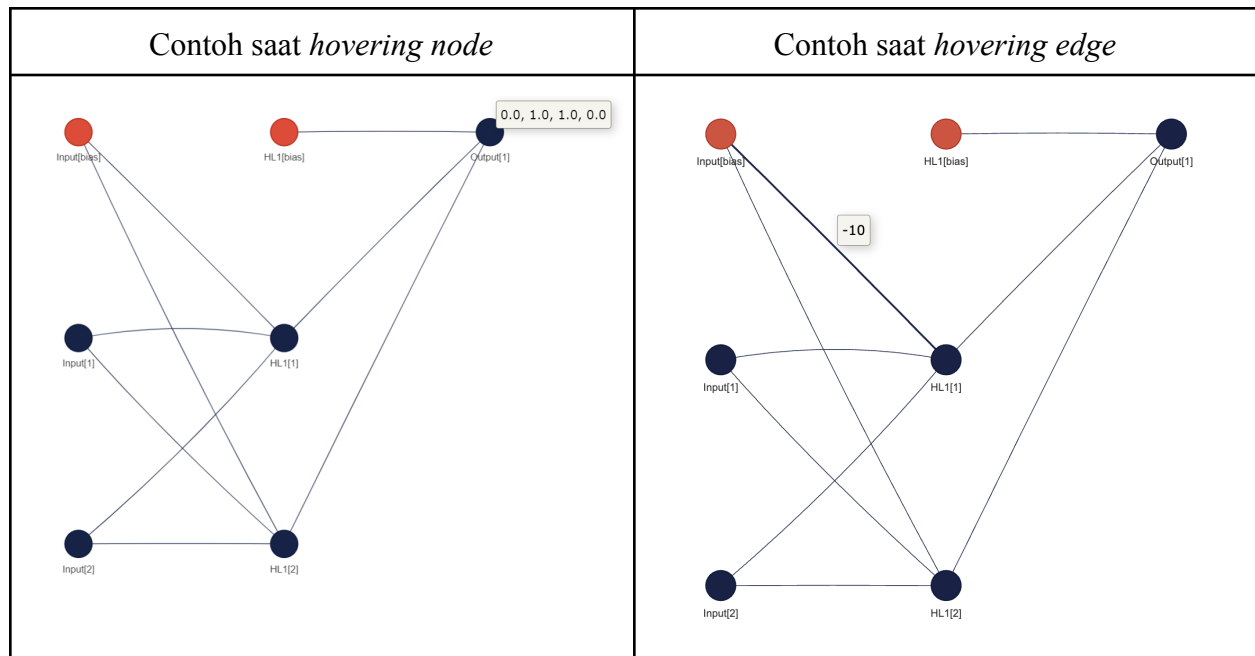
```
test > model > ≡ xor.txt
1  1
2  2 2 2
3 -10 30
4  20 -20
5  20 -20
6  2 1 2
7 -30
8  20
9  20
```

File input untuk model neural network pada baris pertama merupakan N, yaitu banyaknya layer dari neural network. Selanjutnya N-1 input berikutnya akan merepresentasikan weight pada setiap layer. Pada baris pertama input ke-i merupakan A B C yang merepresentasikan banyaknya neuron pada layer ke-i, banyaknya neuron pada layer ke-i+1, dan jenis fungsi aktivasi. Fungsi aktivasi *dimapping* sebagai berikut {0: linear, 1: ReLU, 2: sigmoid, 3: softmax}. Selanjutnya input dilanjutkan berupa weight dengan dimensi (A+1 x B) yang merupakan weight pada layer ke-i.

```
test > input > ≡ input1.txt
1  1 0 0
2  1 0 1
3  1 1 0
4  1 1 1
```

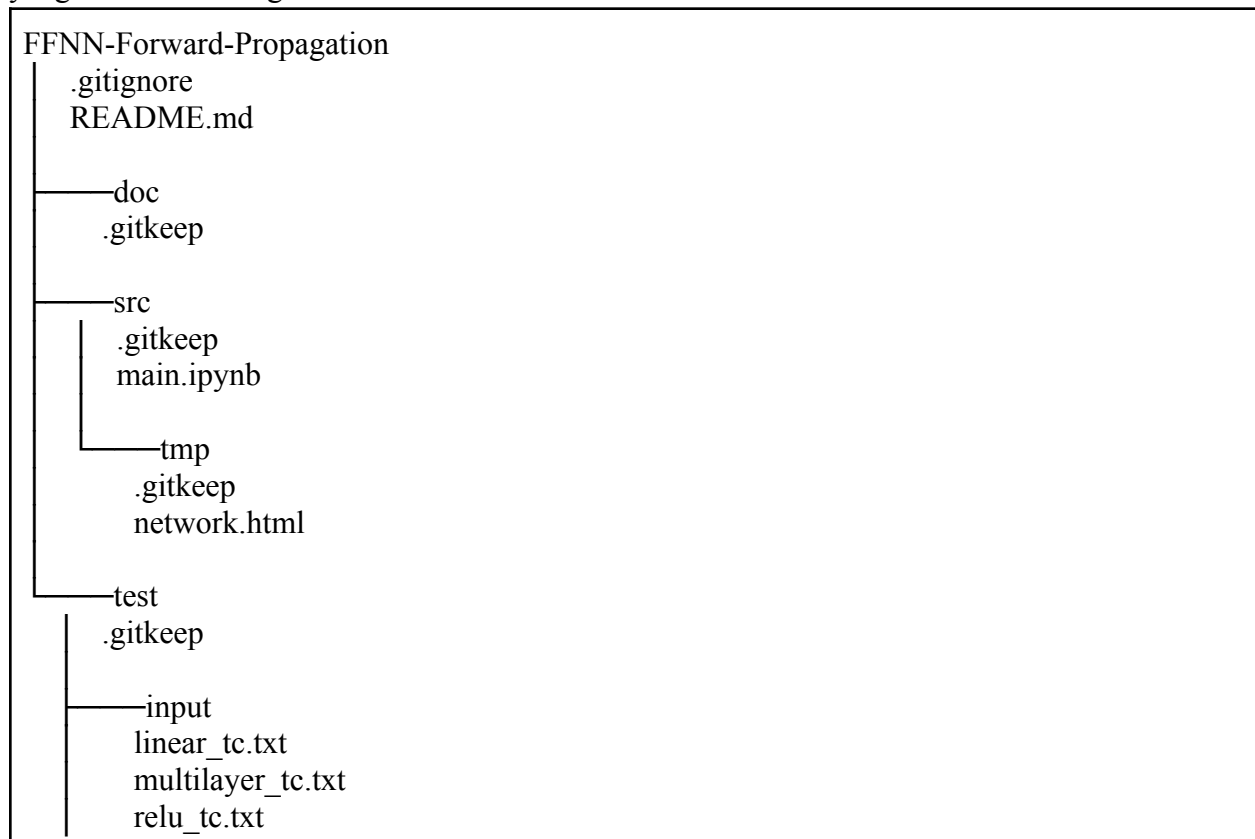
File input untuk input prediksi cukup sederhana. Setiap baris input, merepresentasikan instance yang akan diprediksi. Dengan urutan bias, input-1, input-2, dst.

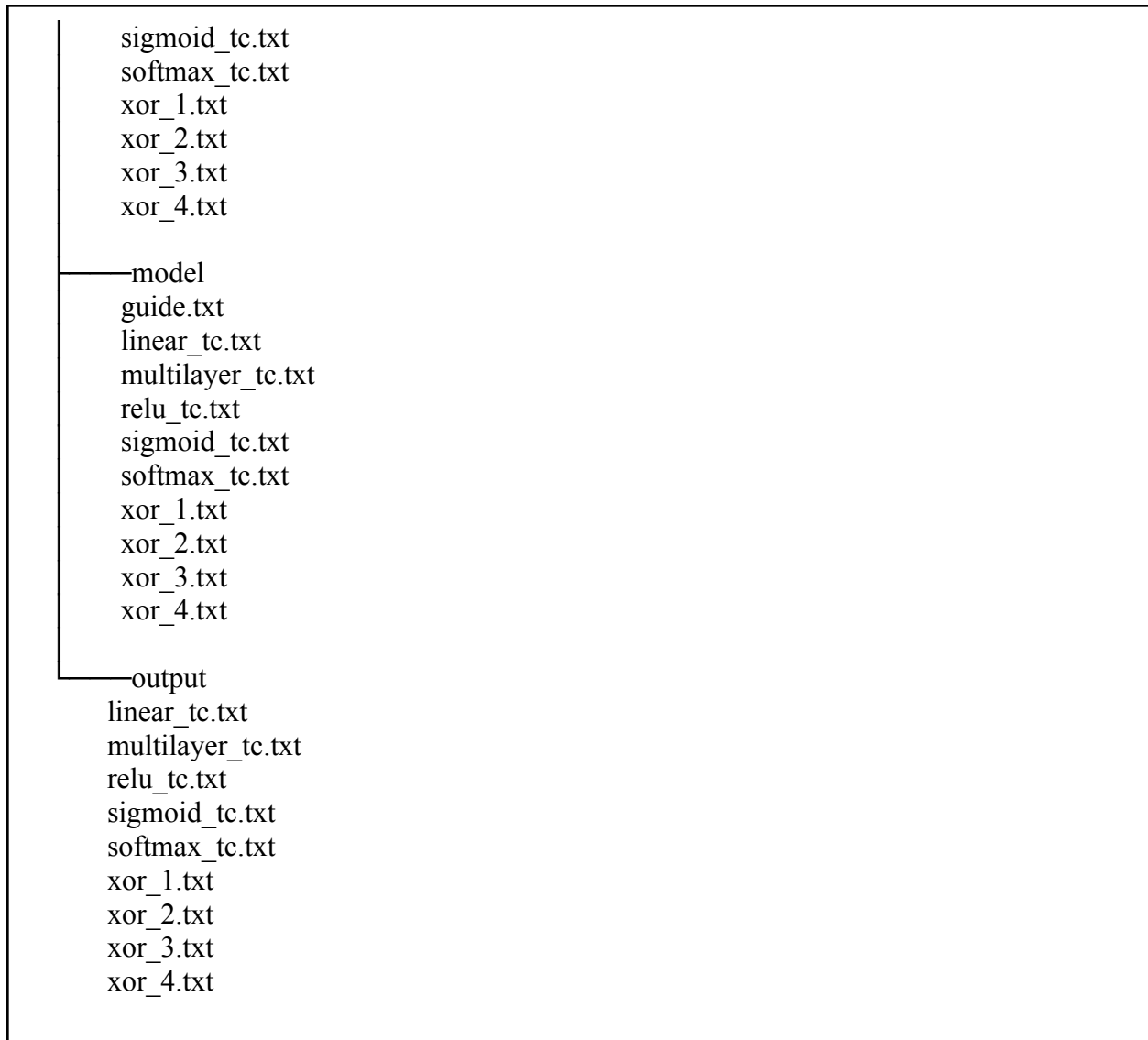
File output berupa visualisasi neural network dalam html. Node dapat digeser dengan cara didrag menggunakan mouse. Setiap node memiliki nama masing-masing yang dapat dilihat di bawah node. Jika node (neuron) *di hover* akan muncul nilai prediksi dari node tersebut. Jika edge *dihover* akan muncul weight dari node tersebut.



## Penjelasan struktur folder

Dalam pengimplementasian Forward Propagation untuk FFNN ini, berikut struktur program yang kami telah bangun:





## Folder test

Folder test merupakan folder berisi .txt yang meliputi masukan dan struktur model file .txt yang bersesuaian dengan program yang kami bangun:

- Folder input  
Input merupakan folder berisikan masukan untuk model yang telah dibangun. Input model dilakukan dengan membaca file .txt pada folder ini.
- Folder model  
Model merupakan folder berisikan struktur dari FFNN yang telah dirancang dalam ekstensi .txt agar memenuhi implementasi model yang telah kami buat. Model yang dibangun memiliki format sebagai berikut:

## FORMAT MODEL

[Jumlah layer (termasuk input dan output)]

// untuk setiap layer:

[Jumlah node pada current layer] [Jumlah layer pada layer selanjutnya] [pilihan activation function(0-3)]

for i in range[0...currNodeLength]

for j in [1...dstNodeLength]

W(i,1)...W(i,j); // i = node asal; j = node tujuan

Pilihan *activation function*:

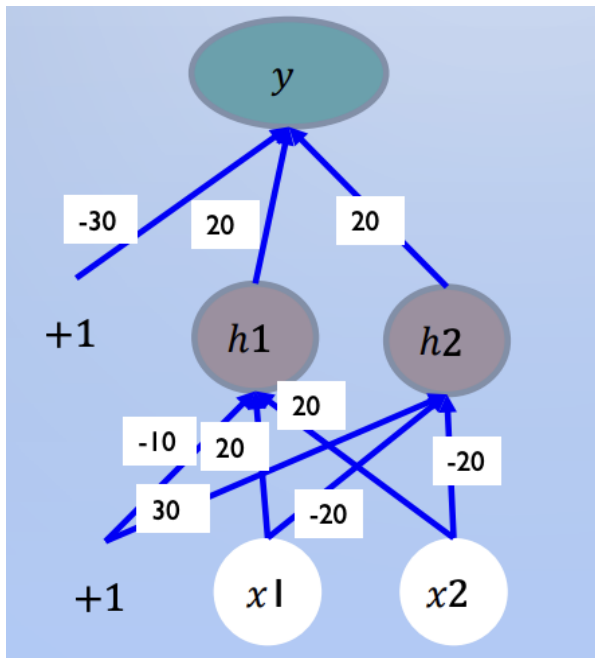
- 0 → linear
- 1 → ReLU
- 2 → sigmoid
- 3 → softmax

Panduan format juga tersedia pada file guide.txt

Berikut cara contoh penggunaan format:

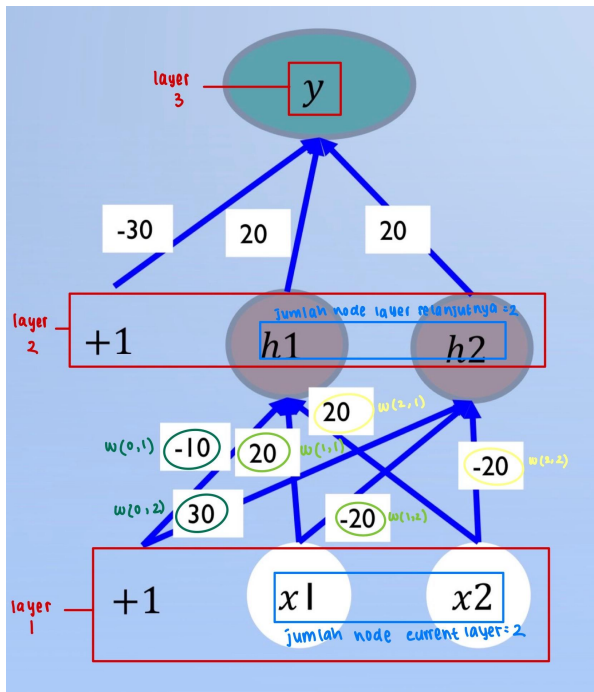
Contoh: XOR (SIGMOID MODEL)

### XOR (SIGMOID MODEL)



```
3
2 2 2
-10 30
20 -20
20 -20
2 1 2
-30
20
20
```

Langkah-langkah membangun .txt:



Pertama-tama kita hitung ada berapa layer, didapatkan ada 3 layer maka update .txt menjadi:

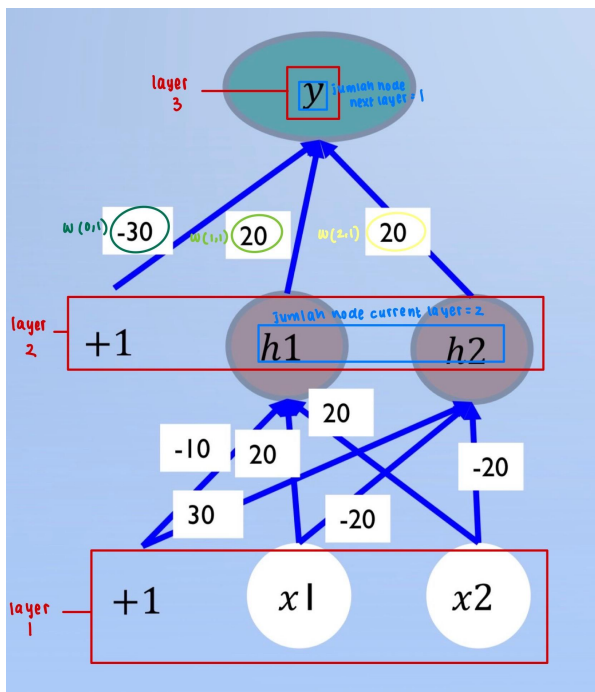
3

Selanjutnya butuh jumlah node current layer, jumlah node next layer, dan fungsi aktivasi. Didapatkan current layer ada 2 node, next layer ada 2 node, dan fungsi aktivasi adalah sigmoid (2), maka update .txt menjadi:

3  
2 2 2

Selanjutnya kita akan menginputkan bobot yang terdapat pada setiap edge diantara layer 1 dan 2 mengikuti urutan sebagai berikut  $W(0,1)$ ,  $W(0,2)$ ;  $W(1,1)$ ,  $W(1,2)$ ;  $W(2,1)$ ,  $W(2,2)$ . Input dilakukan perbaris. Maka .txt akan menjadi:

3  
2 2 2  
-10 30  
20 -20  
20 -20



Selanjutnya adalah input untuk layer 2. Kita memerlukan jumlah current layer, jumlah next layer, dan activation function yang digunakan. Jumlah current layer adalah 2, jumlah next layer adalah 1, dan activation function yang digunakan adalah sigmoid(2). Maka .txt akan menjadi:

```
3
2 2 2
-10 30
20 -20
20 -20
2 1 2
```

Selanjutnya kita akan menginputkan bobot yang terdapat pada setiap edge diantara layer 2 dan 3 mengikuti urutan sebagai berikut  $W(0,1)$ ;  $W(1,1)$ ;  $W(2,1)$ . Input dilakukan perbaris. Maka .txt akan menjadi:

```
3
2 2 2
-10 30
20 -20
20 -20
2 1 2
-30
20
20
```

.txt file selesai dibangun.

## Folder src

Folder ini berisi implementasi dari program FFNN forward propagation yaitu pada file main.ipynb, berikut penjelasan file main.ipynb kami:

### - Library and dependencies

Terdapat dua library yang digunakan untuk mengimplementasikan program ini, yaitu numpy dan pyvis. Numpy merupakan library yang berguna untuk



melakukan operasi-operasi pada array agar menjadi lebih mudah. Pyvis merupakan library visualisasi pada python. Modul Network pada Pyvis dapat digunakan untuk mengimplementasikan graph neural network.

- Class

- *Connected Layer*

*Connected layer* merupakan implementasi berupa setiap input neuron yang terhubung dengan setiap output neuron. *ConnectedLayer* memiliki atribut *input\_size*, *output\_size*, *weights*, *bias*, dan *input*. Class ini memiliki method *forward* untuk menghitung nilai pada layer berikutnya, sebelum diapply fungsi aktivasi. Untuk forward propagation, kalkulasi output neuron yang digunakan adalah sebagai berikut:

$$y_j = b_j + \sum_i x_i w_{ij}$$

- *Activation layer*

Layer ini berfungsi untuk mengimplementasikan *activation function* yang tersedia untuk output yang telah dihasilkan setelah dikalkulasikan *connected layer*. *ActivationLayer* memiliki atribut *activation* dan *input*. Class ini memiliki method *forward* untuk menghitung nilai pada layer berikutnya berdasarkan nilai yang diberikan oleh *ConnectedLayer*.

- *Neural Network*

Bagian ini merupakan proses pembangunan network pada FFNN dengan memanfaatkan seluruh fungsi yang ada untuk membangun layer-layer network. *NeuralNetwork* memiliki atribut array of layers yang dapat berupa *Connected Layer* ataupun *Activation Layer*, sehingga merepresentasikan network secara keseluruhan. Class ini memiliki method *predict* untuk melakukan prediksi terhadap input

- Activation function

Bagian ini berisi fungsi aktivasi yang digunakan untuk menghasilkan output dari input yang diberikan pada suatu node. Fungsi aktivasi yang dapat diimplementasikan pada program kami diantaranya adalah linear, ReLU, sigmoid, dan softmax.

- Linear

Fungsi aktivasi ini merupakan fungsi yang dikenal sebagai *identity function* yang berarti fungsi ini tidak melakukan apa-apa, langsung mengeluarkan hasil perhitungan sebagai output.

- ReLU

*Rectified Linear Unit* merupakan fungsi aktivasi yang tidak mengaktivasi semua neuron yang ada. Neuron akan di deaktivasi jika output transformasi linearnya kurang dari 0 sehingga output dari fungsi ini selalu  $\geq 0$ .

- Sigmoid

Fungsi aktivasi ini merupakan fungsi yang mengeluarkan input antara 0 sampai 1. Semakin positif inputnya, outputnya akan makin mendekati 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Softmax

Fungsi aktivasi ini merupakan fungsi yang mengeluarkan input antara 0 sampai 1 dan dapat diartikan sebagai probabilitas.

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- Loss function

Bagian ini berisi fungsi yang digunakan untuk menghitung kualitas model yang telah dibangun dengan membandingkan antara  $y_{\text{pred}}$  dan  $y_{\text{true}}$ .  $y_{\text{pred}}$  merupakan  $Y$  hasil prediksi sedangkan  $y_{\text{true}}$  merupakan  $Y$  yang bersesuaian dengan data yang dimiliki. Metode yang digunakan adalah dengan menggunakan MSE dan SSE. MSE (*Mean Squared Error*) merupakan penilaian kualitas estimator dengan menggunakan mean atau rata-rata dari kuadrat perbedaan data aktual dengan data estimasi. SSE (*Sum of Square Error*) merupakan penjumlahan seluruh jarak masing-masing data aktual dengan data estimasi. Semakin besar nilai MSE atau SSE nya, maka semakin buruk kualitas dari estimator.

- Model Load Section

Bagian ini berguna untuk melakukan parsing terhadap file input model ataupun prediksi. File .txt akan diolah menjadi sebuah beberapa array yang siap digunakan untuk prediksi dan visualisasi. Bagian ini digunakan untuk membaca model dan memasukkannya dari file .txt yang telah dibangun. Model di load dari ../test/model/ sedangkan input di load dari ../test/input. Pengguna akan memasukkan nama model dan inputan sesuai .txt yang telah ia bangun di folder yang bersesuaian.

- Loading Weights

Bagian ini digunakan untuk meload weight dari model yang telah dibaca dari file .txt

- Predict Output

Bagian ini merupakan bagian untuk memprediksi output dari masukan yang telah diberikan. Implementasinya memanfaatkan fungsi-fungsi yang telah dijelaskan sebelumnya. Data input yang sudah diproses akan diprediksi. Pertama-tama akan diinisiasi sebuah instance `NeuralNetwork`. Kemudian, instance ini akan ditambah layernya satu per satu berdasarkan data input yang sudah diolah. Jika semua layer sudah ditambahkan, akan dipanggil method `predict` untuk mendapatkan nilai dari setiap node neuron saat prediksi.

- Neural Network Visualization

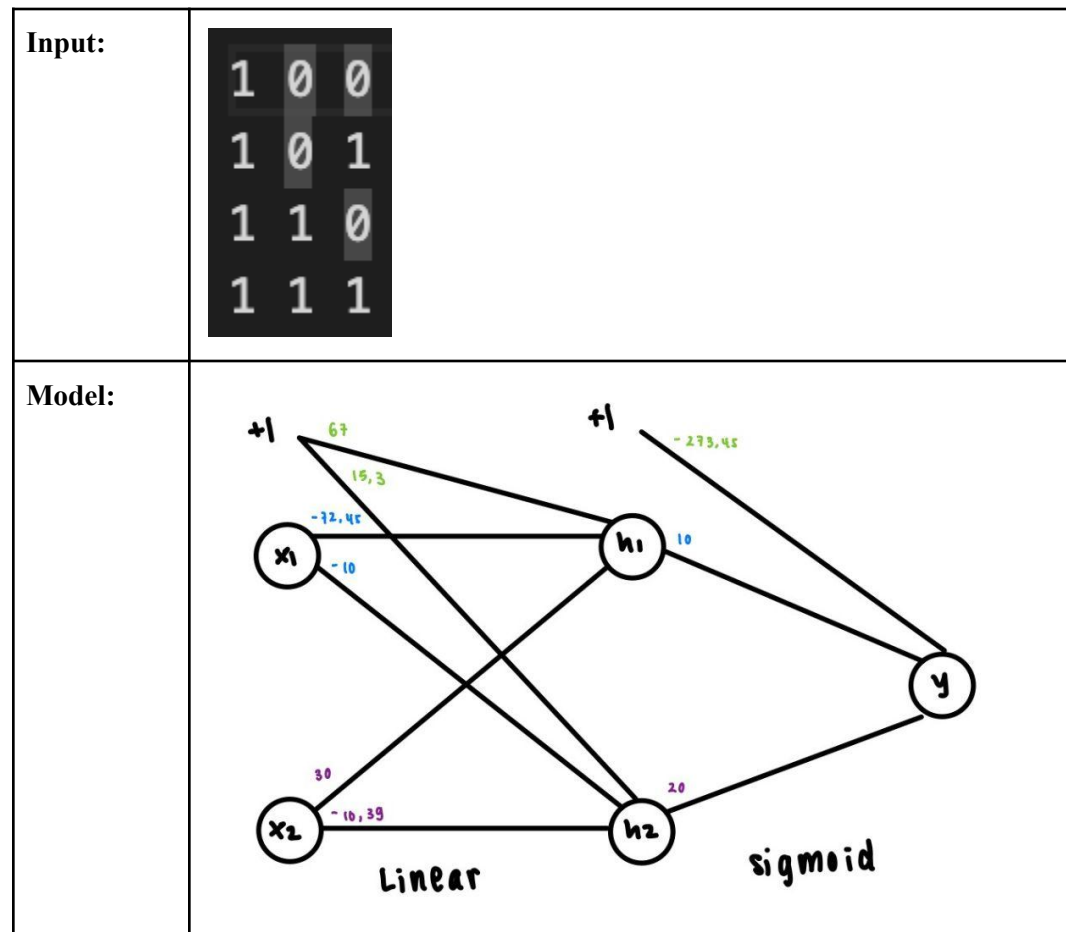
Bagian ini digunakan untuk membangun visualisasi network yang terdiri dari nodes dan edges yang bersesuaian dengan network yang telah dibangun. Visualisasi diimplementasikan menggunakan library `Pyvis`. Terdapat dua fungsi utama yang digunakan yaitu `add_node` untuk membuat node dan `add_edge` untuk membuat edge. Parameter dari fungsi `add_node` terdiri dari `nodes`, `title`, `value`, `x`, `y`, `label`, dan `color`. Parameter dari fungsi `add_edge` terdiri dari `node1`, `node2`, `title`, dan `color`. Sebelum kedua fungsi ini dipanggil, data yang dibutuhkan untuk parameter akan disiapkan terlebih dahulu. `Value` merupakan judul yang didapatkan dari posisi node neuron tersebut berdasarkan input model. `Title` merupakan nilai yang akan muncul jika dihover. Pada kasus ini, `title` merupakan weight sebuah edge ataupun nilai sebuah node neuron. Nilai dari `title` didapatkan dengan mengolah hasil dari `predict` yang sudah didapat sebelumnya.

## II. Hasil Pengujian

Data yang digunakan data XOR:

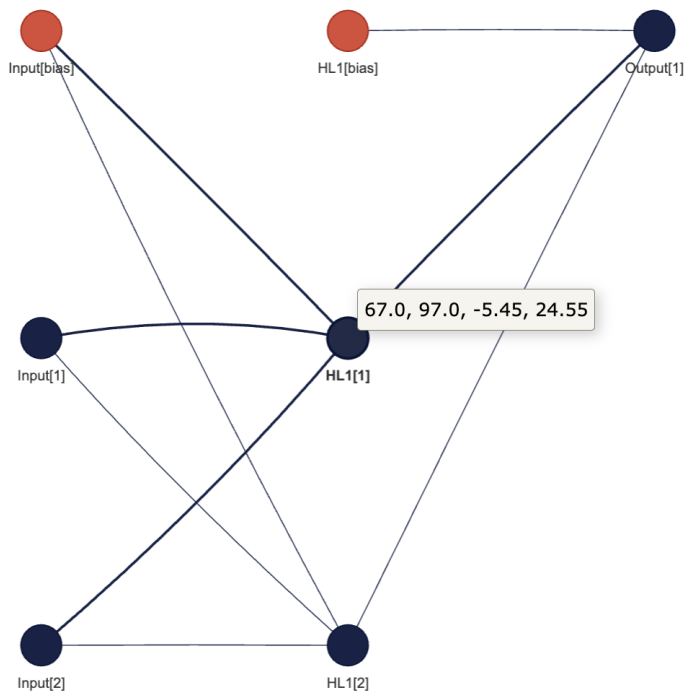
Input X1	Input X2	Output
0	0	0
0	1	1
1	0	1
1	1	0

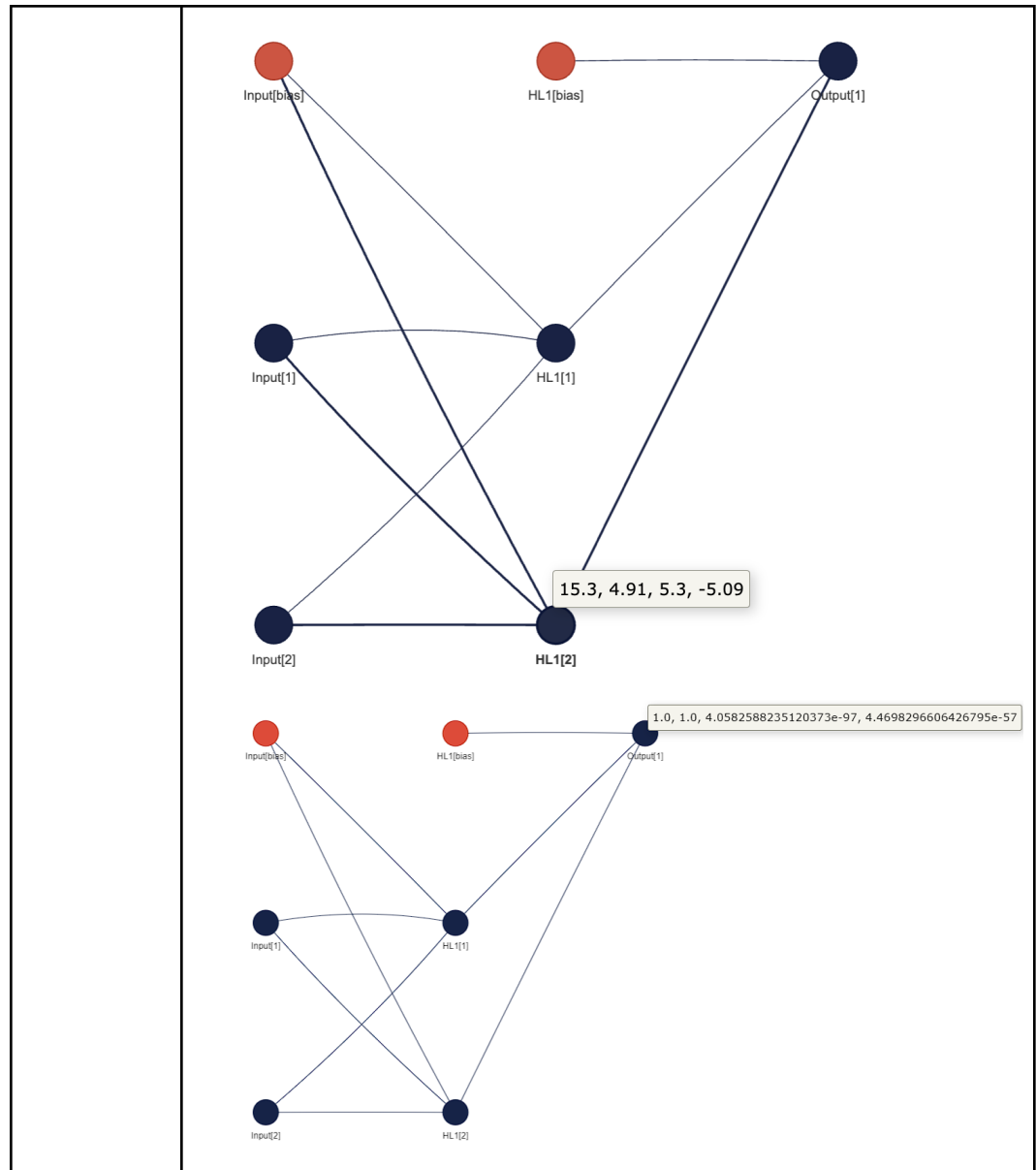
### 1. Pengujian test case buatan sendiri Linear - Sigmoid (xor\_3.txt)



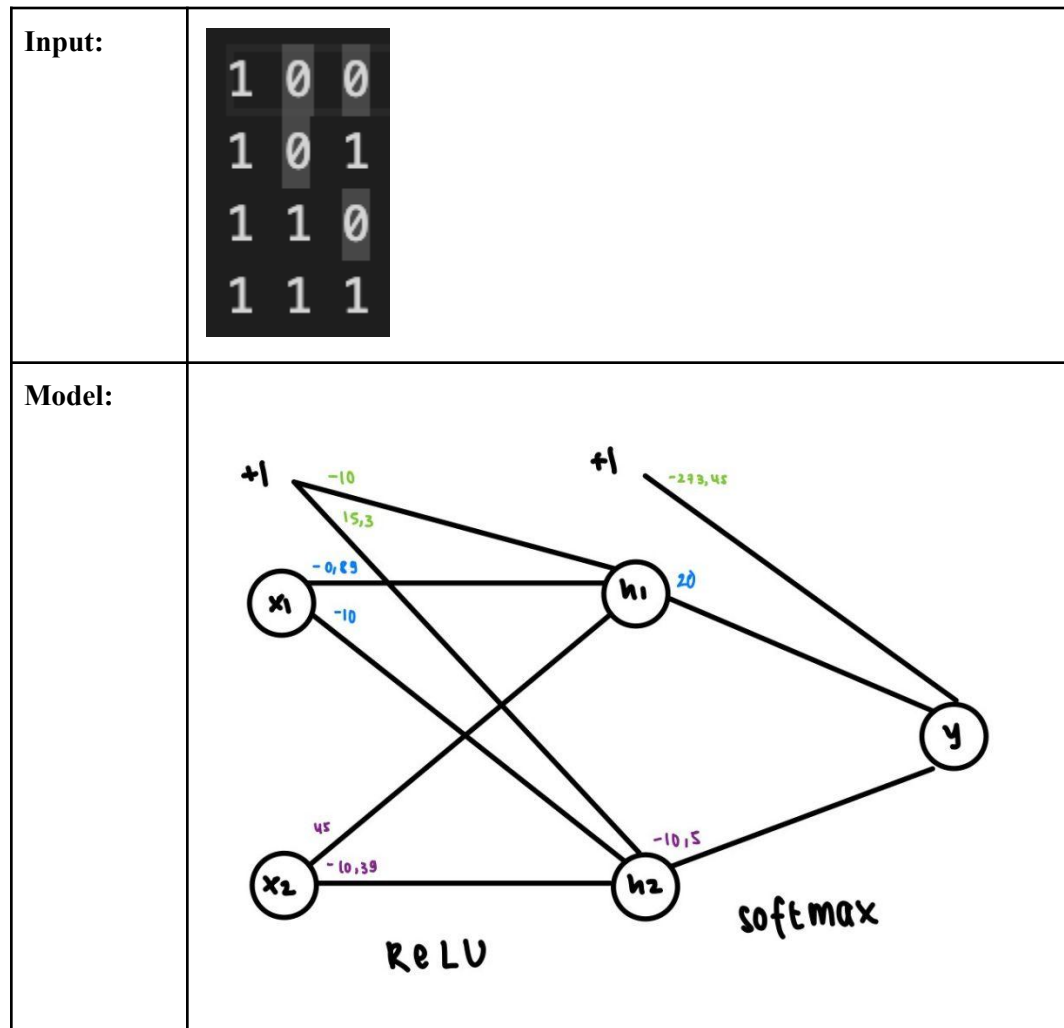
$\beta$   
2 2 0  
67 15.3  
-72.45 -10  
30 -10.39  
2 1 2  
-273.45  
10  
20

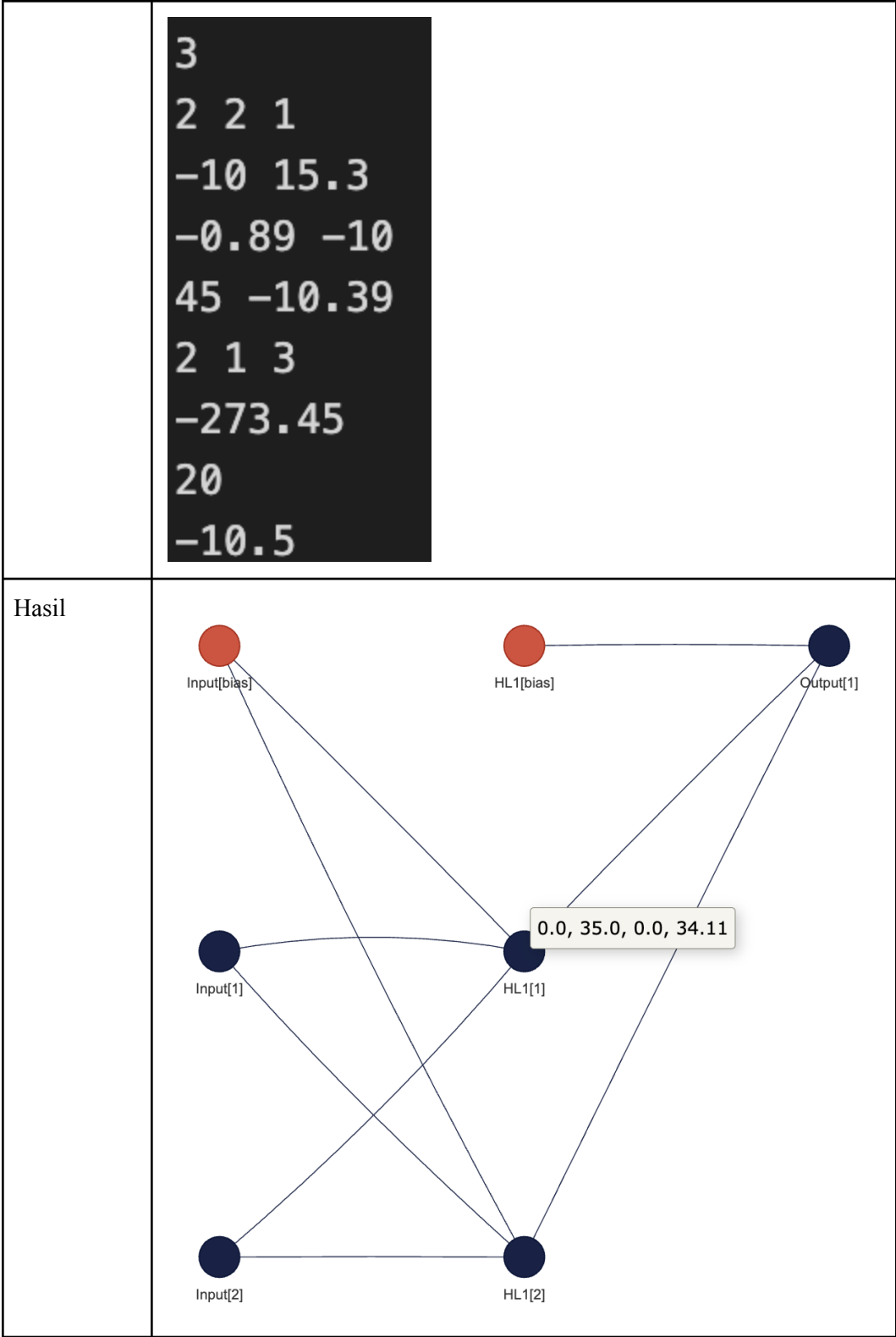
Hasil



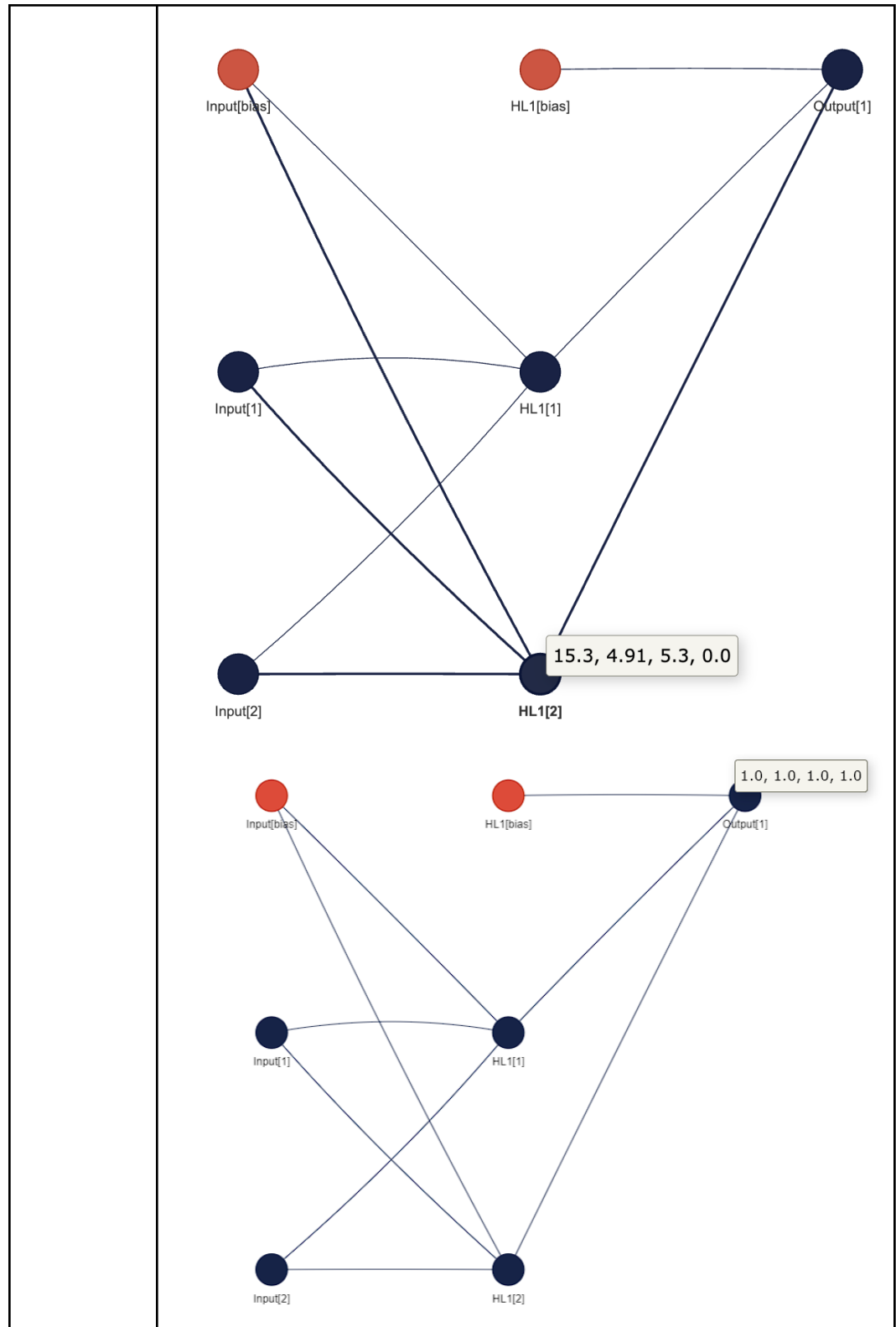


2. Pengujian test case buatan sendiri ReLU - Softmax (xor\_2.txt)

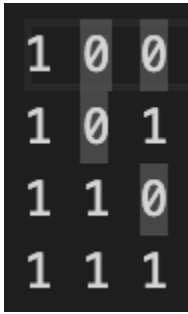
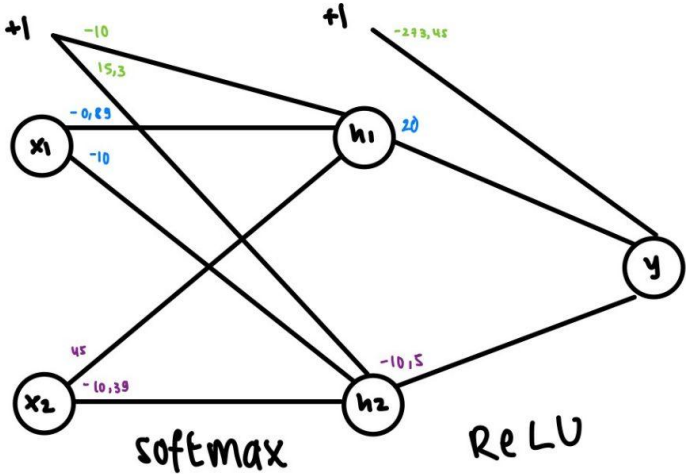




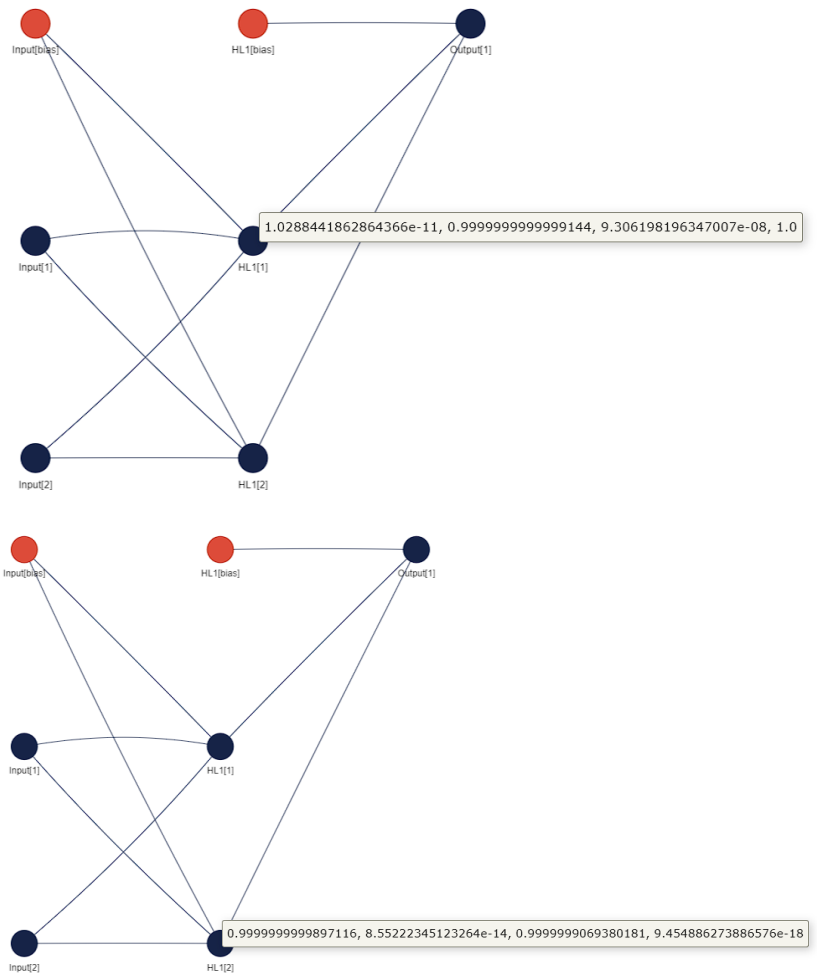


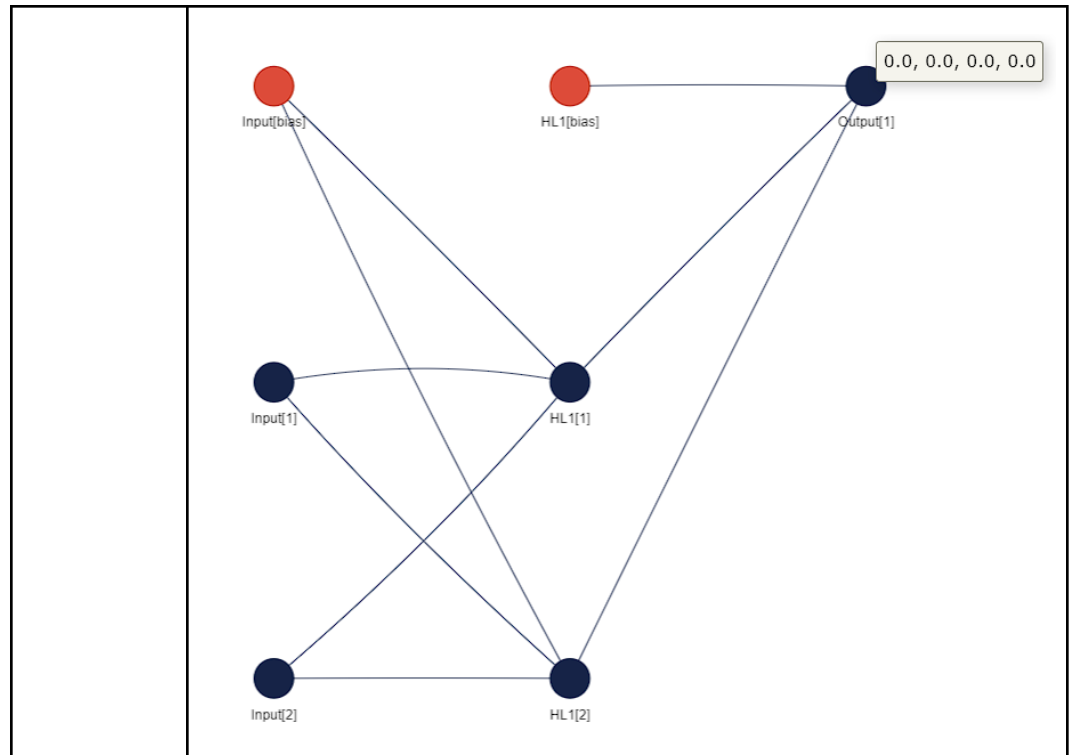


3. Pengujian test case buatan sendiri Softmax - ReLU (xor\_4.txt)

<b>Input:</b>	
<b>Model:</b>	 <pre> 1 3 2 2 2 3 3 -10 15.3 4 -0.89 -10 5 45 -10.39 6 2 1 1 7 -273.45 8 20 9 -10.5 </pre>

## Hasil

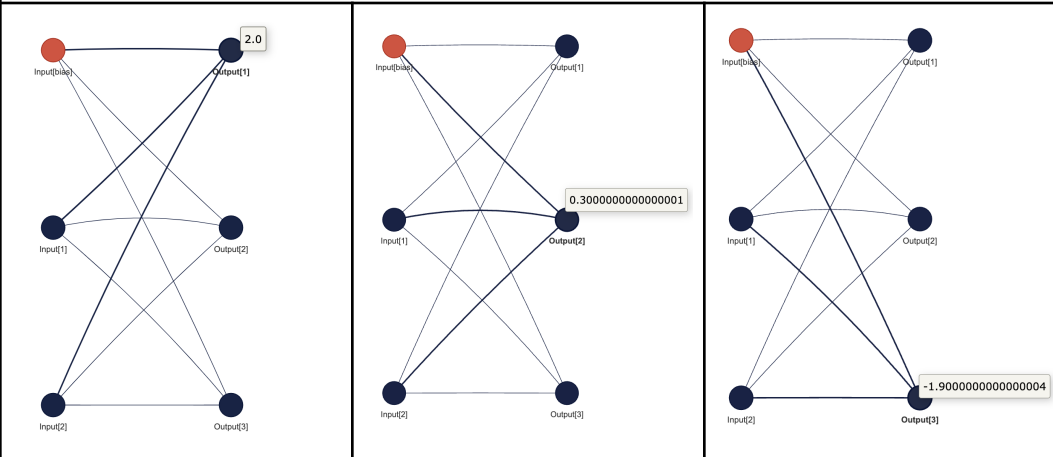




#### 4. Pengujian test case asisten: linear\_tc.txt

Expected output: [ 2.0, 0.3, -1.9]

Output:



Expected max\_sse : 0.000001

Output SSE:

```

parsed_output = [[float(j) for j in i.split()] for i in output_data]
out_pred = out.flatten()
out_true = np.array(parsed_output[:-1]).flatten()
max_sse = parsed_output[-1][0]

print('SSE ---->', sse(out_true, out_pred))
print("sse <= max_sse ---->", sse(out_true, out_pred) <= max_sse)
✓ 0.0s

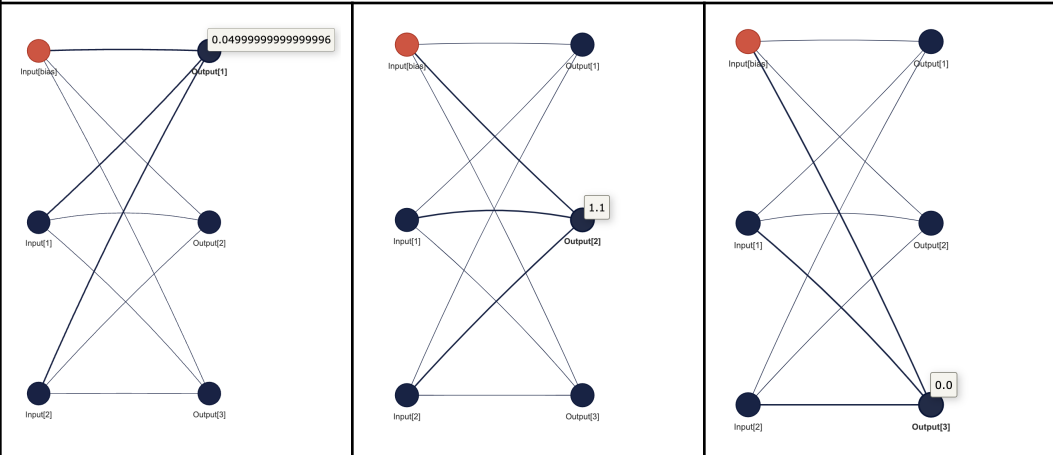
SSE ----> 2.0954117794933126e-31
sse <= max_sse ----> True

```

## 5. Pengujian test case asisten: relu\_tc.txt

Expected output: [0.05, 1.1, 0.0]

Output:



Expected max\_sse : 0.000001

Output SSE:

```

parsed_output = [[float(j) for j in i.split()] for i in output_data]
out_pred = out.flatten()
out_true = np.array(parsed_output[:-1]).flatten()
max_sse = parsed_output[-1][0]

print('SSE ---->', sse(out_true, out_pred))
print("sse <= max_sse ---->", sse(out_true, out_pred) <= max_sse)

✓ 0.0s

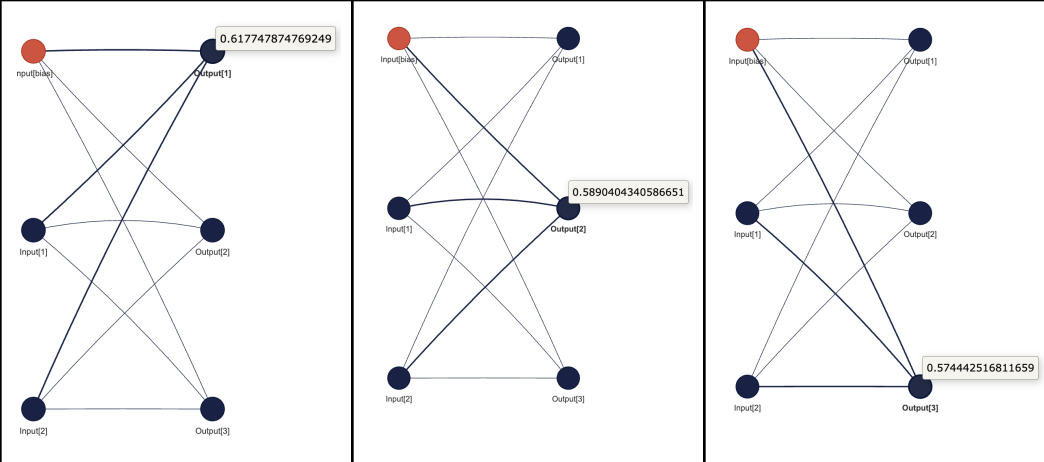
SSE ----> 1.7333369499485123e-33
sse <= max_sse ----> True

```

## 6. Pengujian test case asisten: sigmoid\_tc.txt

Expected output: [0.617747], [0.589040], [0.574442]

Output:



Expected max\_sse : 0.000001

Output SSE:

```

parsed_output = [[float(j) for j in i.split()] for i in output_data]
out_pred = out.flatten()
out_true = np.array(parsed_output[:-1]).flatten()
max_sse = parsed_output[-1][0]

print('SSE --->', sse(out_true, out_pred))
print("sse <= max_sse --->", sse(out_true, out_pred) <= max_sse)
✓ 0.0s

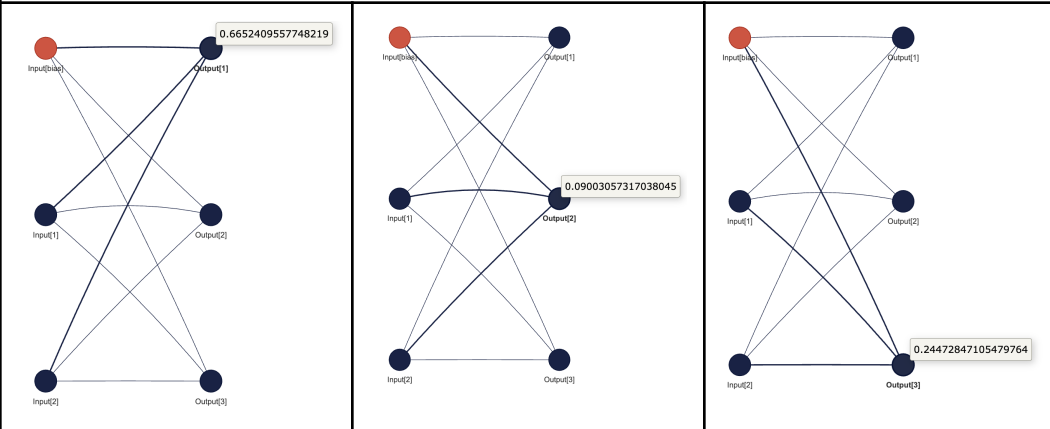
SSE ---> 1.2207224545374987e-12
sse <= max_sse ---> True

```

## 7. Pengujian test case asisten: softmax\_tc.txt

Expected output: [0.665241], [0.090031], [0.244728]

Output:



Expected max\_sse : 0.000001

Output SSE:

```
parsed_output = [[float(j) for j in i.split()] for i in output_data]
out_pred = out.flatten()
out_true = np.array(parsed_output[:-1]).flatten()
max_sse = parsed_output[-1][0]

print('SSE ---->', sse(out_true, out_pred))
print("sse <= max_sse ---->", sse(out_true, out_pred) <= max_sse)

✓ 0.0s

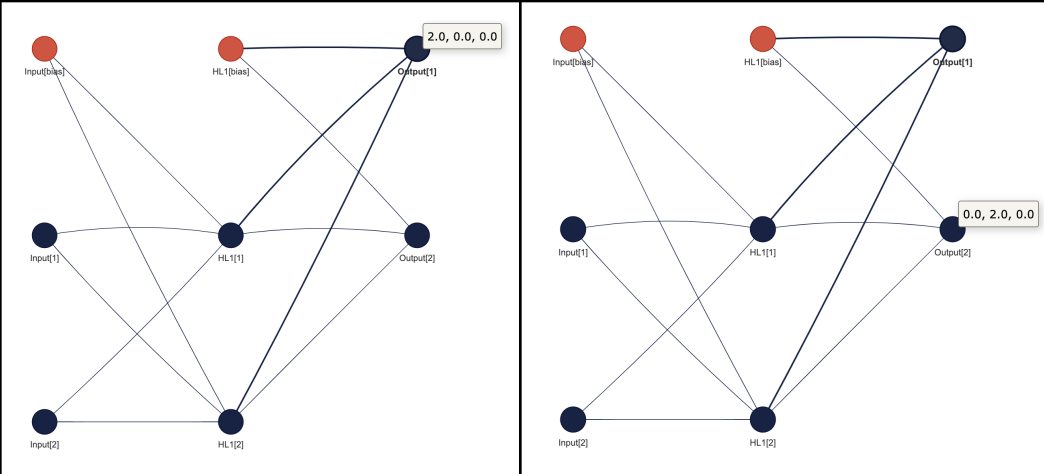
SSE ----> 4.0603201288236806e-13
sse <= max_sse ----> True
```



## 8. Pengujian test case asisten: multilayer\_tc.txt

Expected output: [2.0, 0.0],  
[0.0, 2.0],  
[0.0, 0.0]

Output:



Expected max\_sse : 0.000001

Output SSE:

```

parsed_output = [[float(j) for j in i.split()] for i in output_data]
out_pred = out.flatten()
out_true = np.array(parsed_output[:-1]).flatten()
max_sse = parsed_output[-1][0]

print('SSE ---->', sse(out_true, out_pred))
print("sse <= max_sse ---->", sse(out_true, out_pred) <= max_sse)
✓ 0.0s

SSE ----> 0.0
sse <= max_sse ----> True

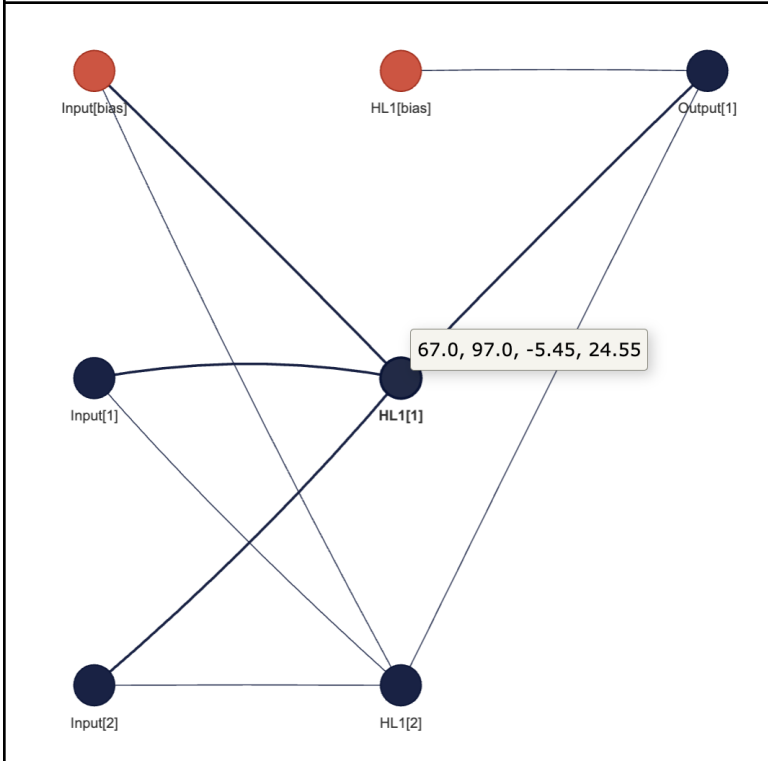
```

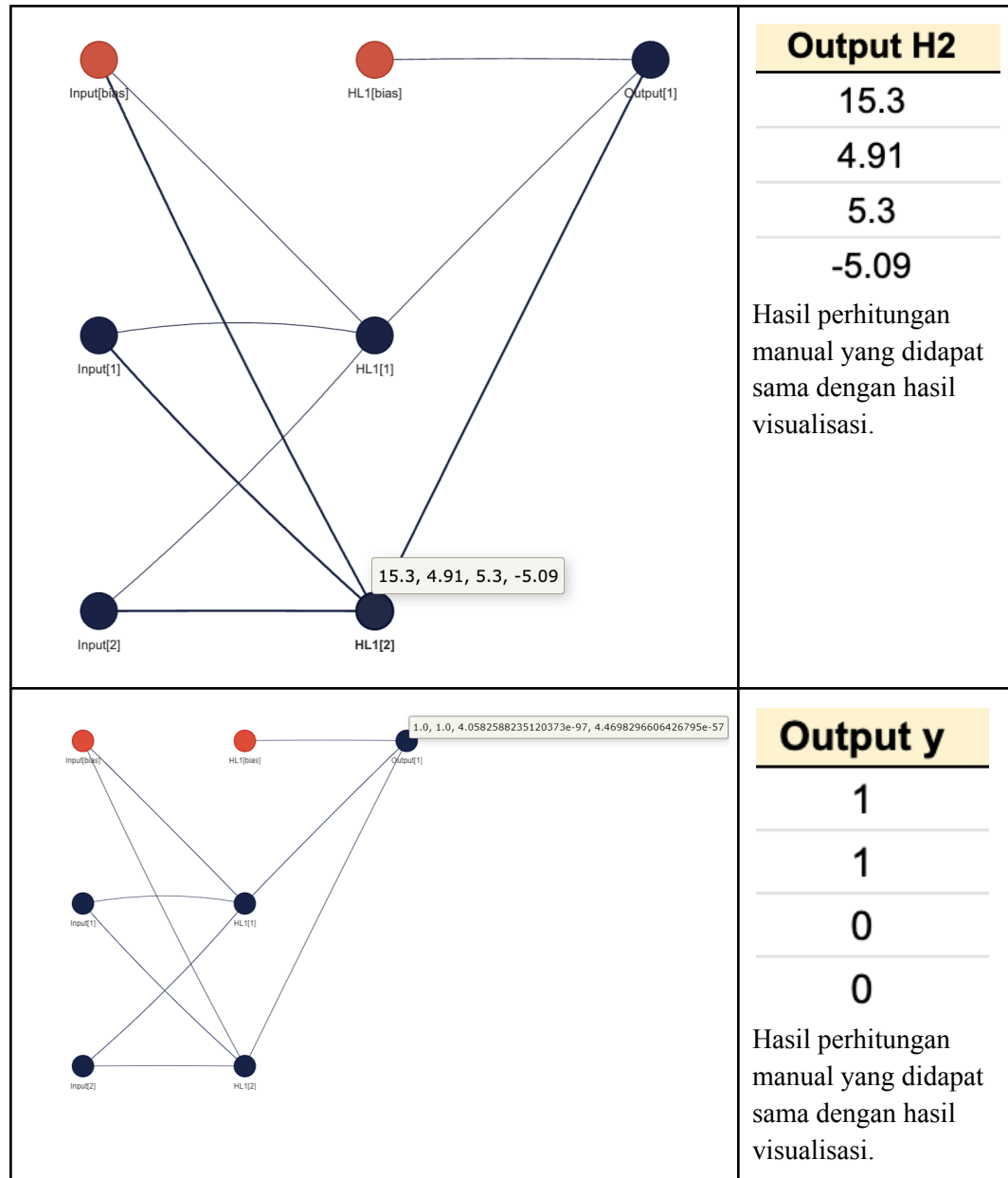
### III. Perbandingan Hasil Perhitungan Manual

Link perhitungan manual dapat diakses pada: [Perhitungan Manual FFNN](#)

Perhitungan manual ini terdiri dari 4 sheets: 2 sheets perhitungan *test case* buatan sendiri dan 2 *test case* lain untuk mencocokkan kebenaran formula pada sheets dengan PPT pembelajaran bersumber dari <https://edunex.itb.ac.id/>.

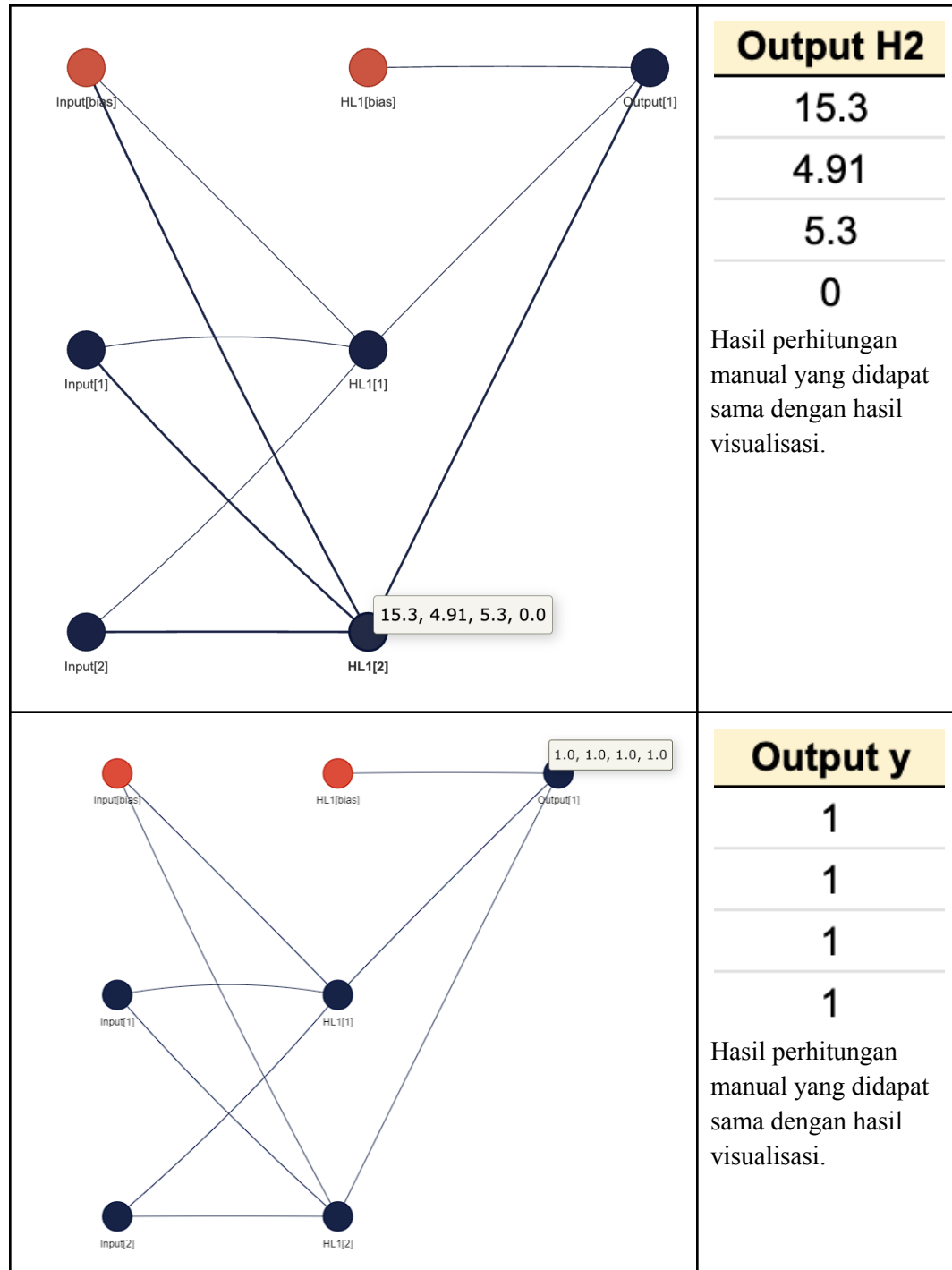
#### 1. Pengujian *test case* buatan sendiri: Linear - Sigmoid (xor\_3.txt)

Hasil Visualisasi	Hasil Perhitungan Manual
	<div>Output H1</div> <div>67</div> <div>97</div> <div>-5.45</div> <div>24.55</div> <p>Hasil perhitungan manual yang didapat sama dengan hasil visualisasi.</p>

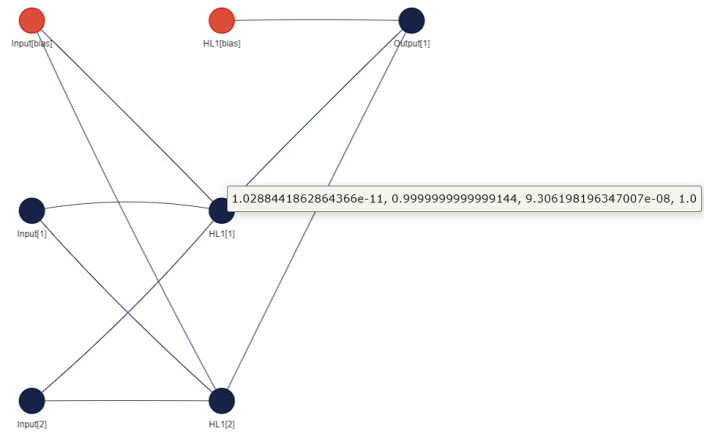
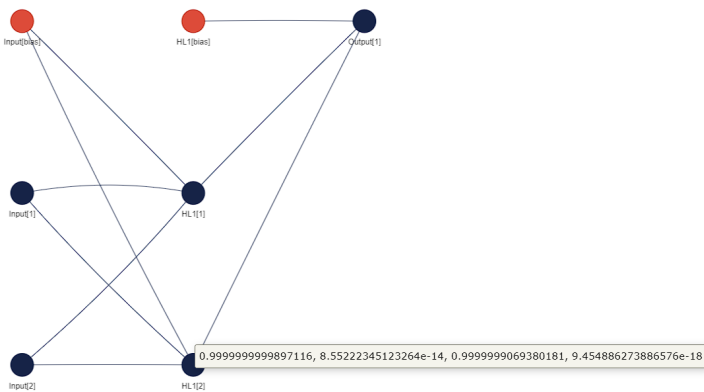


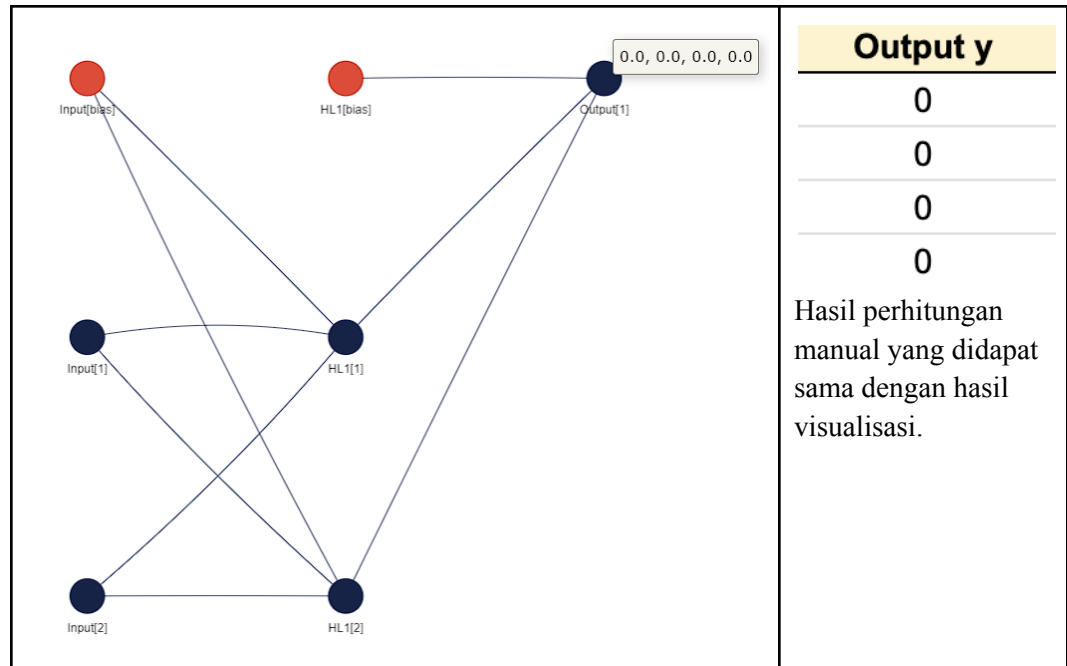
2. Pengujian *test case* buatan sendiri: ReLU - Softmax (xor\_.txt)

Hasil Visualisasi	Hasil Perhitungan Manual
	<div>Output H1</div> <div>0</div> <div>35</div> <div>0</div> <div>34.11</div> <div>Hasil perhitungan manual yang didapat sama dengan hasil visualisasi.</div>



### 3. Pengujian *test case* buatan sendiri: Softmax - ReLU (xor\_4.txt)

Hasil Visualisasi	Hasil Perhitungan Manual					
	<table><tr><th>Output H1</th></tr><tr><td>0</td></tr><tr><td>1</td></tr><tr><td>0.00000009306198196</td></tr><tr><td>1</td></tr></table> <p>Hasil perhitungan manual yang didapat sama dengan hasil visualisasi.</p>	Output H1	0	1	0.00000009306198196	1
Output H1						
0						
1						
0.00000009306198196						
1						
	<table><tr><th>Output H2</th></tr><tr><td>1</td></tr><tr><td>0</td></tr><tr><td>0.99999999069</td></tr><tr><td>0</td></tr></table> <p>Hasil perhitungan manual yang didapat sama dengan hasil visualisasi.</p>	Output H2	1	0	0.99999999069	0
Output H2						
1						
0						
0.99999999069						
0						



#### IV. Pembagian Tugas

<b>NIM</b>	<b>Nama</b>	<b>Tugas</b>
13520075	Samuel Christopher Swandi	Class, Predict Output, Laporan
13520078	Grace Claudia	Helper Function, Predict Output, Laporan
13520085	Ubaidillah Ariq Prathama	Predict Output, Visualization, Laporan
13520109	Patrick Amadeus Irawan	Load Model, Visualization, Laporan