MODUL PRAKTIKUM

MATA KULIAH DATA MINING

PERTEMUAN 10

SEMESTER GENAP

TAHUN AJARAN 2024/2025

Disusun oleh:

Dwi Welly Sukma Nirad S.Kom, M.T

Aina Hubby Aziira M.Eng

M.Faiz Al-Dzikro

Muhammad Fariz

DEPARTEMEN SISTEM INFORMASI

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

IDENTITAS PRAKTIKUM

IDENTITAS MATA KULIAH

Kode mata kuliah	JSI62122
Nama mata kuliah	Data Mining
CPMK yang dibebankan pada praktikum	CPMK-3, CPMK-4 Mahasiswa mampu memahami teknik *deep learning* dalam data mining (CP-2).
Materi Praktikum Pertemuan 10	Konsep Deep Learning
	Konsep CNN
	Arsitektur CNN
	Algoritma Proses CNN
	Implementasi CNN

IDENTITAS DOSEN DAN ASISTEN MAHASISWA

Nama Dosen Pengampu	1. Dwi Welly Sukma Nirad S.Kom, M.T 2. Aina Hubby Aziira M.Eng
Nama Asisten Mahasiswa (Kelas A)	1. 2211523034 - Muhammad Fariz 2. 2211521012 - Rizka Kurnia Illahi 3. 2211521010 - Dhiya Gustita Aqila 4. 2211522013 - Benni Putra Chaniago 5. 2211521017 - Ghina Anfasha Nurhadi 6. 2211523022 - Daffa Agustian Saadi

	7. 2211521007 - Annisa Nurul Hakim 8. 2211522021 - Rifqi Asverian Putra 9. 2211521009 - Miftahul Khaira 10. 2211521015- Nurul Afani 11. 2211523028 - M.Faiz Al-Dzikro
Nama Asisten Mahasiswa (Kelas B)	1. 2211523034 - Muhammad Fariz 2. 2211521012 - Rizka Kurnia Illahi 3. 2211521010 - Dhiya Gustita Aqila 4. 2211522013 - Benni Putra Chaniago 5. 2211521017 - Ghina Anfasha Nurhadi 6. 2211523022 - Daffa Agustian Saadi 7. 2211521007 - Annisa Nurul Hakim 8. 2211522021 - Rifqi Asverian Putra 9. 2211521009 - Miftahul Khaira 10. 2211521015- Nurul Afani 11. 2211523028 - M.Faiz Al-Dzikro

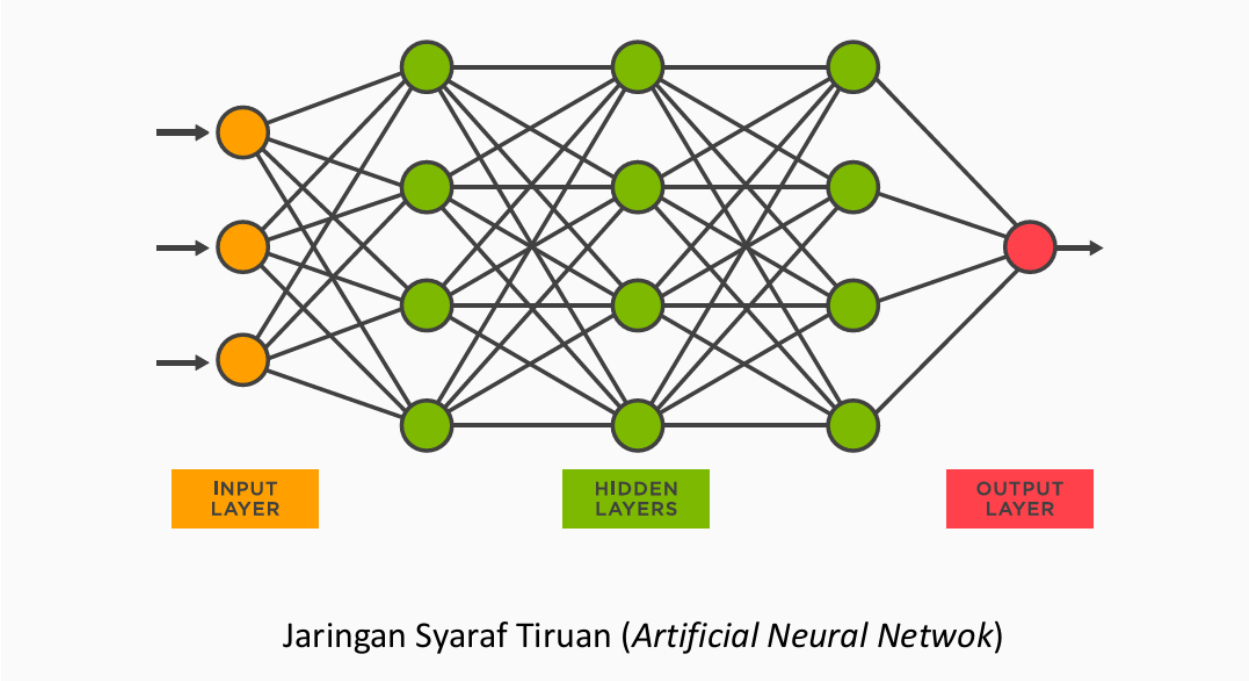
DAFTAR ISI

DEEP LEARNING CNN

A. KONSEP DEEP LEARNING

Pembelajaran mendalam (deep learning) adalah jenis pembelajaran mesin (machine learning) yang dalam hal ini model belajar untuk melakukan tugas klasifikasi langsung dari gambar, teks, atau suara.

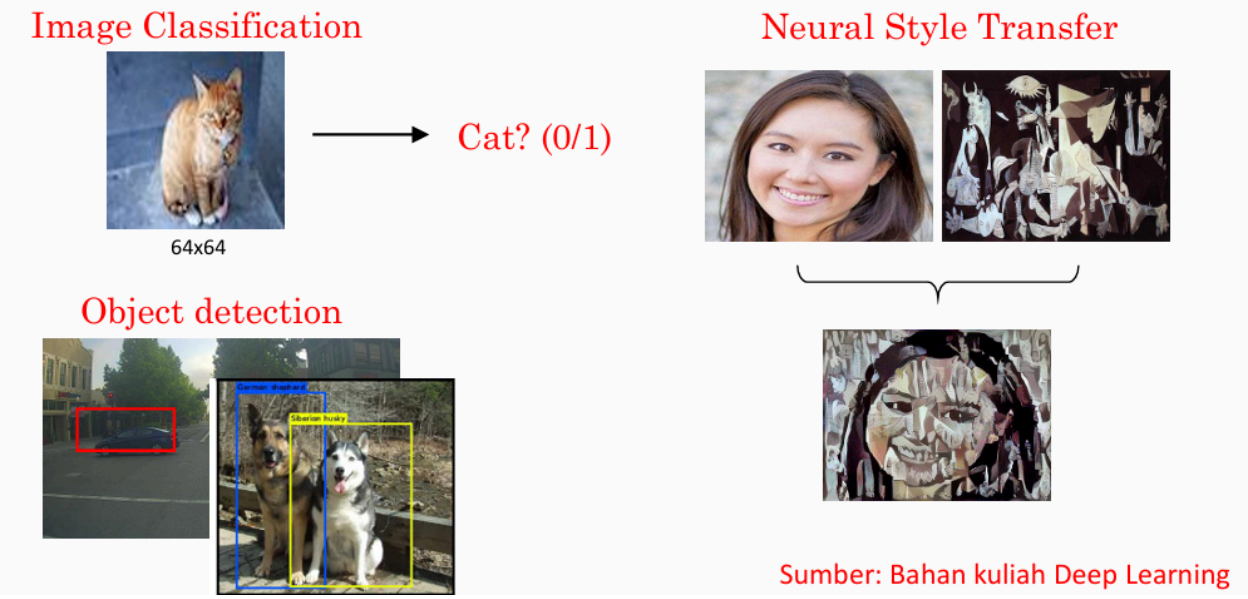
Pembelajaran mendalam biasanya diimplementasikan menggunakan arsitektur jaringan saraf. Istilah "deep" (dalam) mengacu pada jumlah lapisan di dalam jaringan semakin banyak lapisan, semakin dalam jaringan. Jaringan saraf tradisional hanya berisi 2atau 3 lapisan, sedangkan jaringan dalam dapat memiliki ratusan lapisan.



Jaringan Syaraf Tiruan (*Artificial Neural Netwok*)

1. **Computer Vision**

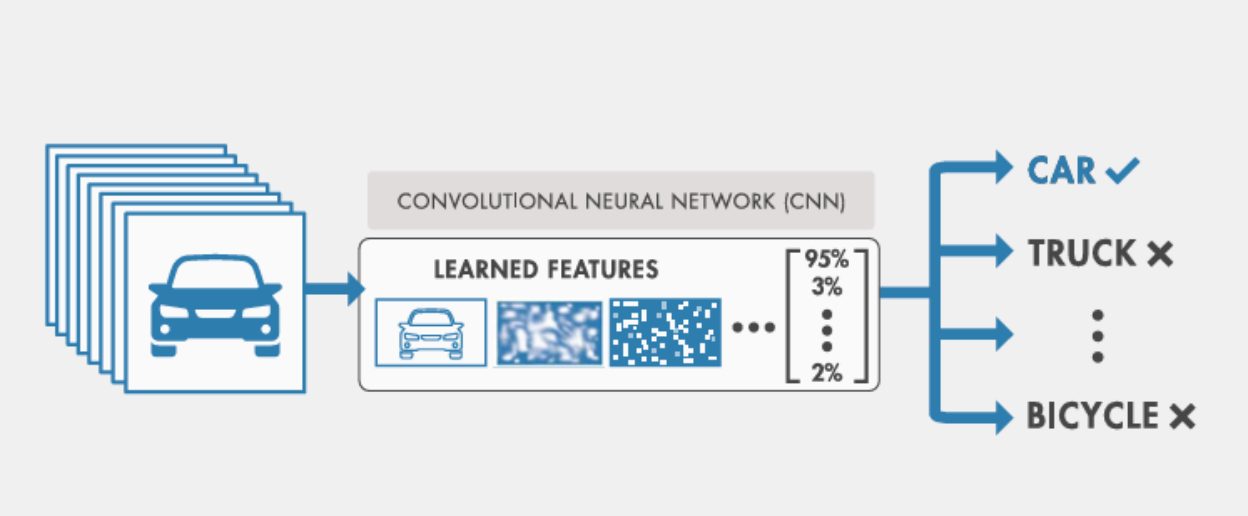
Salah satu kegunaan pembelajaran deep learning adalah di dalam computer vision



Apa yang Membuat Pembelajaran MendalamCocok untuk Computer Vision?

1. Kemudahan akses ke kumpulan besar data yang berlabel

   Kumpulan data seperti ImageNet, COCO, PASCAL VOC tersedia secara free, dan berguna untuk melatih berbagai jenis objek.

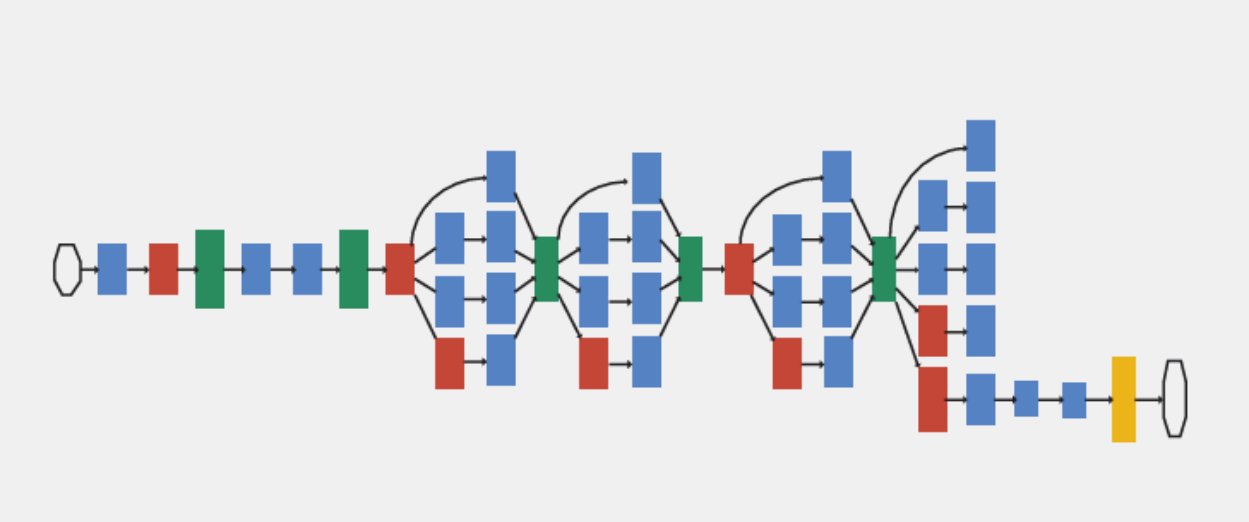


2. Peningkatan daya komputasi

GPU berkinerja tinggi (high performa computing) mempercepat pelatihan sejumlah besar data yang diperlukan untuk pembelajaran mendalam, mengurangi waktu pelatihan dari berminggu-minggu menjadi berjam-jam.

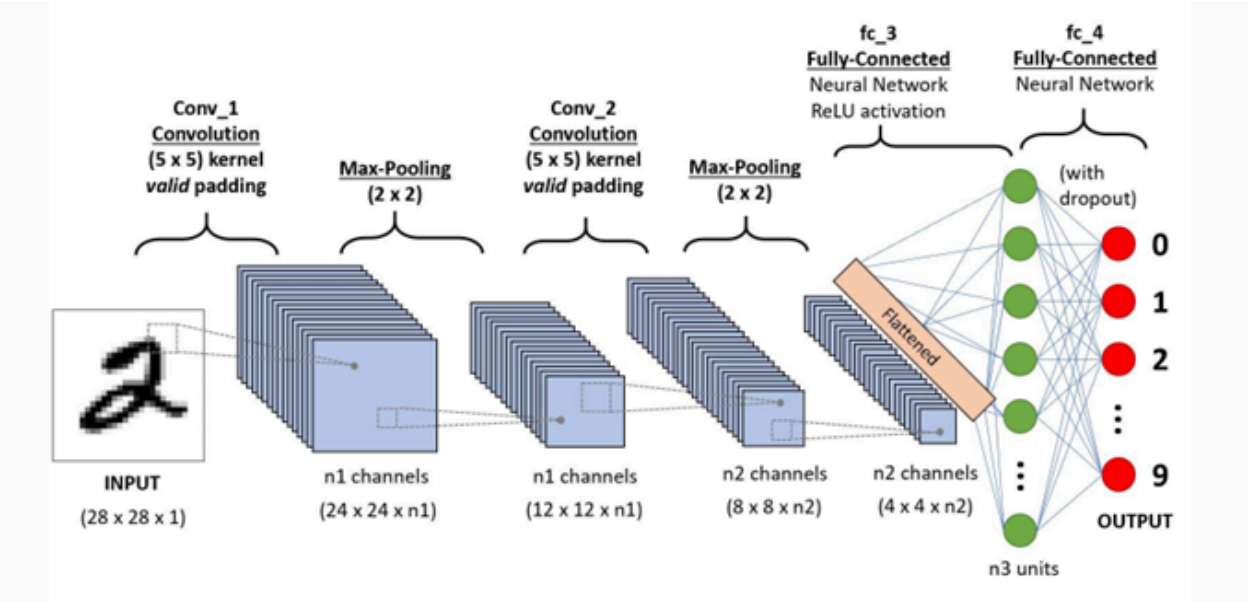
3. Model terlatih (pretrained) yang dibangun Oleh para ahli

Model seperti AlexNet dapat dilatih ulang untuk melakukan tugas pengenalan baru menggunakan teknik yang disebut pembelajaran transfer (transfer learning). Sementara AlexNet dilatih pada 1,3 juta gambar beresolusi tinggi untuk mengenali 1000 objek berbeda, pembelajaran transfer yang akurat dapat dicapai dengan jauh lebih cepat.



B. KONSEP CNN

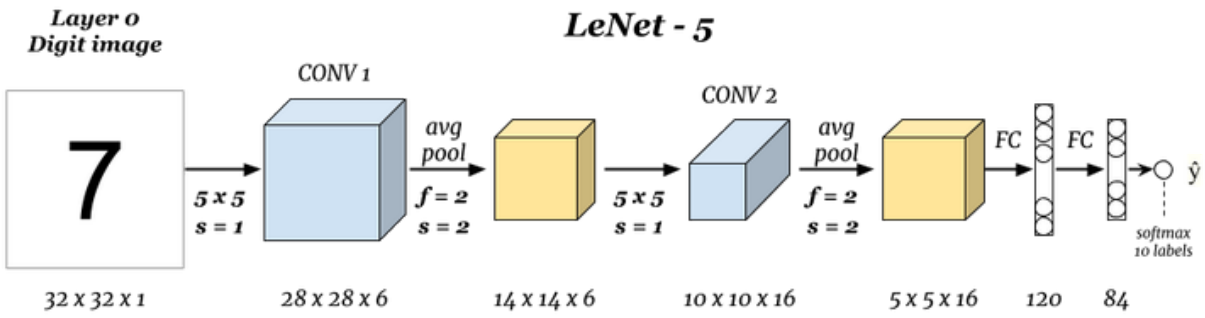
- Convolutional Neural Network (CNN atau ConvNet) adalah algoritma pembelajaran mendalam yang populer, umumnya digunakan untuk menganalisis citra se erti en enalan ob•ek klasifikasi ob•ek dll.
- CNN merupakan arsitektur jaringan untuk pembelajaran mendalam yang belajar langsung dari data, dengan menghilangkan kebutuhan untuk melakukan ekstraksi fitur ecara manual
- CNN dapat disebut juga jaringan syaraf tiruan yang melibatkan konvolusi (CNN = ANN + convolution)



C. ARSITEKTUR CNN

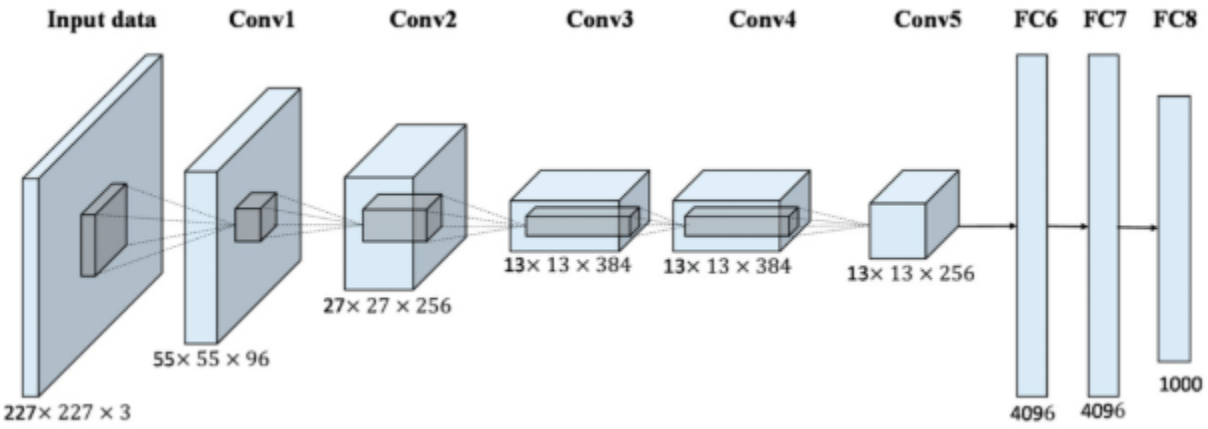
1. LetNet-5

LeNet-5 adalah salah satu arsitektur CNN paling awal dan menjadi tonggak dalam pengembangan CNN. Diperkenalkan oleh Yann LeCun et al. pada tahun 1998, LeNet-5 awalnya digunakan untuk mengenali angka tulisan tangan dalam kode pos dan berhasil menunjukkan kehandalannya. LeNet-5 terdiri dari beberapa lapisan konvolusi dan penggabungan, diikuti oleh beberapa lapisan fully connected. Meskipun arsitektur ini sederhana dibandingkan dengan CNN modern, LeNet-5 telah membuktikan pentingnya pendekatan konvolusi dalam tugas-tugas pengenalan gambar.



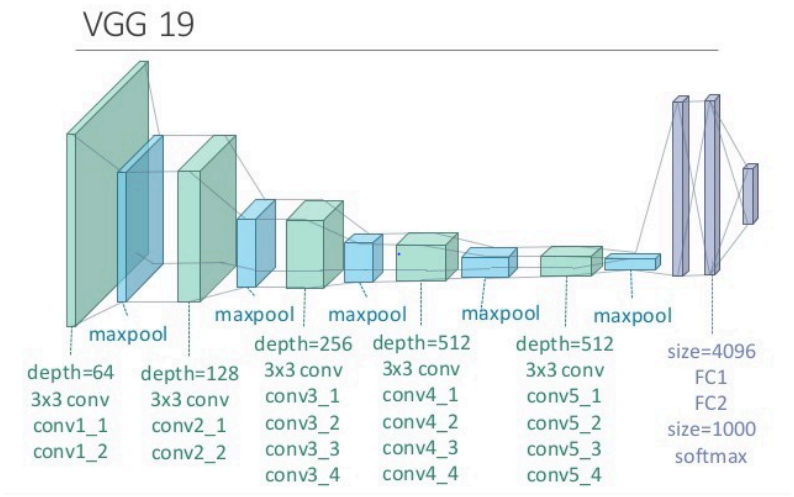
2. AlexNet

AlexNet dianggap sebagai titik balik dalam perkembangan arsitektur CNN modern. Diperkenalkan oleh Alex Krizhevsky et al. pada tahun 2012, AlexNet menjadi pemenang kompetisi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dengan hasil yang mengesankan. AlexNet menggunakan konsep lapisan konvolusi dalam yang lebih dalam daripada LeNet-5, serta menggunakan teknik seperti fungsi aktivasi ReLU (Rectified Linear Unit) dan teknik dropout untuk mengurangi overfitting. Arsitektur ini mengilhami perkembangan banyak arsitektur CNN yang lebih kompleks di masa depan.



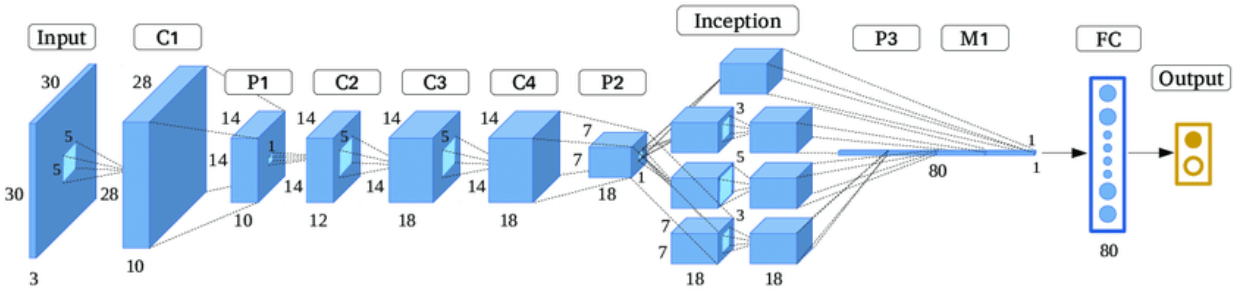
3. VGG(Visual Geometry Group)

VGG adalah arsitektur CNN yang diperkenalkan oleh Karen Simonyan dan Andrew Zisserman pada tahun 2014. Arsitektur ini dikenal dengan kedalaman yang luar biasa, di mana model VGG-16 memiliki 16 lapisan konvolusi dan model VGG-19 memiliki 19 lapisan konvolusi. VGG menekankan pentingnya kedalaman dalam ekstraksi fitur dan memberikan hasil yang sangat baik dalam berbagai tugas pengenalan gambar. Namun, kompleksitas tinggi dari model VGG juga menyebabkan konsumsi sumber daya komputasi yang besar.



4. GoogLeNet(Inceptin)

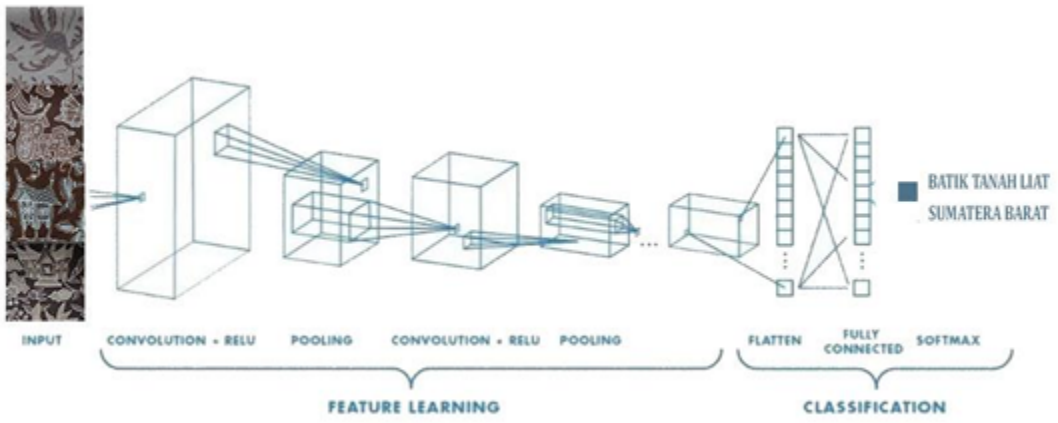
GoogLeNet, juga dikenal sebagai Inception v1, adalah arsitektur CNN yang dikembangkan oleh Google pada tahun 2014. Arsitektur ini terkenal karena penggunaan blok Inception yang kompleks, di mana lapisan konvolusi dengan berbagai ukuran kernel digabungkan secara paralel untuk mengekstraksi fitur-fitur dalam skala dan kompleksitas yang berbeda. Pendekatan ini memungkinkan GoogLeNet untuk menjadi lebih efisien dalam hal jumlah parameter dan sumber daya komputasi.



5. ResNet (Residual Neural Network)

ResNet adalah arsitektur CNN yang diperkenalkan oleh Kaiming He et al. pada tahun 2015. Salah satu masalah dalam melatih jaringan yang sangat dalam adalah gradien yang menghilang atau meledak saat melalui lapisan-lapisan yang dalam. ResNet mengatasi masalah ini dengan menggunakan blok-blok identitas yang memungkinkan gradien melompat atau shortcut. Dengan menggunakan pendekatan ini, ResNet dapat melatih jaringan yang lebih dalam dengan lebih efisien dan secara signifikan meningkatkan kinerja dalam berbagai tugas pengenalan gambar.

D. ALOGRITMA PROSES CNN



Pada bagian feature learning, terdapat lapisan yang berguna menerima input gambar secara langsung diawal dan memprosesnya sampai menghasilkan output data. Lapisan yang ada pada proses ini terdiri dari lapisan konvolusi dan lapisan pooling, dimana setiap proses lapisan tersebut akan menghasilkan feature maps berupa angka-angka yang merepresentasikan gambar untuk kemudian diteruskan pada bagian lapisan klasifikasi.

Pada lapisan classification, terdiri dari beberapa lapisan yang berisi neuron yang terkoneksi penuh (fully connected) dengan lapisan lain. Lapisan ini menerima input dari output layer bagian feature learning yang kemudian diproses pada flatten dengan tambahan beberapa hidden layer pada fully connected hingga menghasilkan output berupa akurasi klasifikasi dari setiap kelas. Bagian dari lapisan arsitektur CNN dijelaskan pada tabel dibawah.

Tabel 1. Algoritma *Convolutional Neural Network*

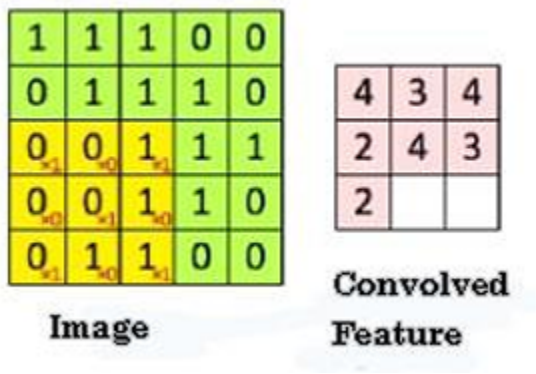
Algoritma Proses *Convolutional Neural Network*	
Feature learning	*a. Input Layer* *b. Convolution Layer* *c. Activation Layer* *d. Pooling Layer*
Classification	*e. Fully Connected Layer* *f. Output Layer*

a. *Input Layer*

Lapisan ini berguna untuk menampung pixel value dari citra yang diinputkan. Citra batik tanah liat yang telah diinputkan memiliki 3 channel warna RGB (Red, Green, Blue).

b. *Convolution Layer*

Convolutional layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (pixels). Proses konvolusi menggunakan kernel dan stride, proses konvolusi ini adalah proses kombinasi antara dua buah matriks yang berbeda untuk menghasilkan suatu nilai matriks yang baru. Dalam pengolahan citra, konvolusi berarti mengaplikasikan sebuah kernel (kotak kuning) pada citra di semua offset yang memungkinkan seperti ilustrasi pada gambar berikut,



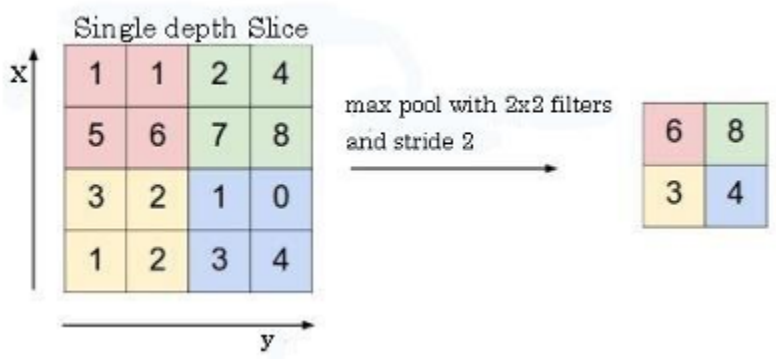
Kotak hijau secara keseluruhan adalah gambar yang akan dilakukan konvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari gambar dapat dilihat dari gambar di sebelah kanan. Tujuannya dilakukan konvolusi pada data gambar yaitu untuk mengekstraksi fitur dari gambar input.

c. *Activation laye*r

Pada lapisan ini, terjadi proses pengubahan nilai nilai feature map pada jarak tertentu tergantung pada fungsi aktivasi yang dipakai, yaitu fungsi aktifasi ReLu. Pada dasarnya fungsi ReLu (Rectified Linear Unit) melakukan "treshold" dari 0 hingga infinity. Fungsi ini menjadi salah satu fungsi yang populer saat ini.

d. *Pooling layer*

Lapisan pooling bekerja di setiap tumpukan feature map dan melakukan pengurangan pada ukurannya. Bentuk lapisan pooling umumnya dengan menggunakan filter dengan ukuran 2x2 yang diaplikasikan dengan langkah sebanyak dua dan beroperasi pada setiap irisan dari inputnya. Berikut ini adalah contoh gambar operasi max-pooling:



Kotak yang berwarna merah, hijau, kuning dan biru pada sisi kiri merupakan kelompok kotak yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan kotak disebelah kanannya.

e. *Fully-connected*

Fully-connected atau lebih dikenal dengan dense layer adalah lapisan dimana semua neuron aktivitas dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan syaraf tiruan

f. *Output Layer*

Layer terakhir yang menghasilkan prediksi, seperti kelas objek dalam citra. Biasanya menggunakan **Softmax** (untuk klasifikasi multi-kelas) atau **Sigmoid** (untuk binary classification).

E. IMPLEMENTASI CNN

1. Import library yang diperlukan

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Dense, Input # type: ignore
```

✓ 28.2s

2. Membaca dataset menggunakan library pandas dengan fungsi read_csv

```
df = pd.read_csv('weatherHistory.csv')
df.head()
```

✓ 0.2s Python

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

3. Melakukan pengecekan terhadap nilai null

```
df.isnull().sum()
```
✓ 0.0s

```
Formatted Date                0
Summary                       0
Precip Type                 517
Temperature (C)               0
Apparent Temperature (C)      0
Humidity                      0
Wind Speed (km/h)             0
Wind Bearing (degrees)        0
Visibility (km)               0
Loud Cover                    0
Pressure (millibars)          0
Daily Summary                 0
dtype: int64
```

4. Melakukan pengecekan data yang duplikat

```
df.duplicated().sum()
```
✓ 0.0s

```
24
```

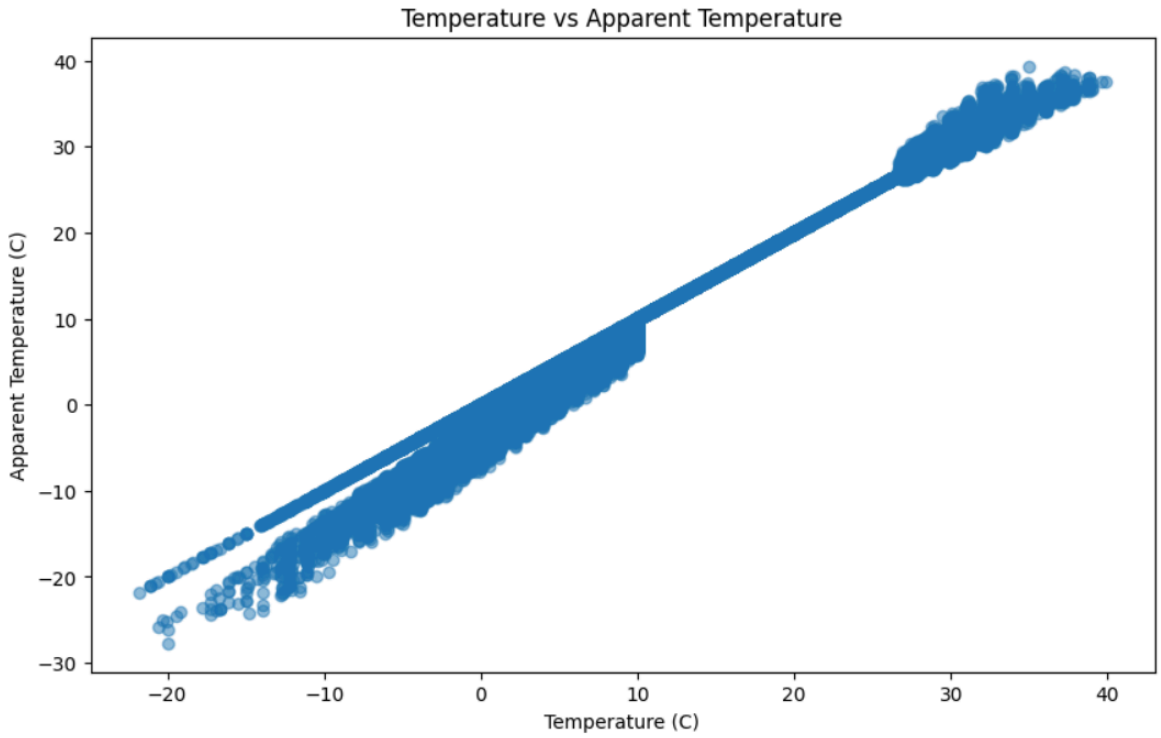
5. Menangani data null dan duplikat dengan cara dilakukan drop

```
df = df.dropna()
df = df.drop_duplicates()
```
✓ 0.1s

6. Melakukan visualisasi untuk atribut Temperature (C) dengan Apparent Temperature (C) untuk melihat hubungannya

```python
# visualize data
plt.figure(figsize=(10, 6))
plt.scatter(df['Temperature (C)'], df['Apparent Temperature (C)'], alpha=0.5)
plt.title('Temperature vs Apparent Temperature')
plt.xlabel('Temperature (C)')
plt.ylabel('Apparent Temperature (C)')
plt.show()
```

✓ 0.5s



7. Melakukan pengecekan outlier dan sekaligus menangani outlier dengan metode clipping

```python
df_original = df.copy()
columns_with_outliers = ['Temperature (C)', 'Apparent Temperature (C)', 'Humidity', 'Wind Speed (km/h)', 'Pressure (millibars)']

Q1 = df[columns_with_outliers].quantile(0.25)
Q3 = df[columns_with_outliers].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.0 * IQR
upper_bound = Q3 + 1.0 * IQR
outlier_mask = False
for col in columns_with_outliers:
    col_mask = (df[col] < lower_bound[col]) | (df[col] > upper_bound[col])
    outlier_mask = outlier_mask | col_mask

outliers = df[outlier_mask]
print(f"Number of rows with outliers: {len(outliers)}")

for col in columns_with_outliers:
    df[col] = df[col].clip(lower=lower_bound[col], upper=upper_bound[col])
```

✓ 0.0s

```
Number of rows with outliers: 17531
```

8. Memvisualisasikan hasil sebelum dan setelah dilakukan penanganan outlier

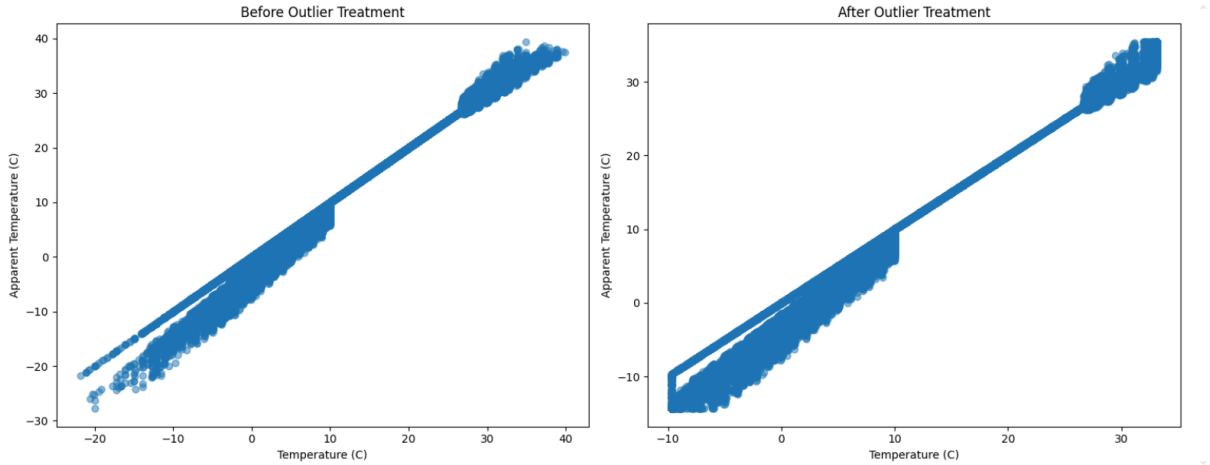
```python
# Visualize the diffference
plt.figure(figsize=(15, 6))

# Before outlier treatment
plt.subplot(1, 2, 1)
plt.scatter(df_original['Temperature (C)'], df_original['Apparent Temperature (C)'], alpha=0.5)
plt.title('Before Outlier Treatment')
plt.xlabel('Temperature (C)')
plt.ylabel('Apparent Temperature (C)')

# After outlier treatment
plt.subplot(1, 2, 2)
plt.scatter(df['Temperature (C)'], df['Apparent Temperature (C)'], alpha=0.5)
plt.title('After Outlier Treatment')
plt.xlabel('Temperature (C)')
plt.ylabel('Apparent Temperature (C)')

plt.tight_layout()
plt.show()
```

0.7s Python



9. Melakukan pemilihan atribut yaitu atribut yang bertipe number kemudian membaginya ke dalam 2 variabel yaitu variabel independen (X) dan variabel dependen (y). Jumlah variabel y sebanyak 20% dari seluruh data. Kemudian data latih atau variabel X dilakukan standarisasi untuk menyeratakan distribusi data

```python
df = df.select_dtypes(include='number')

X = df.drop(['Apparent Temperature (C)'], axis='columns')
y = df[['Apparent Temperature (C)']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

✓ 0.0s

10. Membangun model arsitektur CNN nya dengan model sequential (berurutan). Dimulai dengan menentukan input layer dengan dimensi satu. Dilanjutkan dengan membuat tiga hidden layer dengan fungsi aktivasi ReLU:
    a. Layer pertama memiliki 128 neuron
    b. Layer kedua memiliki 64 neuron
    c. Layer ketiga memiliki 32 neuron

    Lalu membuat layer output dengan jumlah layer sebanyak satu. Kemudian membuat kompilasi model dengan :
    a. optimizer='adam': Menggunakan Adam optimizer, algoritma optimasi yang populer dan efisien.
    b. loss='mse': Menggunakan Mean Squared Error sebagai fungsi loss, standar untuk masalah regresi.
    c. metrics=['mae']: Menggunakan Mean Absolute Error sebagai metrik tambahan untuk mengevaluasi performa model selama pelatihan.

    Terakhir membuat pelatihan modelnyan dengan jumlah latih sebanyak 50 kali dan validasi data nya sebanyak 20% dari data latih.

```python
# Train model
model = Sequential([
    Input(shape=(X_train_scaled.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])

model.compile(
    optimizer='adam',
    loss='mse',
    metrics=['mae']
)

history = model.fit(X_train_scaled, y_train, batch_size=32, epochs=50, validation_split=0.2)
```

✓ 6m 38.8s

```
Epoch 1/50
1919/1919 [==============================] - 11s 4ms/step - loss: 5.0341 - mae: 0.8199 - val_loss: 0.2676 - val_mae: 0.3515
Epoch 2/50
1919/1919 [==============================] - 8s 4ms/step - loss: 0.2336 - mae: 0.3307 - val_loss: 0.2338 - val_mae: 0.3446
Epoch 3/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.2021 - mae: 0.3029 - val_loss: 0.1673 - val_mae: 0.2456
Epoch 4/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1838 - mae: 0.2855 - val_loss: 0.1676 - val_mae: 0.2591
Epoch 5/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1601 - mae: 0.2626 - val_loss: 0.1590 - val_mae: 0.2446
Epoch 6/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1382 - mae: 0.2425 - val_loss: 0.1936 - val_mae: 0.3085
Epoch 7/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1240 - mae: 0.2247 - val_loss: 0.1282 - val_mae: 0.2130
Epoch 8/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1144 - mae: 0.2118 - val_loss: 0.1003 - val_mae: 0.1838
Epoch 9/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1056 - mae: 0.1995 - val_loss: 0.1016 - val_mae: 0.1627
Epoch 10/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.1033 - mae: 0.1962 - val_loss: 0.0933 - val_mae: 0.1644
Epoch 11/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.0978 - mae: 0.1879 - val_loss: 0.0916 - val_mae: 0.1697
Epoch 12/50
1919/1919 [==============================] - 7s 4ms/step - loss: 0.0947 - mae: 0.1825 - val_loss: 0.1111 - val_mae: 0.2195
Epoch 13/50
...
Epoch 49/50
1919/1919 [==============================] - 9s 5ms/step - loss: 0.0630 - mae: 0.1318 - val_loss: 0.0629 - val_mae: 0.1034
Epoch 50/50
1919/1919 [==============================] - 9s 5ms/step - loss: 0.0602 - mae: 0.1266 - val_loss: 0.0738 - val_mae: 0.1409
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

11. Melakukan evaluasi model dengan melihat MSE dan Score R2 nya

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test_scaled)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mse, r2
```

✓ 1.7s

```
600/600 [==============================] - 1s 2ms/step

(0.06493969261646271, 0.99942547082901)
```

12. Melakukan percobaan dengan menginputkan data untuk melihat hasil prediksi

```
print("Coba prediksi suhu nyata berdasarkan input cuaca!")

temperature = float(input("Temperature (°C): "))
humidity = float(input("Humidity (0-1): "))
wind_speed = float(input("Wind Speed (km/h): "))
wind_bearing= float(input("Wind Bearing (degrees): "))
visibility = float(input("Visibility (km): "))
loud_cover = float(input("Loud Cover: "))
pressure = float(input("Pressure (millibar): "))

input_data = pd.DataFrame([{
    'Temperature (C)': temperature,
    'Humidity': humidity,
    'Wind Speed (km/h)': wind_speed,
    'Wind Bearing (degrees)': wind_bearing,
    'Visibility (km)': visibility,
    'Loud Cover': loud_cover,
    'Pressure (millibars)': pressure
}])

input_scaled = scaler.transform(input_data)

prediction = model.predict(input_scaled)

print("\nBentuk hasil prediksi:", prediction.shape)
print("Hasil prediksi mentah:", prediction)

if isinstance(prediction, np.ndarray) and prediction.ndim > 1:
    predicted_value = prediction[0, 0] if prediction.shape[1] == 1 else prediction[0]
else:
    predicted_value = prediction[0]

print(f"\nPrediksi Apparent Temperature (°C): {float(predicted_value):.2f}")
```

✓ 2m 50.9s

```
Coba prediksi suhu nyata berdasarkan input cuaca!
1/1 [==============================] - 0s 35ms/step

Bentuk hasil prediksi: (1, 1)
Hasil prediksi mentah: [[6.661084]]

Prediksi Apparent Temperature (°C): 6.66
```

REFERENSI

Riyadi, Fahrizal. 2022. *Penerapan Convolutional Neural Network (CNN) untuk Klasifikasi Citra Daun Jambu Biji*. Dari https://ejurnal.sttdumai.ac.id/index.php/unitek/article/download/504/350 Diakses pada 20 Mei 2025.

Munir, Rinaldi. 2023. *Convolutional Neural Network (CNN)*. Dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/24-CNN-2022.pdf Diakses pada 20 Mei 2025.

Pemrograman Matlab. 2023. *Jenis-jenis Arsitektur Convolutional Neural Network (CNN) untuk Image Recognition dan Computer Vision*. Dari https://pemrogramanmatlab.com/2023/07/23/jenis-jenis-arsitektur-convolutional-neural-network-cnn-untuk-image-recognition-dan-computer-vision/ Diakses pada 20 Mei 2025.