

**Tugas Besar 1 IF3270 Machine Learning**  
**Semester II tahun 2024/2025**  
**Feedforward Neural Network**



Oleh:

Jimly Nur Arif (13522123)

Samy Muhammad Haikal (13522151)

Muhammad Roihan (13522152)

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

## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>2</b>
<b>DESKRIPSI PERSOALAN</b>	<b>3</b>
<b>PEMBAHASAN</b>	<b>5</b>
1.1 Forward Propagation	5
1.2 Backward Propagation	6
1.3 Kelas dan Objek	7
Tabel 1.3.1	7
Tabel 1.3.2	8
Tabel 1.3.3	9
Tabel 1.3.4	10
<b>HASIL PENGUJIAN</b>	<b>11</b>
2.1 Pengaruh depth dan width	11
2.2 Pengaruh fungsi aktivasi	13
2.3 Pengaruh Learning rate	15
2.4 Pengaruh Inisialisasi bobot	17
2.5 Perbandingan dengan library sklearn	19
<b>KESIMPULAN DAN SARAN</b>	<b>21</b>
<b>PEMBAGIAN TUGAS</b>	<b>23</b>
<b>REFERENSI</b>	<b>24</b>
<b>LAMPIRAN</b>	<b>25</b>

## DESKRIPSI PERSOALAN

Implementasikan suatu modul FFNN yang memenuhi ketentuan-ketentuan berikut:

- FFNN yang diimplementasikan dapat **menerima jumlah neuron dari tiap layer** (termasuk input layer dan output layer)
- FFNN yang diimplementasikan dapat **menerima fungsi aktivasi dari tiap layer**. Pilihan fungsi aktivasi yang harus diimplementasikan adalah sebagai berikut:
  - Linear
  - ReLU
  - Sigmoid
  - Hyperbolic Tangent (tanh)
  - Softmax
- FFNN yang diimplementasikan dapat **menerima fungsi loss** dari model tersebut. Pilihan loss function yang harus diimplementasikan adalah sebagai berikut:
  - [MSE](#)
  - [Binary Cross-Entropy](#)
  - [Categorical Cross-Entropy](#)
- Terdapat mekanisme untuk **inisialisasi bobot** tiap neuron (termasuk bias). Pilihan metode inisialisasi bobot yang harus diimplementasikan adalah sebagai berikut:
  - **Zero initialization**
  - Random dengan distribusi **uniform**.
  - Random dengan distribusi **normal**.
- Model memiliki implementasi **forward propagation** dengan ketentuan sebagai berikut:
  - Dapat menerima input berupa **batch**.
- Model memiliki implementasi **backward propagation** untuk menghitung perubahan gradien:
  - Dapat menangani perhitungan perubahan gradien untuk input data **batch**.
  - Gunakan konsep **chain rule** untuk menghitung gradien tiap bobot terhadap loss function.
- Model memiliki implementasi **weight update** dengan menggunakan **gradient descent** untuk memperbarui bobot berdasarkan gradien yang telah dihitung, berikut persamaannya:

$$W_{new} = W_{old} - \alpha \left( \frac{\partial \mathcal{L}}{\partial W_{old}} \right)$$

$\alpha$  = Learning rate

- Implementasi untuk pelatihan model harus memenuhi ketentuan berikut:
  - Dapat menerima parameter berikut:

- Batch size
  - Learning rate
  - Jumlah epoch
  - Verbose
    - Verbose 0 berarti tidak menampilkan apa-apa selama pelatihan
    - Verbose 1 berarti hanya menampilkan progress bar beserta dengan kondisi training loss dan validation loss saat itu
- Proses pelatihan mengembalikan **histori dari proses pelatihan** yang berisi **training loss** dan **validation loss tiap epoch**.
- Lakukan **pengujian** terhadap implementasi FFNN dengan ketentuan sebagai berikut:
  - Analisis pengaruh beberapa hyperparameter sebagai berikut:
    - Pengaruh **depth (banyak layer)** dan **width (banyak neuron per layer)**
    - Pengaruh **fungsi aktivasi** hidden layer
    - Pengaruh **learning rate**
    - Pengaruh **inisialisasi bobot**
  - Analisis perbandingan hasil prediksi dengan [library sklearn MLP](#)
    - Lakukan satu kali pelatihan dengan hyperparameter yang sama untuk kedua model
    - Hyperparameter yang digunakan dibebaskan
    - Bandingkan hasil akhir prediksinya saja

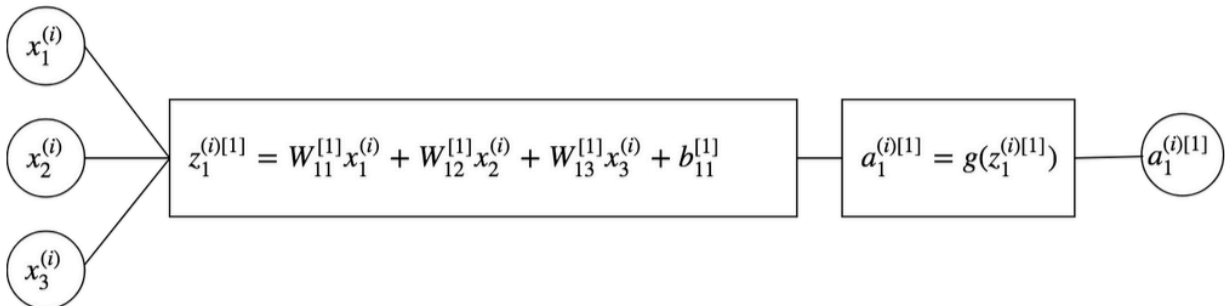
## PEMBAHASAN

### 1.1 Forward Propagation

Forward propagation adalah tahap di mana input melewati seluruh lapisan dalam neural network untuk menghasilkan output akhir. Pada proses ini semua input, bobot, dan bias akan digabung untuk menghasilkan nilai yang akan digunakan untuk input layer selanjutnya. Setiap neuron menerima input dari layer sebelumnya, dikalikan dengan bobot ( $W$ ) dan ditambahkan bias ( $b$ ). Secara matematis, proses ini dituliskan sebagai:

$$Z = \sum_{i \in \text{input}-n} W_i \cdot X_i + b$$

Di mana  $W$  adalah bobot yang menentukan seberapa besar pengaruh setiap input terhadap keluaran neuron, sementara  $b$  adalah bias yang memungkinkan model lebih fleksibel dalam belajar pola data. Setelah nilai  $Z$  dihitung, hasil ini diteruskan ke fungsi aktivasi yang bersifat non-linear, seperti ReLU, tanh, sigmoid, atau softmax atau sigmoid. Fungsi aktivasi ini mengubah  $Z$  menjadi nilai  $A$ , yang akan digunakan sebagai input untuk layer berikutnya.



Keberadaan fungsi aktivasi non-linear sangat penting dalam neural network. Jika tidak digunakan dan jaringan hanya mengandalkan kombinasi linear dari bobot dan bias, maka seluruh jaringan akan setara dengan satu lapisan tunggal, tidak peduli berapa banyak lapisan yang digunakan. Hal ini mengurangi kemampuan jaringan untuk mempelajari pola yang lebih kompleks. Aktivasi non-linear memungkinkan model untuk menangkap hubungan yang lebih rumit dalam data, seperti pola non-linear dalam pengenalan gambar atau pemrosesan bahasa alami.

## 1.2 Backward Propagation

Backward propagation, atau backpropagation, adalah proses yang digunakan untuk memperbarui bobot ( $W$ ) dan bias ( $b$ ) dalam jaringan saraf berdasarkan kesalahan (error) yang dihasilkan selama forward propagation. Proses ini dilakukan dengan menghitung turunan (gradien) dari fungsi loss terhadap parameter-parameter jaringan menggunakan aturan rantai dalam kalkulus.

Backward propagation dimulai dari output layer dengan menghitung selisih antara prediksi jaringan ( $A$ ) dan nilai target ( $Y$ ). Fungsi-fungsi loss yang diimplementasikan adalah

Nama Fungsi Loss	Definisi Fungsi
<a href="#">MSE</a>	$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
<a href="#">Binary Cross-Entropy</a>	$\mathcal{L}_{BCE} = -\frac{1}{n} \sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$ <p><math>y_i</math> = Actual binary label (0 or 1) <math>\hat{y}_i</math> = Predicted value of <math>y_i</math> <math>n</math> = Batch size</p>
<a href="#">Categorical Cross-Entropy</a>	$\mathcal{L}_{CCE} = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^C (y_{ij} \log \hat{y}_{ij})$ <p><math>y_{ij}</math> = Actual value of instance <math>i</math> for class <math>j</math> <math>\hat{y}_{ij}</math> = Predicted value of <math>y_{ij}</math> <math>C</math> = Number of classes <math>n</math> = Batch size</p>

Setelah mendapatkan  $dZ$  pada output layer, propagasi mundur berlanjut ke hidden layer. Perhitungan ini dilakukan secara berulang dari layer terakhir hingga layer pertama. Setelah

semua gradien dihitung, bobot ( $W$ ) dan bias ( $b$ ) diperbarui menggunakan algoritma gradient descent dengan learning rate:

$$W = W - \alpha \cdot dW$$

$$b = b - \alpha \cdot db$$

Backpropagation memungkinkan jaringan belajar dengan menyesuaikan parameter agar loss berkurang secara bertahap. Jika tidak dilakukan backward propagation, jaringan tidak akan mampu memperbaiki kesalahan dan belajar dari data, sehingga tidak bisa melakukan tugasnya dengan baik.

Neural network menggunakan kombinasi forward dan backward propagation untuk mempelajari pola yang kompleks dalam data, seperti pengenalan objek dalam gambar atau pemrosesan bahasa alami.

### 1.3 Kelas dan Objek

Berikut adalah kelas-kelas yang diimplementasikan.

Tabel 1.3.1

CLASS		
	FFNN	Kelas utama untuk memodelkan feedforward neural network
ATTRIBUTE		
	layer_sizes (list): Daftar ukuran tiap layer activations (list): Daftar fungsi aktivasi tiap layer loss_function (str): Fungsi loss yang digunakan weight_inits (list): Daftar konfigurasi inisialisasi bobot	
METHOD		
	forward	Method untuk melakukan forward pass
	backward	Method untuk melakukan backward pass
	_activation_derivative	Method helper untuk menghitung turunan aktivasi
	train	Method untuk melakukan training model

	<code>_compute_loss</code>	Method untuk menghitung loss
	<code>plot_as_graph</code>	Method untuk visualisasi graf
	<code>plot_weight_distribution</code>	Method untuk memplot distribusi bobot di setiap layer
	<code>plot_gradient_distribution</code>	Method untuk memplot distribusi gradien di setiap layer
	<code>save_model</code>	Method untuk menyimpan model
	<code>load_model</code>	Method untuk memuat model

Tabel 1.3.2

CLASS		
	Layer	Kelas untuk memodelkan layer
ATTRIBUTE		
	input_size (int): Jumlah neuron dari layer sebelumnya output_size (int): Jumlah neuron di layer ini activation (str): Fungsi aktivasi ('relu', 'sigmoid', 'softmax', dll) weight_init (dict): Konfigurasi inisialisasi bobot	
METHOD		
	forward	Method untuk melakukan forward pass



	backward	Method untuk melakukan backward pass
	_initialize_weights	Method untuk inisialisasi nilai bobot

Tabel 1.3.3

<b>CLASS</b>		
	Loss	Kelas yang berisi fungsi loss
<b>ATTRIBUTE</b>		
	-	
<b>METHOD</b>		
	mse	Implementasi fungsi Mean Squared Error (MSE)
	mse_derivative	Implementasi turunan fungsi Mean Squared Error (MSE)
	binary_cross_entropy	Implementasi fungsi binary cross entropy
	binary_cross_entropy_derivative	Implementasi turunan fungsi binary cross entropy
	categorical_cross_entropy	Implementasi fungsi categorical cross entropy
	categorical_cross_entropy_derivative	Implementasi turunan fungsi categorical cross entropy

--	--

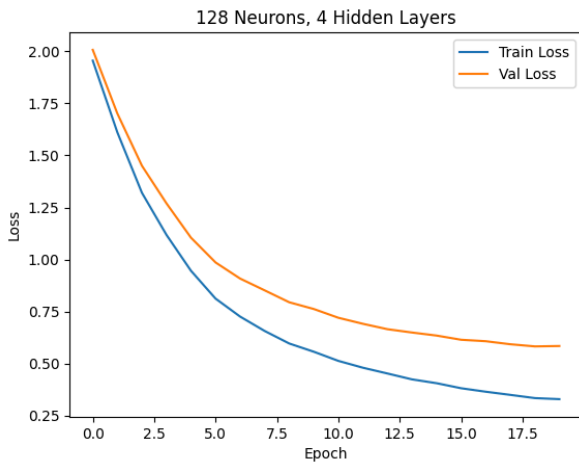
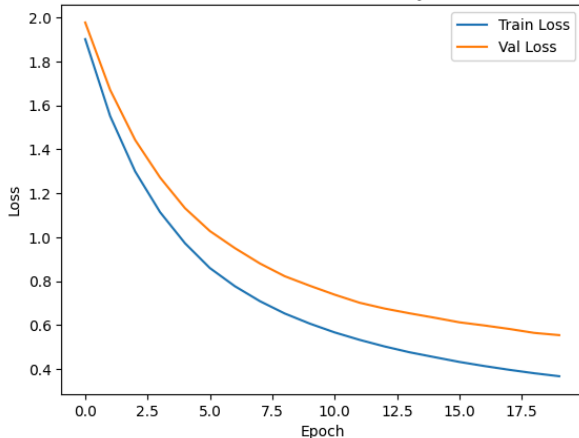
Tabel 1.3.4

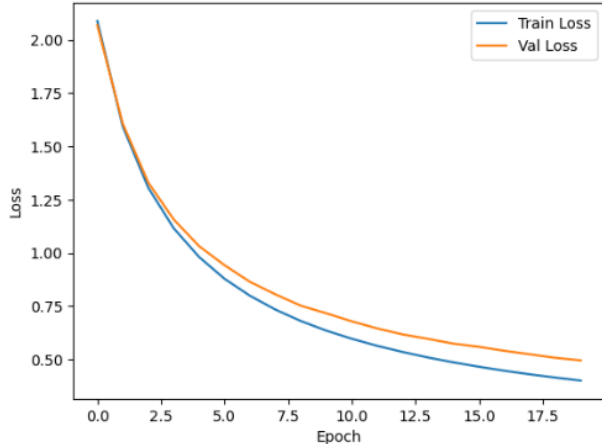
CLASS		
	Activation	Kelas yang berisi fungsi aktivasi
ATTRIBUTE		
	-	
METHOD		
	linear	Implementasi fungsi aktivasi linear
	linear_derivative	Implementasi turunan fungsi aktivasi linear
	relu	Implementasi fungsi aktivasi Rectified Linear Unit(ReLU)
	relu_derivative	Implementasi turunan fungsi aktivasi Rectified Linear Unit(ReLU)
	sigmoid	Implementasi fungsi aktivasi sigmoid
	sigmoid_derivative	Implementasi turunan fungsi aktivasi sigmoid
	tanh	Implementasi fungsi aktivasi tangen hiperbolik (tanh)
	tanh_derivative	Implementasi turunan fungsi aktivasi tangen hiperbolik (tanh)

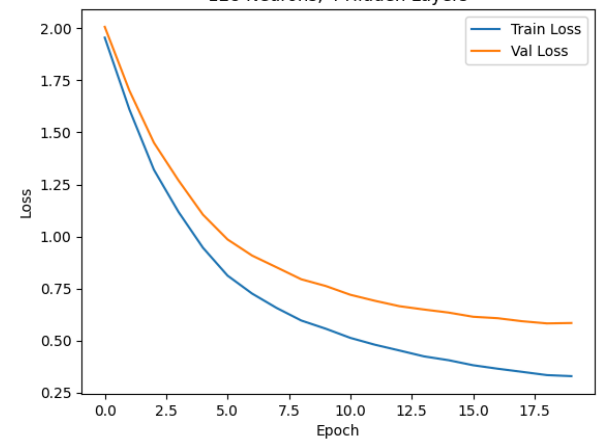
	softmax	Implementasi fungsi aktivasi softmax
	softmax_derivative	Implementasi turunan fungsi aktivasi softmax

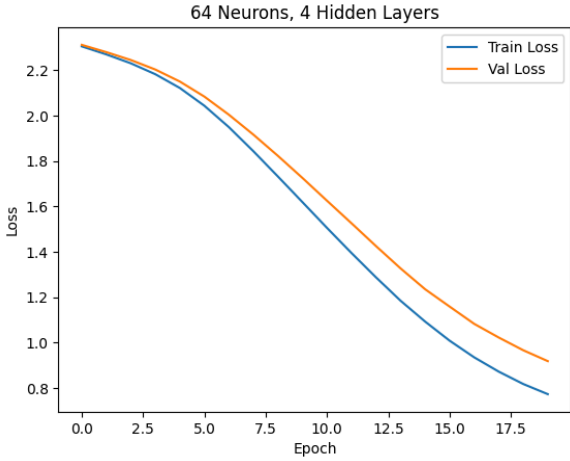
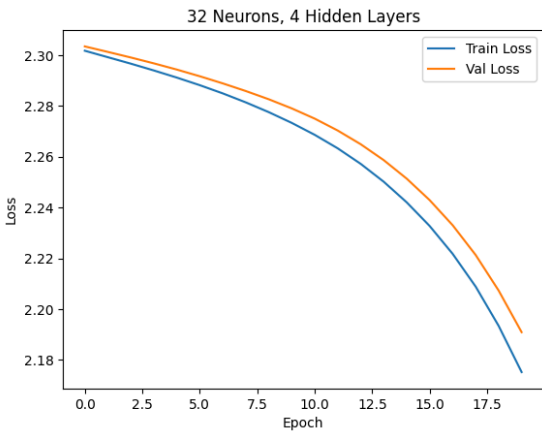
## HASIL PENGUJIAN

### 2.1 Pengaruh depth dan width

Depth	Loss graph																											
128 Neuron, 4 Hidden Layer	 <p>128 Neurons, 4 Hidden Layers</p> <table><thead><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr></thead><tbody><tr><td>0.0</td><td>2.00</td><td>2.00</td></tr><tr><td>2.5</td><td>1.30</td><td>1.40</td></tr><tr><td>5.0</td><td>0.85</td><td>1.00</td></tr><tr><td>7.5</td><td>0.65</td><td>0.85</td></tr><tr><td>10.0</td><td>0.55</td><td>0.75</td></tr><tr><td>12.5</td><td>0.48</td><td>0.68</td></tr><tr><td>15.0</td><td>0.42</td><td>0.62</td></tr><tr><td>17.5</td><td>0.38</td><td>0.60</td></tr></tbody></table>	Epoch	Train Loss	Val Loss	0.0	2.00	2.00	2.5	1.30	1.40	5.0	0.85	1.00	7.5	0.65	0.85	10.0	0.55	0.75	12.5	0.48	0.68	15.0	0.42	0.62	17.5	0.38	0.60
Epoch	Train Loss	Val Loss																										
0.0	2.00	2.00																										
2.5	1.30	1.40																										
5.0	0.85	1.00																										
7.5	0.65	0.85																										
10.0	0.55	0.75																										
12.5	0.48	0.68																										
15.0	0.42	0.62																										
17.5	0.38	0.60																										
128 Neuron, 3 Hidden Layer	 <p>128 Neurons, 3 Hidden Layers</p> <table><thead><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr></thead><tbody><tr><td>0.0</td><td>2.00</td><td>2.00</td></tr><tr><td>2.5</td><td>1.30</td><td>1.40</td></tr><tr><td>5.0</td><td>0.85</td><td>1.00</td></tr><tr><td>7.5</td><td>0.65</td><td>0.85</td></tr><tr><td>10.0</td><td>0.55</td><td>0.75</td></tr><tr><td>12.5</td><td>0.48</td><td>0.68</td></tr><tr><td>15.0</td><td>0.42</td><td>0.62</td></tr><tr><td>17.5</td><td>0.40</td><td>0.60</td></tr></tbody></table>	Epoch	Train Loss	Val Loss	0.0	2.00	2.00	2.5	1.30	1.40	5.0	0.85	1.00	7.5	0.65	0.85	10.0	0.55	0.75	12.5	0.48	0.68	15.0	0.42	0.62	17.5	0.40	0.60
Epoch	Train Loss	Val Loss																										
0.0	2.00	2.00																										
2.5	1.30	1.40																										
5.0	0.85	1.00																										
7.5	0.65	0.85																										
10.0	0.55	0.75																										
12.5	0.48	0.68																										
15.0	0.42	0.62																										
17.5	0.40	0.60																										

128 Neuron, 2 Hidden Layer	<div>128 Neurons, 2 Hidden Layers</div>  <table><caption>Approximate data for 128 Neurons, 2 Hidden Layers</caption><thead><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr></thead><tbody><tr><td>0.0</td><td>2.00</td><td>2.00</td></tr><tr><td>2.5</td><td>1.25</td><td>1.30</td></tr><tr><td>5.0</td><td>0.90</td><td>1.00</td></tr><tr><td>7.5</td><td>0.70</td><td>0.80</td></tr><tr><td>10.0</td><td>0.60</td><td>0.70</td></tr><tr><td>12.5</td><td>0.50</td><td>0.60</td></tr><tr><td>15.0</td><td>0.45</td><td>0.55</td></tr><tr><td>17.5</td><td>0.40</td><td>0.50</td></tr></tbody></table>	Epoch	Train Loss	Val Loss	0.0	2.00	2.00	2.5	1.25	1.30	5.0	0.90	1.00	7.5	0.70	0.80	10.0	0.60	0.70	12.5	0.50	0.60	15.0	0.45	0.55	17.5	0.40	0.50
Epoch	Train Loss	Val Loss																										
0.0	2.00	2.00																										
2.5	1.25	1.30																										
5.0	0.90	1.00																										
7.5	0.70	0.80																										
10.0	0.60	0.70																										
12.5	0.50	0.60																										
15.0	0.45	0.55																										
17.5	0.40	0.50																										
<b>Perbandingan Akurasi</b>																												
TC 1 Accuracy: 84.86%   Waktu: 6.78s																												
TC 2 Accuracy: 85.22%   Waktu: 4.98s																												
TC 3 Accuracy: 84.81%   Waktu: 3.20s																												

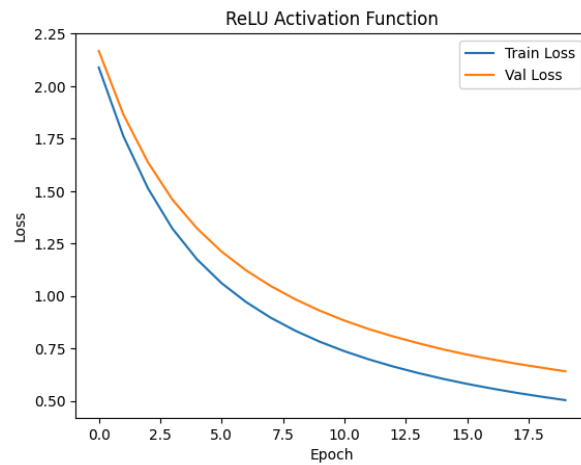
Width	Loss graph																											
128 Neuron, 4 Hidden Layer	<div><div>128 Neurons, 4 Hidden Layers</div><table><thead><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr></thead><tbody><tr><td>0.0</td><td>2.00</td><td>2.00</td></tr><tr><td>2.5</td><td>1.30</td><td>1.40</td></tr><tr><td>5.0</td><td>0.85</td><td>1.00</td></tr><tr><td>7.5</td><td>0.65</td><td>0.85</td></tr><tr><td>10.0</td><td>0.55</td><td>0.75</td></tr><tr><td>12.5</td><td>0.45</td><td>0.65</td></tr><tr><td>15.0</td><td>0.40</td><td>0.60</td></tr><tr><td>17.5</td><td>0.35</td><td>0.55</td></tr></tbody></table></div>	Epoch	Train Loss	Val Loss	0.0	2.00	2.00	2.5	1.30	1.40	5.0	0.85	1.00	7.5	0.65	0.85	10.0	0.55	0.75	12.5	0.45	0.65	15.0	0.40	0.60	17.5	0.35	0.55
Epoch	Train Loss	Val Loss																										
0.0	2.00	2.00																										
2.5	1.30	1.40																										
5.0	0.85	1.00																										
7.5	0.65	0.85																										
10.0	0.55	0.75																										
12.5	0.45	0.65																										
15.0	0.40	0.60																										
17.5	0.35	0.55																										

64 Neuron, 4 Hidden Layer	
32 Neuron, 4 Hidden Layer	
<b>Perbandingan Akurasi</b>	
TC 1 Accuracy: 84.86%   Waktu: 11.90s TC 2 Accuracy: 74.72%   Waktu: 4.28s TC 3 Accuracy: 36.68%   Waktu: 3.61s	

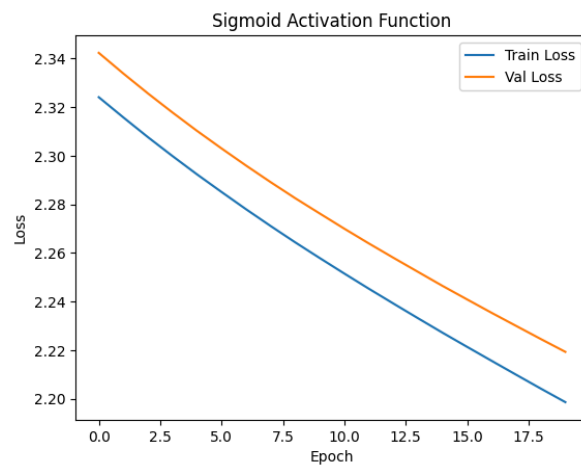
## 2.2 Pengaruh fungsi aktivasi

<b>Fungsi Aktivasi</b>	<b>Loss graph</b>
------------------------	-------------------

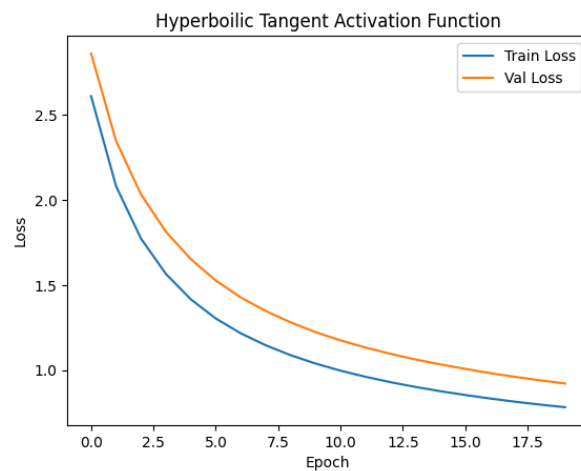
ReLU

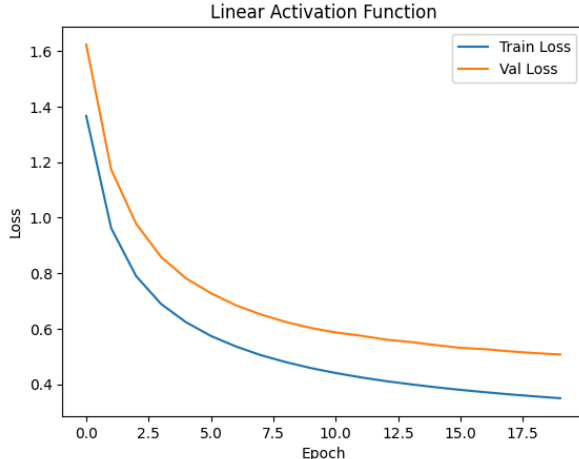


Sigmoid



Hyperbolic  
Tangent  
(tanh)

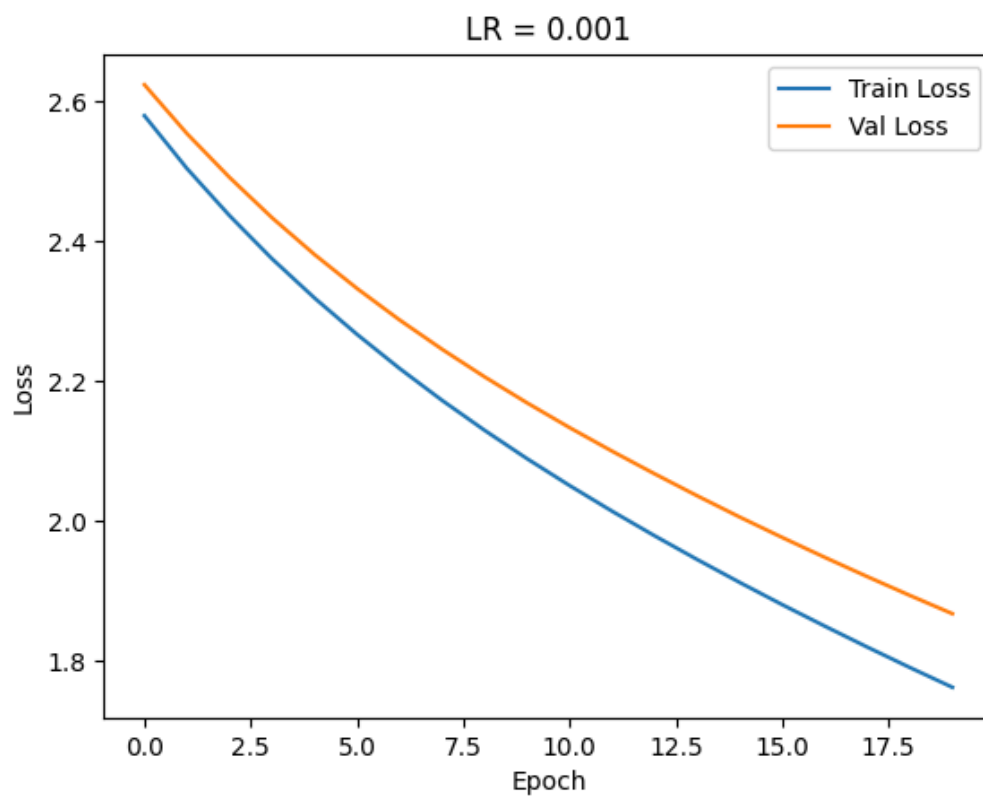


Linear	<div>Linear Activation Function</div>  <table><thead><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr></thead><tbody><tr><td>0.0</td><td>1.35</td><td>1.60</td></tr><tr><td>1.0</td><td>0.95</td><td>1.15</td></tr><tr><td>2.0</td><td>0.75</td><td>0.95</td></tr><tr><td>3.0</td><td>0.65</td><td>0.85</td></tr><tr><td>4.0</td><td>0.58</td><td>0.78</td></tr><tr><td>5.0</td><td>0.52</td><td>0.72</td></tr><tr><td>6.0</td><td>0.48</td><td>0.68</td></tr><tr><td>7.0</td><td>0.45</td><td>0.65</td></tr><tr><td>8.0</td><td>0.43</td><td>0.63</td></tr><tr><td>9.0</td><td>0.41</td><td>0.61</td></tr><tr><td>10.0</td><td>0.40</td><td>0.60</td></tr><tr><td>11.0</td><td>0.39</td><td>0.59</td></tr><tr><td>12.0</td><td>0.38</td><td>0.58</td></tr><tr><td>13.0</td><td>0.37</td><td>0.57</td></tr><tr><td>14.0</td><td>0.36</td><td>0.56</td></tr><tr><td>15.0</td><td>0.35</td><td>0.55</td></tr><tr><td>16.0</td><td>0.34</td><td>0.54</td></tr><tr><td>17.0</td><td>0.33</td><td>0.53</td></tr><tr><td>18.0</td><td>0.32</td><td>0.52</td></tr></tbody></table>	Epoch	Train Loss	Val Loss	0.0	1.35	1.60	1.0	0.95	1.15	2.0	0.75	0.95	3.0	0.65	0.85	4.0	0.58	0.78	5.0	0.52	0.72	6.0	0.48	0.68	7.0	0.45	0.65	8.0	0.43	0.63	9.0	0.41	0.61	10.0	0.40	0.60	11.0	0.39	0.59	12.0	0.38	0.58	13.0	0.37	0.57	14.0	0.36	0.56	15.0	0.35	0.55	16.0	0.34	0.54	17.0	0.33	0.53	18.0	0.32	0.52
Epoch	Train Loss	Val Loss																																																											
0.0	1.35	1.60																																																											
1.0	0.95	1.15																																																											
2.0	0.75	0.95																																																											
3.0	0.65	0.85																																																											
4.0	0.58	0.78																																																											
5.0	0.52	0.72																																																											
6.0	0.48	0.68																																																											
7.0	0.45	0.65																																																											
8.0	0.43	0.63																																																											
9.0	0.41	0.61																																																											
10.0	0.40	0.60																																																											
11.0	0.39	0.59																																																											
12.0	0.38	0.58																																																											
13.0	0.37	0.57																																																											
14.0	0.36	0.56																																																											
15.0	0.35	0.55																																																											
16.0	0.34	0.54																																																											
17.0	0.33	0.53																																																											
18.0	0.32	0.52																																																											
<b>Perbandingan Akurasi</b>																																																													
ReLU accuracy: 82.35%   Waktu: 7.19s																																																													
Sigmoid accuracy: 29.50%   Waktu: 5.54s																																																													
tanh accuracy: 75.48%   Waktu: 5.19s																																																													
Linear accuracy: 85.99%   Waktu: 5.00s																																																													

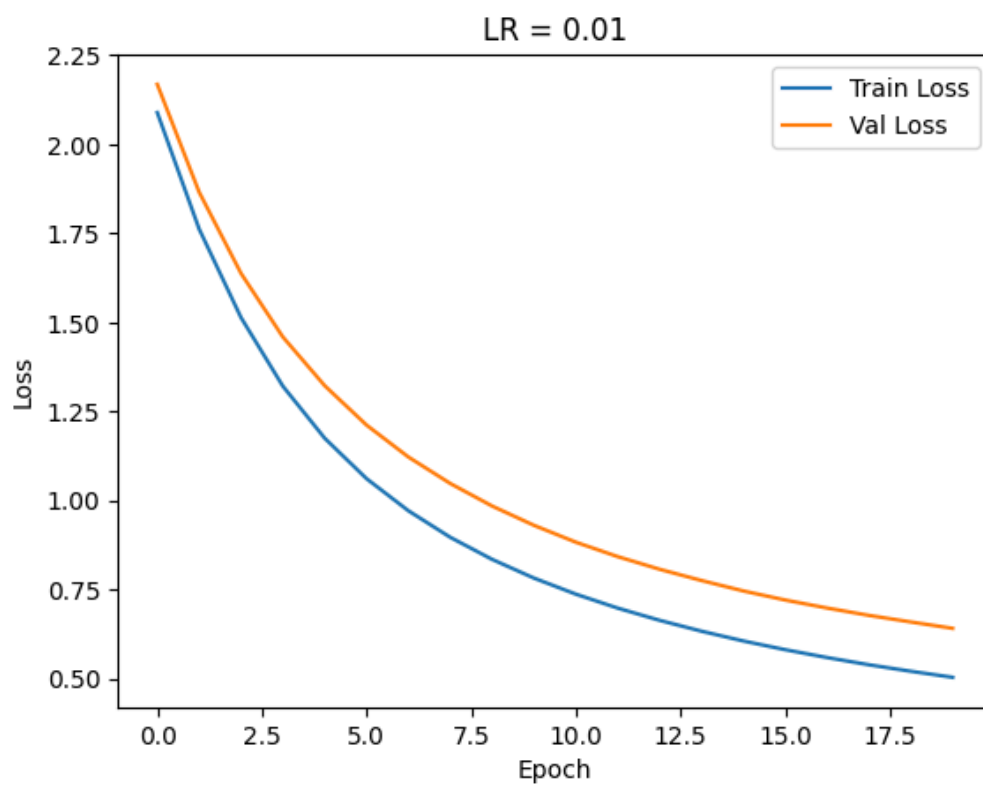
## 2.3 Pengaruh Learning rate

Learning Rate	Loss graph
---------------	------------

0.001



0.01



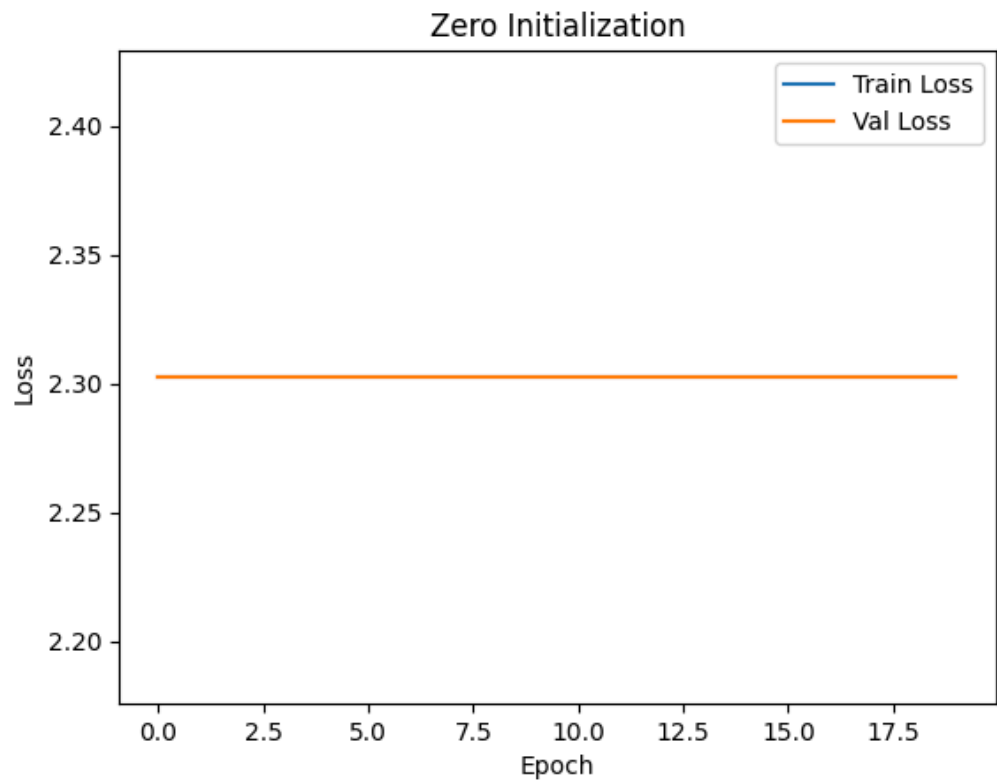


0.1	<div>LR = 0.1</div> <table><caption>Approximate data points from the Loss graph</caption><thead><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr></thead><tbody><tr><td>0.0</td><td>0.80</td><td>0.95</td></tr><tr><td>1.0</td><td>0.55</td><td>0.65</td></tr><tr><td>2.0</td><td>0.40</td><td>0.55</td></tr><tr><td>3.0</td><td>0.35</td><td>0.50</td></tr><tr><td>4.0</td><td>0.30</td><td>0.48</td></tr><tr><td>5.0</td><td>0.25</td><td>0.45</td></tr><tr><td>6.0</td><td>0.23</td><td>0.43</td></tr><tr><td>7.0</td><td>0.20</td><td>0.41</td></tr><tr><td>8.0</td><td>0.18</td><td>0.40</td></tr><tr><td>9.0</td><td>0.16</td><td>0.39</td></tr><tr><td>10.0</td><td>0.15</td><td>0.38</td></tr><tr><td>11.0</td><td>0.14</td><td>0.38</td></tr><tr><td>12.0</td><td>0.13</td><td>0.37</td></tr><tr><td>13.0</td><td>0.12</td><td>0.38</td></tr><tr><td>14.0</td><td>0.11</td><td>0.37</td></tr><tr><td>15.0</td><td>0.10</td><td>0.37</td></tr><tr><td>16.0</td><td>0.09</td><td>0.37</td></tr><tr><td>17.0</td><td>0.08</td><td>0.37</td></tr><tr><td>18.0</td><td>0.07</td><td>0.37</td></tr></tbody></table>	Epoch	Train Loss	Val Loss	0.0	0.80	0.95	1.0	0.55	0.65	2.0	0.40	0.55	3.0	0.35	0.50	4.0	0.30	0.48	5.0	0.25	0.45	6.0	0.23	0.43	7.0	0.20	0.41	8.0	0.18	0.40	9.0	0.16	0.39	10.0	0.15	0.38	11.0	0.14	0.38	12.0	0.13	0.37	13.0	0.12	0.38	14.0	0.11	0.37	15.0	0.10	0.37	16.0	0.09	0.37	17.0	0.08	0.37	18.0	0.07	0.37
Epoch	Train Loss	Val Loss																																																											
0.0	0.80	0.95																																																											
1.0	0.55	0.65																																																											
2.0	0.40	0.55																																																											
3.0	0.35	0.50																																																											
4.0	0.30	0.48																																																											
5.0	0.25	0.45																																																											
6.0	0.23	0.43																																																											
7.0	0.20	0.41																																																											
8.0	0.18	0.40																																																											
9.0	0.16	0.39																																																											
10.0	0.15	0.38																																																											
11.0	0.14	0.38																																																											
12.0	0.13	0.37																																																											
13.0	0.12	0.38																																																											
14.0	0.11	0.37																																																											
15.0	0.10	0.37																																																											
16.0	0.09	0.37																																																											
17.0	0.08	0.37																																																											
18.0	0.07	0.37																																																											
<b>Perbandingan Akurasi</b>																																																													
TC 1 Accuracy: 42.46%   Waktu: 6.75s																																																													
TC 2 Accuracy: 82.35%   Waktu: 6.30s																																																													
TC 3 Accuracy: 90.87%   Waktu: 6.13s																																																													

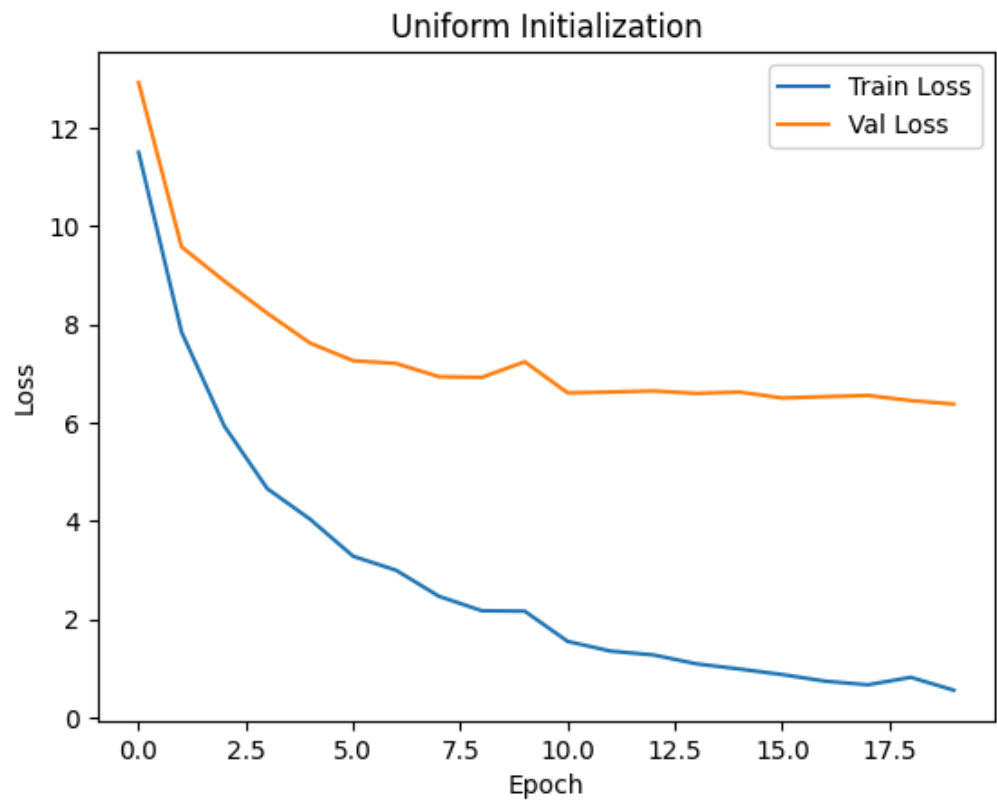
## 2.4 Pengaruh Inisialisasi bobot

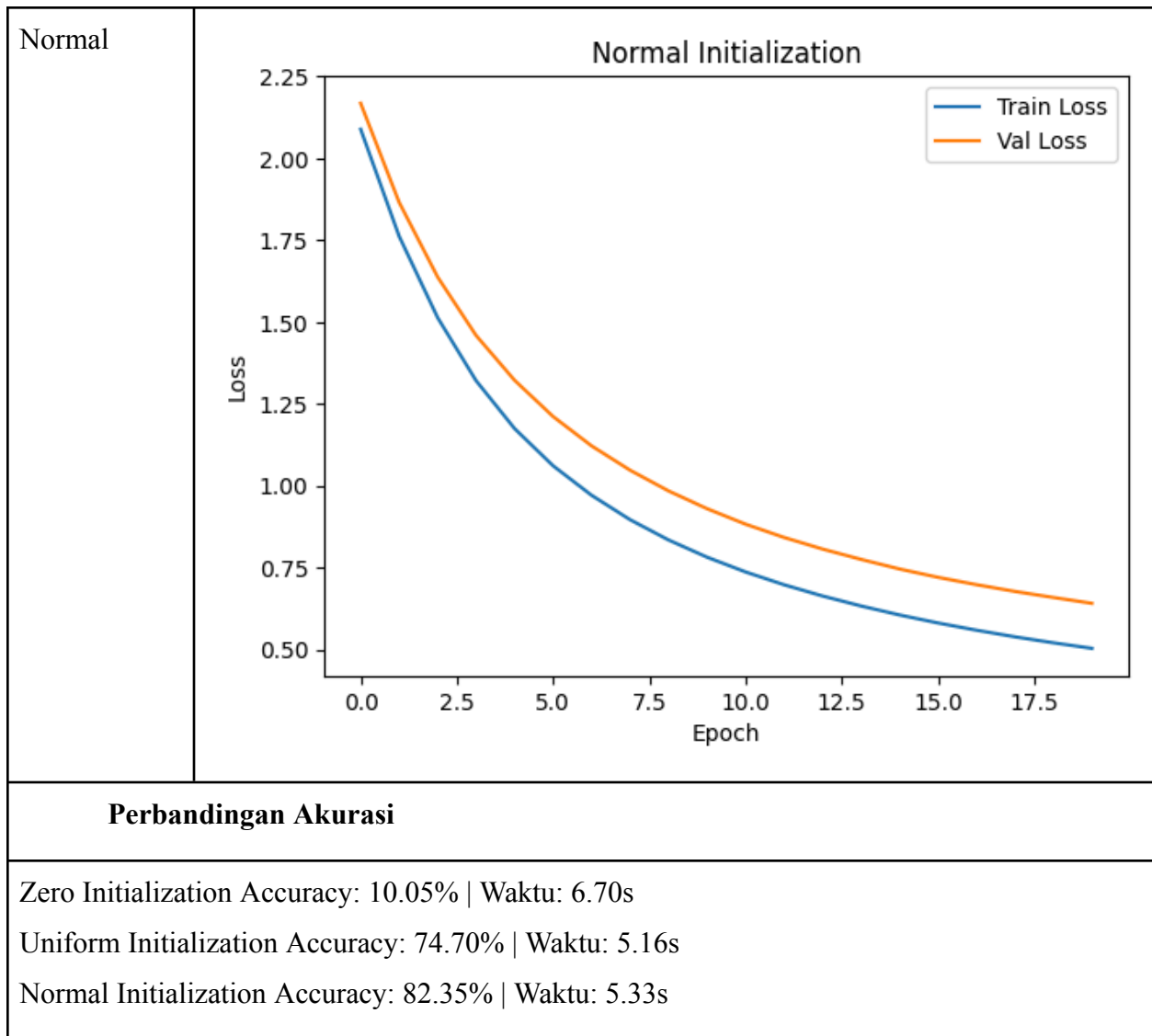
<b>Initialization Method</b>	<b>Loss graph</b>
------------------------------	-------------------

Zero  
Inizialitazion



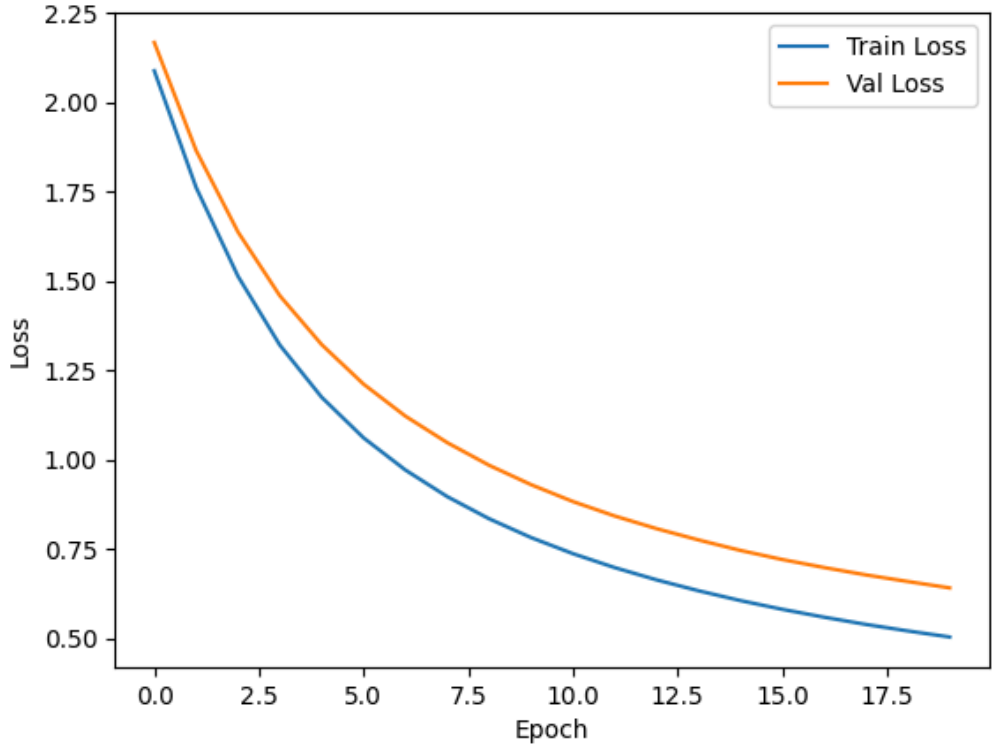
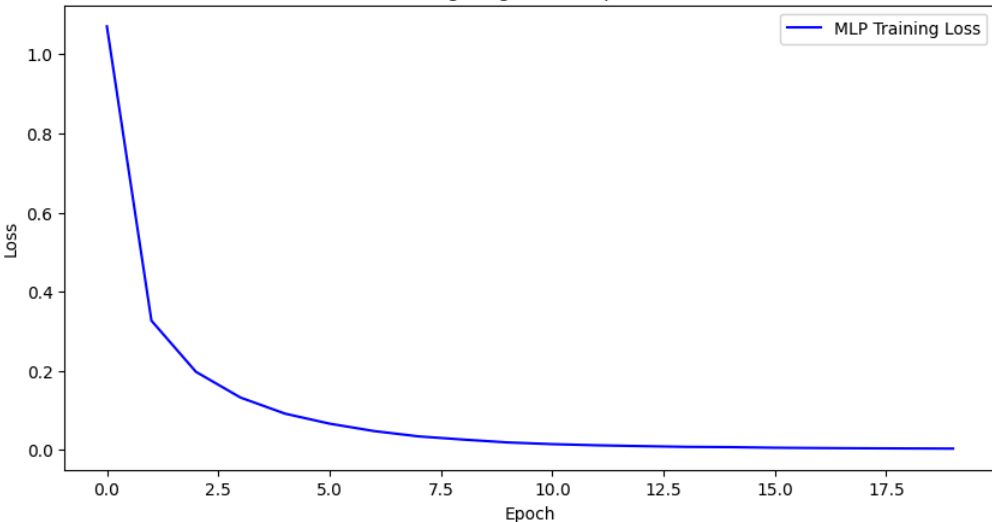
Uniform



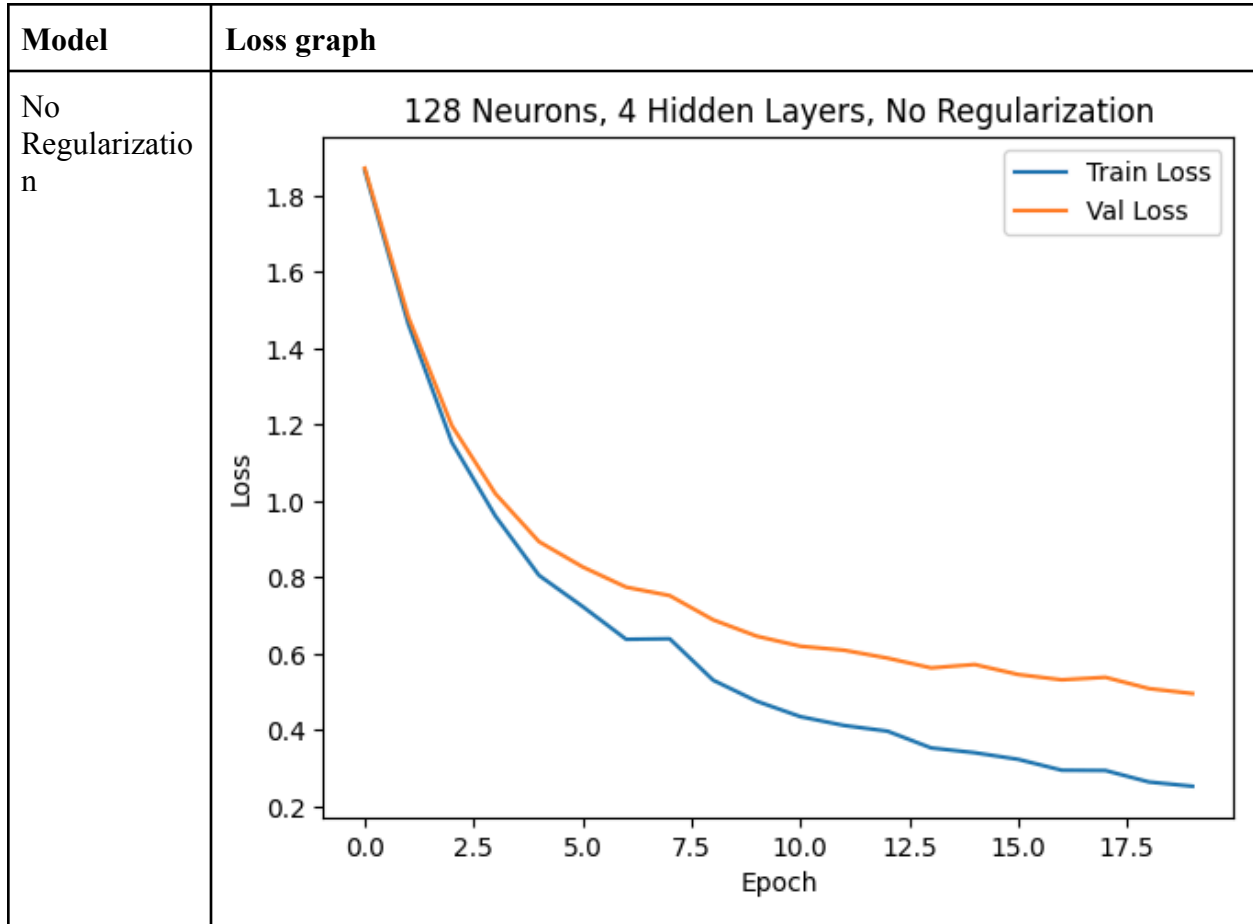


## 2.5 Perbandingan dengan library sklearn

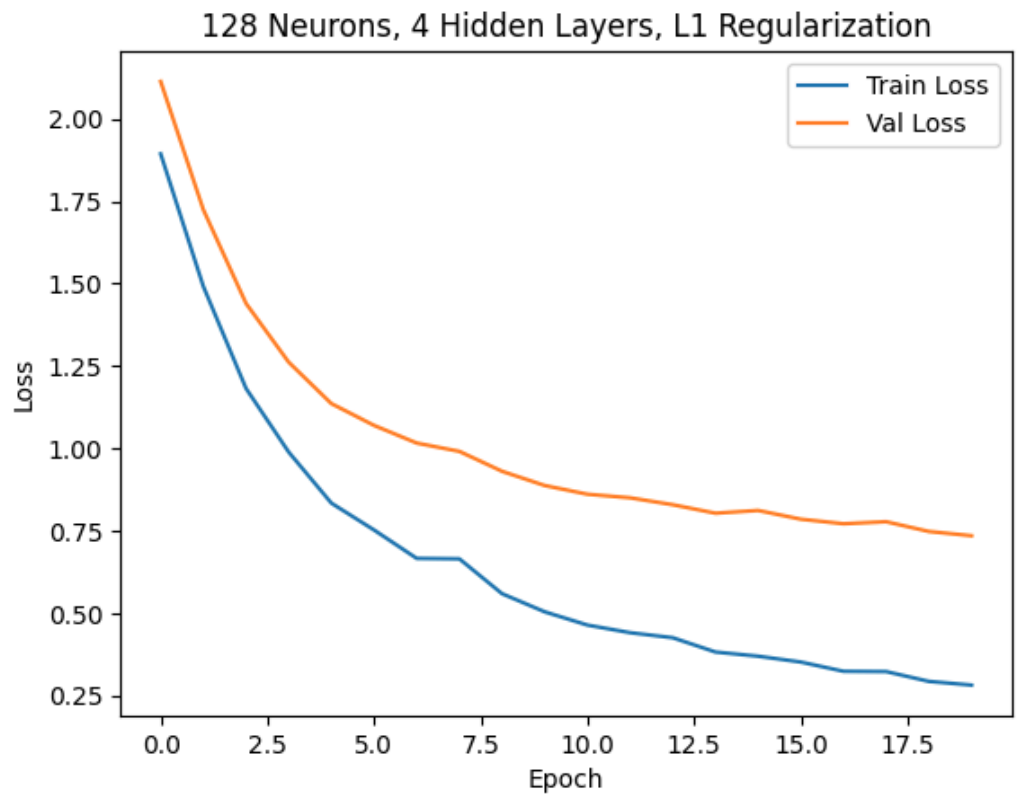
Model	Loss graph
-------	------------

Scratch	 <p>The graph displays the training and validation loss for a model trained from scratch. The x-axis represents the number of epochs (0.0 to 18.0), and the y-axis represents the loss (0.50 to 2.25). The blue line represents the Train Loss, and the orange line represents the Val Loss. Both losses show a steady decrease over time, with the Train Loss consistently lower than the Val Loss after the initial epochs.</p> <table><tr><th>Epoch</th><th>Train Loss</th><th>Val Loss</th></tr><tr><td>0.0</td><td>2.10</td><td>2.20</td></tr><tr><td>2.5</td><td>1.40</td><td>1.60</td></tr><tr><td>5.0</td><td>1.10</td><td>1.25</td></tr><tr><td>7.5</td><td>0.90</td><td>1.05</td></tr><tr><td>10.0</td><td>0.75</td><td>0.90</td></tr><tr><td>12.5</td><td>0.65</td><td>0.80</td></tr><tr><td>15.0</td><td>0.58</td><td>0.72</td></tr><tr><td>17.5</td><td>0.52</td><td>0.65</td></tr></table>	Epoch	Train Loss	Val Loss	0.0	2.10	2.20	2.5	1.40	1.60	5.0	1.10	1.25	7.5	0.90	1.05	10.0	0.75	0.90	12.5	0.65	0.80	15.0	0.58	0.72	17.5	0.52	0.65
Epoch	Train Loss	Val Loss																										
0.0	2.10	2.20																										
2.5	1.40	1.60																										
5.0	1.10	1.25																										
7.5	0.90	1.05																										
10.0	0.75	0.90																										
12.5	0.65	0.80																										
15.0	0.58	0.72																										
17.5	0.52	0.65																										
sklearn	 <p>The graph, titled 'Training Progress Comparison', shows the MLP Training Loss over 18 epochs. The x-axis represents the number of epochs (0.0 to 18.0), and the y-axis represents the loss (0.0 to 1.0). The blue line represents the MLP Training Loss, which decreases rapidly from 1.0 at epoch 0.0 to near 0.0 by epoch 10.0, remaining stable thereafter.</p> <table><tr><th>Epoch</th><th>MLP Training Loss</th></tr><tr><td>0.0</td><td>1.05</td></tr><tr><td>2.5</td><td>0.20</td></tr><tr><td>5.0</td><td>0.10</td></tr><tr><td>7.5</td><td>0.05</td></tr><tr><td>10.0</td><td>0.02</td></tr><tr><td>12.5</td><td>0.01</td></tr><tr><td>15.0</td><td>0.01</td></tr><tr><td>17.5</td><td>0.01</td></tr></table>	Epoch	MLP Training Loss	0.0	1.05	2.5	0.20	5.0	0.10	7.5	0.05	10.0	0.02	12.5	0.01	15.0	0.01	17.5	0.01									
Epoch	MLP Training Loss																											
0.0	1.05																											
2.5	0.20																											
5.0	0.10																											
7.5	0.05																											
10.0	0.02																											
12.5	0.01																											
15.0	0.01																											
17.5	0.01																											
Perbandingan Akurasi																												
FFNN Test Accuracy: 82.35%   Waktu: 5.69s																												
MLP Test Accuracy: 93.09%   Waktu: 3.44s																												

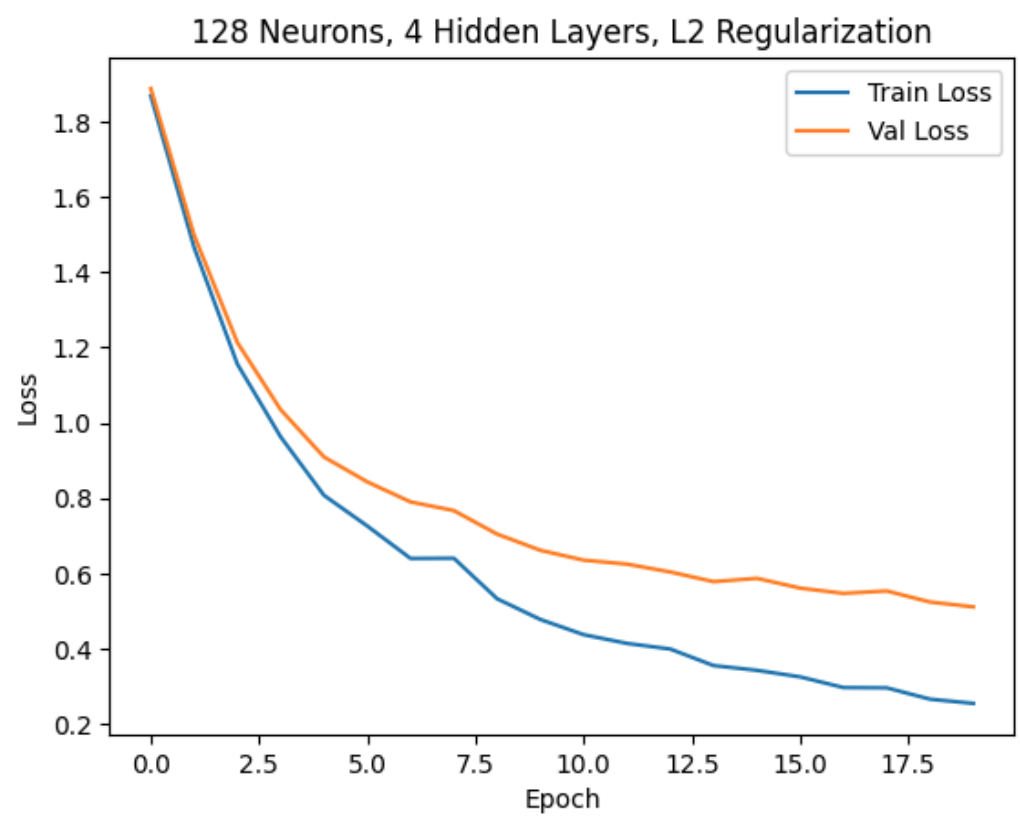
## 2.6. Perbandingan regularisasi



L1



L2



Perbandingan Akurasi
===== Hasil Perbandingan ===== No Regularization Accuracy: 85.84%   Waktu: 6.56s L1: 85.91%   Waktu: 5.43s L2: Accuracy: 85.86%   Waktu: 4.82s =====

## KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian yang kami lakukan

Depth vs. Width:

Lebar jaringan (width) lebih kritis untuk akurasi daripada kedalaman (depth).

Trade-off: Lebih banyak neuron  $\rightarrow$  akurasi  $\uparrow$  tetapi waktu training  $\uparrow$ .

Fungsi Aktivasi:

Aktivasi ReLU direkomendasikan untuk hidden layer.

Hasil tinggi pada aktivasi Linear merupakan anomali

Learning Rate:

Learning rate 0.1 bekerja baik untuk kasus ini,

Inisialisasi:

Inisialisasi normal lebih baik daripada uniform/zero.

Untuk ReLU, gunakan He initialization untuk hasil optimal.

Scratch vs. Library:

Model library (MLP) lebih unggul karena optimisasi otomatis).

Non Reg, L1, L2:

L1 sedikit lebih unggul dalam akurasi, tetapi perbedaannya tidak signifikan

L2 memiliki waktu training paling cepat


Ketiga metode menghasilkan akurasi yang sangat mirip (~85.8%)



## PEMBAGIAN TUGAS

NIM	Tugas
13522123	FFNN, docs
13522151	FFNN, docs
13522152	FFNN, docs

## REFERENSI

-  The spelled-out intro to neural networks and backpropagation: building micrograd
- <https://www.jasonosajima.com/forwardprop>
- <https://www.jasonosajima.com/backprop>
- [https://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html)
- [https://math.libretexts.org/Bookshelves/Calculus/Calculus\\_\(OpenStax\)/14%3A\\_Differentiation\\_of\\_Functions\\_of\\_Several\\_Variables/14.05%3A\\_The\\_Chain\\_Rule\\_for\\_Multivariable\\_Functions](https://math.libretexts.org/Bookshelves/Calculus/Calculus_(OpenStax)/14%3A_Differentiation_of_Functions_of_Several_Variables/14.05%3A_The_Chain_Rule_for_Multivariable_Functions)

## LAMPIRAN

Pranala GitHub: <https://github.com/jimlynurarif/whitebox>