

# **Tugas Besar 1 IF3270 Pembelajaran Mesin**

## **Feedforward Neural Network**



Anggota Kelompok:

Ahmad Farid Mudrika 13522008

Akbar Al-Fattah 13522036

Muhamad Rifki Virziadeili Harisman 13522120

# 1. Deskripsi Persoalan

## 2. Pembahasan

### a. Penjelasan implementasi

#### i. Deskripsi kelas beserta deskripsi atribut dan methodnya

##### a. Kelas ActivationFunction

Kelas ini berisi metode statik untuk fungsi aktivasi dan turunannya. Misalnya, metode linear digunakan untuk menghitung hasil fungsi aktivasi linear, sedangkan metode linear\_derivative digunakan untuk menghitung turunan fungsi aktivasi linear.

##### b. Kelas LossFunction

Kelas ini berisi metode statik untuk fungsi loss dan turunannya. Misalnya, metode mse digunakan untuk menghitung hasil fungsi loss MSE, sedangkan metode mse\_derivative digunakan untuk menghitung turunan fungsi Loss MSE.

##### c. Kelas Neuron

Kelas yang merepresentasikan sebuah neuron dalam jaringan.

Atribut:

id: Identifier unik neuron. Identifier ini unik di keseluruhan jaringan neural.

output: Nilai output neuron

grad: Nilai gradien neuron

Method:

set\_output: Mengatur nilai output

get\_output: Mendapatkan nilai output

##### d. Kelas Layer

Atribut:

id: Identifier unik layer

neurons: Daftar neuron dalam layer

activation: Fungsi aktivasi layer dalam bentuk string.

outputs: Output dari layer

Method:

activate: Mengaplikasikan fungsi aktivasi pada nilai input.

##### e. Kelas FFNN

Kelas ini merepresentasikan jaringan saraf feed-forward.

Atribut:

N\_layer: Jumlah layer dalam jaringan

loss: Fungsi loss yang digunakan ('mse' atau 'cross\_entropy')

activation: Daftar fungsi aktivasi untuk setiap layer

N\_neuron\_layer: Jumlah neuron di setiap layer  
weight\_method: Metode inisialisasi bobot ('normal', 'uniform', atau 'zero')  
neurons: Dictionary yang menyimpan semua neuron dengan ID sebagai kunci  
layers: List layer dalam jaringan  
weights: Dictionary yang menyimpan bobot koneksi antar neuron. Key dictionary ini adalah tuple id neuron dan id neuron layer berikutnya yang terhubung.  
biases: Dictionary yang menyimpan bias untuk setiap neuron.

Method:

\_\_init\_\_: Inisialisasi jaringan neural  
initialize\_weights: Menginisialisasi bobot dan bias dengan metode tertentu yang dipilih.  
forward: Melakukan forward propagation  
backward: Melakukan backpropagation  
\_get\_activation\_derivative: Mendapatkan turunan fungsi aktivasi  
fit: Melatih model dengan data input  
predict: Melakukan prediksi dengan model yang sudah dilatih  
save: Menyimpan model ke file  
load: Memuat model dari file (static method)  
visualize\_network: Visualisasi seluruh jaringan neural  
visualize\_selected\_layers: Visualisasi layer tertentu dari jaringan

## ii. Penjelasan forward propagation

Pada implementasi kami, metode Forward mengimplementasikan *forward propagation* pada FFNN untuk input batch data. Berikut penjelasan langkahnya:

### 1. Validasi Input

Pertama dilakukan pengecekan dimensi input (X.shape[1]) harus sama dengan jumlah neuron di layer input jaringan (self.N\_neuron\_layer[0]), memastikan kompatibilitas arsitektur jaringan dengan data. Nilai input lalu dialokasikan ke input layer

### 2. Iterasi Layer

Untuk setiap layer, dilakukan perhitungan:

- Mengambil output dari layer sebelumnya (selain layer input), dan mengkonversi ke bentuk matrix.

- Membuat matriks `weighted_sum` yang berukuran jumlah sampel x jumlah neuron di layer saat ini.

### 3. Perhitungan Weighted Sum

Untuk tiap neuron di layer saat ini, dihitung weighted sum dengan mengalikan sampel/output dengan bobot tiap koneksi. Berikutnya, ditambahkan nilai bias tiap neuron ke `weighted_sum`.

### 4. Pengaplikasian Fungsi Aktivasi

Hasil `weighted_sum` diaktivasi dengan fungsi aktivasi. Output akan disimpan di tiap neuron dan layer.

### 5. Return Output

Mengembalikan output dari layer terakhir sebagai hasil prediksi jaringan.

## iii. Penjelasan backward propagation dan weight update

Kode ini mengimplementasikan backpropagation untuk memperbarui bobot dan bias jaringan berdasarkan error prediksi. Berikut alur kerjanya:

### 1. Forward Pass Awal

Memanggil `forward(X)` untuk mendapatkan prediksi jaringan (`y_pred`).

Menghitung turunan fungsi loss terhadap output jaringan (`d_loss`).

### 2. Inisialisasi Delta

Membuat array deltas untuk menyimpan gradien error tiap layer.


### 3. Backward Propagation

Propagasi dilakukan iteratif dari layer terakhir sebelum output. Untuk tiap layer, program membangun matriks bobot yang menghubungkan layer tersebut dengan layer berikutnya. Error dari layer berikutnya dipropagasi ke layer saat ini dan dikalikan dengan turunan fungsi aktivasi.

### 4. Update Parameter

Bobot antar neuron diperbarui menggunakan learning rate dikalikan dengan hasil perkalian antara output layer sebelumnya dengan delta layer berikutnya. Pembaruan bias dilakukan dengan mengambil rata-rata delta pada layer berikutnya dan mengalikannya dengan learning rate.

## b. Hasil pengujian

Karena besarnya ukuran jaringan, bobot dan gradien untuk semua pengujian disimpan di file txt di folder berikut  `Data`.

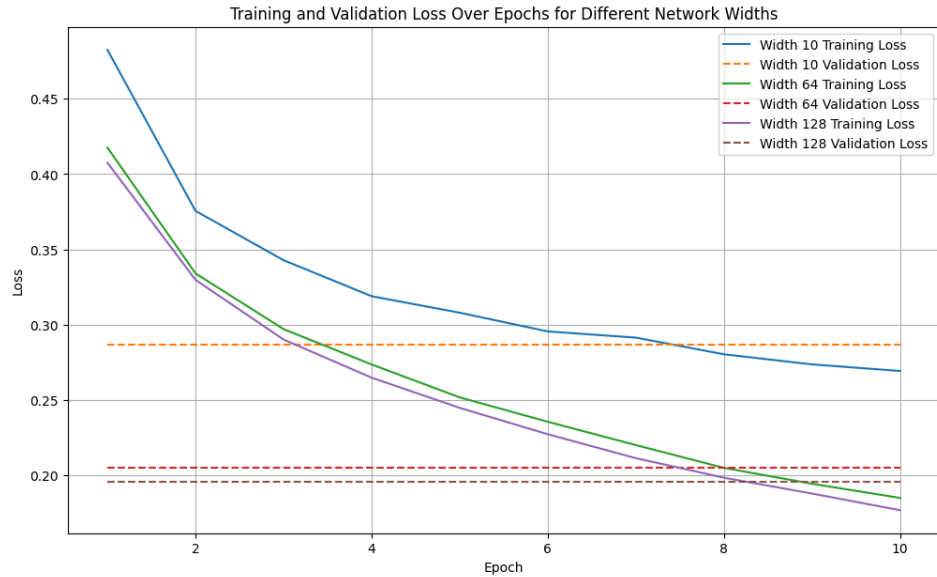
### i. Pengaruh depth dan width

Kasus pertama adalah pengaruh width. Kami menggunakan hyperparameter sebagai berikut:

`width_variations = [[784, 10, 10], [784, 64, 10], [784, 128, 10]]`

```
activation_width = [['relu', 'softmax']] * 3
epochs = 10
```

Dari hasil pengujian kami mendapatkan grafik loss per epoch sebagai berikut.



Proses validasi menghasilkan akurasi sebagai berikut

Validation Accuracies:

Width 10: 0.9199

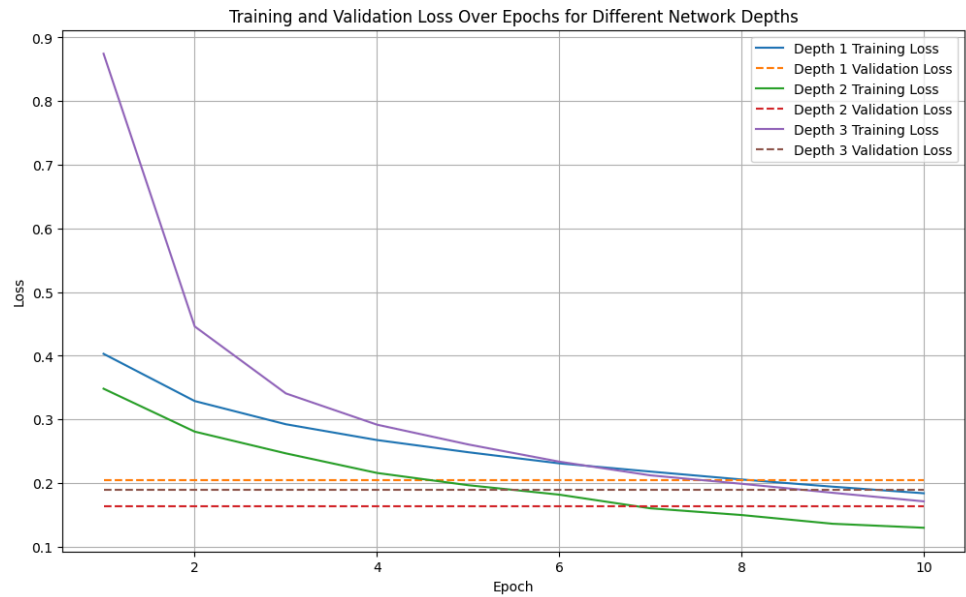
Width 64: 0.9436

Width 128: 0.9470

Untuk kasus kedua yaitu pengaruh depth. Kami menggunakan hyperparameter sebagai berikut,

```
depth_variations = [
    [784, 64, 10],
    [784, 64, 64, 10],
    [784, 64, 64, 64, 10]
]
activation_depth = [
    ['relu', 'softmax'],
    ['relu', 'relu', 'softmax'],
    ['relu', 'relu', 'sigmoid', 'softmax']
]
epochs = 10
```

Dari hasil pengujian kami mendapatkan grafik loss sebagai berikut,



Proses validasi yang dilakukan menghasilkan akurasi sebagai berikut,

Validation Accuracies:

Depth 1: 0.9436

Depth 2: 0.9557

Depth 3: 0.9448

## ii. Pengaruh fungsi aktivasi

Kasus berikutnya adalah pengaruh

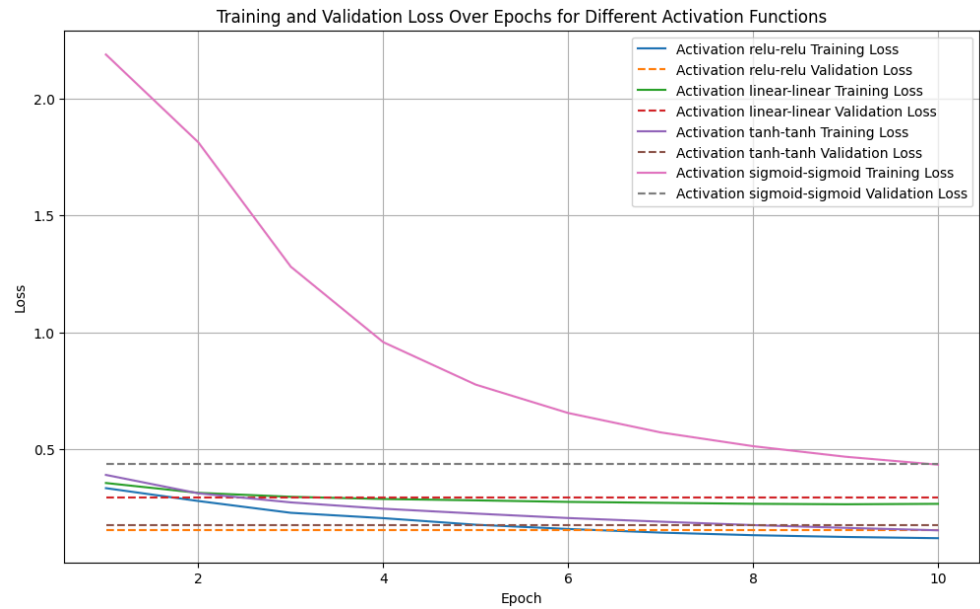
```
activation_functions = [
    ['relu', 'relu', 'softmax'],
    ['linear', 'linear', 'softmax'],
    ['tanh', 'tanh', 'softmax'],
    ['sigmoid', 'sigmoid', 'softmax']
]
```

```
epochs = 10
```

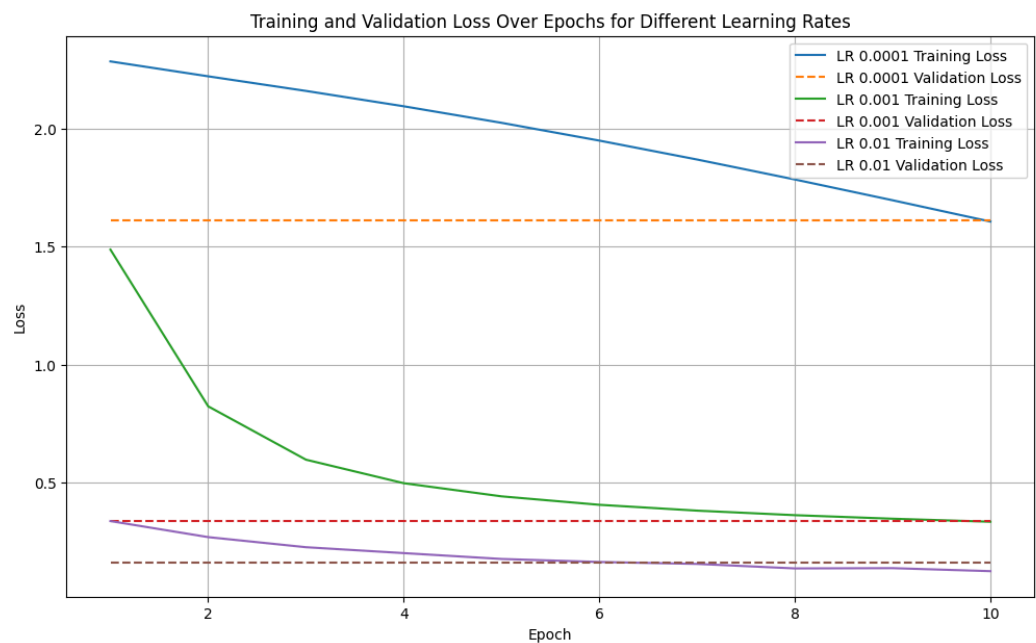
```
layer_sizes = [784, 128, 64, 10]
```

```
loss='cce',
```

```
weight_method='normal'
```



iii. Pengaruh learning rate  
learning\_rates = [0.0001, 0.001, 0.01]  
epochs = 10  
layer\_sizes = [784, 64, 64, 10]  
activations = ['relu', 'relu', 'softmax']  
loss='cce',  
weight\_method='normal'



Validation Accuracies:

LR 0.0001: 0.6244

LR 0.001: 0.9083

LR 0.01: 0.9543

iv. Pengaruh inisialisasi bobot

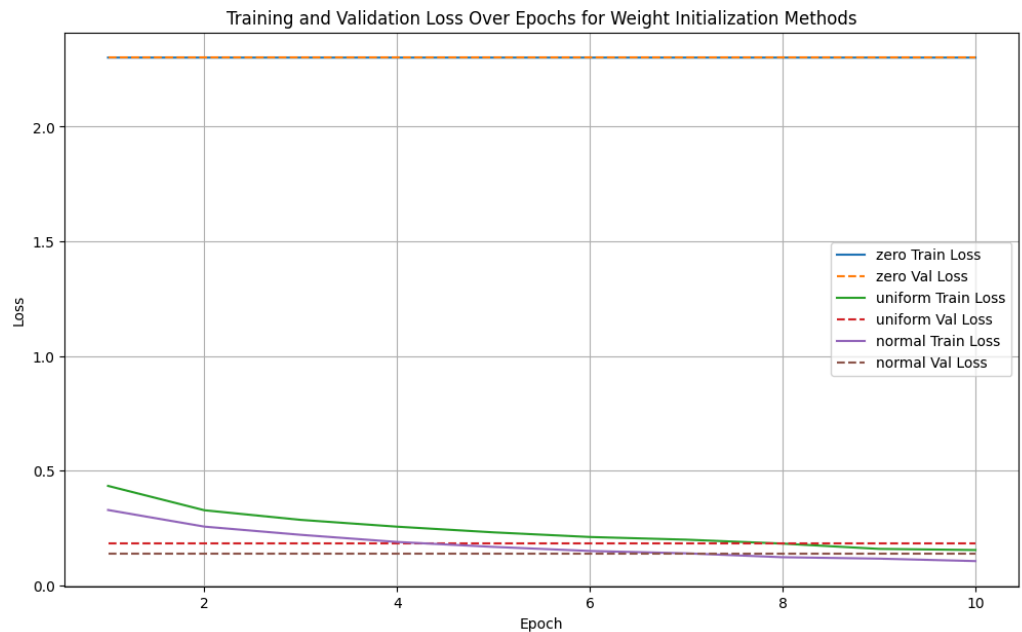
weight\_init\_methods = ['zero', 'uniform', 'normal']

epochs = 10

layer\_sizes=[784, 128, 64, 10],

activations=['relu', 'relu', 'softmax'],

loss='cce',



Final Validation Accuracies (per weight init):

zero: 0.1143

uniform: 0.9486

normal: 0.9604

v. Pengaruh regularisasi

reguralizations = ['L1', 'L2']

epochs = 10

layer\_sizes=[784, 128, 64, 10],

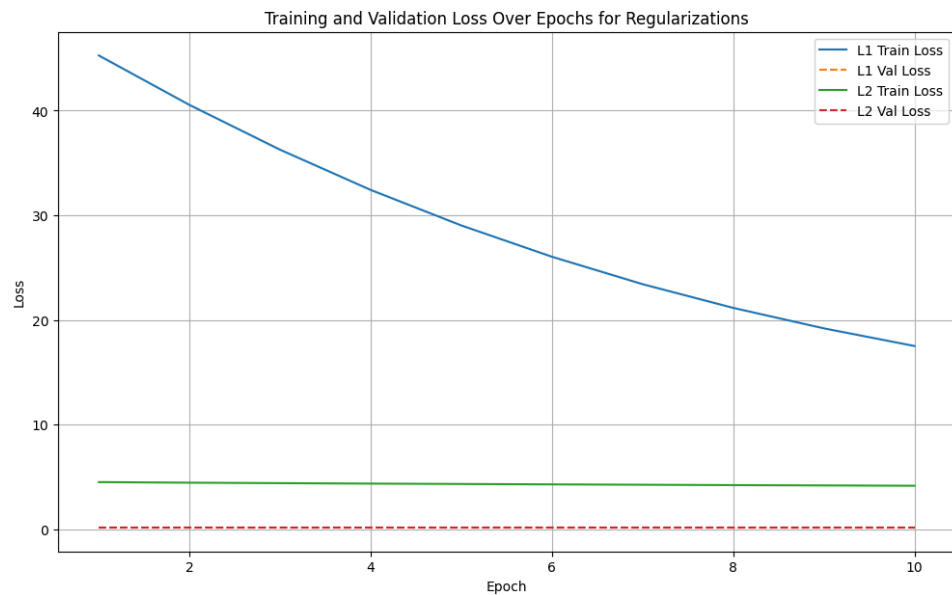
activations=['relu', 'relu', 'softmax'],

loss='cce',

weight\_method="normal",



lambda\_=0.01



Final Validation Accuracies (per Regularizations):

L1: 0.9486

L2: 0.9579

vi. Perbandingan dengan library sklearn

---

Validation Accuracy (sklearn MLPClassifier): 0.9648

Validation Accuracy (Custom FFNN): 0.9579

### 3. Kesimpulan dan Saran

Kesimpulan yang dapat diambil dari tugas besar ini adalah performa FFNN (Feed Forward Neural Network) baik dari sisi akurasi maupun kecepatan eksekusi sangat dipengaruhi oleh pemilihan dan urutan fungsi aktivasi yang tepat, pemilihan ukuran width dan depth yang tepat, pemilihan besarnya learning rate yang sesuai, pemilihan metode inisialisasi bobot yang tepat, dan pemilihan tipe regularisasi yang tepat. Dari hasil eksplorasi, kami mendapatkan bahwa konfigurasi FFNN dengan akurasi terbaik sejauh penulisan laporan ini untuk dataset MNIST adalah:

width = 128,

urutan fungsi activation=['relu', 'relu', 'softmax']

depth = 2,

layer\_sizes=[784, 128, 64, 10]

learning rate = 0.01,

metode inisialisasi bobot = normal,

regularisasi = L2,

epoch=10,

fungsi loss = Categorical Cross Entropy,

lambda regularisasi = 0.01

yang menghasilkan akurasi sebesar 0.9579.

Dibandingkan dengan akurasi dari MLPClassifier milik sklearn yang memiliki akurasi sebesar 0.9648, dapat disimpulkan bahwa model kami dengan konfigurasi model seperti di atas memiliki akurasi yang sangat baik dengan selisih akurasi sebesar 0,0069.

Saran untuk peningkatan akurasi model kami lebih lanjut adalah kami seharusnya melakukan lebih banyak eksperimen dan eksplorasi lebih lanjut untuk menentukan konfigurasi FFNN yang lebih baik dari konfigurasi yang dituliskan pada laporan ini.

#### **4. Pembagian Tugas**

Rifki : Struktur data, fungsi aktivasi, pengujian di ipynb, laporan.

Farid : Backward propagation, save, load, visualisasi, laporan.

Akbar : Fungsi Loss, fungsi aktivasi, forward propagation, laporan.

#### **5. Referensi**

Forward propagation in training neural networks step by step-

[https://www.youtube.com/watch?v=sNTtUV9yE\\_M](https://www.youtube.com/watch?v=sNTtUV9yE_M)

Back Propagation in training neural networks step by step -

<https://www.youtube.com/watch?v=-zI1bldB8to>

Modul FFNN di Edunex

[https://edunexcontentprodhot.blob.core.windows.net/edunex/nullfile/1741577891298\\_IF3270-Mgg03-FFNN-print?sv=2024-11-04&spr=https&st=2025-03-10T03%3A32%3A33Z&se=2027-03-10T03%3A32%3A33Z&sr=b&sp=r&sig=KdLqCzzo5Z3lQwFtxt%2BEoNGBpLkPPB2YMdYxbAAqYzQ%3D&rsct=application%2Fpdf](https://edunexcontentprodhot.blob.core.windows.net/edunex/nullfile/1741577891298_IF3270-Mgg03-FFNN-print?sv=2024-11-04&spr=https&st=2025-03-10T03%3A32%3A33Z&se=2027-03-10T03%3A32%3A33Z&sr=b&sp=r&sig=KdLqCzzo5Z3lQwFtxt%2BEoNGBpLkPPB2YMdYxbAAqYzQ%3D&rsct=application%2Fpdf)