

# **LAPORAN PRAKTIKUM**

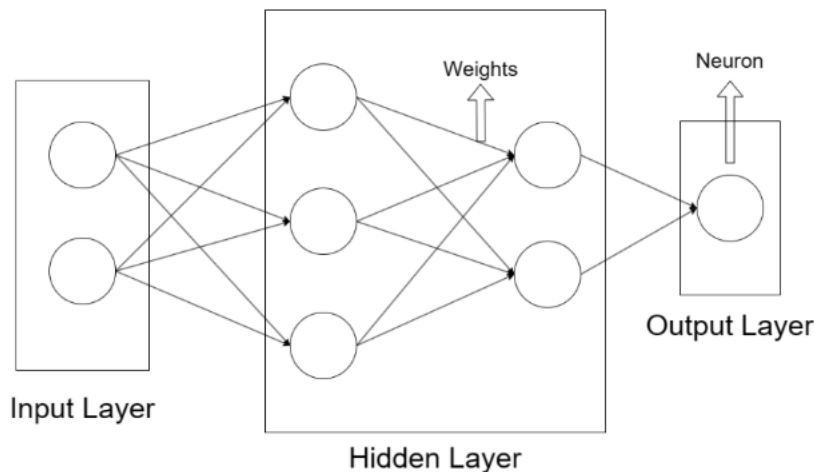
## **DEEP LEARNING**



**NAMA : DESTY MAYANG PRATIWI**  
**NIM : 21110016**

**LABORATORIUM DATA**  
**FAKULTAS INFORMATIKA**  
**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**  
**2023**

# PEMBUATAN ENVIRONMENT DAN PENGENALAN ANN



## A. Pembuatan Environment

- Environment adalah tempat / lingkungan dimana library - library python yang akan digunakan diinstal.
- Env bisa dibuat melalui command prompt atau bisa juga di halaman anaconda.

### 1. Via command prompt

Hal pertama yang perlu dilakukan adalah membuka anaconda prompt di masing - masing device. Setelah itu tuliskan: `conda create --name [Nama Env]`

- a. Langkah Pertama

```
(base) C:\Users\HP>conda create --name DLEnv
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

- b. Setelah itu pilih "y"

```

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate DLenv
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done

```

- c. Aktivasi Environment Sebelum Instal

```

(base) C:\Users\HP>activate DLenv

(DLenv) C:\Users\HP>|

```

- d. Kalau sudah tidak menggunakannya lagi, maka perlu di deactivate dengan cara berikut:

```

(DLenv) C:\Users\HP>conda deactivate

```

- e. Untuk menghapus environment Maka seluruh library yang terinstal dan environment yang dibuat sudah terhapus dan tidak bisa digunakan kembali.

```

(base) C:\Users\HP>conda env remove --name DLenv

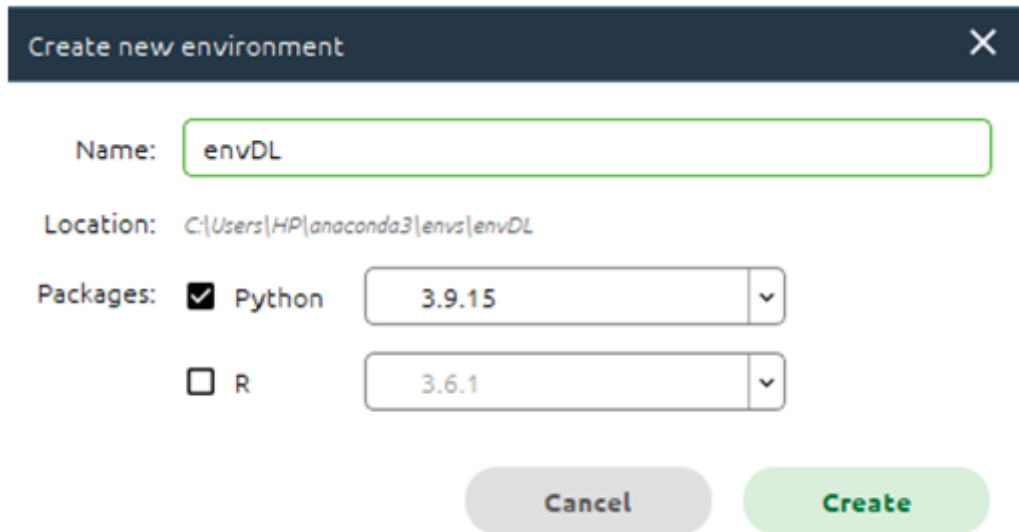
Remove all packages in environment C:\Users\HP\anaconda3\envs\DLenv:

```

## 2. Via halaman Anaconda a. Tekan Tombol Create



b. Lalu Tekan create



Create new environment

Name: envDL

Location: C:\Users\HP\anaconda3\envs\envDL

Packages: ☒ Python 3.9.15 ☐ R 3.6.1

Cancel Create

### Environment Terbentuk

Setelah environment dibuat, maka dilakukan instalasi library - library yang akan digunakan seperti Pandas, Numpy, Matplotlib, Seaborn, Tensorflow, Flask, dan lainnya sesuai kebutuhan.

### Instalasi Library di Environment Baru

- pip install --upgrade pip
- pip install pandas
- pip install numpy
- pip install tensorflow
- pip install matplotlib

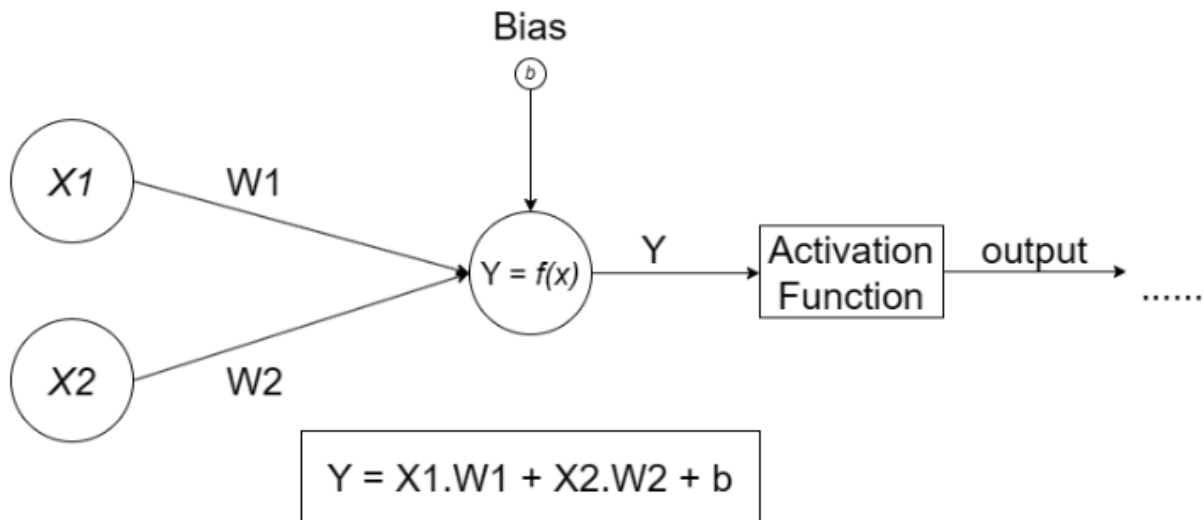
### Pengenalan ANN

Jaringan saraf tiruan, juga dikenal sebagai artificial neural networks, adalah model komputasi yang didasarkan pada struktur dan fungsi otak manusia. Ini terdiri dari neuron, unit kecil yang saling terhubung dengan bobot yang dapat diubah.

Neural network digunakan untuk memproses data, melakukan tugas seperti pengenalan pola, klasifikasi, dan regresi, serta mempelajari pola-pola ini dari data pelatihan.

Lapisan-lapisan neuron memungkinkan neural network untuk mempelajari representasi fitur data yang kompleks. Ini adalah salah satu teknik penting dalam deep learning, yang menggunakan artificial neural networks yang dalam dan kompleks untuk menyelesaikan masalah yang kompleks.

Perhitungan yang terjadi di dalam ANN



## Komponen Utama Artificial Neural Networks

### 1.) Neuron

- Unit dasar dalam ANN yang menerima input, melakukan perhitungan, dan menghasilkan output.
- Input dikalikan dengan bobot yang sesuai dan dijumlahkan.
- Hasilnya kemudian diteruskan melalui fungsi aktivasi.

### 2.) Layer (Lapisan)

- Neuron dikelompokkan dalam 3 lapisan yang berbeda: lapisan input, lapisan tersembunyi (hidden), dan lapisan output.
- ANN terdiri dari satu atau lebih lapisan neuron.
- Lapisan input : menerima data masukan
- Lapisan tersembunyi : Membuat representasi fitur internal.
- Lapisan output : Menghasilkan output terakhir

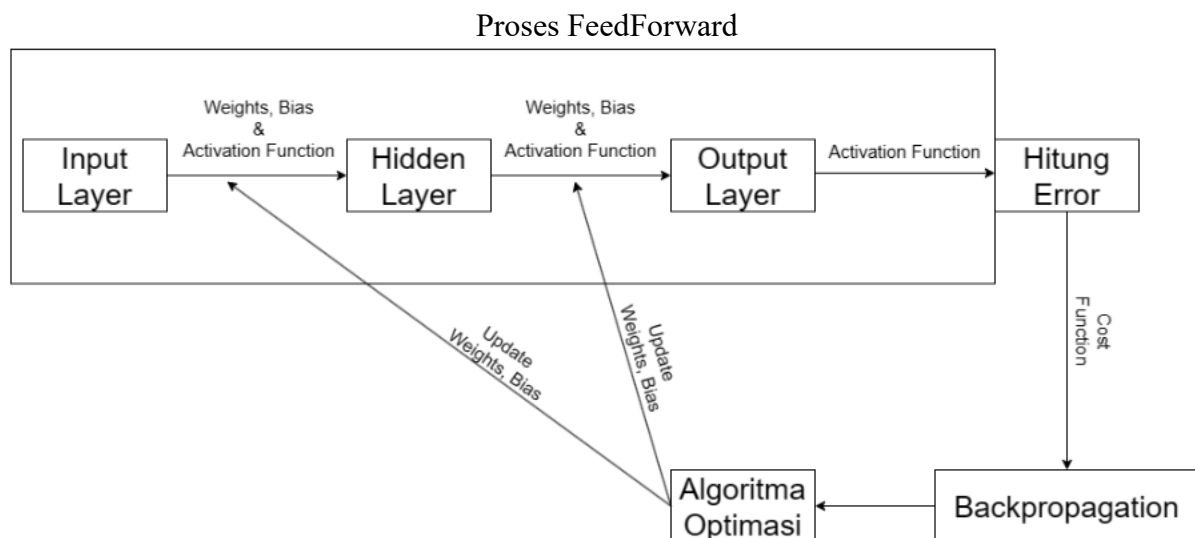
### 3.) Weights (Bobot)

- Bobot menghubungkan neuron dalam lapisan yang berbeda.
- Mereka mengontrol seberapa kuat pengaruh suatu neuron terhadap neuron lain.
- Bobot diubah selama proses pelatihan jaringan.

### 4.) Activation Function (Fungsi Aktivasi)

- Fungsi yang diterapkan pada output neuron untuk menentukan apakah mereka harus aktif atau tidak.
- Contoh fungsi aktivasi adalah ReLU, sigmoid, dan tanh.
- Mereka memperkenalkan non-linearitas ke dalam model.

## Cara Artificial Neural Networks Bekerja



### Penjelasan :

#### 1. Feedforward

Ini merupakan hal pertama yang dilakukan dalam training ANN, Pada proses ini terdiri dari 3 tahap:

- Input layer :

Layer ini menerima input data. Setiap neuron dalam layer ini mewakili satu fitur dari data.

- Hidden layer :

Layer ini berada di antara input dan output layer. Setiap neuron dalam layer ini menerima input, mengalikan dengan bobot, menambahkan bias, dan meneruskan hasil melalui fungsi aktivasi.

- Output layer :

Layer ini memberikan hasil akhir dari neural network. Setiap neuron dalam layer ini juga menerima input, mengalikan dengan bobot, menambahkan bias, dan meneruskan hasil melalui fungsi aktivasi.

#### 1. Hitung Error

Setelah feedforward, kita menghitung error dari hasil output layer dengan menghitung selisihnya dengan output yang sebenarnya.

#### 2. Backpropagation

- Apa itu Backpropagation?

Backpropagation adalah metode yang digunakan dalam neural network untuk menyesuaikan bobot dan bias berdasarkan error atau cost yang dihasilkan oleh output layer. Backpropagation adalah tulang punggung dari pelatihan neural network.

- Cara kerja

Setelah kita mendapatkan error, kita mulai backward pass. Di sini, kita menghitung gradien dari error terhadap bobot dan bias di setiap layer, mulai dari output layer dan bergerak mundur ke input layer

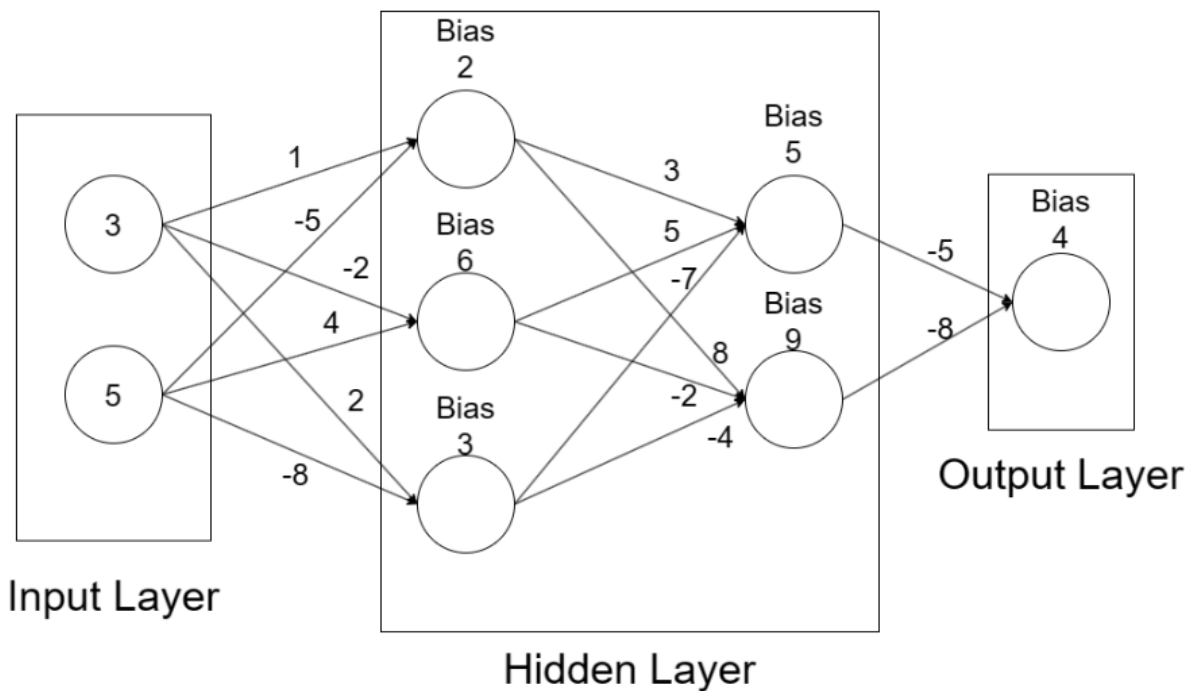
1. Algoritma Optimasi

Setelah kita memiliki gradien, kita dapat menggunakan mereka untuk memperbarui bobot dan bias menggunakan algoritma optimasi. Ini biasanya dilakukan dengan mengurangi gradien yang dihitung dari bobot dan bias saat ini, dikalikan dengan learning rate.

Untuk apa algoritma optimasi?

Algoritma optimasi digunakan dalam training ANN untuk memperbarui bobot dan bias dengan tujuan meminimalkan fungsi cost.

### Latihan FeedForward



In [2]: `pip install --upgrade pip`

Note: you may need to restart the kernel to use updated packages.

```
Usage:
  C:\Users\HP\anaconda3\envs\Desty M\python.exe -m pip install [options] <requirement specifier> [package-index-options] ...
  C:\Users\HP\anaconda3\envs\Desty M\python.exe -m pip install [options] -r <requirements file> [package-index-options] ...
  C:\Users\HP\anaconda3\envs\Desty M\python.exe -m pip install [options] [-e] <vcs project url> ...
  C:\Users\HP\anaconda3\envs\Desty M\python.exe -m pip install [options] [-e] <local project path> ...
  C:\Users\HP\anaconda3\envs\Desty M\python.exe -m pip install [options] <archive url/path> ...

no such option: -u
```

In [3]: `pip install pandas`

```
Requirement already satisfied: pandas in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.20.3 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]: `pip install numpy`

```
Requirement already satisfied: numpy in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (1.24.3)
Note: you may need to restart the kernel to use updated packages.
```



In [6]: pip install tensorflow

```
2.13->tensorflow-intel==2.13.0->tensorflow) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (2.1.1)
Requirement already satisfied: zipp>=0.5 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.11.0)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (0.5.0)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\hp\anaconda3\envs\desty m\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow-intel==2.13.0->tensorflow) (3.2.2)
Note: you may need to restart the kernel to use updated packages.
```

In [20]: pip install matplotlib

```
Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\5e\5d\01\3083e091b57809dad979ea543def62d9d878950e3e74f0c930
Building wheel for ushlex (setup.py): started
Building wheel for ushlex (setup.py): finished with status 'done'
Created wheel for ushlex: filename=ushlex-0.99.1-py3-none-any.whl size=4401 sha256=4c38fc17624ea894250c3d058476e3e24bd435f1eea2a5215262bc7b7d61f5bc
Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\d1\fd\c1\1a3ae6a14d03618418c757b09c0b55dcb07bb3409a6780a127
Successfully built SimpleWebsocketServer typing ushlex
Installing collected packages: ushlex, SimpleWebsocketServer, nh3, typing, rfc3986, pywin32-ctypes, pkginfo, more-itertools, mdurl, docutils, requests-toolbelt, readme-renderer, markdown-it-py, jaraco.classes, rich, keyring, twine, pyloco, matplotlib
Successfully installed SimpleWebsocketServer-0.1.2 docutils-0.20.1 jaraco.classes-3.3.0 keyring-24.2.0 markdown-it-py-3.0.0 matplotlib-0.1.9 mdurl-0.1.2 more-itertools-10.1.0 nh3-0.2.14 pkginfo-1.9.6 pyloco-0.0.139 pywin32-ctypes-0.2.2 readme-renderer-42.0 requests-toolbelt-1.0.0 rfc3986-2.0.0 rich-13.6.0 twine-4.0.2 typing-3.7.4.3 ushlex-0.99.1
Note: you may need to restart the kernel to use updated packages.
```

```
In [8]: #Latihan FEEDFORWARD

import numpy as np

# Fungsi Aktivaasi
def relu(input):
    return max(input, 0)

# Weights
weights = {
    'hl_1_1': np.array([1, -5]),
    'hl_1_2': np.array([-2, 4]),
    'hl_1_3': np.array([2, -8]),
    'hl_2_1': np.array([3, 5, -7]),
    'hl_2_2': np.array([8, -2, -4]),
    'hl_3_output': np.array([-5, -8]),}

# Bias
bias = {
    'b_1_1': 2,
    'b_1_2': 6,
    'b_1_3': 3,
    'b_2_1': 5,
    'b_2_2': 9,
    'b_3_output': 4,}

# Input Data
data = np.array([3, 5])
```

```
In [9]: def hidden_layer_1(input_data):
        node_1 = None
        node_2 = None
        node_3 = None
        return np.array([node_1, node_2, node_3])

        def hidden_layer_2(output_hidden_layer_1):
            node_1 = None
            node_2 = None
            return np.array([node_1, node_2])

        def output_layer(output_hidden_layer_2):
            return None
```

```
In [10]: output_hl_1 = None
output_hl_2 = None
output = None
print(output)
```

None

## Macam-macam Activation Function

Fungsi aktivasi memainkan peran penting dalam arsitektur Neural Network. Fungsi ini bertugas untuk menentukan output neuron berdasarkan set input tertentu.

Fungsi ini memperkenalkan non-linearitas ke dalam model, yang memungkinkan jaringan untuk memodelkan hubungan yang lebih kompleks dalam data.

### a.) Sigmoid

Fungsi aktivasi sigmoid mengubah input menjadi nilai antara 0 dan 1. Ini berguna dalam kasus klasifikasi biner.

Keuntungan :

Output berkisar antara 0 dan 1, berguna untuk mengubah nilai menjadi probabilitas.

Kekurangan :

Sebesar dan sekecil apapun nilai input, Output yang dikeluarkan hanya di sekitar 0 sampai 1.

```
In [11]: #macam-macam activation function
# a.) SIGMOID

import numpy as np

# Membuat fungsi Sigmoid
def sigmoid(input):
    return 1 / (1 + np.exp(-input))

# Contoh data input
x = np.array([-2.0, -1.0, 0.0, 1.0, 2.0])

for i in x:
    print("Angka Input : ", i, "| Output Sigmoid : ", sigmoid(i))

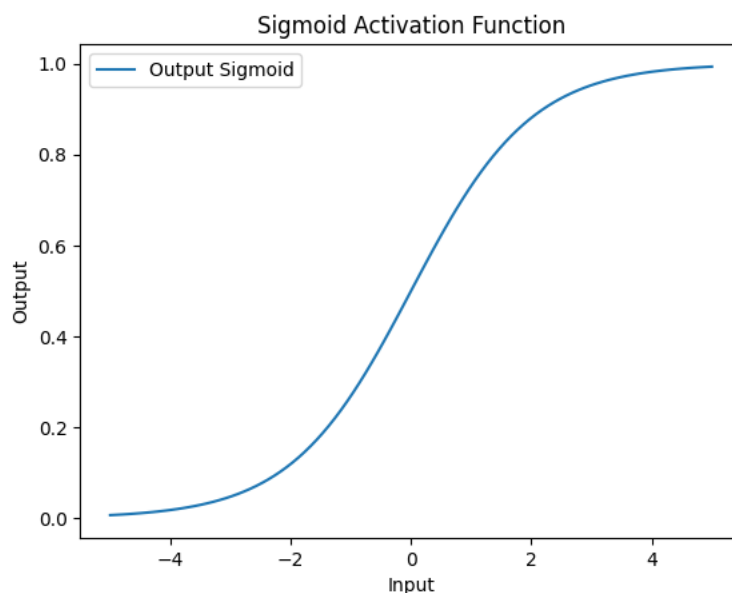
Angka Input : -2.0 | Output Sigmoid : 0.11920292202211755
Angka Input : -1.0 | Output Sigmoid : 0.2689414213699951
Angka Input : 0.0 | Output Sigmoid : 0.5
Angka Input : 1.0 | Output Sigmoid : 0.7310585786300049
Angka Input : 2.0 | Output Sigmoid : 0.8807970779778823
```

```
In [12]: # Membuat Grafik Output Sigmoid
import matplotlib.pyplot as plt

# Membuat data yang akan dimasukkan ke sigmoid
# Data ini memiliki jumlah sebanyak 100 dengan rentang dari -5 sampai 5
x = np.linspace(-5, 5, 100)

# Menyimpan output sigmoid
output = sigmoid(x)

# Membuat grafik output sigmoid
plt.plot(x, output, label='Output Sigmoid')
plt.title('Sigmoid Activation Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```



## b.) ReLU (Rectified Linear Unit)<p>

Fungsi aktivasi ReLU mengubah input yang bernilai negatif menjadi 0, sementara input yang bernilai positif tidak diubah.

Keuntungan :

ReLU sangat efisien secara komputasi.

Kekurangan :

ReLU memiliki apa yang dikenal sebagai "dying ReLU problem" - jika neuron yang menggunakan fungsi ReLU mendapatkan input negatif, maka neuron tersebut akan "mati", atau berhenti belajar

```
In [13]: # b.) ReLU (Rectified Linear Unit)

# Membuat fungsi ReLU
def ReLU(input):
    return np.maximum(input, 0)

# Contoh data input
x = np.array([-2.0, -1.0, 0.0, 1.0, 2.0])

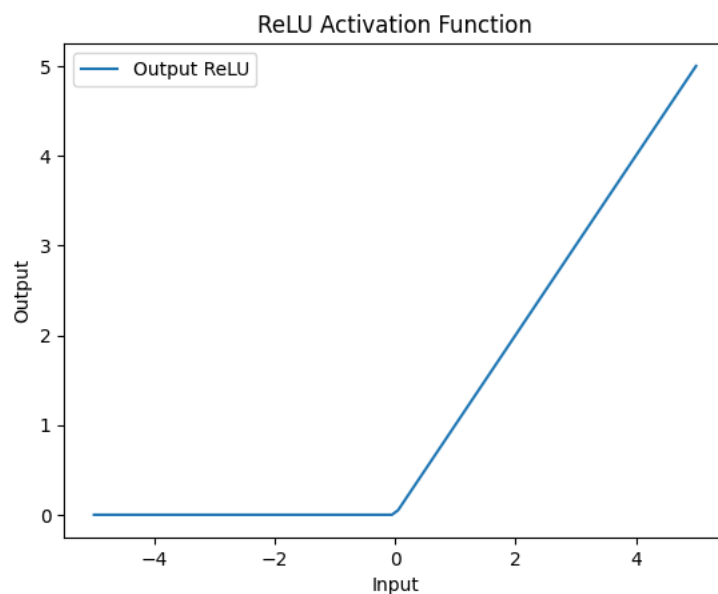
for i in x:
    print("Angka Input : ", i, "| Output ReLU : ", ReLU(i))
```

```
Angka Input : -2.0 | Output ReLU : 0.0
Angka Input : -1.0 | Output ReLU : 0.0
Angka Input : 0.0 | Output ReLU : 0.0
Angka Input : 1.0 | Output ReLU : 1.0
Angka Input : 2.0 | Output ReLU : 2.0
```

```
In [14]: # Membuat Grafik Output ReLU
# Membuat data yang akan dimasukkan ke ReLU
# Data ini memiliki jumlah sebanyak 100 dengan rentang dari -5 sampai 5
x = np.linspace(-5, 5, 100)

# Menyimpan output ReLU
output = ReLU(x)

# Membuat grafik output ReLU
plt.plot(x, output, label='Output ReLU')
plt.title('ReLU Activation Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```



c.) Tanh

Tanh adalah alternatif lain untuk sigmoid. Output dari fungsi ini memiliki rentang lebih besar dari sigmoid yaitu berkisar antara -1 hingga 1, yang menjadikannya lebih baik dalam menangani negatif input

Keuntungan :

Sama seperti sigmoid, tetapi lebih baik menangani input negatif.

Kekurangan :

Sebesar dan sekecil apapun nilai input, Output yang dikeluarkan hanya di sekitar -1 sampai 1.

```
In [15]: # c.) Tanh

# Membuat fungsi tanh
def tanh(input):
    return np.tanh(input)

# Contoh data input
x = np.array([-2.0, -1.0, 0.0, 1.0, 2.0])

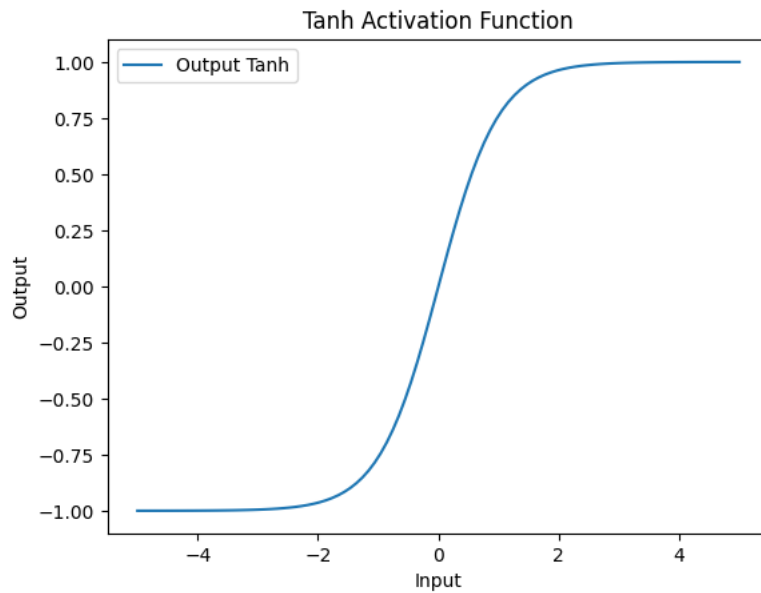
for i in x:
    print("Angka Input : ", i, "| Output Tanh : ", tanh(i))

Angka Input : -2.0 | Output Tanh : -0.9640275800758169
Angka Input : -1.0 | Output Tanh : -0.7615941559557649
Angka Input : 0.0 | Output Tanh : 0.0
Angka Input : 1.0 | Output Tanh : 0.7615941559557649
Angka Input : 2.0 | Output Tanh : 0.9640275800758169

In [16]: # Membuat Grafik Output tanh
# Membuat data yang akan dimasukkan ke tanh
# Data ini memiliki jumlah sebanyak 100 dengan rentang dari -5 sampai 5
x = np.linspace(-5, 5, 100)

# Menyimpan output tanh
output = tanh(x)

# Membuat grafik output tanh
plt.plot(x, output, label='Output Tanh')
plt.title('Tanh Activation Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```



#### d.) Softmax

Fungsi aktivasi softmax digunakan dalam lapisan output jaringan yang digunakan untuk klasifikasi multi-kelas. Ini mengubah nilai input menjadi vektor probabilitas.

Keuntungan :

Berguna untuk klasifikasi multi-kelas.

Kekurangan :

Tidak cocok untuk digunakan di hidden layers.

```
In [17]: # d.) SOFTMAX

# Membuat fungsi Softmax
def softmax(input):
    e_x = np.exp(x - np.max(input))
    return e_x / e_x.sum()

# Contoh data input
x = np.array([-2.0, -1.0, 0.0, 1.0, 2.0])

for i in x:
    print("Angka Input : ", i, "| Output Softmax : ", softmax(i))

Angka Input : -2.0 | Output Softmax : [0.01165623 0.03168492 0.08612854 0.234
12166 0.63640865]
Angka Input : -1.0 | Output Softmax : [0.01165623 0.03168492 0.08612854 0.234
12166 0.63640865]
Angka Input : 0.0 | Output Softmax : [0.01165623 0.03168492 0.08612854 0.2341
2166 0.63640865]
Angka Input : 1.0 | Output Softmax : [0.01165623 0.03168492 0.08612854 0.2341
2166 0.63640865]
Angka Input : 2.0 | Output Softmax : [0.01165623 0.03168492 0.08612854 0.2341
2166 0.63640865]
```

#### e.) Leaky ReLU

Leaky ReLU adalah varian dari fungsi aktivasi ReLU yang memungkinkan sebagian kecil nilai negatif untuk tetap melewati tanpa menjadi nol mutlak. Ini membantu mengatasi masalah "neuron mati" yang bisa terjadi pada ReLU biasa.

Keuntungan :

Mengatasi masalah dying ReLU.

Kekurangan :

Hasilnya mungkin tidak stabil untuk data noise.

In [18]: # E.) LEAKY ReLU

```
# Membuat fungsi Leaky ReLU
def LeakyReLU(input):
    return np.maximum(0.01*input, input)

# Contoh data input
x = np.array([-2.0, -1.0, 0.0, 1.0, 2.0])

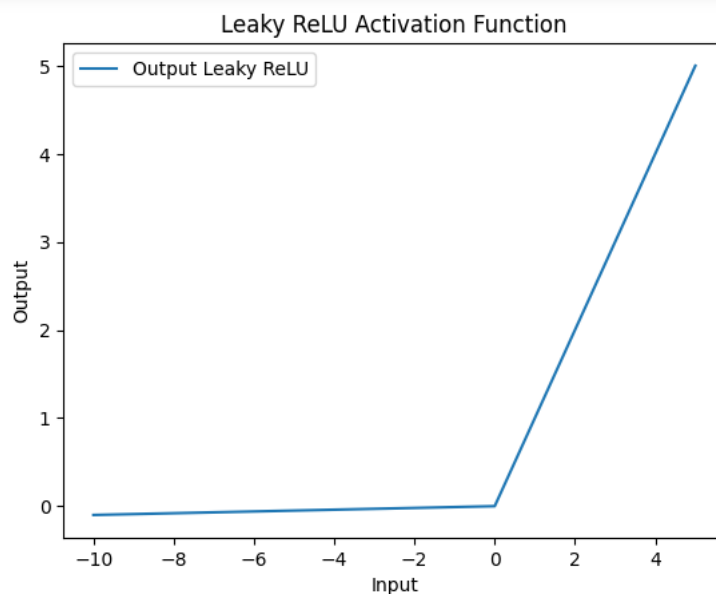
for i in x:
    print("Angka Input : ", i, "| Output Leaky ReLU : ", LeakyReLU(i))
```

Angka Input : -2.0 | Output Leaky ReLU : -0.02  
Angka Input : -1.0 | Output Leaky ReLU : -0.01  
Angka Input : 0.0 | Output Leaky ReLU : 0.0  
Angka Input : 1.0 | Output Leaky ReLU : 1.0  
Angka Input : 2.0 | Output Leaky ReLU : 2.0

```
In [19]: # Membuat Grafik Output Leaky ReLU
# Membuat data yang akan dimasukkan ke Leaky ReLU
# Data ini memiliki jumlah sebanyak 100 dengan rentang dari -10 sampai 5
x = np.linspace(-10, 5, 100)

# Menyimpan output Leaky ReLU
output = LeakyReLU(x)

# Membuat grafik output Leaky ReLU
plt.plot(x, output, label='Output Leaky ReLU')
plt.title('Leaky ReLU Activation Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```



```
In [21]: # contoh Data Target fungsi Loss Binary Crossentropy
# Target Data Hanya mempunyai 2 kelas, 0 dan 1
y = [0, 0, 1, 1, 0, 1, 0]

# contoh Data Target Multi-Kelas yang Menggunakan fungsi Loss Binary Crossentropy
y = [
    [0, 0, 0],
    [255, 0, 0],
    [20, 255, 0],]

# atau juga bisa bentuknya seperti ini
y = [
    [0, 0.1, 0.2],
    [1, 0, 0.1],
    [1, 0.8, 0],]
```

```
In [22]: # contoh Data Target yang Menggunakan fungsi Loss Sparse Categorical Crossentropy
y = [0, 0, 1, 4, 2, 3, 0,]
```

```
In [31]: # Masukkan Data
import pandas as pd

df = pd.read_csv('D:/Semester 5/Prak Deep Learning/Tugas/praktikum_1.csv')
df
```

Out[31]:

	x1	x2	x3	x4	y
0	0.969049	0.811884	0.364106	0.791886	1.339874
1	0.189405	0.437117	0.105088	0.871999	0.288839
2	0.850152	0.090524	0.671884	0.809457	0.314485
3	0.944087	0.132889	0.489603	0.043965	0.940221
4	0.460488	0.903488	0.101475	0.331271	1.243162
...	...	...	...	...	...
9995	0.729730	0.584990	0.365986	0.680118	0.823030
9996	0.408094	0.964827	0.784377	0.542862	0.913176
9997	0.261231	0.547435	0.248148	0.855978	0.348508
9998	0.304984	0.907731	0.031752	0.475175	1.113871



9999 0.721289 0.251400 0.206529 0.433288 0.695005

10000 rows x 5 columns

```
In [33]: from sklearn.model_selection import train_test_split

X = df.drop('y', axis=1).values
Y = df['y'].values

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_
```

```
In [35]: X_train, x_val, Y_train, y_val = train_test_split(x_train, y_train, test_size=0.
```

```
In [36]: from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(16, input_dim=4, activation='tanh'))
model.add(Dense(8, activation='tanh'))
model.add(Dense(4, activation='tanh'))
model.add(Dense(2, activation='tanh'))
model.add(Dense(1, activation='tanh'))
```

```
In [37]: model.compile(loss='mse', optimizer='adam')
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 16)	80
dense_1 (Dense)	(None, 8)	136
dense_2 (Dense)	(None, 4)	36
dense_3 (Dense)	(None, 2)	10
dense_4 (Dense)	(None, 1)	3
Total params: 265 (1.04 KB)		
Trainable params: 265 (1.04 KB)		
Non-trainable params: 0 (0.00 Byte)		

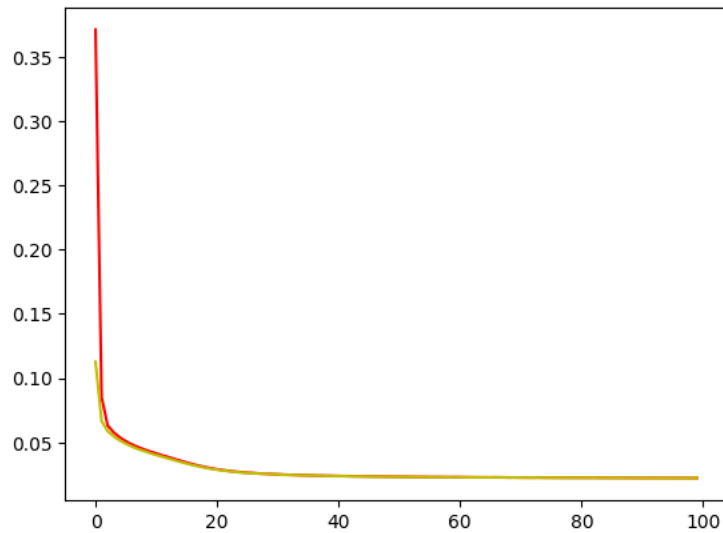
```
In [38]: catatan = model.fit(X_train, Y_train, validation_data=[x_val, y_val], epochs=100
```

```
Epoch 1/100
100/100 [=====] - 1s 4ms/step - loss: 0.3710 - val_
loss: 0.1127
Epoch 2/100
100/100 [=====] - 0s 2ms/step - loss: 0.0851 - val_
loss: 0.0665
Epoch 3/100
100/100 [=====] - 0s 2ms/step - loss: 0.0631 - val_
loss: 0.0586
Epoch 4/100
100/100 [=====] - 0s 2ms/step - loss: 0.0576 - val_
loss: 0.0546
Epoch 5/100
100/100 [=====] - 0s 2ms/step - loss: 0.0538 - val_
loss: 0.0514
Epoch 6/100
100/100 [=====] - 0s 2ms/step - loss: 0.0508 - val_
loss: 0.0489
Epoch 7/100
```

```
In [39]: import matplotlib.pyplot as plt

plt.plot(range(len(catatan.history['loss'])), catatan.history['loss'], 'r',
         range(len(catatan.history['val_loss'])), catatan.history['val_loss'], 'y')
plt.show()
```

```
#JIKA JARAK LOSS LEBIH BESAR JAUH DARIPADA VALIDATION LOSS DISEBUT 'OVERFITTING'
#JIKA JARAK LOSS LEBIH KECIL DARIPADA VALIDATION LOSS DISEBUT '
```



```
In [40]: loss = model.evaluate(x_test, y_test)
print('Loss Model = ', loss)

63/63 [=====] - 0s 1ms/step - loss: 0.0272
Loss Model = 0.02724793553352356
```

```
In [41]: model.predict([
    [0.5, 0.4, 0.6, 0.6],
    [0.1, 0.8, 0.3, 0.7],
    [0.9, 0.10, 0.5, 0.1],
    [0.24, 0.3, 0.7, 0.11]
])

1/1 [=====] - 0s 136ms/step

Out[41]: array([[0.28732136],
                [0.5231616 ],
                [0.83024484],
                [0.25442216]], dtype=float32)
```

<https://github.com/destymaya>

[illegible]