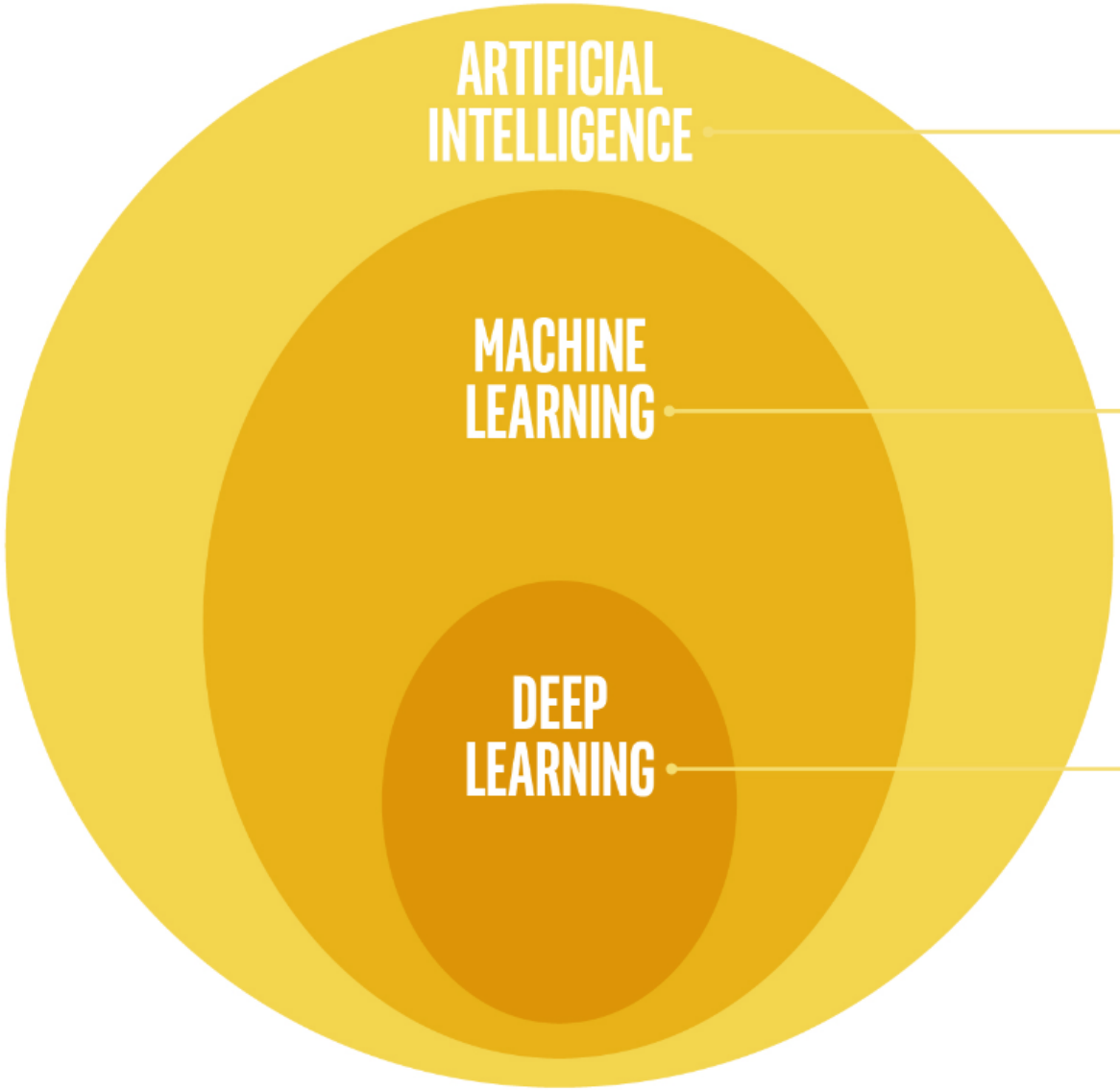
Artificial Neural Network and Its Use Case in Physical Layer

WRITEX
Sabtu, 2 Juni 2023

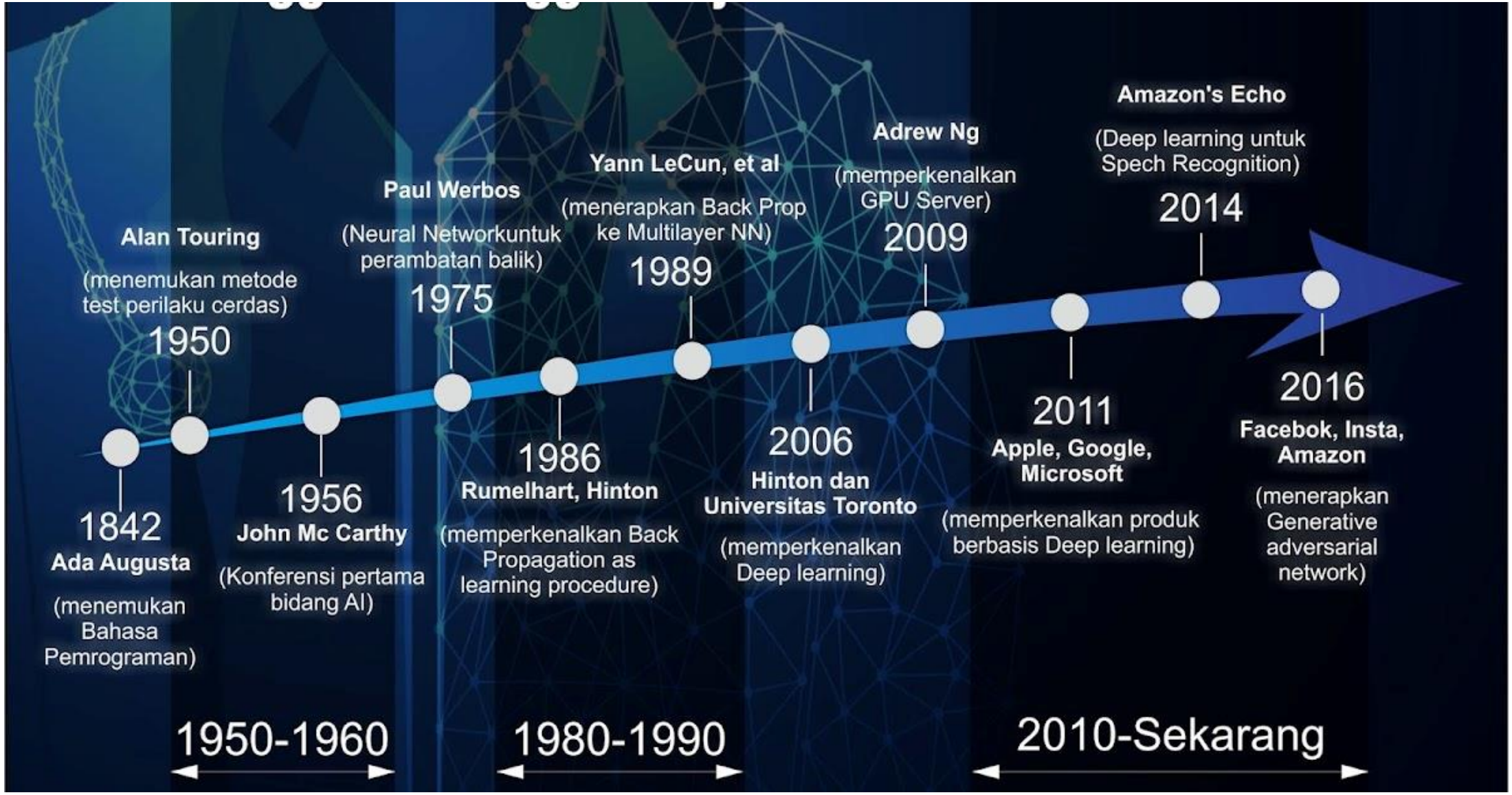
Disusun oleh Gelar Budiman[1,2]
E-mail: gelarbudiman@telkomuniversity.ac.id

Definition of Artificial Intelligence (AI)

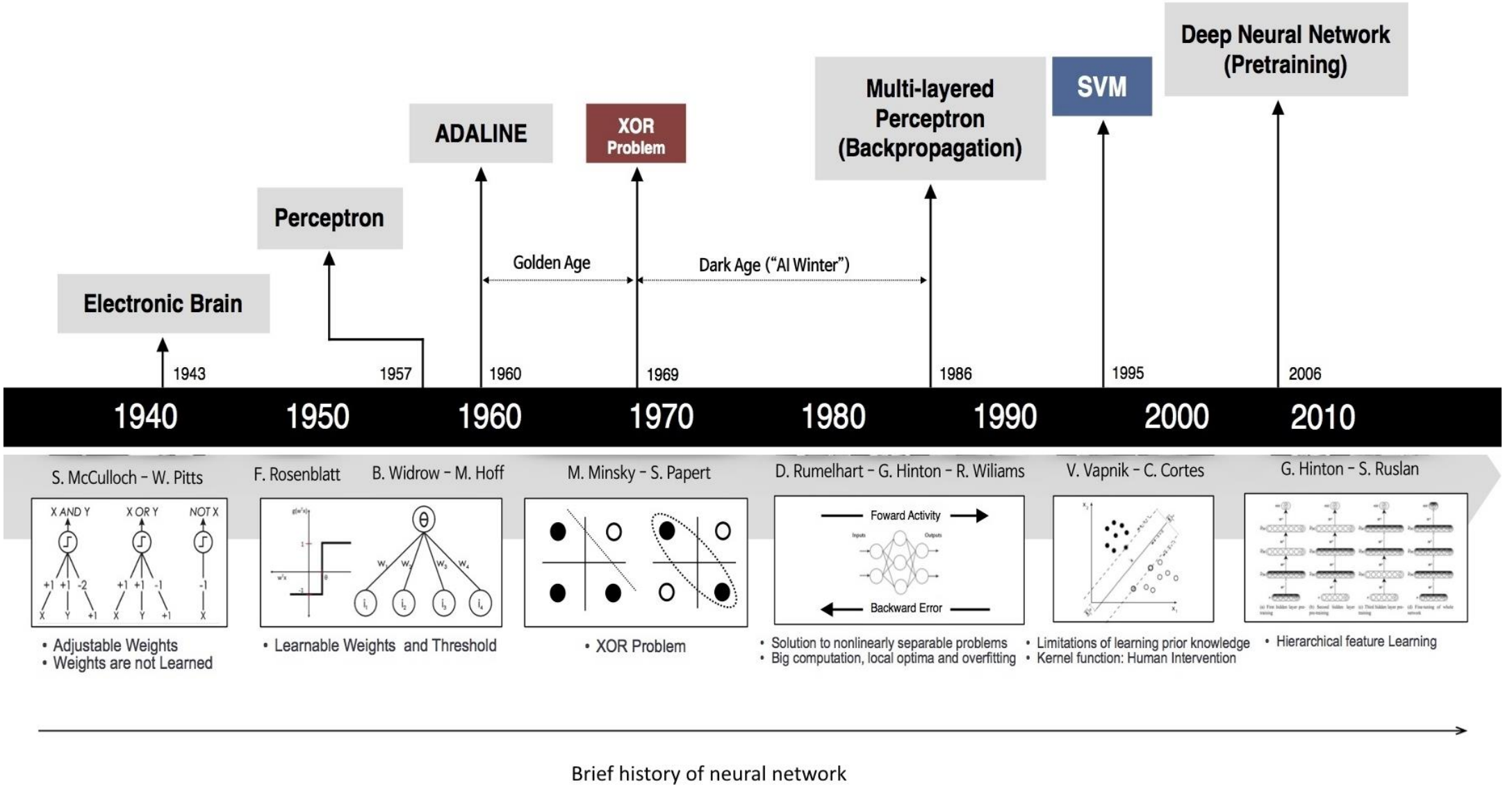


- **AI** is an umbrella term for machines capable of perception, logic, and learning.
- **Machine learning** employs algorithms that learn from data to make predictions or decisions, and whose performance improves when exposed to more data over time.
- **Deep learning** uses many-layered neural networks to build algorithms that find the best way to perform tasks on their own, based on vast sets of data.

History of Artificial Intelligence (AI)

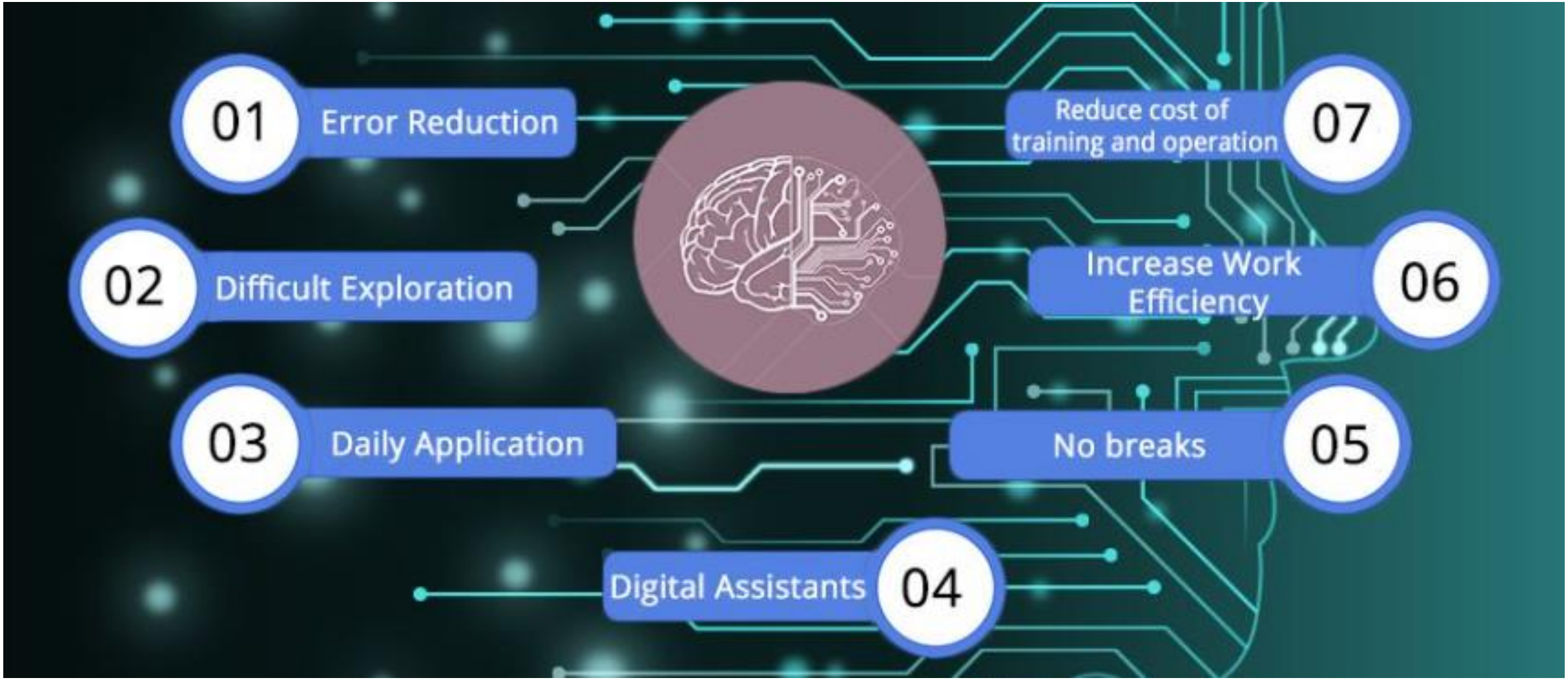


History of Deep Learning

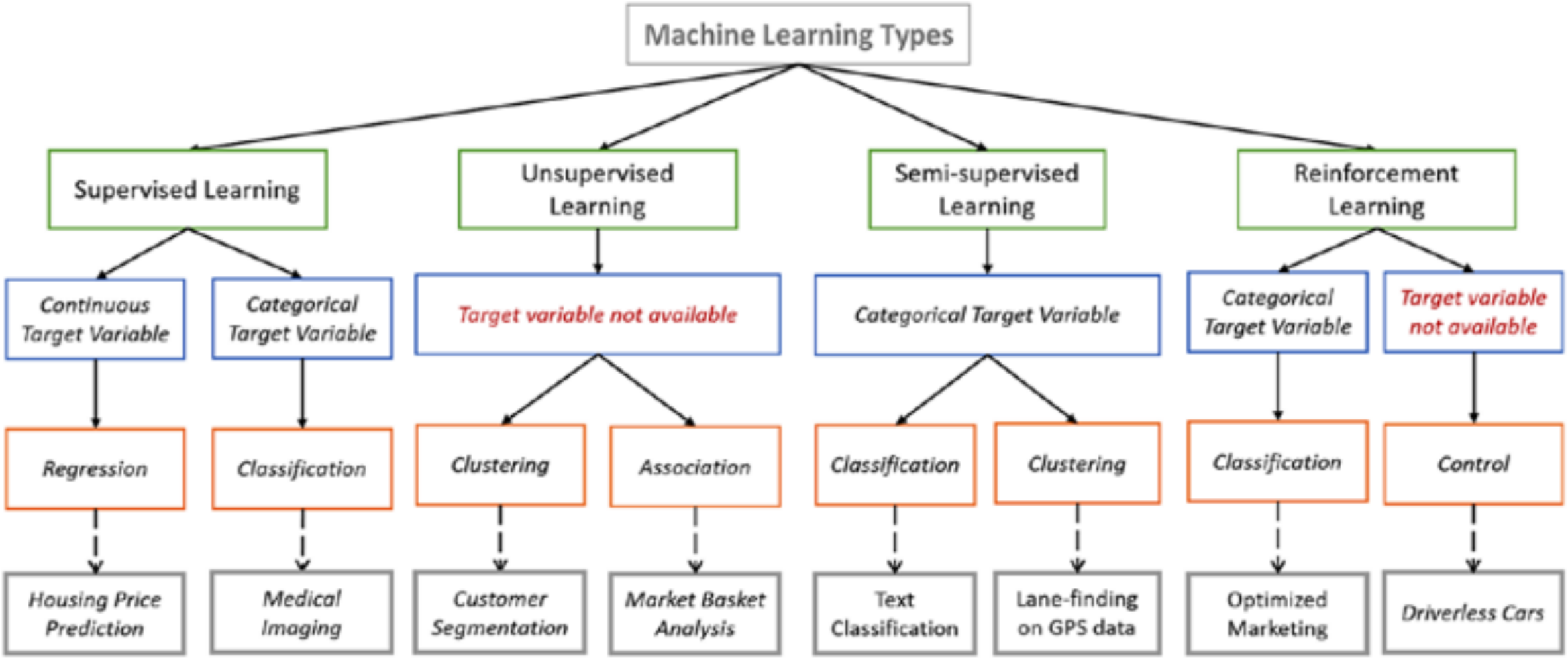


Brief history of neural network

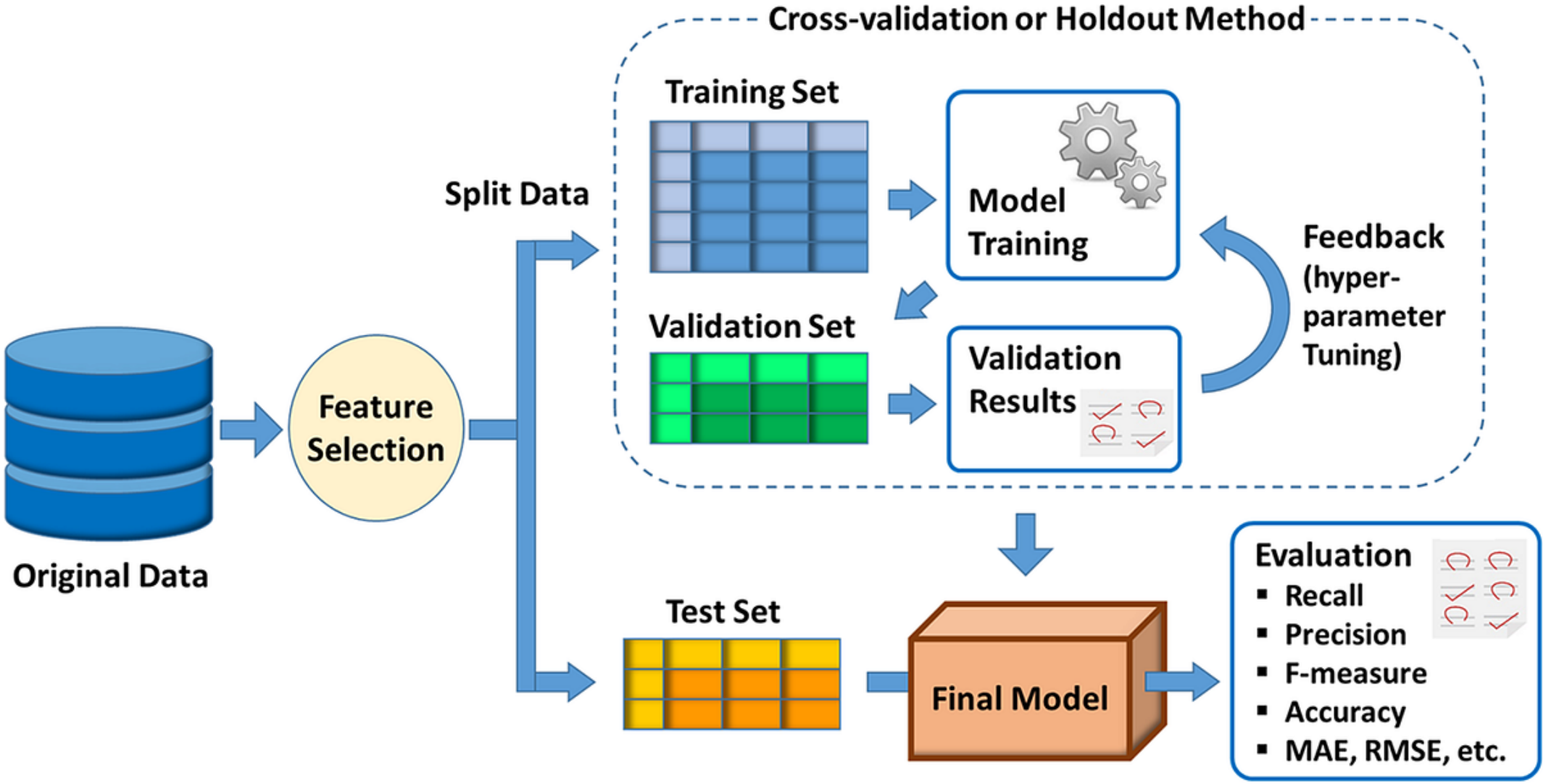
Benefits of AI



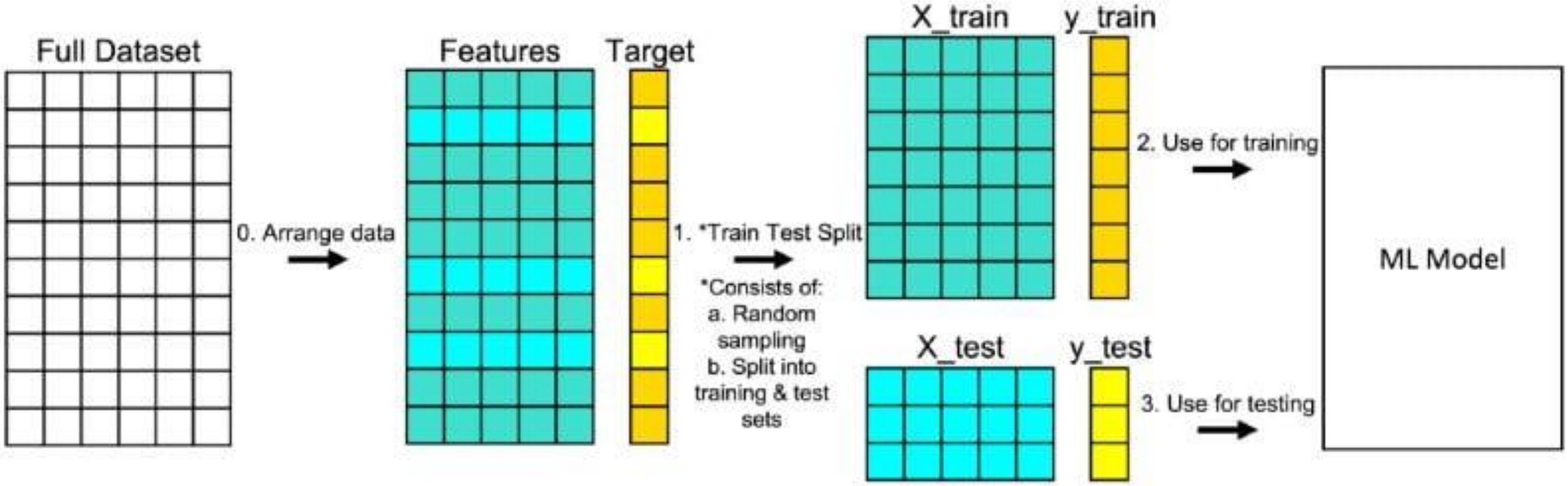
Artificial Intelligence Types



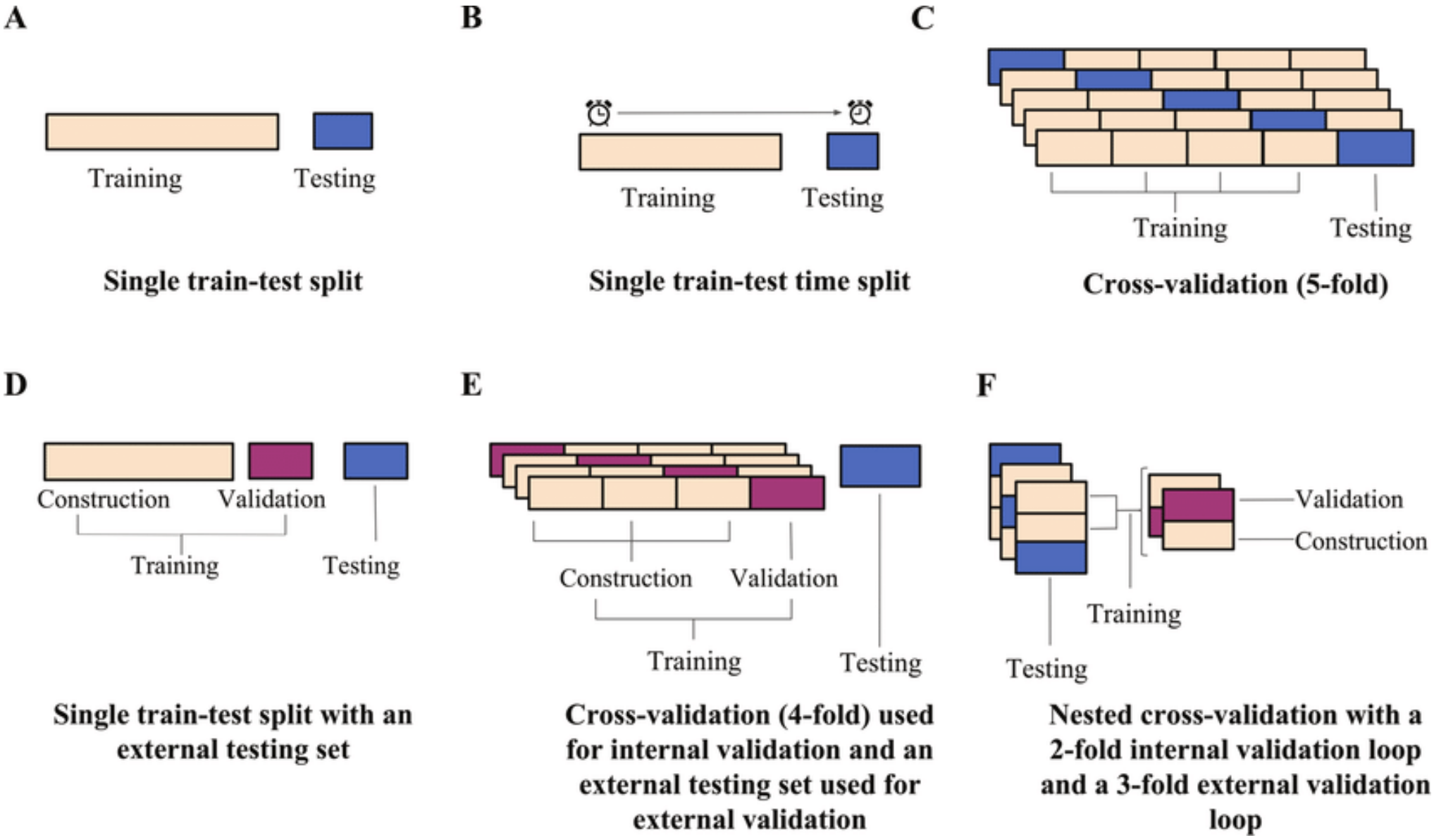
AI Process


Cross-validation or Holdout Method
Training Set
Split Data
Model Training
Feedback (hyper-parameter Tuning)
Validation Set
Validation Results
Feature Selection
Original Data
Test Set
Final Model
Evaluation
Recall
Precision
F-measure
Accuracy
MAE, RMSE, etc.

Splitting Dataset



Validation Verification



Dataset With Continue and Discrete Class

x : variabel bebas y : variabel tak bebas

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

X_train y_train

Panjang Sepal	Lebar Sepal	Panjang Petal	Lebar Petal	Kelas
5.1	3.5	1.4	0.2	Iris Setosa
6.3	3.3	6	2.5	Iris Virginica
7	3	4.6	1.4	Iris Versicolour
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
5.8	3.3	6	2.4	Iris Virginica
6.8	3.1	4.5	1.5	Iris Versicolour
4.9	3	1.4	0.2	Iris Setosa
...	...	...	...	...
6.8	3.2	4.4	1.6	Iris Versicolour

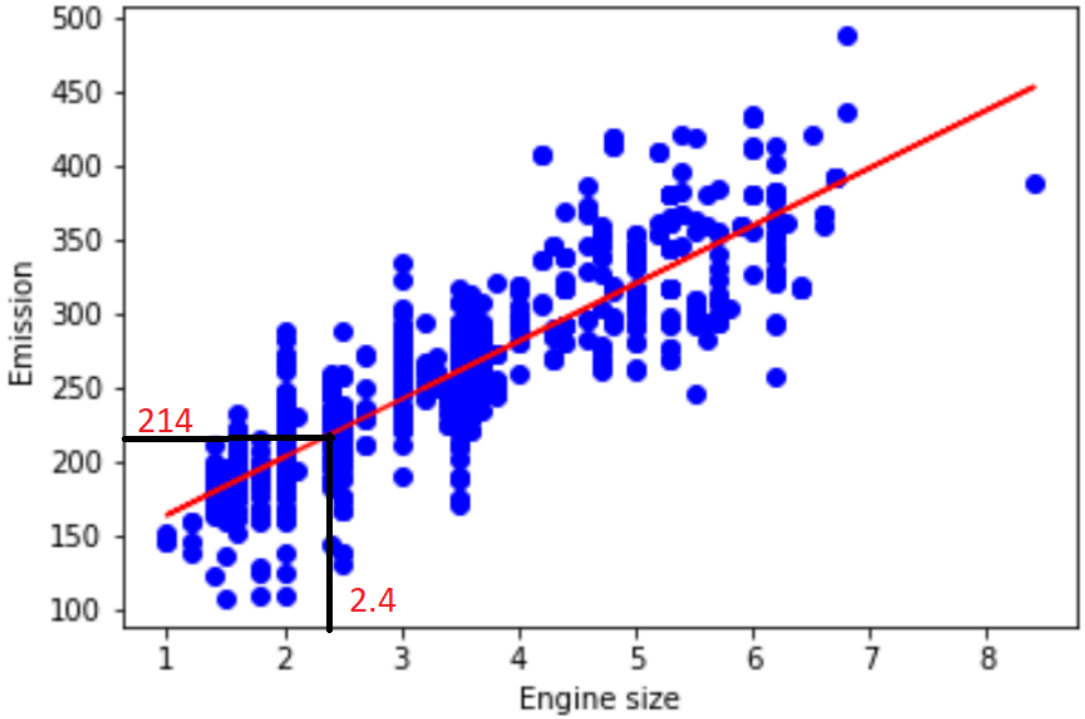
Training Data 70%

Testing Data 30%

X_test y_test

Case : Continue Class Prediction

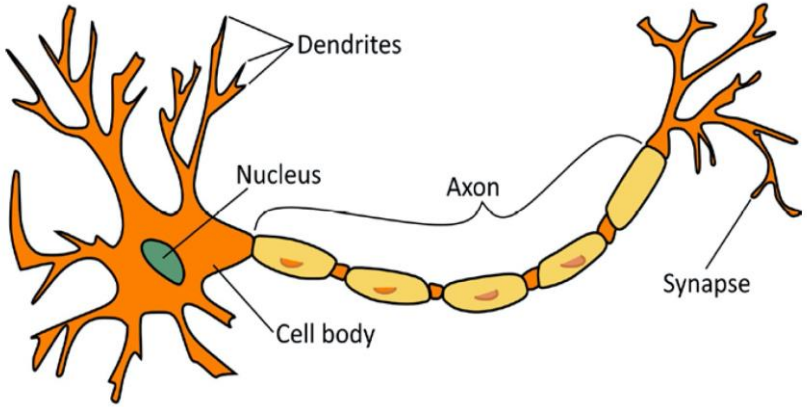
	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	?

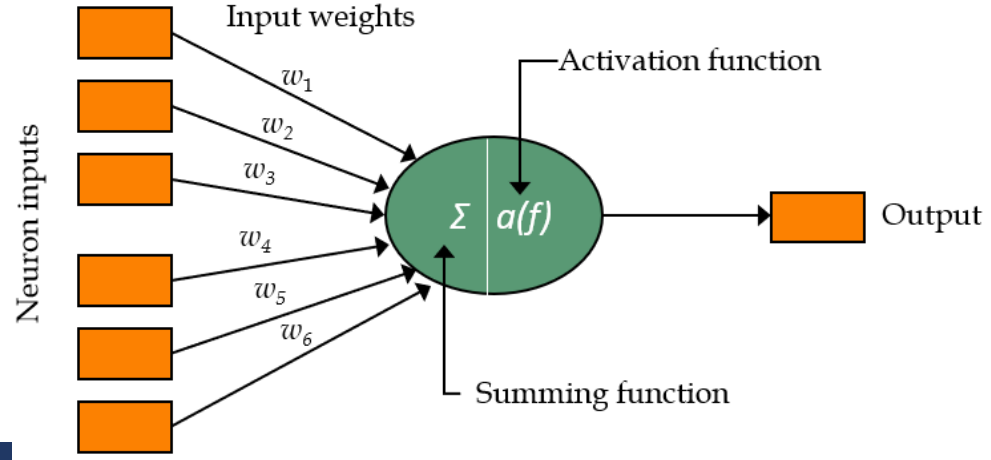


Artificial Neural Network (ANN)

- Salah satu metode mesin pembelajaran yang terinspirasi oleh cara kerja jaringan saraf biologis di otak manusia
- Merupakan jaringan dari unit pemroses kecil yang saling terhubung, yang dimodelkan berdasar sistem saraf manusia
- Konsep ANN bermula pada artikel dari Waffen McCulloch dan Walter Pitts pada tahun 1943 yaitu mencoba untuk memformulasikan model matematis sel-sel otak manusia

Jaringan saraf biologis	ANN
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight





Arsitektur Single-layer Perceptron

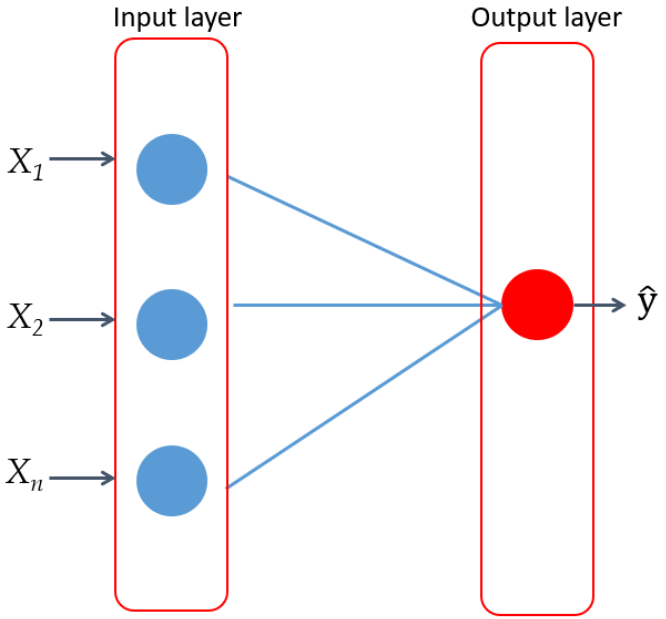
Arsitektur single-layer ANN hanya terdiri dari input layer dan output layer

Unit pemrosesan informasi pada ANN sebagai berikut:

- Satu set link berupa neuron dan bobot w
- Fungsi penambah (penggabung linear) untuk mengitung jumlah perkalian bobot terhadap input X
- Fungsi aktivasi $a(\cdot)$

$$f = \sum_{i=1}^{m} w_i x_i + b$$

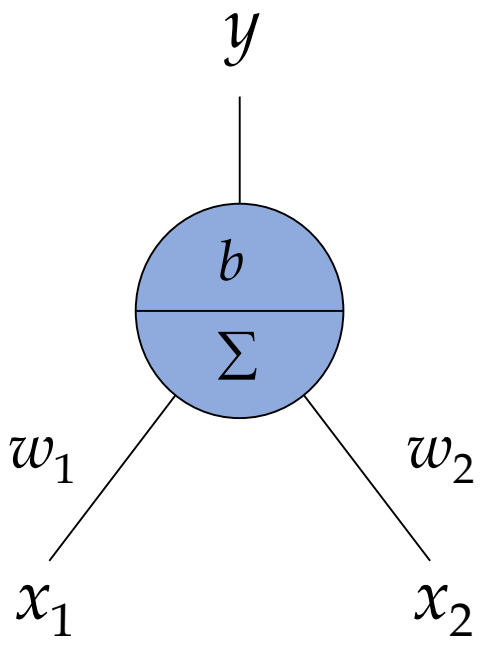
$$y = a(f)$$



Apa yang bisa dilakukan sebuah Neuron ?

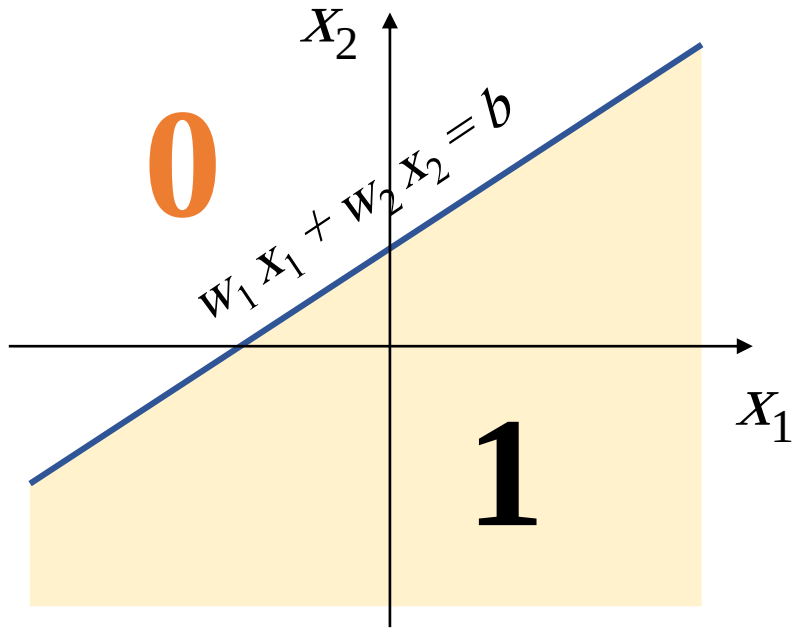
Sebuah neuron pada ANN dapat menyelesaikan permasalahan klasifikasi biner

- Sebagai fungsi pemisah (*hyperspace separation*)
- Sebagai *binary threshold*



$$f(x) = w_1 x_1 + w_2 x_2 - b$$

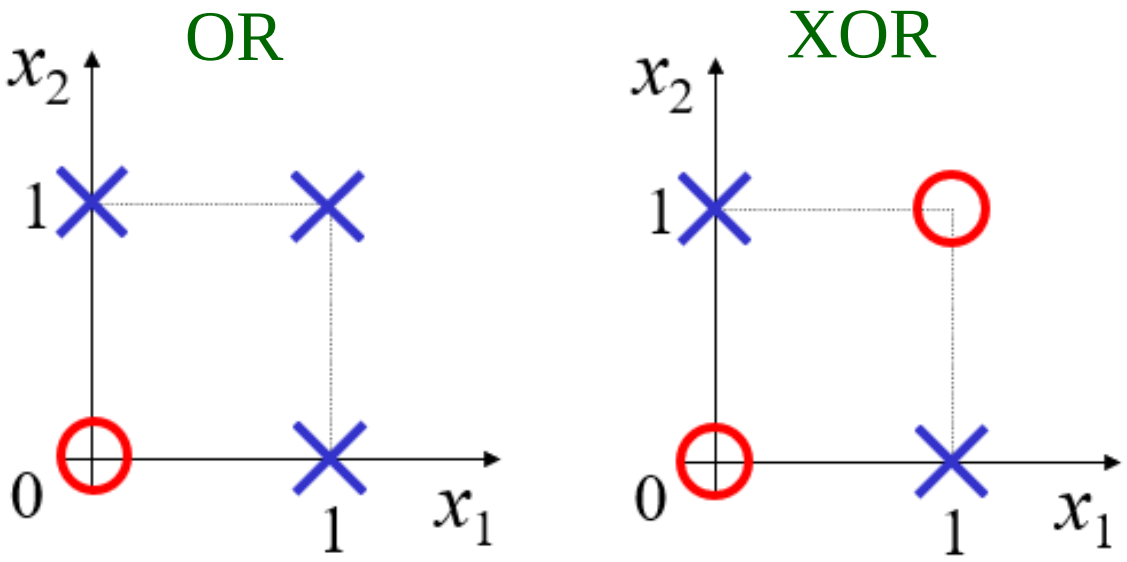
$$y = \begin{cases} 1 & f(x) \geq 0 \\ 0 & otherwise \end{cases}$$

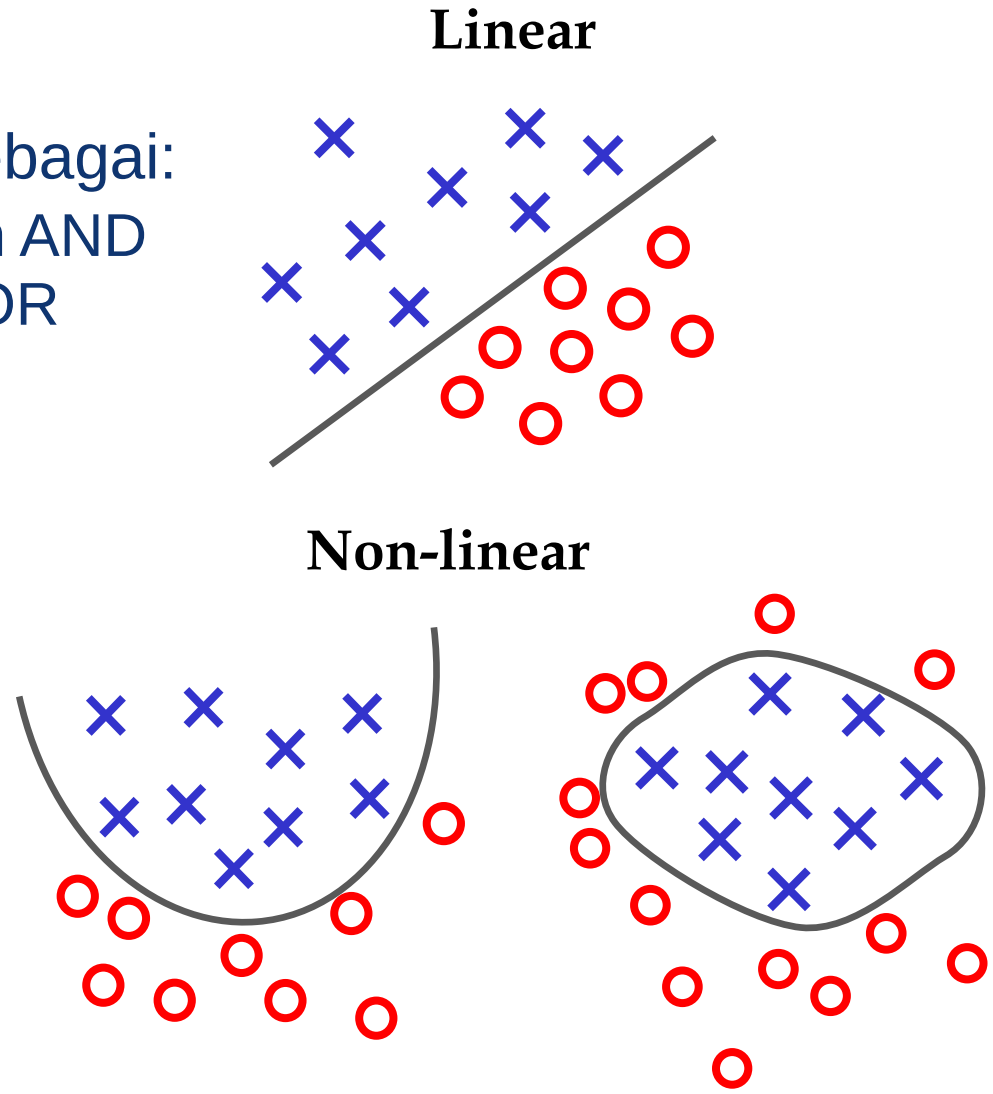


Permasalahan Linear dan Non-Linear

Permasalahan klasifikasi dapat dikategorikan sebagai:

- Permasalahan Linear, misalnya fungsi OR dan AND
- Permasalahan Non-Linear, misalnya fungsi XOR



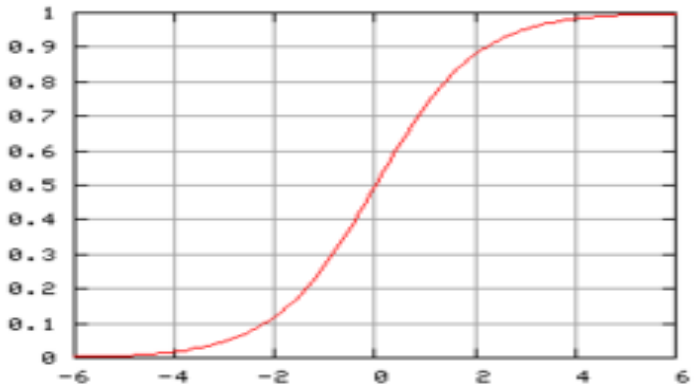


Activation Function

- The activation function turns neurons into non-linear
- Some examples of activation functions commonly used in the ANN method are as follows

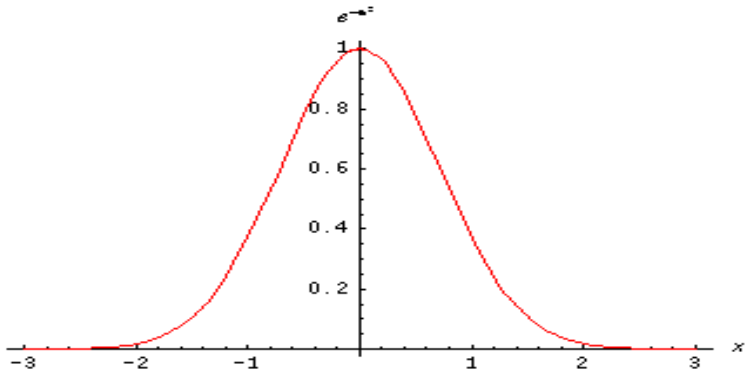
Sigmoid function

$$a(f) = \frac{1}{1+\exp(-f)}$$



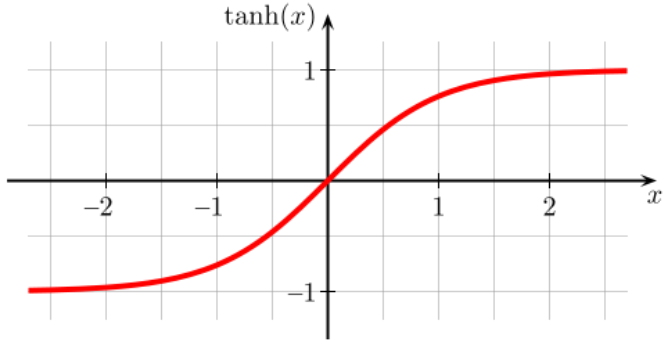
Gaussian function

$$a(f) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{f-\mu}{\sigma}\right)^2\right)$$

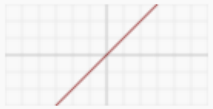
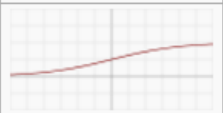
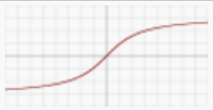
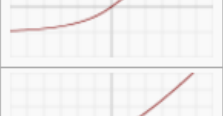


Tangent Hyperbolic function

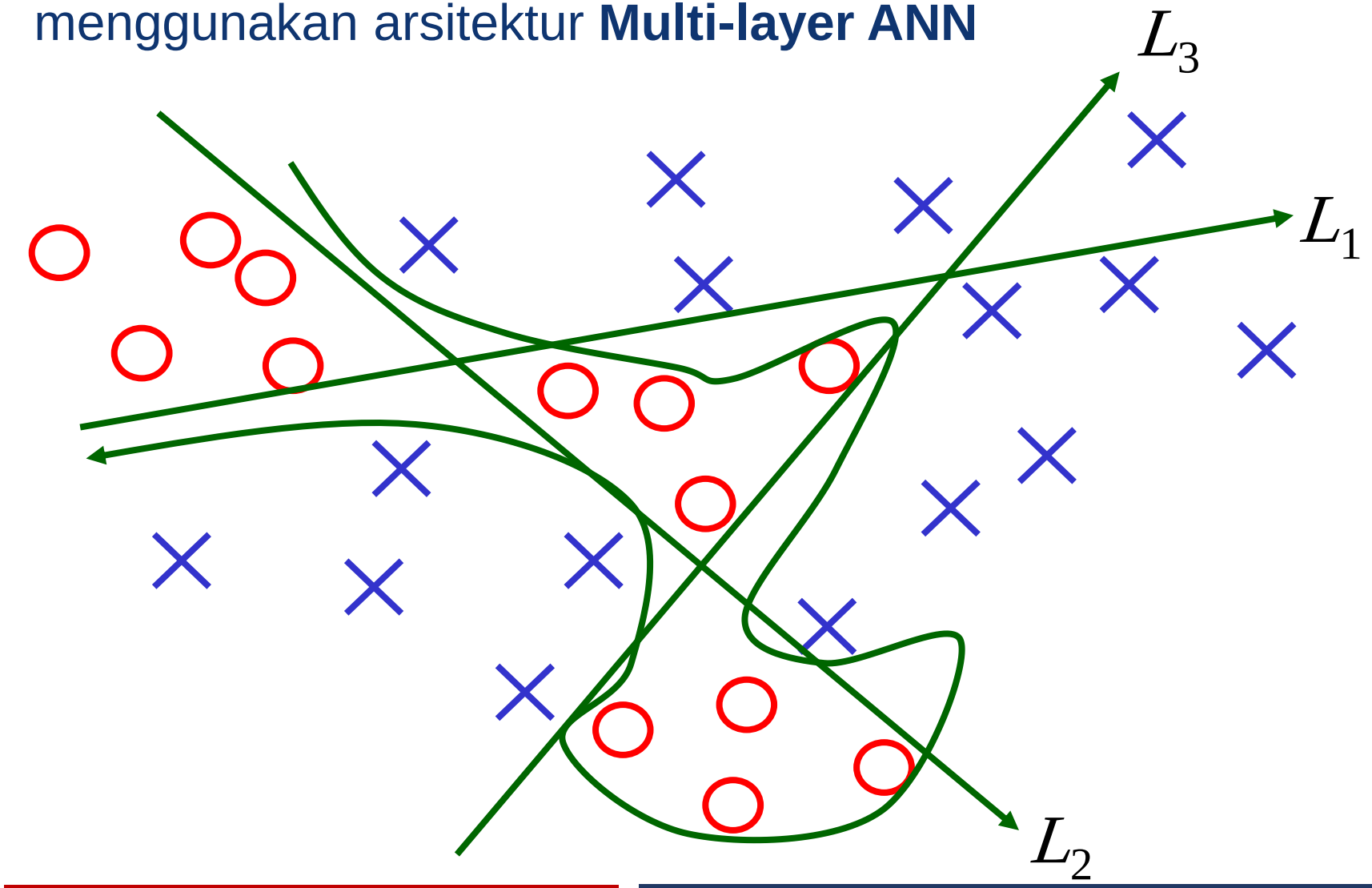
$$\tanh(x) = 2\sigma(2x) - 1$$

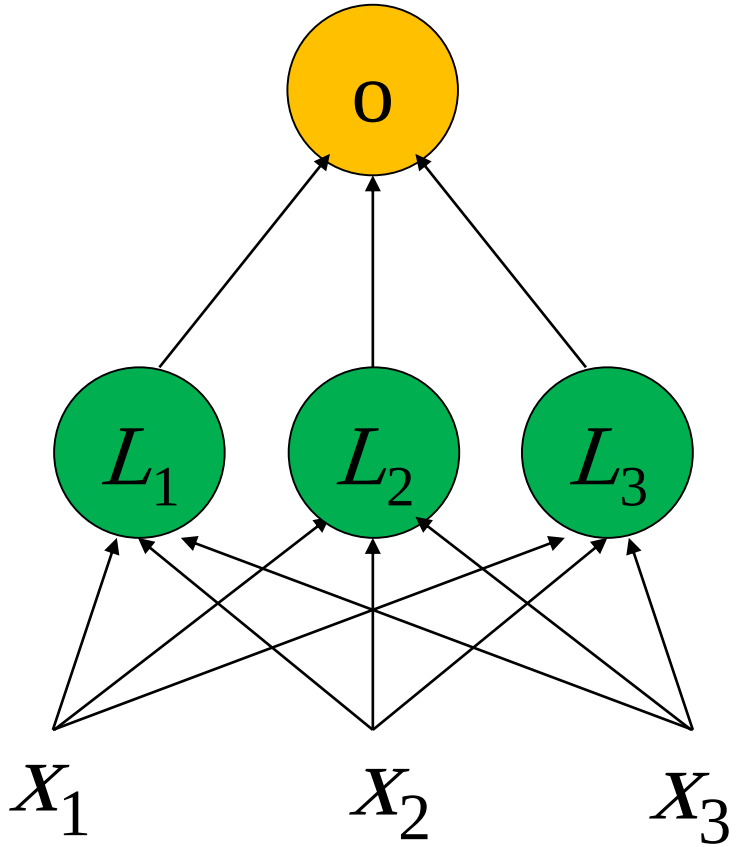


Activation function and its difference

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

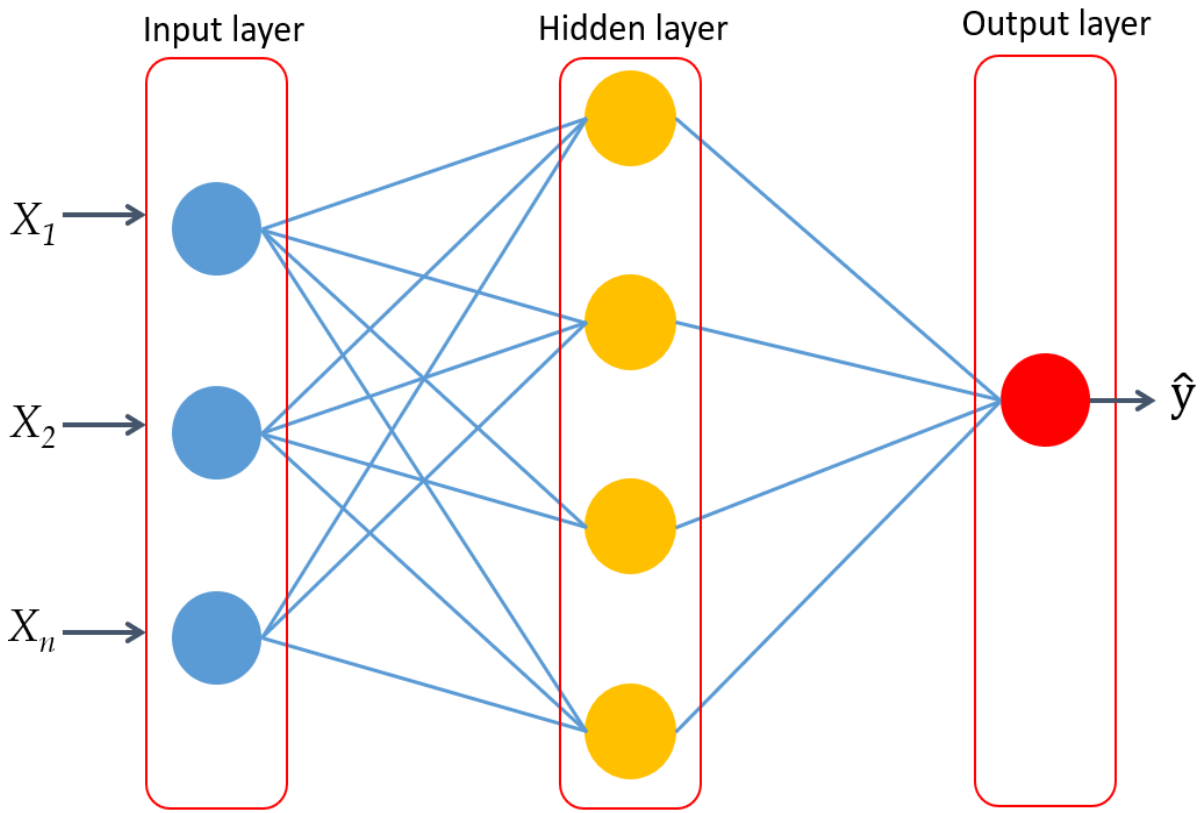
Pada permasalahan non-linear dan permasalahan yang lebih kompleks, menggunakan arsitektur **Multi-layer ANN**





Arsitektur Multi-layer ANN

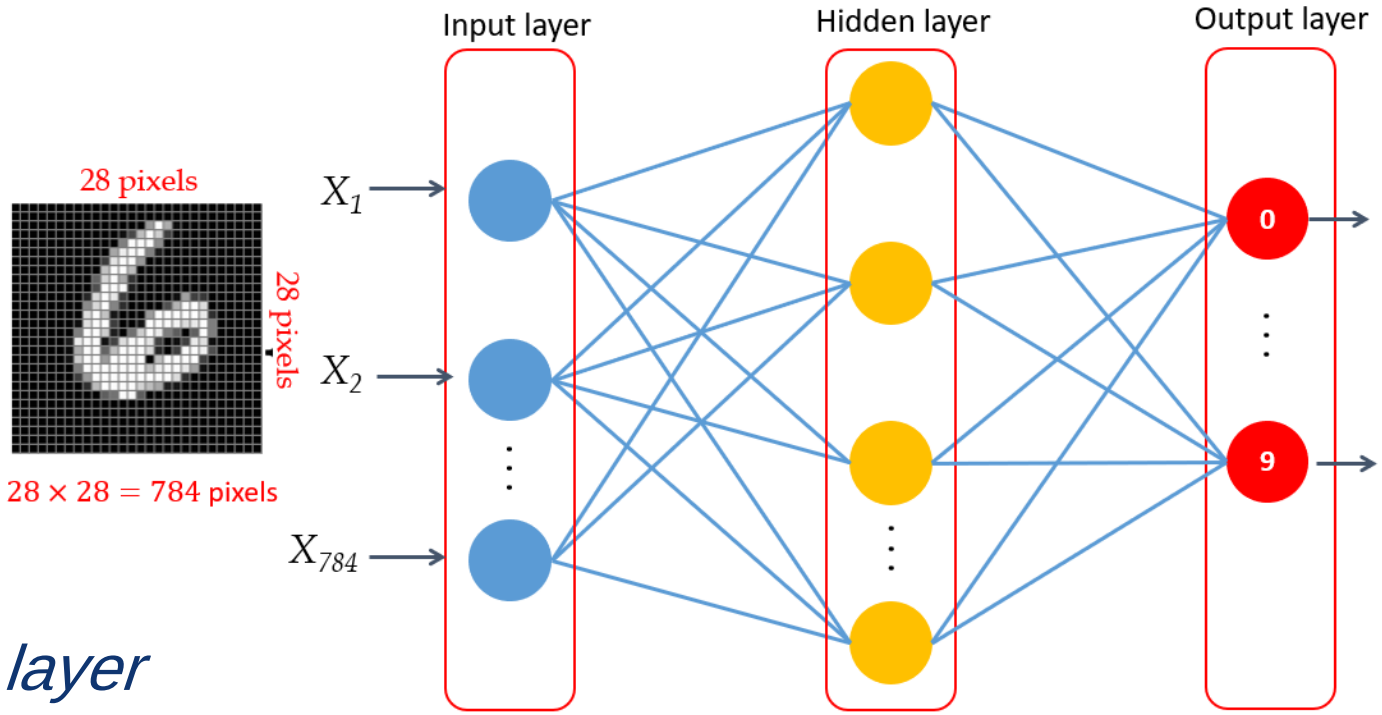
- Terdiri dari tiga layer yaitu:
  - input layer
  - hidden layer
  - output layer
- Hubungan antar neuron pada ANN merupakan ***fully connected network*** **(FCN)**
- **Jumlah *hidden layer*** sebaiknya disesuaikan dengan kompleksitas permasalahan
- **Jumlah neuron pada *hidden layer*** umumnya lebih banyak daripada jumlah *neuron* di *output layer*



Desain arsitektur ANN

Penentuan jumlah *neuron* pada *input layer*

- Jumlah neuron sesuai dengan jumlah fitur pada data input



Penentuan jumlah *neuron* pada *output layer*

- Jumlah neuron sesuai dengan permasalahan
- Pada permasalahan klasifikasi biner dan regresi bisa menggunakan hanya satu *neuron*
- Pada permasalahan klasifikasi *multiclass* menggunakan jumlah *neuron* sesuai jumlah label kelasnya, misalnya: 10 neuron pada pengenalan angka

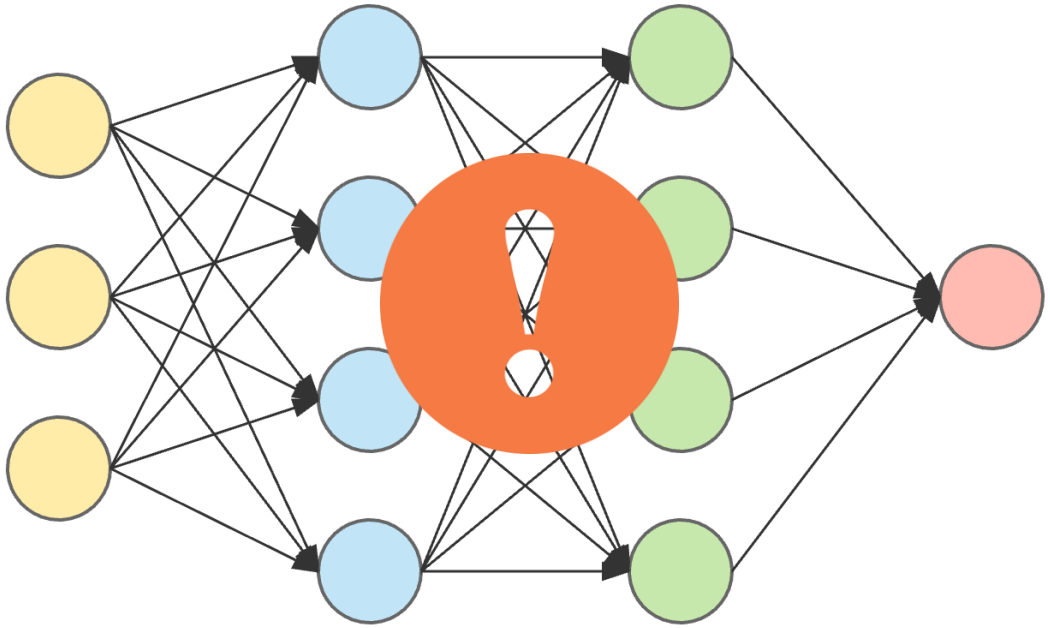
Desain arsitektur ANN

Penentuan jumlah *hidden layer*:

- Semakin banyak jumlah layer memerlukan komputasi waktu lebih lama
- Jumlah layer sebaiknya disesuaikan dengan kompleksitas permasalahan

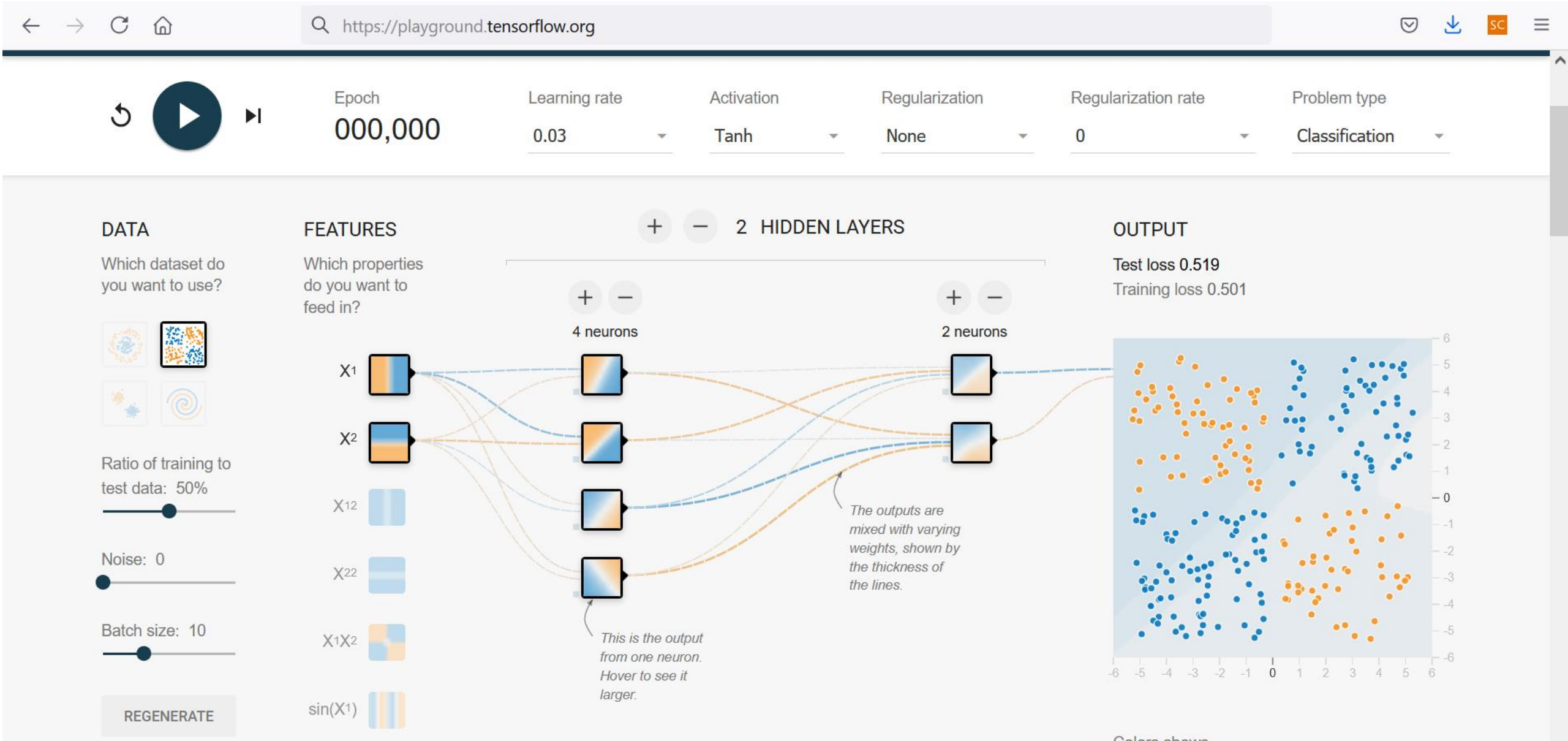
Penentuan jumlah node (neuron) pada *hidden laye*

- Semakin banyak jumlah node memungkinkan mempelajari pola yang lebih rumit
- Untuk mencegah *overfitting* sebaiknya menam jumlah node secara bertahap



Desain arsitektur ANN

https://playground.tensorflow.org/



Tahapan ANN: *Feed Forward*

1. Masukkan vektor *X* ke *input layer*
2. Hitung output setiap neuron di *hidden layer* dan *output layer*

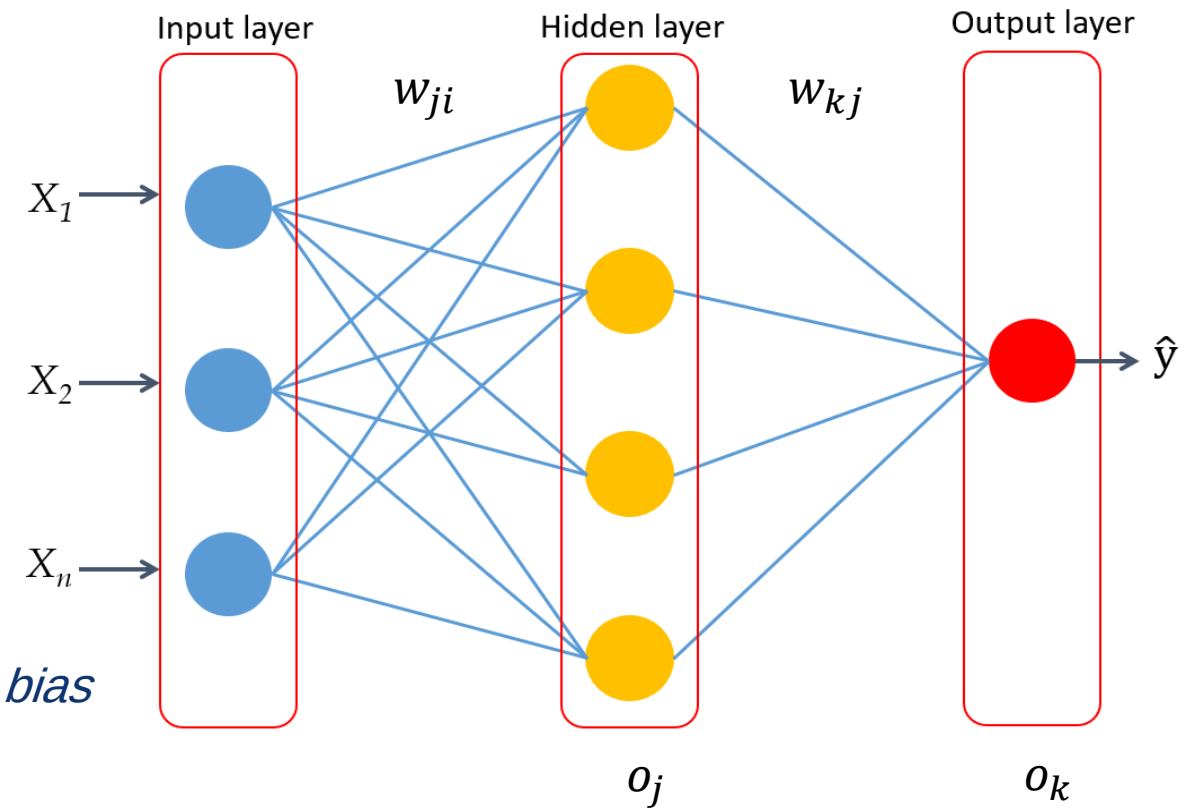
$$o_j = \sum_{i=1}^{n} w_{ji} x_i + b \qquad o_k = \sum_{j=1}^{m} w_{kj} o_j + b$$

dimana x_i dan o_j adalah matrik input dan output neuron pada layer sebelumnya

w_{ji} dan w_{kj} adalah bobot yang menghubungkan antara neuron pada dua layer *berbeda,* dan *b* adalah *bias*

n dan *m* adalah jumlah neuron di layer sebelumnya

3. Hitung fungsi aktivasi pada layer output



$$\hat{y} = a(o_k)$$

Tahapan ANN: Pembelajaran (*Learning*)

1. Inisialisasi bobot *W*

2. Update bobot sehingga output dari ANN adalah konsisten dengan label kelas pada data latih

   a. Fungsi obyektif (fungsi loss):

$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}; W\right)\right)^2$$

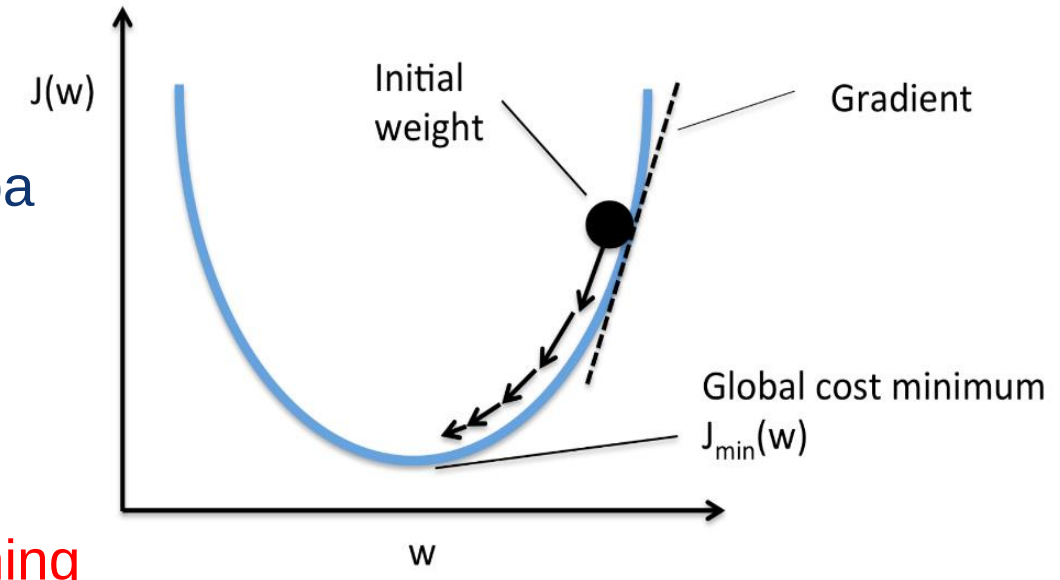
   y – target
   f – prediksi

   b. Menemukan bobot baru dengan **meminimalkan** fungsi obyektif, contoh: **algoritma *backpropagation***

Algoritma Pembelajaran (*Learning*)

Untuk merancang algoritma pembelajaran, ada beberapa hal yang perlu diperhatikan:

1. Kriteria Iterasi berhenti? konvergen, iterasi (epoch)
2. Bagaimana direction? gradient descent
3. Berapa banyak (step) yang diperlukan? nilai learning rate (η)



Algoritma Backpropagation Gradient Descent

- Inisialisasi bobot secara random
- Melakukan iterasi sampai konvergen atau maksimum iterasi
  - Hitung gradient, $\frac{\partial J(W)}{\partial W}$
  - Update bobot, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$
- Mengembalikan nilai bobot

$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

$$J(W) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}; W\right)\right)^2$$

Nilai *learning rate* jika terlalu kecil memerlukan waktu lebih lama untuk konvergen, jika terlalu besar membuat model tidak stabil

Algoritma Pembelajaran (*Learning*)

Algoritma Stochastic Gradient Descent

- Inisialisasi bobot secara random
- Melakukan iterasi sampai konvergen atau maksimum iterasi
  - Baca **setiap** data poin *i*
  - Hitung gradient, $\frac{\partial J_i(W)}{\partial W}$
  - Update bobot, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$
- Mengembalikan nilai bobot

- Inisialisasi bobot secara random
- Melakukan iterasi sampai konvergen atau maksimum iterasi
  - Baca ***batch* B** data poin
  - Hitung gradient, $\frac{\partial J(W)}{\partial W} = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial J_k(W)}{\partial W}$
  - Update bobot, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$
- Mengembalikan nilai bobot

Tahapan Pembelajaran Multi-layer Perceptron ANN

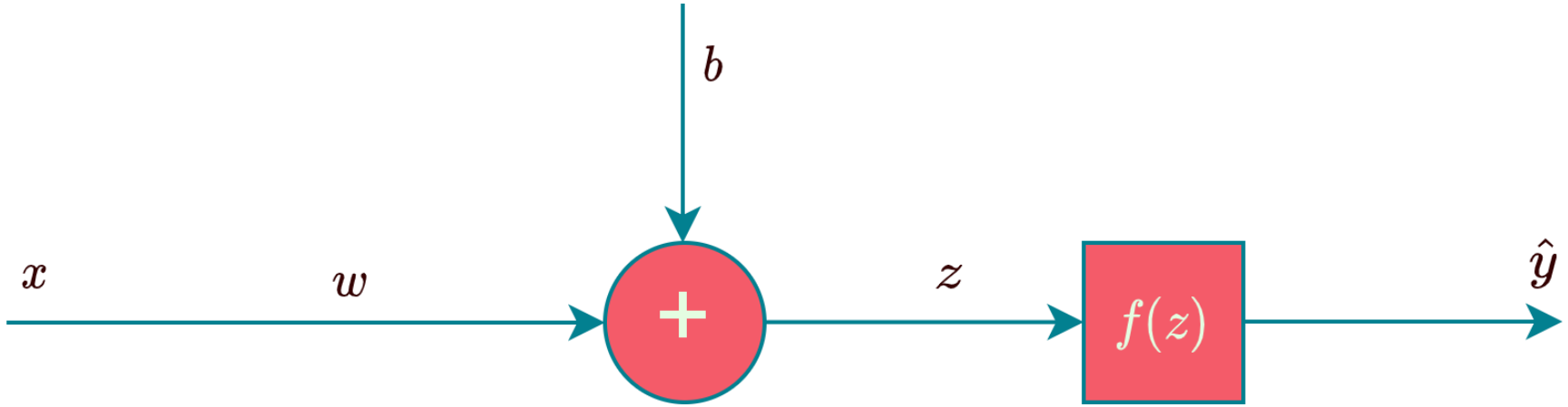
Langkah 0 – Inisialisasi bobot, learning rate, maksimum iterasi
Langkah 1 – Membaca vektor input *X*
Langkah 2 – Lakukan iterasi (*epoch*)
Langkah 3 – Hitung luaran neuron di hidden layer dan output layer
Langkah 4 – Hitung *back propagate error* (pada output layer dan hidden layer)
Langkah 5 – Perbarui semua bobot (pada output layer dan hidden layer)
Langkah 6 – Ulangi langkah 3 – 5 hingga bobot konvergen atau maksimum iterasi
Langkah 7 – Luaran berupa matrik bobot (pada output layer dan hidden layer)

https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/

Tahapan implementasi ANN

- Load data: membaca file data input
- Split data: membagi data menjadi data latih, data validasi, data uji
- Define model: merancang arsitektur atau model ANN
- Compile model: menjalankan model ANN yang sudah dirancang
- Fit model: membangun model ANN berdasarkan data latih
- Evaluation model: mengevaluasi model ANN berdasarkan data validasi
- Save model: menyimpan model ANN
- Prediction: memprediksi output dari data uji menggunakan model ANN yang terbaik

Single Input dan Single Output ANN



Assume activation function used is pure linear activation →

$$y_t = \hat{y} = f(z) = z$$

$$\hat{y}_i = f(z) = f(wx_i + b)$$
$$e_i = \hat{y}_i - y_i$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} e_i^2 = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

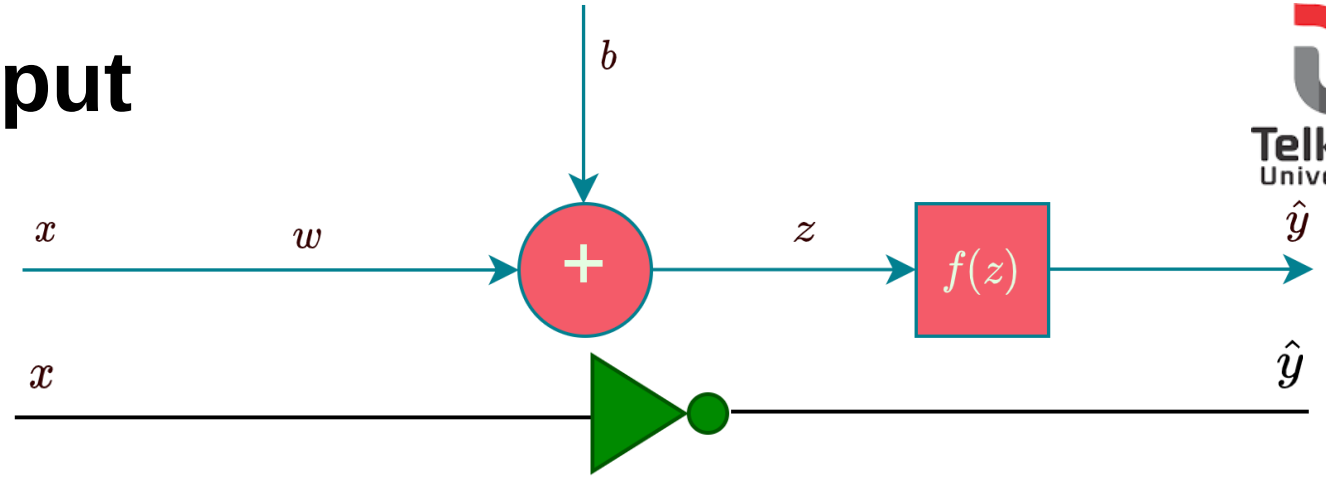
Update weight and bias : (backpropagation)

$$w_{j+i} = w_j - \lambda\frac{dMSE}{dw} = w_j - \frac{\lambda}{N}\sum_{i=1}^{N} 2e_i x_i$$

$$b_{j+i} = b_j - \lambda\frac{dMSE}{db} = b_j - \frac{\lambda}{N}\sum_{i=1}^{N} 2e_i$$

N = number of row dataset
J = epoch index (iteration)
λ = learning rate

Single Input dan Single Output ANN use case (negation)

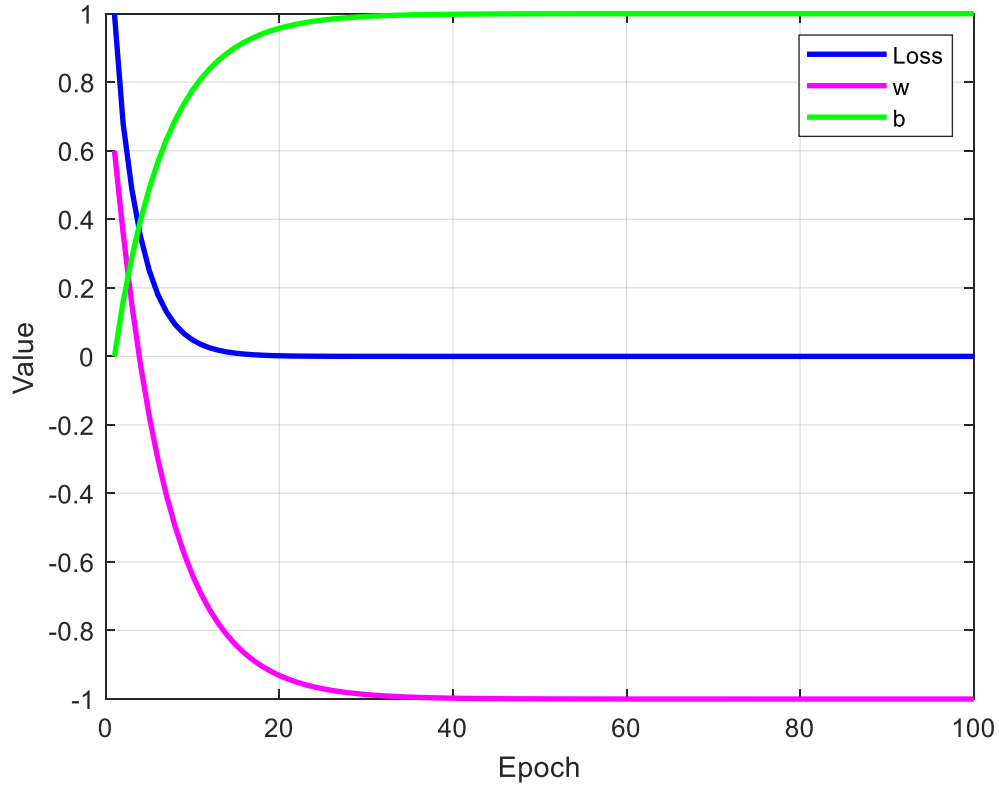


Case : Negation Gate (Not)

Input (x)	Output (y)
1	0
0	1

Target : Calculate w and b to set the output to reach desired output from dataset

Result : Convergence when w=-1 dan b=1
Proved using formula ANN SISO y=wx+b using dataset above !



Multi Input dan Multi Output ANN No Hidden Layer

Assume activation function used is pure linear activation → $y_t = \hat{y} = f(z) = z$

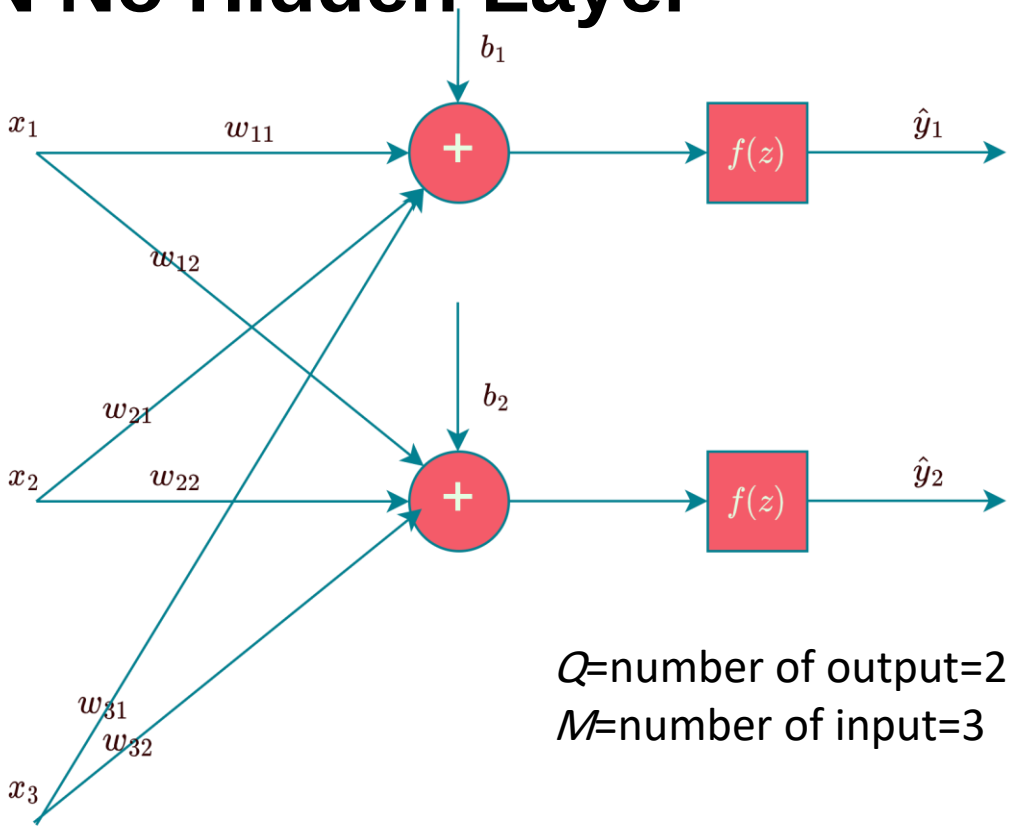
$$\hat{y}_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + b_1$$

$$\hat{y}_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + b_2$$

Simplified by matrix operation :

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

For *M* input and *Q* output :



Q=number of output=2
M=number of input=3

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_Q \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} & \dots & w_{M1} \\ w_{12} & w_{22} & \dots & w_{M2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1Q} & w_{2Q} & \dots & w_{MQ} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_Q \end{bmatrix}$$

Generally for all epoch :

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_Q \end{bmatrix}_j = \begin{bmatrix} w_{11} & w_{21} & \dots & w_{M1} \\ w_{12} & w_{22} & \dots & w_{M2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1Q} & w_{2Q} & \dots & w_{MQ} \end{bmatrix}_j \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}_j + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_Q \end{bmatrix}_j$$

Backpropagation in MIMO ANN no hidden layer

Backpropagation formula :

$$(w_{mk})_{j+i} = (w_{mk})_j - \lambda \frac{dMSE}{dw_{mk}} = (w_{mk})_j - \frac{\lambda}{N} \sum_{i=1}^{N} 2e_{ki}x_{ki}$$

$$(b_k)_{j+i} = (b_k)_j - \lambda \frac{dMSE}{db_k} = (b_k)_j - \frac{\lambda}{N} \sum_{i=1}^{N} 2e_{ki}$$

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{Q} e_{ki}^2 = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{Q} (\hat{y}_{ki} - y_{ki})^2$$

$$\hat{y}_{ki} = \sum_{m=1}^{M} x_{mi} w_{mk} + b_k$$

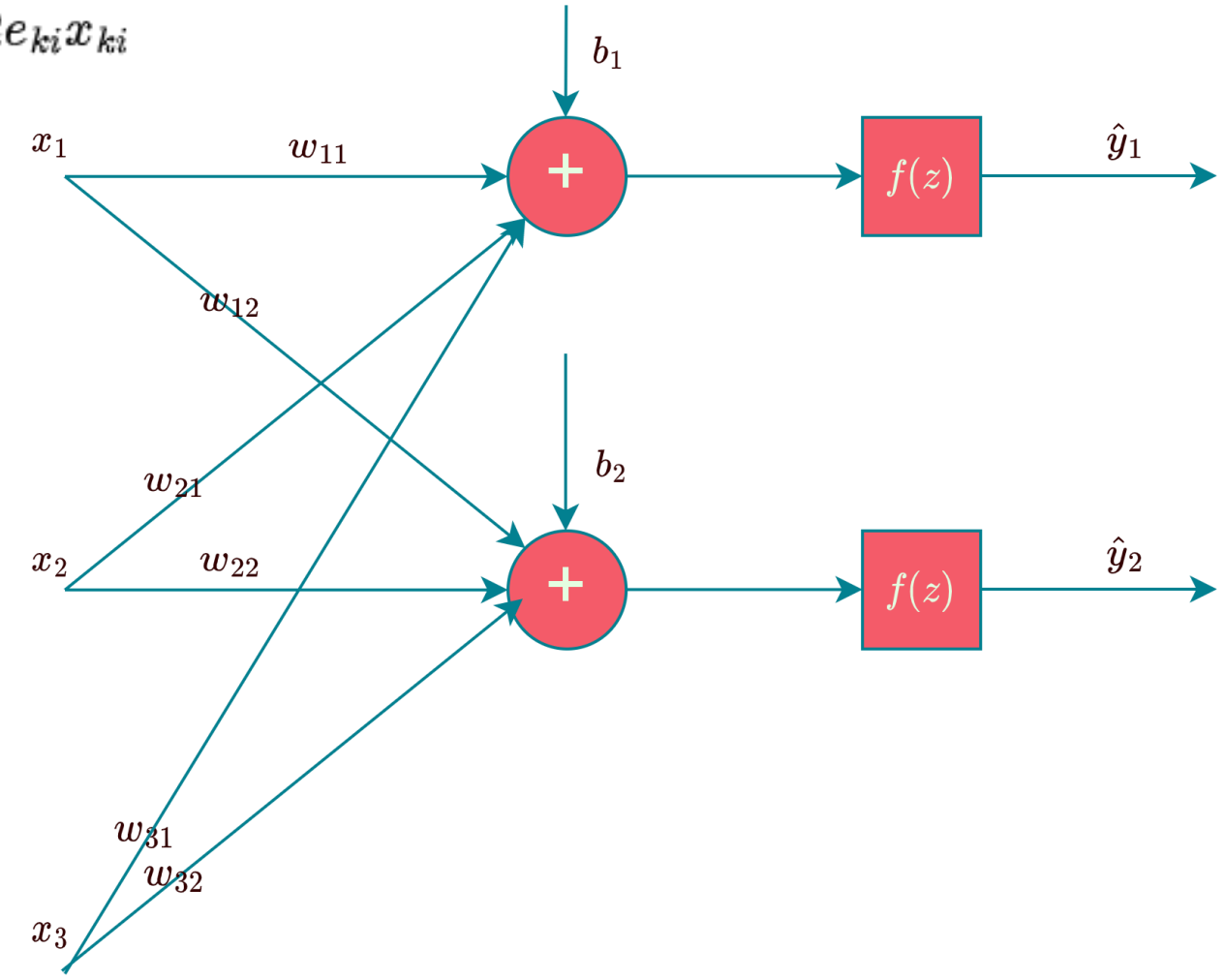
N = Jumlah input

Q = Jumlah output

n = index input

k = index output

i = index baris dataset

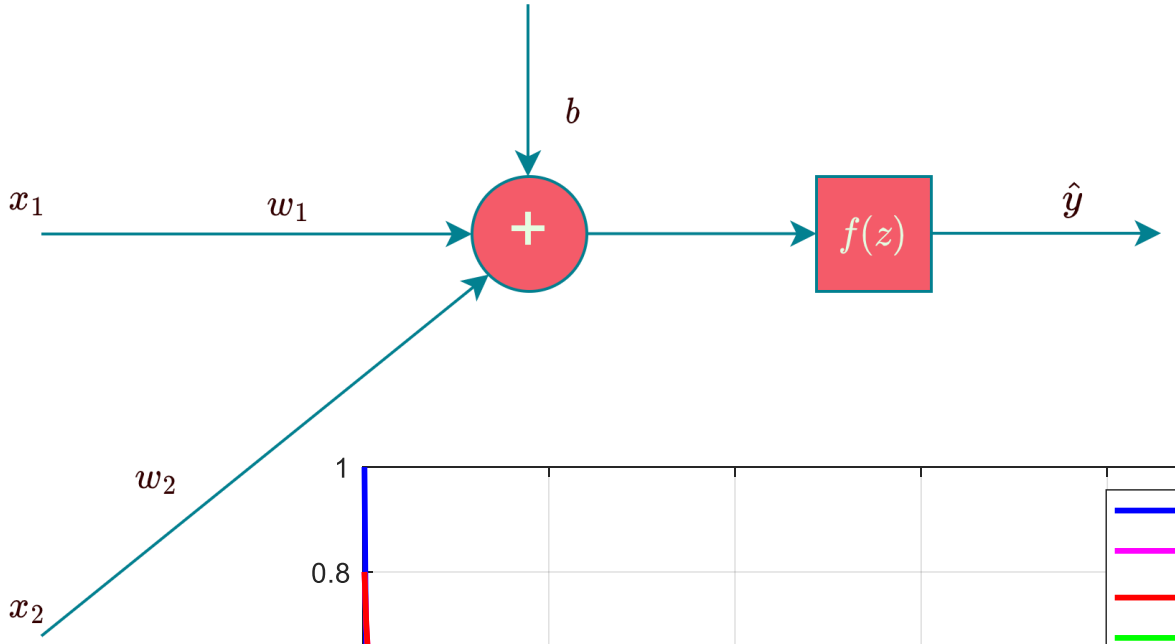


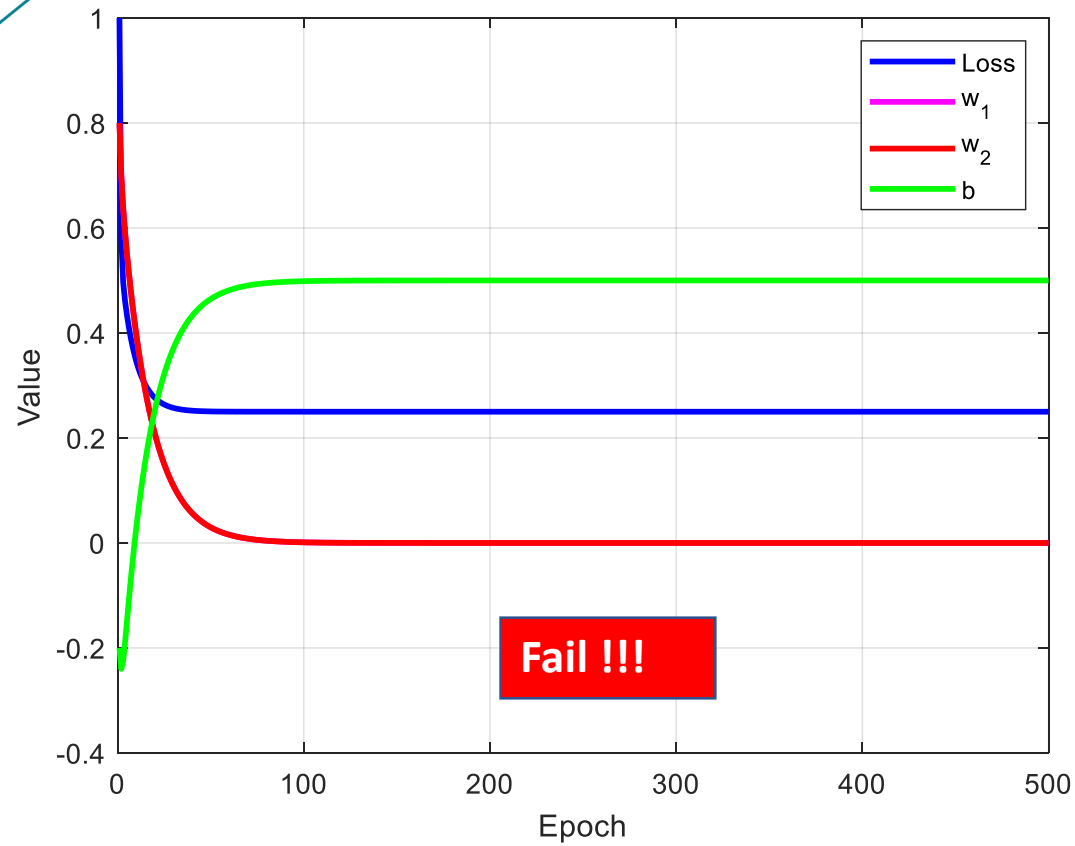
MISO ANN use case (1)

Input 1 (x1)	Input 2 (x2)	Output y=xor(x1,x2)
1	1	0
1	0	1
0	1	1
0	0	0

$$\hat{y} = x_1 w_1 + x_2 w_2 + b$$

$$w_1 = ?, \quad w_2 = ?, \quad b = ?$$



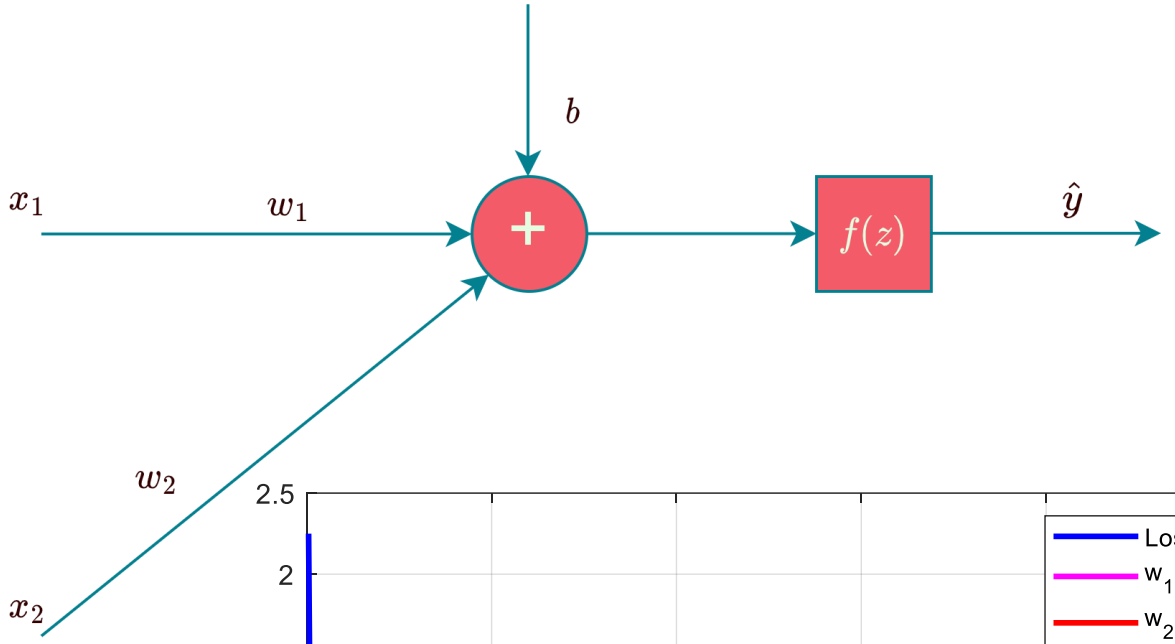


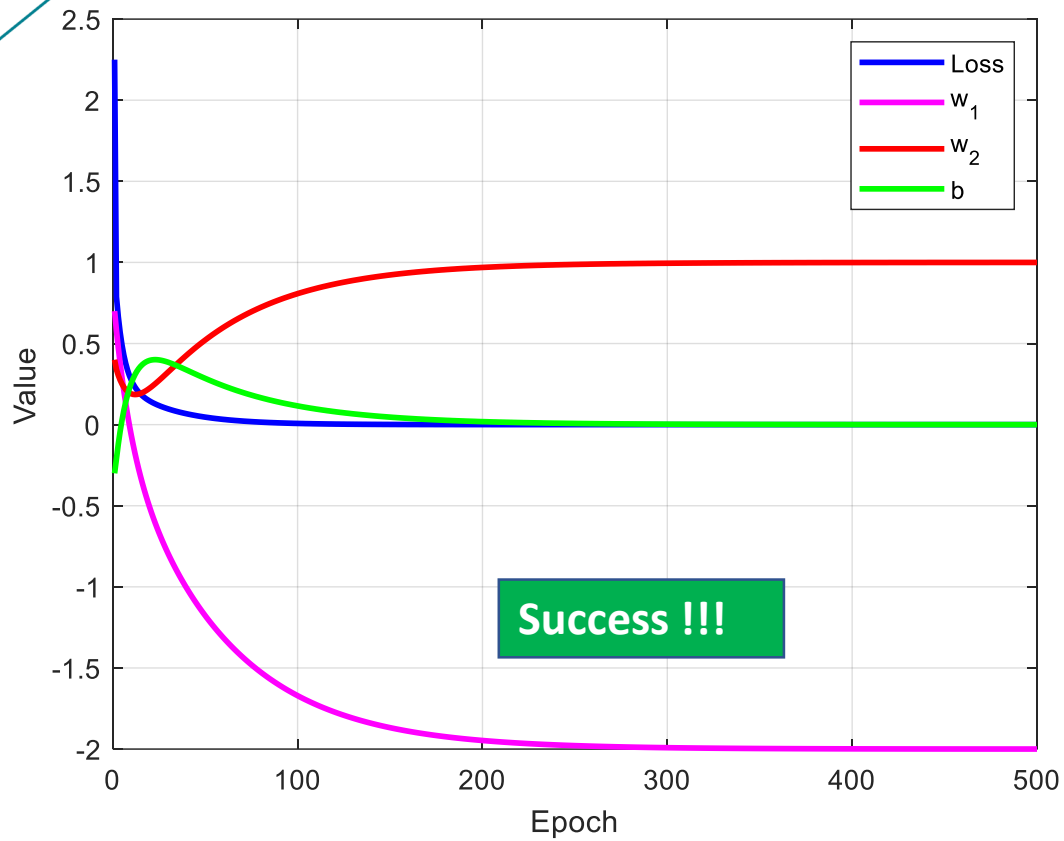
MISO ANN use case (2)

Input 1 (x1)	Input 2 (x2)	p1=x1 *x2	p2=x1+ x2	Output y=xor(x1,x2)
1	1	1	2	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

$$\hat{y} = p_1 w_1 + p_2 w_2 + b$$

$$w_1 = ?, \quad w_2 = ?, \quad b = ?$$





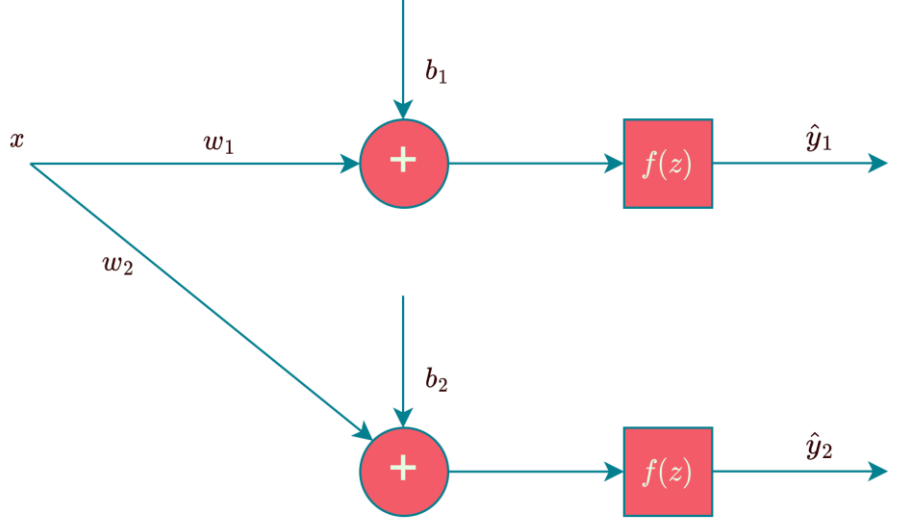
SIMO ANN use case

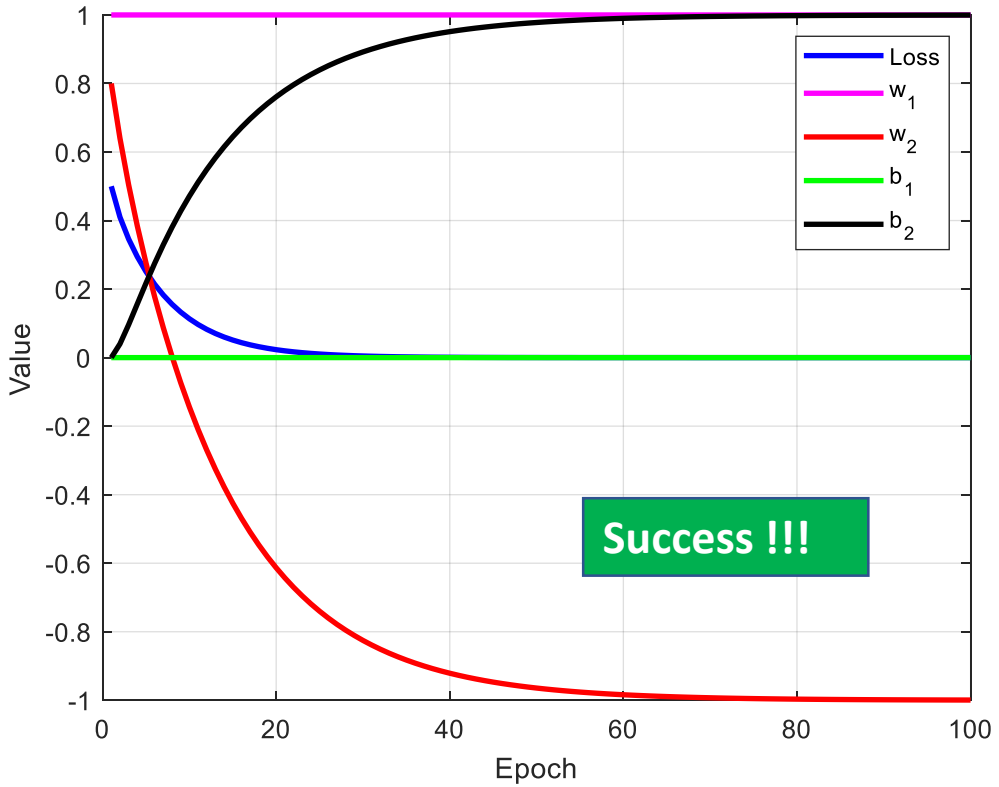
Input (x)	Output 1 (y1=x)	Output2 y2=~x
1	1	0
1	1	0
0	0	1
0	0	1

$$\hat{y}_1 = w_1 x + b_1$$

$$\hat{y}_2 = w_2 x + b_2$$

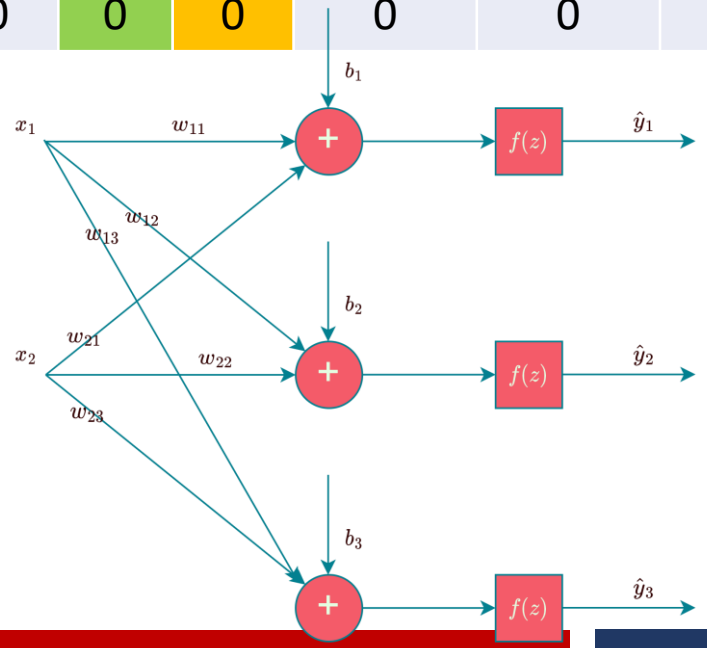
$$w_1 = ?, \quad w_2 = ?, \quad b_1 = ?, \quad b_2 = ?$$



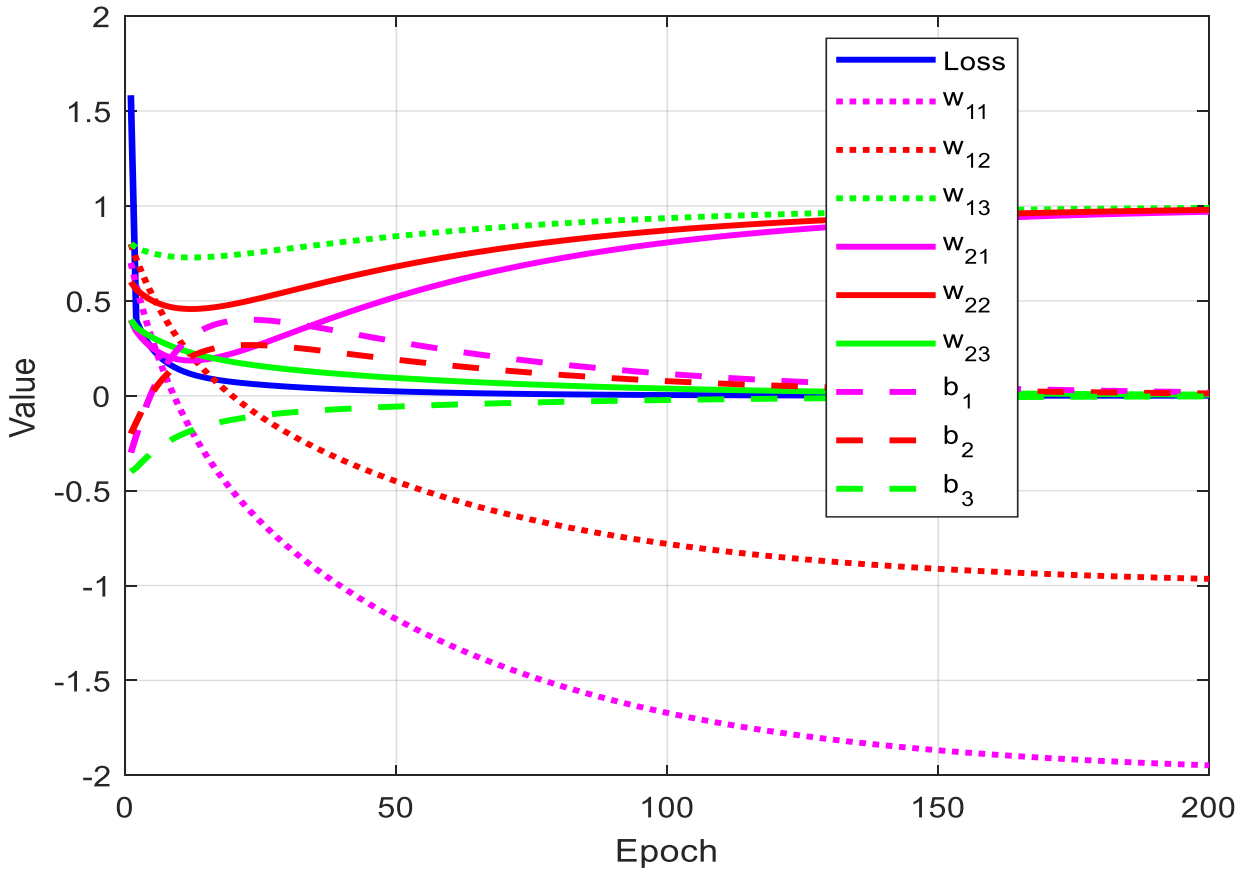


MIMO ANN Use Case

Input 1 (x1)	Input 2 (x2)	p1= x1* x2	p2= x1+ x2	Output 1 y=xor(x1,x2)	Output 2 y2=or(x1,x2)	Output 3 y3=and(x1,x2)
1	1	1	2	0	1	1
1	0	0	1	1	1	0
0	1	0	1	1	1	0
0	0	0	0	0	0	0



$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$



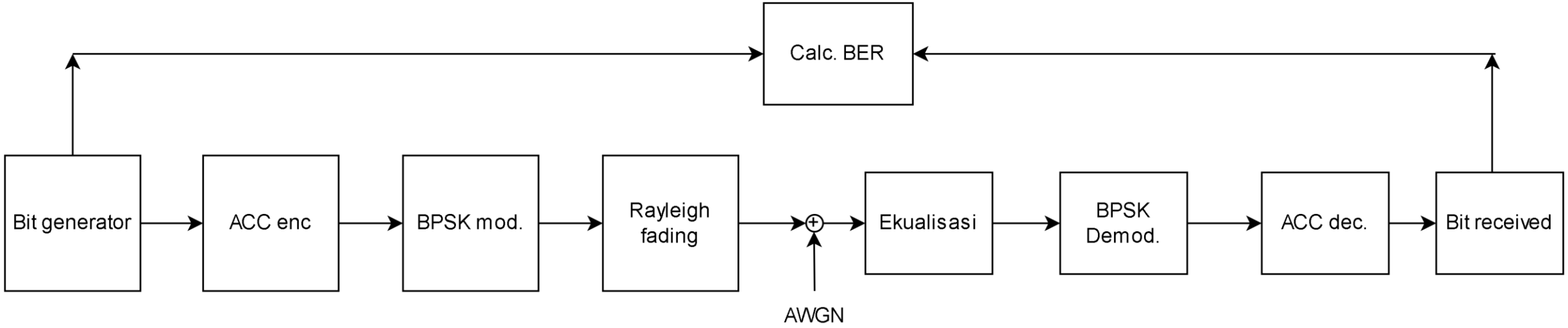
Project 1

1. Buatlah program script python untuk mentraining dataset berikut,

Input 1 (x1)	Input 2 (x2)	p1= x1* x2	p2= x1+ x2	Output 1 y=xor(x1,x2)	Output 2 y2=or(x1,x2)	Output 3 y3=and(x1,x2)
1	1	1	2	0	1	1
1	0	0	1	1	1	0
0	1	0	1	1	1	0
0	0	0	0	0	0	0

1. Plot konvergensi loss nya
2. Hitung berapa semua bobot nya saat konvergen
3. Berapa semua bias nya saat konvergen
4. Jika loss tidak convergen di nol dengan input x1 dan x2, gunakan input p1 dan p2 !

ANN Use Case for Wireless Comm. With Accumulator Encoder



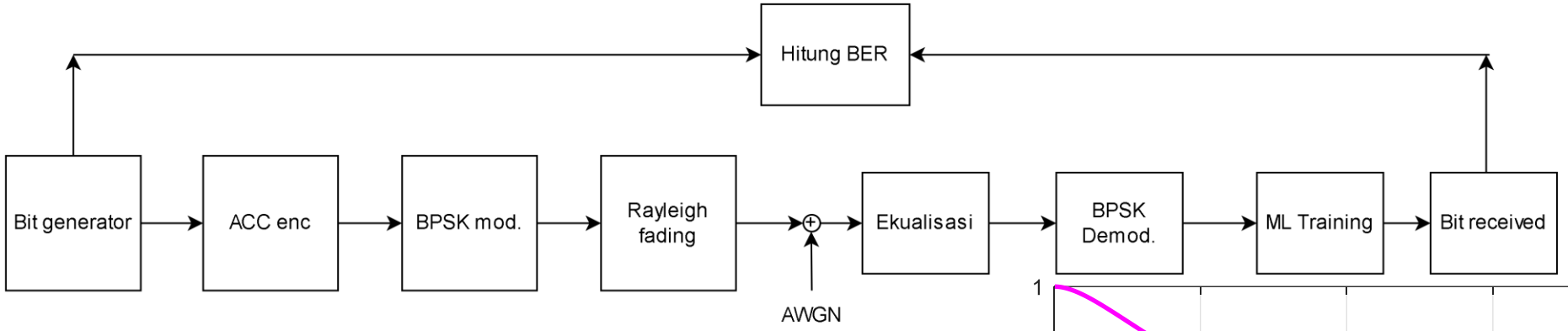
How to Design AI replacing the decoder above?

1. Replace only ACC dec.?
2. Replace BPSK demod and ACC dec.?
3. Replace Equalization, BPSK Demod and ACC dec.?
4. SISO/MIMO?

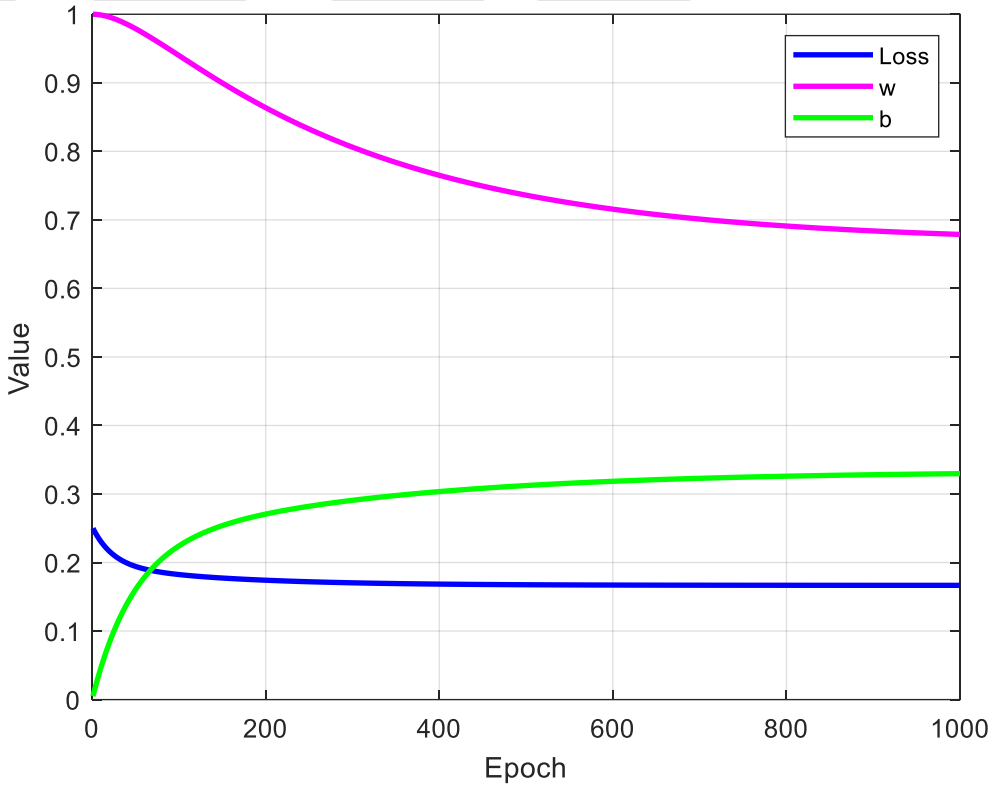
Answer : Set for all cases y as bit generator output

1. Set x as BPSK demod. output
2. Set x as equalizer output
3. Set x as awgn output
4. For SISO ANN, x and y as a column matrix, for MIMO ANN, x and y as a row matrix

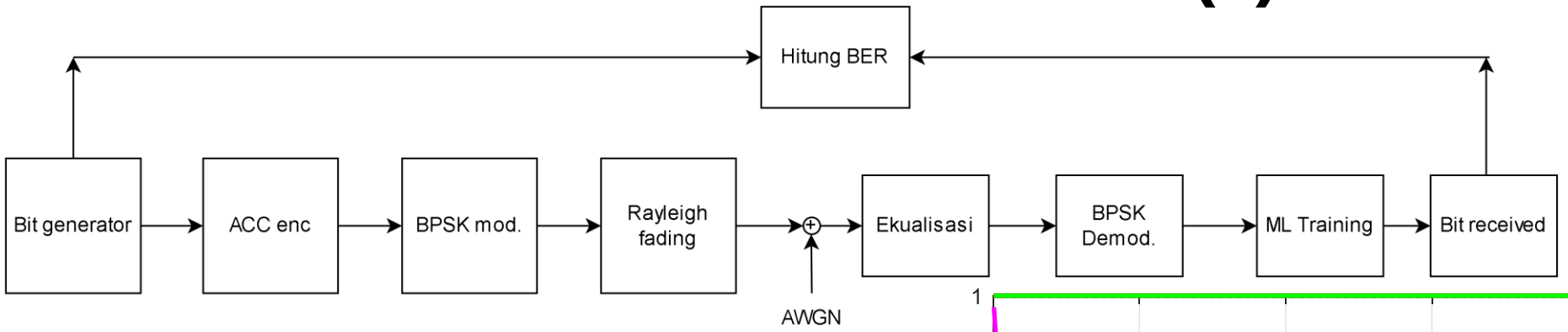
ANN Use Case for Wireless Comm. With Accumulator Encoder – SISO ANN case



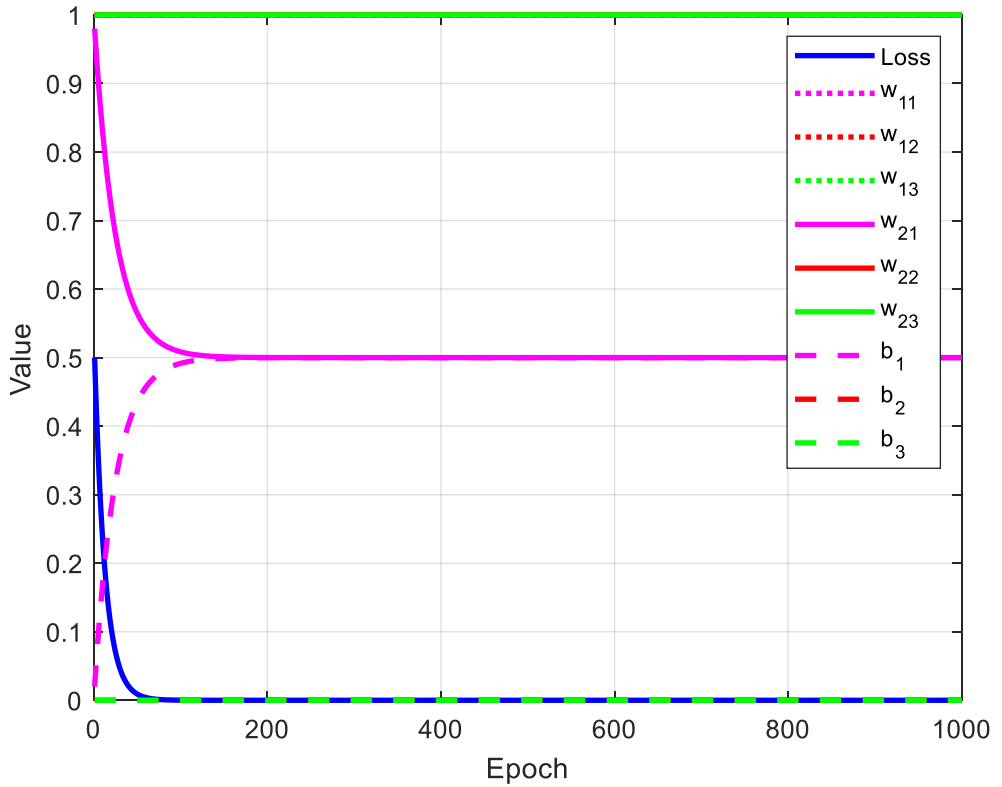
- Lamda=0.01, epoch=1000
- x and y is column matrix
- X is output of bpsk demod
- For SISO ANN case, error has never gone to 0 !!!
- Moreover in the case of ML replacing BPSK demodulation and equalizer
- SNR=10 dB



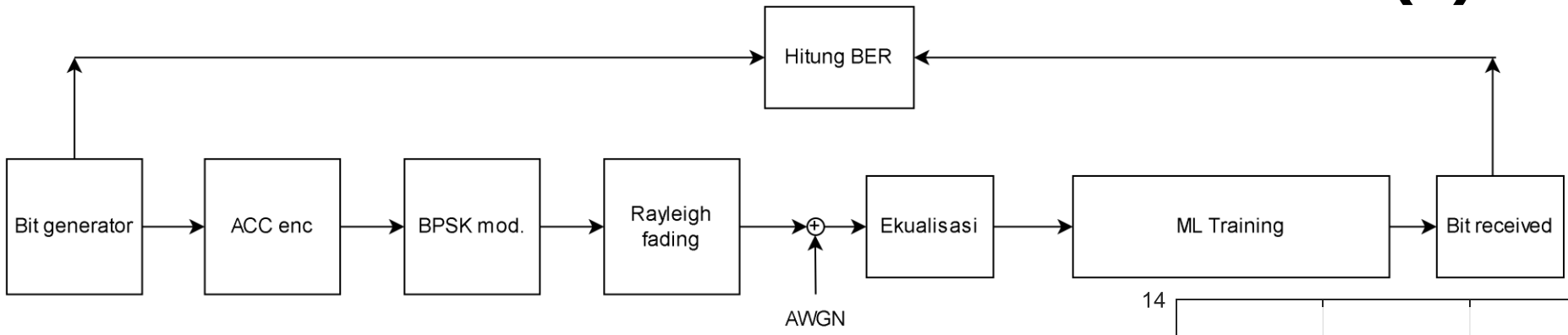
ANN Use Case for Wireless Comm. With Accumulator Encoder – MIMO ANN case (1)



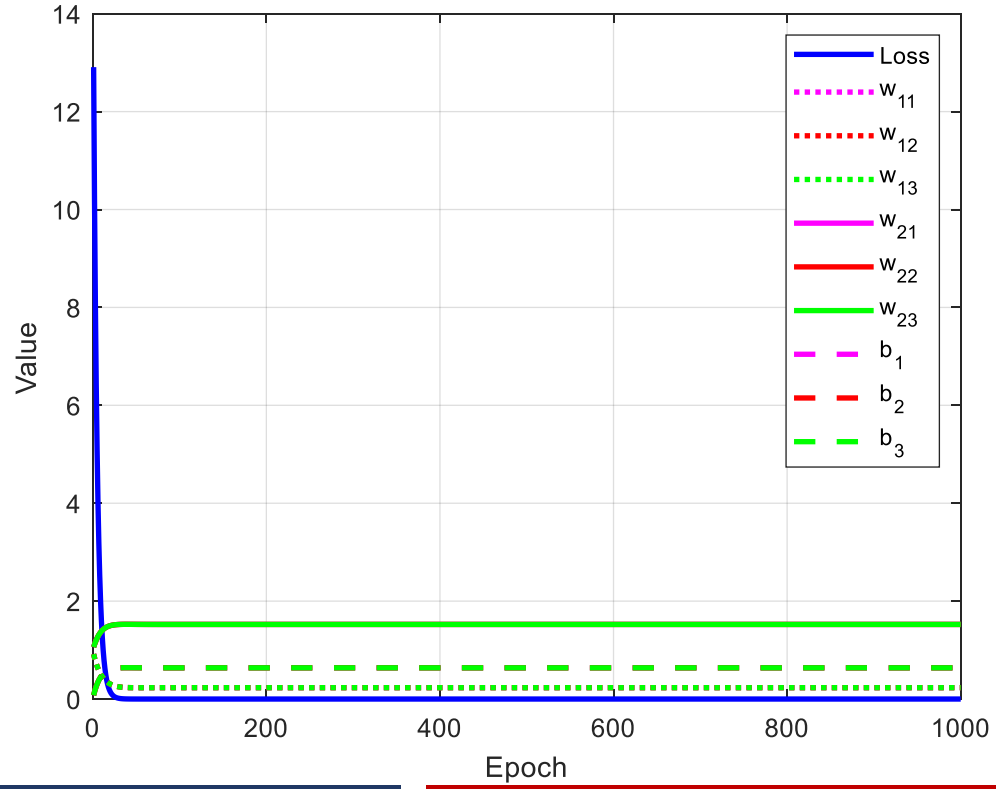
- Lamda=0.01, epoch=1000
- x and y is row matrix
- x is output of BPSK demod
- For MIMO ANN case, error is reaching 0, hurray !!!
- Only 4 bits transmitted
- SNR=10 dB



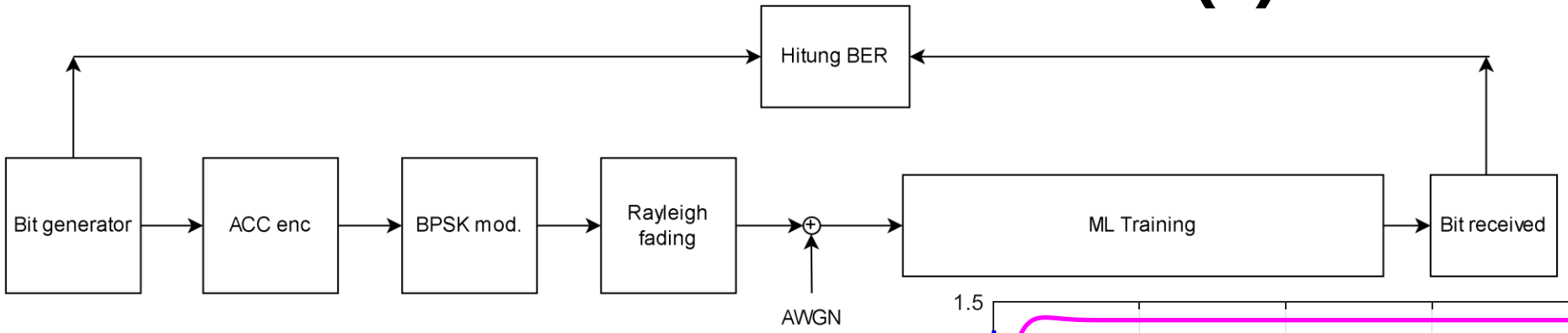
ANN Use Case for Wireless Comm. With Accumulator Encoder – MIMO ANN case (2)



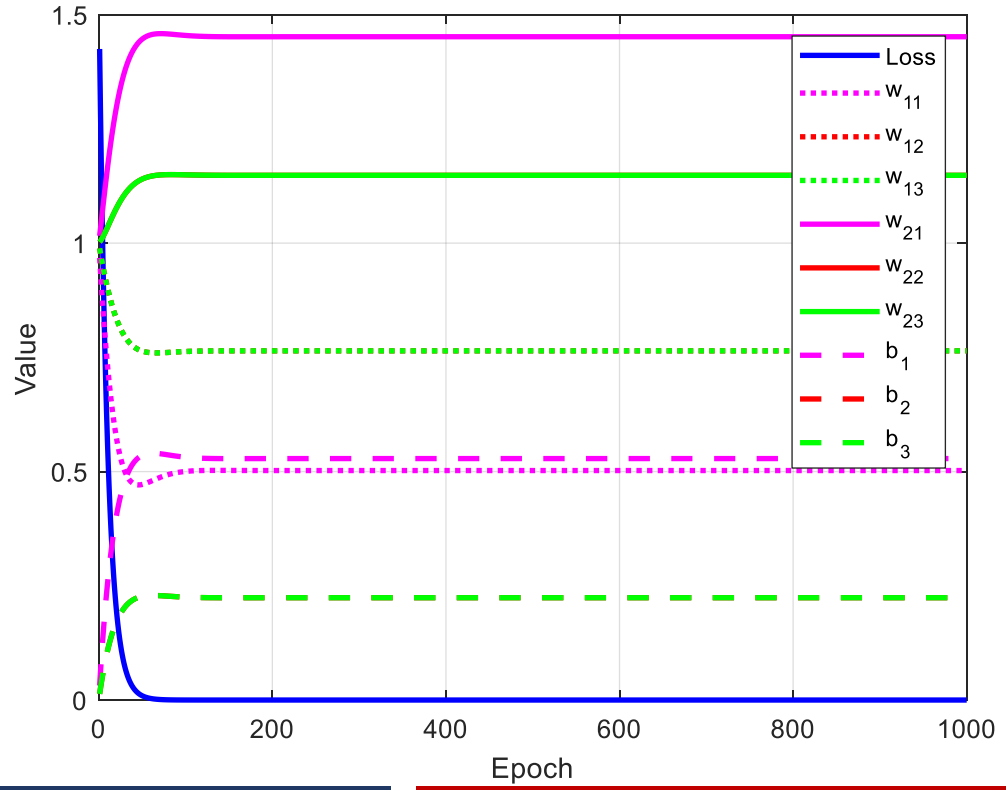
- Lamda=0.01, epoch=1000
- x and y is row matrix
- x is output of equalizer
- For MIMO ANN case, error is reaching 0, hurray !!!
- Only 4 bits transmitted
- SNR=10 dB



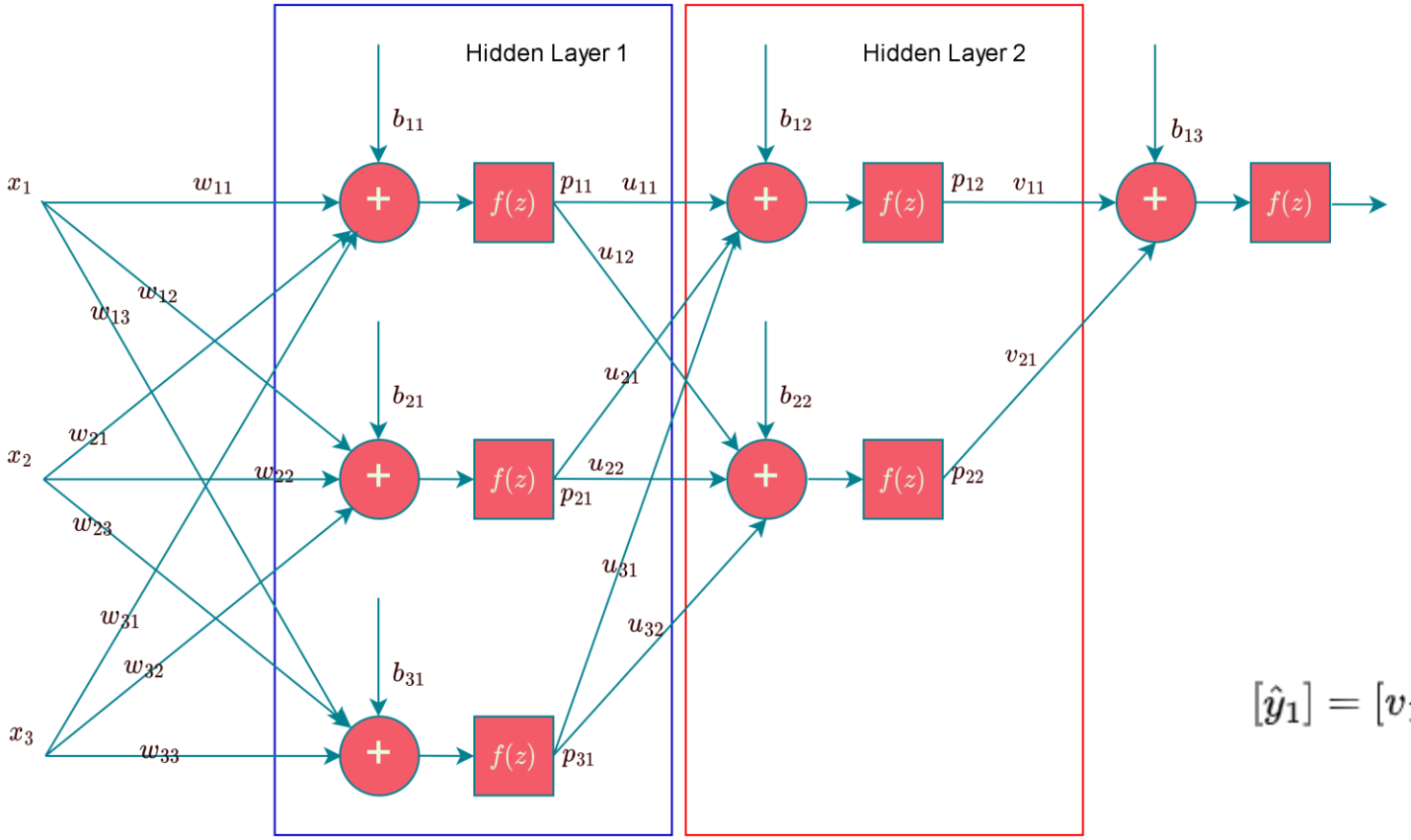
ANN Use Case for Wireless Comm. With Accumulator Encoder – MIMO ANN case (3)



- Lamda=0.01, epoch=1000
- x and y is row matrix
- x is output of AWGN
- For MIMO ANN case, error is reaching 0, hurray !!!
- Only 4 bits transmitted
- SNR=10 dB



ANN MIMO Multi Layer with Hidden Layer



$$\begin{bmatrix} p_{11} \\ p_{21} \\ p_{31} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \end{bmatrix}$$

$$\begin{bmatrix} p_{12} \\ p_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{21} \\ p_{31} \end{bmatrix} + \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix}$$

$$[\hat{y}_1] = [v_{11} \quad v_{21}] \begin{bmatrix} p_{12} \\ p_{22} \end{bmatrix} + [b_{13}]$$

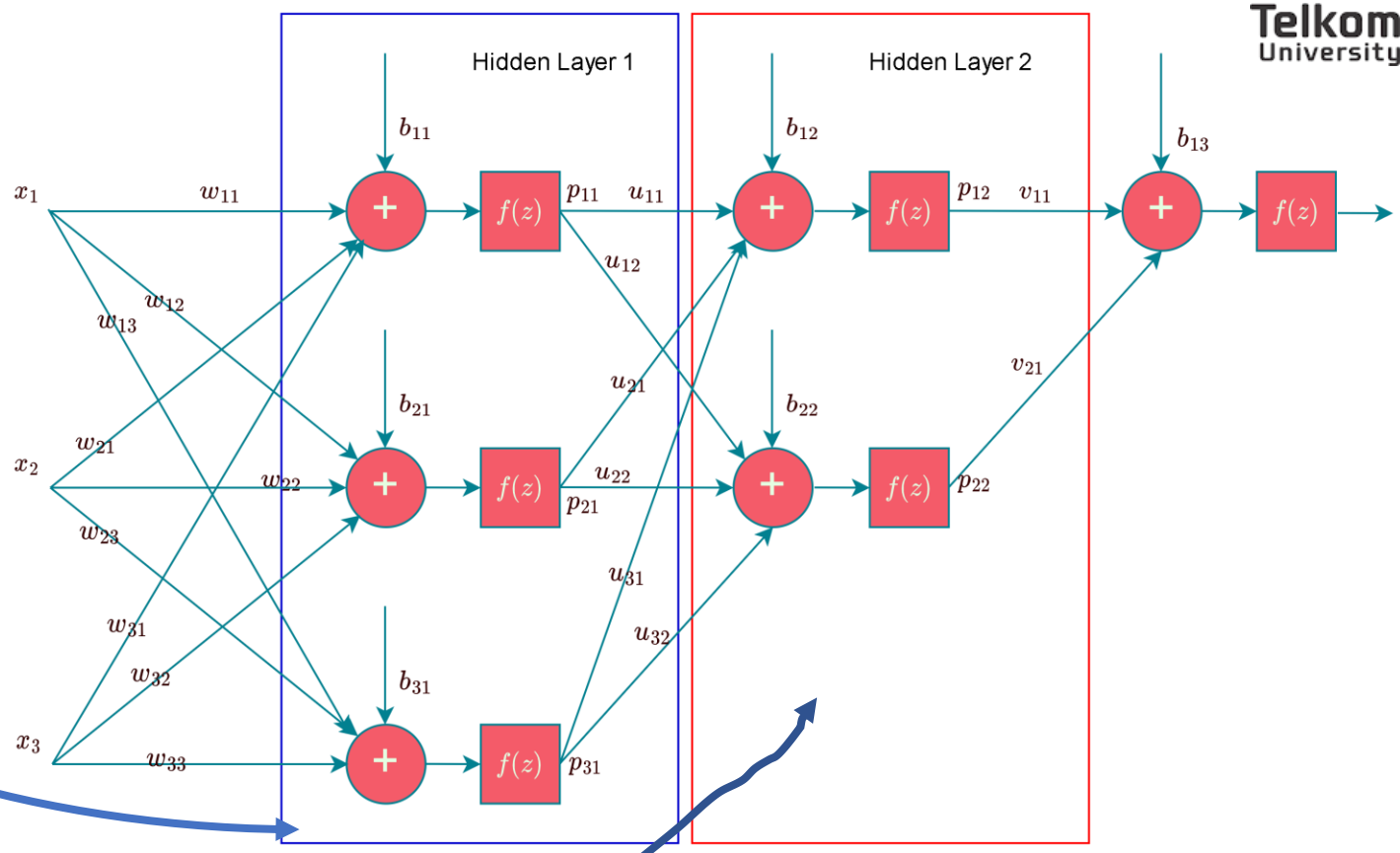
$$[\hat{y}_1] = [v_{11} \quad v_{21}] \left(\begin{bmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{21} \\ p_{31} \end{bmatrix} + \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix} \right) + [b_{13}]$$

$$[\hat{y}_1] = [v_{11} \quad v_{21}] \left(\begin{bmatrix} u_{11} & u_{21} & u_{31} \\ u_{12} & u_{22} & u_{32} \end{bmatrix} \left(\begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \end{bmatrix} \right) + \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix} \right) + [b_{13}]$$

$1 \times 2 \qquad 2 \times 3 \qquad 3 \times 3 \qquad 3 \times 1$

ANN MIMO Multi Layer Case

Input 1 (x1)	Input 2 (x2)	p1=	p2=	Output 1 y=xor(x1,x2)	Output 2 y2=or(x1,x2)	Output 3 y3=and(x1,x2)
1	1	1	2	0	1	1
1	0			1	1	0
0	1			1	1	0
0	0	0	0	0	0	0



Performance : Accuracy

Menunjukkan persentase klasifikasi yang bernilai valid terhadap total klasifikasi yang dilakukan.

$$a = \frac{t}{n} \times 100\%$$

dengan
a adalah akurasi dalam persen,
t adalah jumlah percobaan dengan prediksi valid, dan
n adalah jumlah percobaan

Contoh: Berapa akurasi dari percobaan di samping ini?

Data aktual	Output model (prediksi)	Kesimpulan
mangga	mangga	valid
jeruk	apel	invalid
apel	jeruk	invalid
mangga	apel	invalid
jeruk	jeruk	valid

Contoh

Data aktual	Output model (prediksi)	Kesimpulan
mangga	mangga	valid
jeruk	apel	invalid
apel	jeruk	invalid
mangga	apel	invalid
jeruk	jeruk	valid

Jumlah percobaan valid (t) = 2

Jumlah percobaan invalid = 3

Total Percobaan (a) = 5

$$a = \frac{t}{n} \times 100\%$$

$$= \frac{2}{5} \times 100\%$$

$$= 40\%$$

Akurasi dapat digunakan sebagai ukuran awal mengevaluasi model, namun tidak cukup dengan akurasi saja. Terkadang akurasi tiap kelas perlu diketahui juga.

Confusion matrix

*Bukan metric, namun bermanfaat melihat **sebaran validitas percobaan***

		Kelas Prediksi			
		mangga	apel	jambu	pear
Kelas Aktual	mangga	19	3	2	1
	apel	1	22	1	1
	Jambu	2	2	21	0
	Pear	0	1	1	23

Karakteristik:

- *Ada sumbu data aktual dan sumbu data prediksi (gunakan konvensi)*
- *Setiap kelas terpetakan satu sama lainnya*
- *Percobaan valid berada pada diagonal utama*
- *Matriks berbentuk bujur sangkar*

Confusion matrix

		Kelas Prediksi			
		mangga	apel	jambu	pear
Kelas Aktual	mangga	19	3	2	1
	apel	1	22	1	1
	Jambu	2	2	21	0
	Pear	0	1	1	23

Pada area kotak merah:

terdapat 25 mangga yang di uji, dengan 19 mangga dikenali sebagai mangga (valid), 3 mangga dikenali sebagai apel 9(invalid), 2 mangga dikenali sebagai jambu (invalid), dan 1 mangga dikenali sebagai pear (invalid)

Confusion matrix

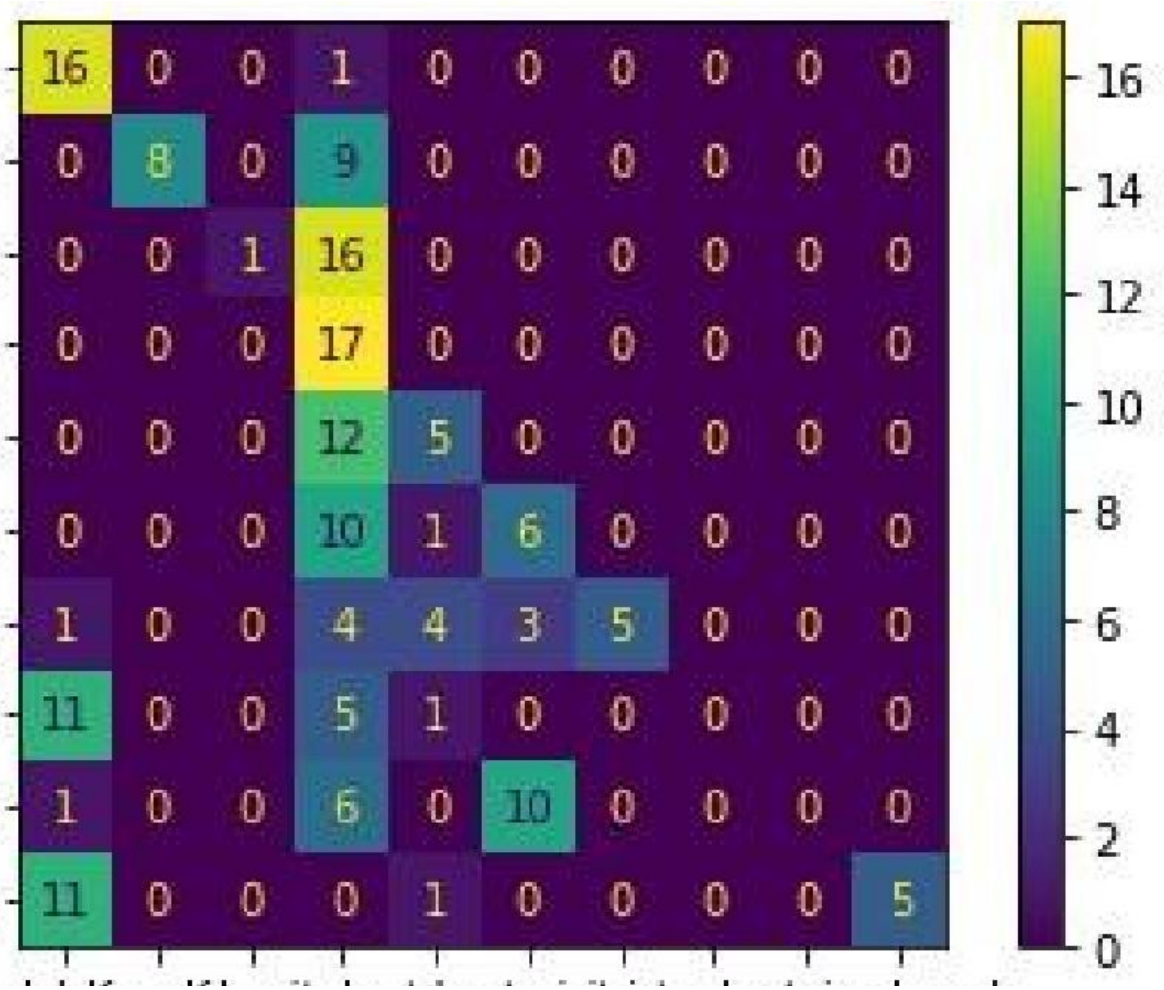
		Kelas Prediksi			
		mangga	apel	jambu	pear
Kelas Aktual	mangga	19	3	2	1
	apel	1	22	1	1
	Jambu	2	2	21	0
	Pear	0	1	1	23

Akurasi untuk pengujian kelas mangga adalah : 19/25 x 100% = 76%

Sedangkan akurasi total pengujian adalah : (19+22+21+23)/(25+25+25+25) x 100% = 85%

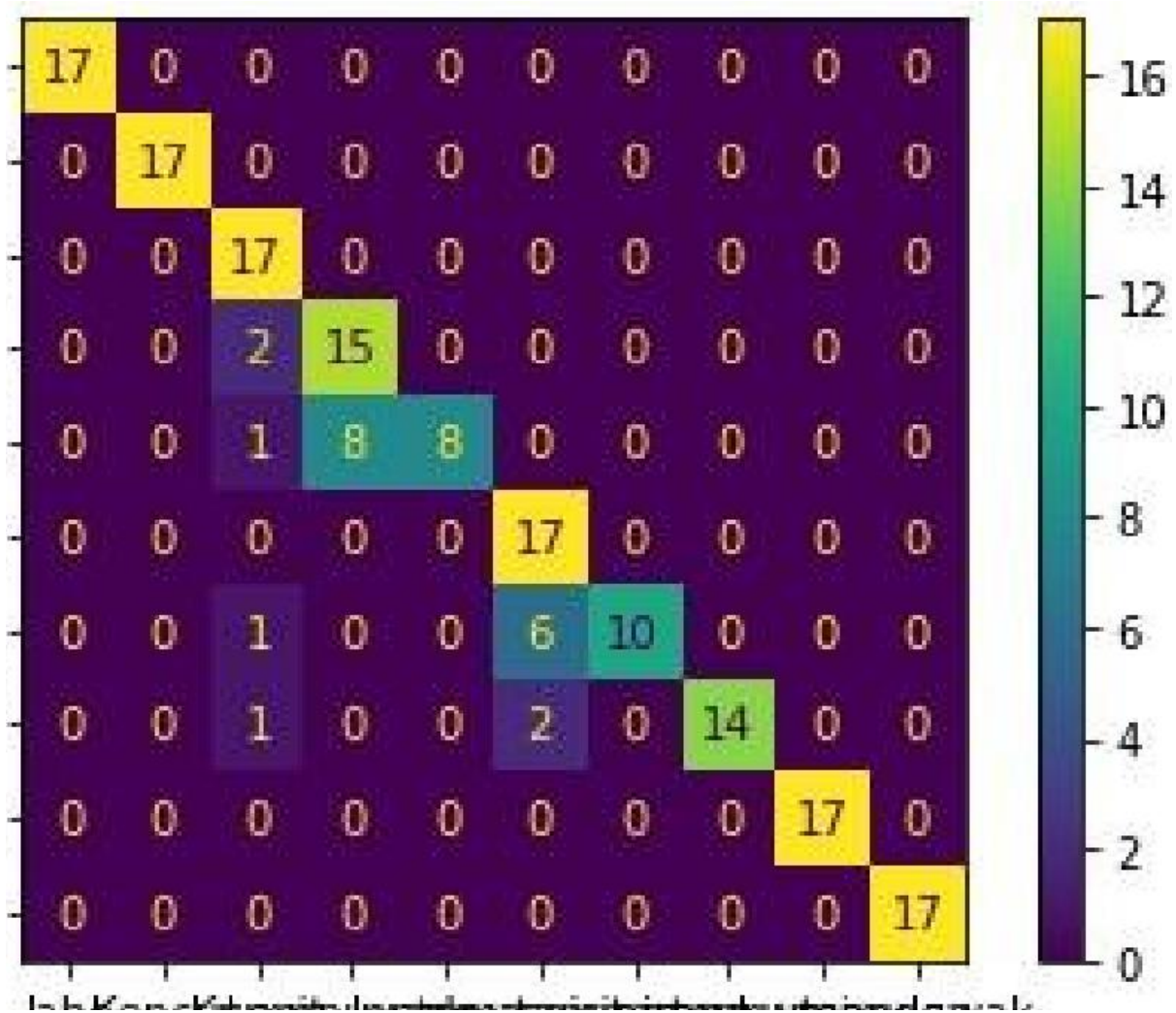
Visualisasi confusion matrix

- Representasi dengan heat map lebih baik.
- Contoh di samping adalah confusion matrix pada suatu percobaan di epoch 1.
- Berapa jumlah kelasnya?
- Jumlah data uji tiap kelas?



Visualisasi confusion matrix

- Contoh disamping hasil pada suatu percobaan pada epoch 4.
- Jumlah kelas = jumlah kolom = jumlah baris. Untuk case di samping, jumlah kelas adalah 10
- Jumlah data set **tiap kelas** sama untuk semua kelas = jumlah elemen tiap baris = 17
- Tanpa melihat nilainya, perbedaan heat map dapat diindera lebih cepat untuk membedakan hasil epoch 1 vs. epoch 4



Binary Classification

- *Hanya ada kelas: 0 atau 1, valid atau invalid, true atau false, positif atau negatif, bagus atau tidak bagus, cantik atau tidak cantik, rekomended atau tidak, lulus atau tidak, spam atau bukan spam, hoax atau bukan hoax, dsb.*

- *Bentuk yang paling umum: satu kelas dinyatakan sebagai kelas **positif** (menjadi fokus dalam klasifikasi), dan satu kelas lainnya dinyatakan sebagai kelas **negatif***

Contoh dalam dunia medis : sampel cairan mukus yang mengandung virus Covid-19 dinyatakan sebagai kelas positif dan sampel yang tidak mengandung virus dinyatakan sebagai kelas negatif

Contoh dalam kebencanaan : gempa yang mengakibatkan tsunami sebagai kelas positif dan yang tidak mengakibatkan tsunami sebagai kelas negatif

Keterbatasan akurasi

Dalam kasus deteksi pasien postifi Covid, sebuah detektor baru, sebut saja detektor X, diujikan pada 100 sampel. Sampel tersebut telah diuji dengan alat yang hasil deteksinya dijadikan acuan validitas (ground truth), yaitu PCR. Dari pengujian PCR sebelumnya diperoleh data bahwa 90 sampel adalah negatif dan 10 sampel adalah positif. Dengan menggunakan detektor X, ke 90 sampel negatif dideteksi negatif. Namun pada 10 sampel positif, diperoleh hasil bawah 5 sample dinyatakan sebagai positif dan 5 sampel sisanya negatif.

Berapa akurasi detektor X? (90+5)/(90+10) x 100% = 95% !

Apakah akurasi 95% merupkan hasil yang baik?

Kesalahan detektor X yang hanya pada 5% dapat berakibat fatal. Apakah ada metrik lain yang menjelaskan kasus semacam ini?

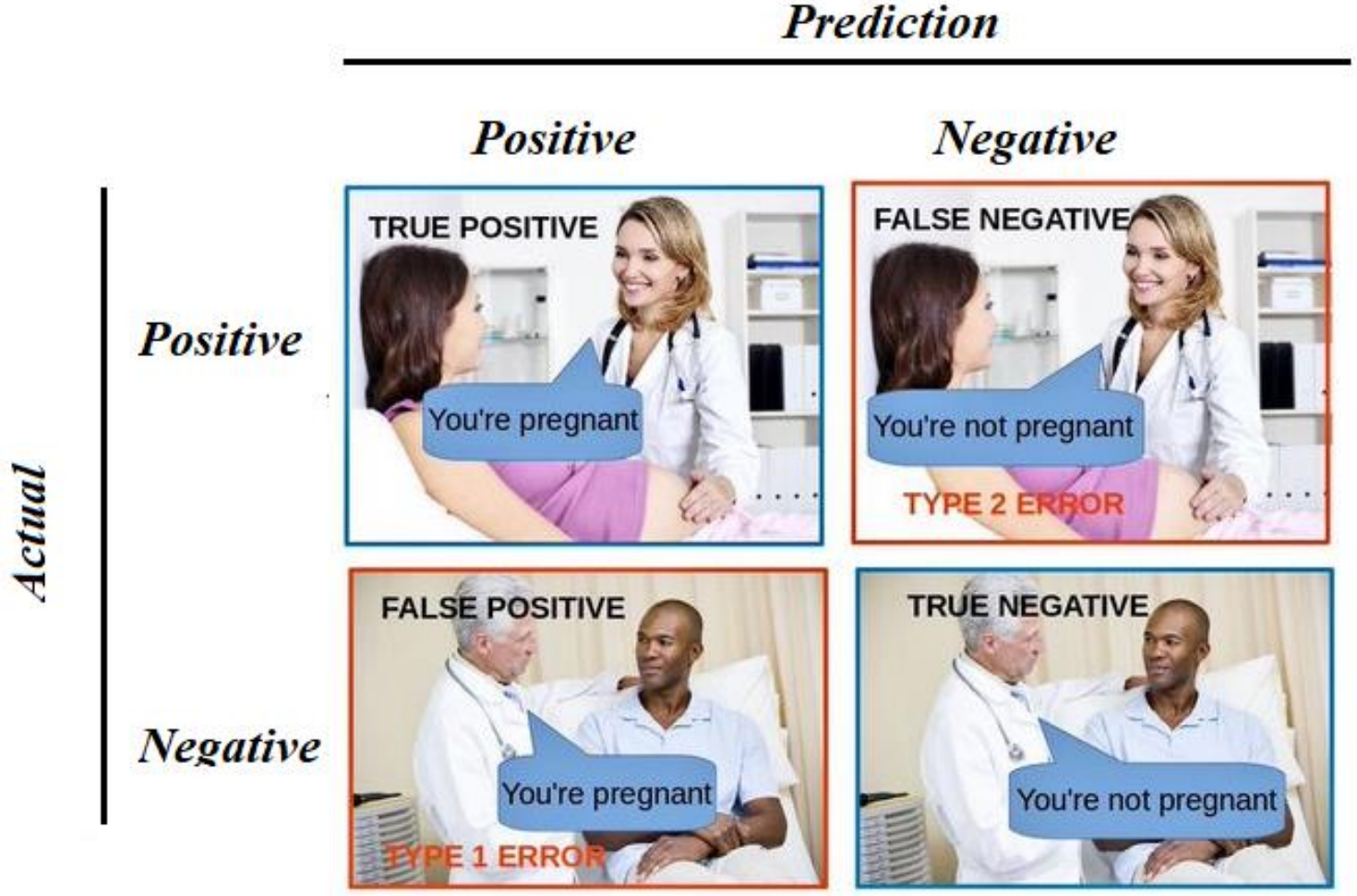
Confusion Matrix untuk Klasifikasi Bir

		Nilai Prediksi	
		Positive	Negative
Nilai Aktual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

True Positive (TP): nilai sesungguhnya adalah positif dan diprediksi positif
False Positive (FP): nilai sesungguhnya adalah negatif namun diprediksi positif
True Negative (TN): nilai sesungguhnya adalah negatif dan diprediksi negatif, dan
False Negative (FN): nilai sesungguhnya adalah positif namun diprediksi negatif.

Klasifikasi yang bernilai **valid** adalah **TP dan TN**

Confusion Matrix untuk Klasifikasi Biner (Ilustrasi

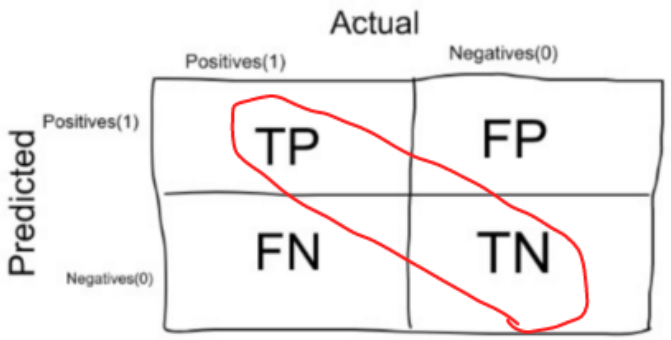


Source: Modifikasi dari https://skappal7.files.wordpress.com/2018/08/confusion-matrix.jpg

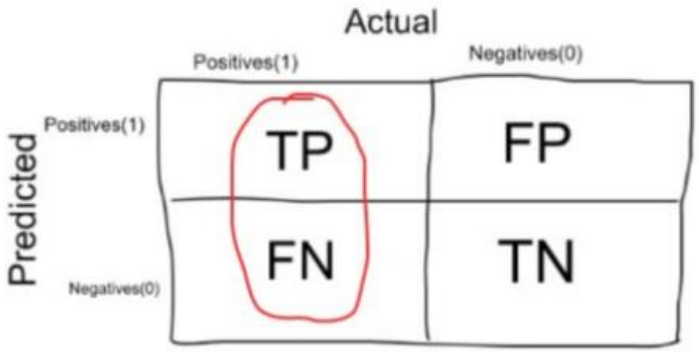
Metrik pada Klasifikasi Biner

		Nilai Prediksi: Positive	Nilai Prediksi: Negative	
Nilai Aktual	Positive	True Positive (TP)	False Negative (FN)	Recall, Sensitivity, True Positive Rate $\frac{TP}{TP + FN}$
	Negative	False Positive(FP)	True Negative (TN)	Specificity, True Negative Rate $\frac{TN}{FP + TN}$ False Positive Rate $\frac{FP}{FP + TN}$
		Precision $\frac{TP}{TP + FP}$.	Negative Predictive Value $\frac{TN}{TN + FN}$.	Accuracy $\frac{TP + TN}{TP + TN + FP + FN}$

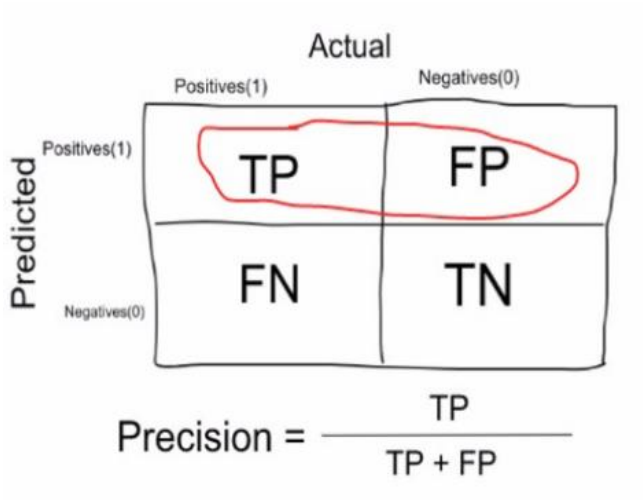
Matriks Performansi Klasifikasi



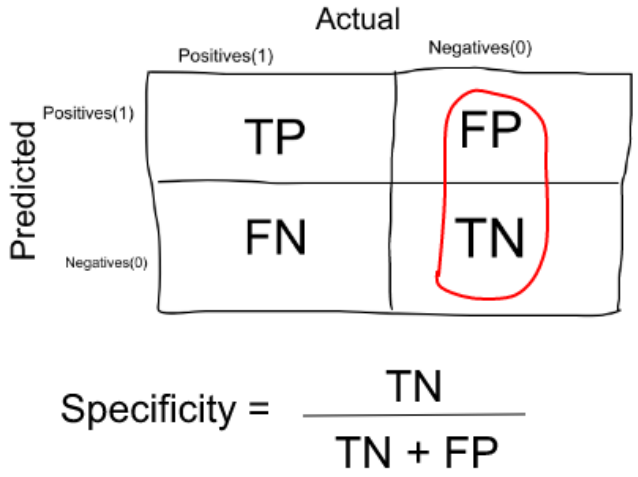
$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$



$$\text{Recall} = \frac{TP}{TP + FN}$$



$$\text{Precision} = \frac{TP}{TP + FP}$$



$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

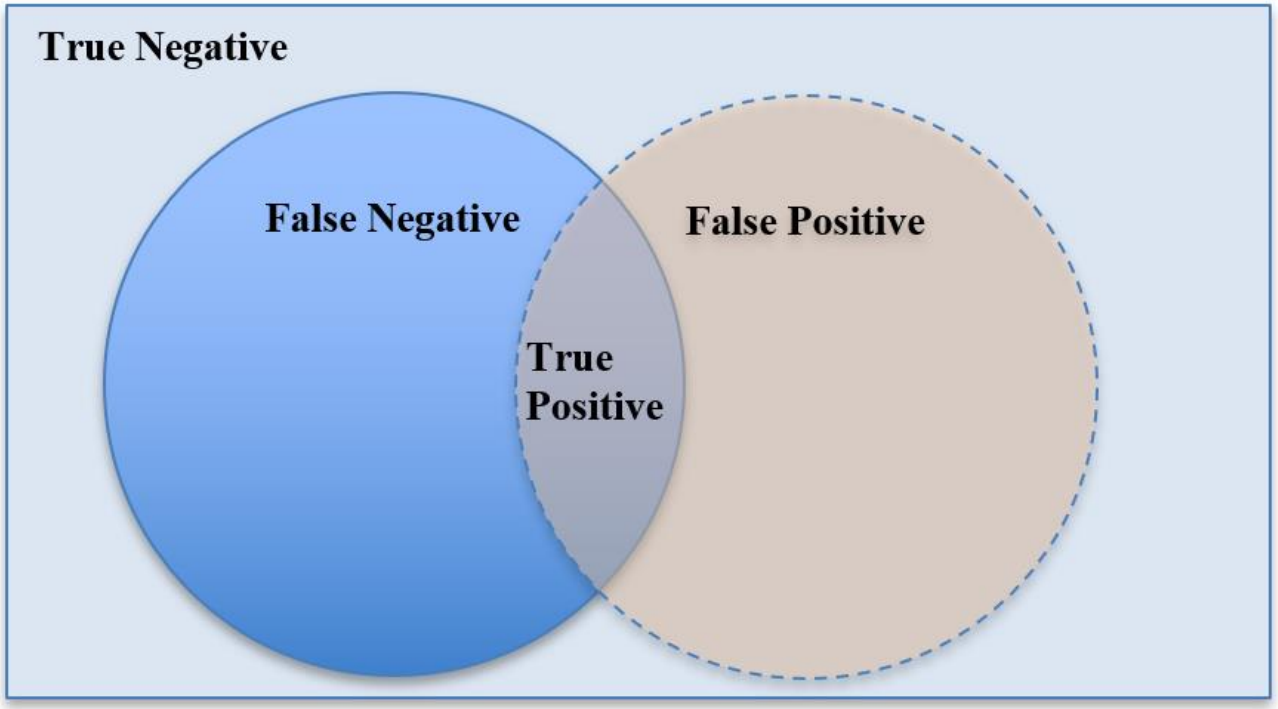
Interpretasi metrik Recall - Precision

Recall dan Precision sering dihitung bersamaan untuk menggambarkan performansi model. Kombinasi yang mungkin untuk keduanya:

- *Low recal low prescision*
- *High recall low precison*
- *Low recall high precision*
- *High recall high precision*

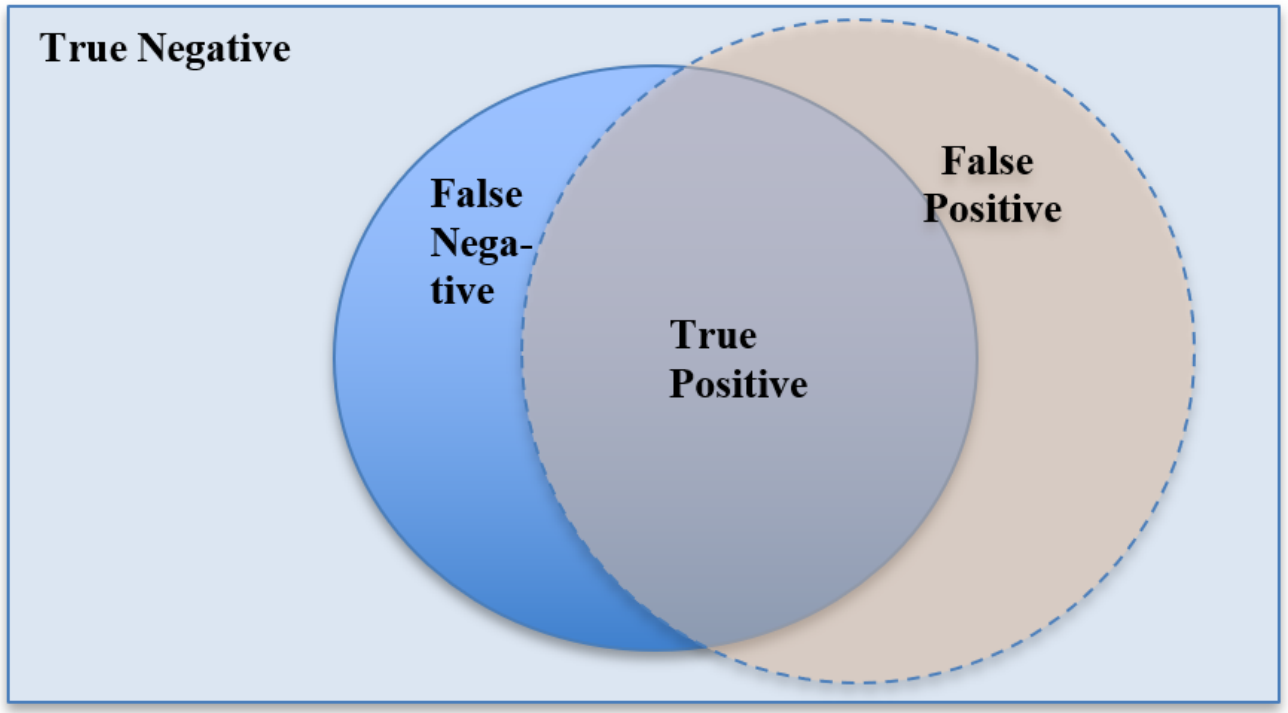
Low recall low precision

- *Model berkinerja kurang baik*
- *Baik False negatif dan maupun false positif bernilai besar pada model ini*

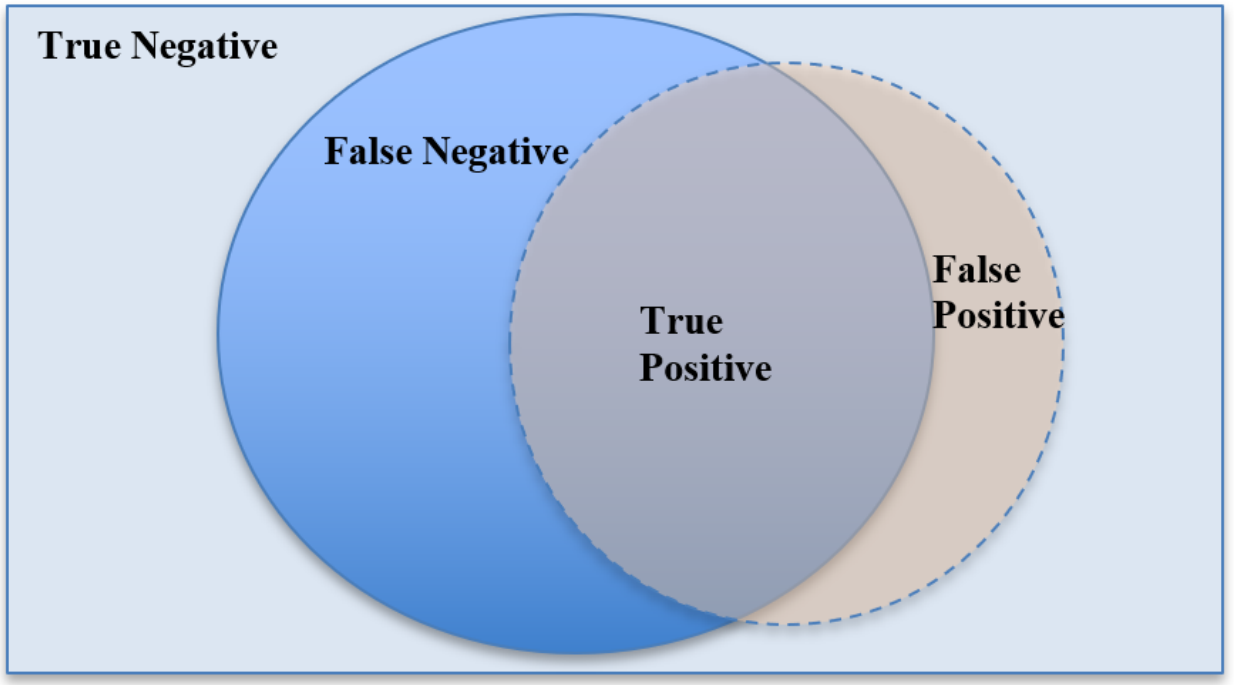


High recall low precision

- *Sebagian besar data positif dapat dikenali dengan baik (False Negative rendah)*
- *Tetapi banyak data negatif dikenali sebagai positif (False Positive tinggi*

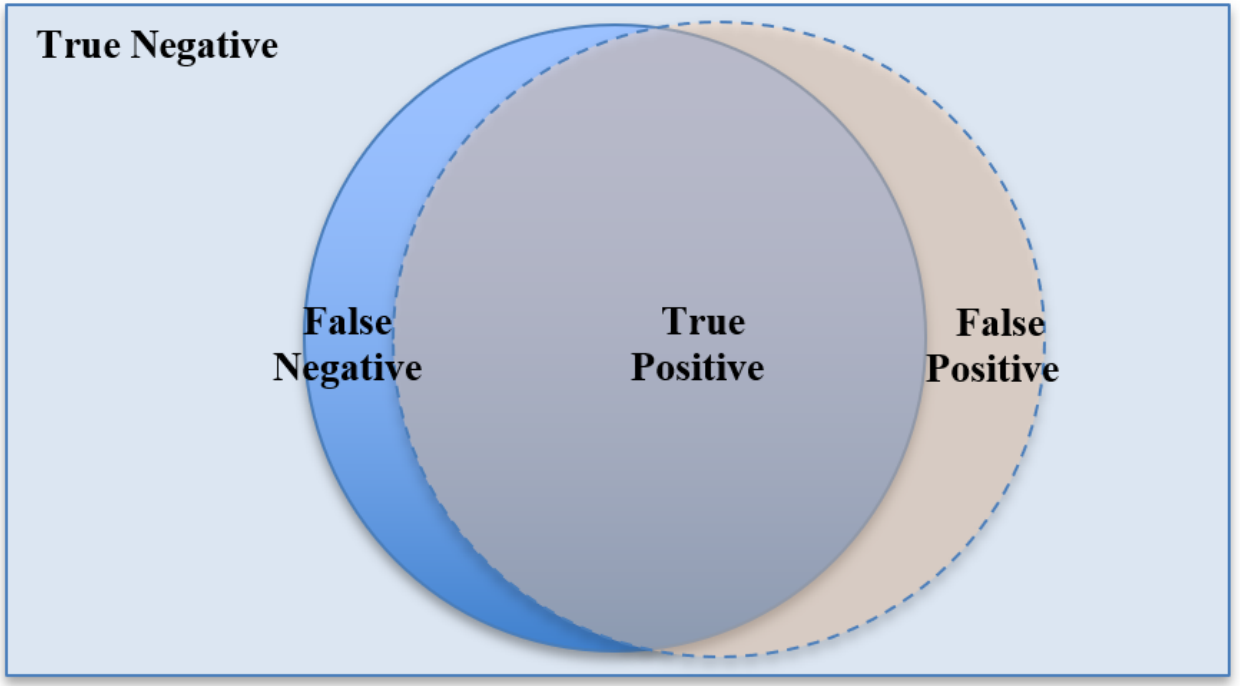


Low recall high precision



- *Banyak data positif yang teridentifikasi negatif (False Negative besar)*
- *Sebagian besar data yang teridentifikasi positif memang benar positif*

High recall high precision



Model memiliki kinerja baik

False Positive maupun False Negaative rendah

True Positve dan True Negative tinggi

F Score

- *Mengukur keseimbangan antara Precison – Recall*
- *Untuk model yang Precision dan Recall sama pentingnya digunakan F-1 Score, dinyatakan:*

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}.$$

- *Untuk kasus recall lebih diutamakan dengan faktor β , maka formula diperluas menjadi:*

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}.$$

Contoh implementasi ANN menggunakan library Scikit-learn

- Load data

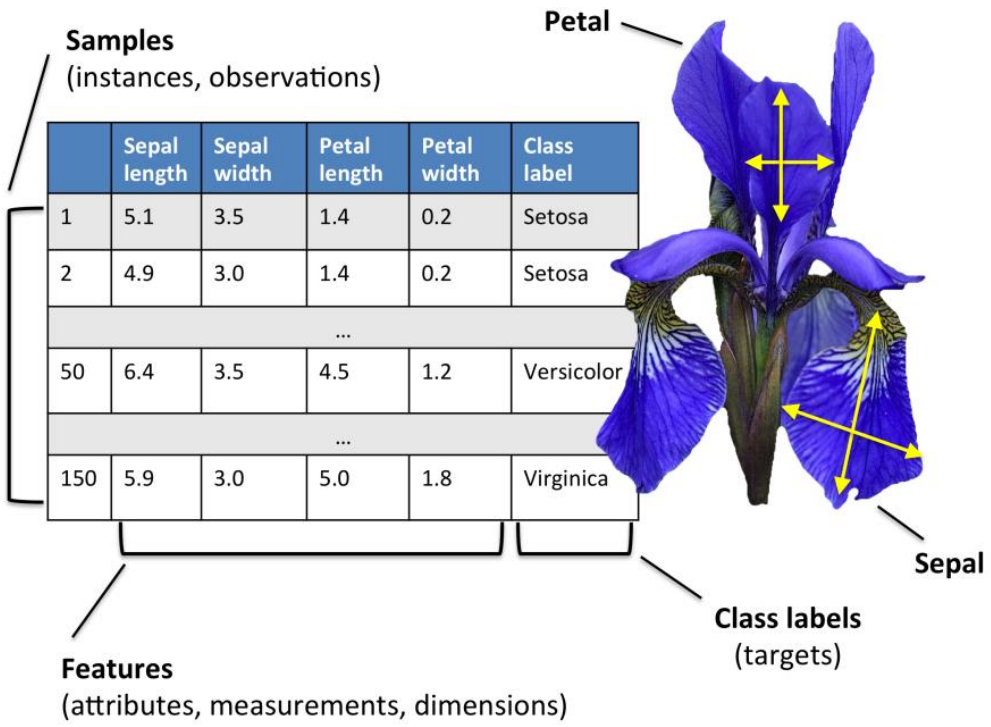
sklearn.datasets.load_iris

```
sklearn.datasets.load_iris(*, return_X_y=False, as_frame=False)
```

Load and return the iris dataset (classification).

The iris dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

Contoh implementasi ANN menggunakan library Scikit-learn

- Split data

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

sklearn.model_selection.train_test_split

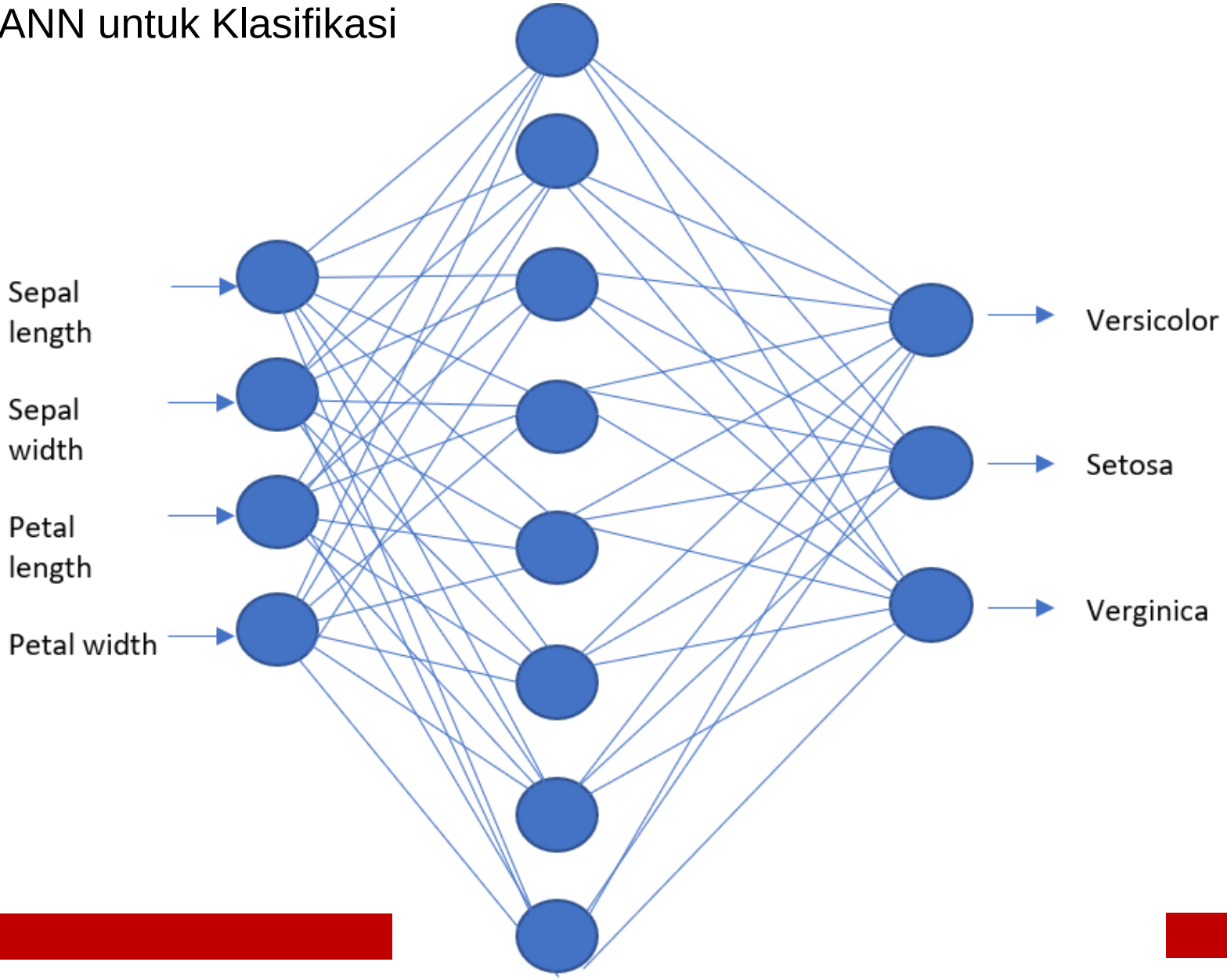
sklearn.model_selection.**train_test_split**(**arrays*, *test_size=None*, *train_size=None*, *random_state=None*, *shuffle=True*, *stratify=None*) [source]

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=.10)
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=.2)
print('X_train', X_train.shape)
print('X_val', X_val.shape)
print('X_test', X_test.shape)
```

```
X_train (108, 4)
X_val (27, 4)
X_test (15, 4)
```

Contoh implementasi ANN menggunakan library Scikit-learn

Arsitektur ANN untuk Klasifikasi Iris



Iris Versicolor


Iris Setosa


Iris Virginica

Contoh implementasi ANN menggunakan library Scikit-learn

- Define and compile model

```
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(hidden_layer_sizes=(100, ), activation='logistic', max_iter= 800)
```

- Fit model and evaluation model

```
from sklearn.metrics import accuracy_score

mlp.fit(X_train, Y_train)
prediksi_val = mlp.predict(X_val)
acc_val = accuracy_score(Y_val, prediksi_val)
print('Akurasi Validasi Training ANN:', acc_val)
```

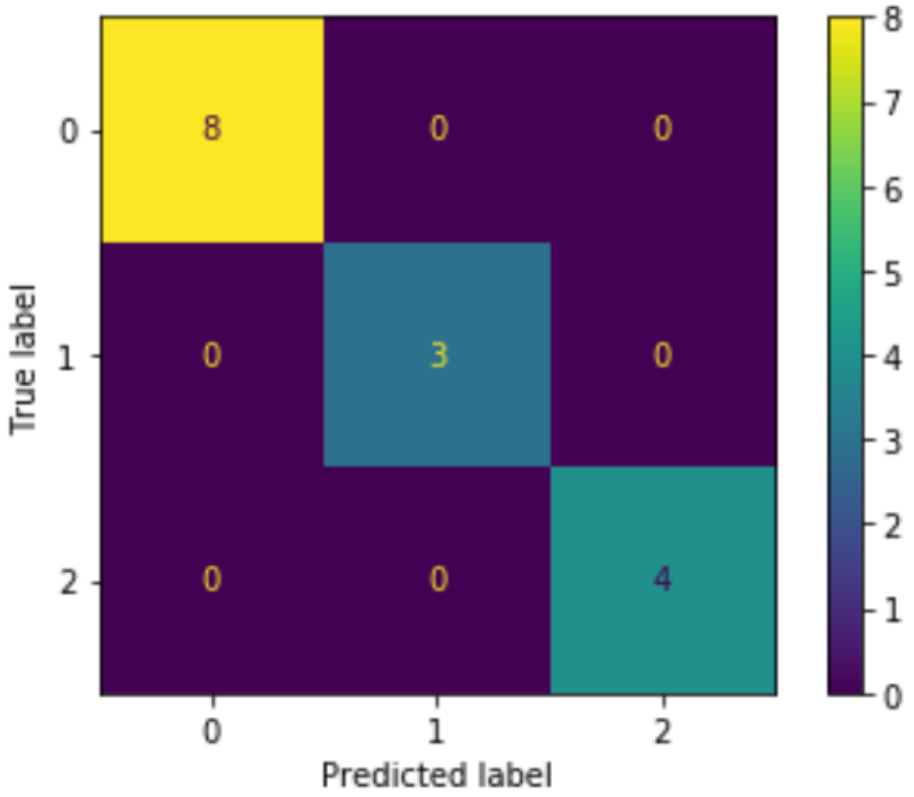
```
Akurasi Validasi Training ANN: 1.0
```

Contoh implementasi ANN menggunakan library Scikit-learn

- Prediction

```
from sklearn.metrics import accuracy_score, plot_confusion_matrix

prediksi_test = mlp.predict(X_test)
acc_test = accuracy_score(Y_test, prediksi_test)
print('Akurasi Testing ANN:', acc_test)
plot_confusion_matrix(mlp, X_test, Y_test)
```

```
Akurasi Testing ANN: 1.0
```



Tools / Lab Online

- TensorFlow is an end-to-end open-source platform for machine learning
- Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible

https://www.tensorflow.org/overview/

https://keras.io/getting_started/

https://keras.io/examples/

Tools / Lab Online

- Keras Model Sequential dan Layers

https://keras.io/api/models/

https://keras.io/guides/sequential_model/

https://keras.io/api/layers/

```
model = keras.Sequential(
    [
        layers.Dense(64, activation="relu", name="layer1"),
        layers.Dense(32, activation="relu", name="layer2"),
        layers.Dense(4, name="layer3"),
    ]
)
```

atau

```
model = keras.Sequential()
model.add(layers.Dense(64, activation="relu"))
model.add(layers.Dense(32, activation="relu"))
model.add(layers.Dense(4))
```

Contoh implementasi ANN menggunakan library keras

- Load data

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=.10)
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=.2)
print('X_train', X_train.shape)
print('X_val', X_val.shape)
print('X_test', X_test.shape)
Y_train = to_categorical(Y_train,3)
Y_val = to_categorical(Y_val,3)
Y_test = to_categorical(Y_test,3)
```

Contoh implementasi ANN menggunakan library keras

- Define model dan compile model

```
from keras.models import Sequential
from keras.layers import Flatten, Dense

model = Sequential()
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(3,activation='softmax’))

model.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['acc'])
```

Contoh implementasi ANN menggunakan library keras

- Fit model

```
model.fit(X_train,Y_train,epochs=64,batch_size=5,validation_data=(X_test,Y_test))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 4)                 0
_________________________________________________________________
dense (Dense)                (None, 64)                320
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 515
Trainable params: 515
Non-trainable params: 0
_________________________________________________________________
```

Contoh implementasi ANN menggunakan library keras

- Evaluation model

```
loss, accuracy = model.evaluate(X_test, Y_test)
print('Akurasi Testing MLP:', accuracy)
```

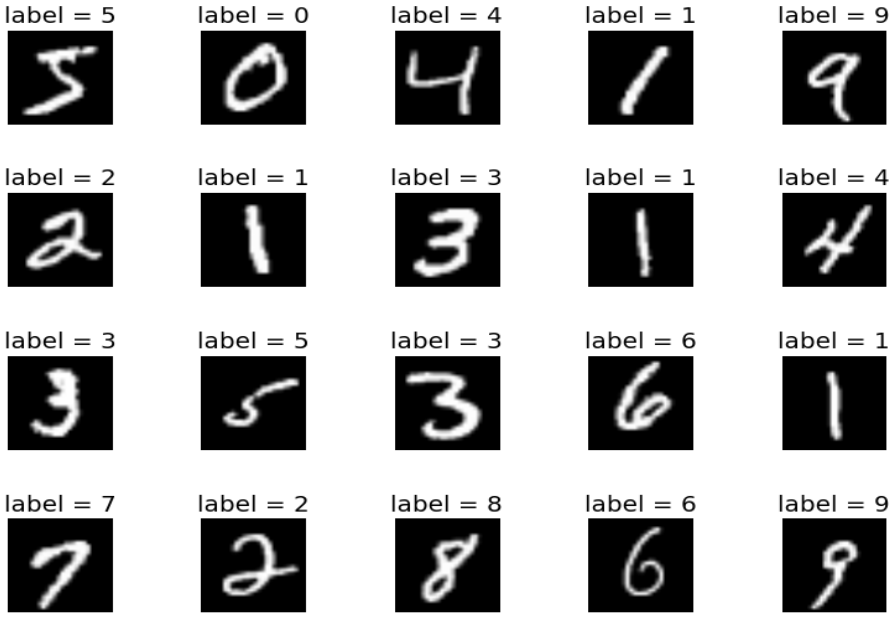
```
1/1 [==============================] - 0s 0s/step - loss: 0.0393 - acc: 1.0000
Akurasi Testing ANN: 1.0
```

Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

Dataset MNIST *Handwritten Digit* dibagi menjadi 3:

- 55,000 training data
- 10,000 test data
- 5,000 validation data

Setiap citra berukuran 28 × 28 pixels dan label kelas dirubah menjadi *one hot encoded*

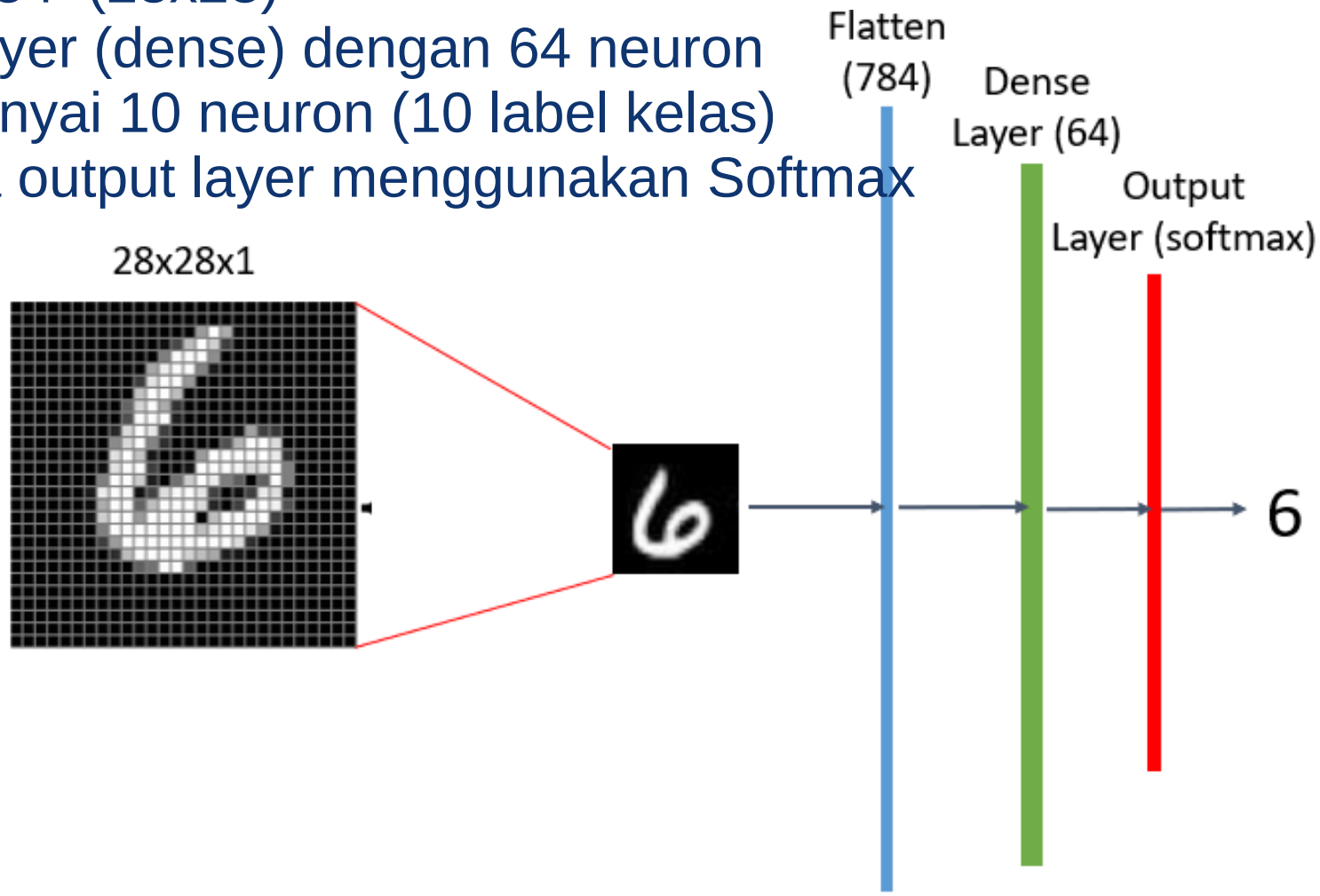


0	[1 0 0 0 0 0 0 0 0 0]
1	[0 1 0 0 0 0 0 0 0 0]
2	[0 0 1 0 0 0 0 0 0 0]
3	[0 0 0 1 0 0 0 0 0 0]
4	[0 0 0 0 1 0 0 0 0 0]
5	[0 0 0 0 0 1 0 0 0 0]
6	[0 0 0 0 0 0 1 0 0 0]
7	[0 0 0 0 0 0 0 1 0 0]

Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

Arsitektur ANN yang digunakan untuk pengenalan angka:

- Ukuran input layer 784 (28x28)
- Terdapat 1 hidden layer (dense) dengan 64 neuron
- Output layer mempunyai 10 neuron (10 label kelas)
- Fungsi aktivasi pada output layer menggunakan Softmax



Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

1. Load data

```
import keras
from keras.datasets import mnist


(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(-1, 28,28,1)
X_test = X_test.reshape(-1, 28,28,1)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

2. Define Model

```
from keras.models import Sequential
from keras.layers import Flatten, Dense


model1 = Sequential()
model1.add(Flatten())
model1.add(Dense(64,activation='relu'))
model1.add(Dense(10,activation='softmax’))
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 64)                50240
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 50,890
Trainable params: 50,890
Non-trainable params: 0
_________________________________________________________________
```

Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

3. Compile Model, Fit Model, Save Model, dan Evaluasi Model

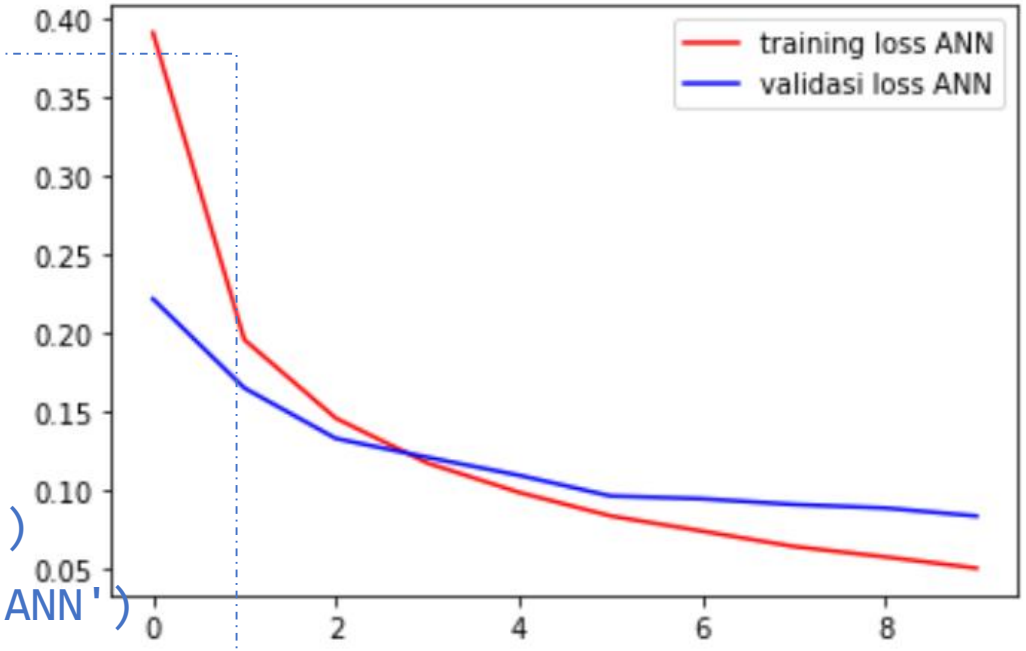
```
model1.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['acc'])
history =
model1.fit(X_train,y_train,epochs=10,batch_size=100,validation_data=(X_test,y_test))
model1.save('my_model1.h5')
model1.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.0839 - acc: 0.9761
[0.08388985693454742, 0.9761000275611877]
```

Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

3. Visualiasasi Evaluasi Model

```
import matplotlib.pyplot as plt


epochs = range(10)
loss1 = history1.history['loss']
val_loss1 = history1.history['val_loss']
plt.plot(epochs,loss1,'r',label='training loss ANN')
plt.plot(epochs,val_loss1,'b',label='validasi loss ANN')
plt.legend()
```



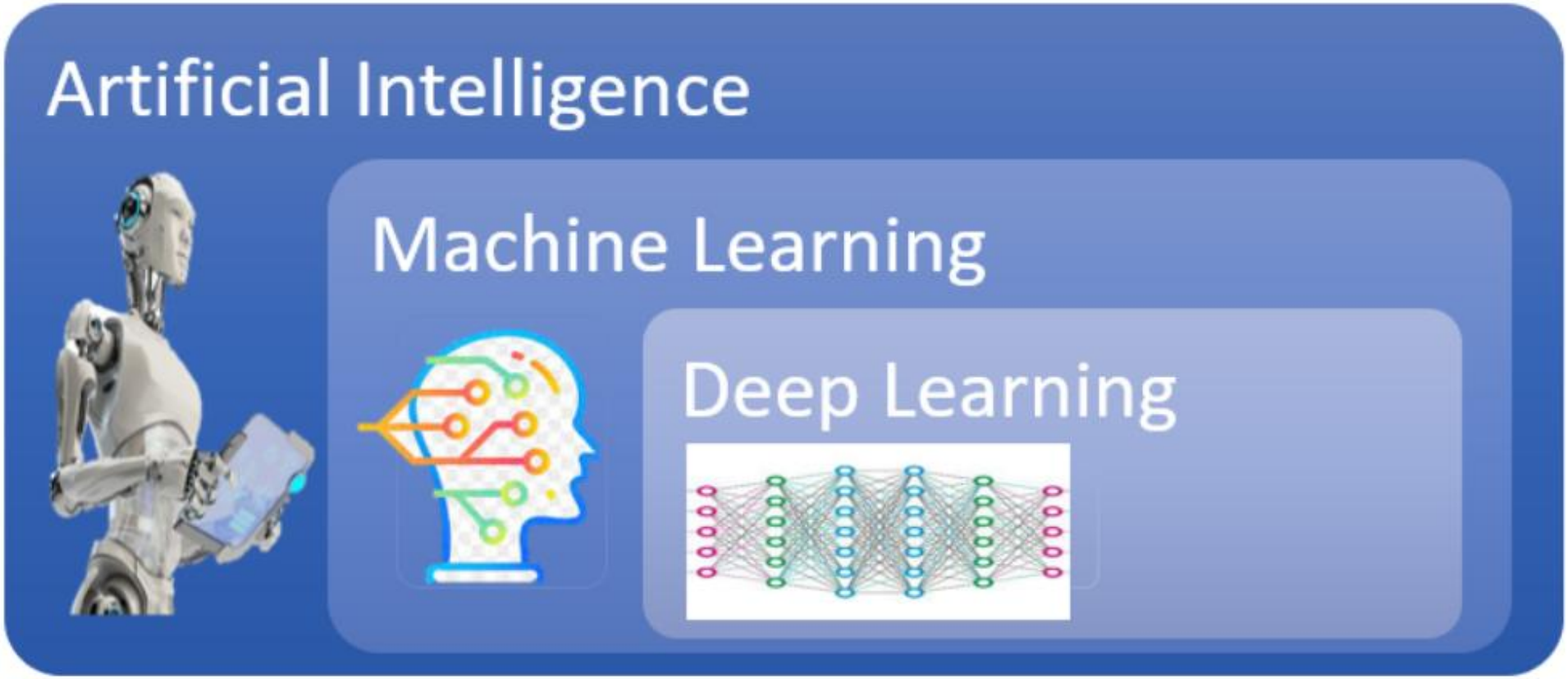
Contoh implementasi arsitektur *fully connected layer* pada pengenalan angka

3. Load Model dan Prediction

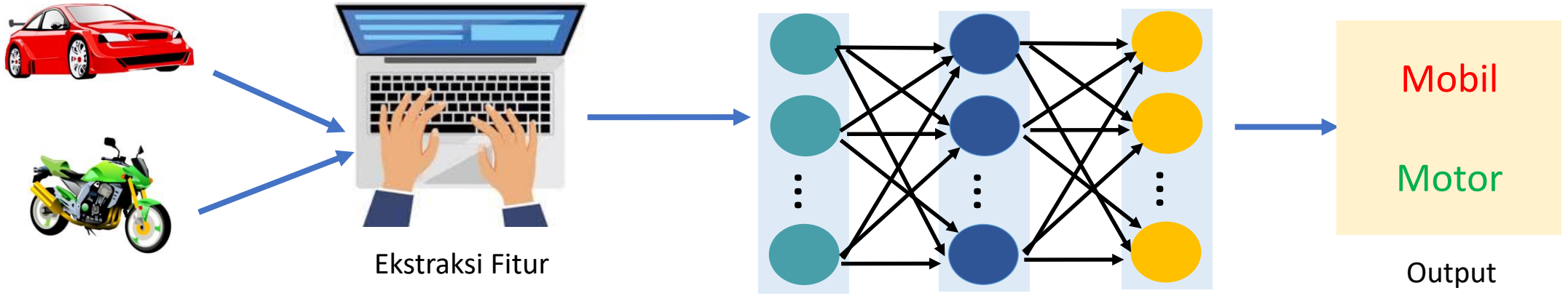
```
import numpy as np
from keras.models import load_model

model_simpan = load_model('my_model.h5')
pred = model_simpan.predict(X_test)
print('label actual:',np.argmax(y_test[30]))
print('label prediction:',np.argmax(pred[30]))
```

```
label actual: 3
label prediction: 3
```

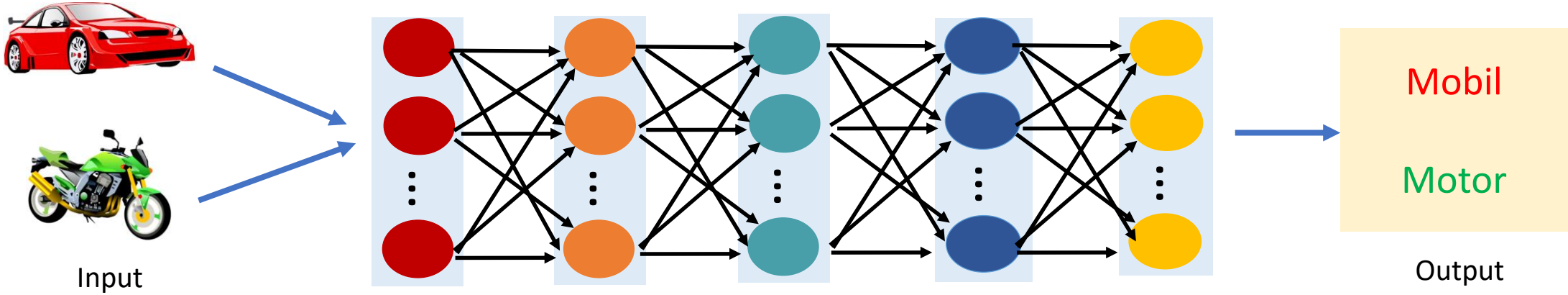
Deep Learning and Use Case



Machine Learning (Konvensional)



Deep Learning



Pengantar Deep Learning

Pendekatan klasifikasi secara konvensional umumnya melakukan ektraksi fitur secara terpisah kemudian dilanjutkan proses pembelajaran menggunakan metode klasifikasi konvensional

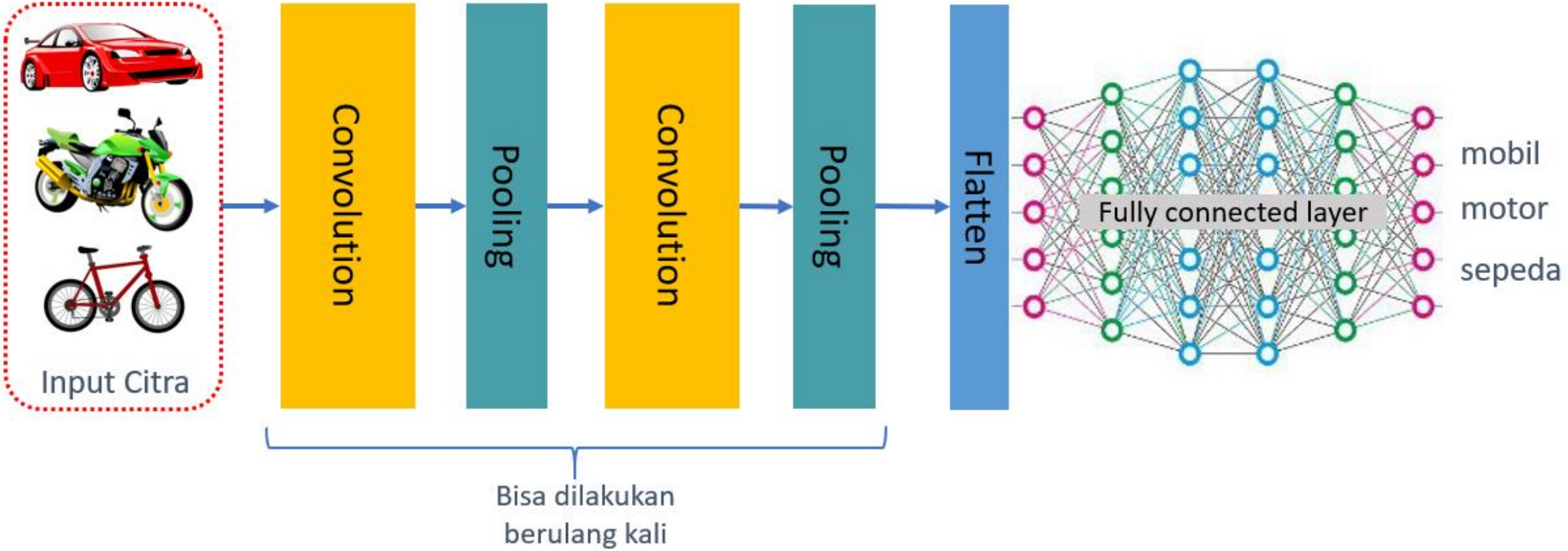
Kelemahan pendekatan konvensional:

- Memerlukan waktu dan pengetahuan lebih untuk ekstraksi fitur
- Sangat tergantung pada satu domain permasalahan saja sehingga tidak berlaku general

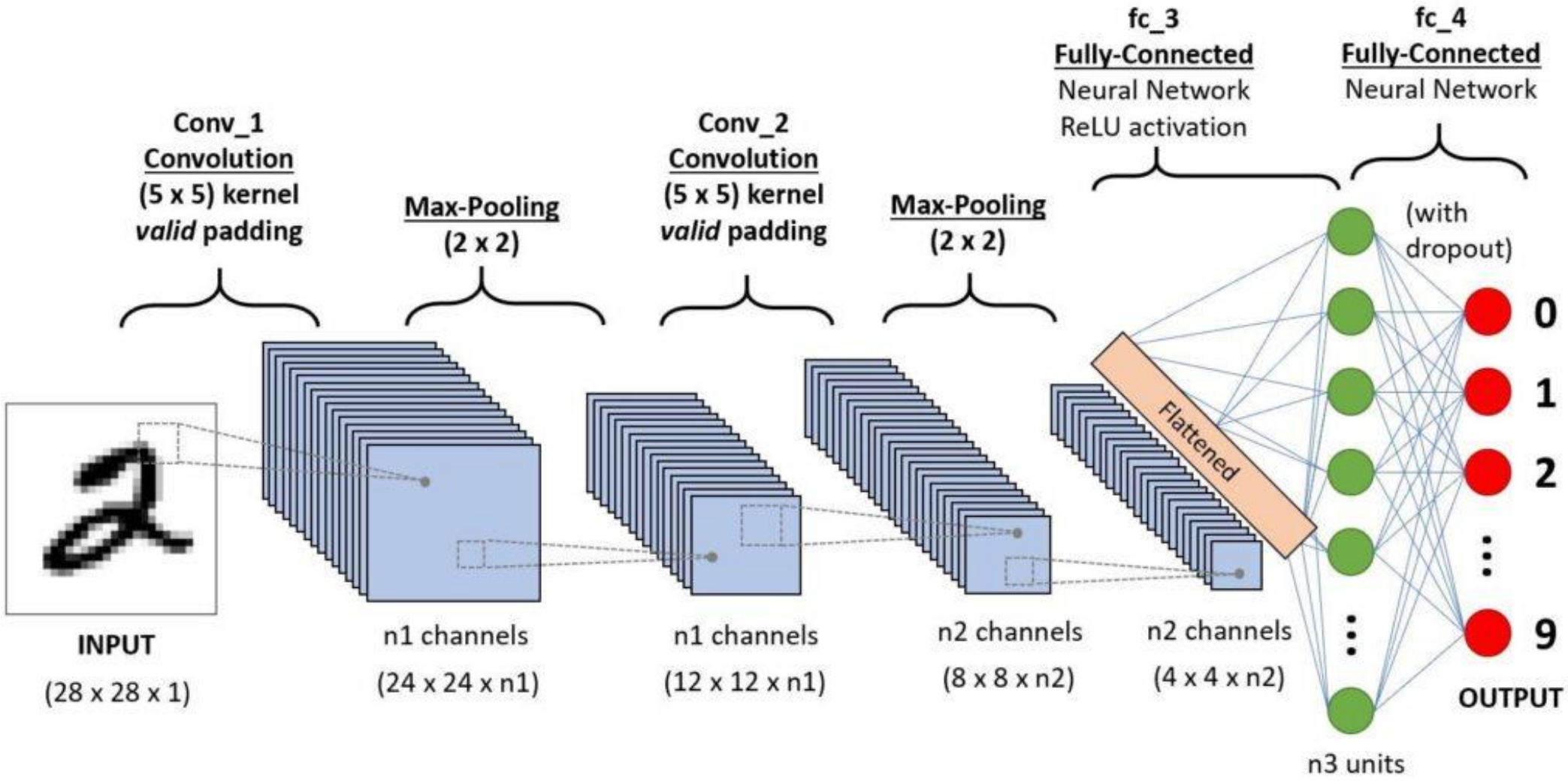
Pendekatan klasifikasi berbasis Deep learning mempelajari representasi hirarki (pola fitur) secara otomatis melalui beberapa tahapan proses *feature learning*

Convolutional Neural Network (CNN)

- CNN merupakan metode Deep Learning yang merupakan salah satu jenis arsitektur ANN
- Ada tiga layer utama yaitu *convolutional layer*, *pooling layer*, dan *fully connected*

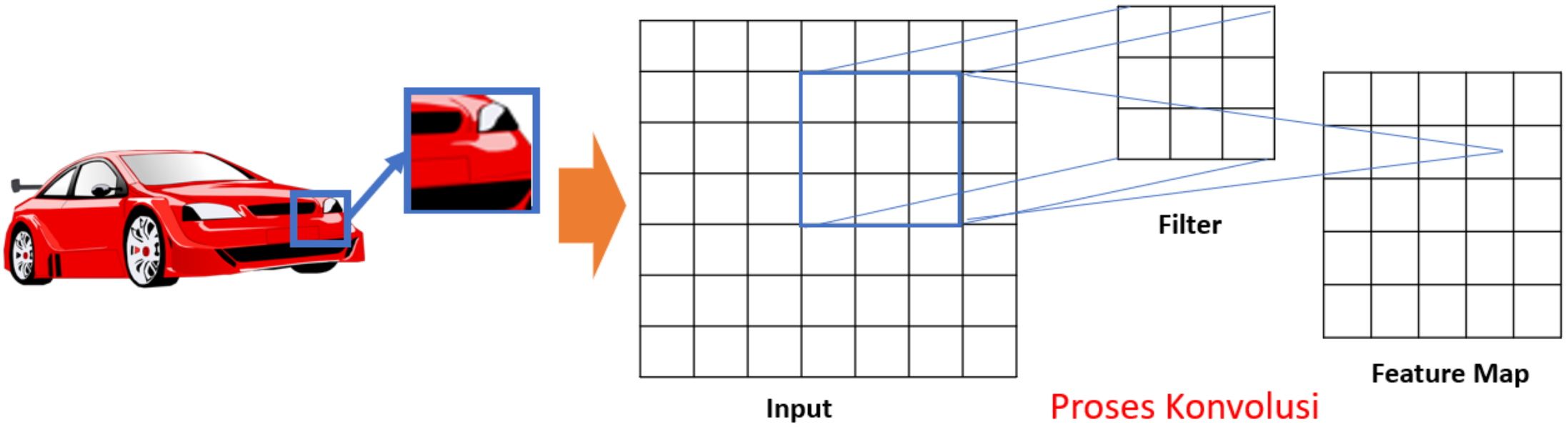


Deep Learning



Convolutional Layer

- *Convolutional layer* merupakan proses konvolusi citra input dengan filter yang menghasilkan *feature map*
- Ukuran matrik citra dan ukuran matrik filter akan mempengaruhi ukuran matrik *feature map*



Convolutional Layer

- Proses konvolusi citra dengan filter dilakukan sliding filter mulai dari kiri atas dari matirk citra sampai kanan bawah
- Rumus konvolusi dari citra *I* dengan filter *K* sebagai berikut:

$$(I * K)(i,j) == \sum_m \sum_n I(m,n)K(i+m, j+n)$$

Citra *I*

30	30	30	0	0	0
30	30	30	0	0	0
30	30	30	0	0	0
30	30	30	0	0	0
30	30	30	0	0	0
30	30	30	0	0	0

*

Filter *K*

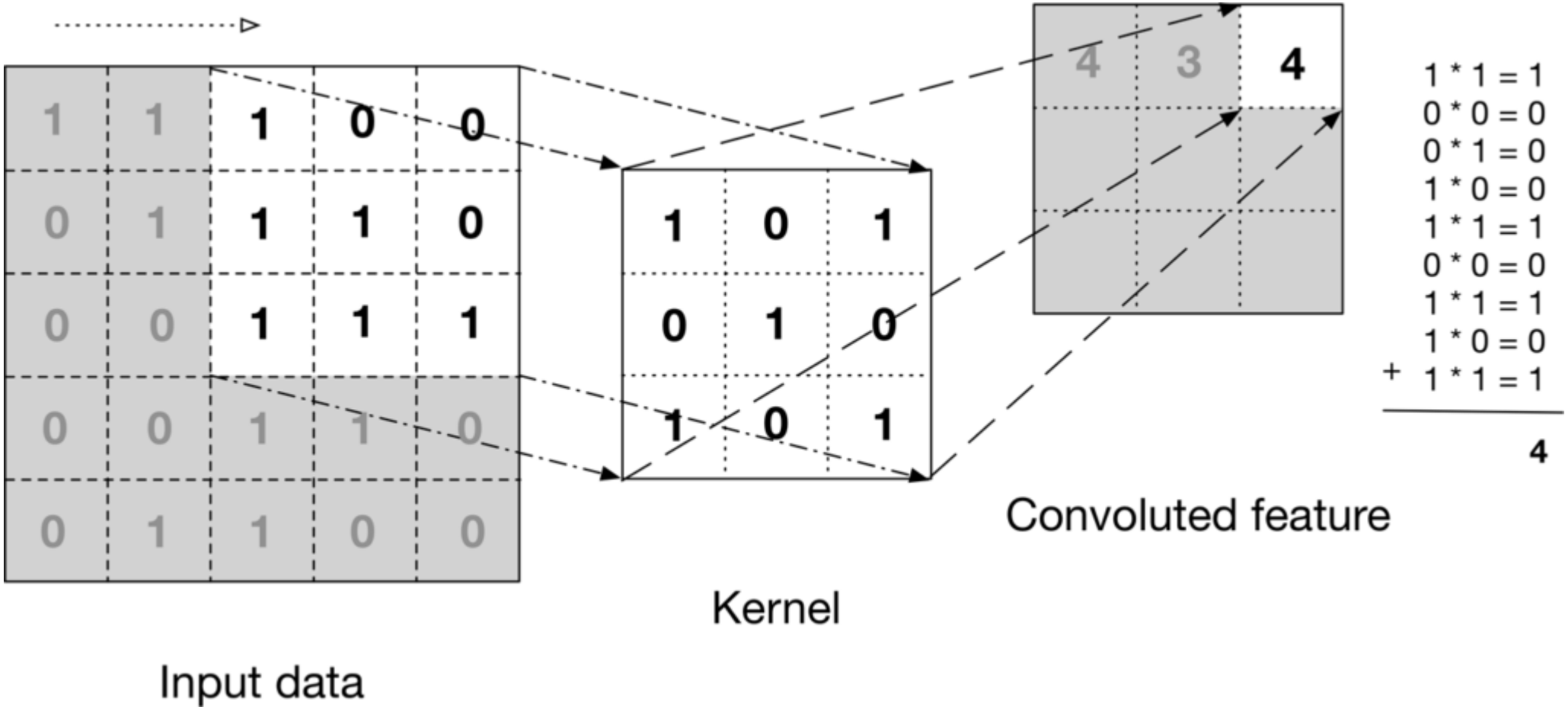
1	0	-1
1	0	-1
1	0	-1

=

Feature Map

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Image Convolution



Median Filter

Diketahui citra 3x4: x =

100	234	213	122
100	90	89	89
200	255	199	178

Specify the filter median of x in row 2 of column 2 with 3x3 masking!

Masking 3x3, didapatkan :

100	234	213
100	90	89
200	255	199

Sort Asc. : 89 90 100 100 **199** 200 213 234 255

Thus, the median filter result in row 2 of column 2 is 199

Characteristics:
- Median filter on the image will eliminate impulsive noise, for example, salt and papper noise, can also reduce Gaussian distributed noise

LPF Filter Masking for Image

Masking Filter 3x3

0.1111	0.1111	0.1111
0.1111	0.1111	0.1111
0.1111	0.1111	0.1111

Feature: Adding all elements produce 1

- LPF imagery will blur the image
- Will smooth the image (smoothing)
- Reduces the effects of Gaussian distributed noise

Masking Filter 5x5

0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400

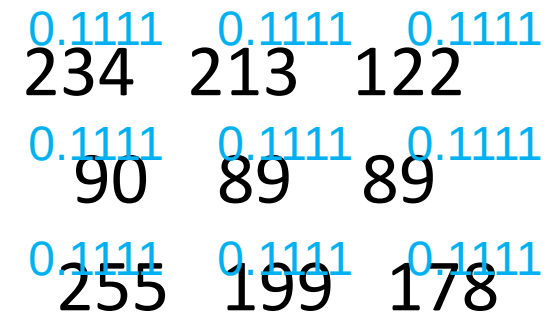
LPF Example

Image in 3x4:

x =

100	234	213	122
100	90	89	89
200	255	199	178

Determine LPF result 3x3 at row 2 column 3 !



Result :

0.1111*(234+213+122+90+89+89+255+199+178)=163.2

Rounded to 163

HPF Filter Masking for Image

Masking Filter 3x3

-1	0	-1
0	4	0
-1	0	-1

0	1	0
1	-4	1
0	1	0

Characteristic: The Masking Filter amounts to 0 if all the elements are added

HPF Example

Diketahui citra 3x4: x =

100	234	213	122
100	90	89	89
200	255	199	178

Determine HPF 3x3 at row 2 column 2, with following masking !

0	1	0
1	-4	1
0	1	0

0	1	0
100	234	213
1	-4	1
100	90	89
0	1	0
200	255	199

0.100+1.234+0.213+1.100+(-4)90+1.89+0.200+1.255+0.199= 318

Clipped to 255

What if an image filter is performed at the edge of the image?

Image 3x4:
x =

100	234	213	122
100	90	89	89
200	255	199	178

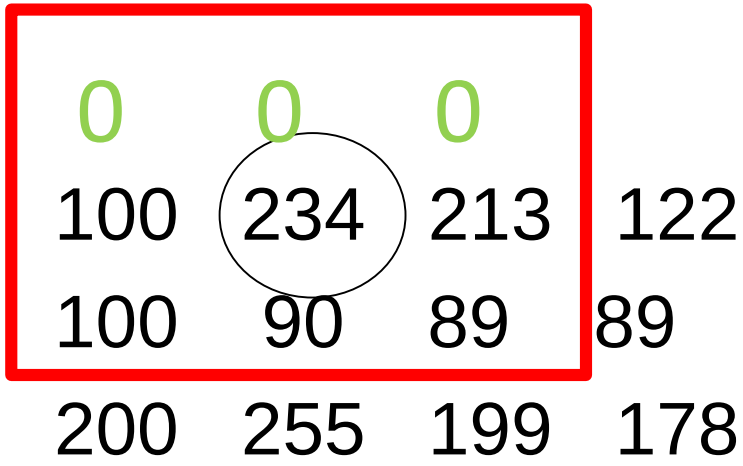
- Specify the filter median from x in row 1 to column 2 with 3x3 masking!
- Specify the filter median of x in row 3 of column 4 with 5x5 masking!

There are 2 methods:

1. Zero Padding
2. Border Replication

Applicable to all types of filtering, median, LPF, HPF

Zero padding

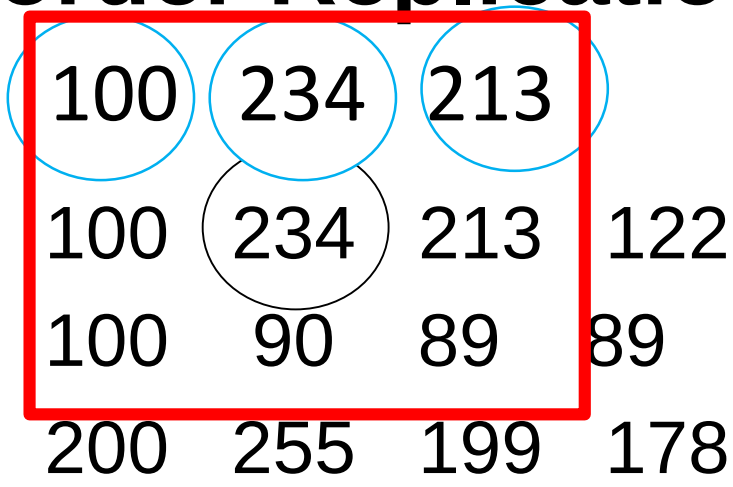


Pixel di luar citra diisi dengan nilai nol

[0 0 0 89 90 100 100 213 234]

90 adalah hasil median filter dg masking 3x3 di baris 2 kolom 1 dg metode zero padding

Border Replication

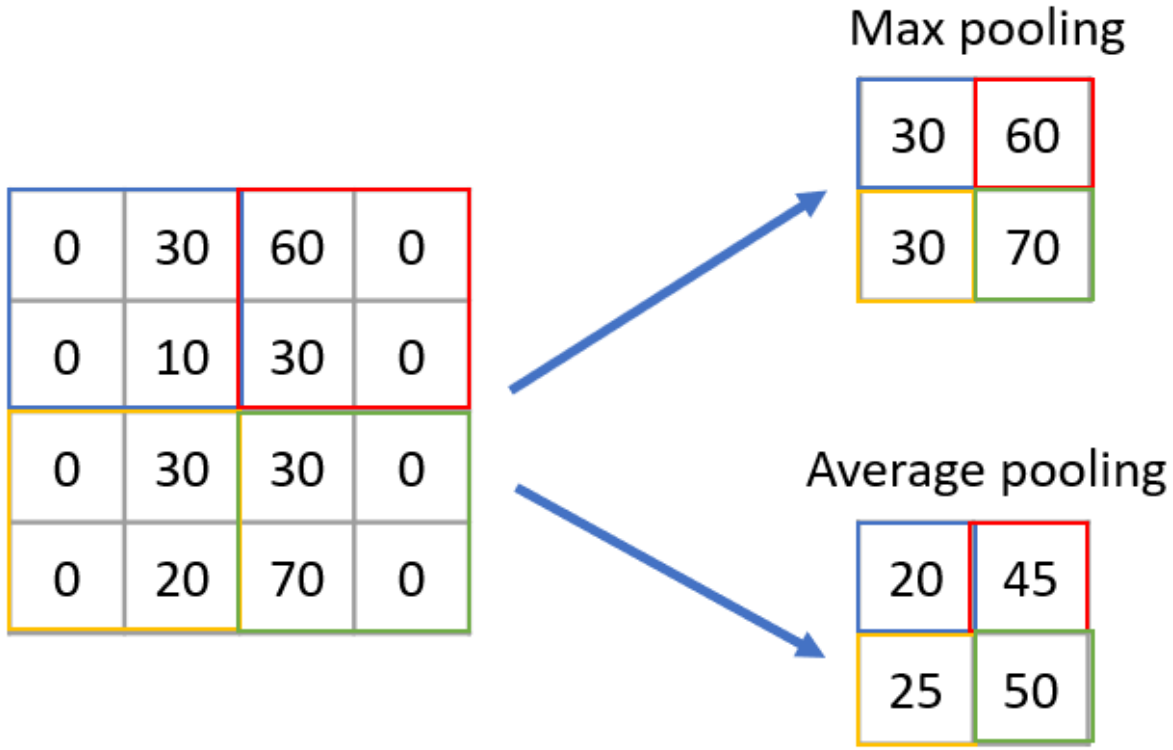


Pixels outside the image are populated with the nearest pixel value

1. [89 90 100 100 100 213 213 234 234]
2. 100 is the median result of filter with 3x3 masking in row 2 column 1 with border replication method

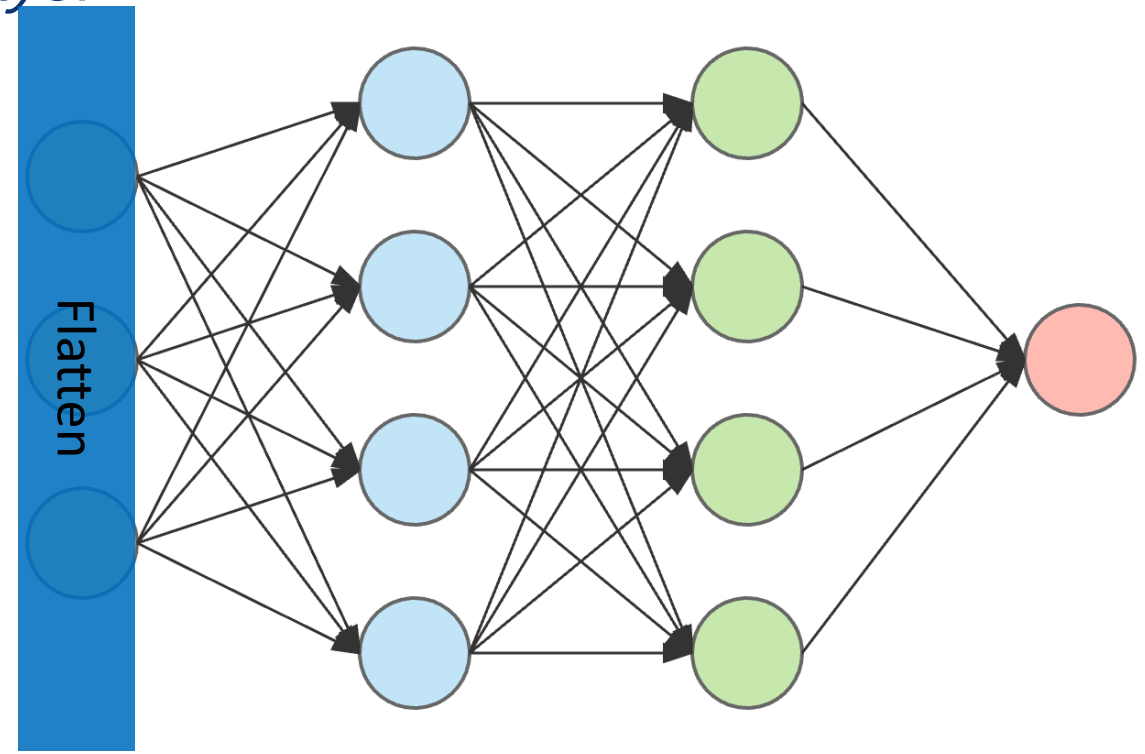
Pooling Layer

- *Pooling layer* digunakan untuk mengurangi ukuran gambar menjadi lebih kecil (*down sample*) dan mengekstrak *salient features*
- *Pooling layer* yang umum digunakan adalah *Maximum pooling* dan *Average pooling*



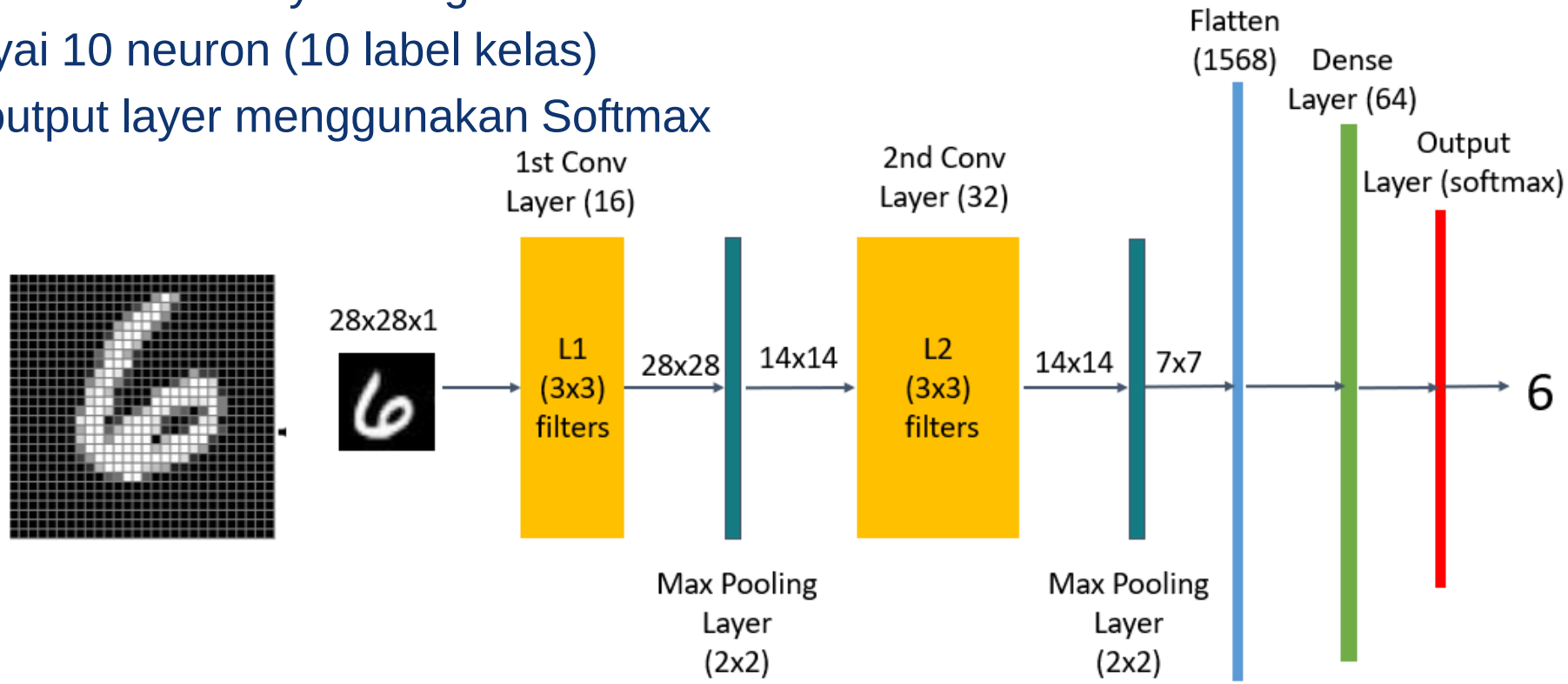
Fully Connected Layer

- *Fully connected layer* merupakan arsitektur *Multi-layer* ANN
- *Feature map* hasil dari proses konvolusi dan pooling, selanjutnya dilakukan proses *flatten* yaitu merubah matrix menjadi vektor sebagai inputan *fully connected layer*



Contoh Implementasi arsitektur CNN pada pengenalan angka

- Citra input 28x28
- Layer pertama konvolusi dengan 16 filter yang berukuran 3x3
- Layer kedua Max pooling dengan filter yang berukuran 2x2
- Layer ketiga konvolusi dengan 32 filter yang berukuran 3x3
- Layer kempat Max pooling dengan filter yang berukuran 2x2
- Layer Flatten dilanjutkan 1 hidden layer dengan 64 neuron
- Output layer mempunyai 10 neuron (10 label kelas)
- Fungsi aktivasi pada output layer menggunakan Softmax



Contoh implementasi arsitektur CNN pada pengenalan angka

1. Define Model CNN

```
from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model2 = Sequential()

model2.add(Conv2D(16,(3,3),activation='relu',input_shape=(28,28,1),padding='same'))

model2.add(MaxPooling2D(2,2))

model2.add(Conv2D(32,(3,3),activation='relu',padding='same'))

model2.add(MaxPooling2D(2,2))

model2.add(Flatten())

model2.add(Dense(64,activation='relu'))

model2.add(Dense(10,activation='softmax’))

model2.summary()
```

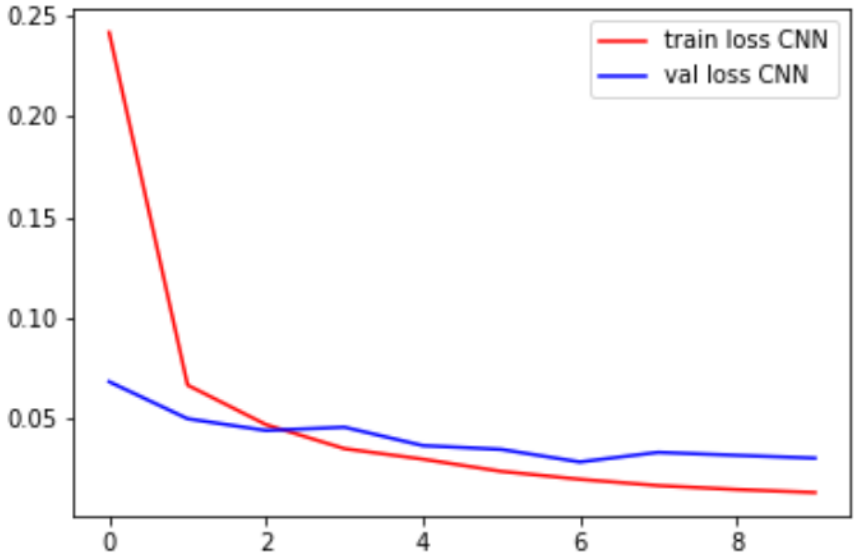
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                100416
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 105,866
Trainable params: 105,866
Non-trainable params: 0
```

Contoh implementasi arsitektur CNN pada pengenalan angka

2. Compile Model, Fit Model, Save Model, dan Evaluasi Model CNN

```
model. Compile(optimizer='adam',loss='categorical_crossentropy',metrics=['acc’])
history =
model.fit(X_train,y_train,epochs=10,batch_size=100,validation_data=(X_test,y_test))
model2.save('my_model2.h5')
model.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0304 - acc: 0.9897

[0.030350750312209l3, 0.989700019359588 6]
```



Contoh implementasi arsitektur CNN pada pengenalan angka

3. Load Model CNN dan Prediction

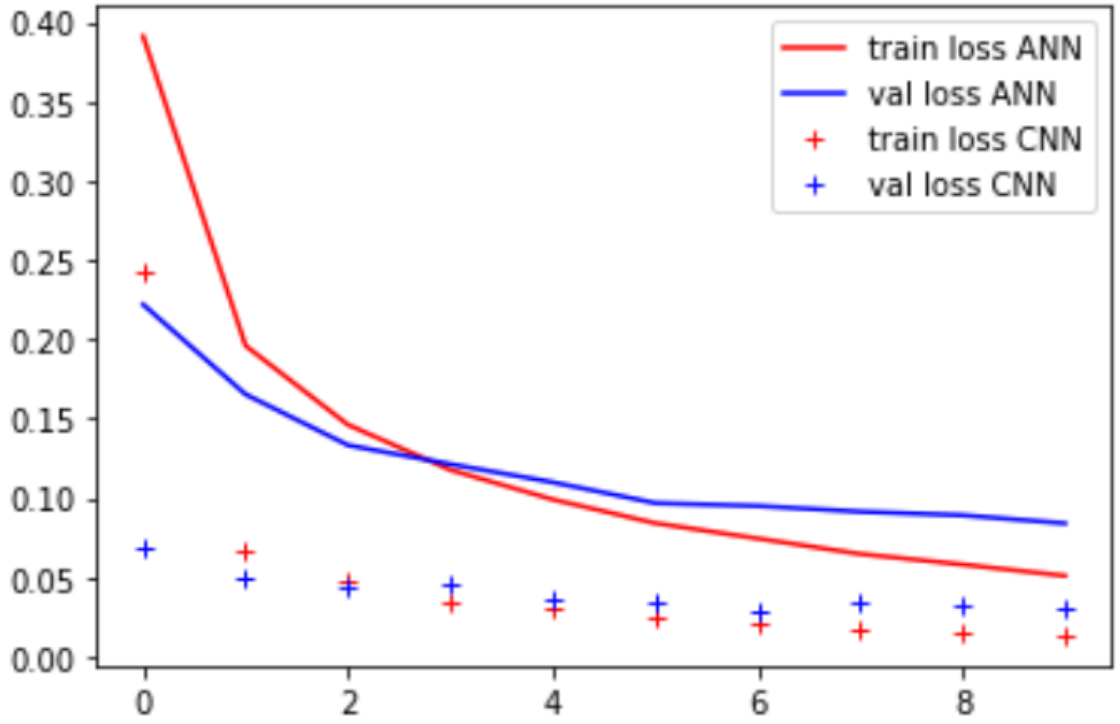
```
import numpy as np
from keras.models import load_model


model_simpan2 = load_model('my_model2.h5')
pred = model_simpan2.predict(X_test)
print('label actual:',np.argmax(y_test[30]))
print('label prediction:',np.argmax(pred[30]))
```

```
label actual: 3
label prediction: 3
```

Contoh Implementasi pada Pengenalan Angka

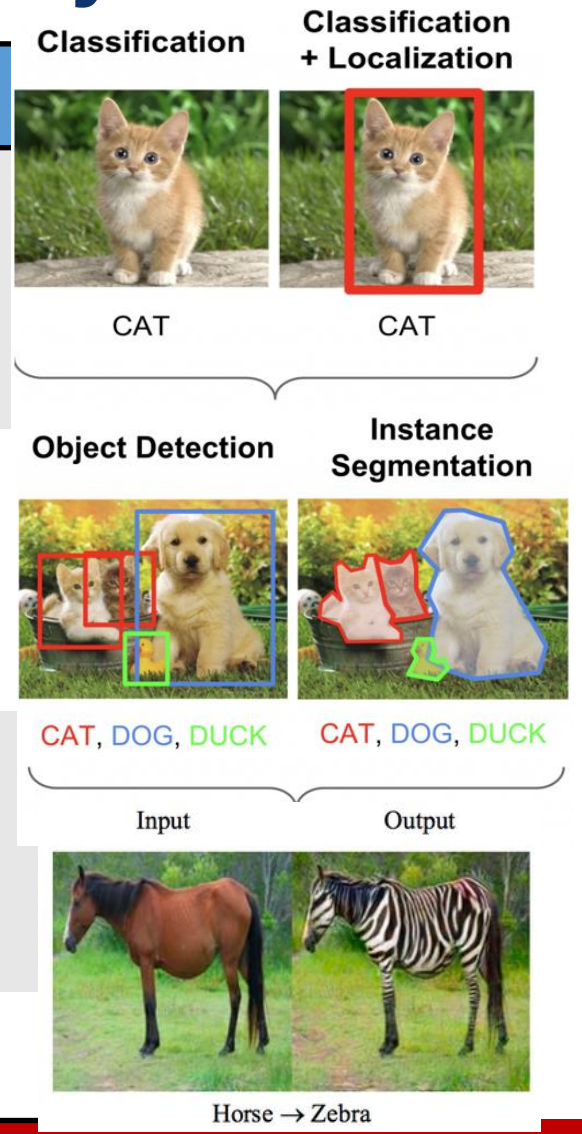
```
import matplotlib.pyplot as plt

epochs = range(10)
loss1 = history1.history['loss']
val_loss1 = history1.history['val_loss']
plt.plot(epochs,loss1,'r',label='train loss ANN')
plt.plot(epochs,val_loss1,'b',label='val loss ANN’)

loss2 = history2.history['loss']
val_loss2 = history2.history['val_loss']
plt.plot(epochs,loss2,'r+',label='train loss CNN')
plt.plot(epochs,val_loss2,'b+',label='val loss CNN')
plt.legend()
```

Perbandingan Loss dari Model CNN dan model ANN



Varian dari Arsitektur CNN dan Tipe Apikasinya

Aplikasi	Arsitektur CNN
Image Classification	• LeNet-5 (1998) • AlexNet (2012) • GoogLeNet/Inception (2014) • VGGNet (2014) • ResNet (2015)
Object Detection	• R-CNN (2013) • Fast R-CNN (2014) • Faster R-CNN (2015) • Single Shot Detector (SSD) (2016) • YOLO (2016), YOLOv3 (2018), YOLOv4 (2020), YOLOv5 (2020)
Semantic (Instance) Segmentation	• Fully Convolutional Network (FCN) (2015) • U-Net (2015) • Feature Pyramid Network (FPN) (2016) • Mask R-CNN (2017 • DeepLab (2016), DeepLabv3 (2017), DeepLabv3+ (2018)
Generative model	• Autoencoders, Variational Autoencoders (VAE) • Generative Adversarial Network (GAN)

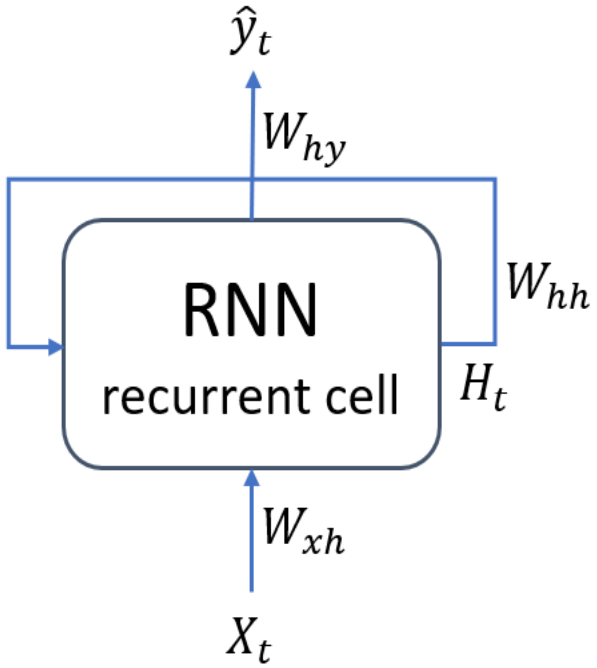


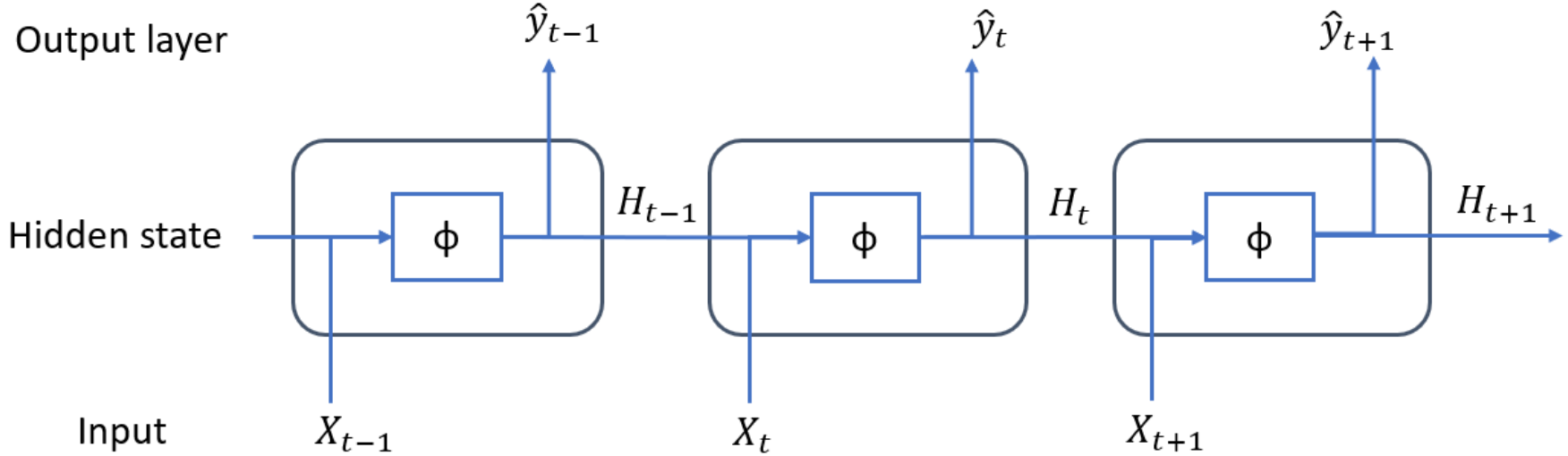
Recurrent Neural Network

Recurrent Neural Network (RNN) adalah salah satu arsitektur ANN yang mampu merepresentasikan data *sequential* misalnya teks, dna, suara, *time series,* dan sebagainya

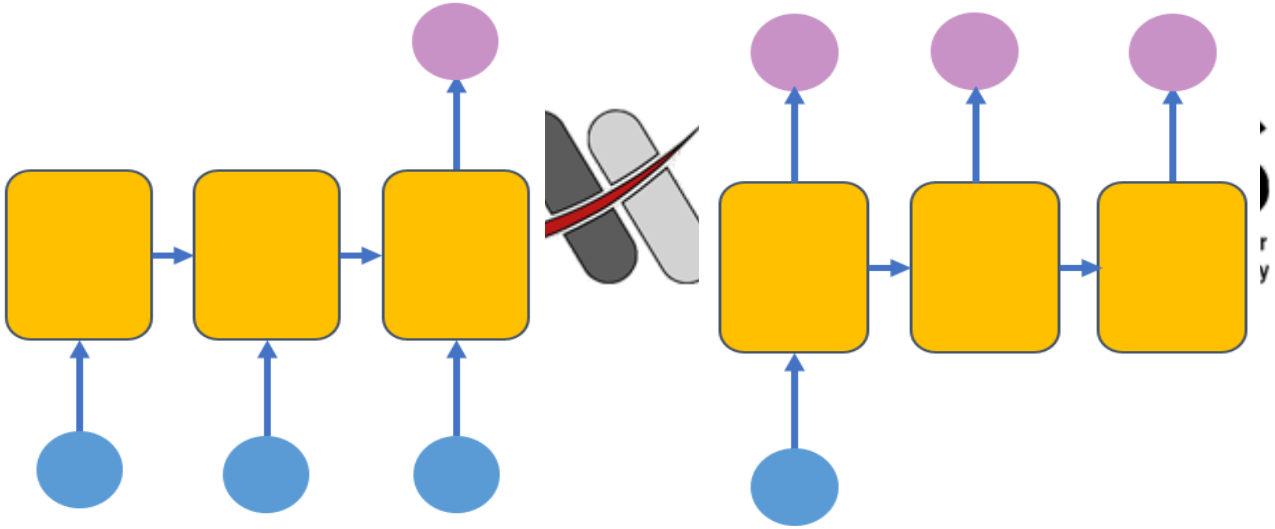
$$\hat{y}_t = H_t W_{hy} + b_y$$

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h)$$





Tipe arsitektur RNN dan aplikasinya



Many to One

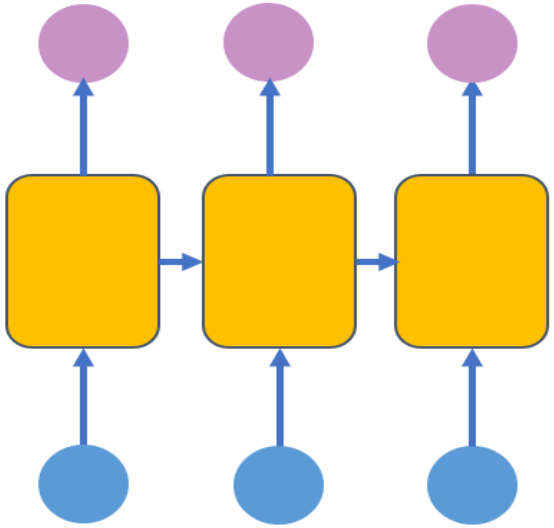
Applications:

- Sentiment classification
- Opinion mining
- Speech recognition
- Automated answer scoring

One to Many

Applications:

- Image captioning
- Text generation

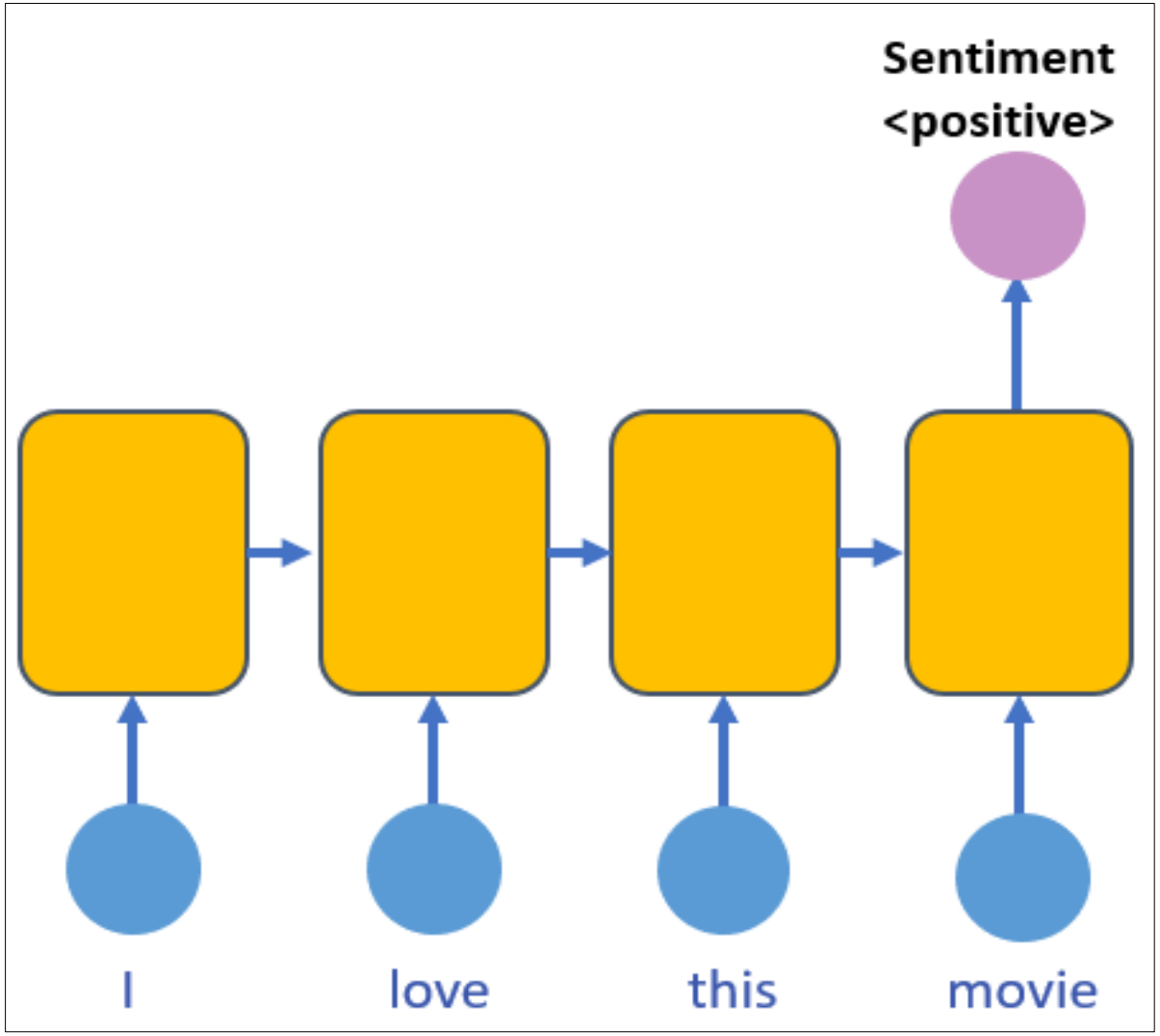


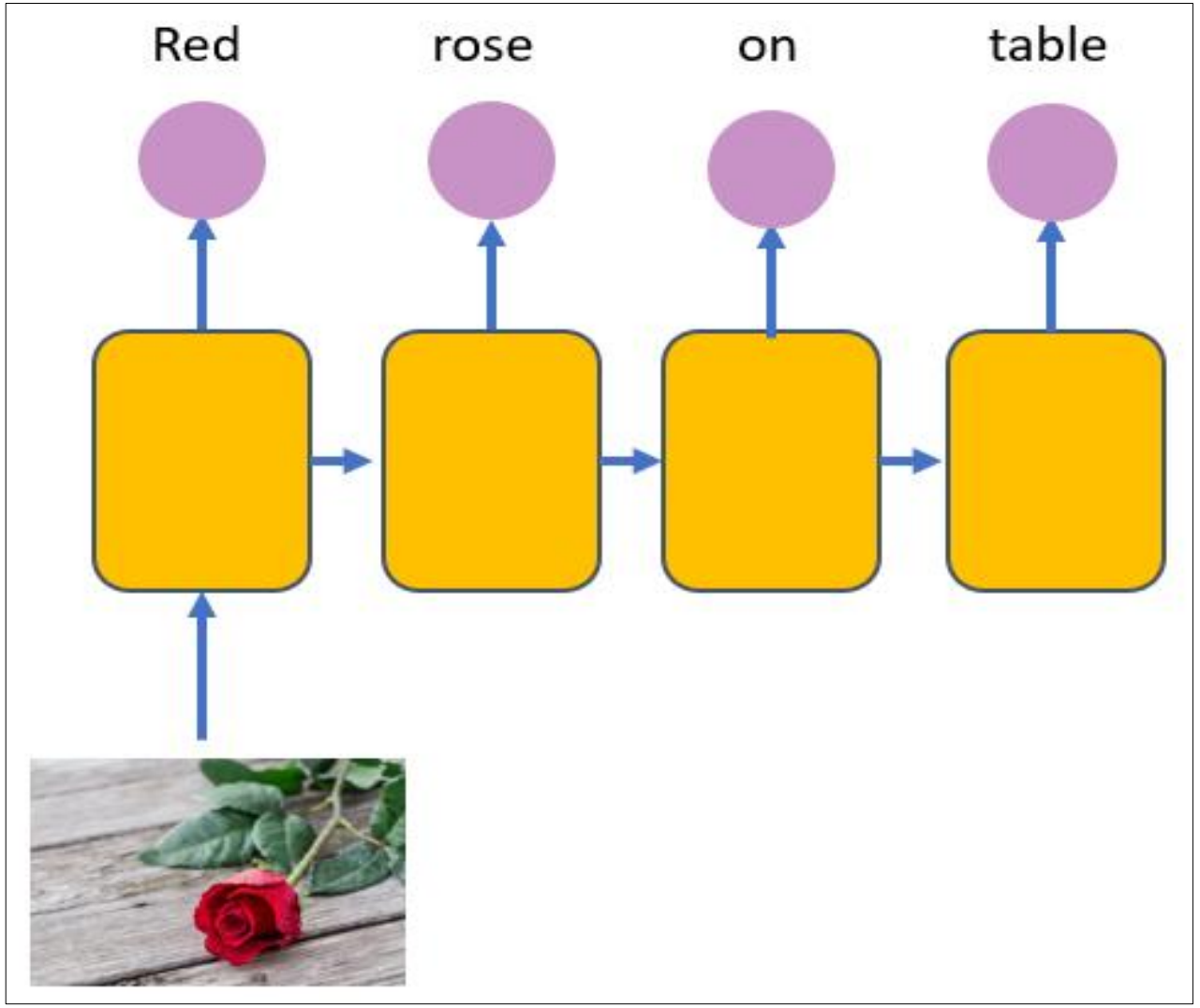
Many to Many

Applications:

- Translation
- Forecasting
- Chatbot
- Music generation

Contoh Aplikasi: *Sentiment classification* & *Image captioning*

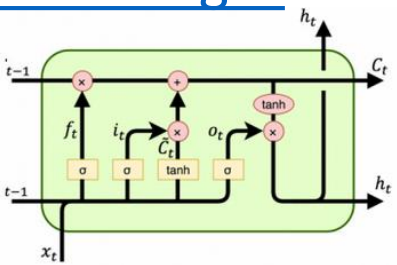


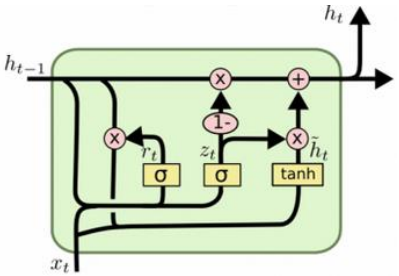


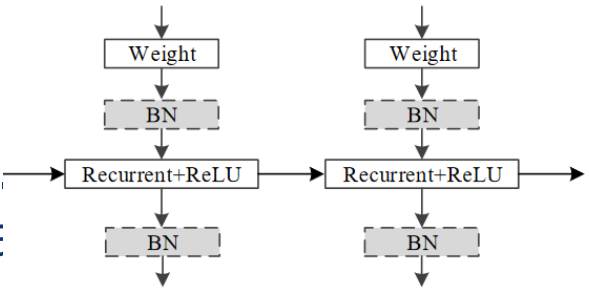
Varian Arsitektur RNN

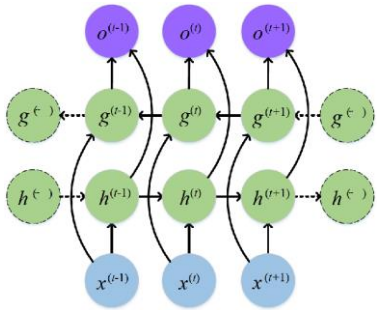
http://dprogrammer.org/rnn-lstm-gru

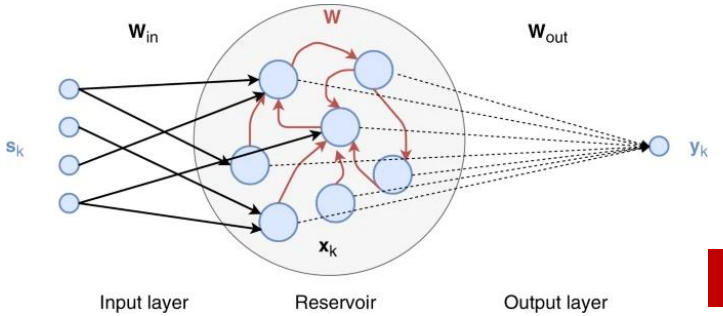
- **Long Short-Term Memory (LSTM):** merupakan salah satu jenis arsitektur RNN yang terdiri dari beberapa unit yaitu input gate, output gate, dan forget gate
- **Gate Recurrent Unit (GRU): m**erupakan simplifikasi dari arsitektur LSTM dengan menggabungkan input gate dan forget gate sehingga jumlah parameter lebih sedikit
- **Independently RNN (IndRNN): a**rsitektur RNN dimana setiap neuron dalam satu layer independen dari yang lain
- **Bi-directional RNN**: merupakan arsitektur RNN menghubungkan dua hidden layer dari arah yang berlawanan ke output yang sama
- **Echo State Network (ESN)**: ide dasar ESN adalah untuk membua jaringan berulang yang terhubung secara acak, yang disebut reservoir











Summary

FCN	CNN	RNN
Data numerik	Data citra, video	Data text, signal, suara, time-series
Jumlah hidden layer sesuai kompleksitas permasalahan	Convolution & Pooling layer	Konsep recurrent dan memperhatikan urutan input
Klasifikasi dan regresi	Klasifikasi, deteksi obyek, *instance segmentation*, generate citra sintetis	Klasifikasi, regresi, *generate text*, *translation*

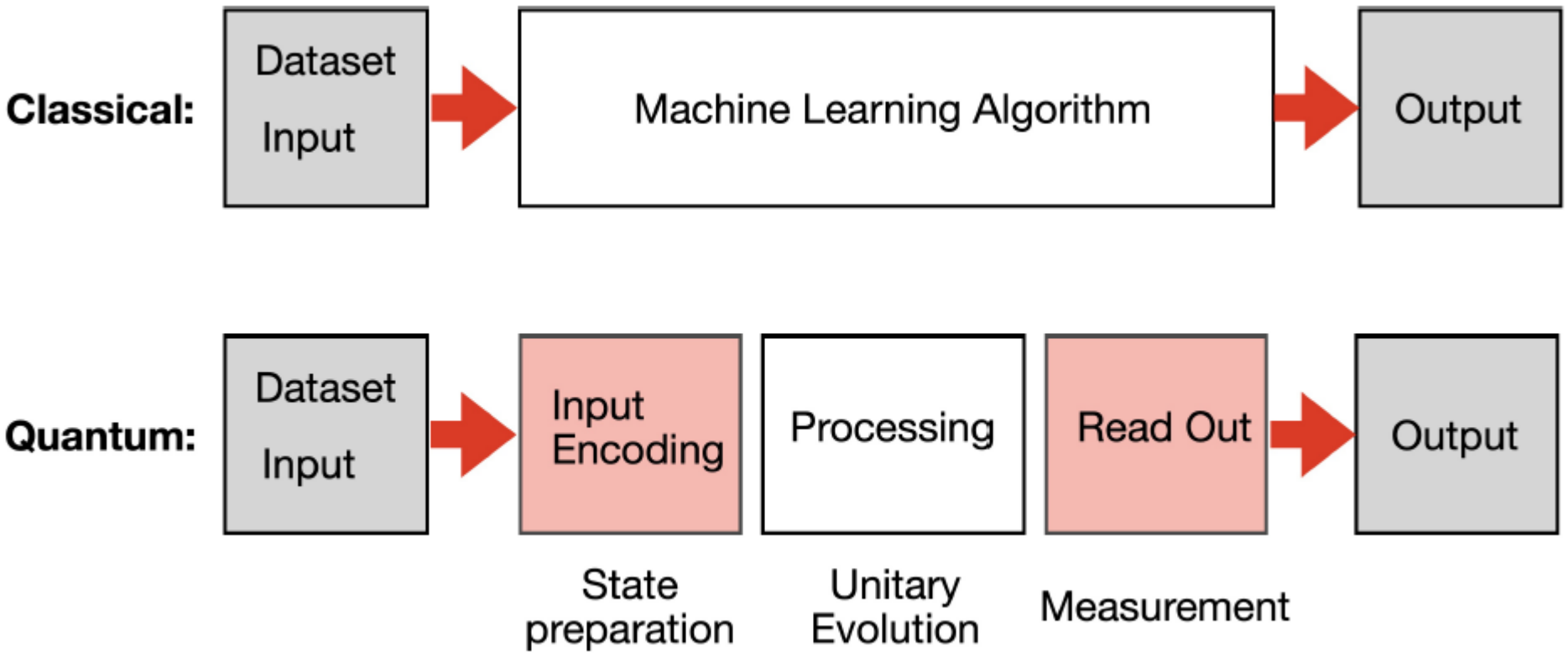
Project 2

Given 16 images with 4 class: p,e,b,a. Set the resolution all images to 128x128. Using 1 hidden layer with 32 neuron with relu activation function, output layer with tanh activation function. Training percentage : 75%.

1. Design ANN using above ANN parameters
2. Design deep learning using above ANN parameters and using 16 5x5 kernel 2D convolution with relu activation, maxpooling 2D 2x2, then using 64 3x3 kernel 2D convolution with relu activation, maxpooling 2D 2x2
3. Calculate the confusion matrix !

Quantum Machine Learning : cBPSK Case

Quantum ML System Model



Ref. : Anwar, Khoirul, and Mohamad Yusoff bin Alias. "Quantum Machine Learning for Demappers of Low Order Modulations of 5G and Beyond."

cBPSK : complex-BPSK

$$x(i) = \frac{1-2b(i)}{\sqrt{2}} + j\frac{1-2b(i)}{\sqrt{2}}$$

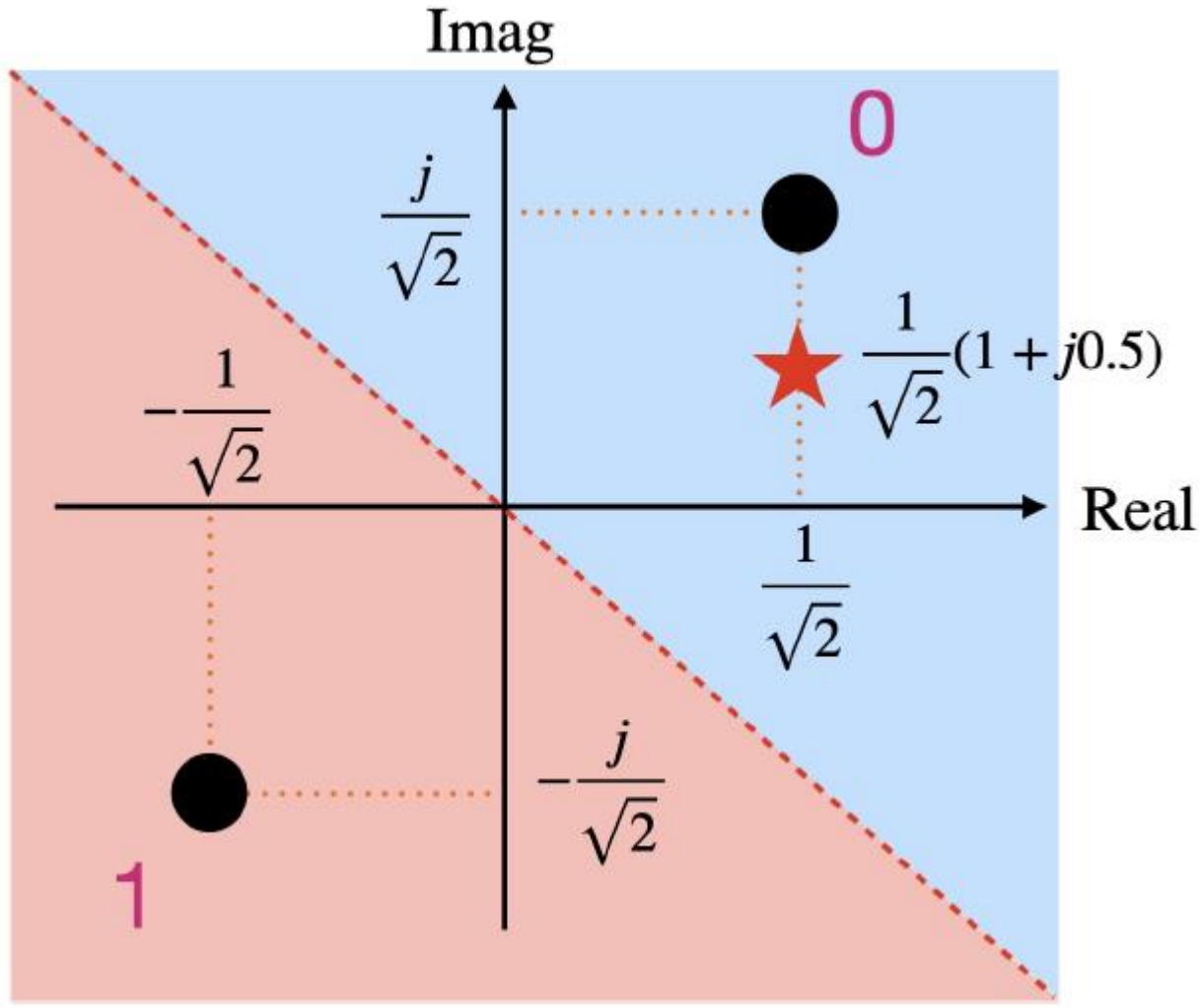
$$b(i) = \{0, 1\}$$

$$x(i) = \left\{ \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \right\}$$

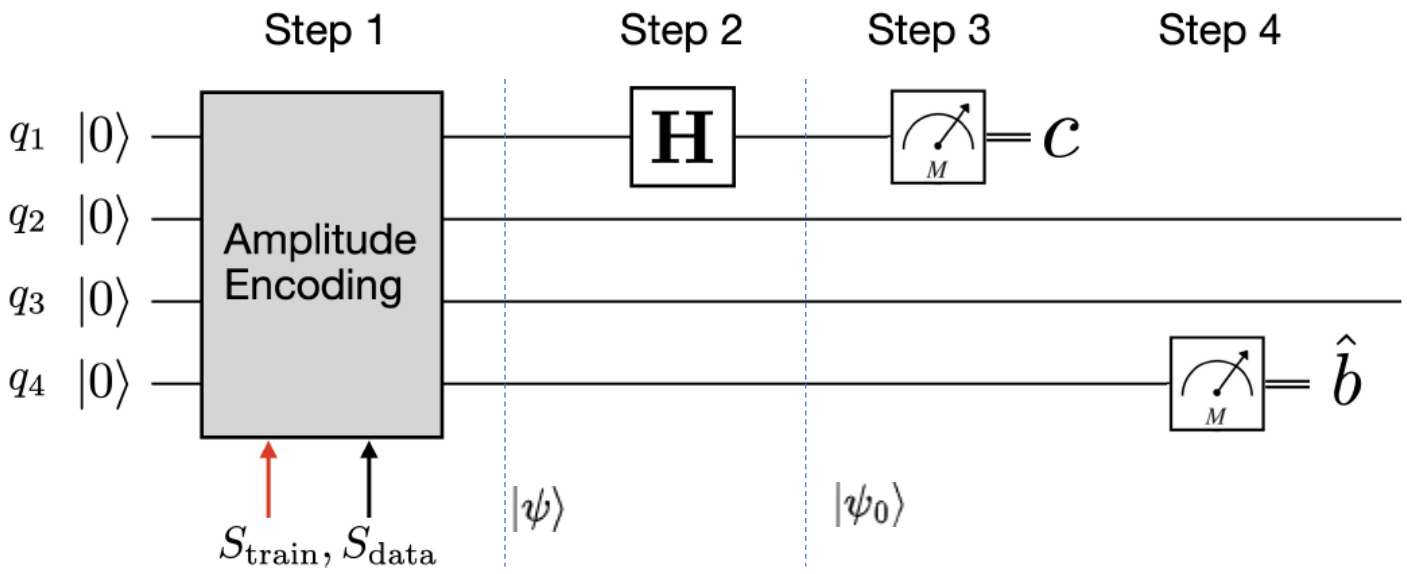
Problem : After distorted by noise the received signal becomes

$$y(i) = \frac{1}{\sqrt{2}} + j\frac{0.5}{\sqrt{2}}$$

Target : Create ML model in receiver to demodulate y(i)



ML Model in Receiver



$$|\psi_0\rangle = \mathbf{H} \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I}|\psi\rangle = \mathbf{HIII}|\psi\rangle$$

$$\mathbf{M}_{10} = \mathbf{M}_0\mathbf{III} = |0\rangle\langle 0| \otimes \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I}$$

$$P_{10} = \langle\psi_0|\mathbf{M}_{10}|\psi_0\rangle$$

$$|\psi_0'\rangle = \frac{\mathbf{M}_{10}|\psi_0\rangle}{\sqrt{P_{10}}}$$

$$\mathbf{M}_{40} = \mathbf{III}\mathbf{M}_0|\psi\rangle = \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{I} \otimes |0\rangle\langle 0||\psi\rangle$$

$$P_{40} = \langle\psi_0'|\mathbf{M}_{40}|\psi_0'\rangle$$

$$S_{train} = \{x_1, x_2\} = \left\{\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}\right\}$$

$$S_{data} = y = \frac{1}{\sqrt{2}} + j\frac{0.5}{\sqrt{2}}$$

$$\hat{b} = \begin{cases} 0, & P_{40} \geq 0.5 \\ 1, & P_{40} < 0.5 \end{cases}$$

$$|\psi\rangle = |q_1 q_2 q_3 q_4\rangle$$

$$= \Re(x_1)|0000\rangle + \Im(x_1)|0010\rangle + \Re(x_2)|0101\rangle + \Im(x_2)|0111\rangle + \Re(y)|1000\rangle + \Im(y)|1010\rangle + \Re(y)|1101\rangle + \Im(y)|1111\rangle$$

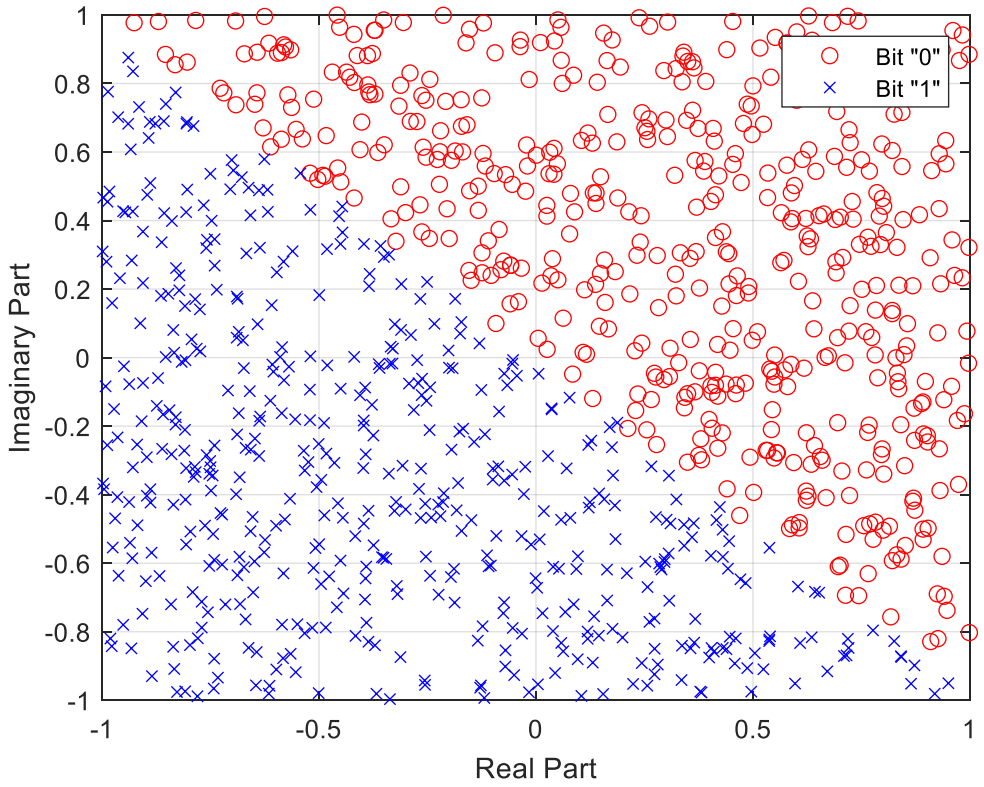
$$= \frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{\sqrt{2}}|0010\rangle - \frac{1}{\sqrt{2}}|0101\rangle - \frac{1}{\sqrt{2}}|0111\rangle + \frac{1}{\sqrt{2}}|1000\rangle + \frac{1}{2\sqrt{2}}|1010\rangle + \frac{1}{\sqrt{2}}|1101\rangle + \frac{1}{2\sqrt{2}}|1111\rangle$$

Result

$$x(i) = \frac{1-2b(i)}{\sqrt{2}} + j\frac{1-2b(i)}{\sqrt{2}}$$

$$b(i) = \{0, 1\}$$

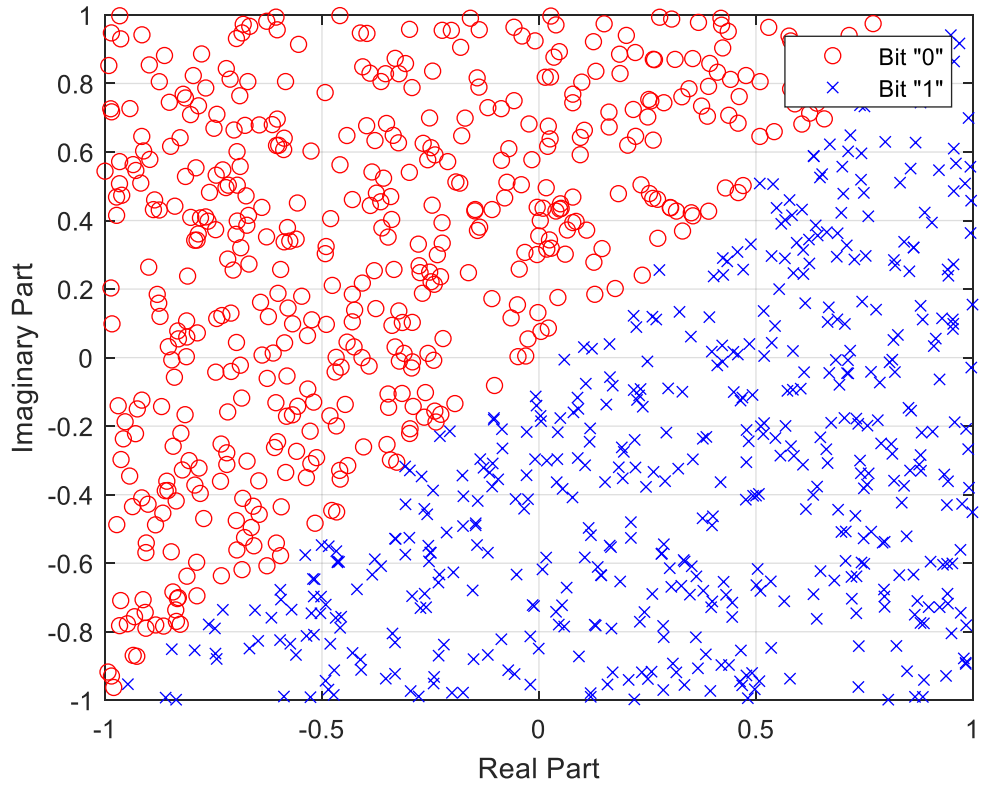
$$x(i) = \left\{ \frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \right\}$$



$$x(i) = -\frac{1-2b(i)}{\sqrt{2}} + j\frac{1-2b(i)}{\sqrt{2}}$$

$$b(i) = \{0, 1\}$$

$$x(i) = \left\{ -\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}} \right\}$$



Thank you