Direktorat Jenderal Pendidikan Tinggi, Riset, dan, Teknologi
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi
Republik Indonesia

MICROCREDENTIAL: ASSOCIATE DATA SCIENTIST

01 November – 10 Desember 2021

Pertemuan ke-14

Membangun Model 5 (ANN Lanjutan, SOM, RNN, Deep Learning)

Profil Pengajar: Nama Lengkap dan Gelar Akademik

Contak Pengajar:

Ponsel:

xxxxxx

Email:

xxxxxxx

Jabatan Akademik: .....

Latar Belakang Pendidikan:

- S1: .....
- S2: .....
- S3: .....

Riwayat/Pengalaman Pekerjaan:

- Dosen
- Xxxx
- Xxxx
- Xxxx
- xxxx

KODE UNIT : **J.62DMI00.013.1**

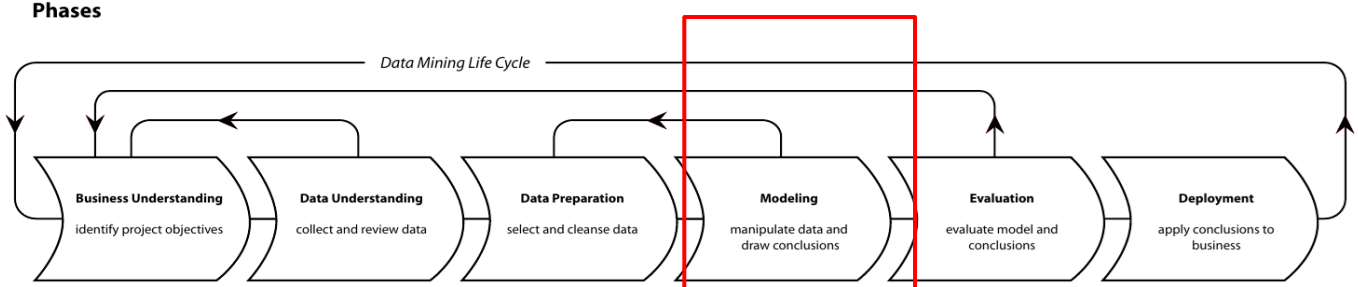
JUDUL UNIT : **Membangun Model**

DESKRIPSI UNIT : Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan dalam membangun model.

ELEMEN KOMPETENSI	KRITERIA UNJUK KERJA
1. Menyiapkan parameter model	1.1 Parameter-parameter yang sesuai dengan model diidentifikasi. 1.2 Nilai toleransi parameter evaluasi pengujian ditetapkan sesuai dengan tujuan teknis.
2. Menggunakan *tools* pemodelan	2.1 ***Tools* untuk membuat model** diidentifikasi sesuai dengan tujuan teknis *data science*. 2.2 Algoritma untuk teknik pemodelan yang ditentukan dibangun menggunakan *tools* yang dipilih. 2.3 Algoritma pemodelan dieksekusi sesuai dengan **skenario pengujian** dan *tools* untuk membuat model yang telah ditetapkan. 2.4 **Parameter model** algoritma dioptimasi untuk menghasilkan **nilai parameter evaluasi** yang sesuai dengan skenario pengujian.

1. Konteks variabel
   1.1 Termasuk di dalam skenario pengujian adalah komposisi *data training* dan *data testing*, cara pemilihan data *training* dan data testing seperti *percentage splitting, random selection,* atau *cross validation.*
   1.2 Yang dimaksud dengan parameter model di antaranya arsitektur model, banyaknya *layer* atau simpul, *learning rate* untuk *neural network,* nilai *k* untuk *k-means*, nilai *pruning* untuk *decision tree.*
   1.3 Nilai parameter evaluasi adalah nilai ambang batas *(threshold)* yang bisa diterima.
   1.4 Yang dimaksud dengan *tools* pemodelan di antaranya perangkat lunak *data science* di antaranya: *rapid miner, weka*, atau *development* untuk bahasa pemrograman tertentu seperti *python* atau R.

Phases



Determine Business Objectives
Background
Business Objectives
Business Success Criteria
(Log and Report Process)

Assess Situation
Inventory of Resources, Requirements, Assumptions, and Constraints
Risks and Contingencies
Terminology
Costs and Benefits
(Log and Report Process)

Determine Data Mining Goals
Data Mining Goals
Data Mining Success Criteria
(Log and Report Process)

Produce Project Plan
Project Plan
Initial Assessment of Tools and Techniques
(Log and Report Process)

Collect Initial Data
Initial Data Collection Report
(Log and Report Process)

Describe Data
Data Description Report
(Log and Report Process)

Explore Data
Data Exploration Report
(Log and Report Process)

Verify Data Quality
Data Quality Report
(Log and Report Process)

Data Set
Data Set Description
(Log and Report Process)

Select Data
Rationale for Inclusion/ Exclusion
(Log and Report Process)

Clean Data
Data Cleaning Report
(Log and Report Process)

Construct Data
Derived Attributes
Generated Records
(Log and Report Process)

Integrate Data
Merged Data
(Log and Report Process)

Format Data
Reformatted Data
(Log and Report Process)

Select Modeling Technique
Modeling Technique
Modeling Assumptions
(Log and Report Process)

Generate Test Design
Test Design
(Log and Report Process)

Build Model Parameter Settings
Models
Model Description
(Log and Report Process)

Assess Model
Model Assessment
Revised Parameter
(Log and Report Process)

Evaluate Results
Align Assessment of Data Mining Results with Business Success Criteria
(Log and Report Process)

Approved Models
Review Process
Review of Process
(Log and Report Process)

Determine Next Steps
List of Possible Actions
Decision
(Log and Report Process)

Plan Deployment
Deployment Plan
(Log and Report Process)

Plan Monitoring and Maintenance
Monitoring and Maintenance Plan
(Log and Report Process)

Produce Final Report
Final Report
Final Presentation
(Log and Report Process)

Review Project
Experience Documentation
(Log and Report Process)

Generic Tasks
Specialized Tasks
(Process Instances)

a visual guide to CRISP-DM methodology

SOURCE CRISP-DM 1.0
http://www.crisp-dm.org/download.htm
DESIGN Nicole Leaper
http://www.nicoleleaper.com

Course Definition

UK J.62DMI00.013.1 - Membangun Model (ANN)
a. Menyiapkan parameter model
b. Menggunakan tools pemodelan

Menjelaskan ANN lanjutan, SOM, RNN, dan Deep Learning

Learning Objective

Peserta mampu melakukan proses pemodelan Artificial Neural Network (lanjutan), SOM, RNN, dan Deep Learning

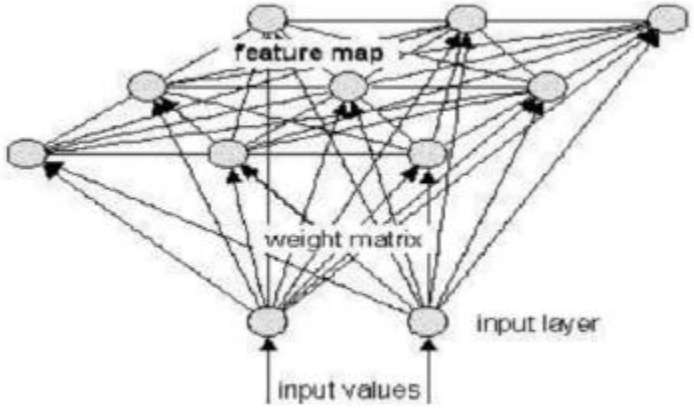
Jaringan Saraf Tiruan Kohonen/Self Organizing Maps (SOM)

Jaringan Saraf Tiruan/Self Organizing Maps (SOM)

- SOM merupakan suatu topologi JST yang menerapkan metode pembelajaran tanpa supervise (*unsupervised learning),* sehingga di dalam pembelajarannya tidak membutuhkan pola yang berfungsi sebagai target
- Dapat digunakan untuk pengelompokkan (*clustering*) yang dilakukan berdasarkan kemiripan fitur atau karakteristik data.

Jaringan Saraf Tiruan/Self Organizing Maps (SOM)

- SOM diawali dengan input nilai, kemudian didapatkan matriks bobot dari input tersebut yang akan membentuk jaringan untuk menentukan cluster.
- Dasar algoritma SOM adalah rangkaian unit pemrosesan array satu atau dua dimensi yang menyerupai jaringan dengan nilai *threshold,* dan ditandai adanya pengukuran jarak antar input (Kohonen, 1982)



Pengukuran Jarak SOM

- Antar neuron pada algoritma SOM diukur jarak kemiripannya (*similarity)* untuk menentukan apakah dapat berada dalam cluster yang sama atau berbeda.
- Penentuan kemiripan antar input vector dilakukan dengan pengukuran jarak. Beberapa pengukuran jarak yang digunakan yaitu:
  - *Euclidean Distance*
  - *Correlation*
  - *Direction Cosine*
  - *Block Distance*

Euclidean Distance adalah metode mengukur jarak untuk melihat kemiripan suatu data dengan data lainnya, sehingga dapat menentukan cluster dari data tersebut.

Algoritma SOM

1. Bobot setiap node diinisialisasi.
2. Sebuah vektor dipilih secara acak dari kumpulan data pelatihan.
3. Setiap node diperiksa untuk menghitung bobot mana yang paling mirip dengan vektor input. Node pemenang umumnya dikenal sebagai *Best Matching Unit* (BMU).
4. Kemudian lingkungan BMU dihitung. Jumlah *neighbors* berkurang dari waktu ke waktu.
5. Bobot yang menang dihargai dengan menjadi lebih seperti vektor sampel. *Neighbors* juga menjadi lebih seperti vektor sampel. Semakin dekat sebuah node ke BMU, semakin banyak bobotnya yang diubah dan semakin jauh *neighbors* dari BMU, semakin sedikit yang dipelajarinya.
6. Ulangi langkah 2 untuk N iterasi.

sumber:
https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

Jaringan Saraf Tiruan/Self Organizing Maps (SOM)

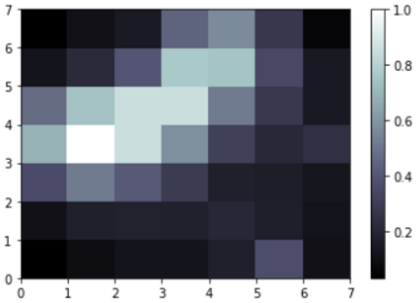
Best Matching Unit (BMU) adalah teknik yang menghitung jarak dari setiap bobot ke vektor sampel, dengan menjalankan semua vektor bobot. Berat dengan jarak terpendek adalah pemenangnya. Ada banyak cara untuk menentukan jarak, namun metode yang paling umum digunakan adalah *Euclidean Distance*, dan itulah yang digunakan dalam implementasi berikut.

sumber:
https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

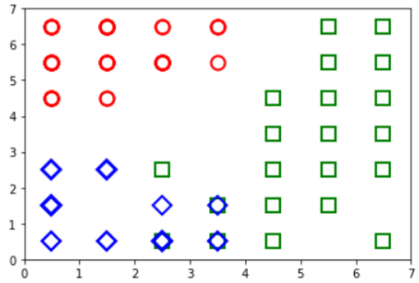
Jaringan Saraf Tiruan/Self Organizing Maps (SOM)

Implementation:

Pada bagian implementasi, ada berbagai Python libraries (minisom, sompy) yang dapat langsung digunakan untuk mengimplementasikan SOM. contoh implementasi pada dataset iris. Berikut adalah hasilnya:



In simpler terms, the darker parts represent clusters, while the lighter parts represent the division of the clusters.



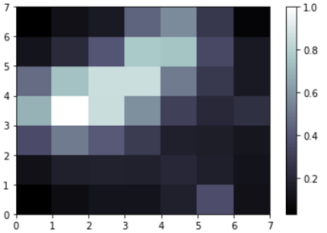
(Red Circles , Iris-setosa), (Green, Iris-versicolor) , (Blue, Iris-virginica)

https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

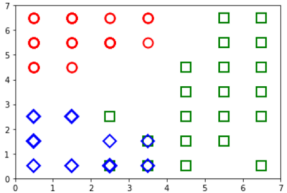
Jaringan Saraf Tiruan/Self Organizing Maps (SOM)

Inference:

Jika jarak rata-rata tinggi, maka bobot di sekitarnya sangat berbeda dan warna terang diberikan ke lokasi bobot. Jika jarak rata-rata rendah, warna yang lebih gelap ditetapkan. Peta yang dihasilkan menunjukkan bahwa konsentrasi kelompok spesies yang berbeda lebih dominan di tiga zona. Gambar pertama hanya memberitahu kita tentang di mana kepadatan spesies lebih besar (daerah yang lebih gelap) atau lebih kecil (daerah yang lebih terang). Visualisasi kedua memberitahu kita bagaimana mereka secara khusus dikelompokkan.



In simpler terms, the darker parts represent clusters, while the lighter parts represent the division of the clusters.



(Red Circles , Iris-setosa), (Green, Iris-versicolor) , (Blue, Iris-virginica)

sumber:
https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

Jaringan Saraf Tiruan/Self Organizing Maps (SOM)

Kontra:

- Membangun model generatif untuk data, yaitu model tidak mengerti bagaimana data dibuat.
- Tidak *gently* saat menggunakan data kategorikal, bahkan lebih buruk untuk data tipe campuran.
- Waktu untuk menyiapkan model lambat, sulit untuk dilatih melawan data yang berkembang perlahan

sumber:
https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

SOM

Seperti yang disebutkan sebelumnya, jenis jaringan saraf tiruan (*artificial neural network*) tanpa pengawasan, menggunakan pembelajaran kompetitif untuk memperbarui bobotnya.

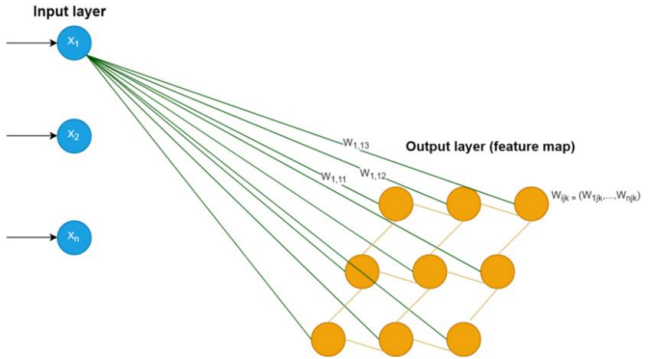
Pembelajaran kompetitif didasarkan pada tiga proses:

- Kompetisi
- Kerja sama
- Adaptasi

sumber:
https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065

SOM

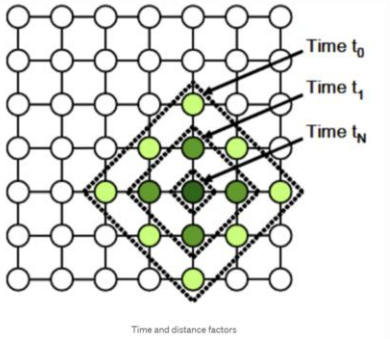
- Seperti yang dikatakan sebelumnya setiap neuron dalam SOM diberi vektor bobot dengan dimensi yang sama dengan ruang input.
- Pada contoh di bawah ini, di setiap neuron dari lapisan keluaran akan memiliki sebuah vektor dengan dimensi n.
- Menghitung jarak antara setiap neuron (neuron dari lapisan output) dan data input, dan neuron dengan jarak terendah akan menjadi pemenang kompetisi.
- Metrik Euclidean biasanya digunakan untuk mengukur jarak.



sumber:
https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065

SOM

- Memperbarui vektor neuron pemenang dalam proses akhir (adaptasi) tetapi itu bukan satu-satunya, juga *neighbors* akan diperbarui.
- Bagaimana memilih *neighbors*?
- Untuk memilih *neighbors* menggunakan fungsi kernel lingkungan, fungsi ini tergantung pada dua faktor: waktu (waktu bertambah setiap data input baru) dan jarak antara neuron pemenang dan neuron lainnya (seberapa jauh neuron dari neuron pemenang).
- Gambar di bawah ini menunjukkan bagaimana neuron pemenang (Yang paling hijau di tengah) *neighbors* dipilih tergantung pada jarak dan faktor waktu.

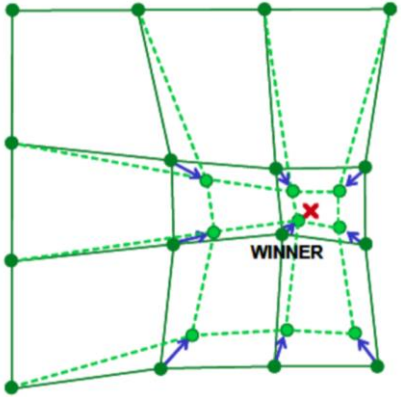


sumber:
https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065

SOM

Adaptasi

Setelah memilih neuron pemenang dan *neighbors*, menghitung pembaruan neuron. Neuron-neuron terpilih tersebut akan diupdate tetapi tidak update yang sama, semakin jauh jarak antara neuron dan data input semakin berkurang disesuaikan seperti terlihat pada gambar di bawah ini :



neurons of the output layer update

sumber:
https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065

SOM

Satu hal yang menarik dari SOM adalah bahwa sistemnya didasarkan pada pembelajaran kompetitif. Neuron (atau node) bersaing untuk memutuskan mana yang akan merespon (diaktifkan) melalui serangkaian input dan neuron ini disebut pemenang. SOM dapat diimplementasikan memiliki koneksi penghambatan lateral, kapasitas neuron pemenang untuk mengurangi aktivitas *neighbouring* dengan memberikan umpan balik negatif kepada mereka. Konsep lain yang digarap SOM adalah peta topografi. Informasi yang disimpan dari input diwakili oleh beberapa neuron *neighbouring* dan mereka dapat berinteraksi dengan koneksi pendek. Neuron keluaran dari peta topografi adalah fitur dari data masukan.

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

SOM

Bagaimana SOM bekerja?

Titik-titik di ruang input memiliki titik koresponden di ruang output. Di *Kohonen Networks*, semacam SOM, ada satu lapisan dengan dua dimensi dan titik-titik input terhubung sepenuhnya dengan neuron pada lapisan ini.

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

SOM

Pada awal proses *Self Organization*, bobot diinisialisasi dengan nilai acak. Setelah itu, untuk kompetisi, semua neuron akan menghitung fungsi diskriminasi di bawah ini, atas fitur input. Neuron dengan nilai terkecil akan menjadi pemenangnya.

Fungsi ini akan menunjukkan neuron apa yang paling mirip dengan vektor input.

$$d_j(\mathbf{x}) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

D = dimension of the inputs; x = the inputs; w = the weights

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

SOM

Ketika neuron *fired*, *neighbors* akan lebih *excited* daripada neuron jauh. Proses ini disebut lingkungan topologi dan dihitung sebagai berikut:

$$T_{j,I(\mathbf{x})} = \exp(-S^2_{j,I(\mathbf{x})} / 2\sigma^2)$$

Dimana S adalah jarak lateral antara neuron, I(x) adalah indeks dari neuron pemenang dan adalah jumlah *neighbors* dan jumlah ini menurun seiring waktu. Jumlah lingkungan topologi akan berkurang, cenderung nol karena jarak ke pemenang meningkat.

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

SOM

Dengan t adalah jumlah epoch dan η(t) laju *learning* pada saat itu, bobot diperbarui dengan rumus ini:

$$\Delta w_{ji} = \eta(t) \,.\, T_{j,I(\mathbf{x})}(t) \,.\, (x_i - w_{ji})$$

Seperti yang dilihat, bobot dipindahkan sesuai dengan lingkungan topologi, menyebabkan neuron jauh memiliki pembaruan kecil. Ini akan menghasilkan efek seperti neuron pemenang menarik neuron lainnya.

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

Hands-On SOM

Import library

```
from minisom import
MiniSom

import numpy as np
import matplotlib.pyplot
as plt
%matplotlib inline
```

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

Hands-On SOM

Read the image

```
img = plt.imread('tree.jpg')
```

Reshaping the pixels matrix

```
pixels = np.reshape(img,
(img.shape[0]*img.shape[1], 3)) /
255.
```

```
array([[0.81960784, 0.76862745, 0.80392157],
       [0.81960784, 0.76862745, 0.80392157],
       [0.81960784, 0.76862745, 0.80392157],
       ...,
       [0.21568627, 0.2745098 , 0.04313725],
       [0.20392157, 0.2627451 , 0.03137255],
       [0.20392157, 0.2627451 , 0.03137255]])
```

sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

Hands-On SOM

SOM initialization and training

```
print('training...')
som = MiniSom(2, 3, 3, sigma=1.,
              learning_rate=0.2, neighborhood_function='bubble')  # 3x3 = 9 final colors
som.random_weights_init(pixels)
starting_weights = som.get_weights().copy()  # saving the starting weights
som.train(pixels, 10000, random_order=True, verbose=True)

print('quantization...')
qnt = som.quantization(pixels)  # quantize each pixels of the image
print('building new image...')
clustered = np.zeros(img.shape)
for i, q in enumerate(qnt):  # place the quantized values into a new image
    clustered[np.unravel_index(i, shape=(img.shape[0], img.shape[1]))] = q
print('done.')
```

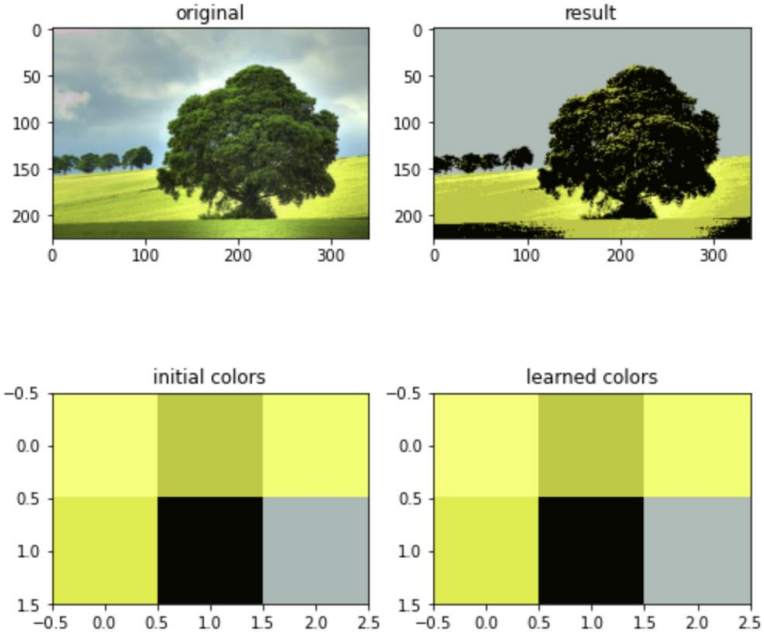
```
training...
 [  4230 / 10000 ]  42% - 0:00:00 left
 [  6272 / 10000 ]  63% - 0:00:00 left
 [  8446 / 10000 ]  84% - 0:00:00 left
 [ 10000 / 10000 ] 100% - 0:00:00 left
 quantization error: 0.13033700751279656
quantization...

building new image...
done.
```


sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

Hands-On SOM

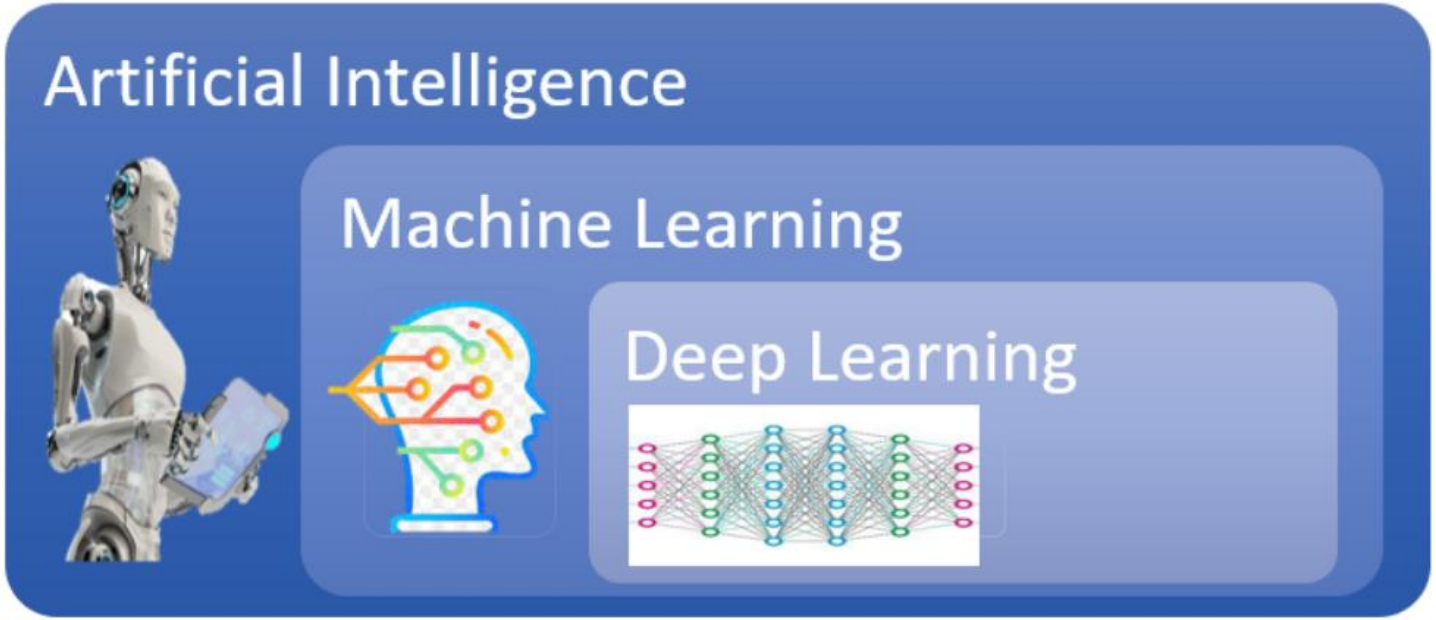
Show the result

```
plt.figure(figsize=(7, 7))
plt.figure(1)
plt.subplot(221)
plt.title('original')
plt.imshow(img)
plt.subplot(222)
plt.title('result')
plt.imshow(clustered)

plt.subplot(223)
plt.title('initial colors')
plt.imshow(starting_weights, interpolation='none')
plt.subplot(224)
plt.title('learned colors')
plt.imshow(som.get_weights(), interpolation='none')

plt.tight_layout()
plt.savefig('hasil/som_color_quantization.png')
plt.show()
```



sumber:
https://medium.com/neuronio/discovering-som-an-unsupervised-neural-network-12e787f38f9

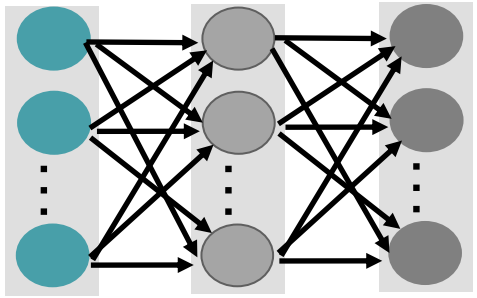
Pengantar Deep Learning



Machine Learning (Konvensional)

Inpu
t

Ekstraksi Fitur



Klasifikasi

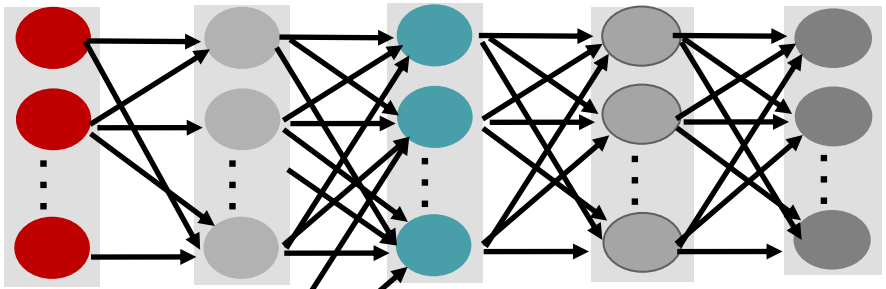
Mobil

Motor

Outp
ut

Deep Learning

Inpu
t



Ekstraksi Fitur + Klasifikasi

Mobil

Motor

Outpu
t

Pengantar Deep Learning

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

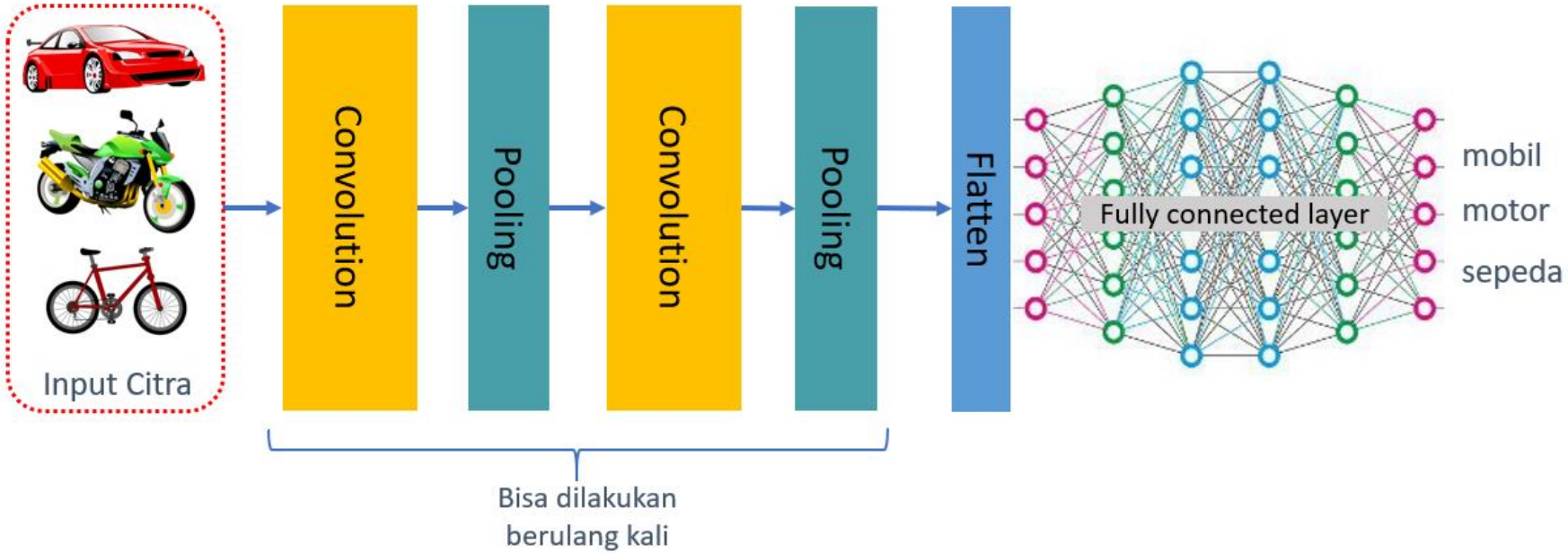
Kelemahan pendekatan konvensional:

- 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 layer*



Bagian Arsitektur CNN

1. **Convolutional Layer**

Menggunakan operasi konvolusi dari teori pengolahan citra.
Berperan untuk menghasilkan "*feature image/map*,"gambar yang berisi fitur penting dari gambar input.

2. **Pooling Layer**

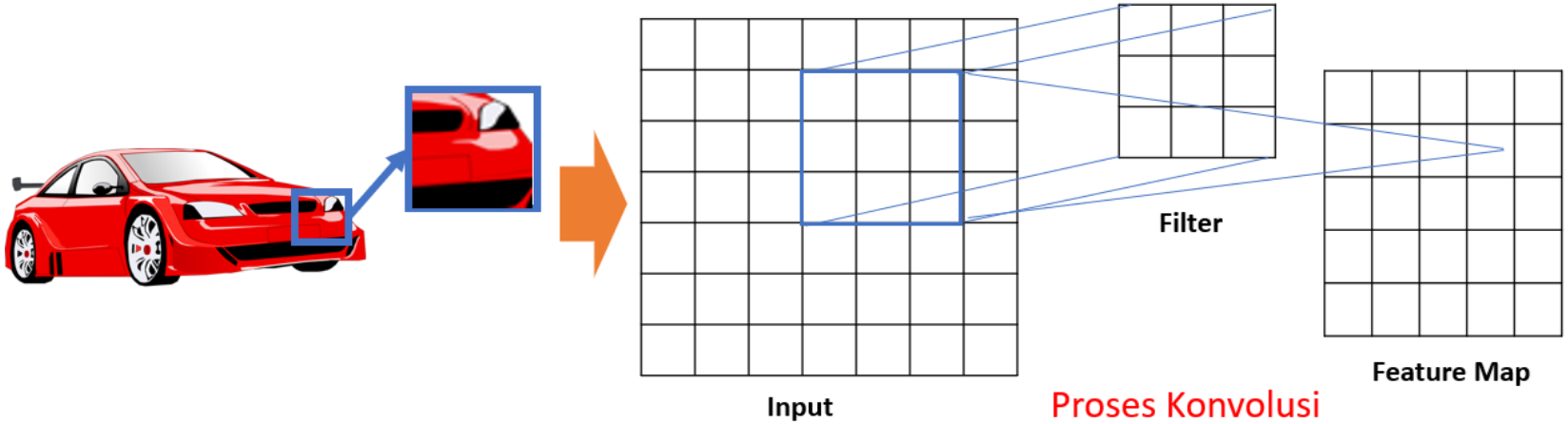
Berperan untuk memperkecil dimensi *feature image.*
Jenis: Max-pooling, Average pooling, dll.

3. **Fully-connected Layer**

Multi Layer Perceptron biasa
Berperan untuk menghasilkan output klasifikasi akhir

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*

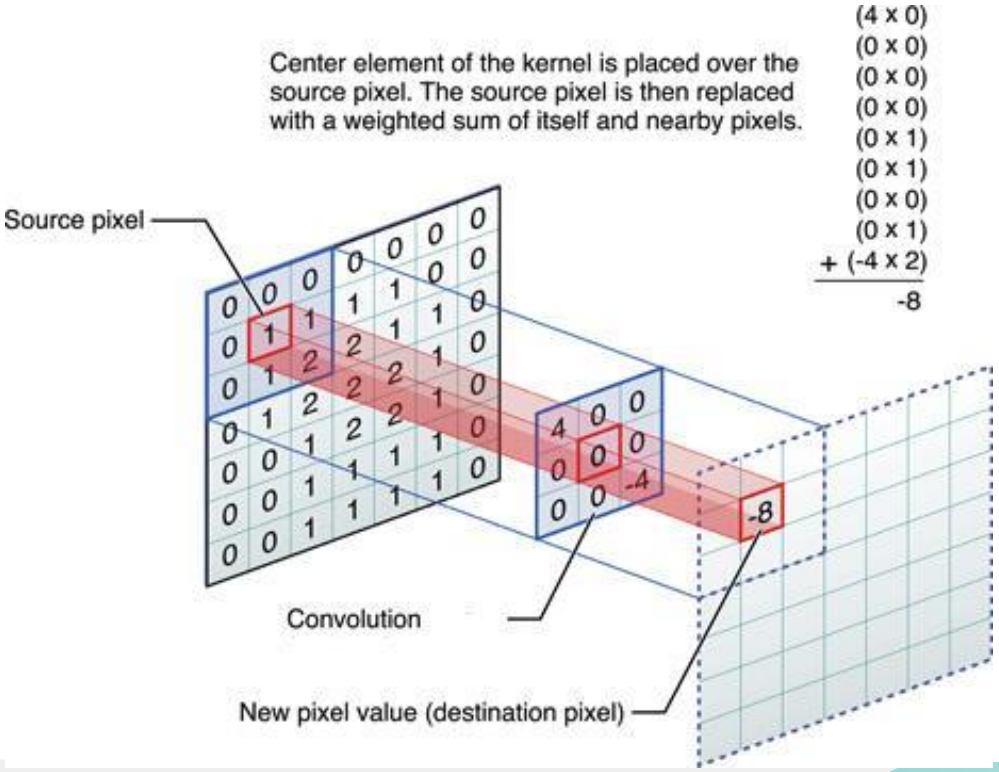


Convolutional Layer

Dalam dunia Ilmu Komputer, konvolusi erat kaitannya dengan bidang ilmu *Image Processing*.

Konvolusi merupakan sebuah operasi antar dua buah matrix…
Matrix Gambar (input)
Matrix Kernel
… sedemikian sehingga menghasilkan sebuah matrix yang berisi fitur-fitur penting dari input.

Convolutional Layer



Convolutional Layer

- Proses konvolusi citra dengan filter dilakukan sliding filter mulai dari kiri atas dari matrik 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

Convolutional Layer

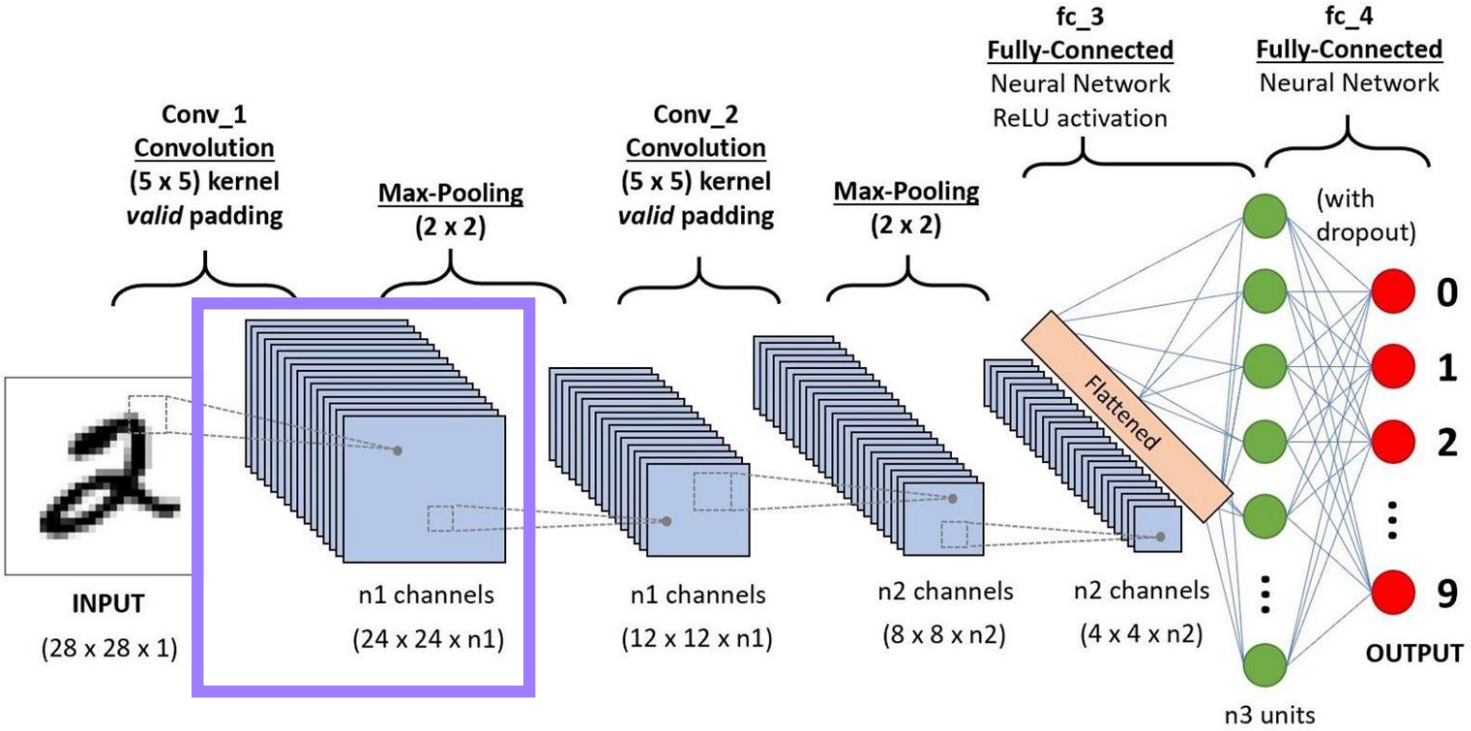
1. Jadi, layer konvolusi itu gunanya buat apa?

Mencari fitur-fitur penting sebuah gambar.
fitur memberikan nilai informasi yang jauh lebih besar dibanding gambar input itu sendiri.

2. Kernel konvolusi kan ada banyak jenisnya, pakai yang mana?

- Bukan kita yang menentukan apa kernelnya, tapi CNN yang akan belajar dan mencari kernel apa yang tepat!
- Dengan kata lain, nilai-nilai dalam kernel itulah yang dipelajari oleh CNN.
- Nilai dalam kernel = Weight yang harus dipelajari.

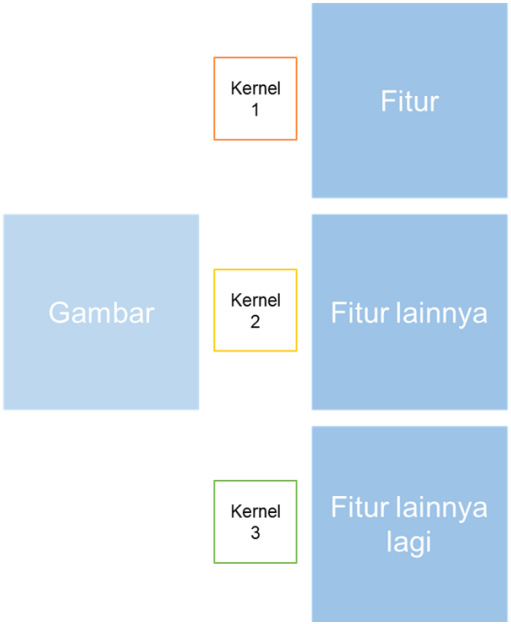
Convolutional Layer

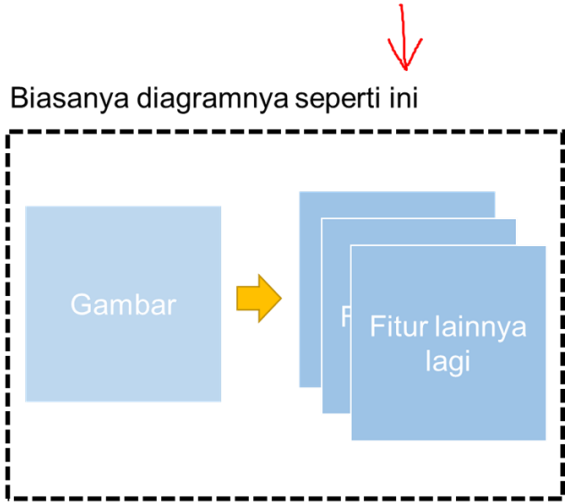


Feature-map ada banyak

Convolutional Layer

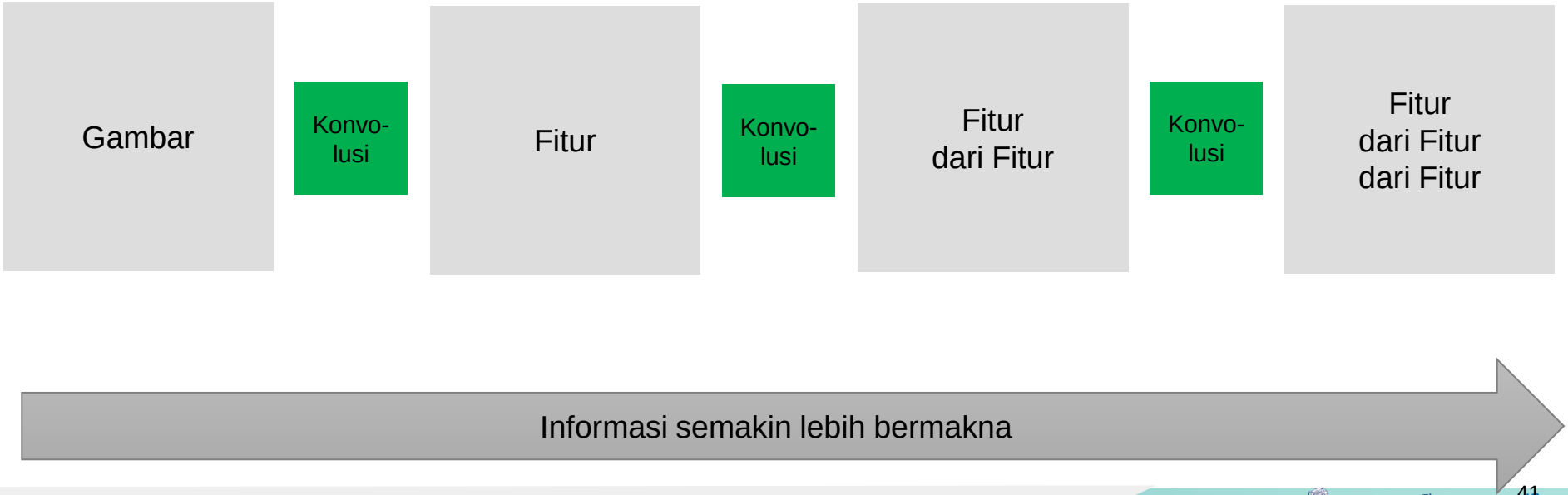
Banyak Kernel = Banyak Jenis Fitur





Convolutional Layer

Tujuan Layer Konvolusi ditumpuk



Pooling Layer

Merupakan layer yang berperan untuk mereduksi dimensionalitas output dari layer sebelumnya.
Membuat ukuran *feature image* menjadi lebih kecil.

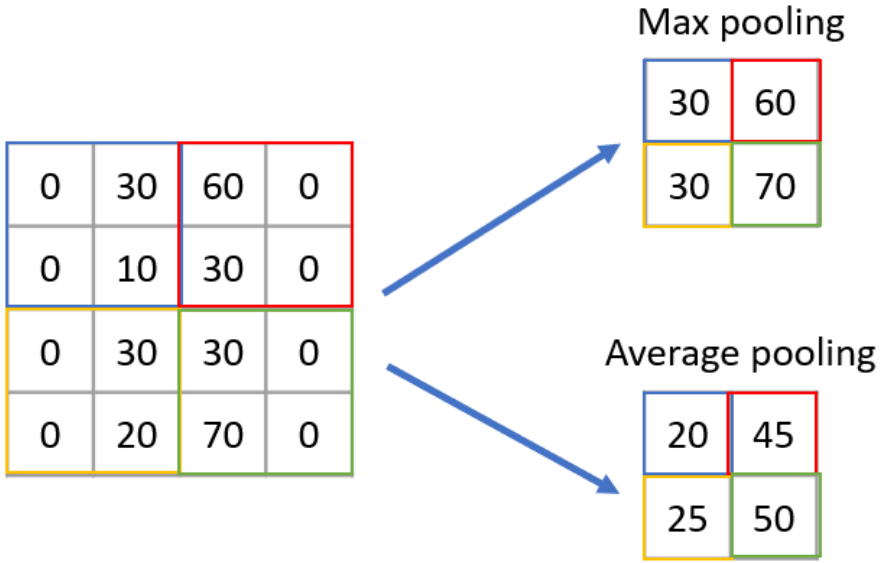
Mengapa kita membutuhkan ini?

1. Curse of dimensionality
   - Ingat problem awal kita sebelum masuk ke CNN.
   - Makin besar dimensi = Network makin sulit dilatih.

2. Informasi gambar tidak bergantung pada posisi ruang.
   - Gambar kucing ya gambar kucing, sekalipun kepalanya lagi miring, atau posisi hidungnya ada diatas gambar.
   - *Feature image* di-pool = Informasi “kucing” menjadi padat dan singular.
   - *Translational Invariance.*

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*



Pooling Layer

Merupakan layer yang berperan untuk mereduksi dimensionalitas output dari layer sebelumnya.
Membuat ukuran *feature image* menjadi lebih kecil.

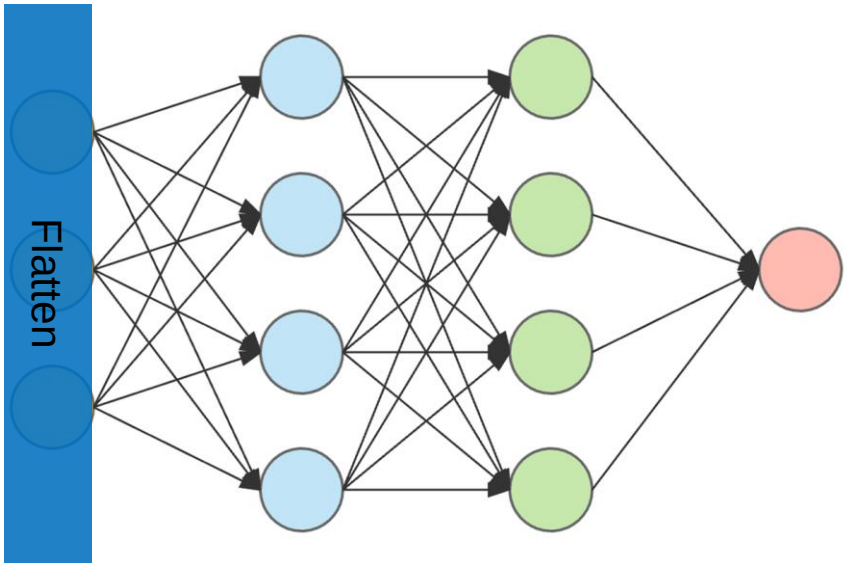
Mengapa kita membutuhkan ini?

1. Curse of dimensionality
   - Ingat problem awal kita sebelum masuk ke CNN.
   - Makin besar dimensi = Network makin sulit dilatih.

2. Informasi gambar tidak bergantung pada posisi ruang.
   - Gambar kucing ya gambar kucing, sekalipun kepalanya lagi miring, atau posisi hidungnya ada diatas gambar.
   - *Feature image* di-pool = Informasi “kucing” menjadi padat dan singular.
   - *Translational Invariance.*

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*



Fully Connected Layer

Hanyalah sebuah Multi-Layer Perceptron.

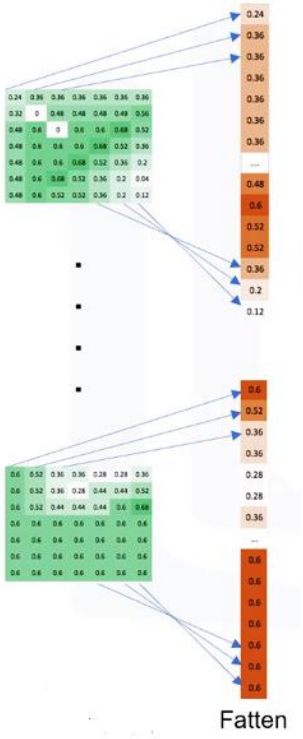
- Bernama "*fully-connected*" karena setiap neuron di layer satu terhubung ke seluruh neuron di layer lain.
- Berbeda dengan apa yang dilakukan di Convolution dan Pooling Layer.
- Berperan untuk melakukan klasifikasi akhir.

Disetiap layer *fully-connected* activation function yang digunakan bebas, Kecuali, akhir layer *fully-connected*.

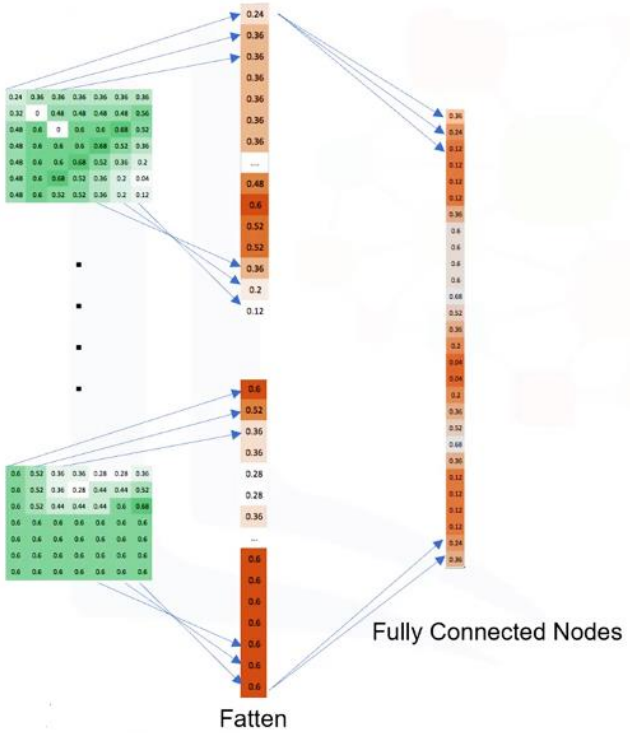
Layer akhir bertugas memberikan probabilitas klasifikasi.

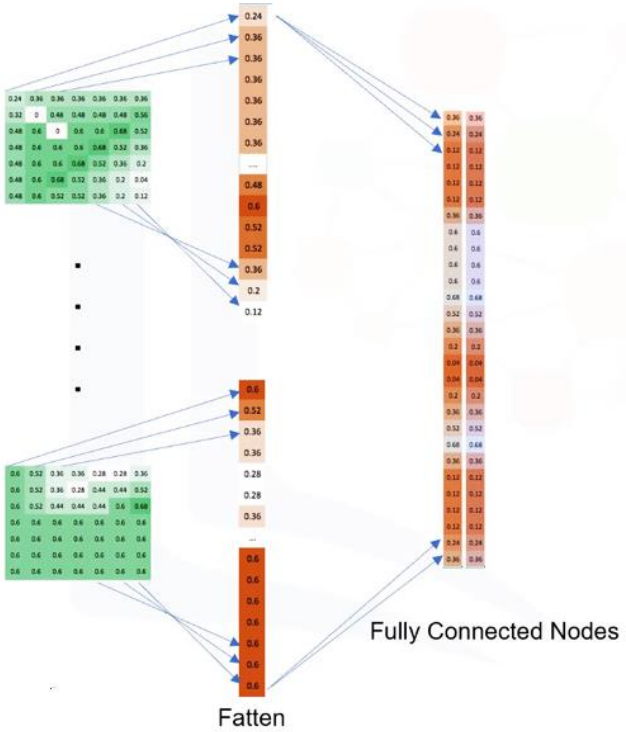
Softmax Function.

Proses Fully-Connected Layer

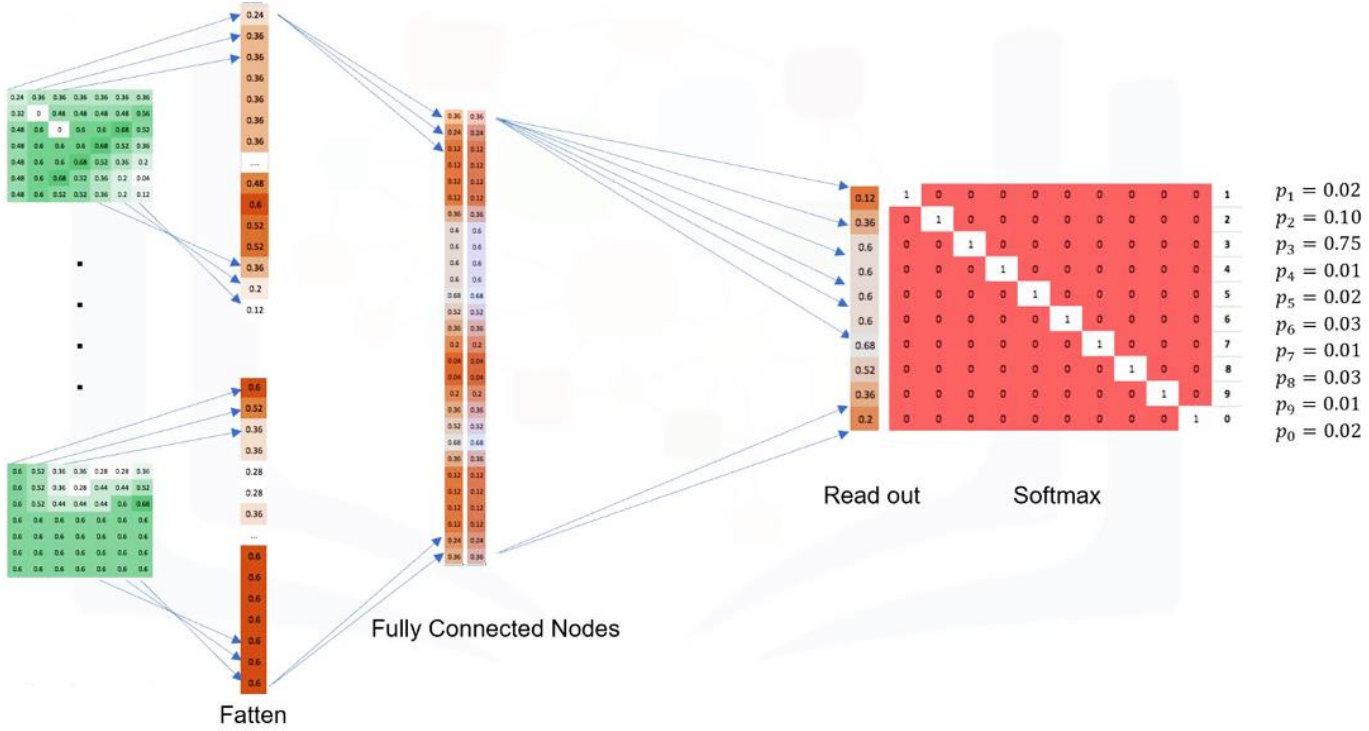


Proses Fully-Connected Layer

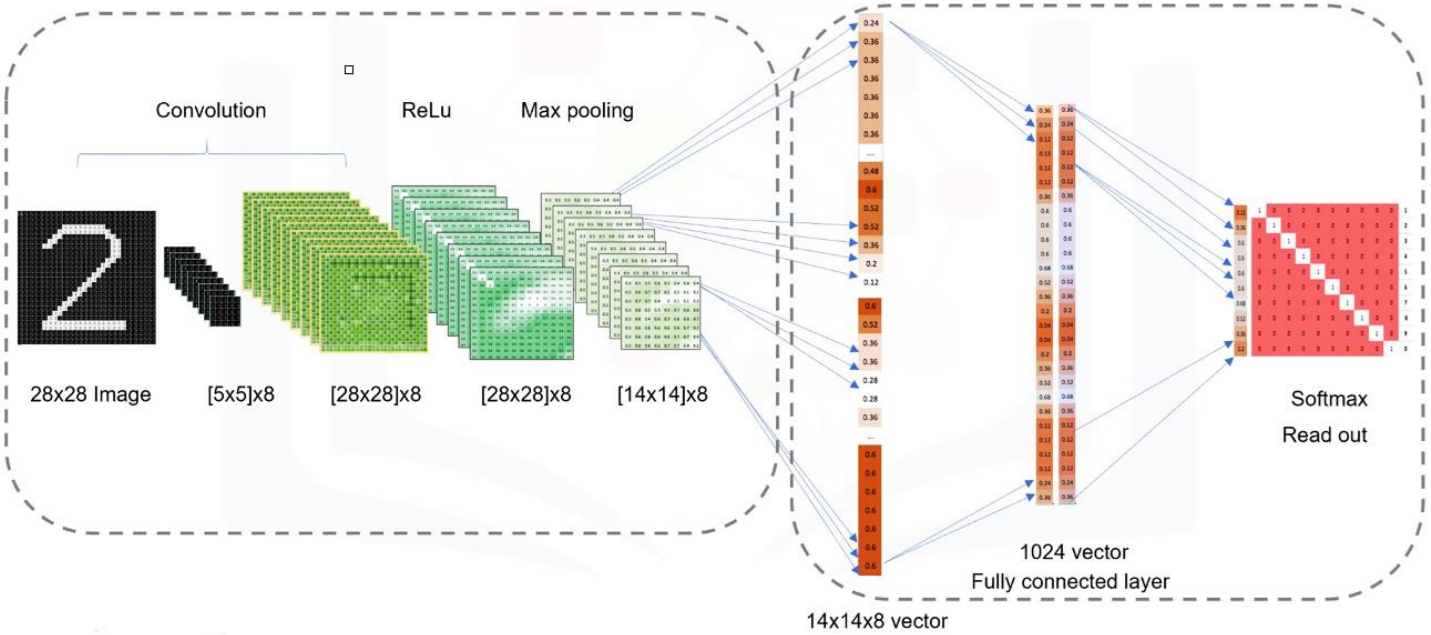



Fully Connected Nodes
Fatten

Proses Fully-Connected Layer

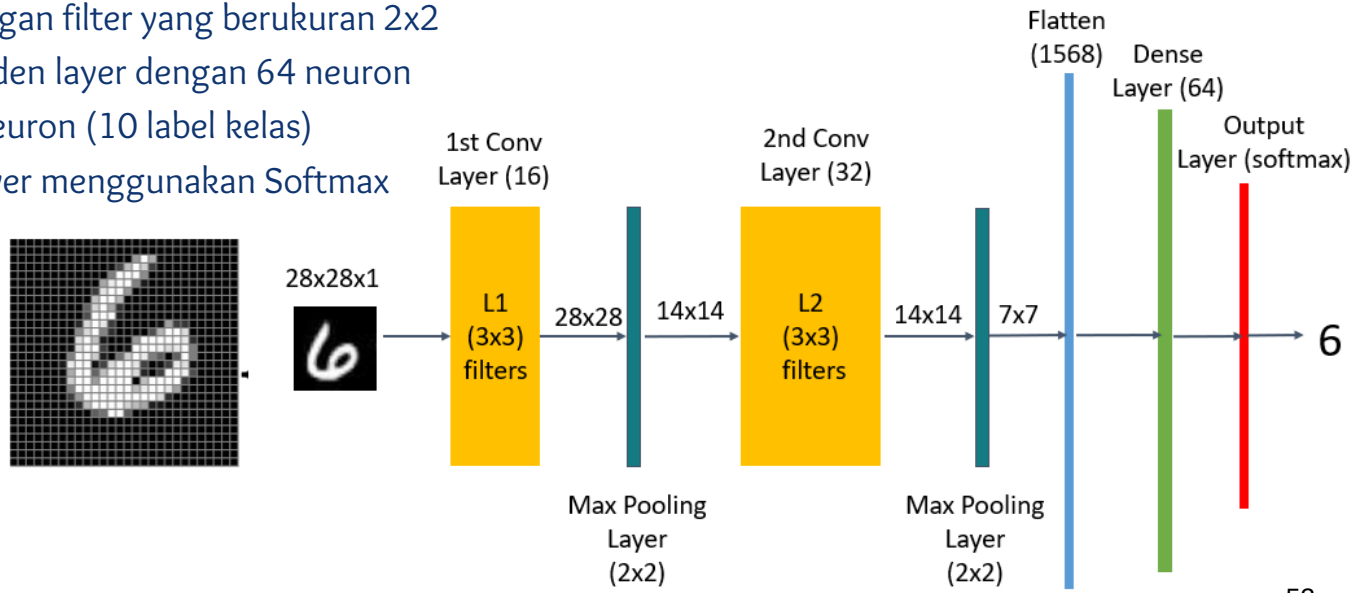


Arsitektur CNN Full Version



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()
```

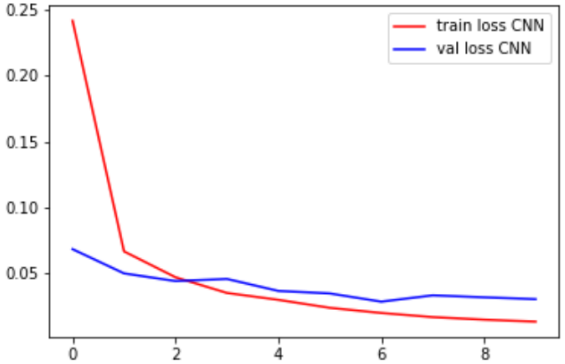
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160
_________________________________________________________________
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
[0.03035075031220913, 0.9897000193595886]
```



Contoh implementasi arsitektur CNN pada pengenalan angka

3. Load Model CNN dan Prediction

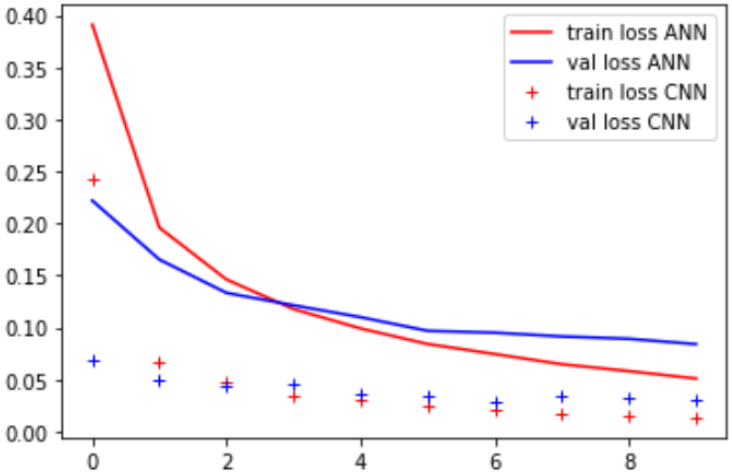
```
import numpy as np
from keras.models import load_model

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

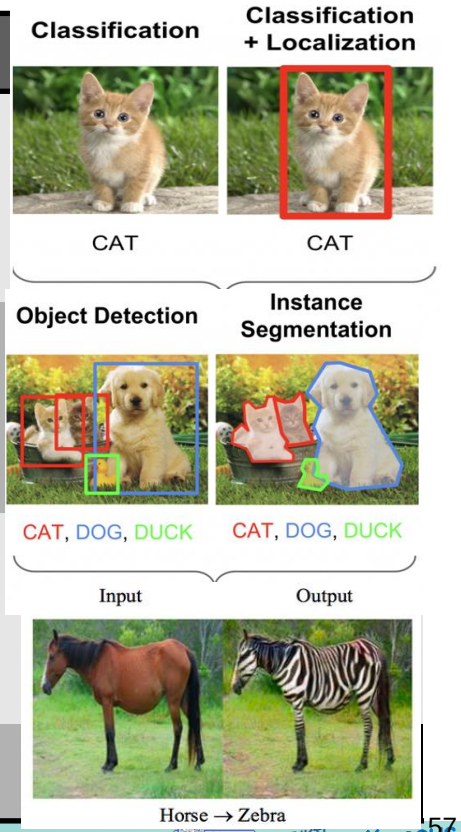
```
import matplotlib.pyplot as plt

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



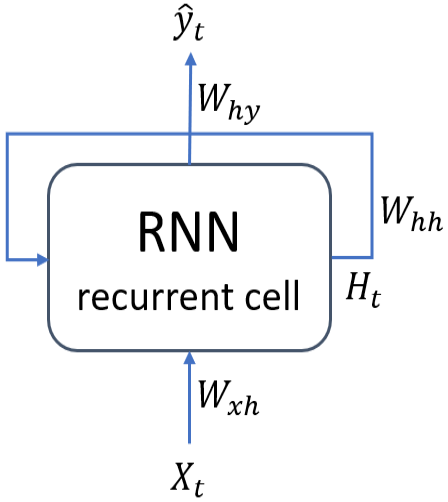
Varian dari Arsitektur CNN dan Tipe Apikasinya

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)



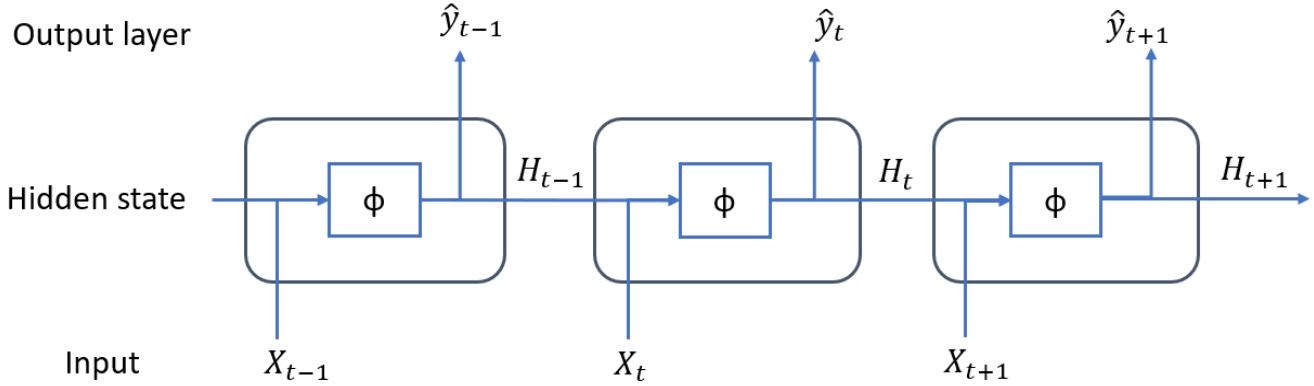
Recurrent Neural Network

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

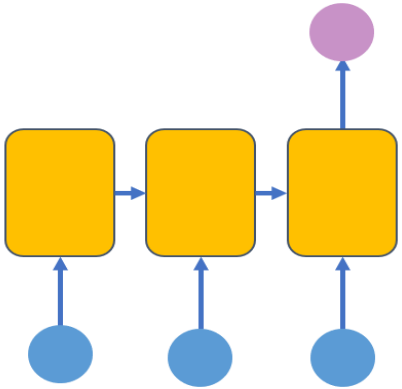


$$\hat{y}_t = H_t W_{hy} + b_y$$

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

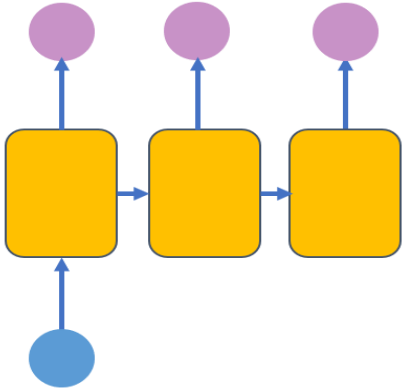


Tipe arsitektur RNN dan aplikasinya



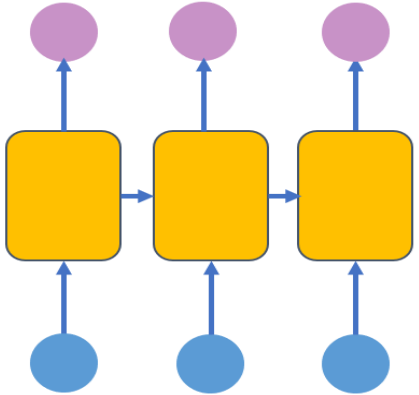
Many to One

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



One to Many

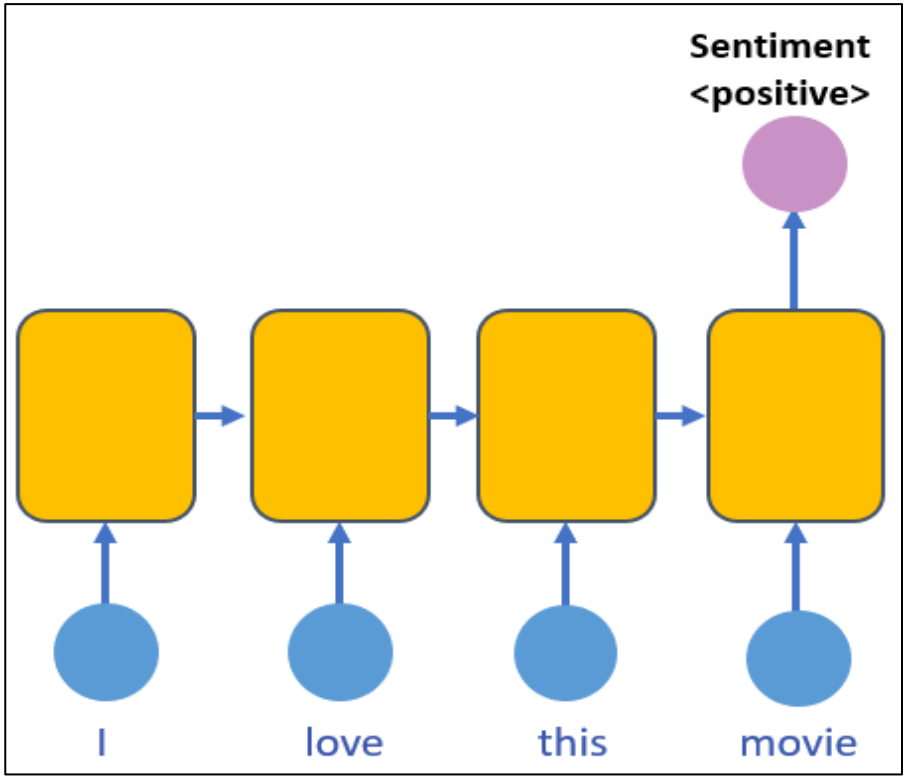
Applications:
- Image captioning
- Text generation

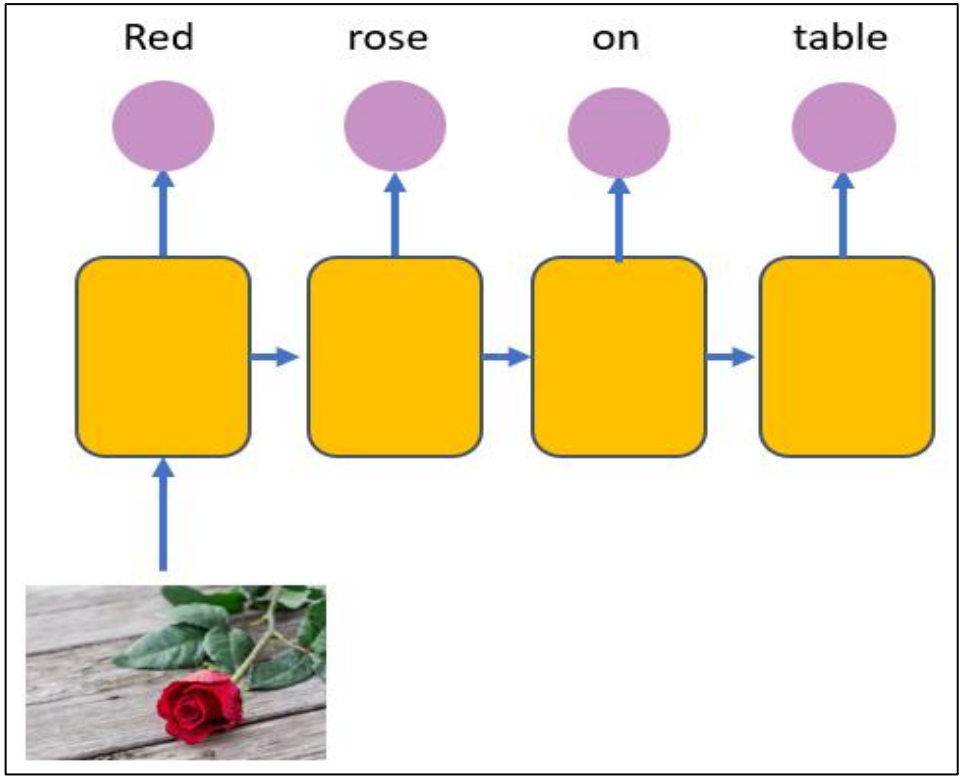


Many to Many

Applications:
- Translation
- Forecasting
- Chatbot
- Music generation

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

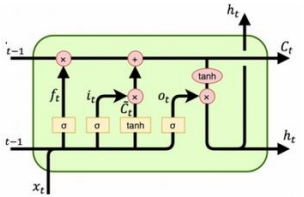


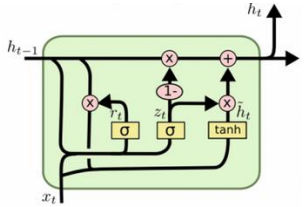


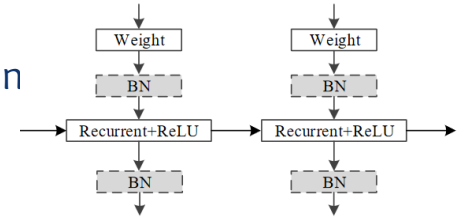
http://dprogrammer.org/rnn-lstm-gru

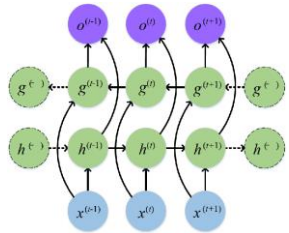
Varian Arsitektur RNN

- **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 membuat 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*

Referensi

- 虞台文, Feed-Forward Neural Networks, Course slides presentation
- Andrew Ng, Machine Learning, Course slides presentation
- Michael Negnevitsky, Artificial Intelligence : A Guide to Intelligent Systems, Second Edition, Addision Wesley, 2005.
- Hung-yi Lee, Deep Learning Tutorial
- Alexander Amini, Intro to Deep Learning, MIT 6.S191, 2021

Pembuat Modul

Dr. Eng. Chastine Fatichah, S.Kom, M.Kom
Institut Teknologi Sepuluh Nopember
email: chastine@if.its.ac.id

Quiz / Tugas

- Quiz dapat diakses melalui https://spadadikti.id/

Terima kasih