LAPORAN TUGAS BESAR BAGIAN A
IF3270 PEMBELAJARAN MESIN

ARTIFICIAL NEURAL NETWORK

Diajukan sebagai tugas besar Mata Kuliah IF3270 Pembelajaran Mesin

Semester II Tahun Akademik 2023/2024

Anggota Kelompok:

Jason Rivalino	13521008
Muhammad Rifko Favian	13521075
Moch. Sofyan Firdaus	13521083
Fazel Ginanda	13521098

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023/2024

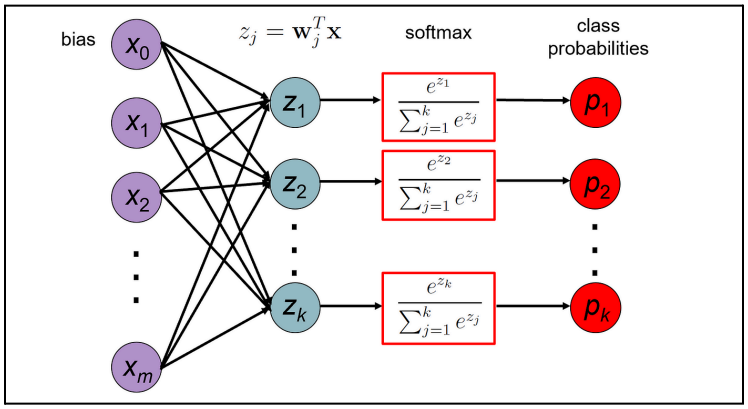
BAB I
DESKRIPSI TUGAS

Tugas besar kali ini adalah untuk membuat rancangan program dalam bahasa Python untuk membuat jaringan saraf tiruan pada bagian *feed forward neural network* (FFNN). Pada pengerjaan kali ini, terdapat beberapa spesifikasi dan batasan yang perlu dipenuhi antara lain sebagai berikut:

Spesifikasi

1. Program berfungsi sebagai *neural network* yang dapat melakukan *feed forward* dari *input* yang diberikan.
2. Program dapat menerima masukan dari file JSON.
3. Implementasi pada fungsi aktivasi berikut:
    1) Linear
    2) ReLU
    3) Sigmoid
    4) Softmax

$$
\begin{aligned}
&1.\ f(\text{net}) = net \\
&2.\ f(\text{net}) = max\{0, net\} \\
&3.\ f(\text{net}) = 1/(1+e^{-net}) \\
&4.\ f(\text{net}_i) = \frac{e^{net_i}}{\sum e^{net_i}}
\end{aligned}
$$



Gambar 1. Fungsi softmax

4. Program dapat menyimpan bobot (*weights*) dan struktur model.
5. Implementasi forward propagation untuk FFNN dengan kemampuan:
    a. Menampilkan model berupa struktur jaringan dan bobotnya, formatnya bebas.
    b. Memberikan output untuk input 1 instance.
    c. Memberikan output untuk input berupa batch.
6. Direkomendasikan membuat layer dan fungsi aktivasi secara modular.

Batasan

1. Satu layer memiliki fungsi aktivasi yang sama; layer yang berbeda dapat memiliki fungsi aktivasi yang berbeda.

BAB II
PENJELASAN IMPLEMENTASI

Untuk implementasi dari program yang dibuat, program ini sendiri dibuat dengan memanfaatkan tiga buah library yaitu numpy untuk proses perhitungan pada fungsi aktivasi dalam layer, lalu json untuk proses import format file JSON, dan terakhir yaitu Graphviz untuk membantu proses visualisasi dari graf untuk keseluruhan model struktur jaringan FFNN.

Adapun untuk struktur program yang ada secara keseluruhan dibagi menjadi dua kelas yaitu kelas untuk menjalankan proses pencarian nilai output FFNN dan juga kelas Layer yang berfungsi untuk memproses hasil pada setiap layer berdasarkan fungsi aktivasi yang terdapat pada input file.

Proses yang terjadi yaitu dimulai dari pembacaan file test dengan format JSON pada program. Dari hasil pembacaan file tersebut kemudian untuk berbagai macam model akan didefinisikan nilai-nilai yang diperlukan untuk menjalankan FFNN antara lain:

1. Input_size: jumlah ukuran data input pada program
2. Input_array: nilai-nilai yang diinput, berupa satu instance atau batch
3. Layers: informasi untuk setiap layers (ada jumlah neuron dan juga jenis fungsi aktivasi)

Setelah berbagai nilai ini didefinisikan, kemudian menjalankan fungsi FFNN dengan proses sebagai berikut:

1. Inisiasi fungsi FFNN dengan memasukkan beberapa variabel yang telah didefinisikan sebelumnya
2. Melakukan operasi perhitungan dengan untuk setiap layers yang ada berdasarkan jumlah neuron dan fungsi aktivasi dan juga mendefinisikan bias dan bobot (*weights*) berdasarkan input pada file (bias terletak pada weight baris pertama dan bobot untuk baris sisanya)
3. Penambahan layer beserta informasi terkait weight dan berdasarkan model yang telah didefinisikan dengan looping hingga total jumlah layers
4. Menjalankan fungsi forward untuk proses forward propagation dan melakukan perpindahan dari layer ke layer hingga sampai pada output
5. Mendapatkan hasil pada output

6. Memeriksa kesesuaian output yang dihasilkan dengan nilai expected output pada test case yang diuji

Setelah nilai akhir dari FFNN didapatkan, proses selanjutnya adalah untuk memvisualisasikan penggambaran dari graf untuk proses forward propagation. Untuk visualisasinya, disini akan memanfaatkan library Graphviz kelas Diagraph untuk menampilkan graf keseluruhannya. Prosesnya dimulai dari mendefinisikan node untuk input, lalu node untuk berbagai macam layer (jika file input memiliki data yang bersifat multilayer), penghubungan dari satu input ke berbagai macam layer hingga akhirnya mencapai node untuk output program.

BAB III
HASIL PENGUJIAN DAN PERBANDINGAN

A. Hasil Pengujian

1. Pengujian dengan file relu.json (dari spesifikasi tugas besar)

   Hasil harapan:

```
    "expect": {
        "output": [[0.05, 1.1, 0.0]],
        "max_sse": 0.000001
    }
```

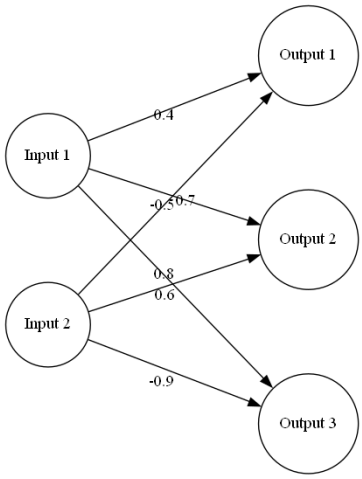
   Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: reluspek.json

Output:
[[0.05 1.1  0.  ]]

Max SSE: 0.00

Test Result:
Passed
```



2. Pengujian dengan file relu.json (dari test case)

   Hasil harapan:

```
    "expect": {
      "output": [[0.31, 0, 0.375]],
      "max_sse": 0.000001
    }
```

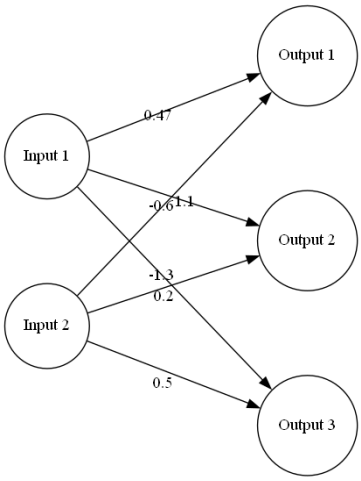
Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: relutc.json

Output:
[[0.31  0.    0.375]]

Max SSE: 0.00

Test Result:
Passed
```



3. Pengujian dengan file multilayer.json

   Hasil harapan:

```
"expect": {
  "output": [[0.4846748]],
  "max_sse": 0.000001
}
```

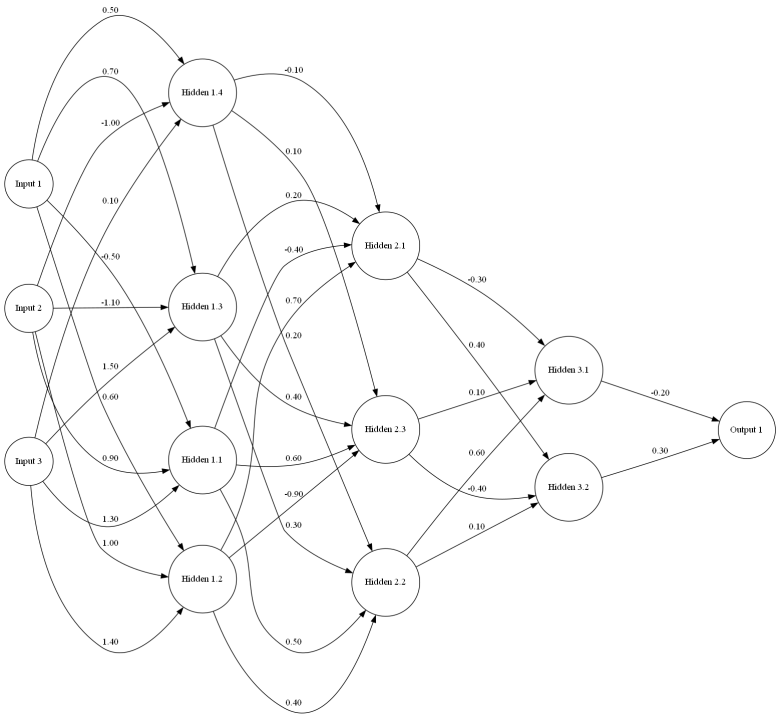
Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: multilayer.json

Output:
[[0.4846748]]

Max SSE: 0.00

Test Result:
Passed
```



4. Pengujian dengan file softmax.json

   Hasil harapan:

```
"expect": {
    "output": [
      [0.76439061, 0.21168068, 0.02392871]
    ],
    "max_sse": 0.00001
  }
```

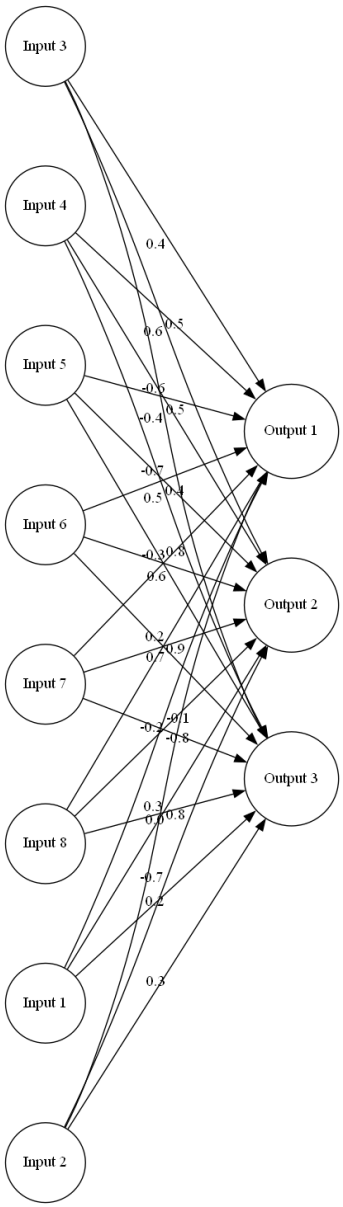
   Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: softmax.json

Output:
[[0.76439061 0.21168068 0.02392871]]

Max SSE: 0.00

Test Result:
Passed
```



5. Pengujian dengan file linear.json

   Hasil harapan:

```
"expect": {
        "output": [[-11], [-8], [-5], [-2], [1], [4], [7], [10],
[13], [16]],
      "max_sse": 0.000000000001
    }
```

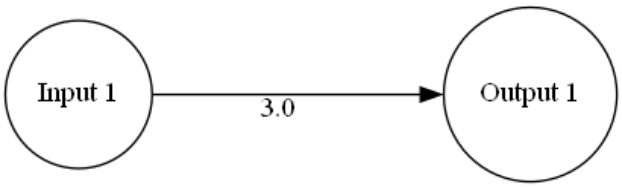
   Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: linear.json

Output:
[[-11.]
 [ -8.]
 [ -5.]
 [ -2.]
 [  1.]
 [  4.]
 [  7.]
 [ 10.]
 [ 13.]
 [ 16.]]

Max SSE: 0.00

Test Result:
Passed
```



6. Pengujian dengan file sigmoid.json

   Hasil harapan:

```
    "expect": {
      "output": [
        [0.41197346, 0.8314294, 0.53018536, 0.31607396],
        [0.78266141, 0.80843631, 0.55350518, 0.64278501],
        [0.58987524, 0.82160954, 0.75436518, 0.34919895],
        [0.6722004,  0.81660439, 0.59020258, 0.50870988],
        [0.47322841, 0.82808466, 0.69105452, 0.29358323]
      ],
      "max_sse": 0.000001
    }
```

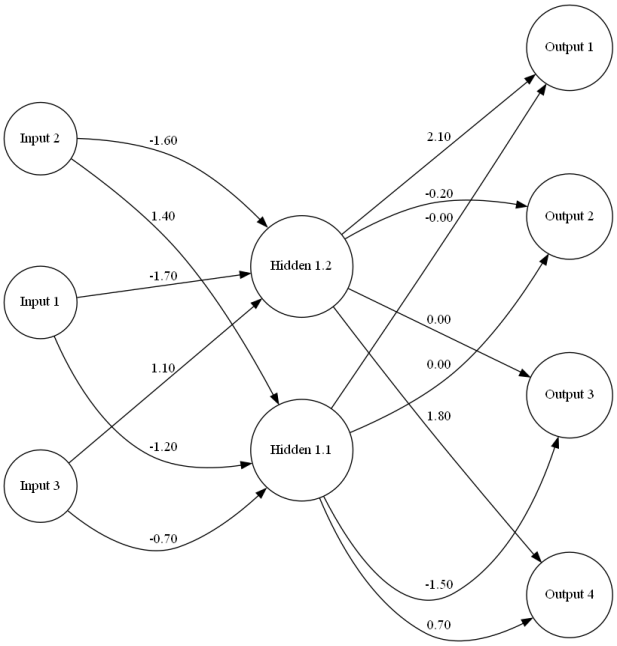
Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: sigmoid.json

Output:
[[0.41197346 0.8314294  0.53018536 0.31607396]
 [0.78266141 0.80843631 0.55350518 0.64278501]
 [0.58987524 0.82160954 0.75436518 0.34919895]
 [0.6722004  0.81660439 0.59020258 0.50870988]
 [0.47322841 0.82808466 0.69105452 0.29358323]]

Max SSE: 0.00

Test Result:
Passed
```



7. Pengujian dengan file multilayer_softmax.json

   Hasil harapan:

```
"expect": {
    "output": [[0.7042294, 0.2957706]],
    "max_sse": 0.000001
  }
```

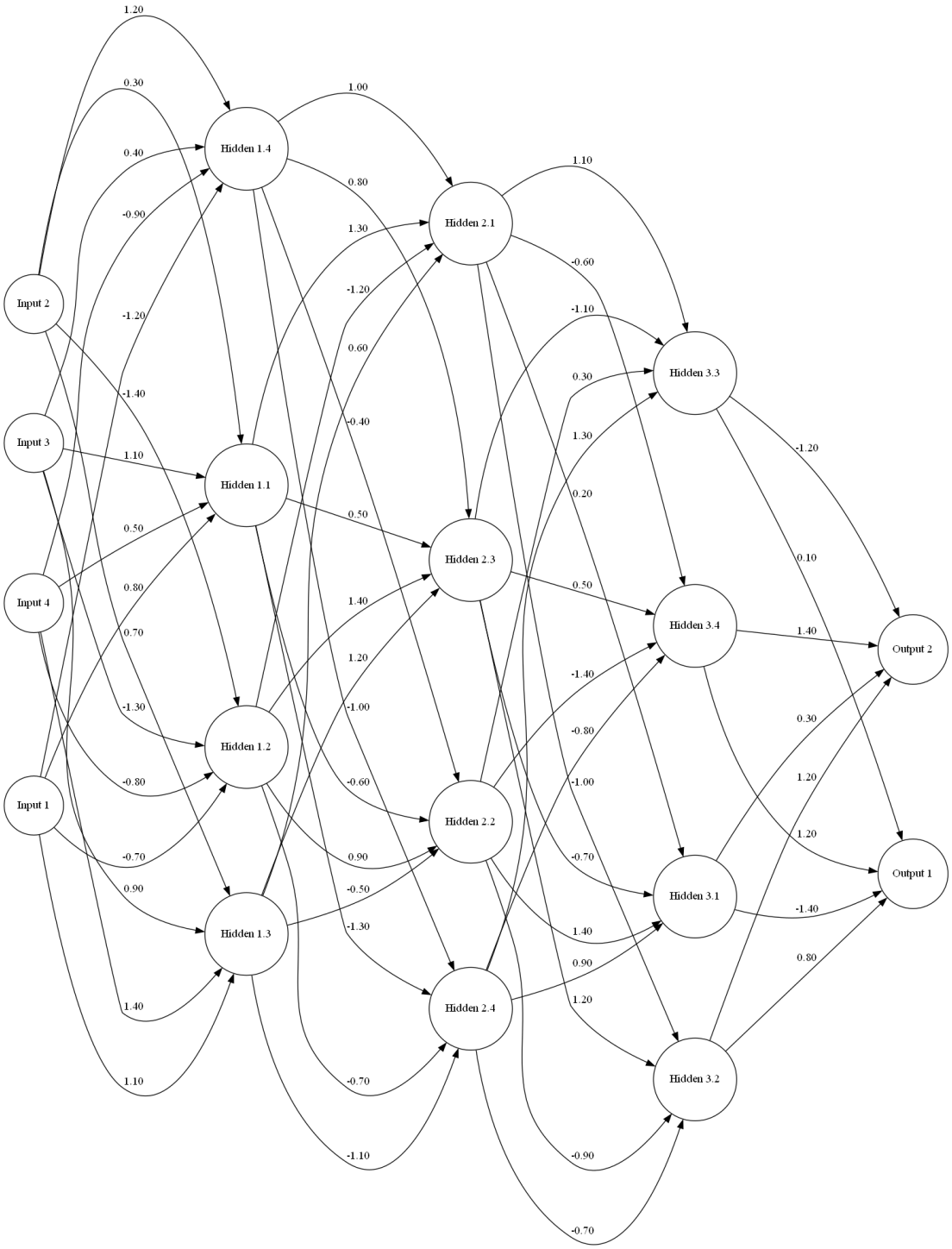
Nilai output yang didapat dari program beserta visualisasi grafnya:

```
Test case: multilayer_softmax.json

Output:
[[0.7042294 0.2957706]]

Max SSE: 0.00

Test Result:
Passed
```


Input 2
Input 3
Input 4
Input 1
Hidden 1.4
Hidden 1.1
Hidden 1.2
Hidden 1.3
Hidden 2.1
Hidden 2.3
Hidden 2.2
Hidden 2.4
Hidden 3.3
Hidden 3.4
Hidden 3.1
Hidden 3.2
Output 2
Output 1
1.20
0.30
0.40
-0.90
-1.20
-1.40
1.10
0.50
0.80
0.70
-1.30
-0.80
-0.70
0.90
1.40
1.10
1.00
0.80
1.30
-1.20
0.60
-0.40
0.50
1.40
1.20
-1.00
-0.60
0.90
-0.50
-1.30
-0.70
-1.10
1.10
-0.60
-1.10
0.30
1.30
0.20
0.50
-1.40
-0.80
-1.00
-0.70
1.40
0.90
1.20
-0.90
-0.70
-1.20
0.10
1.40
0.30
1.20
1.20
-1.40
0.80

B. Perbandingan Hasil Pengujian dengan Perhitungan Manual

1. Perhitungan manual untuk file relu.json (dari spesifikasi tugas besar)

Input	
-1	0,5

Bias		
0,1	0,2	0,3

Weights		
0,4	-0,5	0,6
0,7	0,8	-0,9

Activation Function = ReLU

Weighted Sum 1	0,05
Weighted Sum 2	1,1
Weighted Sum 3	-0,75
ReLU for Weighted Sum 1	0,05
ReLU for Weighted Sum 2	1,1
ReLU for Weighted Sum 3	0
Final Result	[0,05;1,1;0]

- Weighted Sum 1 = -1*0,4 + 0,5*0,7 + 0,1 = 0,05
- Weighted Sum 2 = -1*-0,5 + 0,5*0,8 + 0,2 = 1,1
- Weighted Sum 3 = -1*0,6 + 0,5*-0,9 + 0,3 = -0,75

ReLU Activation

- ReLU1 -> MAX(WeightedSum1, 0) = 0,05
- ReLU2 -> MAX(WeightedSum2, 0) = 1,1
- ReLU3 -> MAX(WeightedSum3, 0) = 0

Final Result = [0,05; 1,1; 0]

Dari hasil perhitungan manual yang ada, hasil yang didapat sudah sama dengan yang terdapat dalam program yaitu [0,05; 1,1; 0] sehingga jawaban sudah benar.

2. Perhitungan manual untuk file relu.json (dari test case)

Input	
1,5	-0,45

Bias		
0,1	0,2	0,3

Weights		
0,47	-0,6	0,2
1,1	-1,3	0,5

Activation Function = ReLU

Weighted Sum 1	0,31
Weighted Sum 2	-0,115
Weighted Sum 3	0,375
ReLU for Weighted Sum 1	0,31
ReLU for Weighted Sum 2	0
ReLU for Weighted Sum 3	0,375
Final Result	[0,31;0;0,375]

- Weighted Sum 1 = 1,5*0,47 + -0,45*1,1 + 0,1 = 0,31
- Weighted Sum 2 = 1,5*-0,6 + -0,45*-1,3 + 0,2 = -0,115
- Weighted Sum 3 = 1,5*0,2 + -0,45*0,5 + 0,3 = 0,375

ReLU Activation

- ReLU1 -> MAX(WeightedSum1, 0) = 0,31
- ReLU2 -> MAX(WeightedSum2, 0) = 0
- ReLU3 -> MAX(WeightedSum3, 0) = 0,375

Final Result = [0,31; 0; 0,375]

Dari hasil perhitungan manual yang ada, hasil yang didapat sudah sama dengan yang terdapat dalam program yaitu [0,31; 0; 0,375] sehingga jawaban sudah benar.

3. Perhitungan manual untuk file softmax.json

Input							
-1	1	2,8	1,8	-0,45	0,24	0,15	0,2

Bias		
0,1	0,9	-0,1

Weighted Sum 1	3,022	20,53231528
Weighted Sum 2	1,738	5,685960123
Weighted Sum 3	-0,442	0,642749635

Weights		
-0,2	0,8	0,2
0,3	-0,7	0,3
0,4	0,6	-0,4
0,5	0,5	0,5
-0,6	0,4	0,6
-0,7	-0,3	0,7
0,8	0,2	-0,8
0,9	-0,1	0

Softmax for Weighted Sum 1	0,764390609
Softmax for Weighted Sum 2	0,211680683
Softmax for Weighted Sum 3	0,023928708

Final Result
[0,76439061; 0,21168068; 0,02392871]

Activation Function = Softmax

- Weighted Sum 1 = -1*-0,2 + 1*0,3 + 2,8*0,4 + 1,8*0,5 + -0,45*-0,6 + 0,24*-0,7 + 0,15*0,8 + 0,2*0,9 + 0,1 = 3,022
- Weighted Sum 2 = -1*0,8 + 1*-0,7 + 2,8*0,6 + 1,8*0,5 + -0,45*0,4 + 0,24*-0,3 + 0,15*0,2 + 0,2*-0,1 + 0,9 = 1,738
- Weighted Sum 3 = -1*0,2 + 1*0,3 + 2,8*-0,4 + 1,8*0,5 + -0,45*0,6 + 0,24*0,7 + 0,15*-0,8 + 0,2*0 + -0,1 = -0,442

Softmax Activation

- e^3,022 = 20,53231528
- e^1,738 = 5,685960123
- e^-0,442 = 0,642749635

- Softmax1 = 20,53231528 / (20,53231528+5,685960123+0,642749635) = 0,76439061
- Softmax2 = 5,685960123 / (20,53231528+5,685960123+0,642749635) = 0,21168068
- Softmax3 = 0,642749635 / (20,53231528+5,685960123+0,642749635) = 0,02392871

Final Result = [0,76439061; 0,21168068; 0,02392871]

Dari hasil perhitungan manual yang ada, hasil yang didapat sudah sama dengan yang terdapat dalam program yaitu [0,31; 0; 0,375] sehingga jawaban sudah benar.

4. Perhitungan manual untuk file linear.json

Input									
-4	-3	-2	-1	0	1	2	3	4	5

Bias	1		Linear 1	-11		Linear 6	4
			Linear 2	-8		Linear 7	7
Weight	3		Linear 3	-5		Linear 8	10
			Linear 4	-2		Linear 9	13
Activation Function = Linear			Linear 5	1		Linear 10	16

- Linear 1 = -4*3 + 1 = -11
- Linear 2 = -3*3 + 1 = -8
- Linear 3 = -2*3 + 1 = -5
- Linear 4 = -1*3 + 1 = -2
- Linear 5 = 0*3 + 1 = 1
- Linear 6 = 1*3 + 1 = 4
- Linear 7 = 2*3 + 1 = 7
- Linear 8 = 3*3 + 1 = 10
- Linear 9 = 4*3 + 1 = 13
- Linear 10 = 5*3 + 1 = 16

Final Result: [[-11], [-8], [-5], [-2], [1], [4], [7], [10], [13], [16]]

Dari hasil perhitungan manual yang ada, hasil yang didapat sudah sama dengan yang terdapat dalam program yaitu [[-11], [-8], [-5], [-2], [1], [4], [7], [10], [13], [16]] sehingga jawaban sudah benar.

BAB IV
PEMBAGIAN KERJA

Nama	NIM	Pembagian Kerja
Jason Rivalino	13521008	Visualisasi Graf dengan Graphviz, Laporan
Muhammad Rifko Favian	13521075	FFNN Class, Visualisasi Graf dengan Graphviz, Implementasi perhitungan nilai output, Laporan
Moch. Sofyan Firdaus	13521083	Layer Class
Fazel Ginanda	13521098	Fungsi aktivasi