CALVIN
INSTITUTE OF TECHNOLOGY
God's People for God's Glory

Neural Networks

Hendrik Santoso Sugiarto

IBDA2032 – *Artificial Intelligence*

Capaian Pembelajaran

- Perceptron
- Multilayer Perceptron
- Deep Learning

Neural Networks

- Paradigma machine learning yang menggunakan perspektif neuroscience
- Dapat digunakan untuk regresi, klasifikasi, clustering, representasi
- Sulit diinterpretasi dan proses pelatihan sangat berat
- Tidak dapat dijadikan probabilistik
- Performa sangat baik, terutama untuk data skala besar

CALVIN INSTITUTE OF TECHNOLOGY

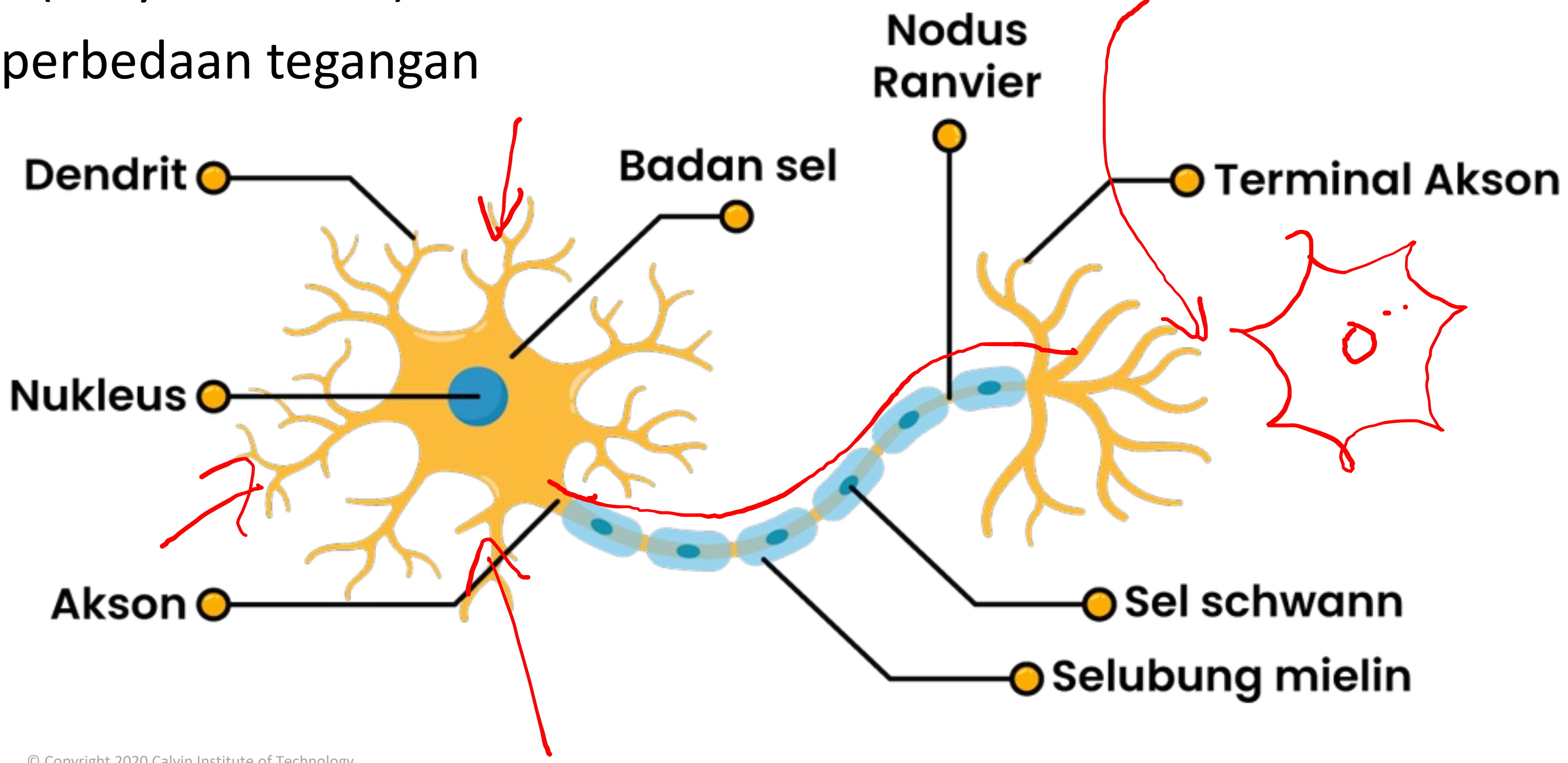
Perceptron

Sel Syaraf

Komunikasi antar sel syaraf melalui sinyal elektrik dan kimiawi

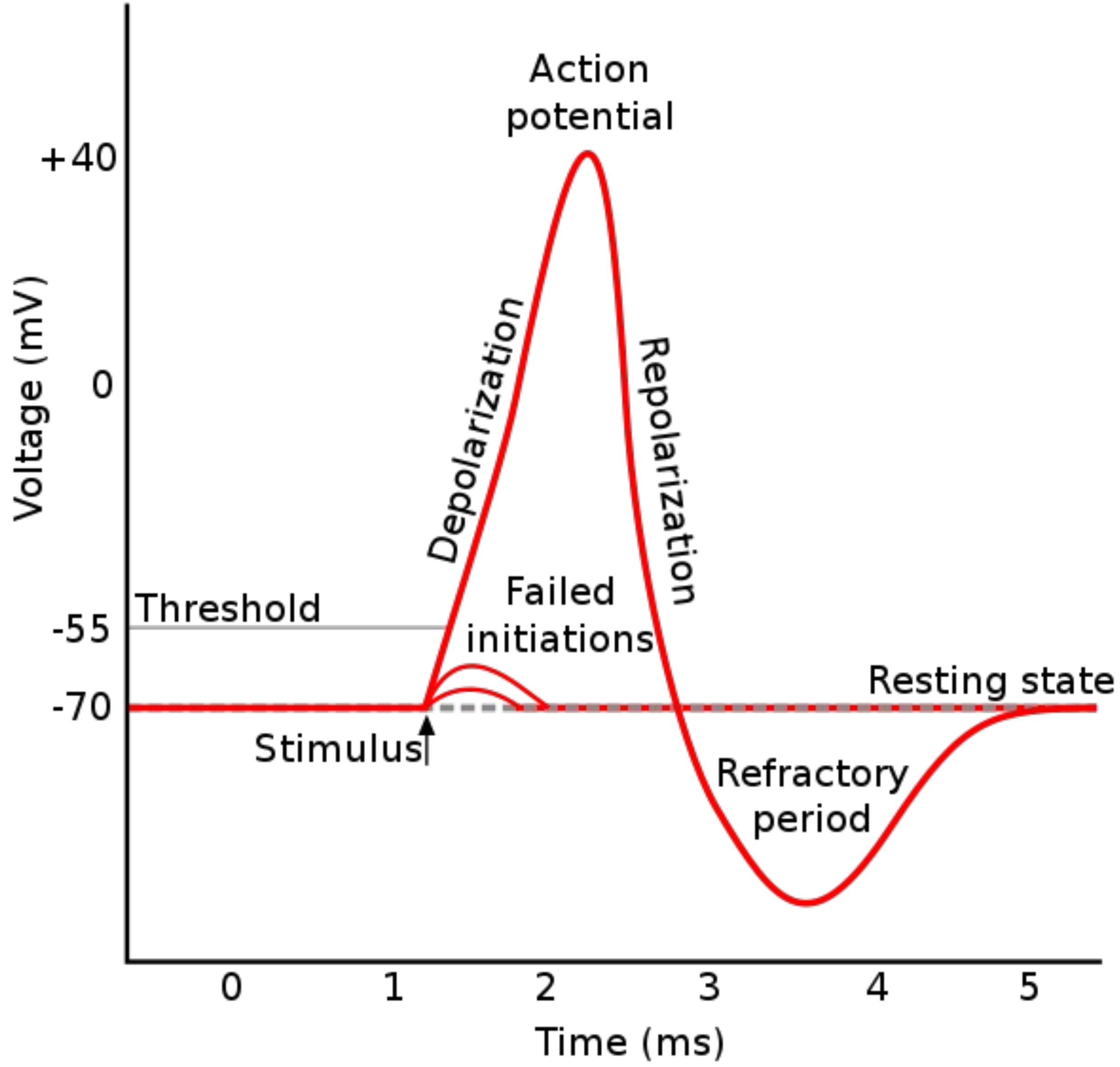
neurotransmitter
dopamin
serotonin
oxytocin

- Input: sinyal elektrik
- Bobot: neurotransmitter (senyawa kimia)
- Aktivasi: batas ambang perbedaan tegangan
- Output: sinyal elektrik

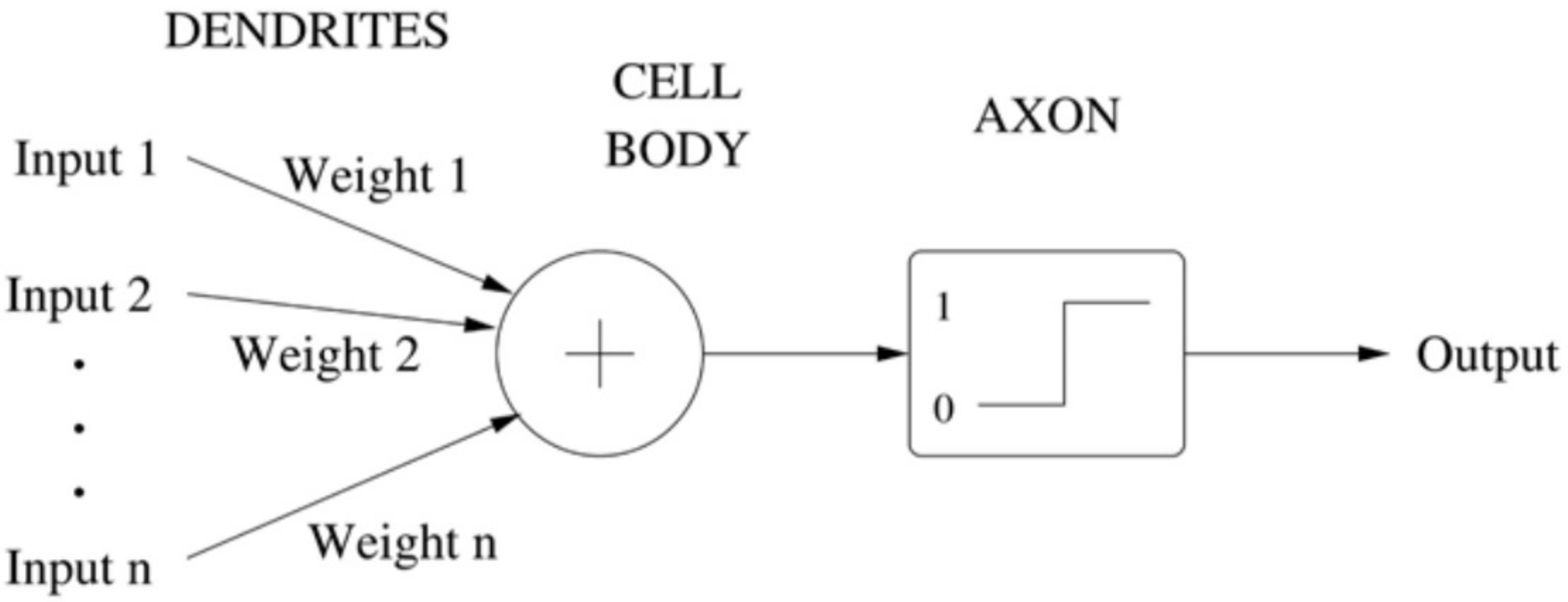


Aktivasi Sel Syaraf

- Sel syaraf akan teraktivasi jika akumulasi stimulus sinyal melewati ambang batas



Perceptron (1958)



- Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.

CALVIN INSTITUTE OF TECHNOLOGY

Perceptron

$\hat{y} = (\theta^T x + \theta_0)$

$\hat{y} = \sigma(\theta^T x + \theta_0)$

- Akan menjadi persamaan linear ($y = mx + c$) jika fungsi aktivasinya linear

Output

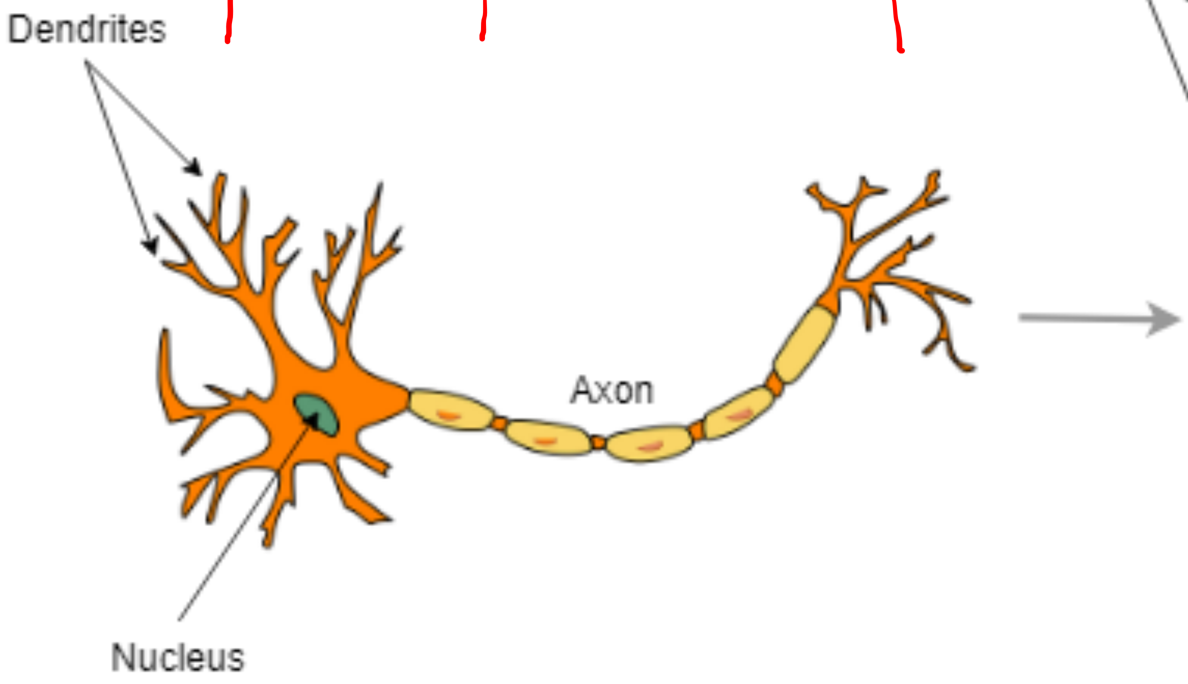
Linear combination of inputs

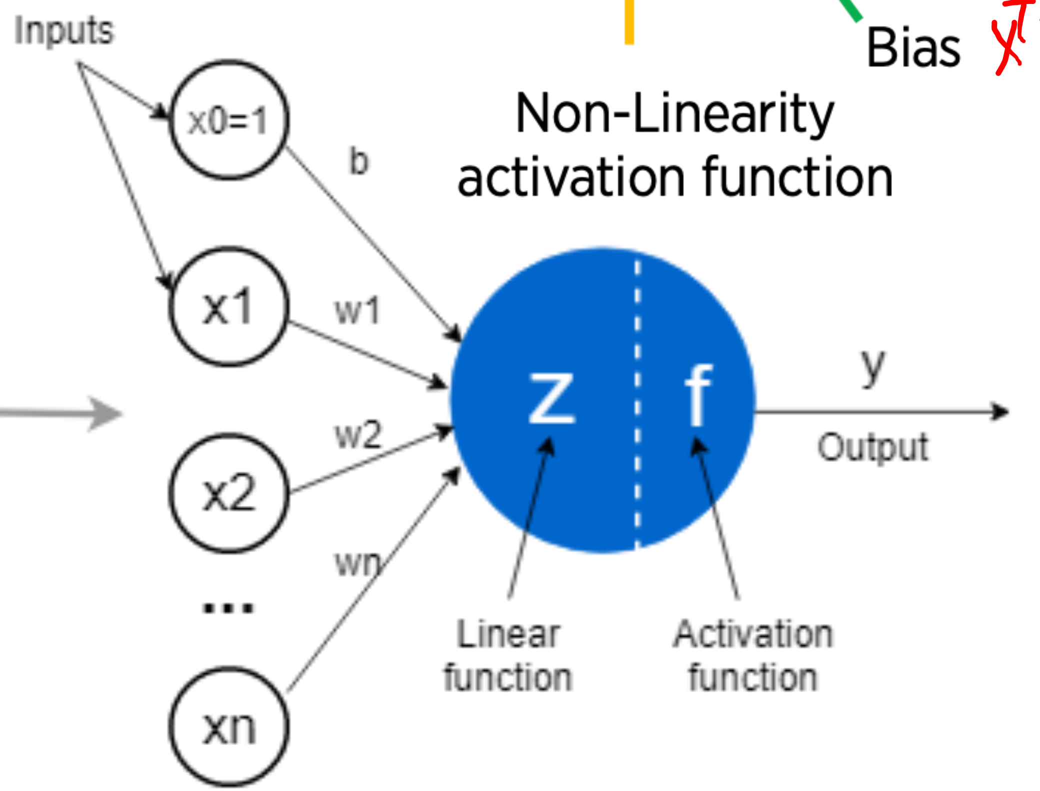
$$\hat{y} = g\left(w_0 + \sum_{i=0}^{m} x_i\, w_i\right)$$

Bias $X^T \vec{W}$

Non-Linearity activation function

$W_0 + W_1 X_1 + W_2 X_2 + \dots + W_n X_n$





Uji Pemahaman

$x_1\, x_2\, x_3\, x_4\, x_5$

- Berapakah output dari sebuah neuron dengan 5 input (1,2,3,2,1), dengan bobot (3,2,1,0,-1), dan fungsi aktivasi linear?

$w_1\, w_2\, w_3\, w_4\, w_5$

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4 + x_5 w_5$$

$$1 \cdot 3 + 2 \cdot 2 + 3 \cdot 1 + 2 \cdot 0 + 1 \cdot -1$$

$$a = 3 + 4 + 3 + 0 - 1 = 9$$

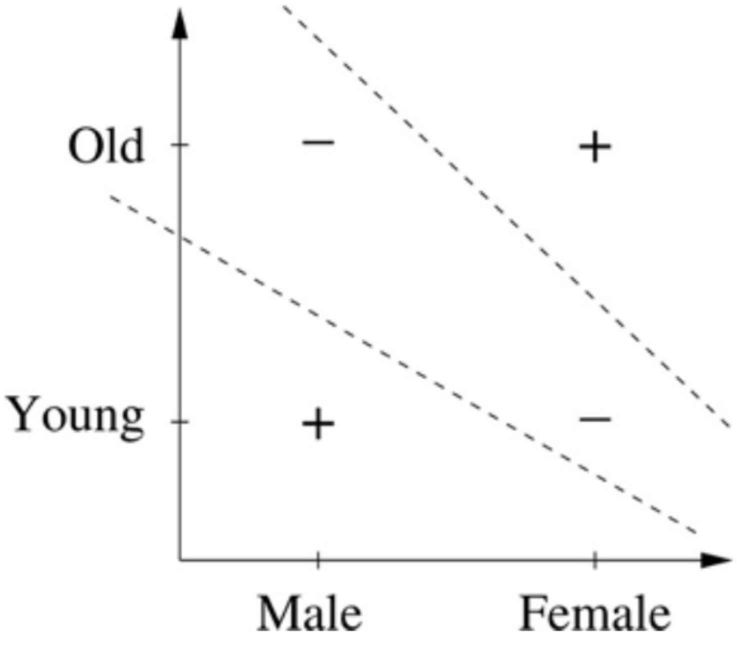
$$y = f(a) = a = 9$$

Algoritma Perceptron

- Inisiasi bobot $\boldsymbol{w}$ secara acak
- Untuk setiap data $\boldsymbol{x}$ dengan label y:
  - Prediksi dengan bobot sekarang:
    - $y' = \operatorname{sgn}(\boldsymbol{w} \cdot \boldsymbol{x})$
  - Jika prediksi salah, perbaharui bobot w:
    - $\boldsymbol{w} = \boldsymbol{w} + a(y - y')\boldsymbol{x}$
    - Jika $y = 1$ dan $y' = -1$, bobot akan diperbesar
    - Jika $y = -1$ dan $y' = 1$, bobot akan dikurangi
- Ulangi sampai konvergen
- Konvergen:
  - Data terpisahkan secara linear: konvergen
  - Data tidak terpisahkan: konvergen menuju solusi minimum terdekat

Kritik terhadap Perceptron

- Misky and Papert menulis sebuah buku berjudul *Perceptrons*
- Berisi semua contoh kelemahan Perceptron
- Salah satu yang paling terkenal: XOR

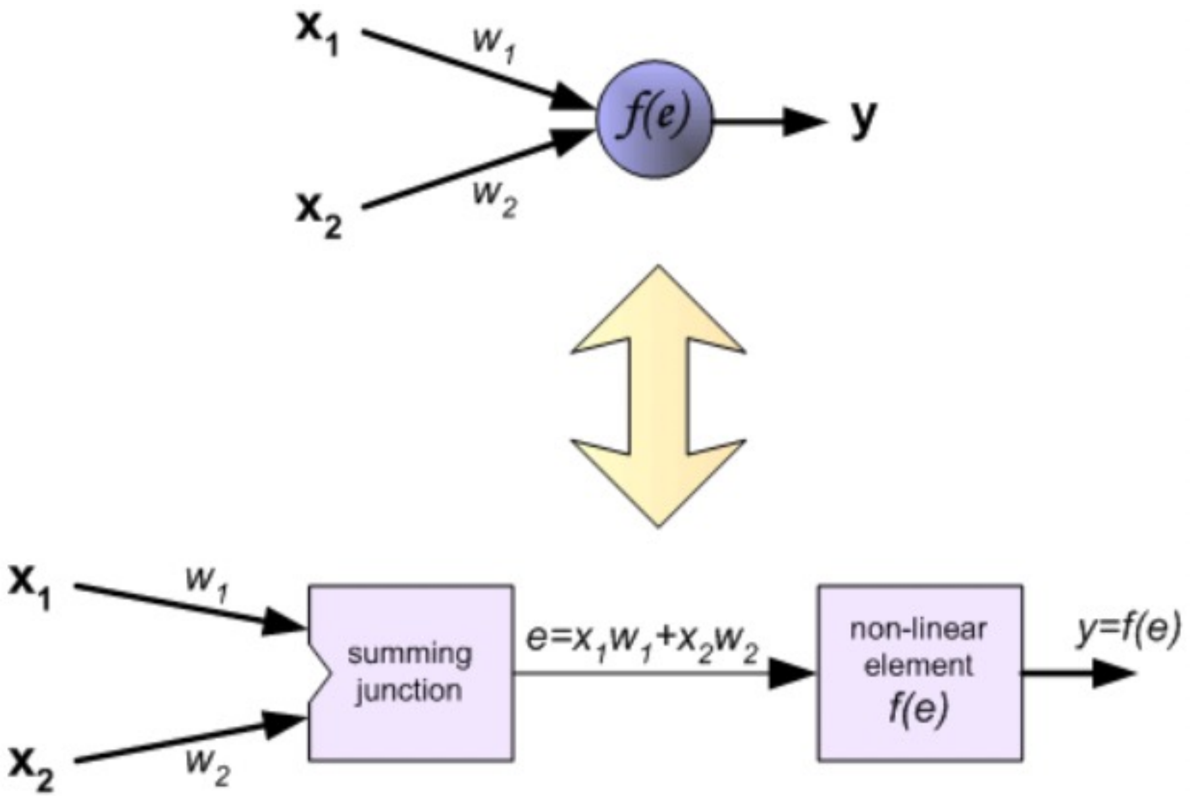


MLP

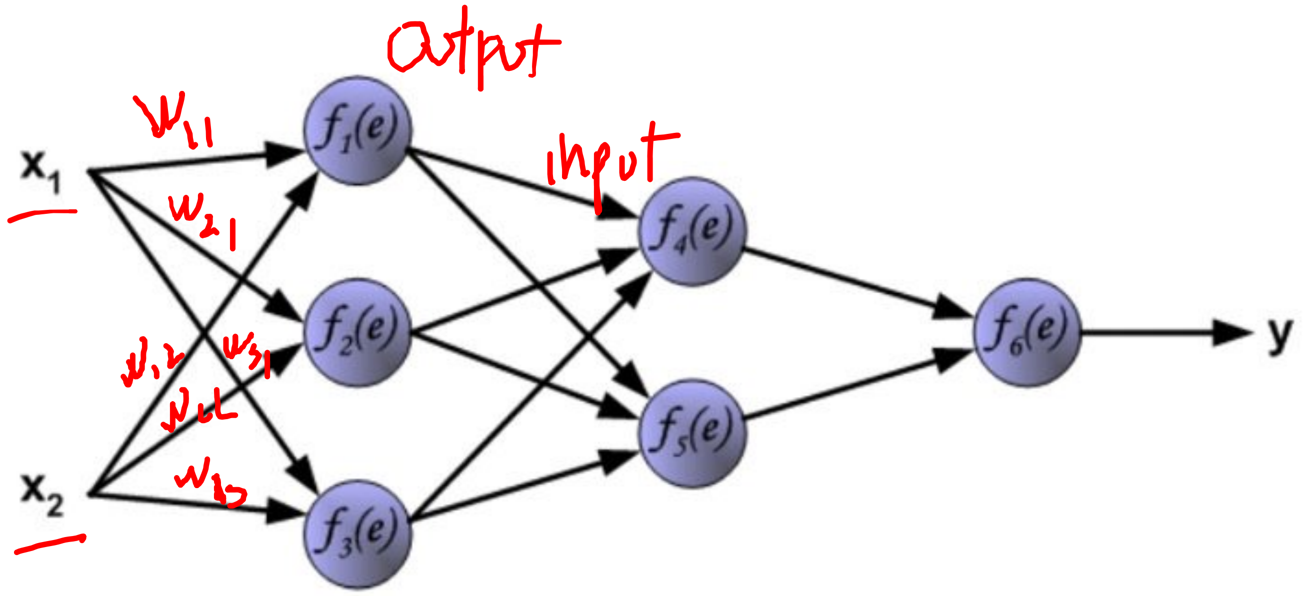
God's People for God's Glory
CALVIN
INSTITUTE OF TECHNOLOGY

Multilayer Perceptron

- Perceptron



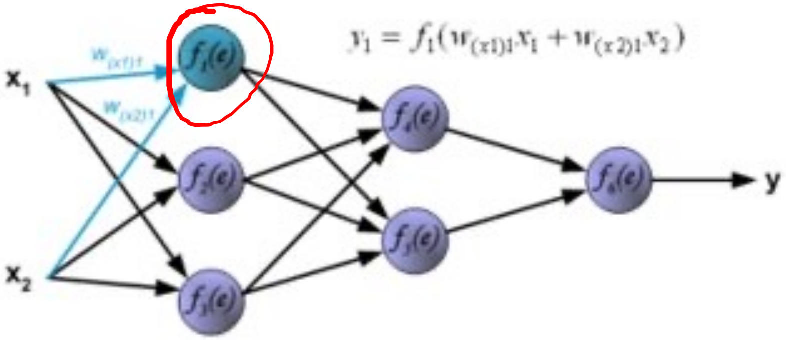
MLP

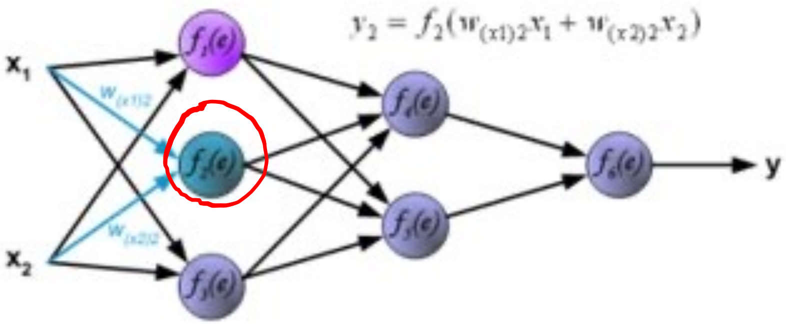


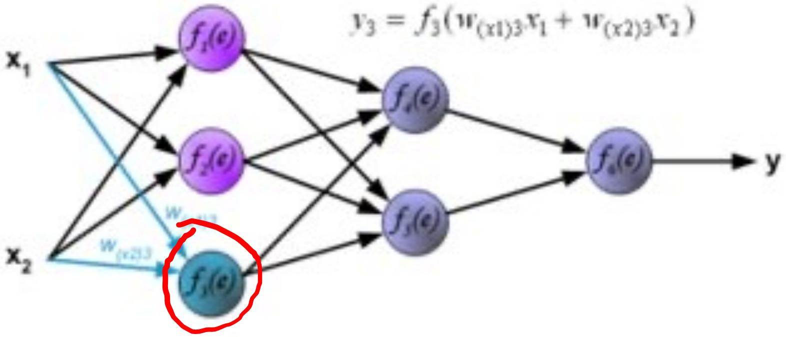
https://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Feedforward Propagation

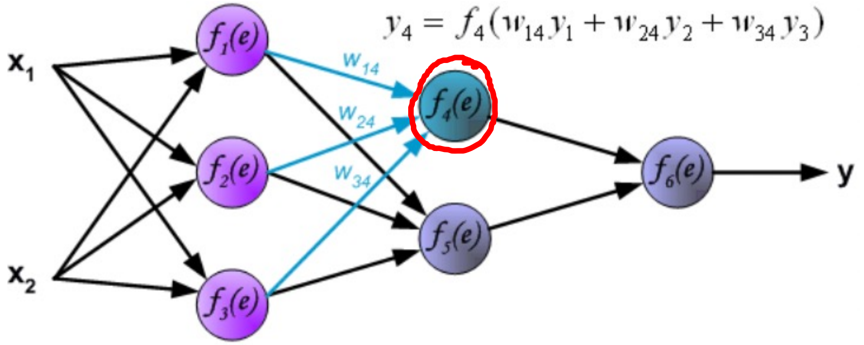
1st layer

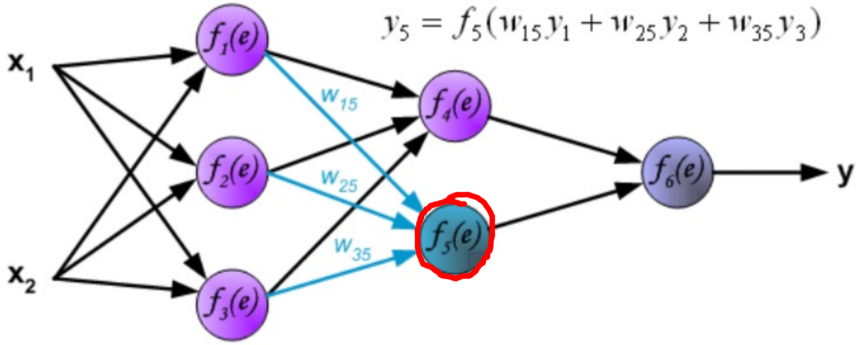




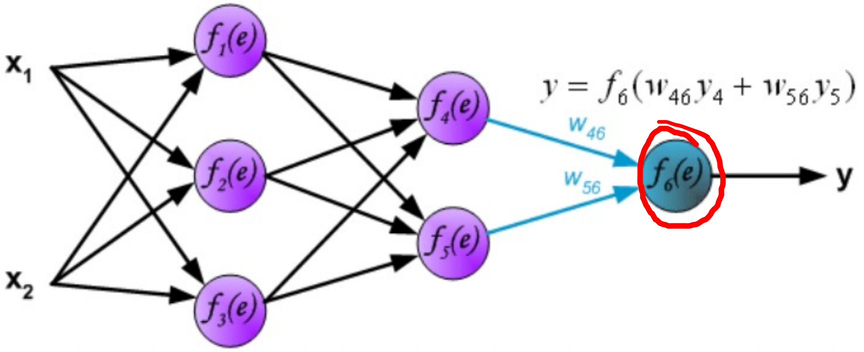


2nd layer





3rd layer

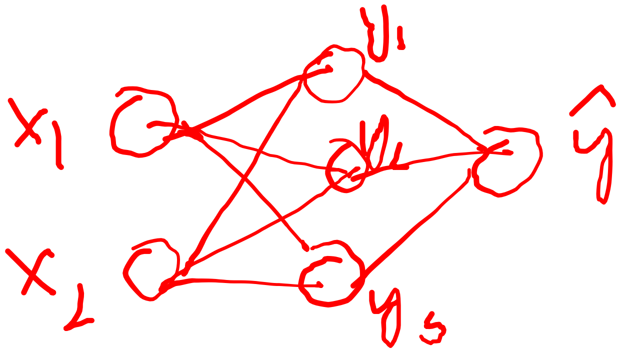


CALVIN INSTITUTE OF TECHNOLOGY

Uji Pemahaman

$$\hat{y} = f(w_1 x_1 + w_2 x_2 + w_0)$$
$$= w_1 x_1 + w_2 x_2 + w_0$$

- Buktikan bahwa linear MLP akan tereduksi menjadi perceptron biasa



$$y_1 = W_{11} x_1 + W_{12} x_2$$
$$y_2 = W_{21} x_1 + W_{22} x_2$$
$$y_3 = W_{31} x_1 + W_{32} x_2$$

$$y = W_\alpha y_1 + W_\beta y_2 + W_\gamma y_3$$
$$= W_\alpha W_{11} x_1 + W_\alpha W_{12} x_2 + W_\beta W_{21} x_1 + W_\beta W_{22} x_2 + W_\gamma W_{31} x_1 + W_\gamma W_{32} x_2$$
$$= \overbrace{(W_\alpha W_{11} + W_\gamma W_{31} + W_\beta W_{21})}^{W_1} x_1 + \overbrace{(W_\alpha W_{12} + W_\beta W_{22} + W_\gamma W_{32})}^{W_2} x_2$$

$$y = W_1 x_1 + W_2 x_2$$

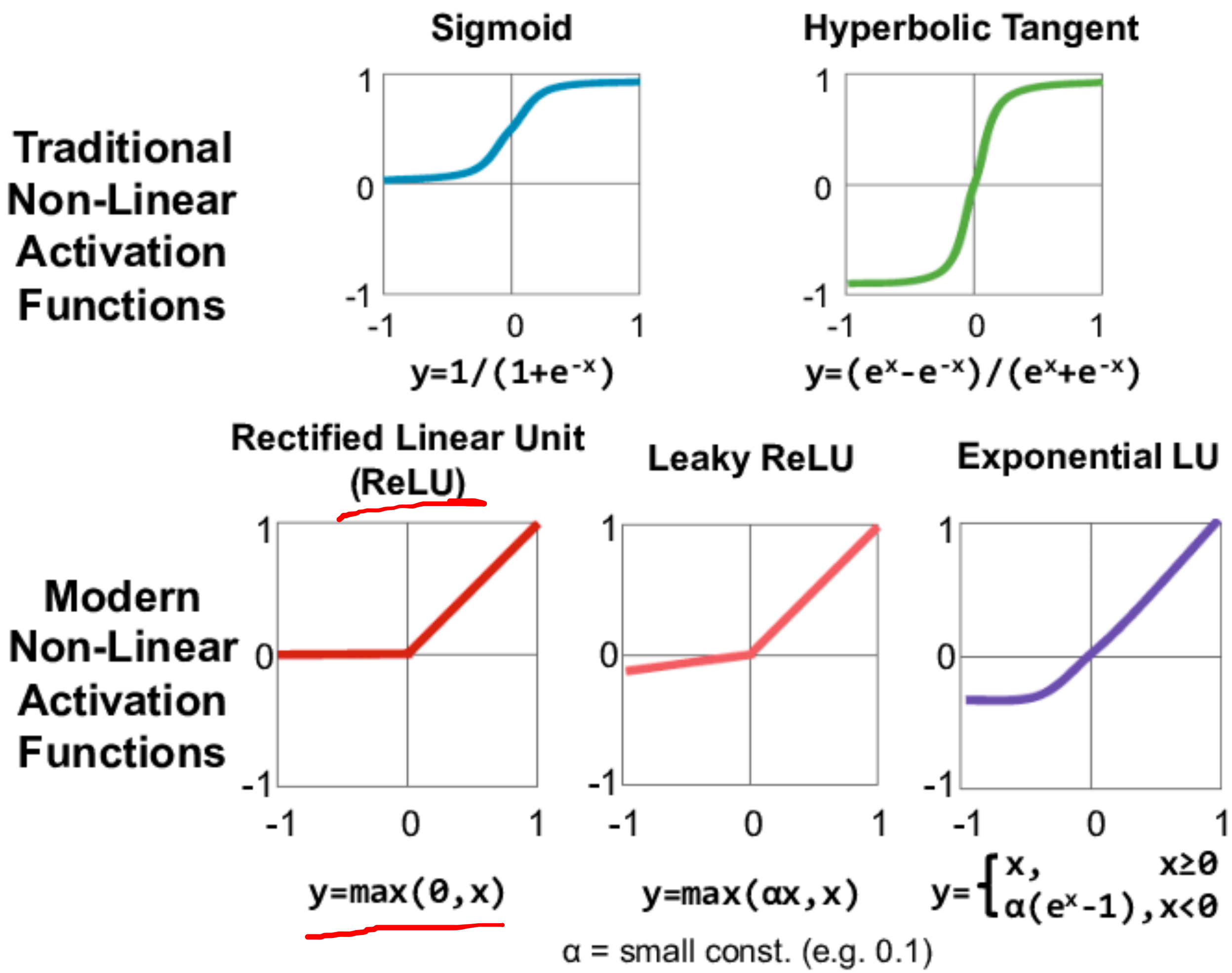
CALVIN INSTITUTE OF TECHNOLOGY

Aktivasi Non-Linear

$y = x < 0$

$y = y * X$

- Tanpa aktivasi non-linear, MLP akan menjadi Perceptron biasa

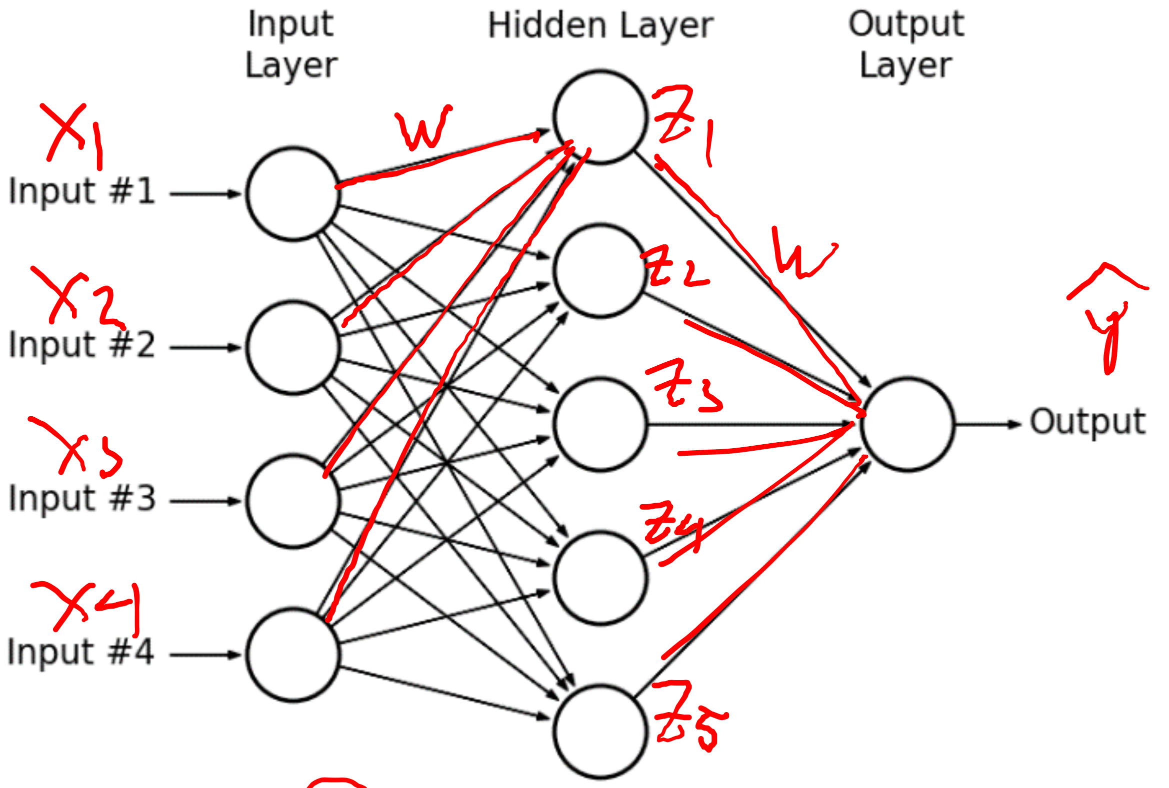


$$\text{ReLU} \quad \begin{array}{c} x \\ \begin{pmatrix} -0.2 \\ 0.3 \\ 0.1 \\ -0.5 \\ 0.7 \\ 0.9 \\ 1.1 \end{pmatrix} \end{array} = \begin{array}{c} y \\ \begin{pmatrix} 0 \\ 0.3 \\ 0.1 \\ 0 \\ 0.7 \\ 0.9 \\ 1.1 \end{pmatrix} \end{array}$$

Representasi MLP

Representasi 1 data

- $\boldsymbol{z}^{[1]} = \boldsymbol{W}^{[1]}\vec{\boldsymbol{x}} + \boldsymbol{b}^{[1]}$
- $\vec{\boldsymbol{a}}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$
- $\boldsymbol{z}^{[2]} = \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$
- $\boldsymbol{a}^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[2]})$



$$\begin{pmatrix} -w_1^{[1]T}- \\ -w_2^{[1]T}- \\ -w_3^{[1]T}- \\ -w_4^{[1]T}- \\ -w_5^{[1]T}- \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \\ b_5^{[1]} \end{pmatrix} = \begin{pmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \\ z_5^{[1]} \end{pmatrix}, \begin{pmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \\ a_5^{[1]} \end{pmatrix} = \sigma\begin{pmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \\ z_5^{[1]} \end{pmatrix}$$

$$\left(-w^{[2]T}-\right)\begin{pmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \\ a_5^{[1]} \end{pmatrix} + \left(b^{[2]}\right) = \left(z^{[2]}\right),\ \left(a^{[2]}\right) = \sigma\left(z^{[2]}\right)$$

CALVIN INSTITUTE OF TECHNOLOGY

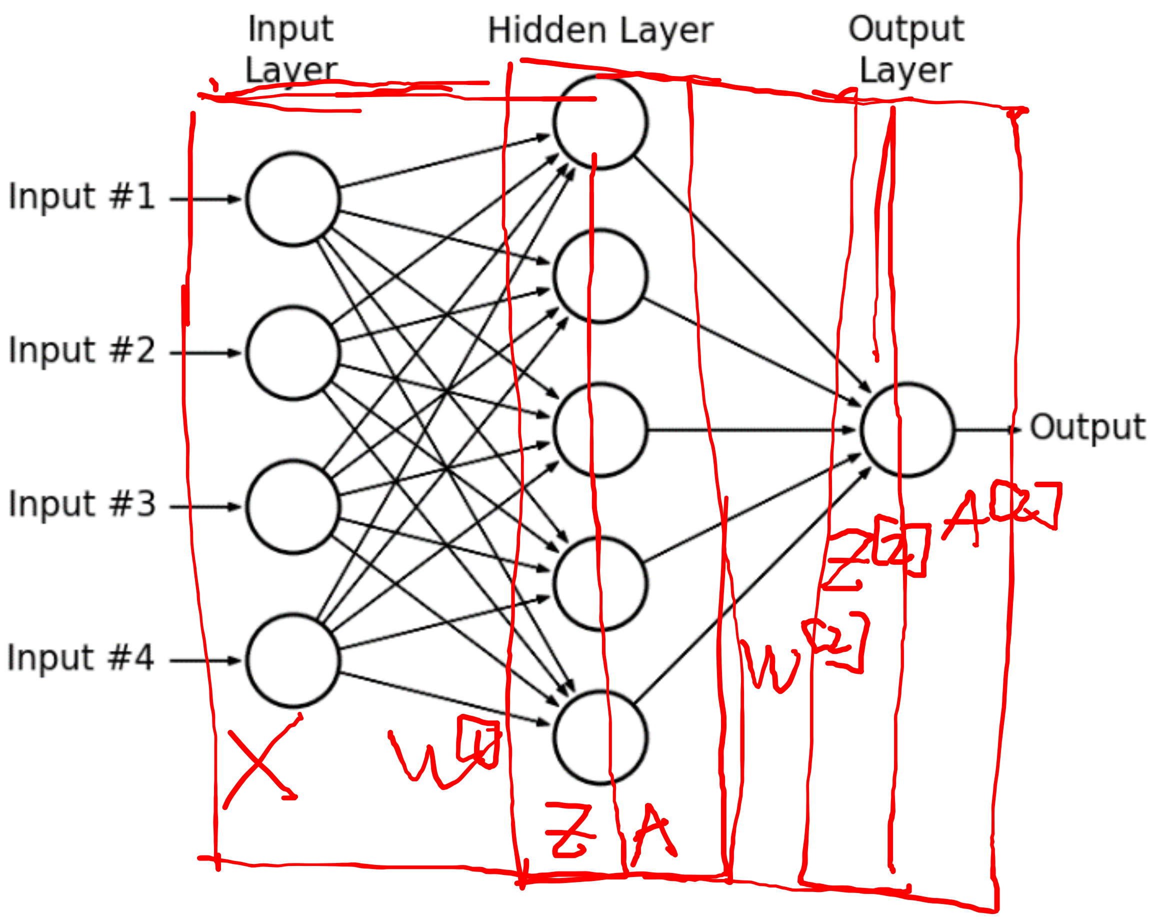
Representasi MLP

Representasi Tensor m data

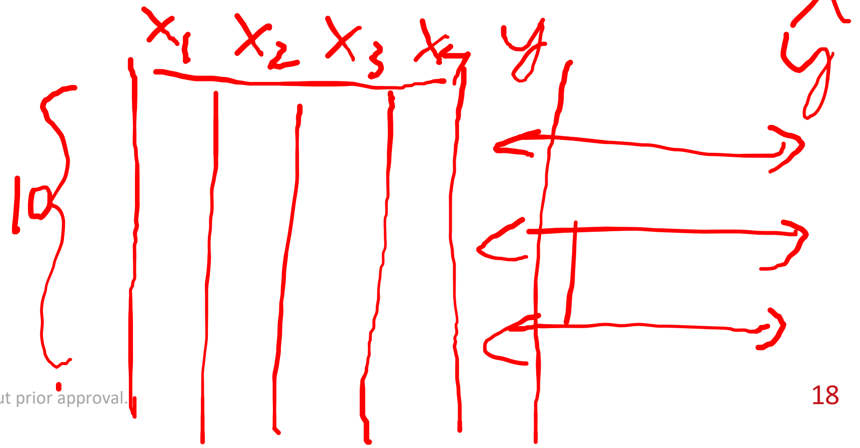
- $Z^{[1]} = W^{[1]}X + b^{[1]}$
- $A^{[1]} = \sigma(Z^{[1]})$
- $Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
- $A^{[2]} = \sigma(Z^{[2]})$

mdata

- $X = \left(\begin{array}{ccc} | & | & | \\ x^{(1)} & \ldots & x^{(m)} \\ | & | & | \end{array}\right)$ } n fitur
- $A^{[1]} = \left(\begin{array}{ccc} | & | & | \\ a^{[1](1)} & \ldots & a^{[1](m)} \\ | & | & | \end{array}\right)$



m=10
n=4



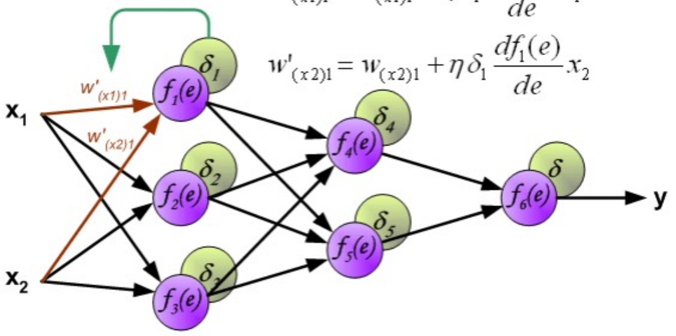
Gradient Descent

Perbaharui bobot w untuk setiap lapisan (layer) dengan memakai gradient descent, berdasarkan akumulasi error pada masing-masing neuron

$$w^{[l]}_{11} := w^{[l]}_{11} - \frac{\partial J(y,\hat{y})}{\partial w_{11}}$$

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

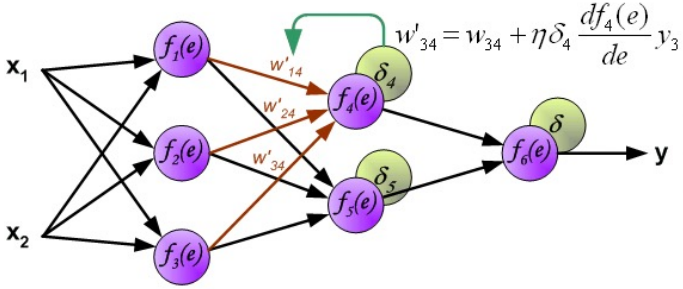
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

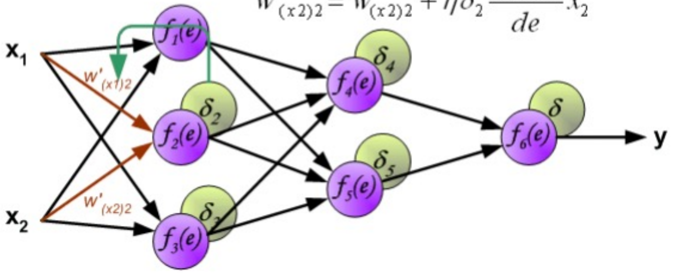
$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$



$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1$$

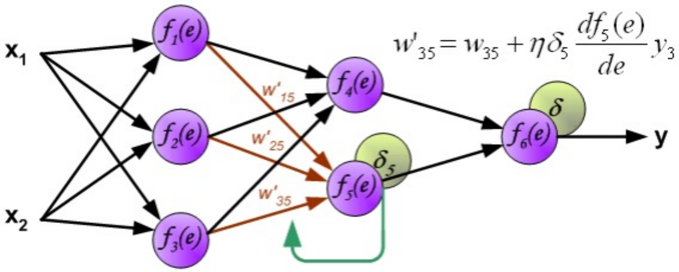
$$w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$



$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

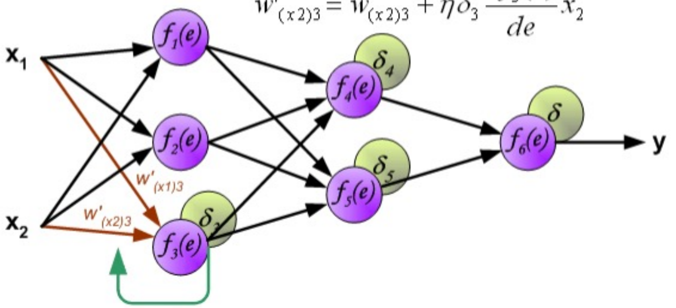
$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$



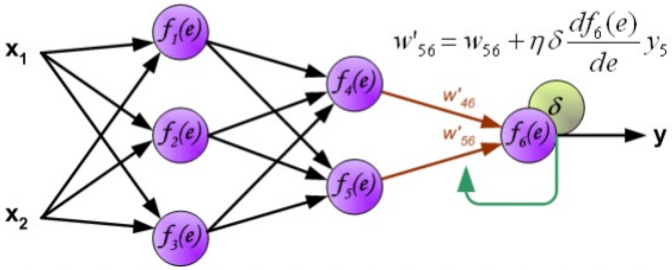
$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$



$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$



Algoritma Backpropagation

- Definisikan $g[u^{(i)}] = \frac{\partial u^{(n)}}{\partial u^{(i)}}$
- $g[u^{(n)}] \leftarrow 1$
- Untuk $j = n - 1$ sampe 1 hitung
  - $\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}$
  - Menggunakan $g[u^{(j)}] \leftarrow \sum_{i:j \in Pa(u^{(i)})} g[u^{(i)}] \frac{\partial u^{(i)}}{\partial u^{(j)}}$
- Return $\{g[u^{(i)}] | i = 1, \dots, n\}$

$y_1 = (w_{11} x_1 + w_{12} x_2)$

$y_2 = (w_{21} x_1 + w_{22} x_2)$

$\hat{y} = (w_1^{[2]} y_1 + w_2^{[2]} y_2)$

$J = (\hat{y} - y)^2$

$\frac{\partial J}{\partial w_{11}} = 2(\hat{y} - y) w_1 x_1$

Feedforward dan Backpropagation pada MLP

Algorithm 6.3 Forward propagation through a typical deep neural network and the computation of the cost function. The loss $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$ depends on the output $\hat{\boldsymbol{y}}$ and on the target $\boldsymbol{y}$ (see section 6.2.1.1 for examples of loss functions). To obtain the total cost J, the loss may be added to a regularizer $\Omega(\theta)$, where θ contains all the parameters (weights and biases). Algorithm 6.4 shows how to compute gradients of J with respect to parameters $\boldsymbol{W}$ and $\boldsymbol{b}$. For simplicity, this demonstration uses only a single input example $\boldsymbol{x}$. Practical applications should use a minibatch. See section 6.5.7 for a more realistic demonstration.

Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \dots, l\}$, the weight matrices of the model
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \dots, l\}$, the bias parameters of the model
Require: $\boldsymbol{x}$, the input to process
Require: $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \dots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda\Omega(\theta)$

Algorithm 6.4 Backward computation for the deep neural network of algorithm 6.3, which uses in addition to the input $\boldsymbol{x}$ a target $\boldsymbol{y}$. This computation yields the gradients on the activations $\boldsymbol{a}^{(k)}$ for each layer k, starting from the output layer and going backwards to the first hidden layer. From these gradients, which can be interpreted as an indication of how each layer's output should change to reduce error, one can obtain the gradient on the parameters of each layer. The gradients on weights and biases can be immediately used as part of a stochastic gradient update (performing the update right after the gradients have been computed) or used with other gradient-based optimization methods.

After the forward computation, compute the gradient on the output layer:
$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
for $k = l, l-1, \dots, 1$ **do**
  Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if f is element-wise):
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$
  Compute gradients on weights and biases (including the regularization term, where needed):
  $\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$
  $\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$
  Propagate the gradients w.r.t. the next lower-level hidden layer's activations:
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top} \boldsymbol{g}$
end for

Deep learning, p212-213

CALVIN INSTITUTE OF TECHNOLOGY

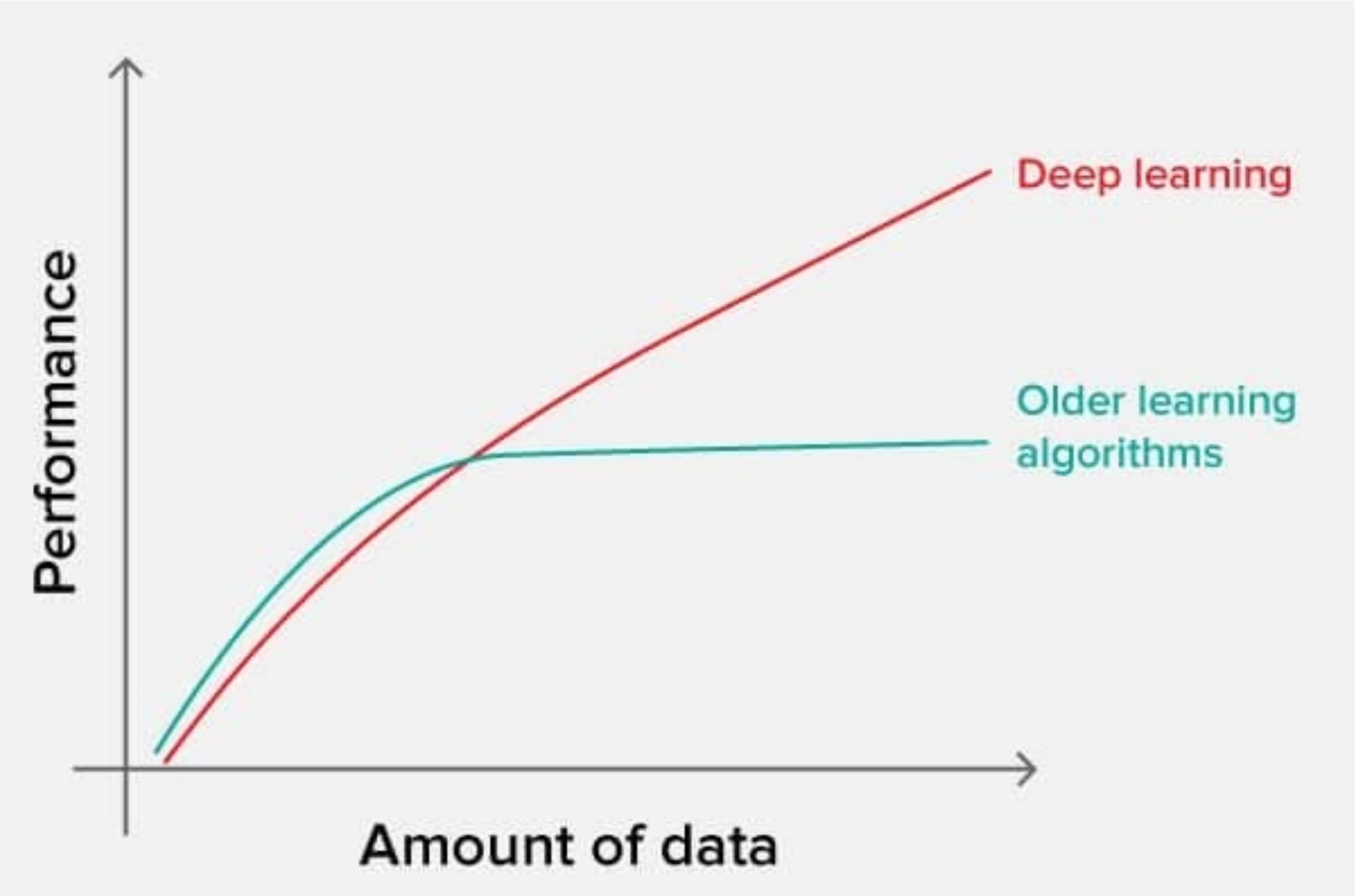
Deep Learning

Deep Learning

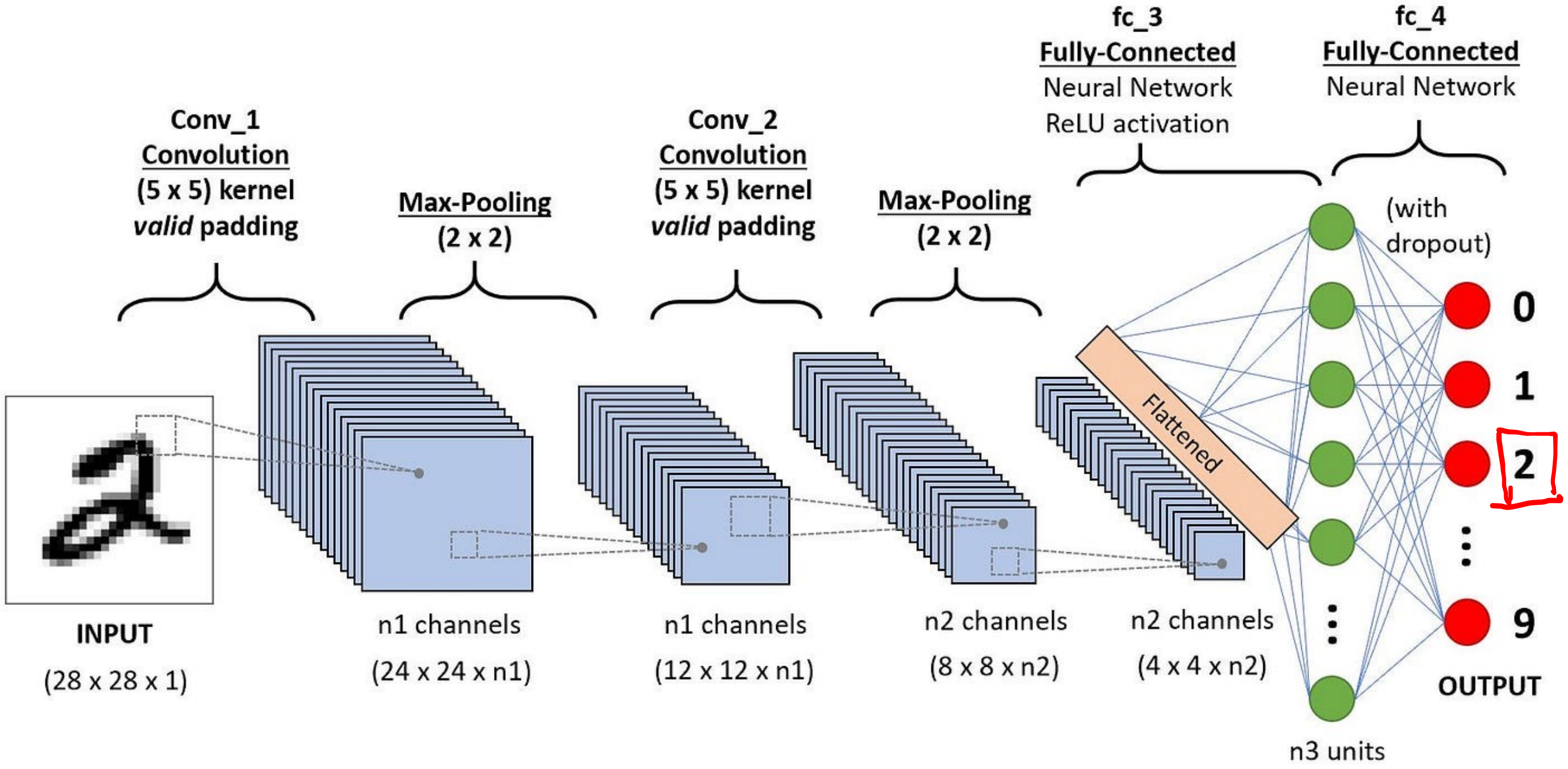
- Memanfaatkan kapasitas pembelajaran dari neural network yang berlapis-lapis
- Arsitektur yang umum:
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
  - Graph Neural Network (GNN)
  - Transformer
- Aplikasi:
  - Computer Vision
  - Natural Language Processing
  - Knowledge Graph

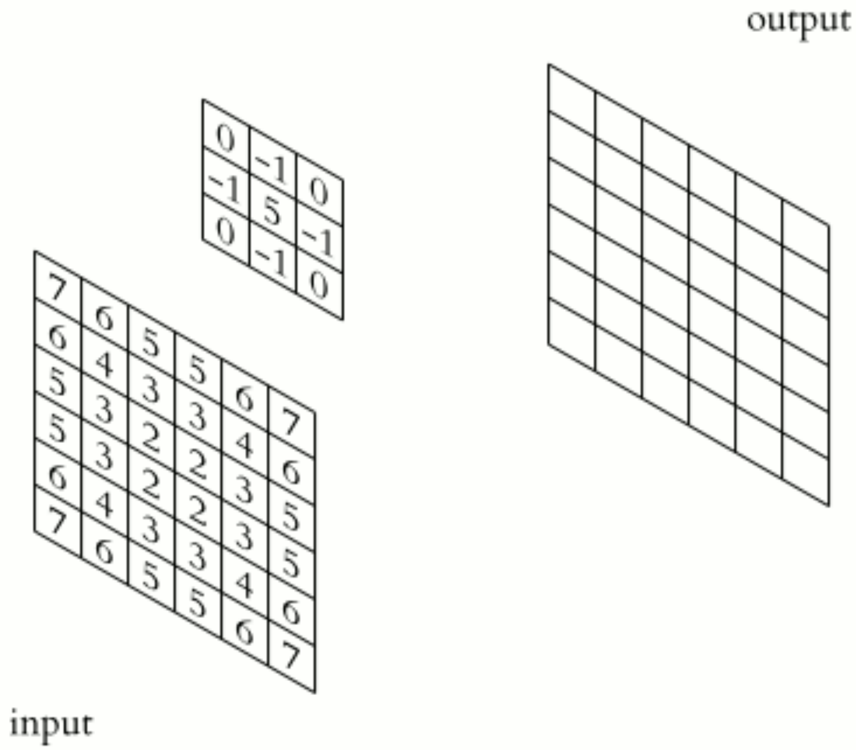
Mengapa Deep Learning?

- Dapat menggeneralisasi data yang sangat besar



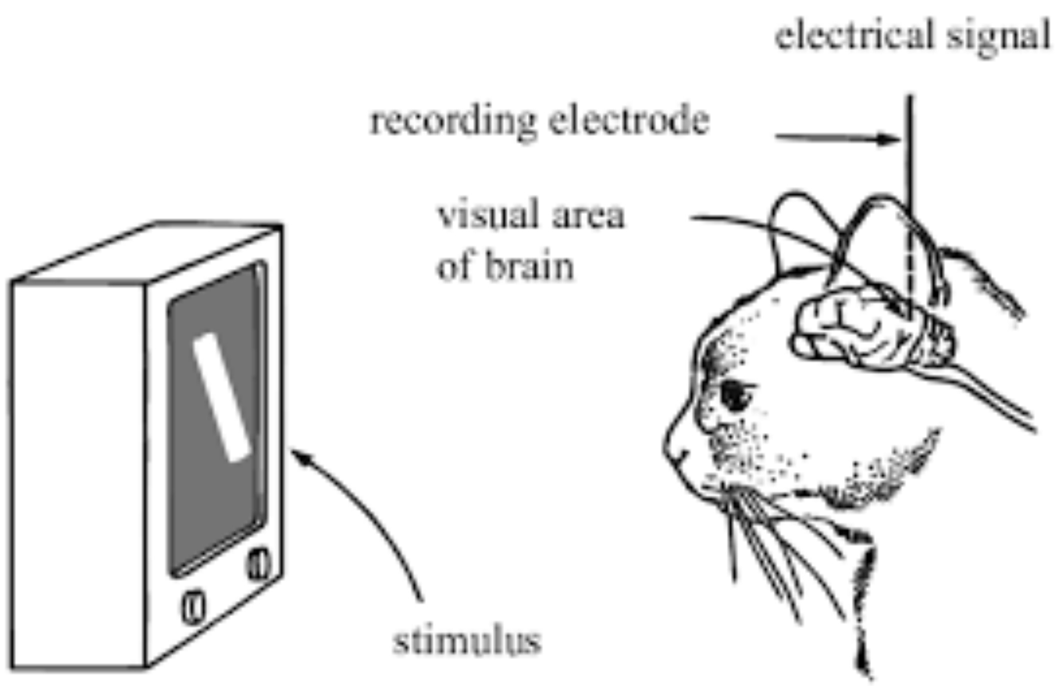
Convolutional Neural Network

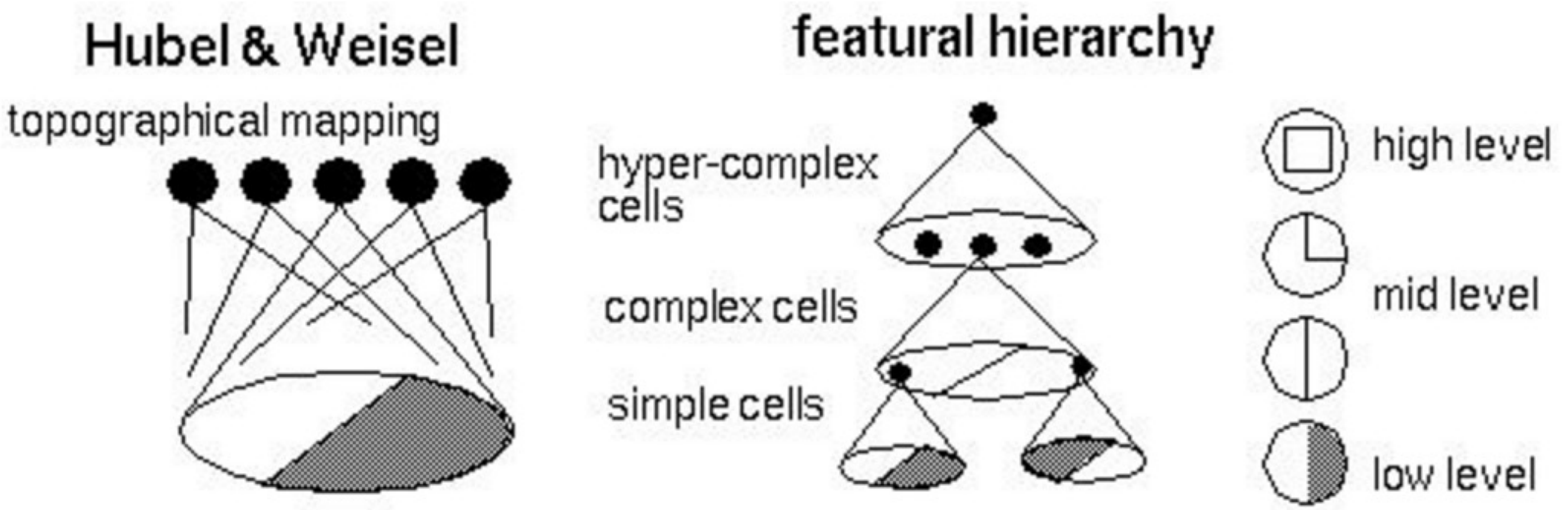




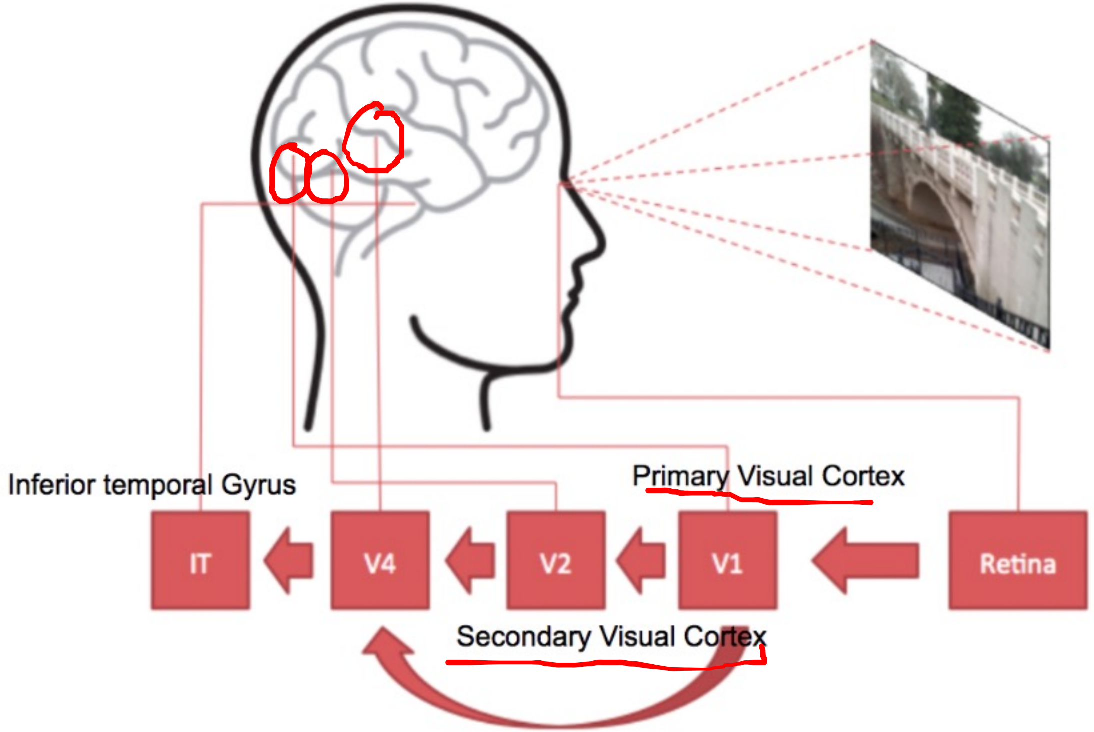
Insight dari Neurosains

- **1959**: Hubel & Wiesel
  - Medan resepsi dari satu neuron di korteks striate pada kucing
  - https://www.youtube.com/watch?v=IOHayh06LJ4
- **1962**: Receptive Fields
  - Interaksi binocular dan arsitektur fungsional di korteks visual pada kucing
- Hierarchical Organizations
- **2005**: Neural abstraction
  - Medial temporal lobe
  - Menangkap abstraksi
  - “Halle Berry Neuron”





Visual Cortex



V1: Edge detection, etc.

V2: Extract simple visual properties (orientation, spatial frequency, color, etc)

V4: Detect object features of intermediate complexity

TI: Object recognition.

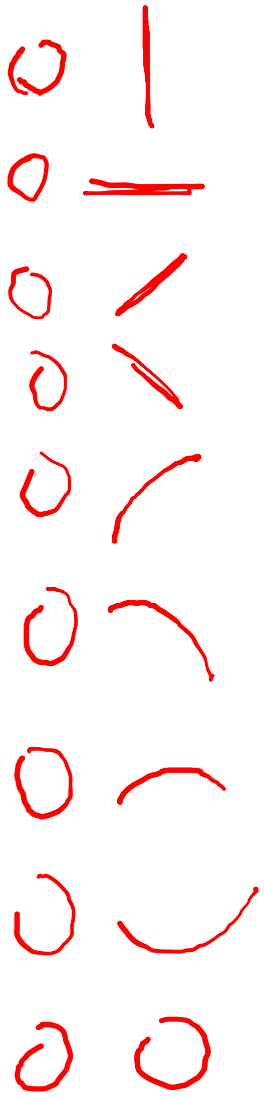
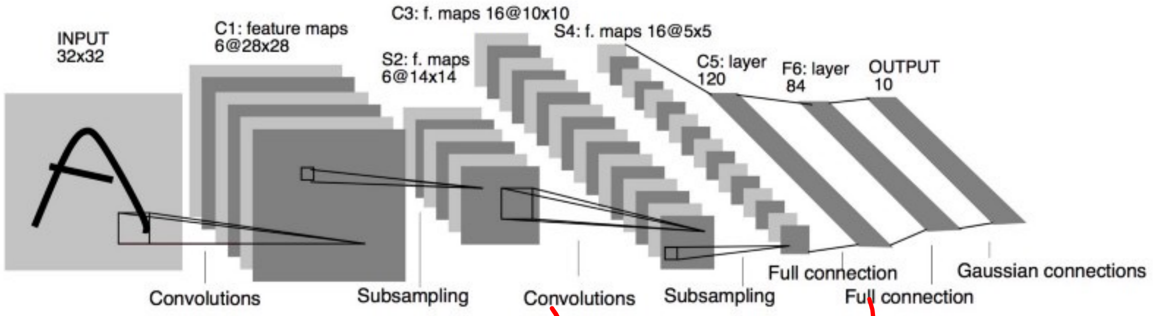
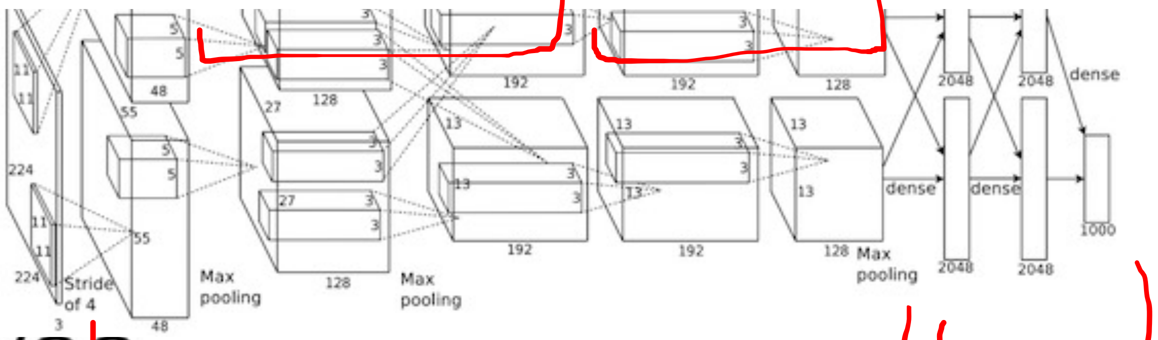


Fig. 3. Illustration of the human visual cortex system. (Image source: Wang & Raj 2017)

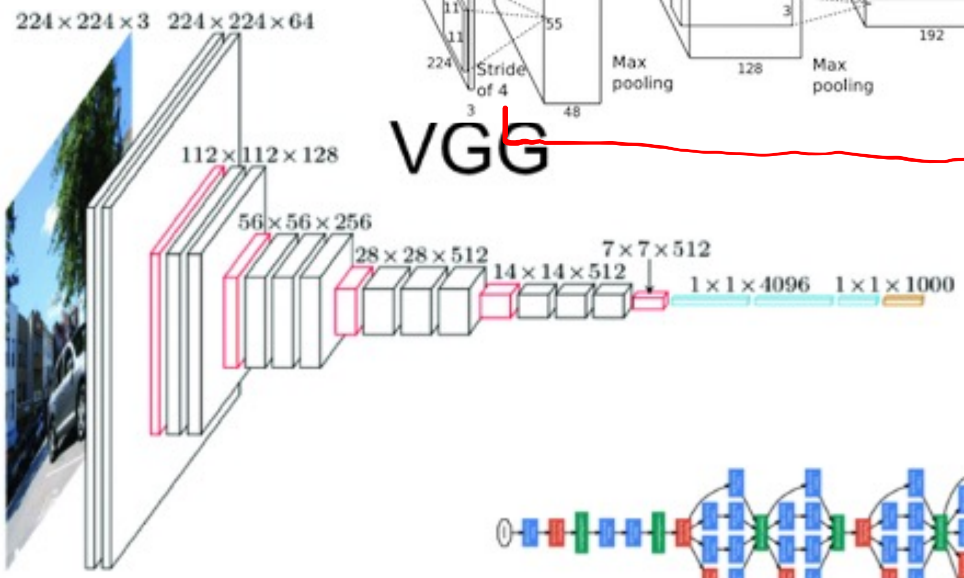
Sejarah CNN

- 1998: LeNet
  - 2 Lapisan Konvolusi
  - 2 Lapisan MLP
- 2012: AlexNet
  - 5 Lapisan Konvolusi
  - 3 Lapisan MLP
- 2014: VGG
  - Sangat Dalam (19 lapisan)
- 2014: GoogleNet
  - Menggunakan Modul Insepsi
  - 22 Lapisan
- 2015: ResNet
  - Menggunakan Modul Residual
  - 152 lapisan
- 2016: SqueezeNet, ENET, Xception
  - Memiliki semua fitur sebelumnya
  - Lebih efisien

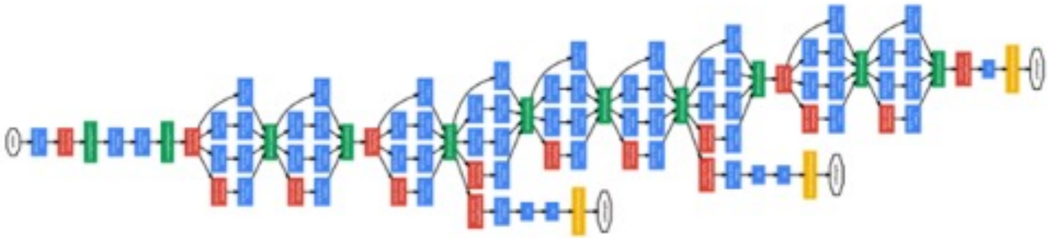




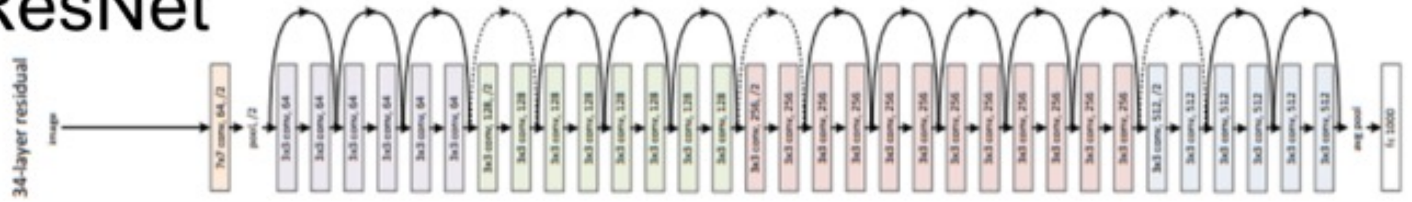
VGG



GoogLeNet



ResNet



Akurasi dan Operasi

- “An Analysis of Deep Neural Network Models for Practical Applications” Alfredo Canziani, Adam Paszke, Eugenio Culurciello Published 2016 in ArXiv

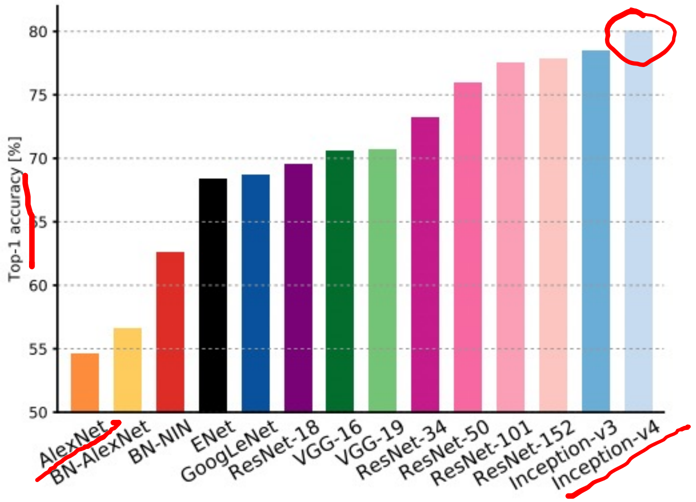


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

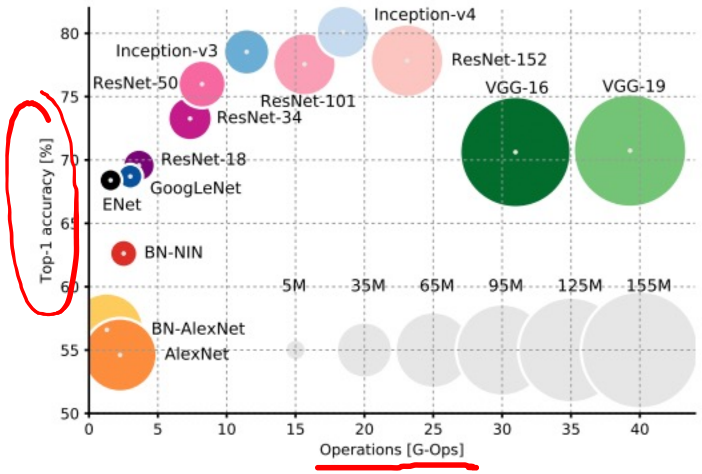


Figure 2: **Top1 *vs.* operations, size ∝ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same *y*-axis, and the grey dots highlight the centre of the blobs.

CALVIN INSTITUTE OF TECHNOLOGY

Recurrent Neural Network

- CNN efektif dalam memproses **grid** data
- RNN efektif dalam memproses **sequence** data

- Misalkan ada Neural Network yang memiliki input/output berikut
- Input: I am studying deep learning
- Output: Saya sedang belajar pembelajaran yang mendalam

- Contoh tersebut menunjukkan bahwa:
  - Konteks dan pola urutan sebelum-sesudah (sequence) sangat penting
  - Terdapat jenis data dengan input maupun output berbentuk sequence
  - Sulit untuk membuat fixed context window (mungkin ada kalimat yang lebih panjang dari sebelumnya)

MicroNetwork Motifs

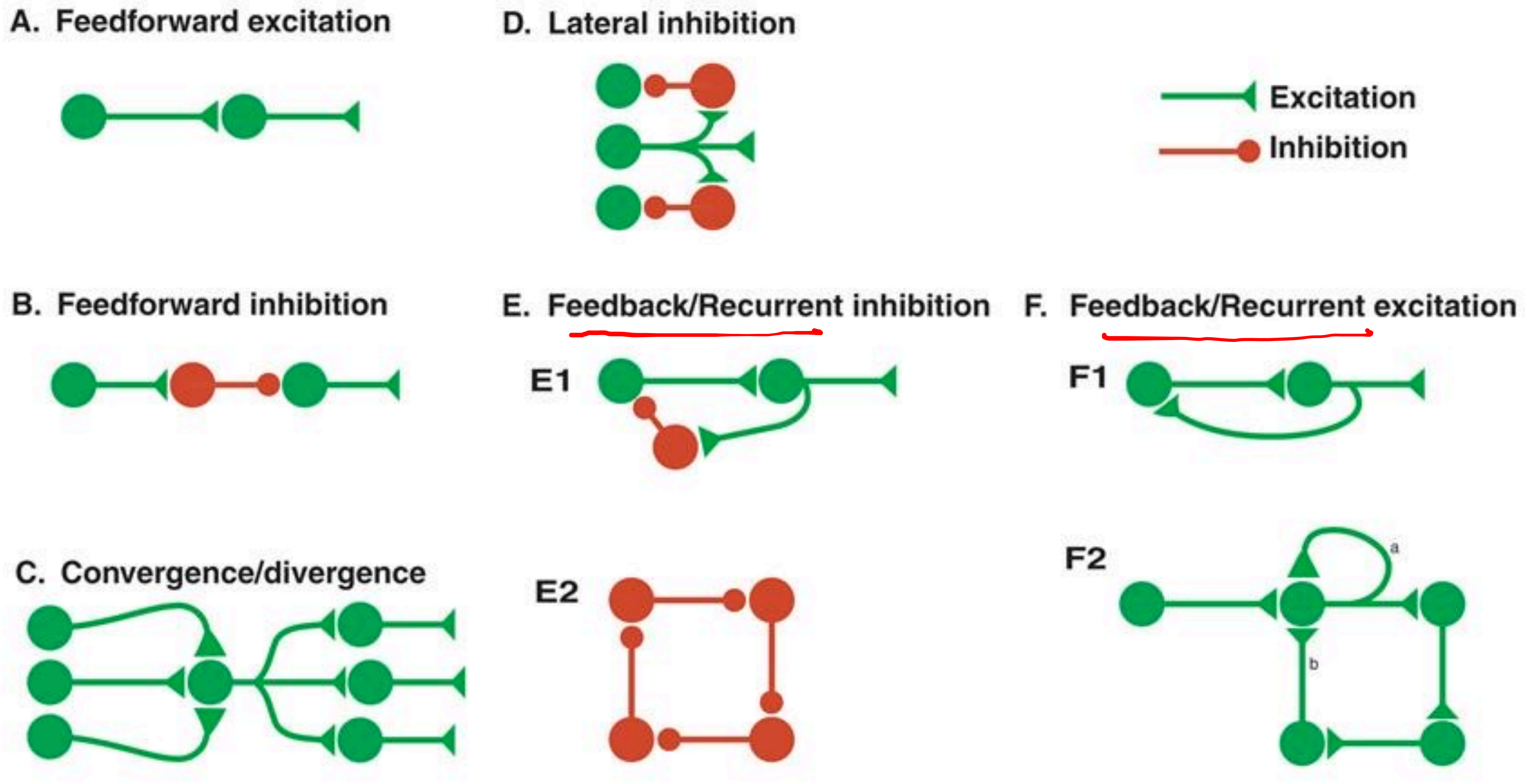
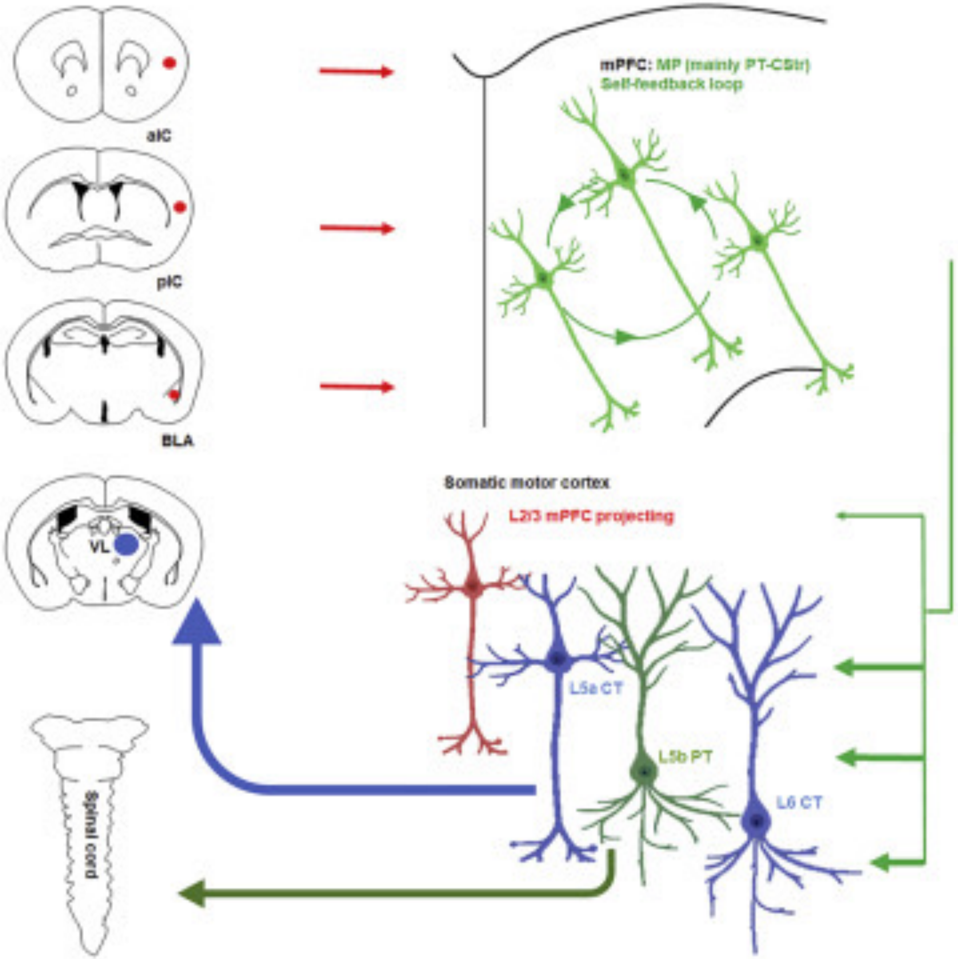
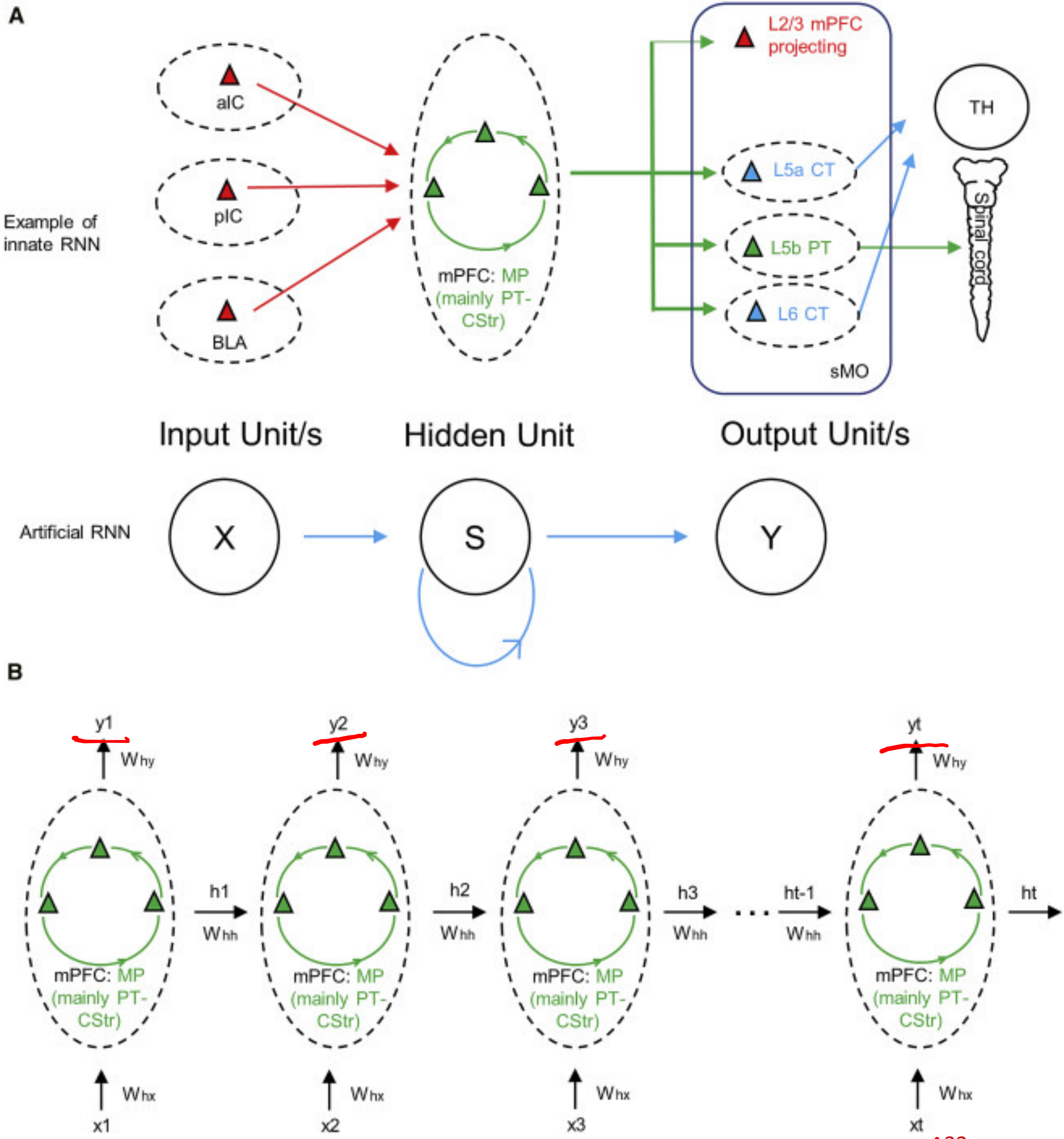


Figure 6

Basis Neurosains

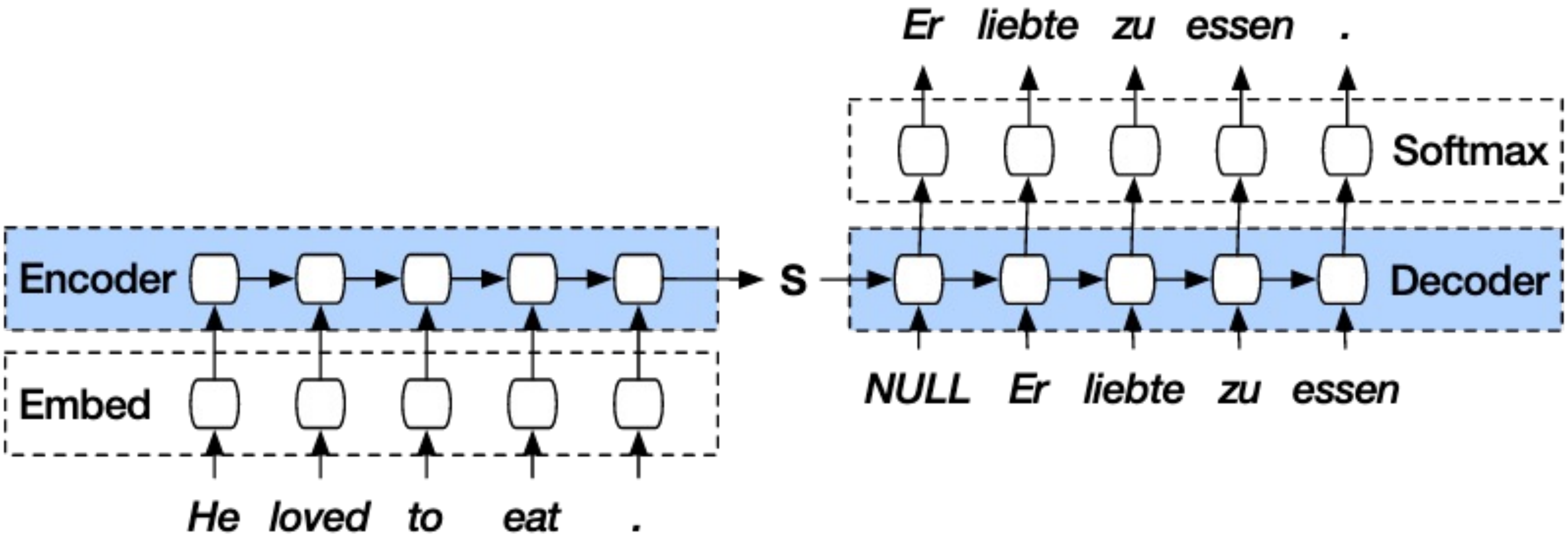




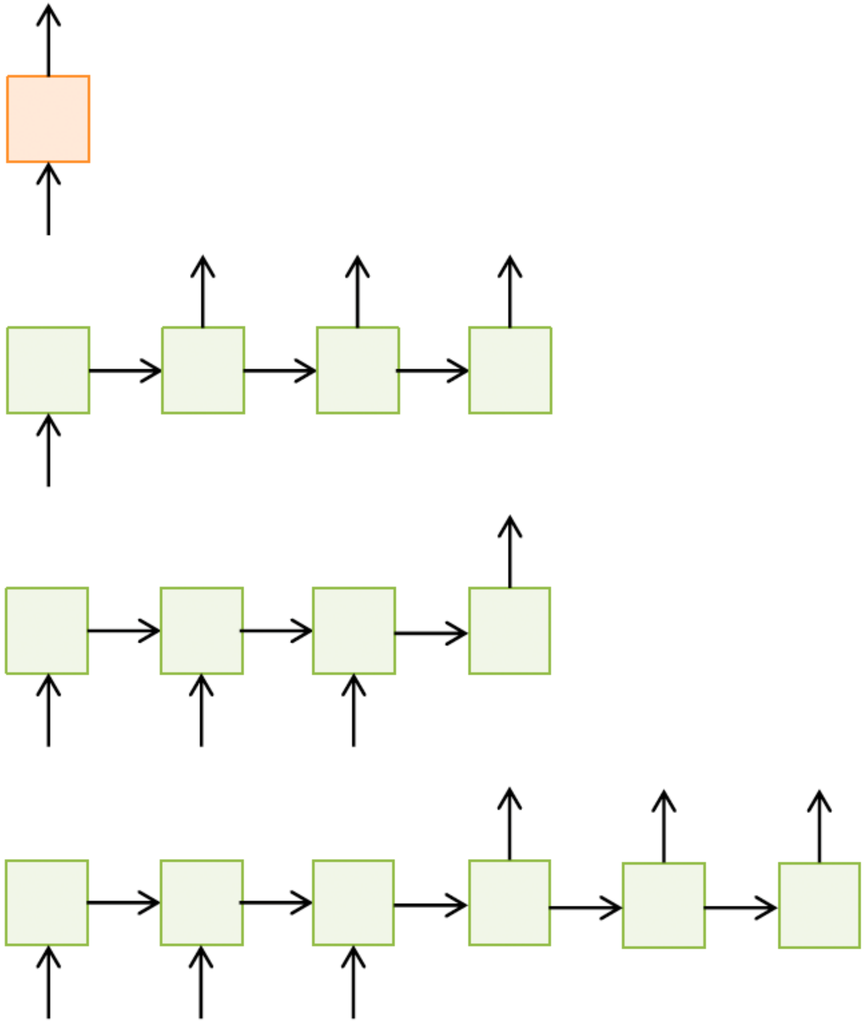
CALVIN INSTITUTE OF TECHNOLOGY

RNN

- RNN dapat menerima input maupun menghasilkan output dengan bentuk sequence (urut-urutan)



Skenario input-output RNN



Single - Single — Feed-forward Network MLP

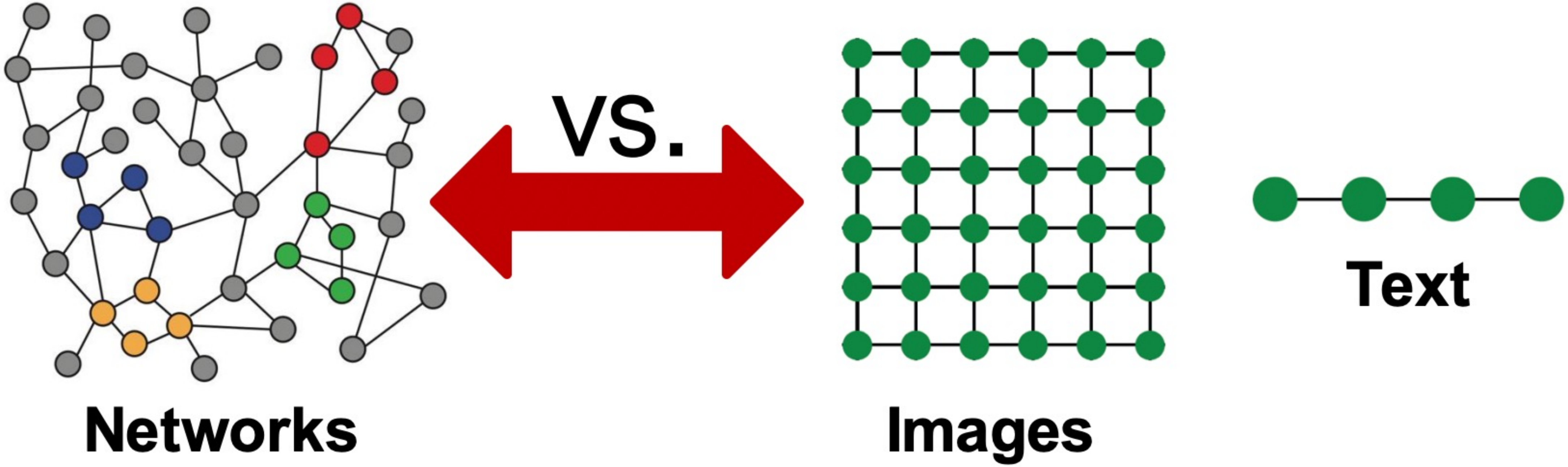
Single - Multiple — Image Captioning

Multiple - Single — Sentiment Classification

Multiple - Multiple — Machine Translation

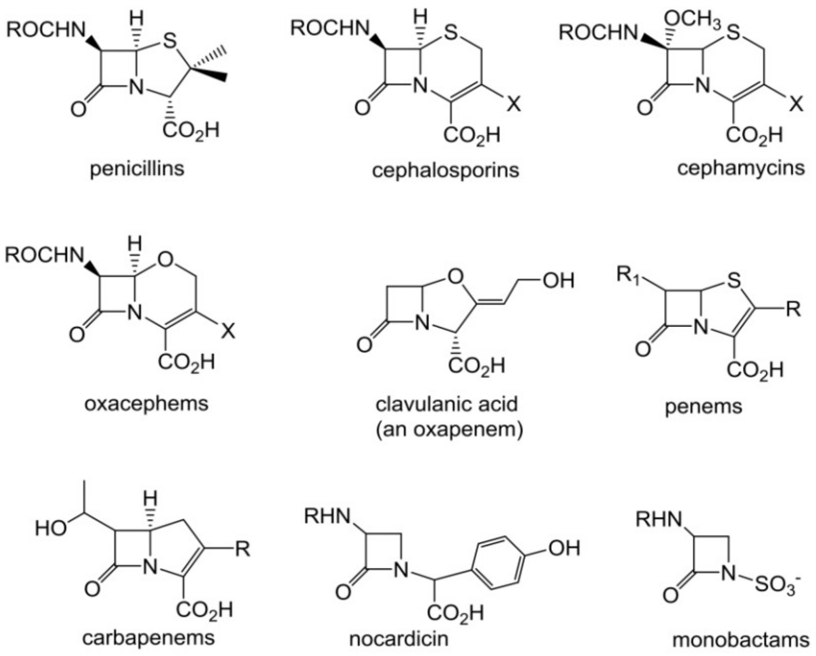
Graph Neural Network

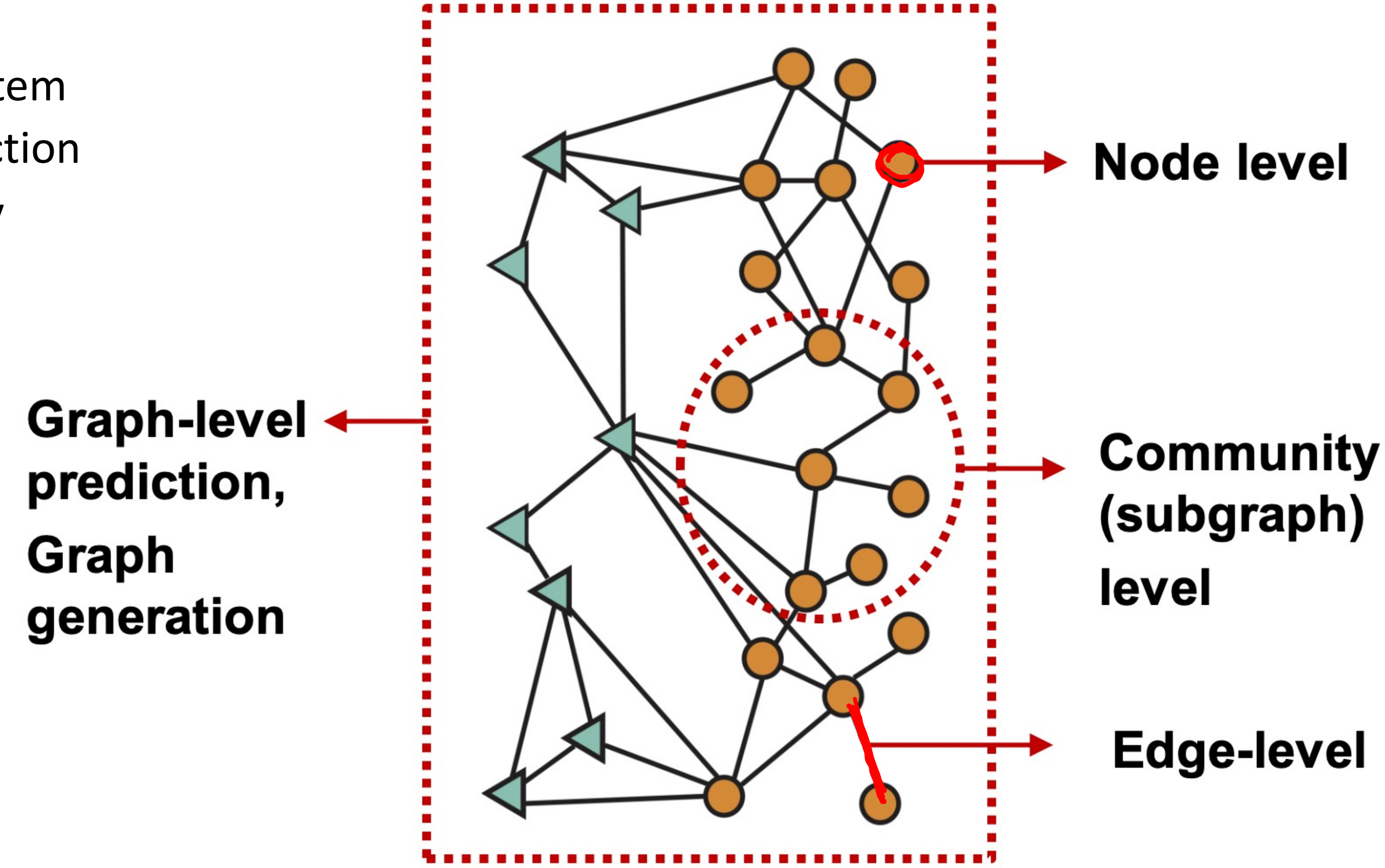
- CNN efektif dalam memproses **grid** data
- RNN efektif dalam memproses **sequence** data
- GNN efektif dalam memproses **graph** data



Level Jejaring

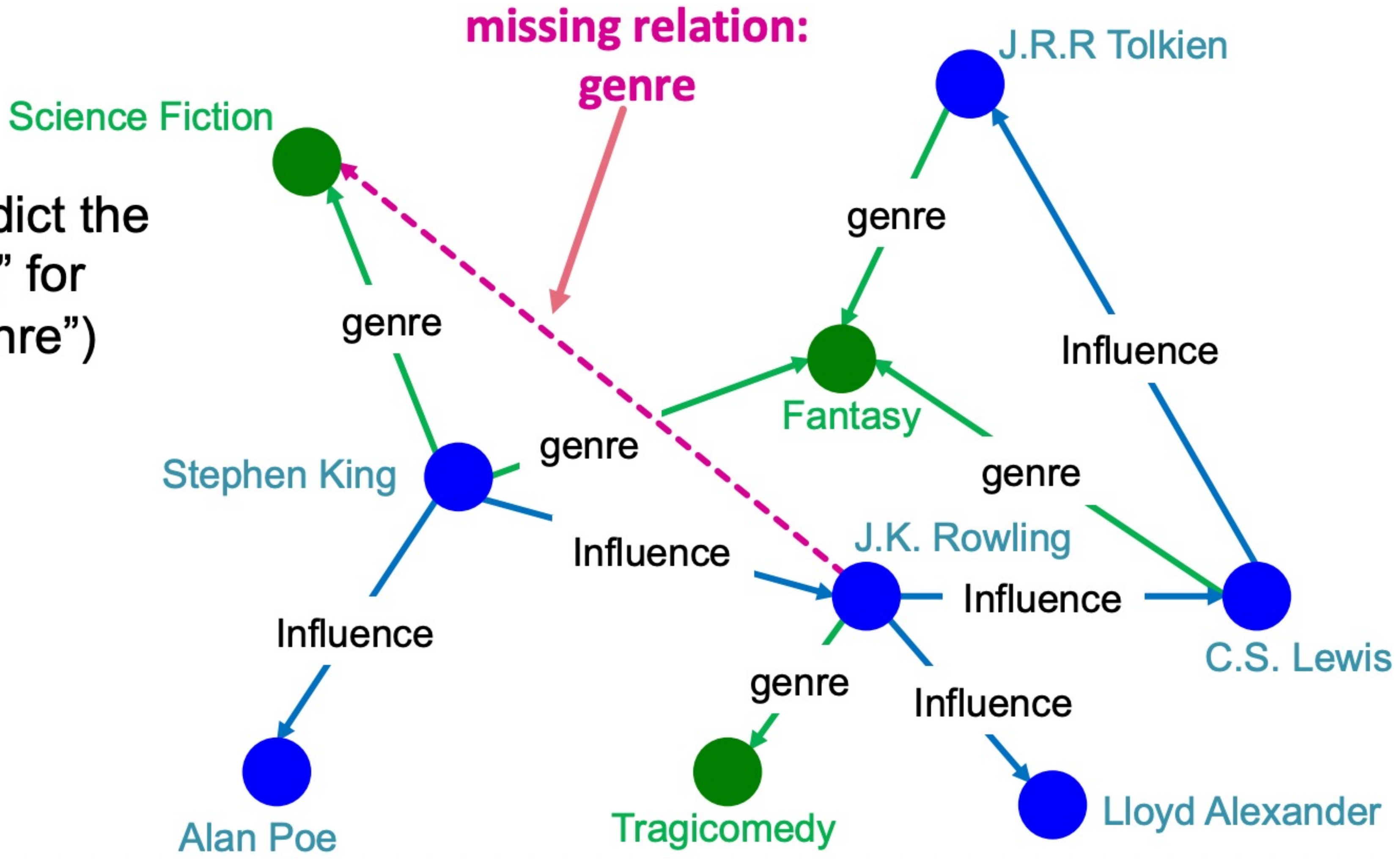
- Permodelan dapat dilakukan pada level jejaring yang berbeda:
  - Prediksi node: protein folding
  - Prediksi link: recommender system
  - Prediksi subgraph: traffic prediction
  - Prediksi jejaring: drug discovery





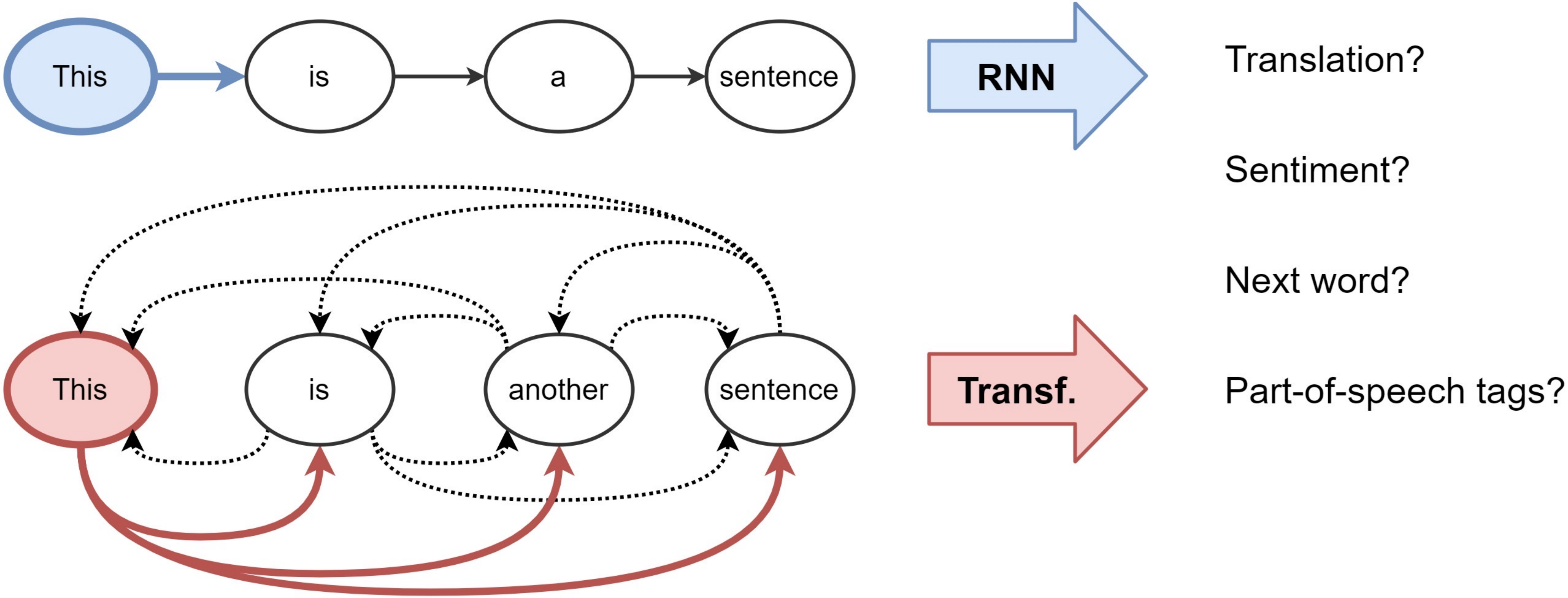
Contoh: prediksi missing relation

Example task: Predict the tail "Science Fiction" for ("J.K. Rowling", "genre")



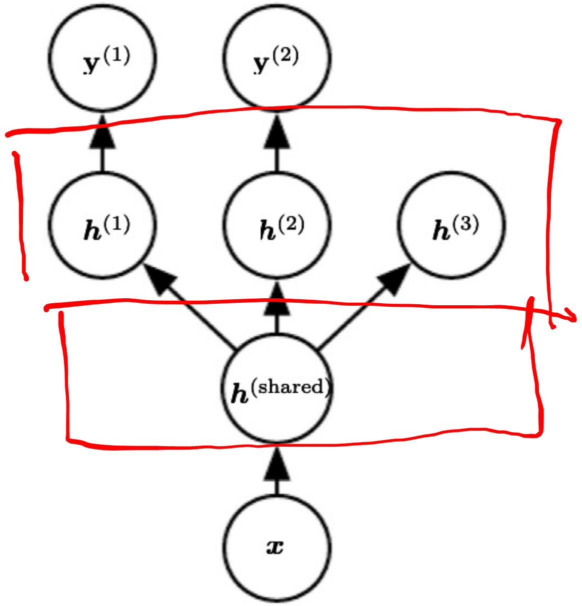
Contoh: Prediksi jalan

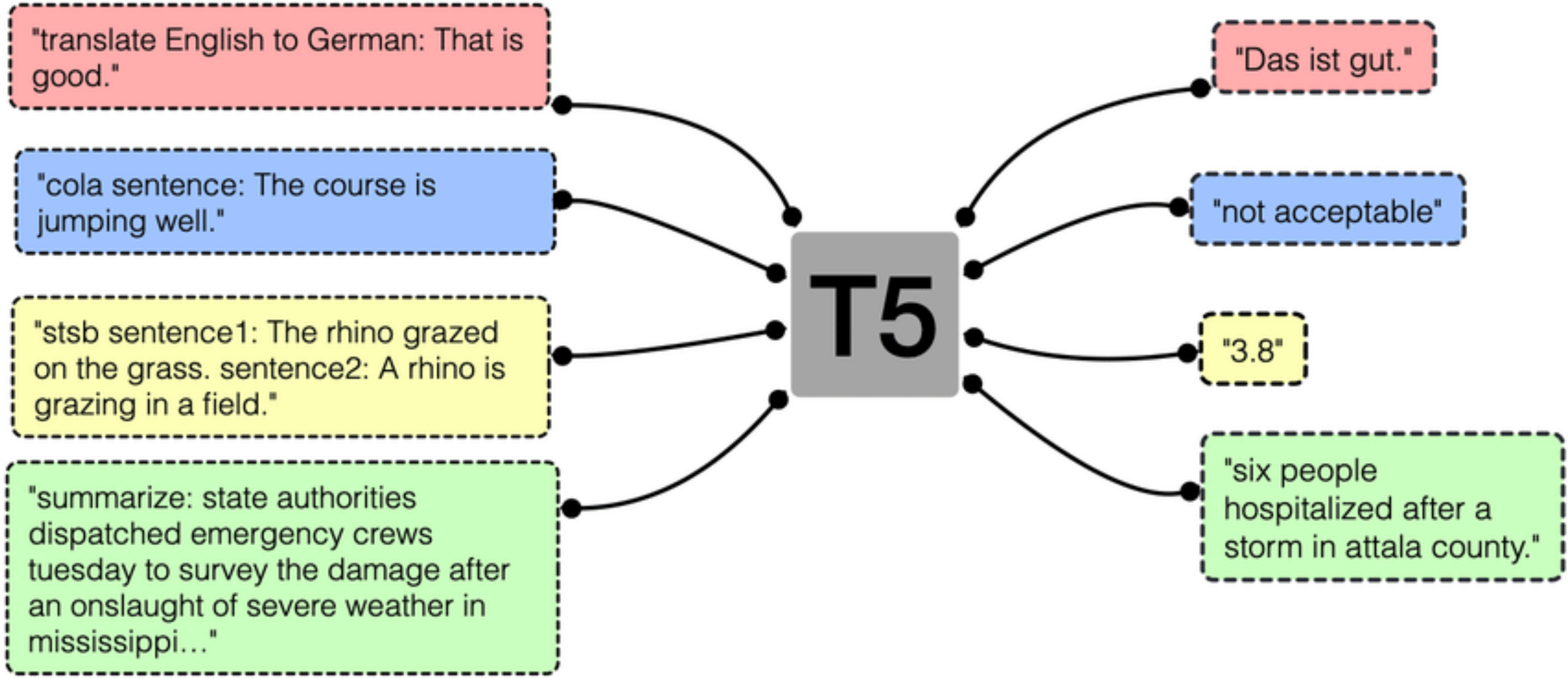
Transformer adalah GNN



Multitask Learning

- Pemodelan multitask dapat membantu generalisasi
- Dipakai jika jumlah data untuk beberapa aplikasi sekaligus mirip
- Model dapat dibagi menjadi 2 bagian (beserta parameternya):
  - Task Specific Parameters
  - Generic parameters

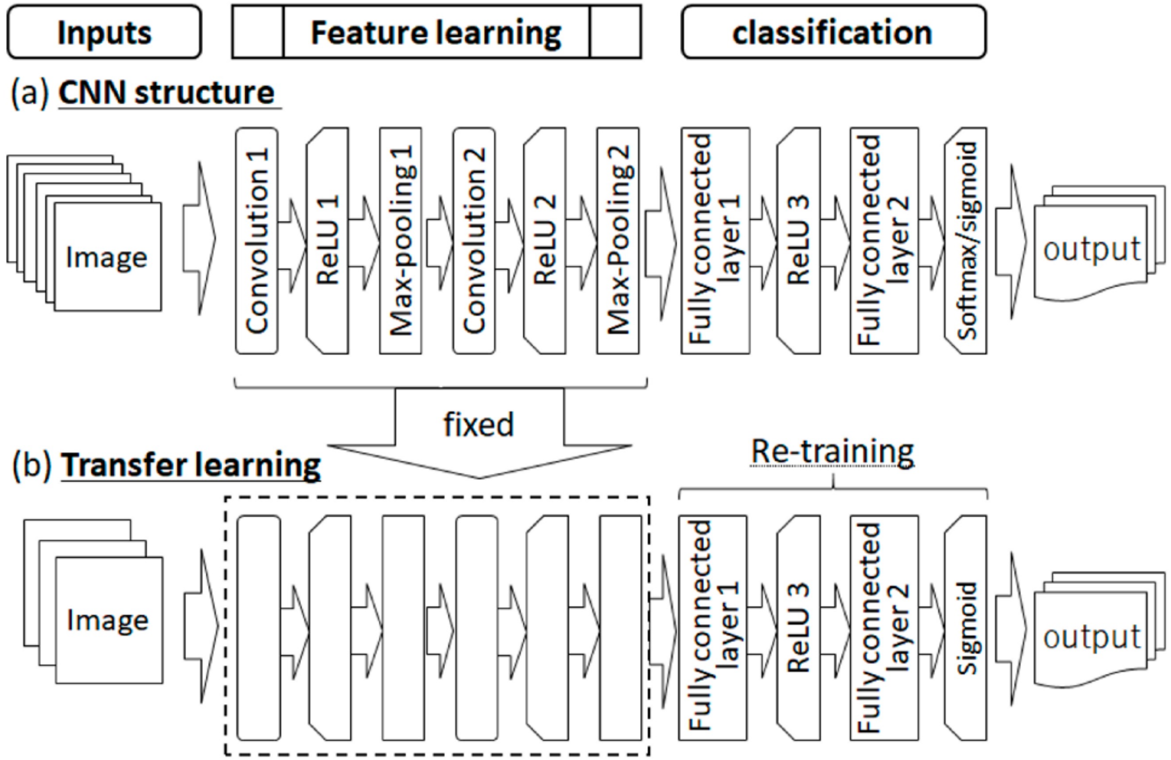




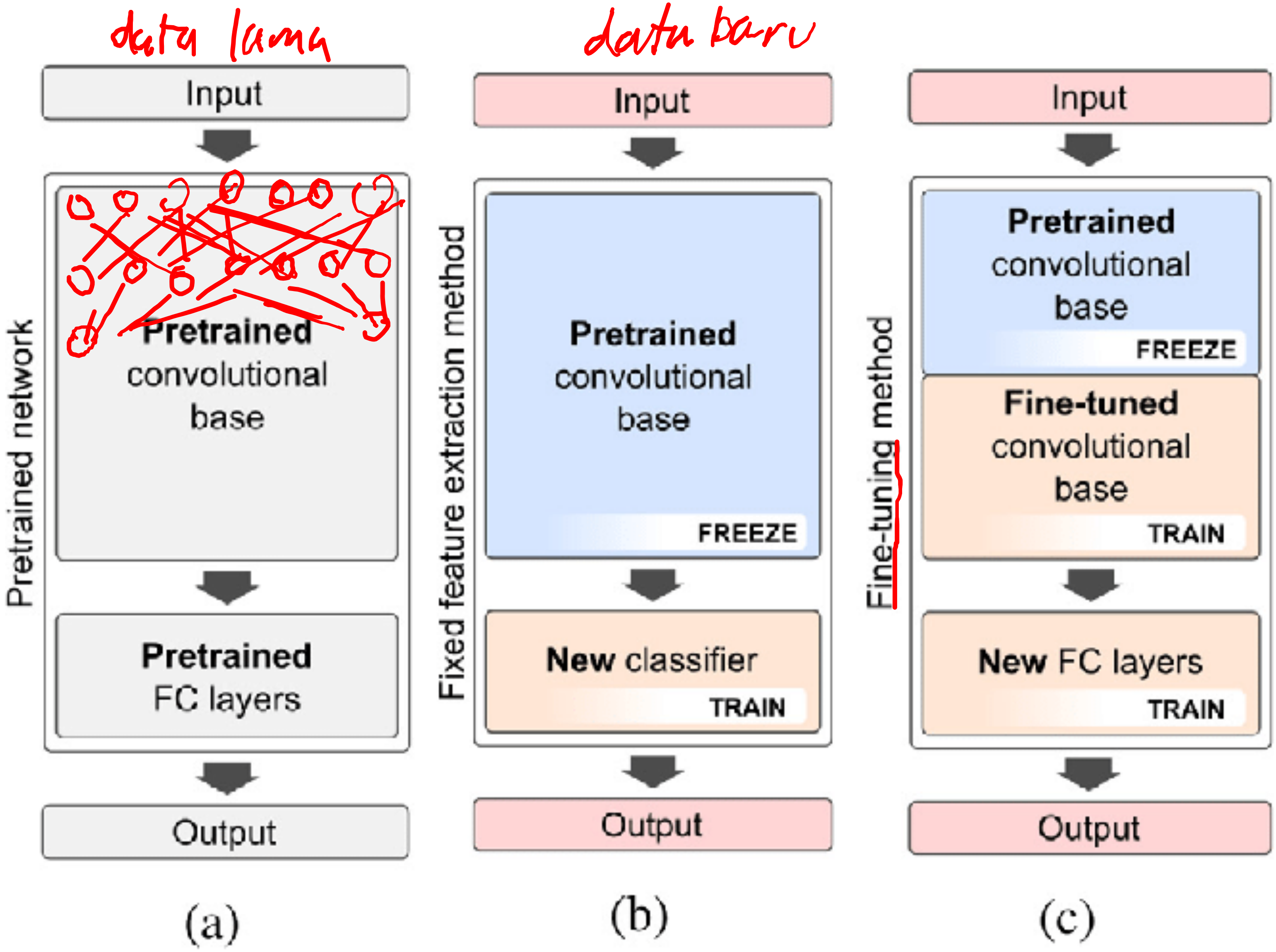
CALVIN INSTITUTE OF TECHNOLOGY

Transfer Learning

- Aplikasi A dan B mempunyai input yang sama
- Dipakai jika data untuk aplikasi A jauh lebih banyak daripada aplikasi B
- Fitur level rendah (low level) dari A dapat membantu proses pembelajaran B



Finetune


data lama
data baru
Input
Pretrained network
Pretrained convolutional base
Pretrained FC layers
Output
(a)
Input
Fixed feature extraction method
Pretrained convolutional base
FREEZE
New classifier
TRAIN
Output
(b)
Input
Fine-tuning method
Pretrained convolutional base
FREEZE
Fine-tuned convolutional base
TRAIN
New FC layers
TRAIN
Output
(c)

Kelemahan Deep Learning

- Beban komputasi besar
- Black box (proses berpikir tidak dapat ditelusuri)