

Laporan Kecerdasan Buatan  
Ujian Tengah Semester  
2022



Oleh :

Asha Antania Anjani

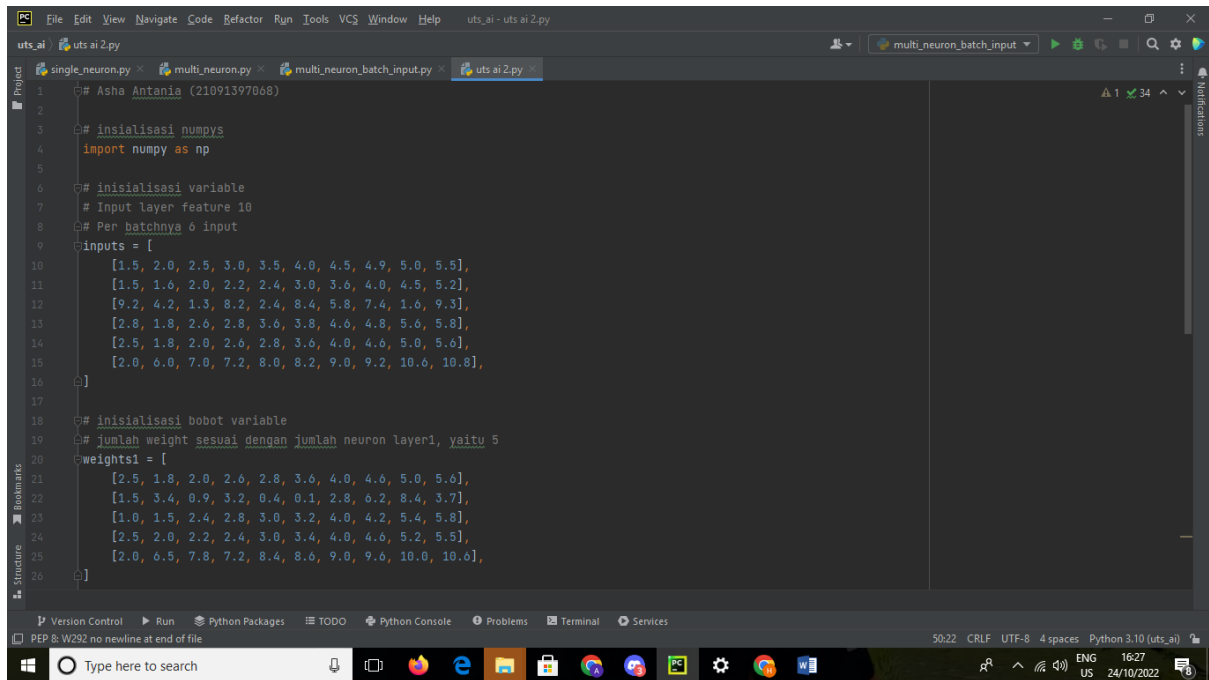
(21091397068)

Manajemen Informatika

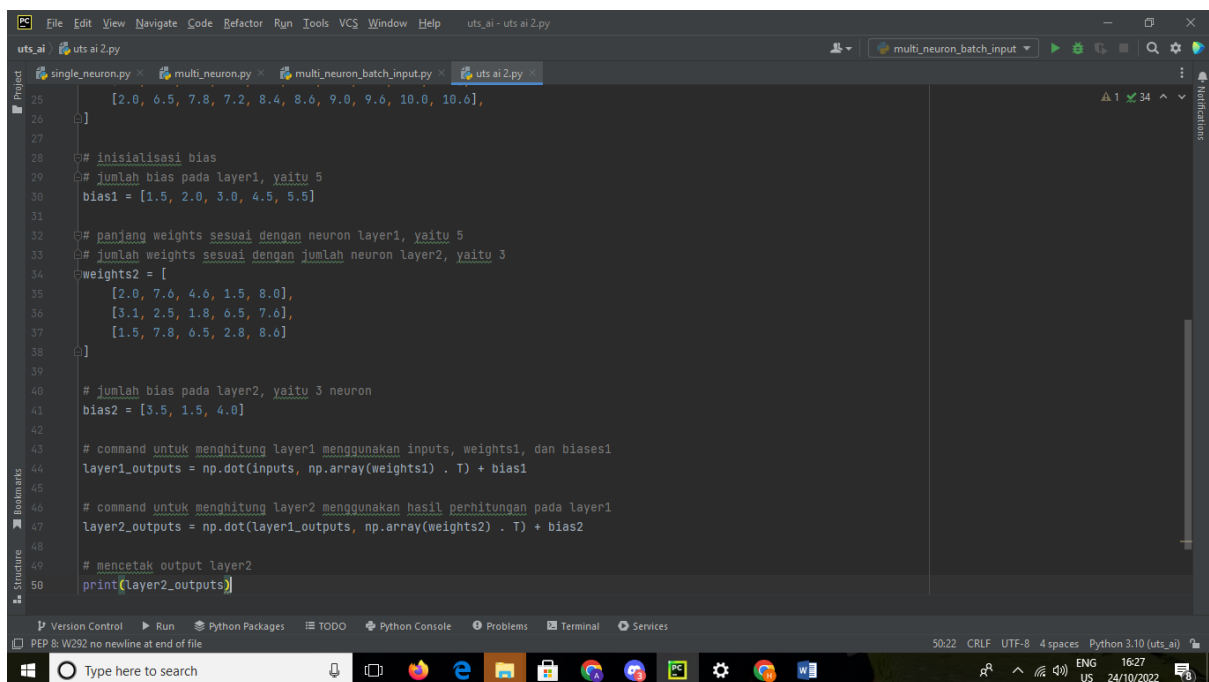
Fakultas Vokasi

Nama : Asha Antania

Nim : 21091397068



```
1 Asha Antania (21091397068)
2
3 # inisialisasi numpys
4 import numpy as np
5
6 # inisialisasi variable
7 # Input layer feature 10
8 # Per batchnya 6 input
9 inputs = [
10     [1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 4.9, 5.0, 5.5],
11     [1.5, 1.6, 2.0, 2.2, 2.4, 3.0, 3.6, 4.0, 4.5, 5.2],
12     [9.2, 4.2, 1.3, 8.2, 2.4, 8.4, 5.8, 7.4, 1.6, 9.3],
13     [2.8, 1.8, 2.6, 2.8, 3.6, 3.8, 4.6, 4.8, 5.6, 5.8],
14     [2.5, 1.8, 2.0, 2.6, 2.8, 3.6, 4.0, 4.6, 5.0, 5.6],
15     [2.0, 6.0, 7.0, 7.2, 8.0, 8.2, 9.0, 9.2, 10.6, 10.8],
16 ]
17
18 # inisialisasi bobot variable
19 # jumlah weight sesuai dengan jumlah neuron layer1, yaitu 5
20 weights1 = [
21     [2.5, 1.8, 2.0, 2.6, 2.8, 3.6, 4.0, 4.6, 5.0, 5.6],
22     [1.5, 3.4, 0.9, 3.2, 0.4, 0.1, 2.8, 6.2, 8.4, 3.7],
23     [1.0, 1.5, 2.4, 2.8, 3.0, 3.2, 4.0, 4.2, 5.4, 5.8],
24     [2.5, 2.0, 2.2, 2.4, 3.0, 3.4, 4.0, 4.6, 5.2, 5.5],
25     [2.0, 6.5, 7.8, 7.2, 8.4, 8.6, 9.0, 9.6, 10.0, 10.6],
26 ]
```



```
25     [2.0, 6.5, 7.8, 7.2, 8.4, 8.6, 9.0, 9.6, 10.0, 10.6],
26 ]
27
28 # inisialisasi bias
29 # jumlah bias pada layer1, yaitu 5
30 bias1 = [1.5, 2.0, 3.0, 4.5, 5.5]
31
32 # panjang weights sesuai dengan neuron layer1, yaitu 5
33 # jumlah weights sesuai dengan jumlah neuron layer2, yaitu 3
34 weights2 = [
35     [2.0, 7.6, 4.6, 1.5, 8.0],
36     [3.1, 2.5, 1.8, 6.5, 7.6],
37     [1.5, 7.8, 6.5, 2.8, 8.6]
38 ]
39
40 # jumlah bias pada layer2, yaitu 3 neuron
41 bias2 = [3.5, 1.5, 4.0]
42
43 # command untuk menghitung layer1 menggunakan inputs, weights1, dan biases1
44 layer1_outputs = np.dot(inputs, np.array(weights1) . T) + bias1
45
46 # command untuk menghitung layer2 menggunakan hasil perhitungan pada layer1
47 layer2_outputs = np.dot(layer1_outputs, np.array(weights2) . T) + bias2
48
49 # mencetak output layer2
50 print(layer2_outputs)
```

The screenshot shows a Visual Studio Code editor window with a Python file named `uts_ai_2.py`. The code defines a list of inputs for a neural network. Below the code, the 'Run' output window displays the results of the script execution.

```
1 # Asha Antania (21091397068)
2
3 # inisialisasi numpy
4 import numpy as np
5
6 # inisialisasi variable
7 # Input layer feature 10
8 # Per batchnya 6 input
9 inputs = [
10     [1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 4.9, 5.0, 5.5],
11     [1.5, 1.6, 2.0, 2.2, 2.4, 3.0, 3.6, 4.0, 4.5, 5.2],
12     [9.2, 4.2, 1.3, 8.2, 2.4, 8.4, 5.8, 7.4, 1.6, 9.3],
13     [2.8, 1.8, 2.6, 2.8, 3.6, 3.8, 4.6, 4.8, 5.6, 5.8],
14     [2.5, 1.8, 2.0, 2.6, 2.8, 3.6, 4.0, 4.6, 5.0, 5.6]]
```

Run: multi\_neuron\_batch\_input

```
[[ 147.94  141.08  144.34  321.24 1294.67]
 [ 123.57  119.32  121.83  266.22 1084.48]
 [ 1130.8 1089.4 1092.3 2441.1 10073.55]
 [ 139.61  134.    138.09  299.52 1215.84]
 [ 308.92  294.72  297.4  681.46 2710.3 ]
 [ 569.06  521.2   544.54 1291.7  4783.82]]
```

Process finished with exit code 0

#### Penjelasan :

Pada baris 4 terdapat syntax untuk mengimport library python yaitu numpy. Pada baris 9 sampai 15 terdapat syntax variabel inputs yang bertipe data array yang memiliki 6 batch yang masing-masing batch nya terdiri dari 10 layer. Pada baris 20 sampai 25 terdapat variabel weights1 dan baris 34 sampai 37 terdapat variabel weights2 yang memiliki tipe data multiple array yang menampung nilai neuron berjumlah 5 baris variabel weights1 dan 3 baris variabel weights2 . Pada baris 30 terdapat variabel bias1 dan baris 41 untuk variabel bias2 yang nantinya digunakan untuk menghitung hasil dari output. Pada baris 44 dan 47 terdapat np.dot yang berfungsi untuk mengembalikan nilai array dan np.array yang berfungsi untuk mengembalikan variabel weights yang memiliki data array 2 dimensi agar bisa ditampilkan oleh console. Lalu saat di run yang menampilkan hasil.

Input 6*10	weights 5*10		output input * weight1
1.5 2.0 2.5 3.0 3.5 4.0 4.5 4.9 5.0 5.5	2.5 1.8 2.0 2.6 2.8 3.6 4.0 4.6 5.0 5.6	=	140.69 128.03 139.68 141.34 316.74]
1.5 1.6 2.0 2.2 2.4 3.0 3.6 4.0 4.5 5.2	1.5 3.4 0.9 3.2 0.4 0.1 2.8 6.2 8.4 3.7		[118.29 109.71 117.32 118.83 261.72]
9.2 4.2 1.3 8.2 2.4 8.4 5.8 7.4 1.6 9.3	1.0 1.5 2.4 2.8 3.0 3.2 4.0 4.2 5.4 5.8		[208.76 167.26 192.52 206.41 445.1]
2.8 1.8 2.6 2.8 3.6 3.8 4.6 4.8 5.6 5.8	2.5 2.0 2.2 2.4 3.0 3.4 4.0 4.6 5.2 5.5		[134.57 123.91 132. 135.09 295.02]
2.5 1.8 2.0 2.6 2.8 3.6 4.0 4.6 5.0 5.6	2.0 6.5 7.8 7.2 8.4 8.6 9.0 9.6 10.0 10.6		[292.24 268. 292.72 294.4 676.96]]
2.0 6.0 7.0 7.2 8.0 8.2 9.0 9.2 10.6 10.8			

output input * weight1		biases 1		output layer 1																																																												
<table> <tr><td>140.69</td><td>128.03</td><td>139.68</td><td>141.34</td><td>316.74</td></tr> <tr><td>118.29</td><td>109.71</td><td>117.32</td><td>118.83</td><td>261.72</td></tr> <tr><td>208.76</td><td>167.26</td><td>192.52</td><td>206.41</td><td>445.1</td></tr> <tr><td>134.57</td><td>123.91</td><td>132.</td><td>135.09</td><td>295.02</td></tr> <tr><td>292.24</td><td>268.</td><td>292.72</td><td>294.4</td><td>676.96</td></tr> </table>	140.69	128.03	139.68	141.34	316.74	118.29	109.71	117.32	118.83	261.72	208.76	167.26	192.52	206.41	445.1	134.57	123.91	132.	135.09	295.02	292.24	268.	292.72	294.4	676.96	+	<table> <tr><td>1.5</td><td>2.0</td><td>3.0</td><td>4.5</td><td>5.5</td></tr> </table>	1.5	2.0	3.0	4.5	5.5	=	<table> <tr><td>147.94</td><td>141.68</td><td>144.34</td><td>321.24</td><td>1294.67</td></tr> <tr><td>123.57</td><td>119.32</td><td>121.83</td><td>266.22</td><td>1084.48</td></tr> <tr><td>1130.8</td><td>1089.4</td><td>1092.3</td><td>2441.1</td><td>10073.55</td></tr> <tr><td>139.61</td><td>134.</td><td>138.09</td><td>299.52</td><td>1215.84</td></tr> <tr><td>308.92</td><td>294.72</td><td>297.4</td><td>681.46</td><td>2710.3</td></tr> <tr><td>569.06</td><td>521.2</td><td>544.54</td><td>1291.7</td><td>4783.82</td></tr> </table>	147.94	141.68	144.34	321.24	1294.67	123.57	119.32	121.83	266.22	1084.48	1130.8	1089.4	1092.3	2441.1	10073.55	139.61	134.	138.09	299.52	1215.84	308.92	294.72	297.4	681.46	2710.3	569.06	521.2	544.54	1291.7	4783.82
140.69	128.03	139.68	141.34	316.74																																																												
118.29	109.71	117.32	118.83	261.72																																																												
208.76	167.26	192.52	206.41	445.1																																																												
134.57	123.91	132.	135.09	295.02																																																												
292.24	268.	292.72	294.4	676.96																																																												
1.5	2.0	3.0	4.5	5.5																																																												
147.94	141.68	144.34	321.24	1294.67																																																												
123.57	119.32	121.83	266.22	1084.48																																																												
1130.8	1089.4	1092.3	2441.1	10073.55																																																												
139.61	134.	138.09	299.52	1215.84																																																												
308.92	294.72	297.4	681.46	2710.3																																																												
569.06	521.2	544.54	1291.7	4783.82																																																												

output layer 1		weight 2 3*5																																																																	
<table> <tr><td>147.94</td><td>141.68</td><td>144.34</td><td>321.24</td><td>1294.67</td></tr> <tr><td>123.57</td><td>119.32</td><td>121.83</td><td>266.22</td><td>1084.48</td></tr> <tr><td>1130.8</td><td>1089.4</td><td>1092.3</td><td>2441.1</td><td>10073.55</td></tr> <tr><td>139.61</td><td>134.</td><td>138.09</td><td>299.52</td><td>1215.84</td></tr> <tr><td>308.92</td><td>294.72</td><td>297.4</td><td>681.46</td><td>2710.3</td></tr> <tr><td>569.06</td><td>521.2</td><td>544.54</td><td>1291.7</td><td>4783.82</td></tr> </table>	147.94	141.68	144.34	321.24	1294.67	123.57	119.32	121.83	266.22	1084.48	1130.8	1089.4	1092.3	2441.1	10073.55	139.61	134.	138.09	299.52	1215.84	308.92	294.72	297.4	681.46	2710.3	569.06	521.2	544.54	1291.7	4783.82		<table> <tr><td>2.0,</td><td>7.6,</td><td>4.6,</td><td>1.5,</td><td>8.0</td></tr> <tr><td>3.1,</td><td>2.5,</td><td>1.8,</td><td>6.5,</td><td>7.6</td></tr> <tr><td>1.5,</td><td>7.8,</td><td>6.5,</td><td>2.8,</td><td>8.6</td></tr> </table>	2.0,	7.6,	4.6,	1.5,	8.0	3.1,	2.5,	1.8,	6.5,	7.6	1.5,	7.8,	6.5,	2.8,	8.6	=	<table> <tr><td>4725.616</td><td>4419.672</td><td>5334.555</td></tr> <tr><td>3964.803</td><td>3699.717</td><td>4476.519</td></tr> <tr><td>6527.453</td><td>6222.367</td><td>7372.206</td></tr> <tr><td>4894.696</td><td>4578.98</td><td>5525.964</td></tr> <tr><td>4463.601</td><td>4170.879</td><td>5039.027</td></tr> <tr><td>9907.822</td><td>9247.463</td><td>11174.866</td></tr> </table>	4725.616	4419.672	5334.555	3964.803	3699.717	4476.519	6527.453	6222.367	7372.206	4894.696	4578.98	5525.964	4463.601	4170.879	5039.027	9907.822	9247.463	11174.866
147.94	141.68	144.34	321.24	1294.67																																																															
123.57	119.32	121.83	266.22	1084.48																																																															
1130.8	1089.4	1092.3	2441.1	10073.55																																																															
139.61	134.	138.09	299.52	1215.84																																																															
308.92	294.72	297.4	681.46	2710.3																																																															
569.06	521.2	544.54	1291.7	4783.82																																																															
2.0,	7.6,	4.6,	1.5,	8.0																																																															
3.1,	2.5,	1.8,	6.5,	7.6																																																															
1.5,	7.8,	6.5,	2.8,	8.6																																																															
4725.616	4419.672	5334.555																																																																	
3964.803	3699.717	4476.519																																																																	
6527.453	6222.367	7372.206																																																																	
4894.696	4578.98	5525.964																																																																	
4463.601	4170.879	5039.027																																																																	
9907.822	9247.463	11174.866																																																																	

	biases 2		output layer 2																					
+	<table> <tr><td>3.5,</td><td>1.5,</td><td>4.0</td></tr> </table>	3.5,	1.5,	4.0	=	<table> <tr><td>4729.116</td><td>4421.172</td><td>5338.555</td></tr> <tr><td>3968.303</td><td>3701.217</td><td>4480.519</td></tr> <tr><td>6530.953</td><td>6223.867</td><td>7376.206</td></tr> <tr><td>4898.196</td><td>4580.48</td><td>5529.964</td></tr> <tr><td>4467.101</td><td>4172.379</td><td>5043.027</td></tr> <tr><td>9911.322</td><td>9248.936</td><td>11178.866</td></tr> </table>	4729.116	4421.172	5338.555	3968.303	3701.217	4480.519	6530.953	6223.867	7376.206	4898.196	4580.48	5529.964	4467.101	4172.379	5043.027	9911.322	9248.936	11178.866
3.5,	1.5,	4.0																						
4729.116	4421.172	5338.555																						
3968.303	3701.217	4480.519																						
6530.953	6223.867	7376.206																						
4898.196	4580.48	5529.964																						
4467.101	4172.379	5043.027																						
9911.322	9248.936	11178.866																						

Setelah mendapatkan hasil layer 1 lalu lanjut mencari layer 2 dg cara output layer 1 dikali weight transpose 2 ditambah biases 2, jadi keluar output layer 2.