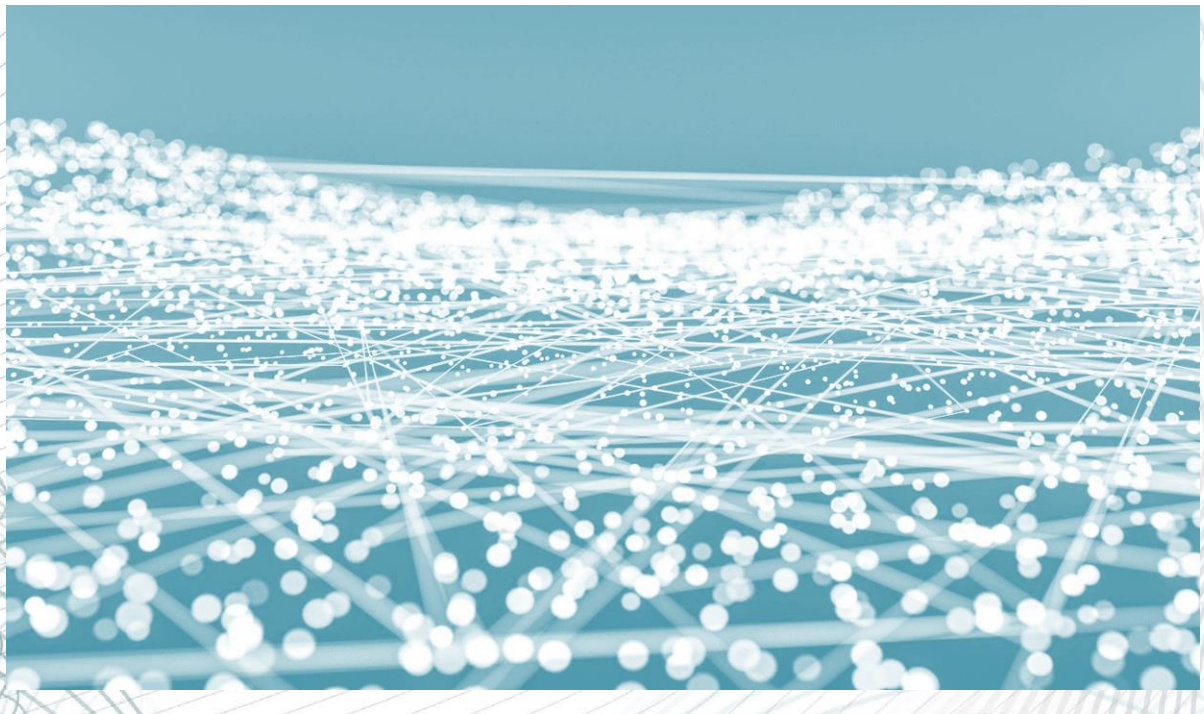
MODUL PRAKTIKUM

SD4102-Deep Learning



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

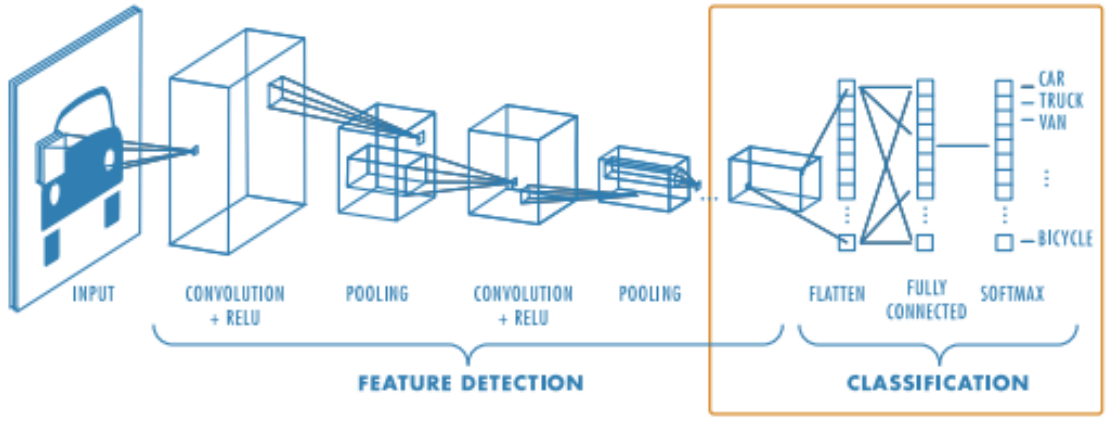
2024

MODUL 5

Convolutional Neural Network (CNN)

A. Konsep Dasar

Convolutional Neural Networks (CNN) adalah jenis arsitektur neural network yang dirancang khusus untuk pemrosesan data grid seperti gambar. CNN terdiri dari beberapa lapisan yang saling terhubung, yang mana lapisan utamanya adalah lapisan konvolusi (convolutional layer). Lapisan ini berfungsi untuk mengekstraksi fitur dari gambar melalui operasi konvolusi, di mana filter (kernel) akan berjalan di seluruh gambar untuk mendeteksi pola-pola seperti tepi, tekstur, atau bentuk. Hasil dari operasi ini adalah sebuah feature map, yang kemudian diteruskan ke lapisan-lapisan berikutnya untuk pengenalan pola yang lebih kompleks seperti pada gambar 1.



Gambar 1. Convolutional Neural Networks

Selain lapisan konvolusi, CNN juga menggunakann parameter penting lain seperti stride, padding, dan pooling yang mampu mempengaruhi ukuran output dari lapisan konvolusi. Berikut adalah penjelasan parameter-parameter pada CNN:

1) Stride

Stride adalah besaran langkah yang dilakukan oleh kernel untuk bergerak melalui semua submatriks yang terdapat pada gambar. Jika stride = 1, filter akan bergeser satu piksel setiap kali; jika stride = 2, filter akan bergeser dua piksel setiap kali.

2) Padding

Padding adalah proses menambahkan angka 0 pada setiap sisi yang terdapat pada matriks gambar. Padding dilakukan untuk mengontrol ukuran output dari lapisan konvolusi dan menjaga informasi pada tepi gambar.

3) Pooling

Pooling merupakan proses yang digunakan untuk mereduksi data. Pooling dilakukan terhadap submatriks yang terdapat pada matriks gambar. Terdapat beberapa pooling yang umum digunakan, diantaranya max pooling dan average pooling.

Pada gambar 1 menunjukan setelah melalui beberapa lapisan konvolusi dan pooling, data biasanya diteruskan ke fully connected layer, di mana semua neuron saling terhubung. Lapisan ini bertujuan untuk menggabungkan fitur-fitur yang diekstraksi oleh lapisan sebelumnya dan menghasilkan prediksi akhir. Biasanya, CNN digunakan dalam berbagai aplikasi deep learning seperti pengenalan objek, klasifikasi gambar, deteksi wajah, dan analisis video, karena kemampuannya untuk menangani data visual yang kompleks dengan lebih efisien dibandingkan model neural network lainnya.

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

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

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar *Convolutional Neural Network (CNN)*.
2. Mahasiswa mampu membangun model *Convolutional Neural Network (CNN)* untuk data gambar dengan parameter-parameter konvolusi.
3. Mahasiswa mampu menganalisis hasil evaluasi model *Convolutional Neural Network (CNN)* untuk data gambar.

C. Dataset dan bahasa pemrograman Python

C.1 Dataset dalam praktikum

Dalam praktikum modul 5 akan digunakan dataset dengan rincian sebagai berikut:

1) Praktikum 1 pada D1: data satu gambar

2) Praktikum 2 pada D2 : dataset CIFAR-10 yaitu dataset yang terdiri dari 60.000 gambar warna berukuran 32x32 piksel dengan 10 kelas (seperti pesawat, mobil, burung, dll.).

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. Alih-alih menulis kode TensorFlow dari awal, Anda bisa menggunakan Keras untuk membangun arsitektur model dengan cara yang lebih sederhana dan cepat.

Praktikum deep learning pada modul 5 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 CNN

D.1 Demonstrasi CNN sederhana dengan Tensorflow dan Keras

1) Import Library

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
```

2) Pembuatan Gambar Tunggal Sebagai Input

Pada bagian ini, kita membuat satu gambar acak berukuran 28x28 piksel dengan 1 channel (grayscale):

```
# Buat gambar acak sebagai input (misalnya gambar 28x28 grayscale)
# Atau bisa ganti dengan gambar lain jika ada
input_image = np.random.rand(28, 28, 1).astype('float32')
```

Gambar di sini adalah array acak dengan nilai antara 0 hingga 1 (float), dengan ukuran yang sesuai untuk digunakan dalam CNN. Ini hanya untuk ilustrasi;

```
# Label untuk gambar tersebut (misalnya, label 3)
input_label = np.array([3])
```

Anda dapat mengganti ***input_image*** dengan gambar lain jika ada gambar nyata yang ingin digunakan.

3) Mengubah Format Data

Karena model Keras mengharapkan input dalam format batch **(dimensi (batch_size, height, width, channels)),** gambar tunggal kita perlu diubah agar memiliki dimensi batch, meskipun hanya satu gambar:

```
# Perluas dimensi agar sesuai dengan format input Keras (batch size, height, width, channels)
X_train_single = np.expand_dims(input_image, axis=0)
```

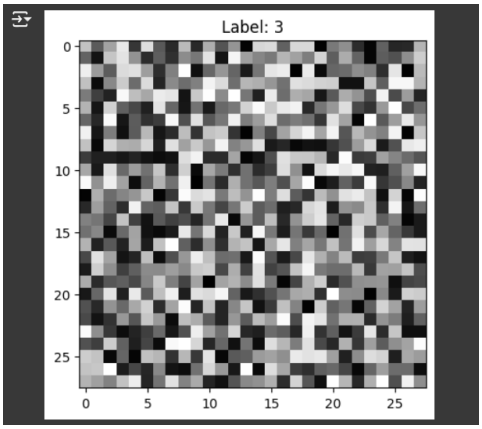
Sekarang **X_train_single** memiliki dimensi (1, 28, 28, 1) (1 gambar, tinggi 28, lebar 28, dan 1 channel grayscale).

4) Visualkan Gambar

Untuk melihat gambar yang digunakan dalam pelatihan, kita menampilkannya menggunakan matplotlib:

```
# Visualisasi gambar input
plt.imshow(input_image.squeeze(), cmap='gray')
plt.title(f"Label: {input_label[0]}")
plt.show()
```

Output:



5) Membangun Model CNN

Convolutional Layers: Model CNN dibangun dengan dua lapisan convolusi (Conv2D). Setiap lapisan convolusi diikuti oleh lapisan pooling untuk menurunkan resolusi gambar (downsampling):

```
# Bangun model CNN sederhana
model = models.Sequential()

# Convolutional Layer pertama dengan stride dan padding
model.add(layers.Conv2D(32, (3, 3), strides=(1, 1), padding='same', activation='relu', input_shape=(28, 28, 1)))
# Pooling Layer (2x2)
model.add(layers.MaxPooling2D((2, 2)))
# Convolutional Layer kedua
model.add(layers.Conv2D(64, (3, 3), strides=(1, 1), padding='same', activation='relu'))
# Pooling Layer (2x2)
model.add(layers.MaxPooling2D((2, 2)))
```

Conv2D: Menggunakan filter ukuran 3x3, stride 1x1, dan padding same, yang menjaga ukuran output layer sama dengan input.

MaxPooling2D: Menggunakan pooling 2x2 untuk mengurangi resolusi input dengan faktor 2.

Fully Connected Layer (Dense Layer): Setelah melalui lapisan convolusi, data dipipihkan menjadi vektor 1D dan diumpankan ke fully connected layer yang berfungsi untuk klasifikasi:

```
# Flatten output untuk fully connected layer
model.add(layers.Flatten())

# Fully connected layer
model.add(layers.Dense(64, activation='relu'))

# Output layer dengan 10 unit (untuk klasifikasi 10 kelas) dengan softmax
model.add(layers.Dense(10, activation='softmax'))

# Tampilkan arsitektur model
model.summary()
```

Lapisan terakhir menggunakan 10 neuron, yang sesuai dengan 10 kelas (angka 0-9), dengan aktivasi softmax untuk menghasilkan probabilitas.

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d_6 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_7 (Conv2D)	(None, 14, 14, 64)	18,496
max_pooling2d_7 (MaxPooling2D)	(None, 7, 7, 64)	0
flatten_3 (Flatten)	(None, 3136)	0
dense_6 (Dense)	(None, 64)	200,768
dense_7 (Dense)	(None, 10)	650

Total params: 220,234 (860.29 KB)
Trainable params: 220,234 (860.29 KB)
Non-trainable params: 0 (0.00 B)

6) Kompilasi dan Pelatihan Model

Model dikompilasi menggunakan optimizer Adam dan fungsi loss sparse_categorical_crossentropy, yang cocok untuk masalah klasifikasi:

```
# Kompilasi model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Model kemudian dilatih hanya pada satu gambar selama 10 epoch:

```
# Latih model menggunakan satu gambar dan label yang sesuai
model.fit(X_train_single, input_label, epochs=10)
```

7) Evaluasi model

Setelah pelatihan, model dievaluasi pada gambar yang sama. Karena hanya menggunakan satu gambar, hasil evaluasi (accuracy) tidak relevan dalam konteks generalisasi, tetapi ini memberi kita gambaran tentang bagaimana model menangani gambar itu sendiri:

```
# Evaluasi model (tentunya, evaluasi ini tidak signifikan karena hanya satu gambar)
loss, accuracy = model.evaluate(X_train_single, input_label)
print(f"Loss: {loss}, Accuracy: {accuracy}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - accuracy: 1.0000 - loss: 1.3351e-05
Loss: 1.3351351299206726e-05, Accuracy: 1.0
```

Praktik pada D.1 menggunakan data satu gambar tidak direkomendasikan untuk praktik nyata. Model CNN memerlukan dataset yang besar untuk mampu mengenali gambar lain yang serupa. **Bagian ini hanya untuk melatih pemahaman bagaimana membangun model dengan paremeter-parameter tertentu.**

D.2 Implementasi CNN untuk dataset CIFAR10 dari Tensorflow

1) Import Library

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

2) Import Dataset CIFAR10

```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

Gambar CIFAR-10 memiliki nilai piksel antara 0 dan 255. Dalam machine learning, seringkali lebih baik untuk menormalisasi data agar berada dalam rentang [0, 1] dengan cara membagi nilai piksel dengan 255.0 untuk membantu mempercepat proses pelatihan.

3) Membangun model CNN

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

4) Kompilasi dan Pelatihan Model

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

5) Evaluasi model

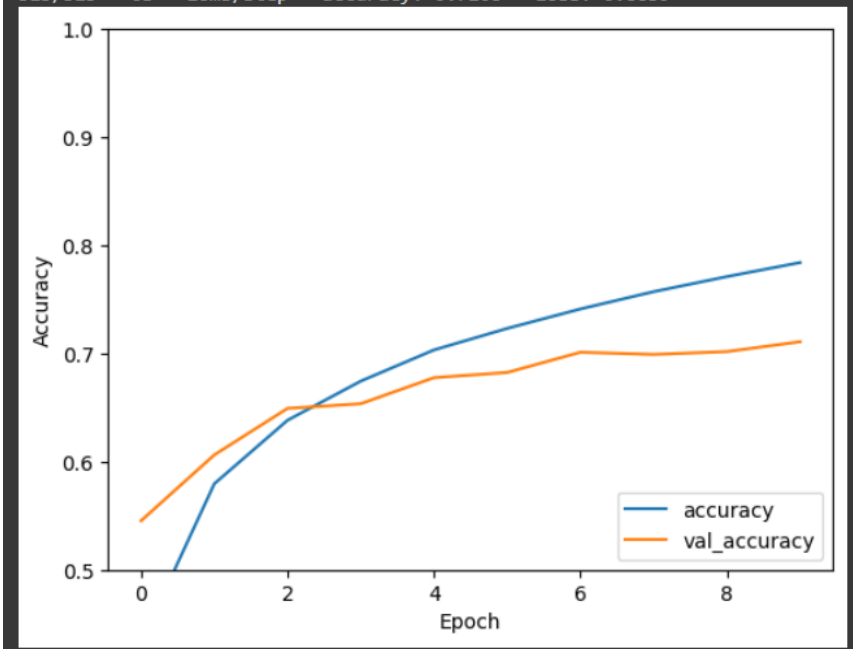
```
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

6) Plot grafik evaluasi

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 6s - 18ms/step - accuracy: 0.7108 - loss: 0.8839
```



Didapatlan plot akurasi pelatihan dan akurasi validasi berdasarkan hasil pelatihan model CNN.

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.