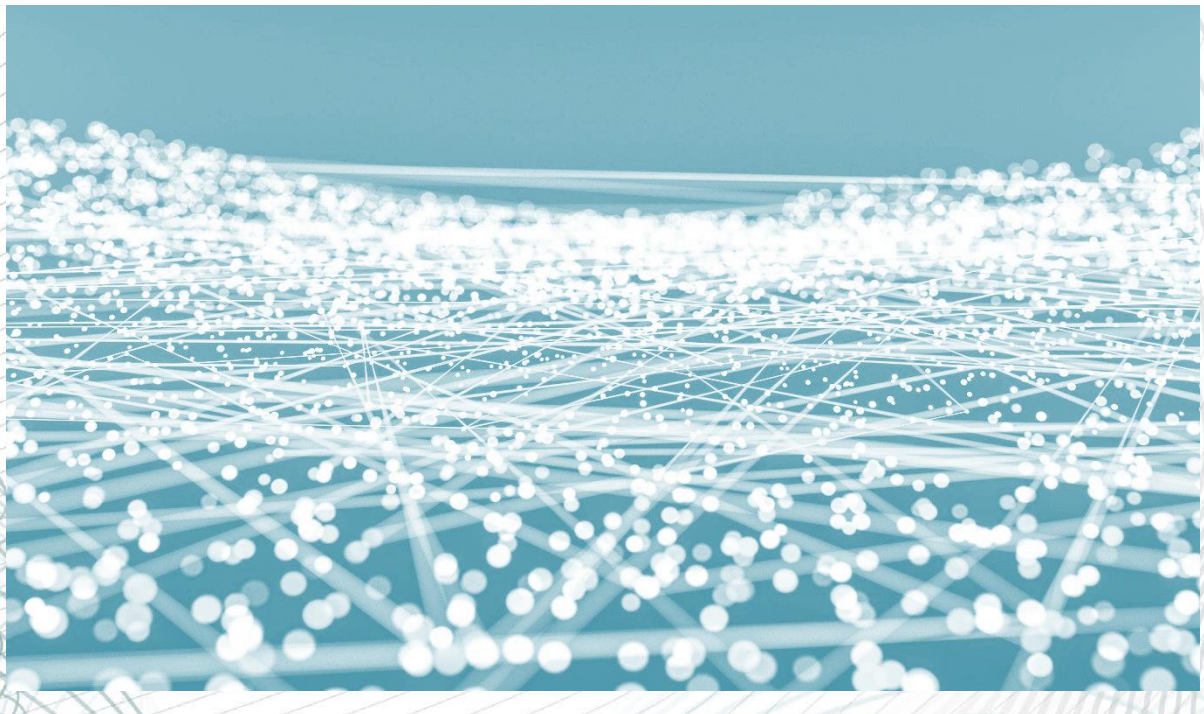
MODUL PRAKTIKUM

SD4102-Deep Learning



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

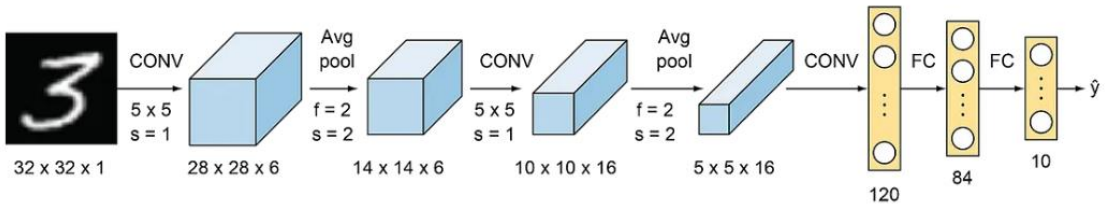
2024

MODUL 6

Modern CNN

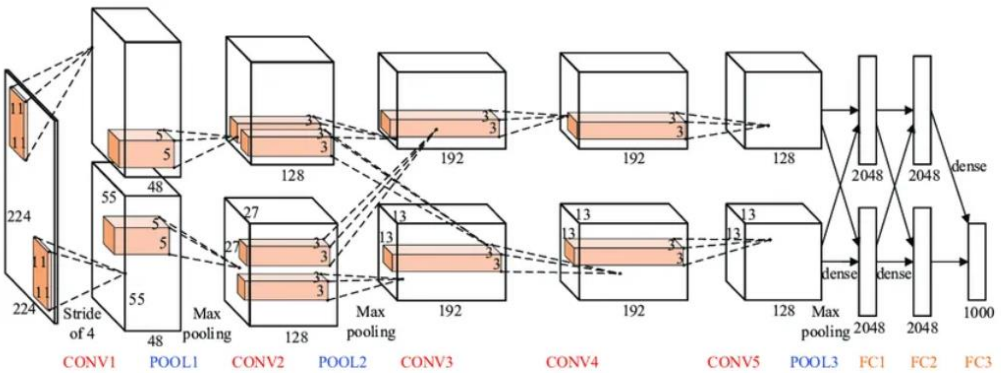
A. Konsep Dasar

Convolutional Neural Network (CNN) memiliki arsitektur yang terus dikembangkan diantaranya LeNet, AlexNet, dan VGG. Ketiga arsitektur ini sangat berpengaruh dalam pengembangan pengolahan citra dan visi komputer. LeNet, yang diperkenalkan oleh Yann LeCun pada tahun 1998, adalah salah satu model CNN pertama yang digunakan untuk pengenalan karakter dalam sistem pengenalan tulisan tangan. Arsitektur ini terdiri dari beberapa lapisan konvolusi dan lapisan pooling yang diikuti oleh lapisan fully connected. LeNet menunjukkan pentingnya pembelajaran fitur otomatis dari gambar dan menjadi dasar bagi banyak model CNN yang lebih kompleks.



Lenet-5 Architecture

AlexNet dirancang oleh Alex Krizhevsky dan timnya, menandai awal revolusi dalam penggunaan CNN untuk klasifikasi gambar, terutama setelah suksesnya dalam kompetisi ImageNet pada tahun 2012. AlexNet memiliki arsitektur yang lebih dalam dibandingkan LeNet, dengan delapan lapisan (lima lapisan konvolusi dan tiga lapisan fully connected) dan menggunakan teknik regularisasi seperti dropout untuk mencegah overfitting. Selain itu, AlexNet memanfaatkan GPU untuk mempercepat pelatihan, memungkinkan pengolahan data yang lebih besar dan kompleks.



AlexNet Architecture

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan arsitektur pengembangan *Convolutional Neural Network (CNN)* dalam pemahaman Deep Learning.

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar arsitektur LeNet dan AlexNet dari *Convolutional Neural Network (CNN).*
2. Mahasiswa mampu membangun model dengan arsitektur LeNet dan AlexNet pada *Convolutional Neural Network (CNN)* untuk data gambar.
3. Mahasiswa mampu menganalisis hasil evaluasi model arsitektur LeNet dan AlexNet pada *Convolutional Neural Network (CNN)* untuk data gambar.

C. Dataset dan bahasa pemrograman Python

C.1 Dataset dalam praktikum

Dalam praktikum modul ini akan digunakan dataset MNIST (Modified National Institute of Standards and Technology) adalah salah satu dataset paling terkenal dalam pembelajaran mesin, yang berisi 70.000 gambar grayscale dari angka tulisan tangan, dengan 60.000 gambar untuk pelatihan dan 10.000 gambar untuk pengujian. Setiap gambar berukuran 28x28 piksel dan dilabeli dengan angka dari 0 hingga 9, mewakili total 10 kelas. Dataset ini diambil dari tulisan tangan pekerja pos dan siswa di AS, sehingga mencerminkan variasi gaya tulisan. MNIST sering digunakan sebagai benchmark untuk algoritma pengenalan pola dan pembelajaran mendalam, serta merupakan titik awal yang populer bagi pemula dalam eksperimen pengenalan citra dan pengembangan model klasifikasi.

C.2 Bahasa Pemrograman Python

Pada praktikum kali ini, kita akan menggunakan IDE Python Service dari Google Colab, sebelumnya perlu menyiapkan akun google pribadi untuk mengakses servis ini di akun google masing-masing.

Library yang akan digunakan diantaranya Pandas, TensorFlow, dan Keras.

1) NumPy : digunakan untuk operasi matematika berbasis array dan matriks. NumPy sangat penting dalam komputasi numerik dengan Python pada praktikum ini.
2) TensorFlow : framework yang kuat dan fleksibel untuk machine learning dan deep learning, cocok untuk proyek skala besar dan aplikasi industri.
3) Keras : menyediakan antarmuka yang lebih sederhana untuk membangun model, membuat pengembangan deep learning lebih cepat dan mudah dipahami.

Integrasi TensorFlow dan Keras

Sejak TensorFlow 2.0, Keras menjadi API default untuk membangun model di TensorFlow, membuat pengembangan model deep learning jauh lebih mudah dan efisien. Tanpa membangun code TensorFlow dari awal, Anda bisa menggunakan Keras untuk membangun arsitektur model dengan cara yang lebih sederhana dan cepat.

Praktikum deep learning pada modul ini dapat maksimal dipraktikan dengan pemahaman bidang atau mata kuliah terkait seperti algoritma pemrograman, struktur data, dan machine learning.

Dataset dan Source code akan diberikan saat praktikum berlangsung dan dapat anda unduh pada folder berikut Modul CNN 2024 .

D. Implementasi Arsitektur dalam CNN

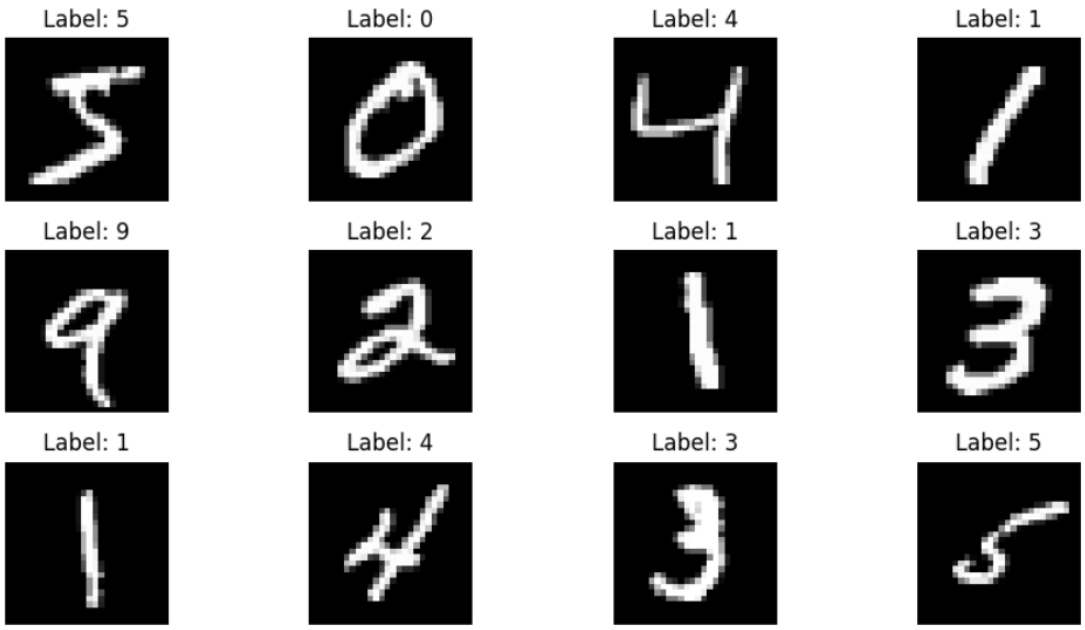
D.1 Arsitektur LeNet

1) Import Library dan load dataset

```
import tensorflow as tf
from tensorflow.keras import layers, Model
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# Preprocess dataset
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0  # Normalize dan reshape
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0

y_train = to_categorical(y_train, 10)  # One-hot encoding untuk label
y_test = to_categorical(y_test, 10)
```

2) Tampilkan sample dataset

```
import matplotlib.pyplot as plt

# Tampilkan 12 sampel gambar dari dataset MNIST
plt.figure(figsize=(10, 5))
for i in range(12):
    plt.subplot(3, 4, i + 1)
    plt.imshow(x_train[i].reshape(28, 28), cmap='gray')  # reshape kembali ke ukuran 28x28
    plt.title(f"Label: {y_train[i].argmax()}")  # Menampilkan label asli dari gambar
    plt.axis('off')
plt.tight_layout()
plt.show()
```



3) Membangun model

```
# Definisikan LeNet Model
def LeNet(input_shape=(28, 28, 1), num_classes=10):
    return tf.keras.Sequential([
        layers.Input(shape=input_shape),
        layers.Conv2D(6, 5, activation='relu'),
        layers.MaxPooling2D(2),
        layers.Conv2D(16, 5, activation='relu'),
        layers.MaxPooling2D(2),
        layers.Flatten(),
        layers.Dense(120, activation='relu'),
        layers.Dense(84, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
```

4) Tampilkan ringkasan model

```
# Initialize and compile the model
lenet_model = LeNet(input_shape=(28, 28, 1), num_classes=10)
lenet_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 24, 24, 6)	156
max_pooling2d (MaxPooling2D)	(None, 12, 12, 6)	0
conv2d_1 (Conv2D)	(None, 8, 8, 16)	2,416
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 16)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 120)	30,840
dense_1 (Dense)	(None, 84)	10,164
dense_2 (Dense)	(None, 10)	850

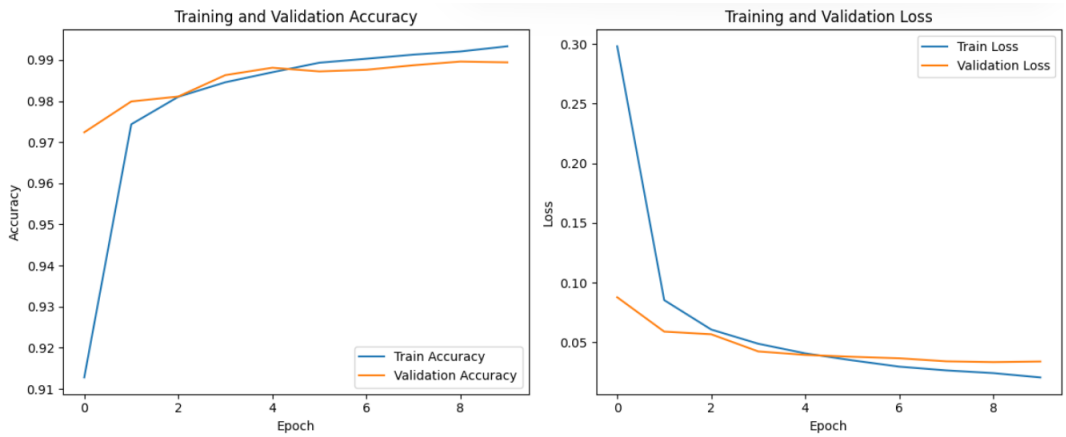
Total params: 44,426 (173.54 KB)
Trainable params: 44,426 (173.54 KB)
Non-trainable params: 0 (0.00 B)

5) Kompilasi dan Lakukan Pelatihan Model

```
lenet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = lenet_model.fit(x_train, y_train, epochs=4, batch_size=128, validation_data=(x_test, y_test))

# Evaluate the model
test_loss, test_accuracy = lenet_model.evaluate(x_test, y_test, verbose=0)
print(f"\nTest Accuracy: {test_accuracy:.4f}")
```

```
Epoch 1/4
469/469 ━━━━━━━━━━━━━━━━━━━━ 27s 53ms/step - accuracy: 0.7917 - loss: 0.6994 - val_accuracy: 0.9693 - val_loss: 0.0951
Epoch 2/4
469/469 ━━━━━━━━━━━━━━━━━━━━ 40s 51ms/step - accuracy: 0.9722 - loss: 0.0919 - val_accuracy: 0.9753 - val_loss: 0.0719
Epoch 3/4
469/469 ━━━━━━━━━━━━━━━━━━━━ 42s 52ms/step - accuracy: 0.9797 - loss: 0.0642 - val_accuracy: 0.9820 - val_loss: 0.0523
Epoch 4/4
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 52ms/step - accuracy: 0.9839 - loss: 0.0515 - val_accuracy: 0.9850 - val_loss: 0.0444
```

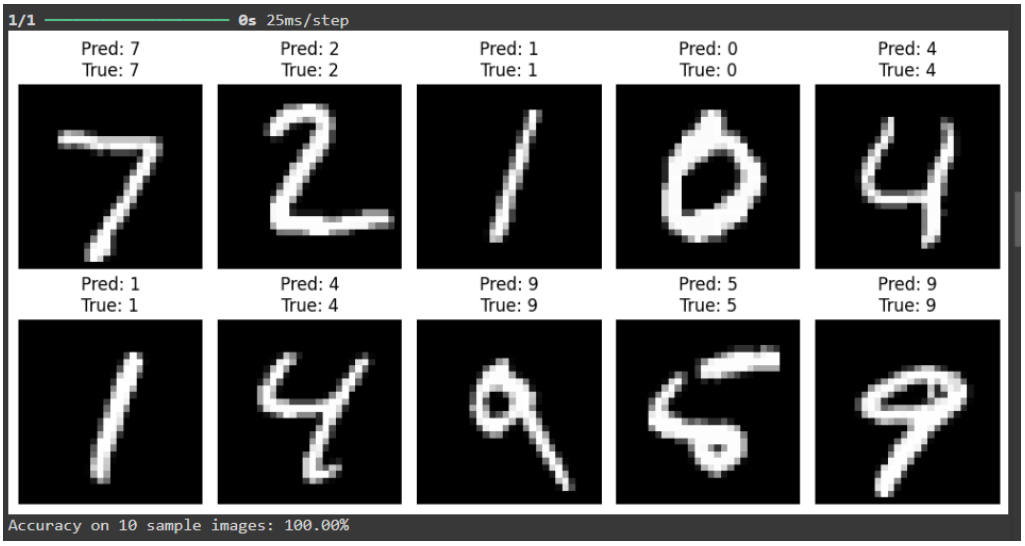
6) Plot hasil pelatihan



7) Uji gambar dengan 10 sample pada dataset

```
import numpy as np


# Select 10 samples without labels
x_sample = x_test[:10]
y_sample = y_test[:10]

# Predict using the trained model
y_pred = lenet_model.predict(x_sample)
y_pred_labels = np.argmax(y_pred, axis=1)
y_true_labels = np.argmax(y_sample, axis=1)

# Plot the 10 images with predicted and actual labels
plt.figure(figsize=(10, 5))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_sample[i].reshape(28, 28), cmap='gray')
    plt.title(f"Pred: {y_pred_labels[i]}\nTrue: {y_true_labels[i]}")
    plt.axis('off')

# Display the plot
plt.tight_layout()
plt.show()

# Calculate and print accuracy on the 10 samples
accuracy = np.sum(y_pred_labels == y_true_labels) / len(y_true_labels)
print(f"Accuracy on 10 sample images: {accuracy * 100:.2f}%")
```

Output :



8) Lakukan Evaluasi model dari data uji

```
import pandas as pd
from sklearn.metrics import classification_report

report = classification_report(y_true_labels, y_pred_labels, output_dict=True)
report_df = pd.DataFrame(report).transpose()
print("Tabel Evaluasi Model LeNet pada MNIST:")
print(report_df[['precision', 'recall', 'f1-score', 'support']])
```

D.2 Arsitektur AlexNet

Arsitektur AlexNet terdiri dari delapan lapisan yang dapat dilatih, yaitu lima lapisan konvolusi dan tiga lapisan fully connected. AlexNet menggunakan ukuran kernel besar (11x11) untuk lapisan konvolusi pertama dan diikuti oleh lapisan-lapisan dengan ukuran kernel yang lebih kecil (3x3). Model ini juga mengimplementasikan teknik regularisasi seperti dropout untuk mengurangi overfitting, serta menggunakan ReLU (Rectified Linear Unit) sebagai fungsi aktivasi. Selain itu, AlexNet memanfaatkan pemrosesan GPU untuk mempercepat pelatihan dan meningkatkan efisiensi dalam pengolahan data besar. Dengan arsitekturnya yang dalam dan kompleks, AlexNet berhasil meningkatkan akurasi klasifikasi gambar secara signifikan, menjadikannya sebagai salah satu model dasar dalam perkembangan jaringan saraf mendalam. Berikut adalah model AlexNet yang dapat digunakan :

```python
model = models.Sequential()

    # Layer 1: Convolutional Layer
    model.add(layers.Conv2D(96, kernel_size=(11, 11), strides=(4, 4), activation='relu', input_shape=input_shape))
    model.add(layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
    # Layer 2: Convolutional Layer
    model.add(layers.Conv2D(256, kernel_size=(5, 5), padding='same', activation='relu'))
    model.add(layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
    # Layer 3: Convolutional Layer
    model.add(layers.Conv2D(384, kernel_size=(3, 3), padding='same', activation='relu'))
    # Layer 4: Convolutional Layer
    model.add(layers.Conv2D(384, kernel_size=(3, 3), padding='same', activation='relu'))
    # Layer 5: Convolutional Layer
    model.add(layers.Conv2D(256, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
    # Flatten the output before feeding it to the fully connected layers
    model.add(layers.Flatten())
    # Layer 6: Fully Connected Layer
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))  # Dropout layer to reduce overfitting
    # Layer 7: Fully Connected Layer
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))  # Dropout layer to reduce overfitting
    # Layer 8: Output Layer
    model.add(layers.Dense(num_classes, activation='softmax'))

    return model
```

Dengan model AlexNet diatas ikuti langkah-langkah pada D.1

TUGAS INDIVIDU

- Tugas individu dikerjakan saat praktikum berlangsung dengan waktu yang ditentukan.
- Soal tugas individu terbagi menjadi tiga level soal yaitu *Basic*, *Medium*, *dan Hard*. Praktikan dapat menyelesaikan ketiganya untuk mendapatkan poin maksimal.
- Namun praktikan cukup menyelesaikan minimal satu soal jika estimasi waktu praktikum sudah akan berakhir agar tidak melewatkan penilaian test lisan.