Deep Learning – Supervised 1

Arrie Kurniawardhani

Crash Course ML|DL – 28 Februari 2020

UNIVERSITAS
ISLAM
INDONESIA

VALUES | INNOVATION | PERFECTION

Deep Learning ~ Supervised ~

Content :

1. Neural Network
2. Learning Parameter
3. Adam
4. Regularization
5. Convolutional neural network

Neural Network

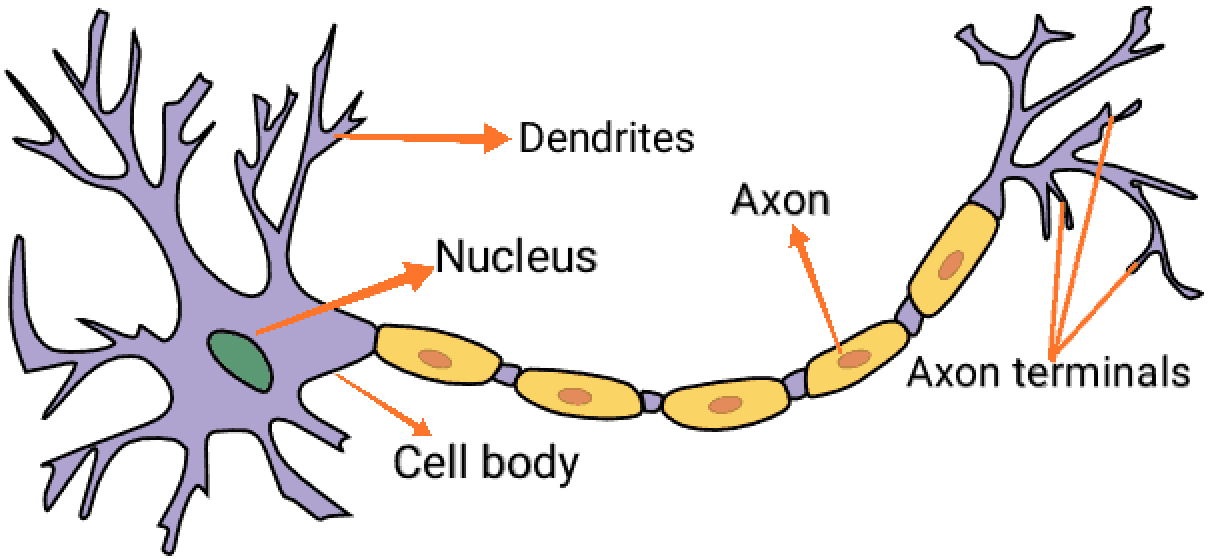
ANNs have been successfully applied in wide range of domains such as:

1. Classification of data – Is this flower a rose or tulip?
2. Anomaly detection – Is the particular user activity on the website a potential fraudulent behavior?
3. Speech recognition – Hey Siri! Can you tell me a joke?
4. Audio generation – Jukedeck, can you compose an uplifting folk song?
5. Time series analysis – Is it good time to start investing in stock market?

And the list goes on…

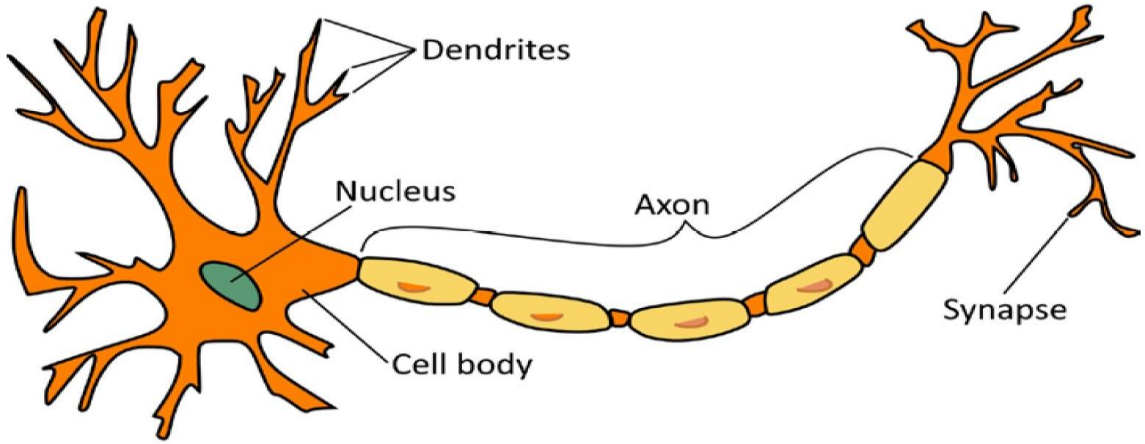
Neural Network

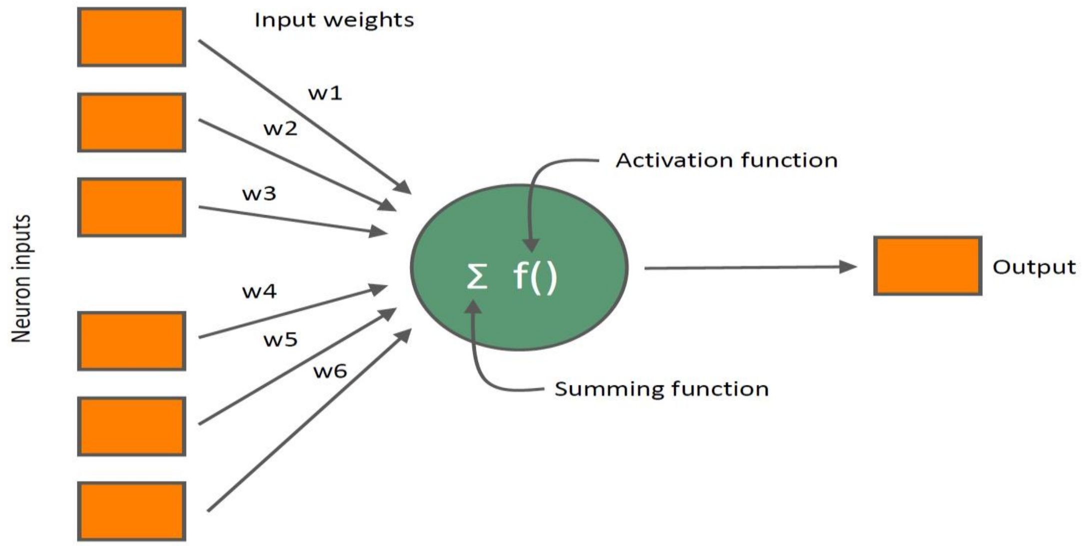
1. Artificial neural network (or neural network for short)
2. A predictive model motivated by the way the brain operates



[1]

Neural Network

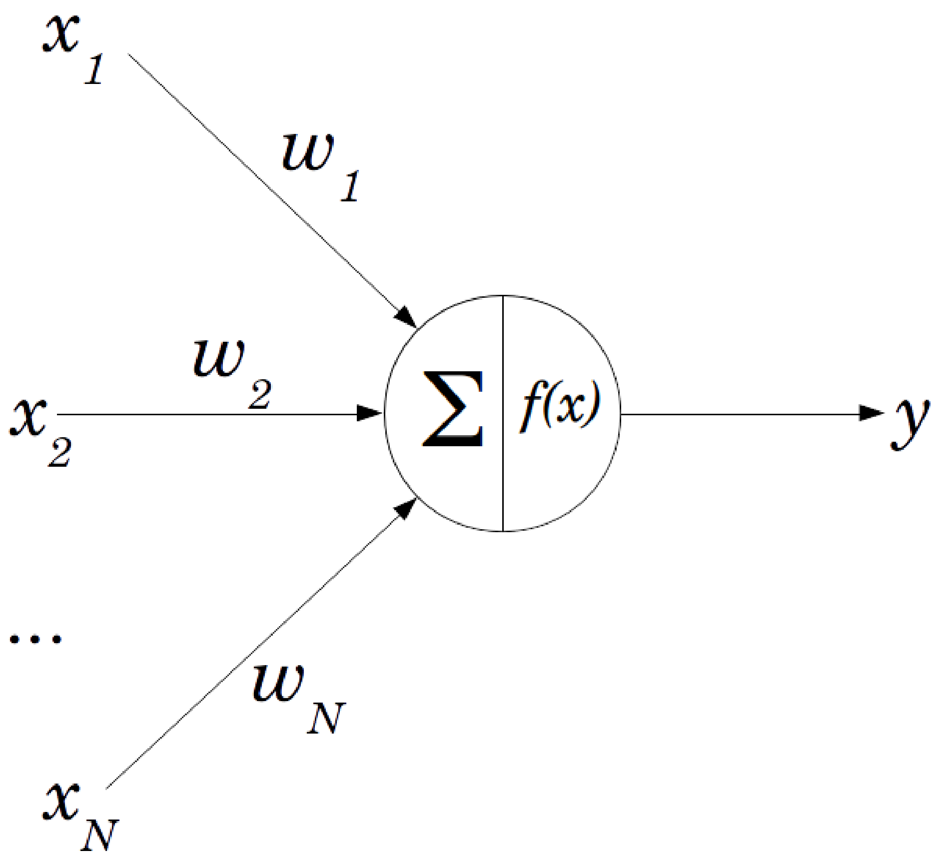




[2]

Neural Network

~ McCulloch-Pitts model of Neuron (1943 model)



[3]

- x = node input
- w = Weight
- f(x) = activation function

$$y = f\left(\sum_{i=1}^{n}(x_i w_i)\right)$$

Neural Network

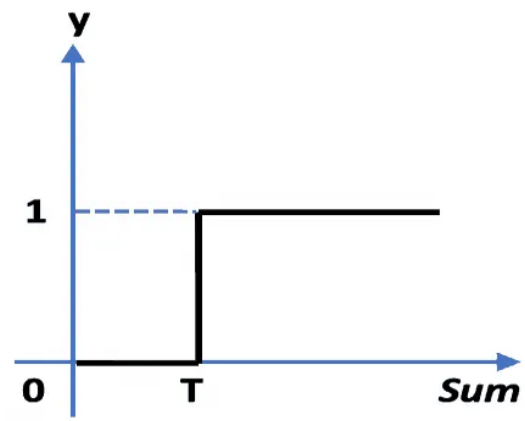
» The summing function is given by:

$$y = x_1 w_1 + x_2 w_2 + \ ... + x_n w_n = \sum_{i=1}^{n} (x_i w_i)$$

» The activation function is given by:

$$f\left(\sum_{i=1}^{n} (x_i w_i)\right)$$

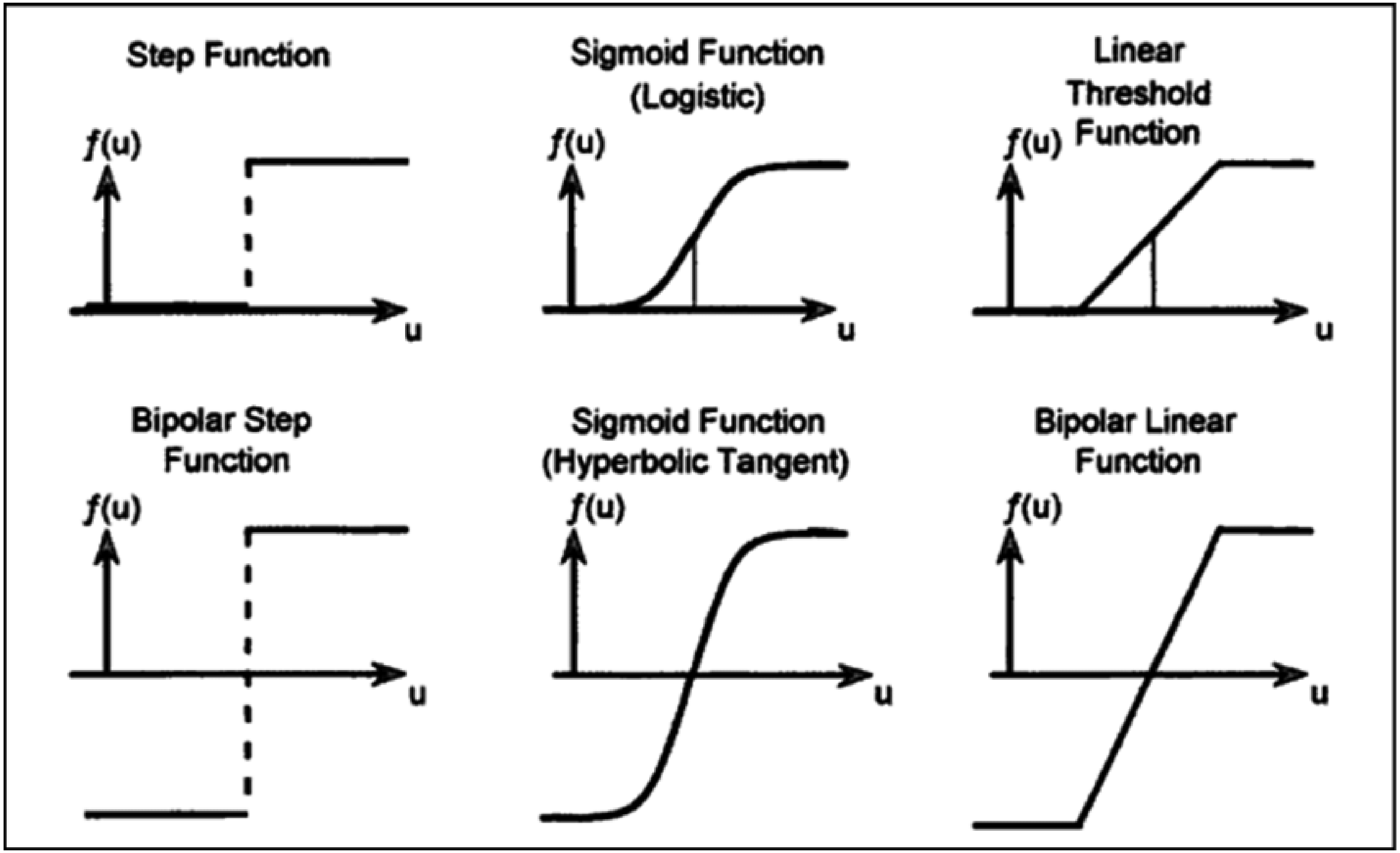
» Output: $y = \begin{cases} 0, & f(x) < T \\ 1, & f(x) \geq T \end{cases}$



Activation function

1. a mathematical “gate”
2. Mathematical equations that determine the output of a neural network
3. Determines whether it should be activated (“fired”) or not
4. Also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1

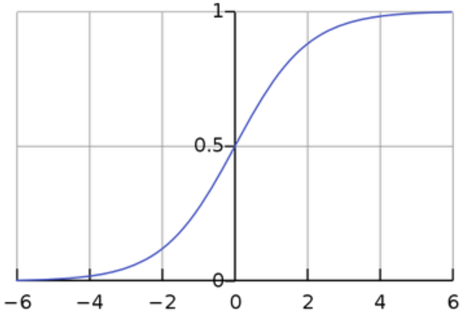
Activation function



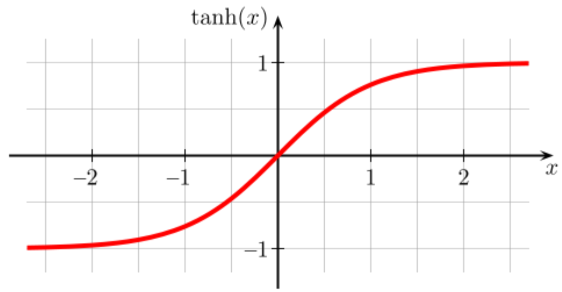
[4]

Activation function

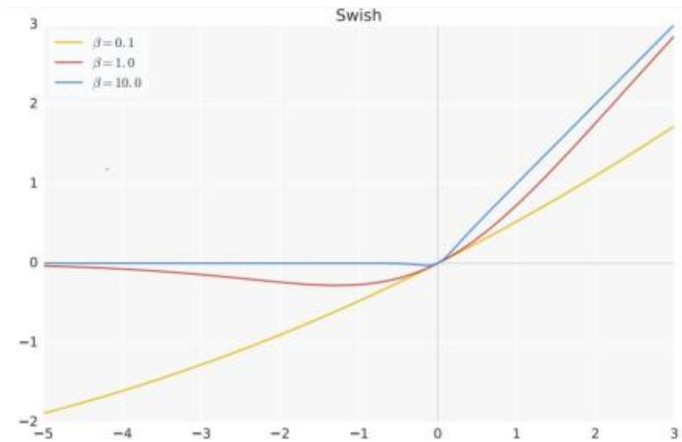
Sigmoid



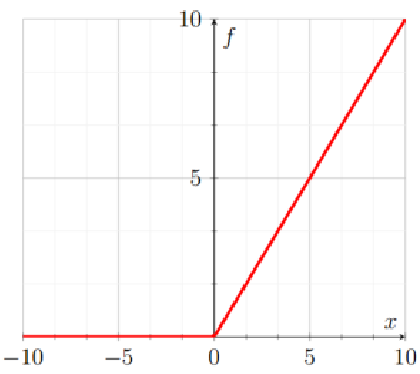
Tanh

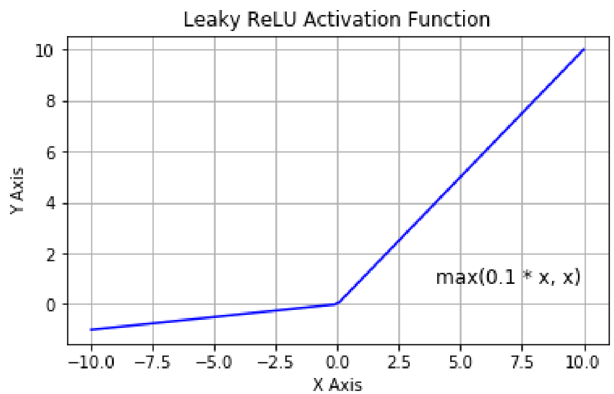


Swish



ReLU





[5]

Activation function

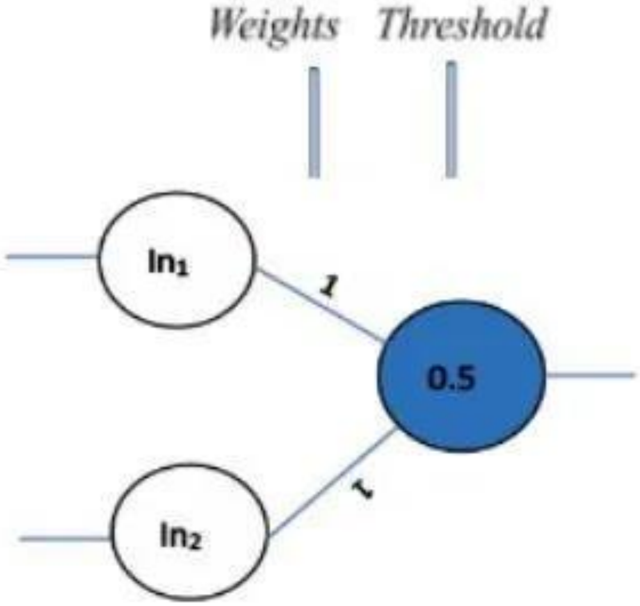
1. The softmax is a popular choice for the output layer activation and should not be used for a regression task as well.
2. Sigmoid and Tanh were widely used as hidden layer activation functions, but those suffer vanishing gradient problem.
3. ReLU is the most popular and commonly used as hidden layer activation.

Neural Network

- Contoh:

OR Gate

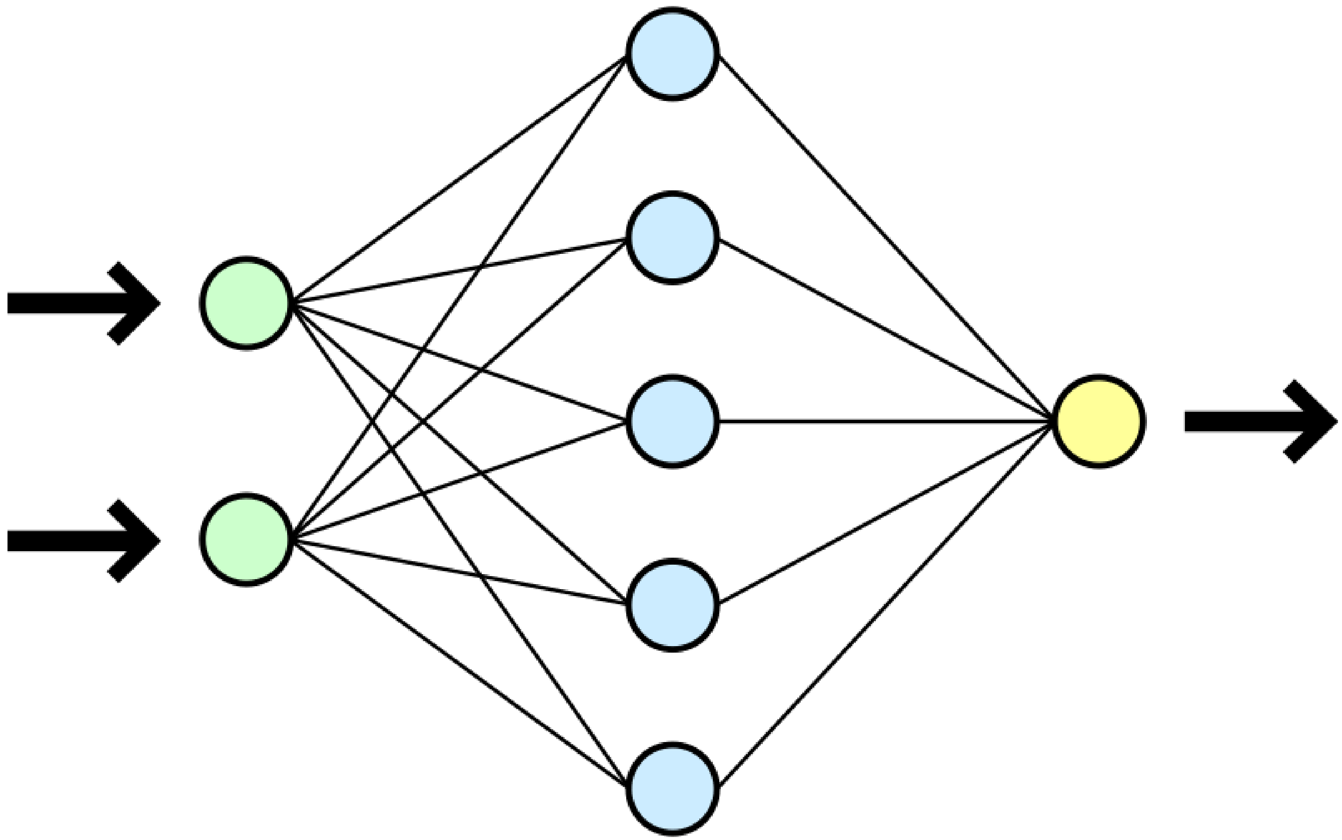
In_1	In_2	Out
0	0	0
1	0	1
0	1	1
1	1	1



[1]

Neural Network

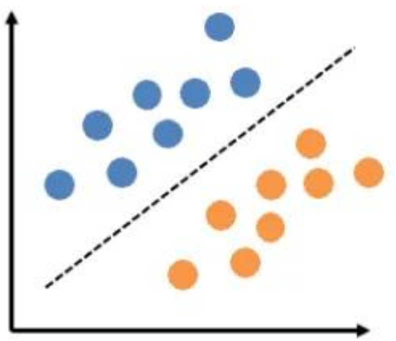
More neuron



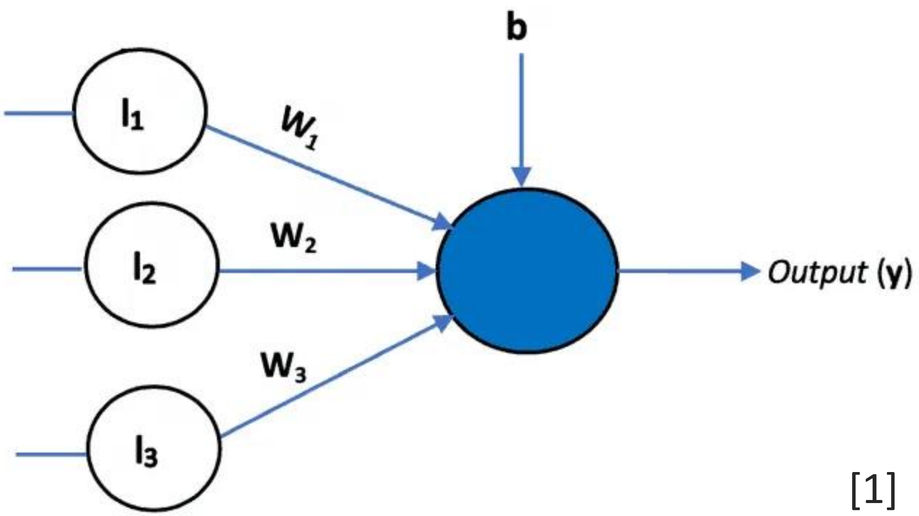
[6]

Perceptron

1. The simplest type of neural network that helps with linear (or binary) classifications of data



2. The learning rule for training the neural network was first introduced with this model



[1]

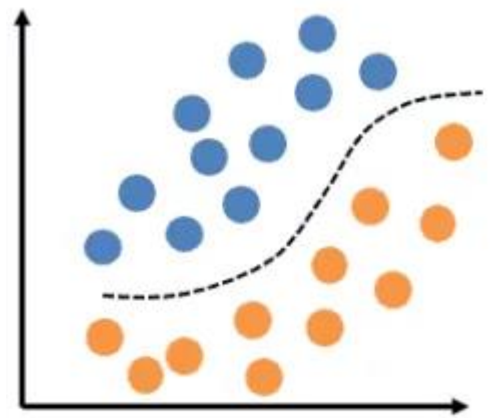
Perceptron

1. In addition to the variable weight values, the perceptron added an extra input that represents bias.
2. **Bias** is used to adjust the output of the neuron along with the weighted sum of the inputs.
3. It's just like the intercept added in a linear equation.

$$y = f\left(w_0 + \sum_{i=1}^{n}(x_i w_i)\right) \quad \text{or} \quad y = f\left(b + \sum_{i=1}^{n}(x_i w_i)\right)$$

Multilayer Perceptron

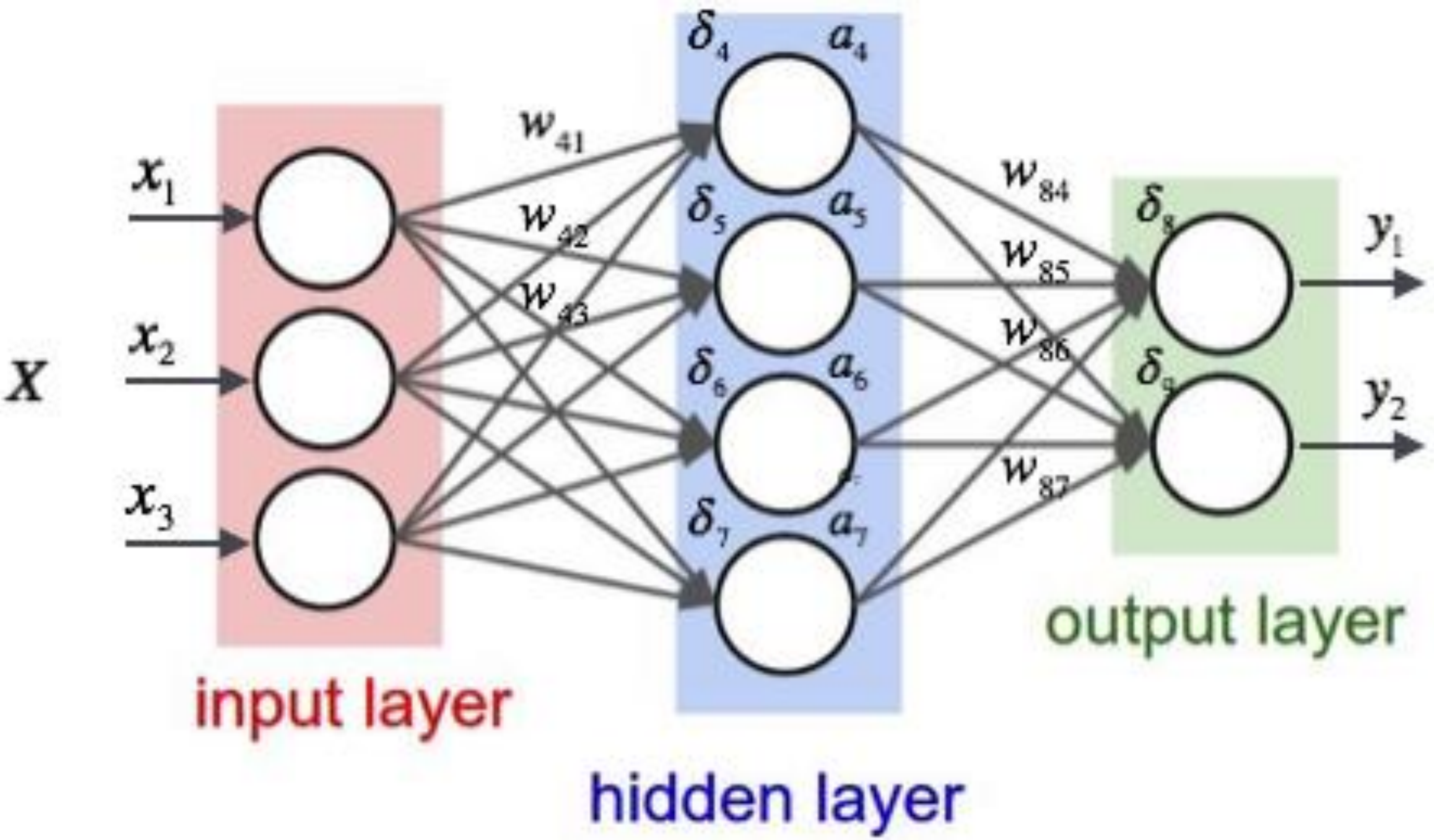
1. Input data is not linearly separable
2. Need more linear line to make non-linear line, thus add hidden layers
3. A non-linear activation function such as sigmoid



[1]

UNIVERSITAS ISLAM INDONESIA

Multilayer Perceptron



[7]

Training phase

a) Need to determine 'weight' and 'bias' value through training phase

b) The entire goal of training a neural network is to minimize **<u>error</u>** (difference in predicted and expected outputs) by adjusting its weights and biases.

c) Training process is terminated when our model's predicted output is almost same as the expected output

Training phase

1. **Initialize the weights**

   a. Initialized to small random numbers (e.g., ranging from -1 to 1, or -0.5 to 0.5).

   b. Each unit has a bias associated with it, and the biases are similarly initialized to small random numbers.

Training phase

1. Initialize the weights
2. **Propagate the input forward**
   a. The weighted sum of input values is calculated
   b. The result is passed to an activation function (e.g Sigmoid)

Training phase

1. Initialize the weights
2. Propagate the input forward
3. **Calculate the error**
   a. Calculate the error, i.e., the difference between our predicted output and expected output
   b. Loss function (Mean Squared Error)

$$e_j = \sum_{j=1}^{m} \left(y_j - \widehat{y_j}\right)^2$$

$\widehat{y_j}$ = predicted output

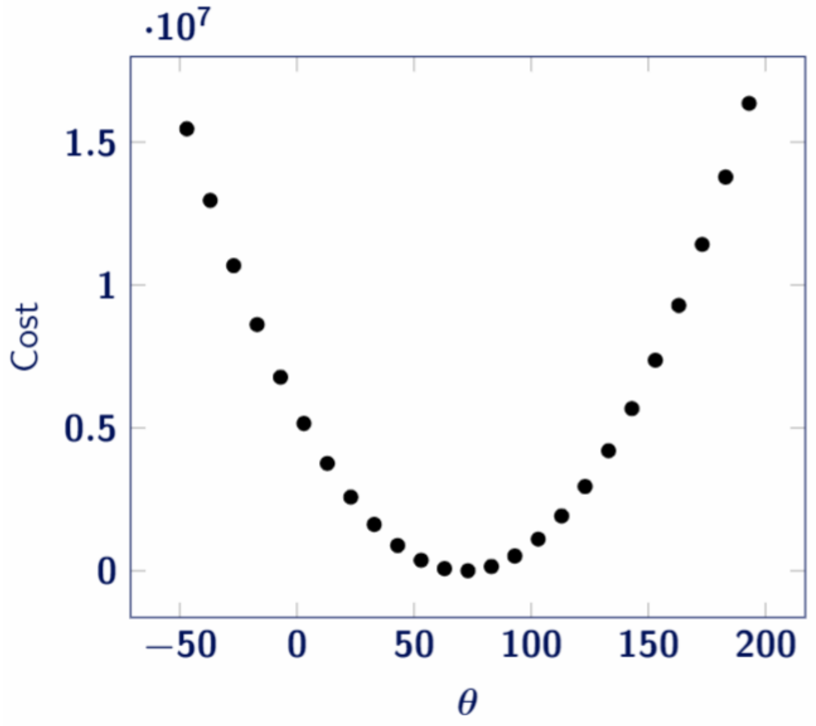
y_j = expected output

Training phase

» Error/cost function:

$$e_j = \sum_{j=1}^{m} \left(y_j - \widehat{y_j}\right)^2$$

$$e(w)_j = \frac{1}{2} \sum_{j=1}^{m} \left(y(w)_j - \widehat{y_j}\right)^2$$



» Gradient descent:

$$w'_i = w_i - \sigma \frac{\partial}{\partial w_i} e(w)$$

Training phase

- Derivative the error:

$$\frac{\partial}{\partial w_i} e(w) = \frac{1}{2} \frac{\partial}{\partial w_i} \left(y(w)_j - \widehat{y_j} \right)^2$$

- (fog)’ = (f’og)g’)

$$\frac{\partial}{\partial w_i} e(w) = \frac{2}{2} \left(y(w)_j - \widehat{y_j} \right) \frac{\partial}{\partial w_i} \left(y(w)_j - \widehat{y_j} \right)$$

$$\frac{\partial}{\partial w_i} e(w) = -\left(y(w)_j - \widehat{y_j} \right) \frac{\partial}{\partial w_i} \widehat{y_j}$$

$$\frac{\partial}{\partial w_i} e(w) = -\left(y(w)_j - \widehat{y_j} \right) \frac{\partial}{\partial w_i} (x_1 w_1 + x_2 w_2 + \ \ldots + x_n w_n)$$

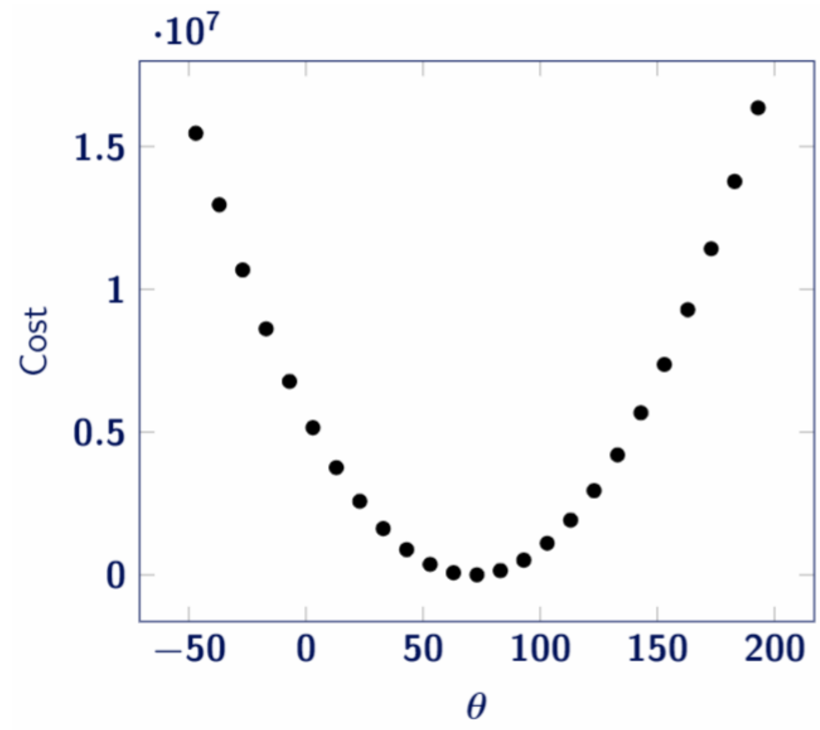
$$\frac{\partial}{\partial w_i} e(w) = -\left(y(w)_j - \widehat{y_j} \right) * x_i$$

Training phase

- Gradient descent:

$$w'_i := w_i - \sigma \frac{\partial}{\partial w_i} e(w)$$

$$w'_i = w_i - \sigma\left(y(w)_j - \widehat{y_j}\right) * x_i$$



UNIVERSITAS ISLAM INDONESIA

Training phase

1. Initialize the weights

2. Propagate the input forward

3. Calculate the error

4. **Backpropagate the error**, update the weight

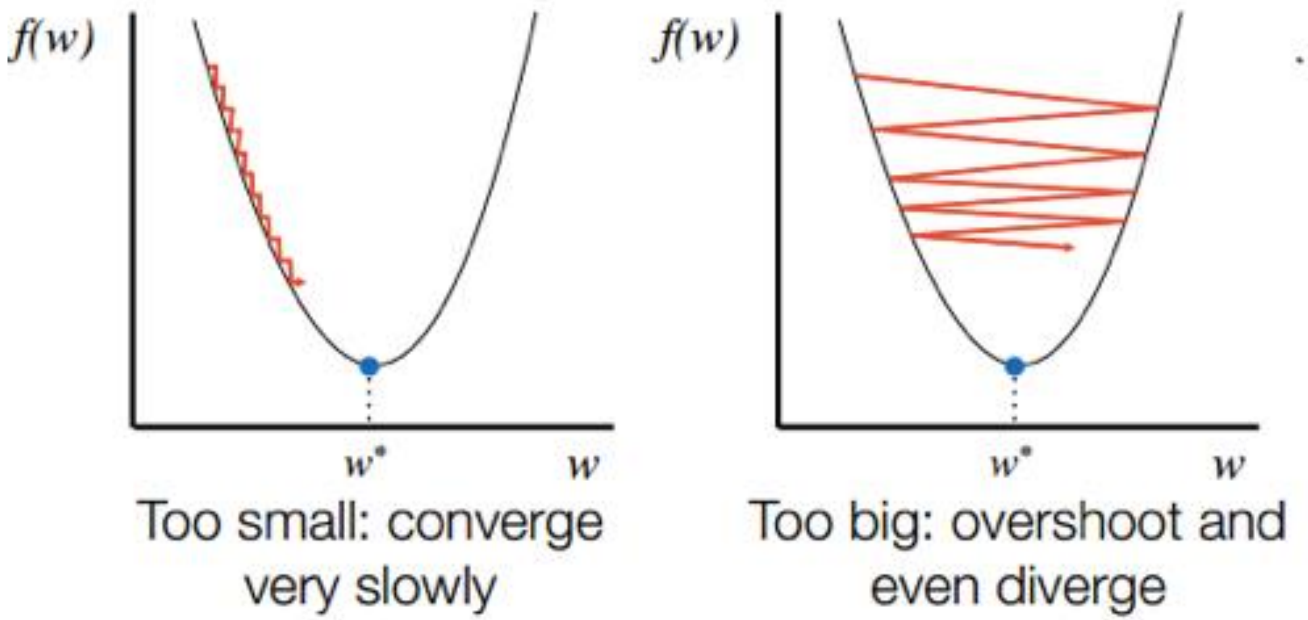
   a. New weight : $w_{ij} = w_{ij} + (\sigma * e_j * x_i)$

   b. New bias : $b_i = b_j + (\sigma * e_j)$

   c. σ = Learning rate

Learning rate

1. A constant typically varying between 0 to 1
2. It decides the rate at which the value of weights and bias should vary/change.



[8]

Training phase

- Steps 2 and 4 are repeated until one of the following terminating conditions is met:
  a. The error is minimized to the least possible value
  b. The training has gone through the maximum number of epochs
  c. There is no further reduction in error value
  d. The training error is almost same as that of validation error

Learning Parameter

1. Parameters
   a. The coefficients of the model, need to be set, and updated by the model itself.
   b. Weight, bias
2. Hyperparameters
   a. The coefficients of the model, need to be set, But the model will not update them
   b. Learning rate, number of hidden layer, number of neurons, type of activation function, etc

Adam Optimization

1. Adaptive Moment Estimation
2. Gradient descent – stochastic gradient descent – AdaGrad (Adaptive Gradient Algorithm) – RMSprop (Root mean square propagation) – Adam
3. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training.
4. Adam maintains a single learning rate for each network weight (parameter) and separately adapted as learning unfolds.

Adam Optimization

1. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.
2. Adam optimizer is often the best choice, since it allows you to set different hyperparameters and customize your NN.

Regularization

1. Keep our model simple and avoid overfitting
2. The idea is that regularization adds a penalty to the model if weights are great/too many.
3. Indeed, it adds to our loss function a new term which tends to increase (hence, the loss increases too) if the re-calibration procedure increases weights.
4. Trade off between number of feature and error

Regularization

There are two kinds of regularization:

1. Lasso regularization (L1)
2. Bridge regularization (L2)

$$Loss\ Function = L(\boldsymbol{w}) + \lambda \sum w_i^2$$

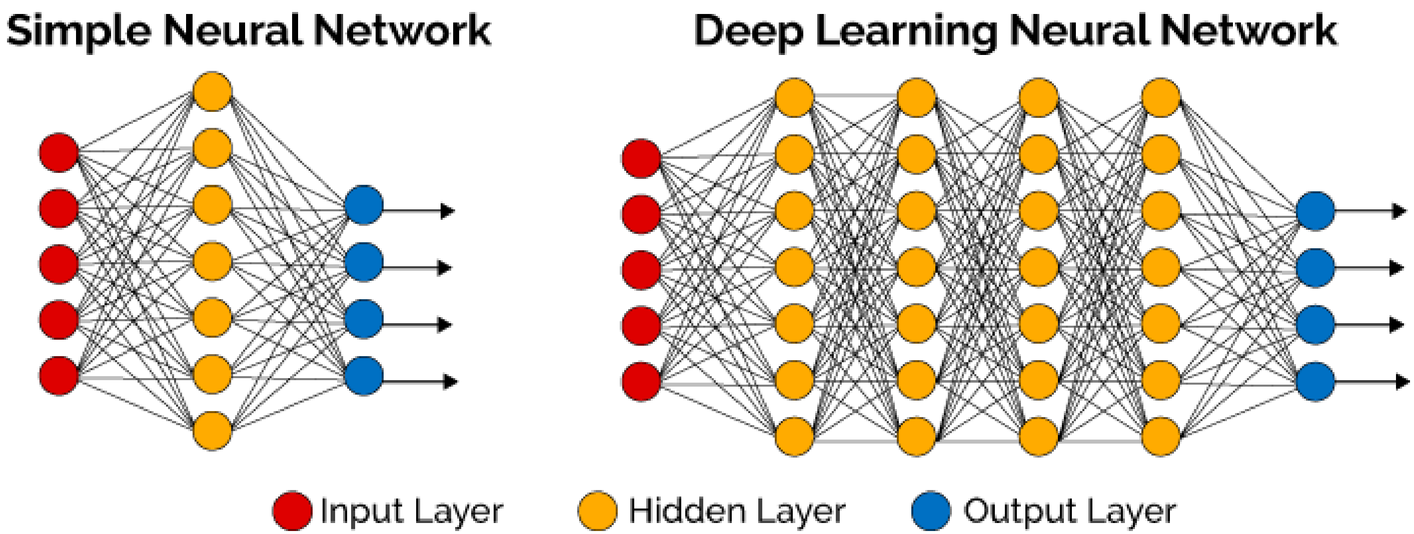
L2 Regularization

$$Loss\ Function = L(\boldsymbol{w}) + \lambda \sum |w_i|$$

L1 Regularization

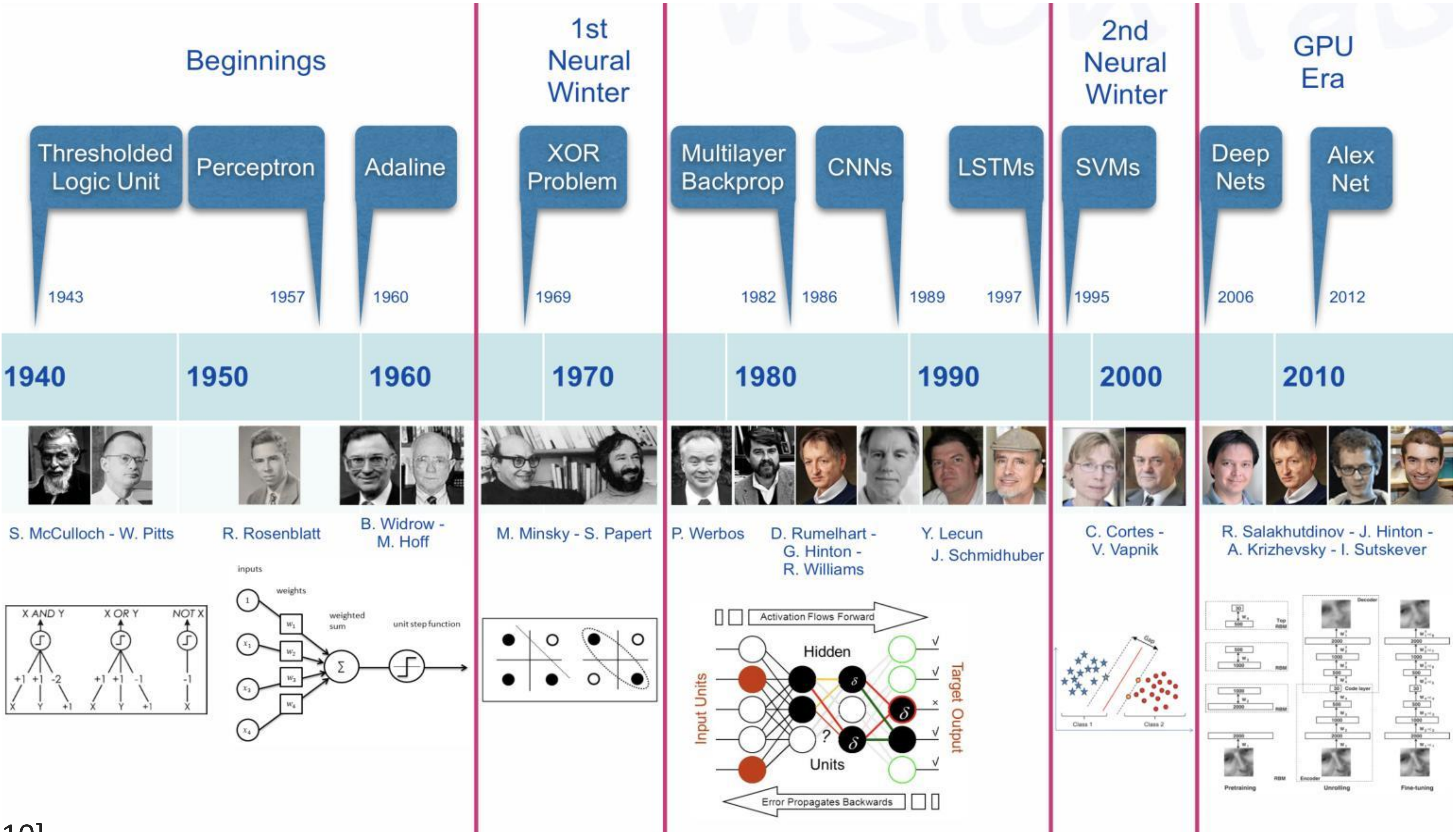
Deep Learning

1. Has more hidden layers of neurons.
2. Experts suggests that neural networks increase in accuracy with the number of hidden layers.



[9]

Deep Learning



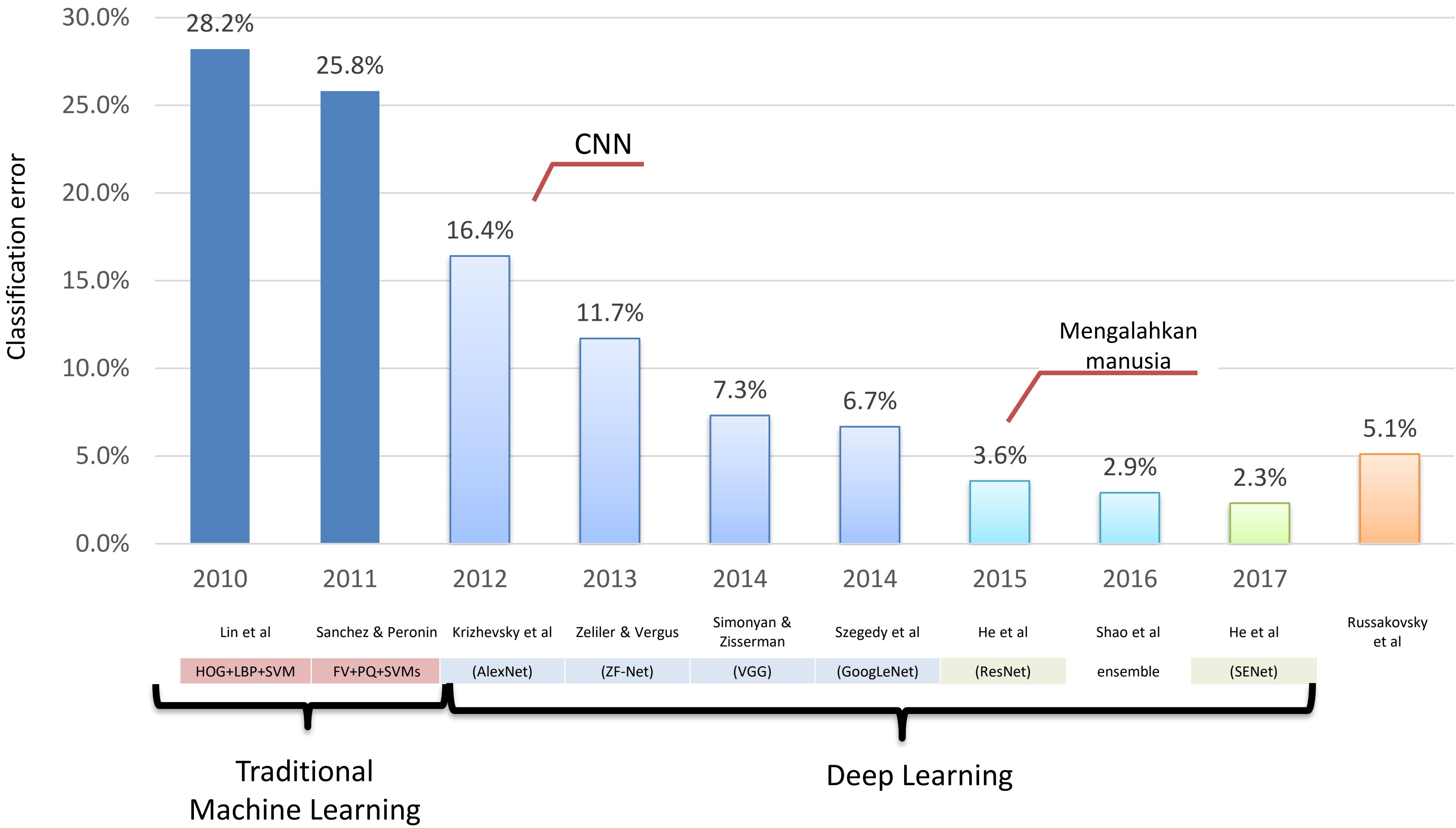
[10]

UNIVERSITAS ISLAM INDONESIA

CNN

1. ConvNet diperkenalkan oleh Yann LeCun et al pada 1988
2. Facebook uses neural nets for their automatic tagging algorithms
3. AlexNet membuat ConvNet menjadi populer saat memenangkan Imagenet Challenge 2012
4. Google for their photo search
5. Amazon for their product recommendations
6. Pinterest for their home feed personalization

CNN



Classification error
30.0%
25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
28.2%
25.8%
16.4%
11.7%
7.3%
6.7%
3.6%
2.9%
2.3%
5.1%
CNN
Mengalahkan manusia
2010
2011
2012
2013
2014
2014
2015
2016
2017
Lin et al
Sanchez & Peronin
Krizhevsky et al
Zeliler & Vergus
Simonyan & Zisserman
Szegedy et al
He et al
Shao et al
He et al
Russakovsky et al
HOG+LBP+SVM
FV+PQ+SVMs
(AlexNet)
(ZF-Net)
(VGG)
(GoogLeNet)
(ResNet)
ensemble
(SENet)
Traditional Machine Learning
Deep Learning

Image Classification

What We See

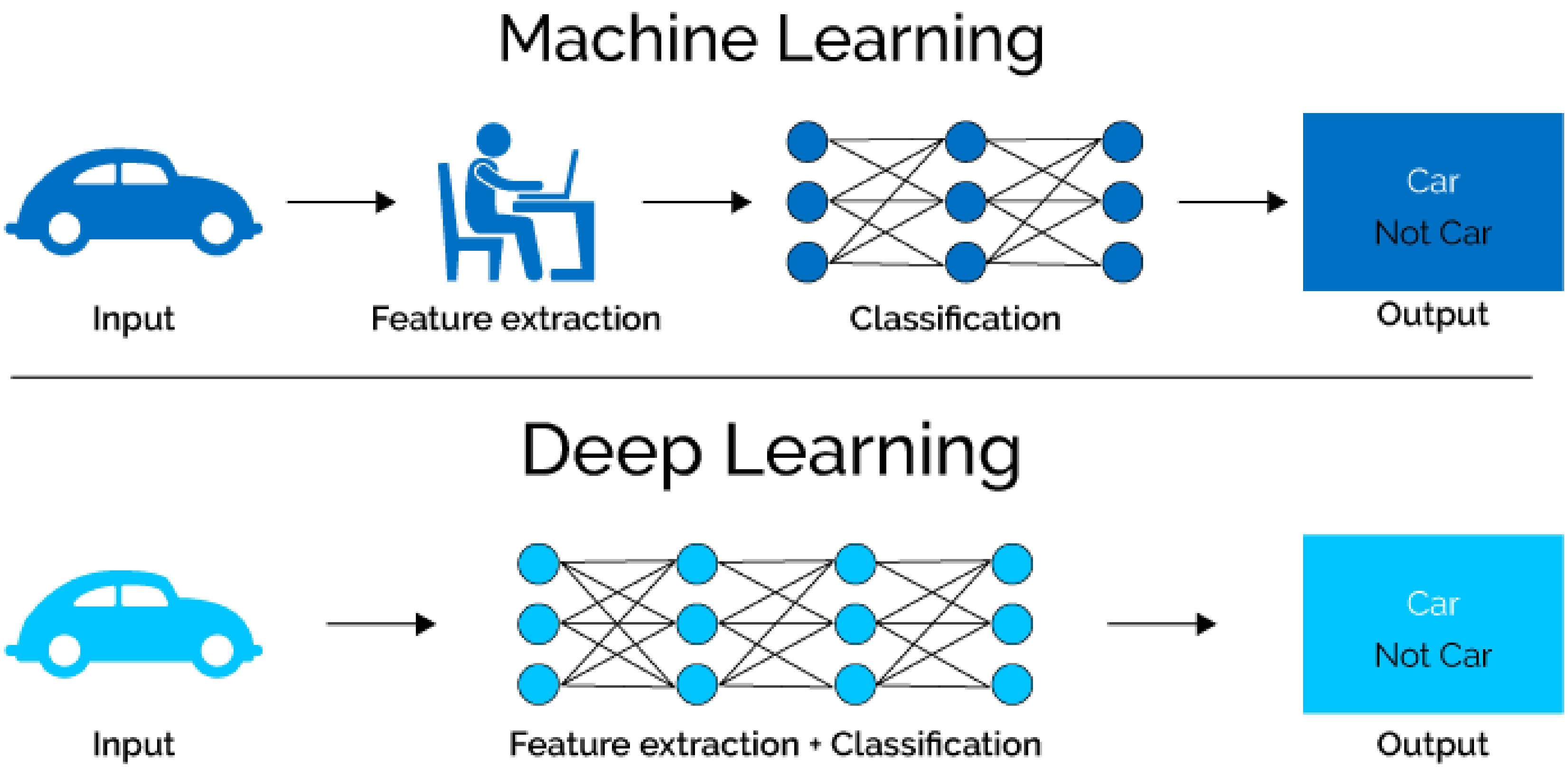
08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	26	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	63	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	78	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	14	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

What Computers See

[11]

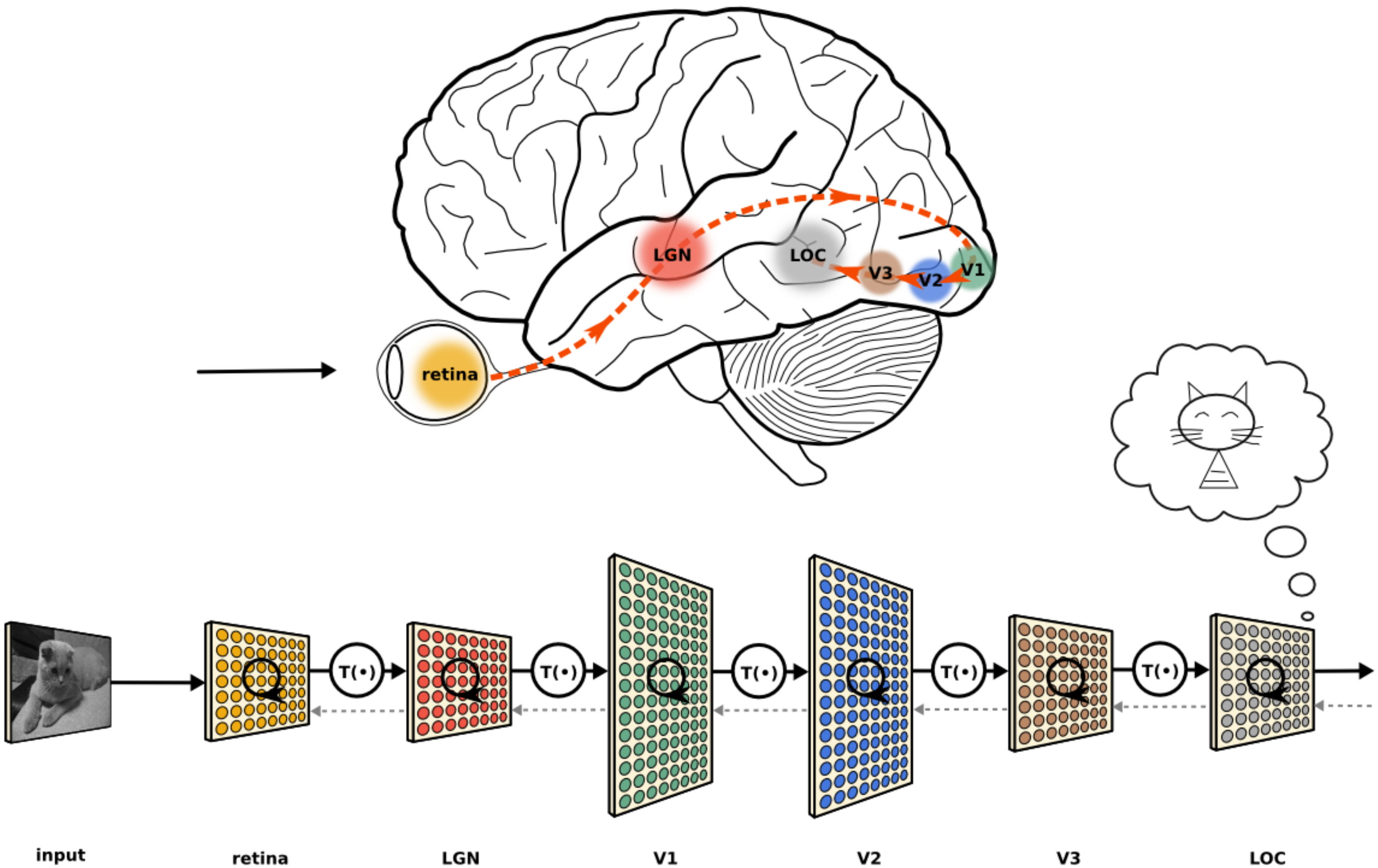
- Skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we need to share with our fellow machines

Image Classification



[12]

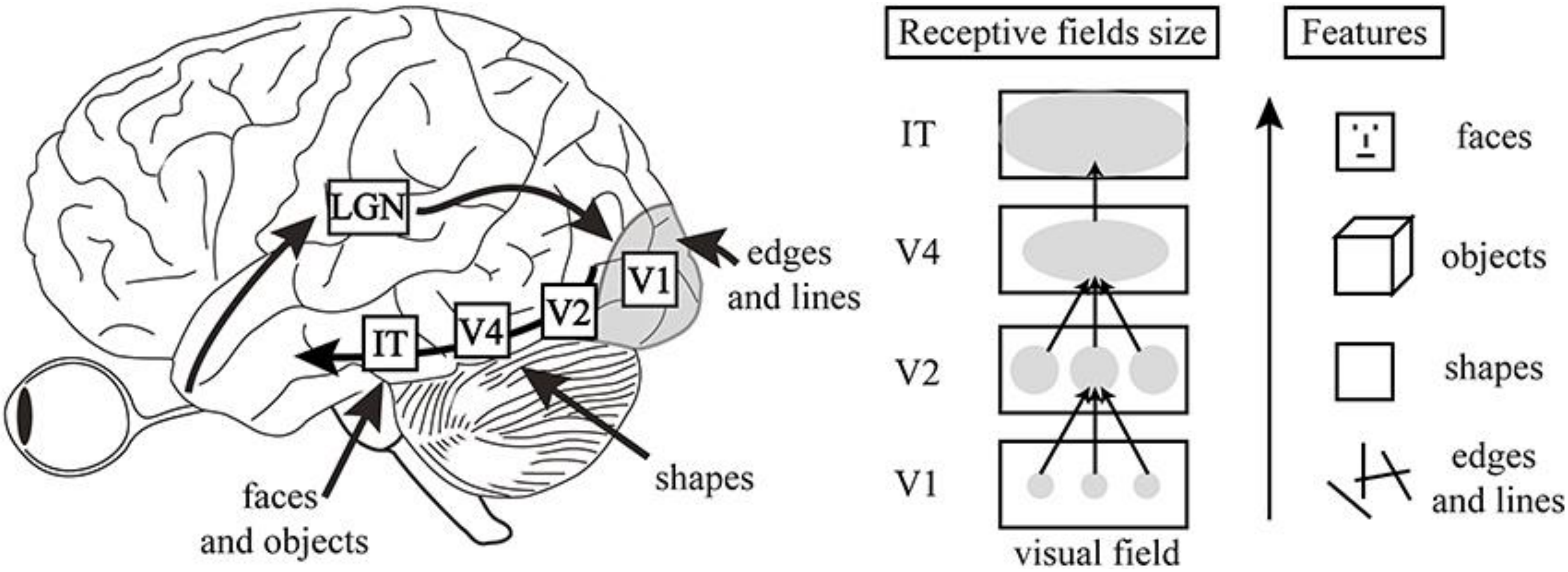
Inheritance from the real world



[13]

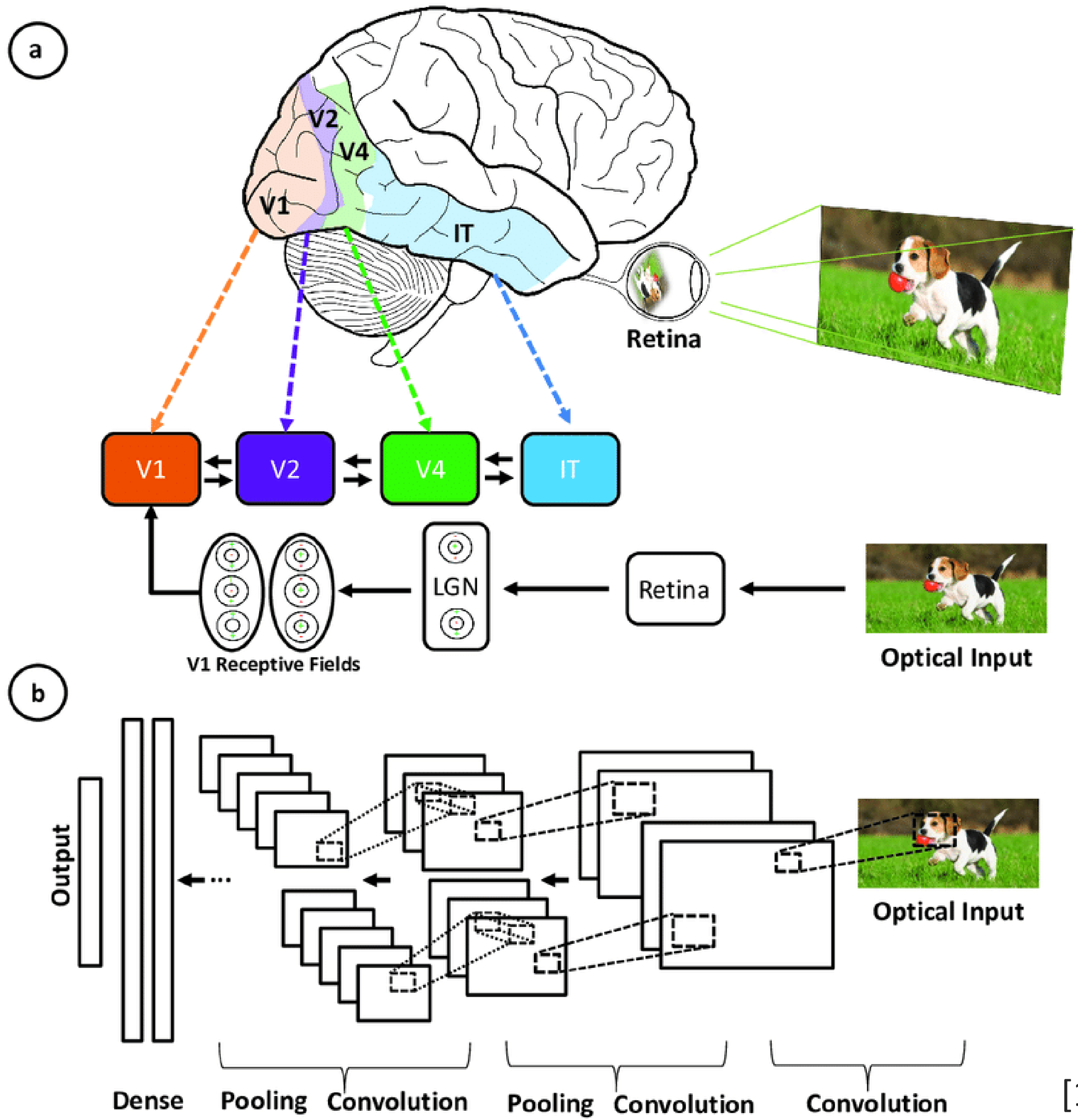
UNIVERSITAS ISLAM INDONESIA

A visual explanation



[14]

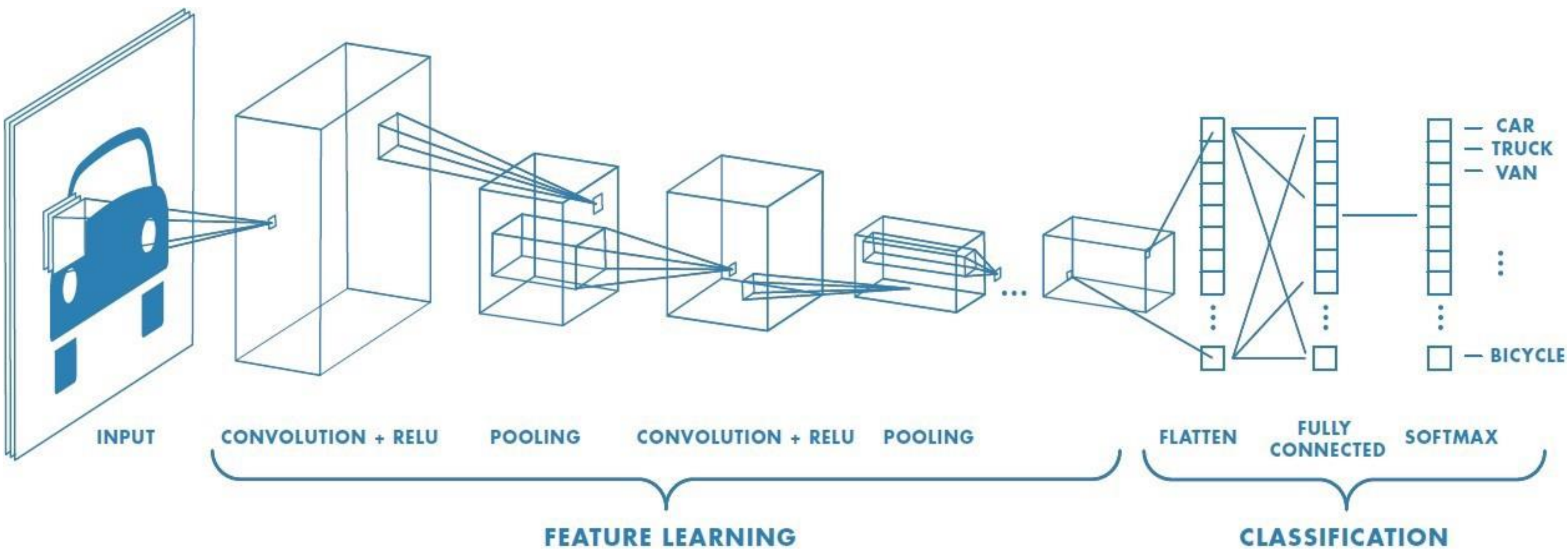
Visual Pathway and CNN



[15]

CNN

1. Layers: Convolutional, pooling (downsampling), and Fully Connected, and output.
2. Activation function: ReLU after Convolution and Fully Connected Layer and SoftMax for the output



[16]

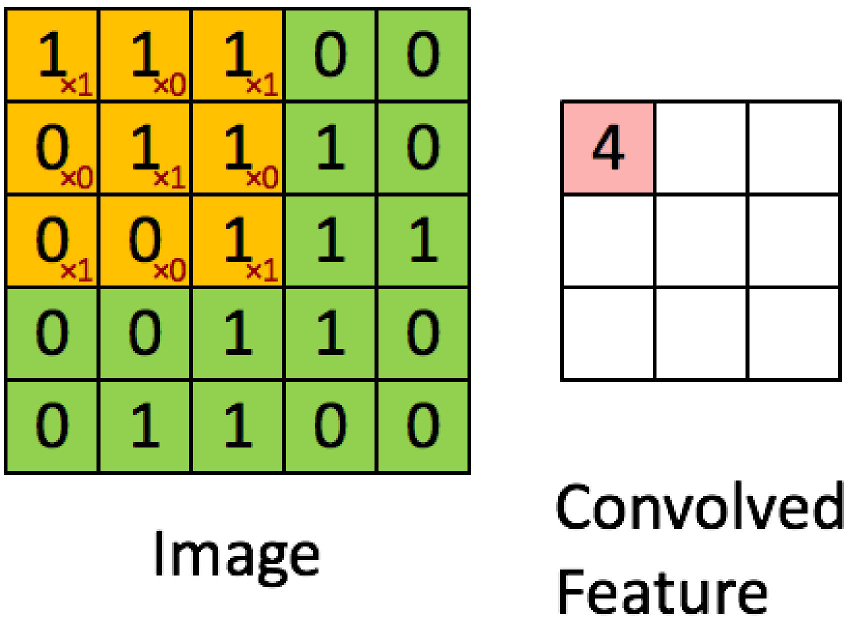
Understanding Convolution

» Convolution in a pixel over One layer (green)

» Filter/kernel (yellow, x1 or x0)

» Receptive field (green part that become yellow)

» Activation Map or Feature Map or Convolved Feature



[16]

Understanding Convolution

» Convolution in a pixel over three layers of color channel (RGB)

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 \quad + \quad -498 \quad + \quad 164 \quad + 1 = -25$$

Bias = 1

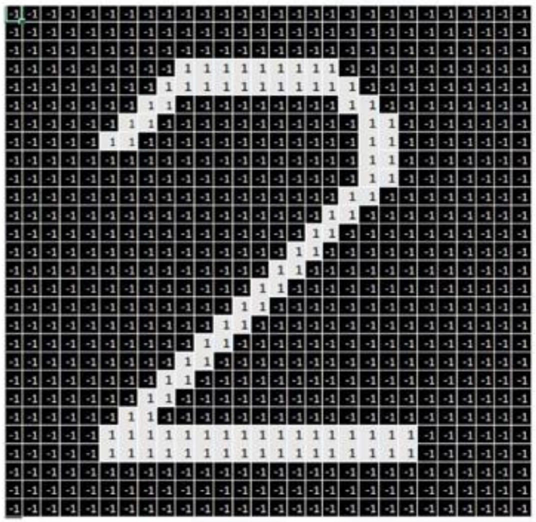
Output

-25				...
				...
				...
				...
...	...	...	...	...

[16]

Convolution Layer

28x28 Matrix

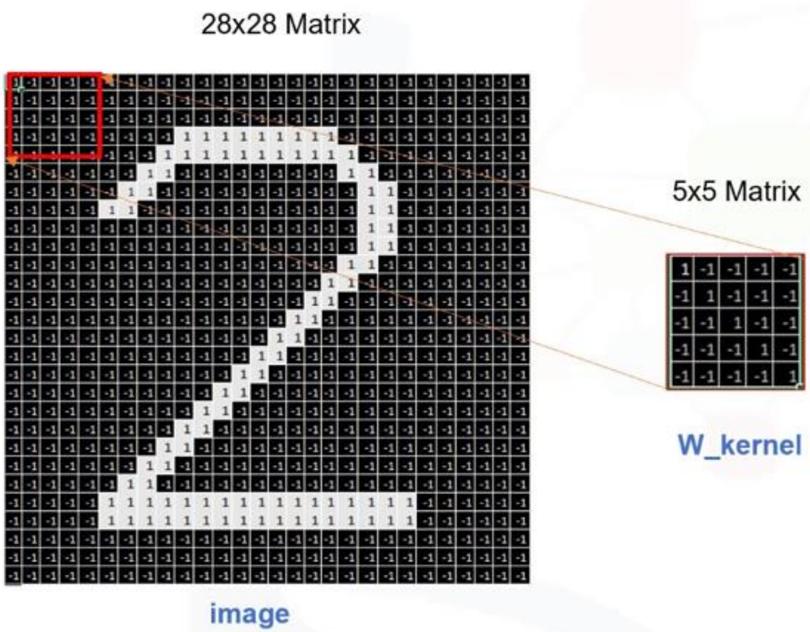


image

tf.nn.conv2d(image, W_kernel)

[17]

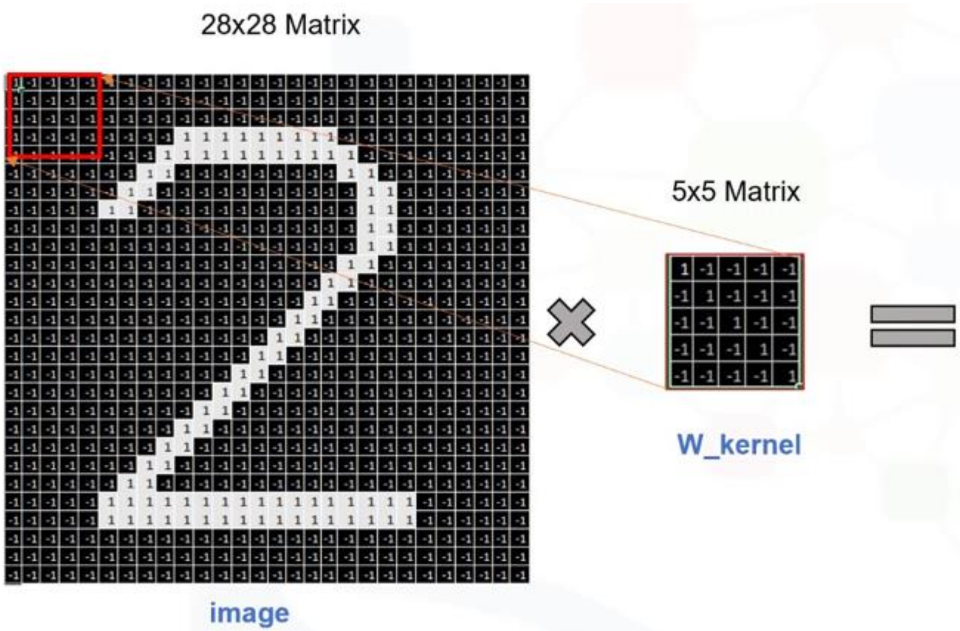
Convolution Layer



tf.nn.conv2d(image, W_kernel)

[17]

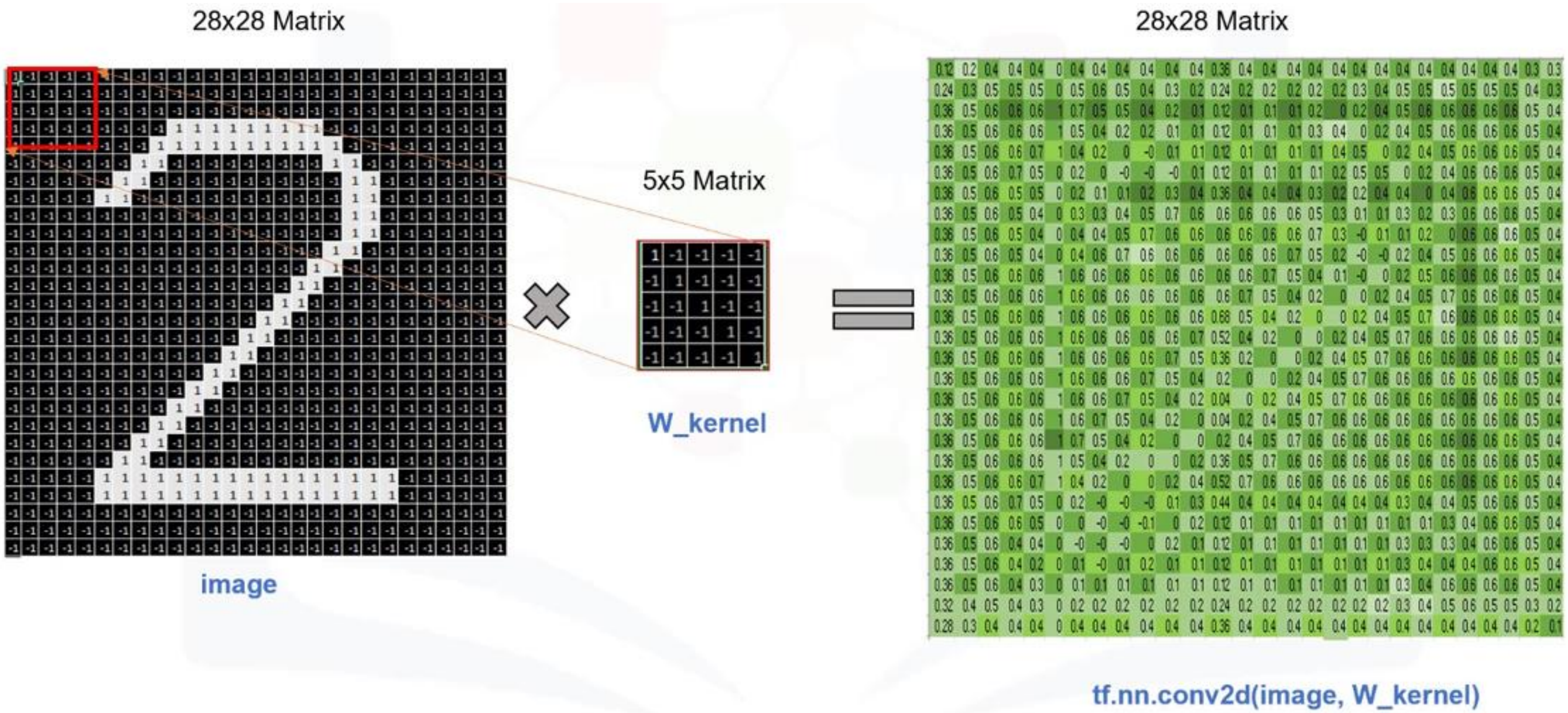
Convolution Layer



tf.nn.conv2d(image, W_kernel)

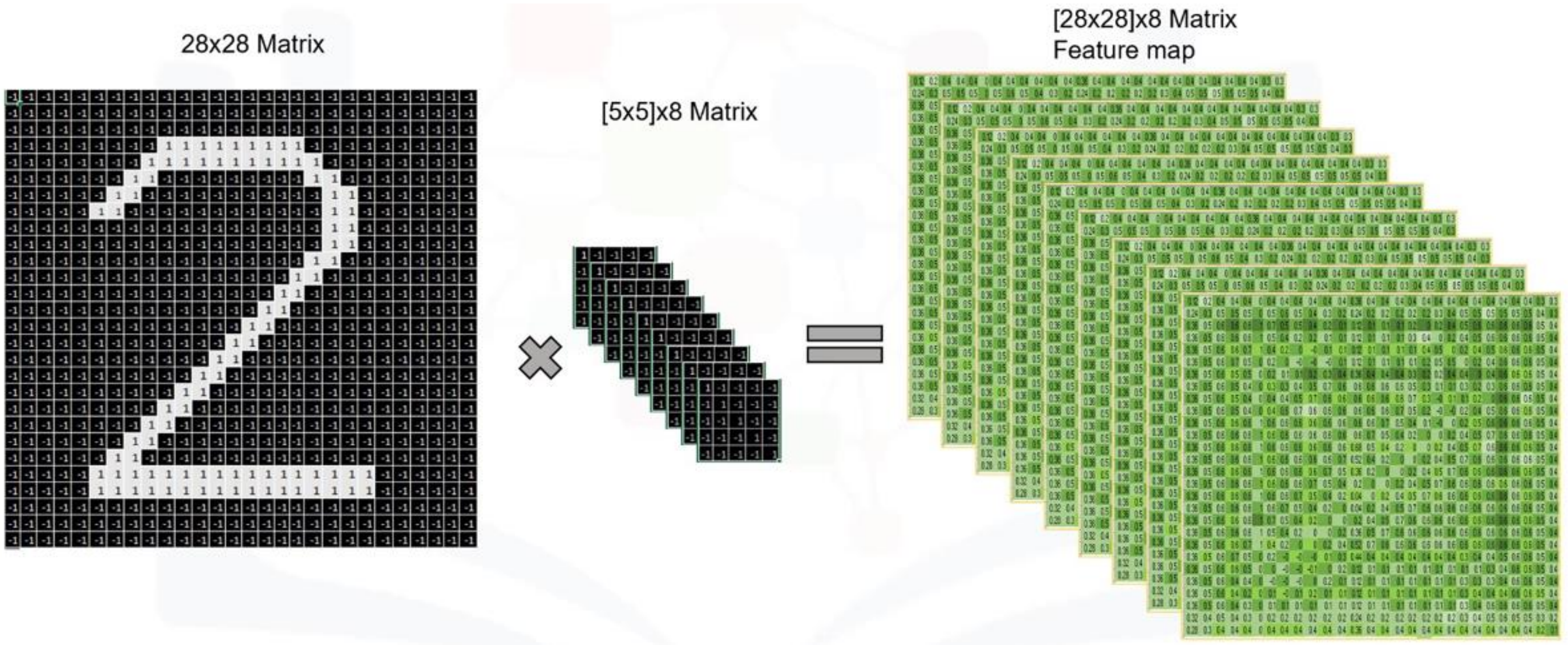
[17]

Convolution Layer



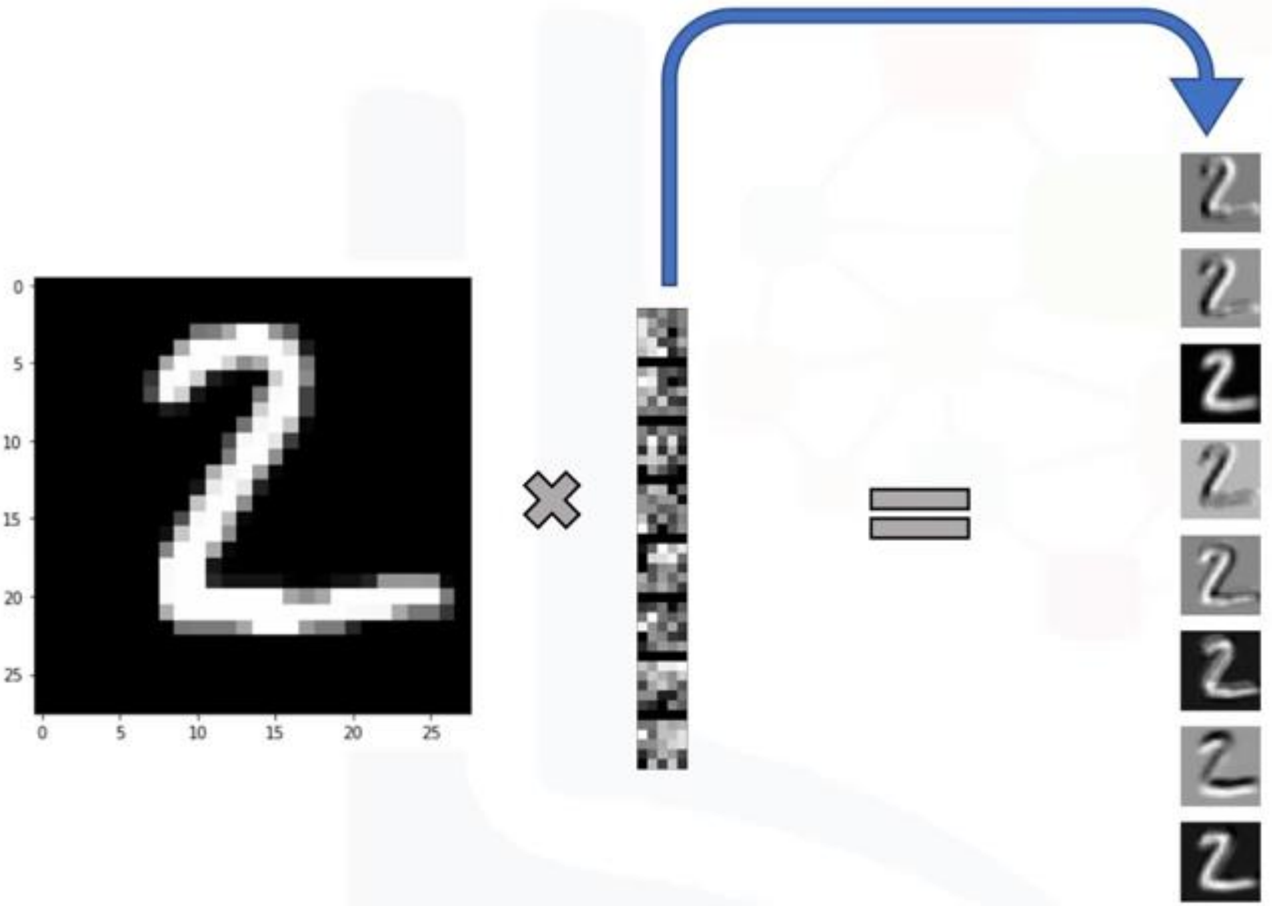
[17]

Convolution Layer



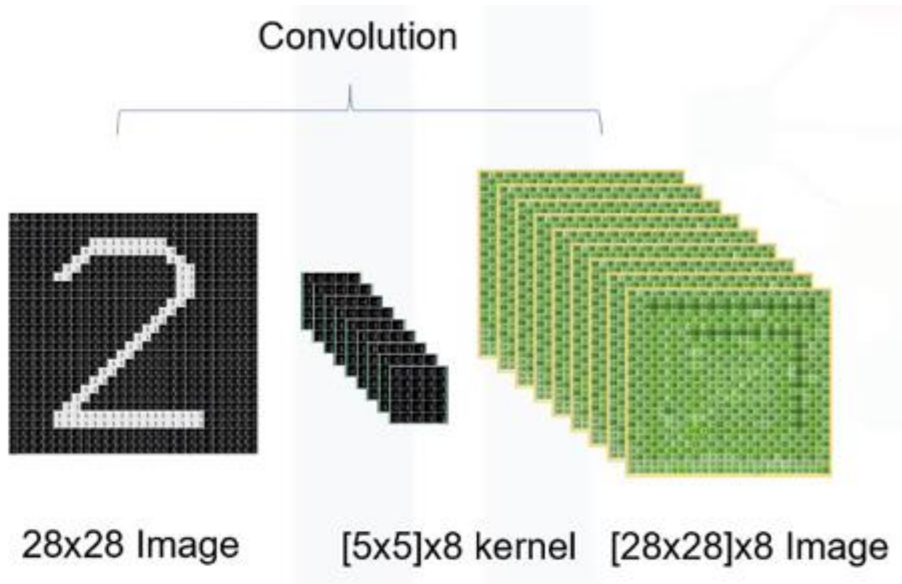
[17]

Convolution Layer



[17]

CNN Architecture



[17]

Convolution Layer

1. The computer is able perform image classification by looking for low level features such as edges and curves, and then building up to more abstract concepts through a series of convolutional layers.
2. Filter in first conv layer are designed to detect low level features such as edges, curves, simple colors.
3. Next Convolution Layer are designed to detect higher-level features

UNIVERSITAS ISLAM INDONESIA

ReLU

1. Rectified Linear Units -- activation layer

$$f(x) = \max(0, x)$$

2. Chosen because a lot faster (because of the computational efficiency) without making a significant difference to the accuracy

3. Provide nonlinearities and preservation of dimension that help to improve the robustness of the network and control overfitting

ReLU

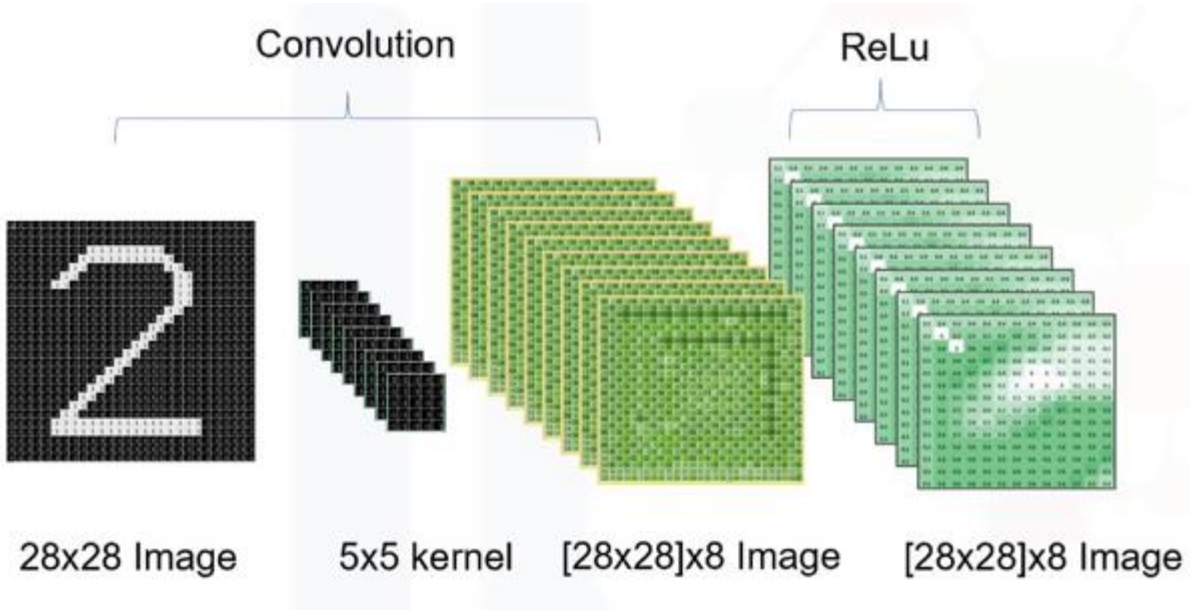
0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	-0.2	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2
0.5	0.6	-0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	-0	0.1	0.1	0.1	0.1
0.5	0.6	0.7	0.5	0.4	0.2	0	-0	-0	-0	0.1	0.1	0.1
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4

convolved1

0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.3	0	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2
0.5	0.6	0	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1
0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1
0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0.1	0.1	0.1	0.1
0.5	0.6	0.7	0.5	0.4	0.2	0	0	0	0	0.1	0.1	0.1
0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4
0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6
0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5
0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4

[17]

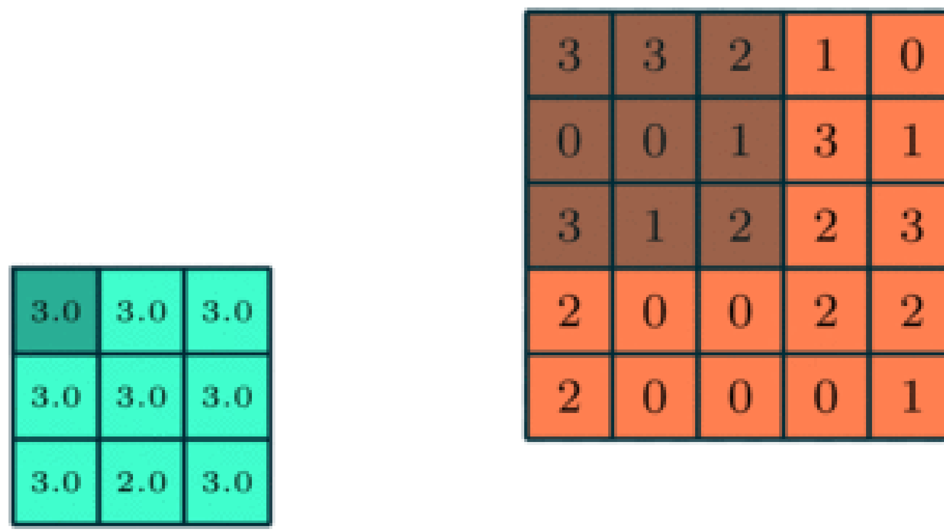
CNN Architecture



[17]

Pooling Layers

1. Also referred to as a downsampling layer
2. Maxpooling being the most popular
3. Maxpooling returns the maximum value (green) from the portion of the image (orange) covered by the Kernel (brown)



[16]

Pooling Layers

0.1	0.2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.2	0.3	0.5	0.5	0.5	0.5	0.5	0.6	0.5	0.4	0.3	0.2	0.2	0.2	0.2
0.4	0.5	0.6	0.6	0.6	0.6	0.7	0.5	0.5	0.4	0.2	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0.2	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.6	0.7	0.5	0.4	0.2	0	0	0.1	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.7	0.5	0.4	0.2	0	0	0	0	0.1	0.1	0.1	0.1
0.4	0.5	0.6	0.5	0.5	0.4	0.2	0.1	0.1	0.2	0.3	0.4	0.4	0.4	0.4
0.4	0.5	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.5	0.7	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.5	0.4	0.4	0.4	0.6	0.7	0.6	0.6	0.6	0.6	0.6	0.6
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0
0.4	0.5	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.7	0.5	0.4	0.2	0	0

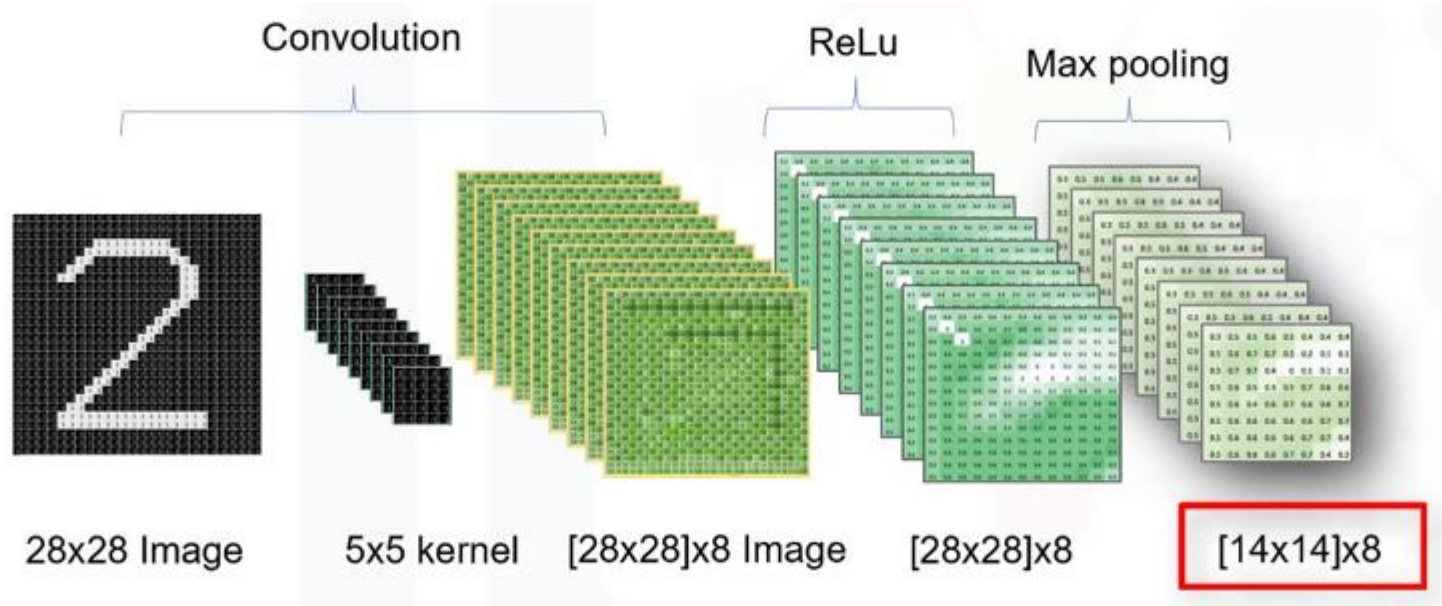
conv1_ReLu

0.3	0.5	0.5	0.6	0.5	0.4	0.4	0.4
0.5	0.6	0.7	0.7	0.5	0.2	0.1	0.1
0.5	0.7	0.7	0.4	0	0.1	0.1	0.1
0.5	0.6	0.5	0.3	0.5	0.7	0.6	0.6
0.5	0.6	0.4	0.6	0.7	0.6	0.6	0.7
0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7
0.5	0.6	0.6	0.6	0.6	0.7	0.7	0.4
0.5	0.6	0.6	0.6	0.7	0.7	0.4	0.2

tf.nn.max_pool(conv1_ReLu, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1])

[17]

CNN Architecture

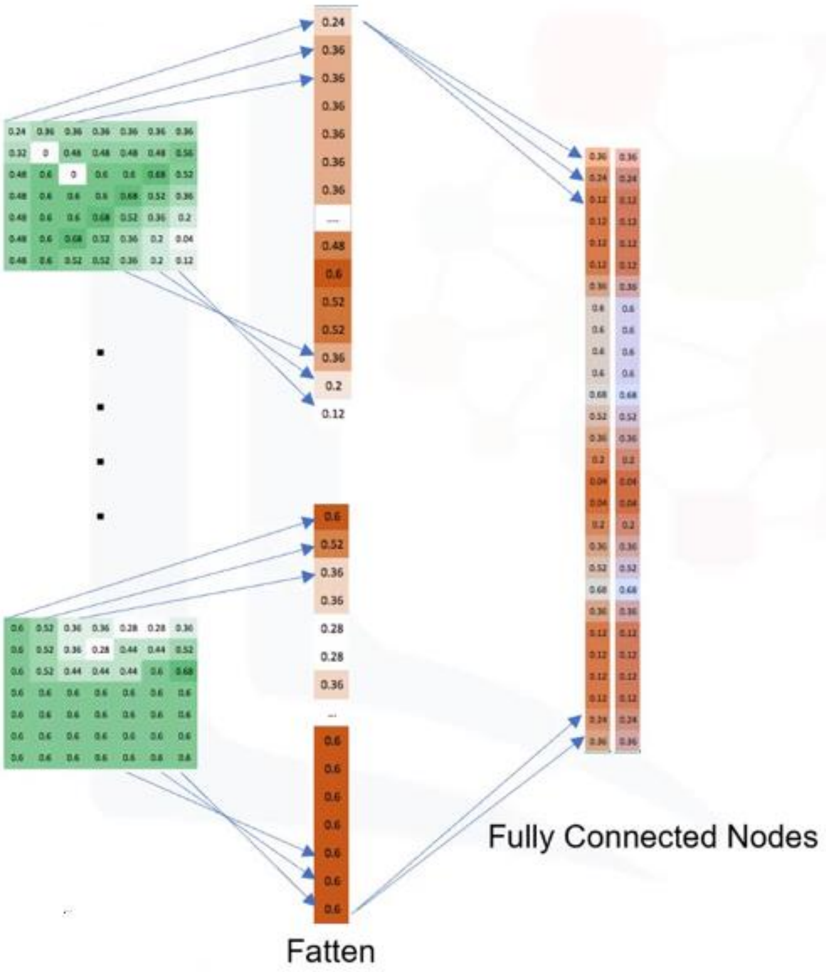


[17]

Fully Connected Layer

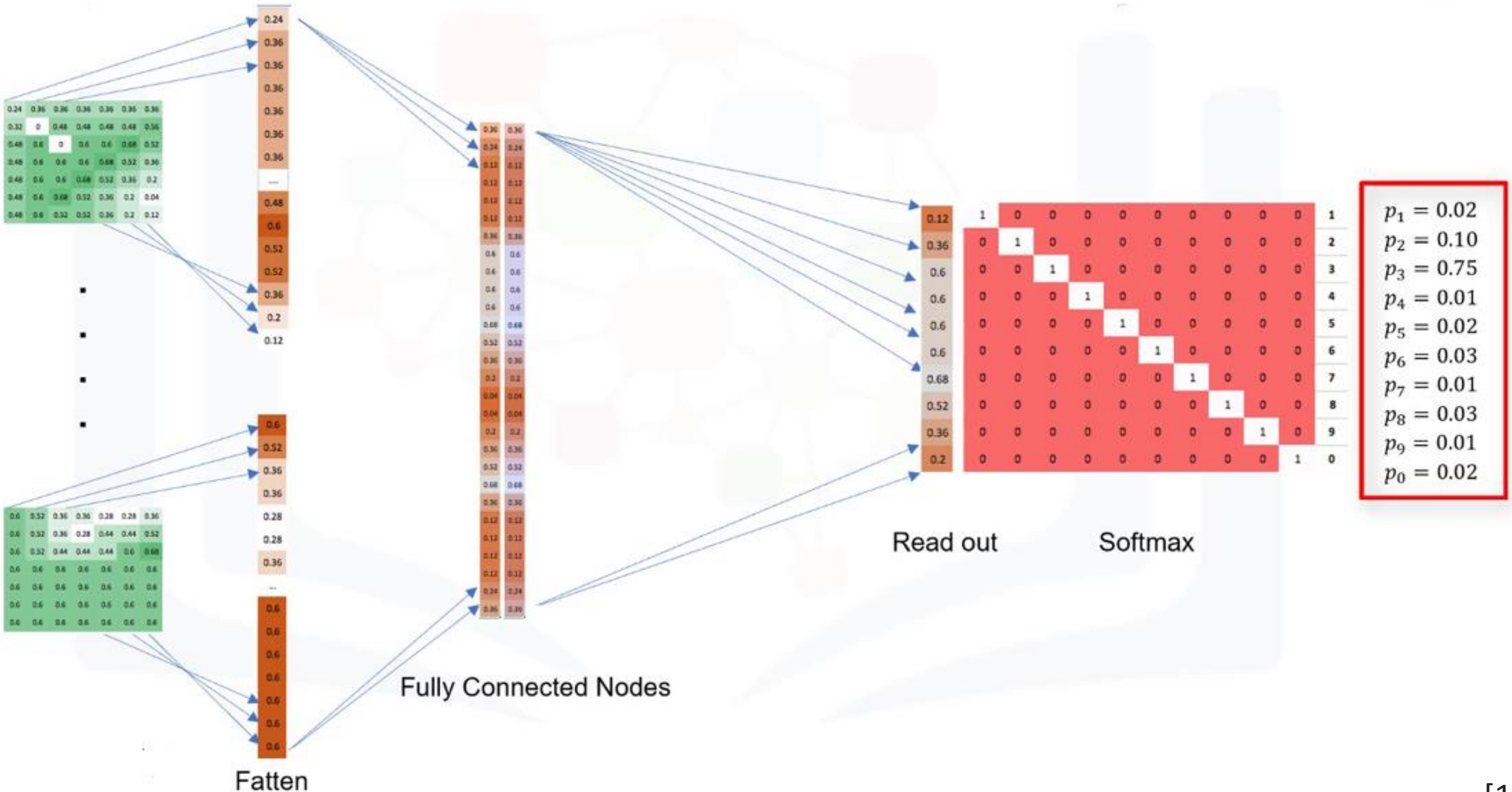
1. Layer in the end of the network.
2. looks at what high level features most strongly correlate to a particular class
3. Input: output is of the convolution-ReLU layer or pool layer preceding it
4. Output: the number of classes

Fully Connected Layer



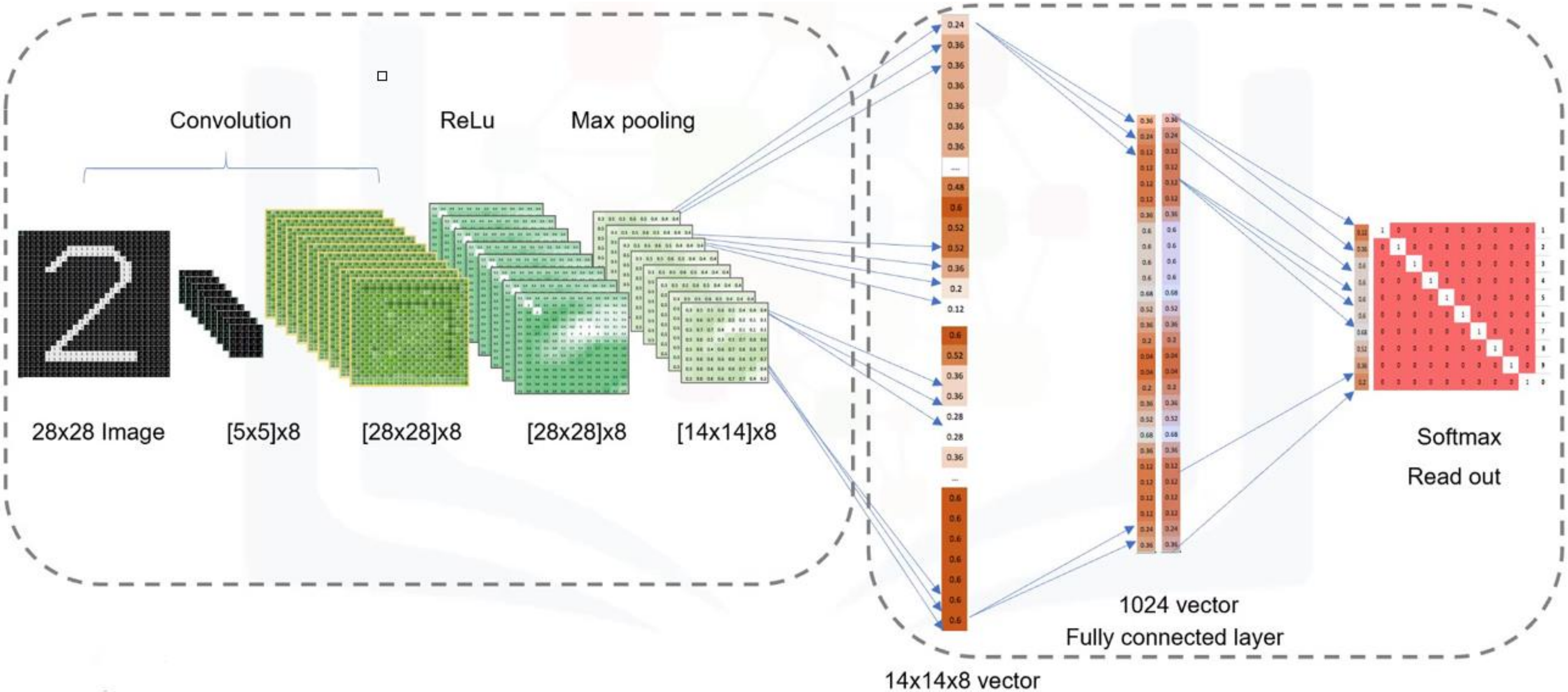
[17]

Fully Connected Layer



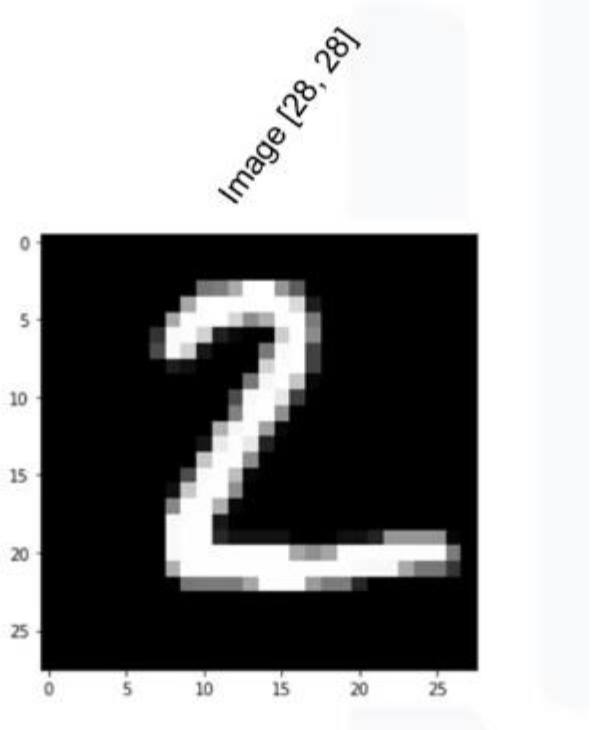
[17]

CNN Architecture



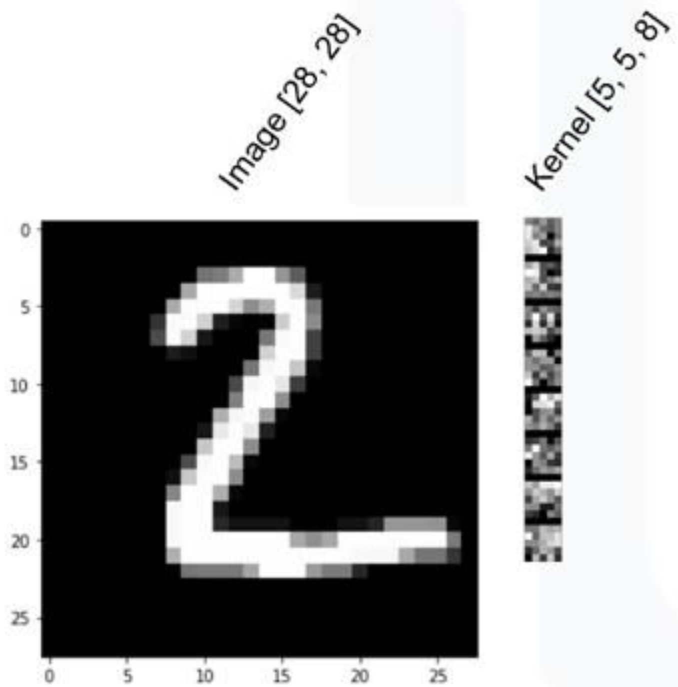
[17]

CNN Architecture



[17]

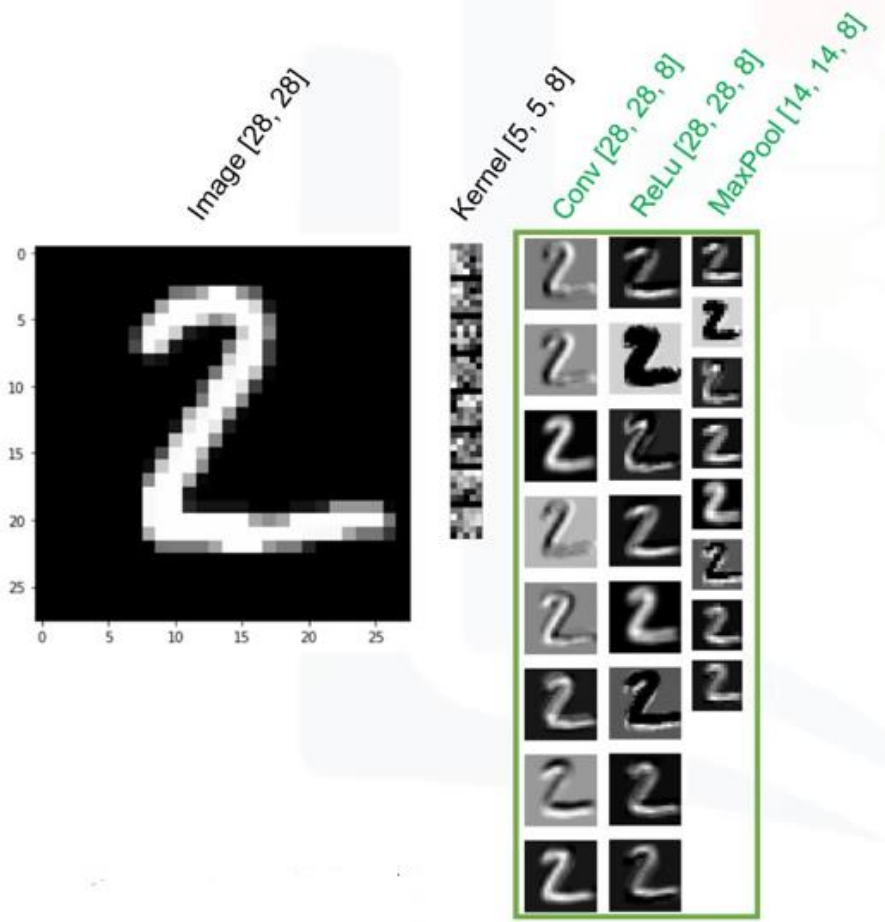
CNN Architecture



[17]

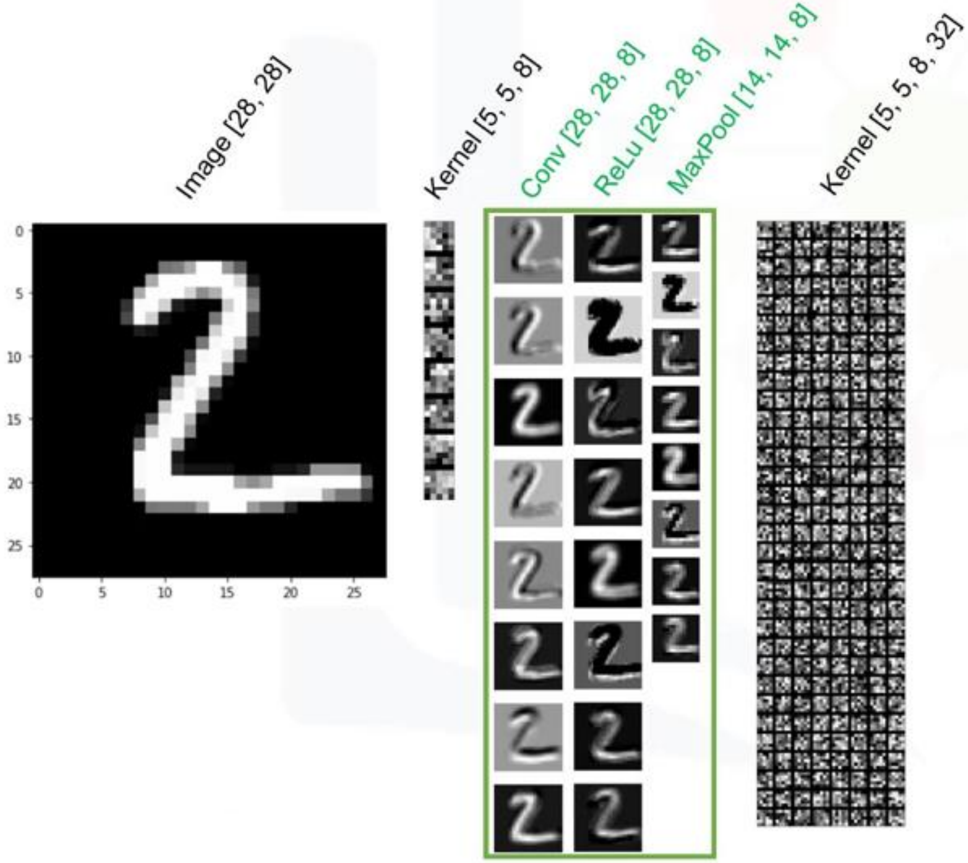
UNIVERSITAS ISLAM INDONESIA

CNN Architecture



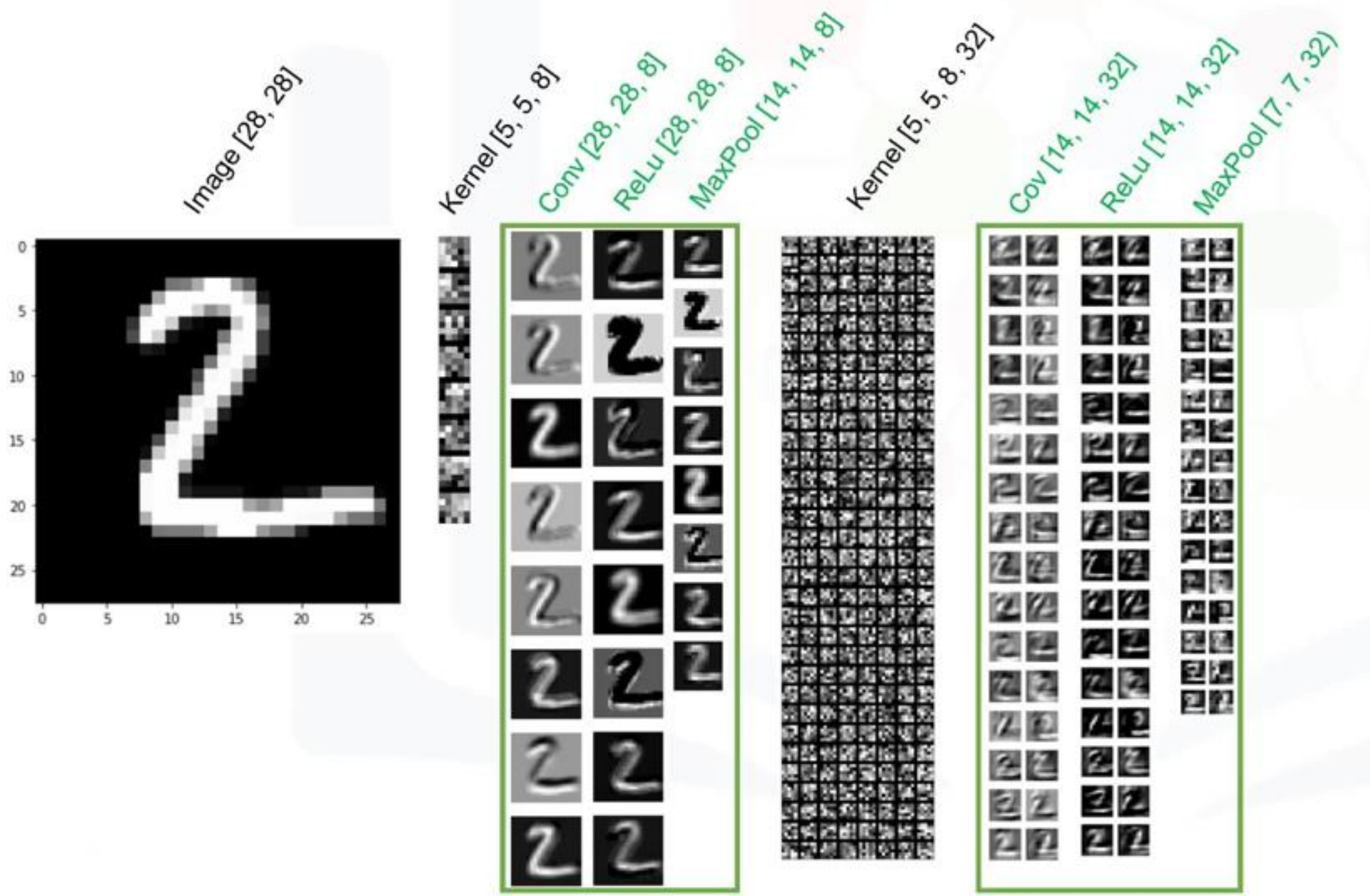
[17]

CNN Architecture



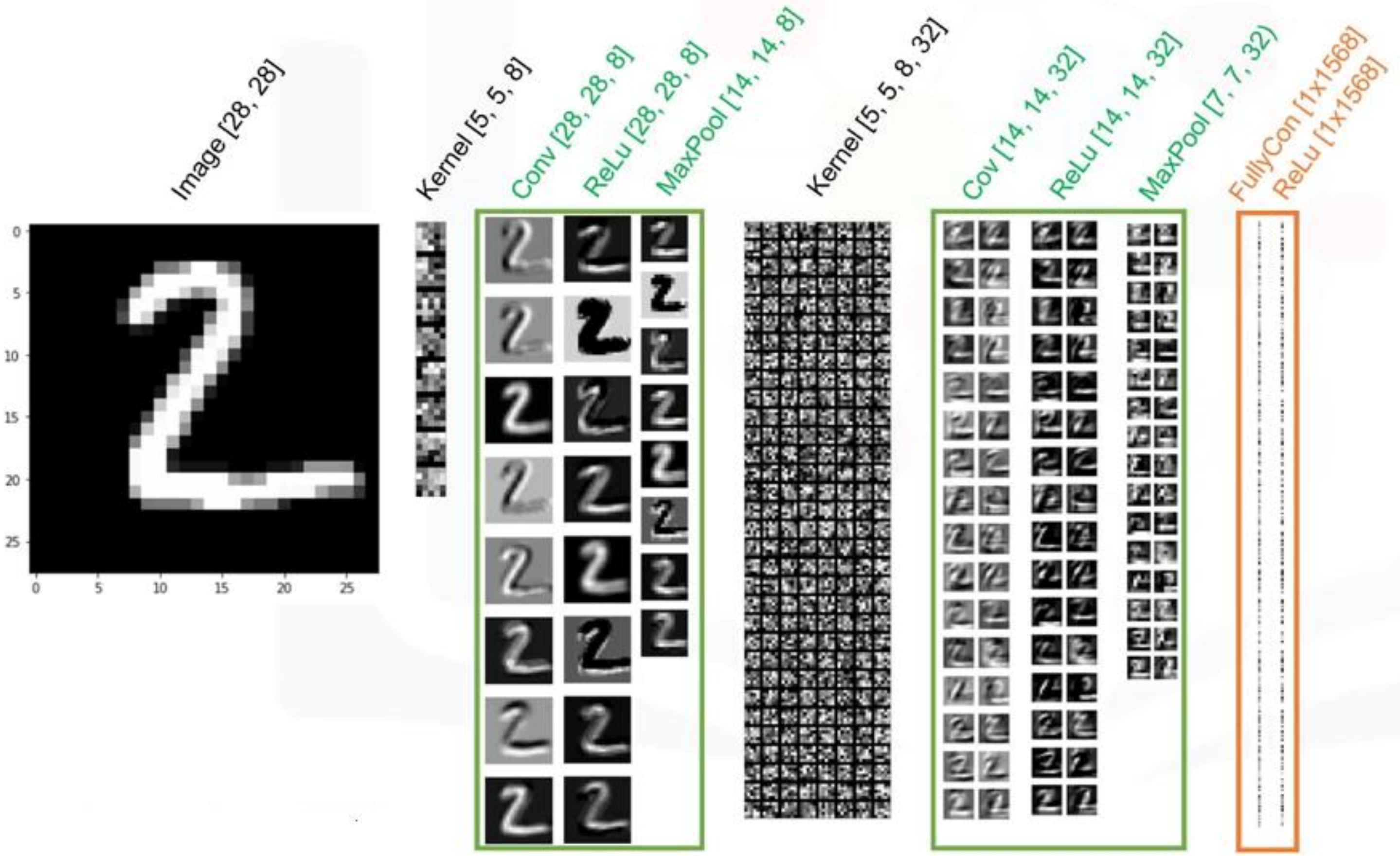
[17]

CNN Architecture



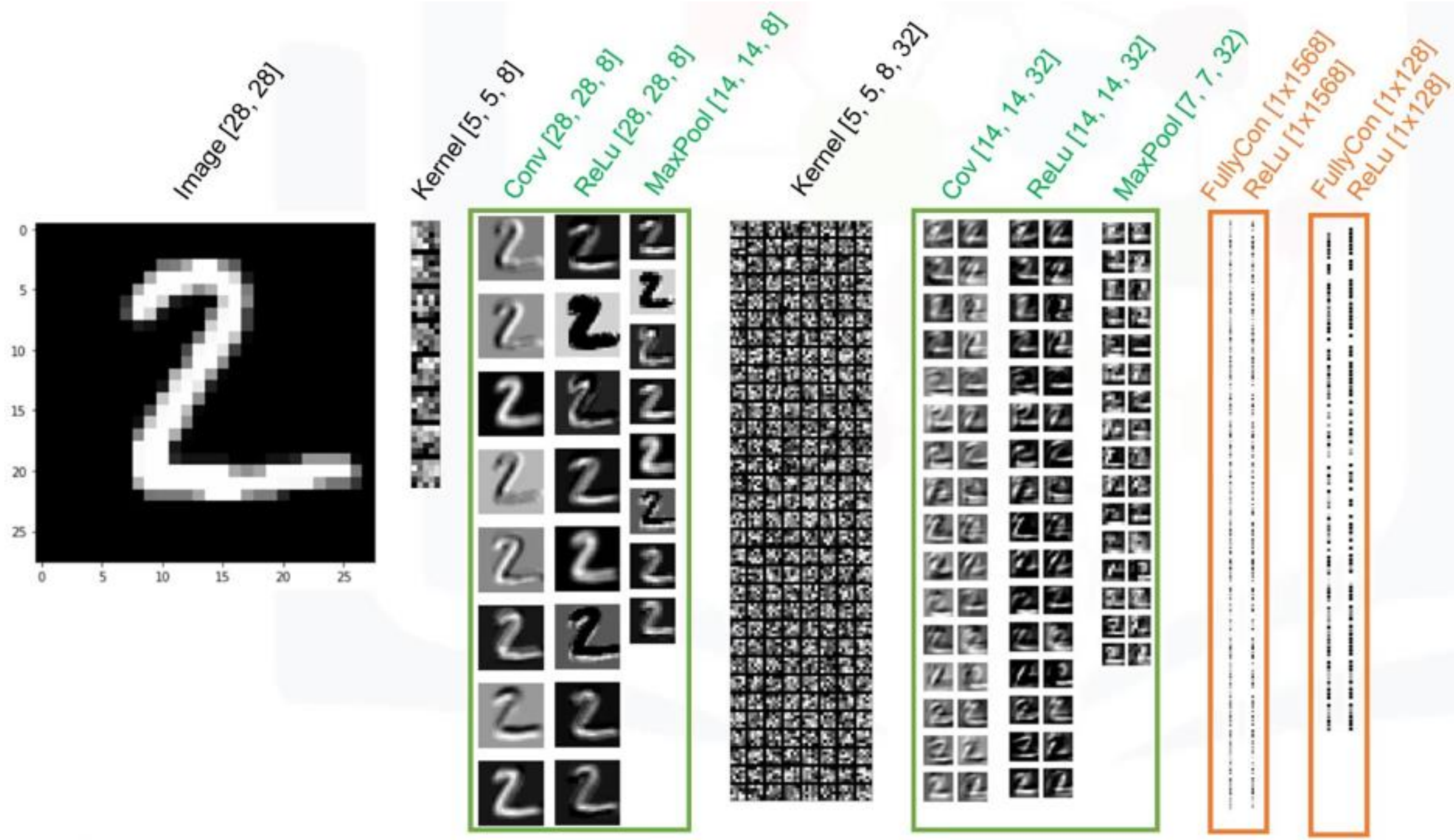
[17]

CNN Architecture



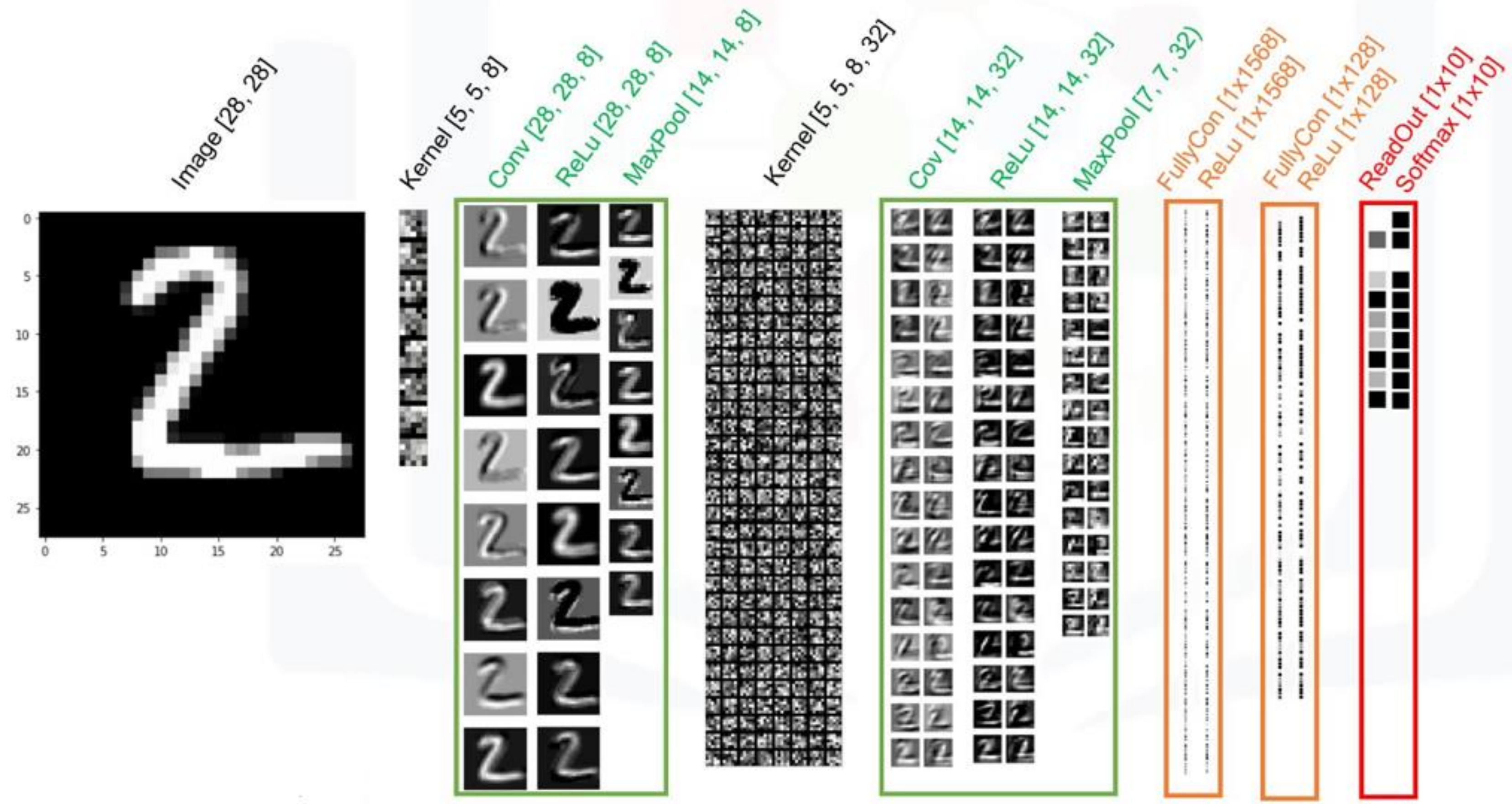
[17]

CNN Architecture



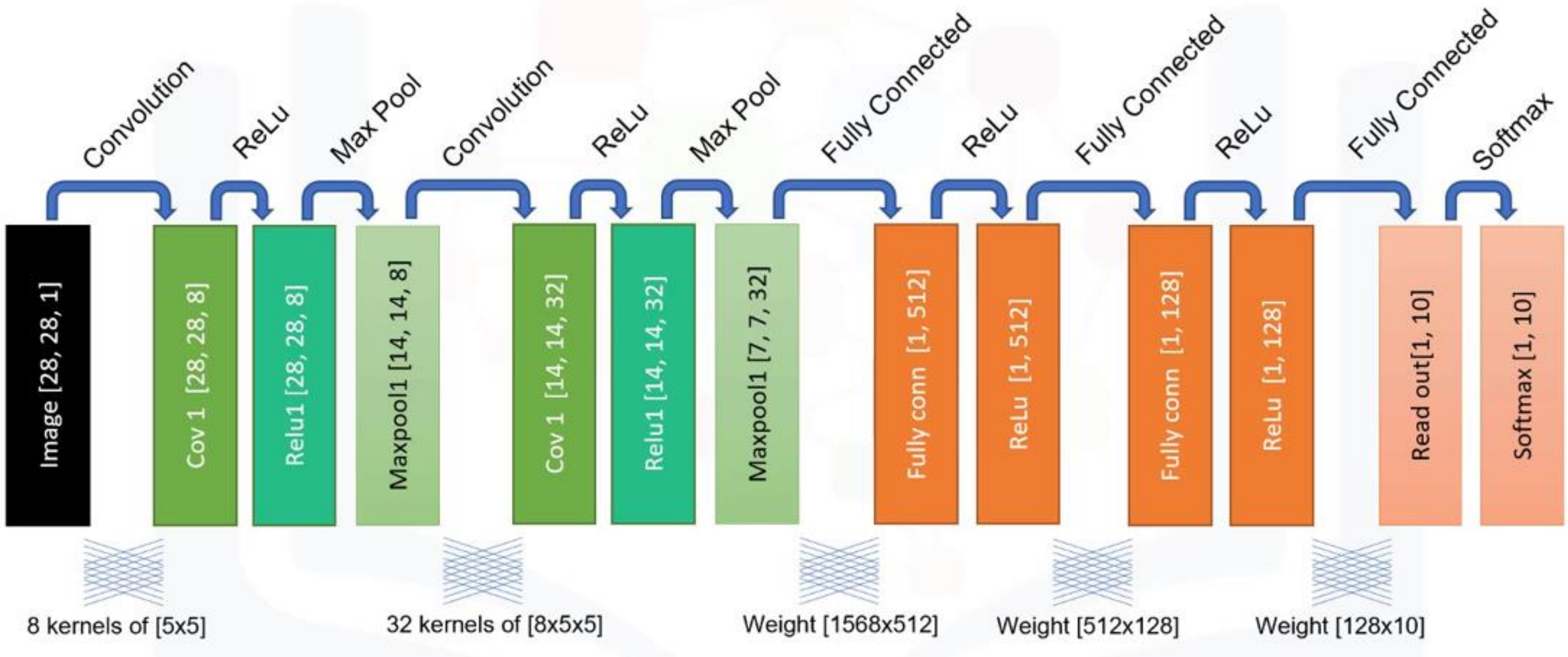
[17]

CNN Architecture



[17]

CNN Training



[17]

UNIVERSITAS ISLAM INDONESIA

Thanks for attention

arrie.kurniawardhani@uii.ac.id

Sumber

1. https://roioverload.com/biology-inspires-the-artificial-neural-network/
2. https://www.ee.co.za/article/application-of-machine-learning-algorithms-in-boiler-plant-root-cause-analysis.html/application-of-machine-learning-algorithms-in-boiler-plant-root-cause-analysis-fig-1
3. https://www.lucidarme.me/simplest-perceptron-update-rules-demonstration/
4. https://www.researchgate.net/figure/Three-different-types-of-transfer-function-step-sigmoid-and-linear-in-unipolar-and_fig8_306323136
5. https://yashuseth.blog/2018/02/11/which-activation-function-to-use-in-neural-networks/
6. https://id.m.wikipedia.org/wiki/Berkas:Neural_network.svg
7. https://code-ai.mk/neural-network-with-c/
8. https://srdas.github.io/DLBook/GradientDescentTechniques.html
9. https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351

Sumber

10. https://www.reddit.com/r/artificial/comments/d4pm74/a_brief_history_of_ai_from_1940s_till_today_image/
11. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
12. https://morioh.com/p/ed56b4fdbf1c
13. https://www.jackreeceejini.com/2019/07/animal-intelligence.html
14. https://www.frontiersin.org/articles/10.3389/fncom.2014.00135/full
15. https://www.researchgate.net/figure/2-Illustration-of-the-corrispondence-between-the-areas-associated-with-the-primary_fig3_317277576
16. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
17. https://cognitiveclass.ai/