Modul Praktikum

Kecerdasan Buatan

10

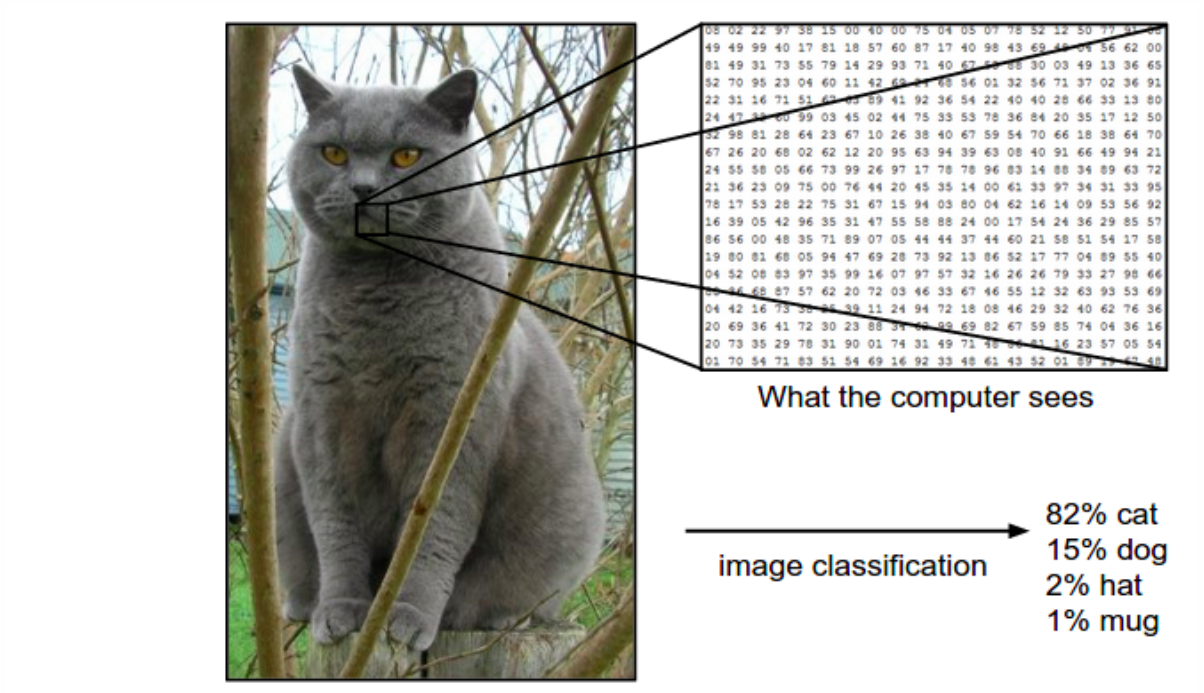
Daftar Isi

TensorFlow – Image Processing

Image Processing

Bagi manusia, kemampuan untuk melihat dan menganalisis segala hal di lingkungan sekitar itu hal mudah dilakukan dan diterapkan setiap hari. Tanpa kita sadari, kita telah melakukan prediksi tentang apa yang telah kita lihat sehari-hari dan melabeli sesuatu berdasarkan apa yang kita pelajari sejak kecil. Sebagai contoh, kita dapat dengan mudah untuk mengetahui object-object pada gambar dibawah ini.

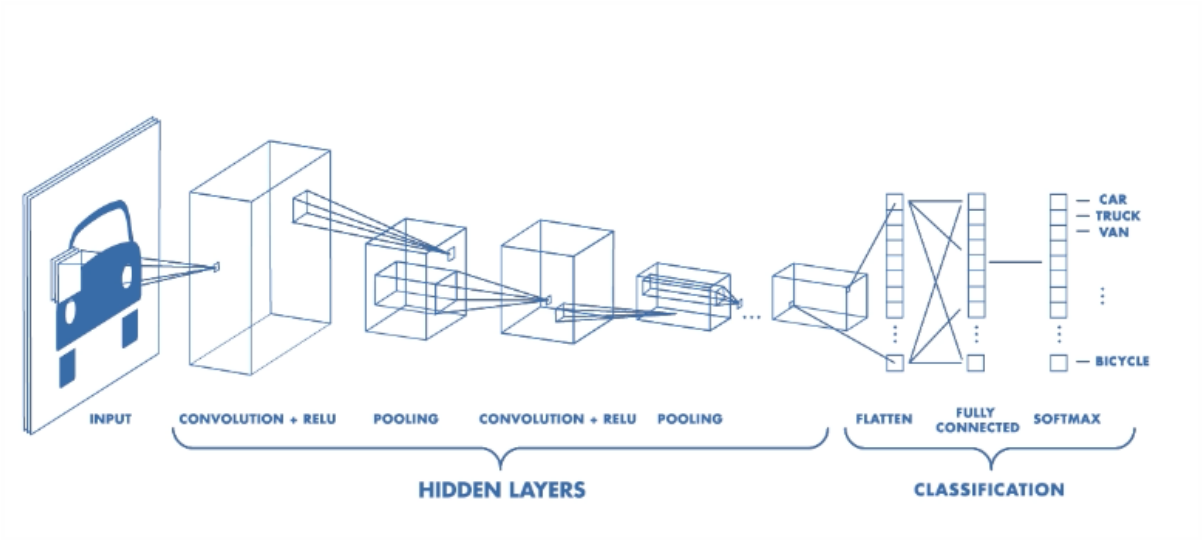
Cara komputer 'melihat' berbeda dengan cara kita melihat. Komputer merupakan mesin yang hanya bisa memproses data berupa angka. Gambar yang dilihat komputer berbentuk 2-Dimensioan array angka yang biasa disebut dengan pixels. Pada gambar hitam putih, komputer melihatnya sebagai matriks 2 dimensi dan gambar yang memiliki warna adalah matriks 2x3 dimana dimensi ketiga adalah RGB (Red, Green, Blue). Gambar dibawah ini adalah representasi bagaimana sebuah komputer melihat gambar sebagai 2-Dimensional array.



Convolutional Neural Network (CNN)

Convolutional Neural Network adalah algoritma Deep Learning yang digunakan untuk memproses inputan data gambar, menentukan kepentingan (bobot dan bias yang dapat dipelajari) ke berbagai aspek dalam gambar dan berfungsi untuk membedakan object satu dengan object lainnya.

CNN dibagi menjadi 2 bagian yaitu Feature Extraction Layer dan Fully-Connected Layer(MLP)



Convolution

Convolution layer atau lapisan konvolusi adalah *core* dari Convolutional Neural Network. Lapisan ini menghasilkan gambar baru yang berisi *features* dari gambar yang telah di-input-kan. Proses yang terjadi pada lapisan ini adalah konvolusi mengaplikasikan filter pada gambar. Filter yang diaplikasikan tersebut adalah berupa matriks bisa ukuran 1x1, 3x3, atau 5x5. Proses konvolusi ini menghasilkan *feature map* yang kemudian akan dipergunakan pada saat lapisan aktivasi (Activation Layer). Ilustrasi dibawah ini menggambarkan proses konvolusi bekerja.

Kita bisa membuat convolutional layer di tensorflow menggunakan **tf.keras.layers.Conv2D.**

```
tf.keras.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format=None,
```

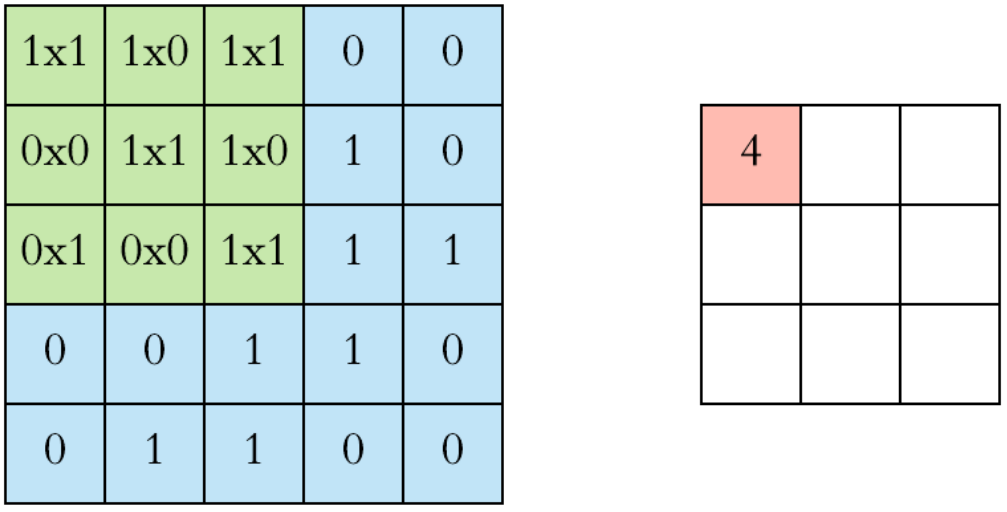
```
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer='glorot_uniform',
    bias_initializer='zeros',
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

Parameter yang biasa digunakan pada umumnya seperti berikut.

- filter
  - Bilangan bulat, dimensi ruang keluaran (yaitu jumlah filter keluaran dalam konvolusi).
- kernel_size
  - Bilangan bulat atau tupel/daftar 2 bilangan bulat, menentukan tinggi dan lebar jendela konvolusi 2D. Dapat berupa bilangan bulat tunggal untuk menentukan nilai yang sama untuk semua dimensi spasial.
- strides
  - Sebuah bilangan bulat atau tupel/daftar dari 2 bilangan bulat, menentukan langkah konvolusi sepanjang tinggi dan lebar. Dapat berupa bilangan bulat tunggal untuk menentukan nilai yang sama untuk semua dimensi spasial. Menentukan nilai langkah apa pun != 1 tidak kompatibel dengan menentukan nilai kecepatan_dilatasi apa pun != 1.
- padding
  - salah satu dari "valid" atau "sama" (peka huruf besar-kecil). "valid" berarti tidak ada padding. "sama" menghasilkan padding dengan nol secara merata ke kiri/kanan atau atas/bawah input. Saat padding="same" dan strides=1, output memiliki ukuran yang sama dengan input.
- activation
  - Fungsi aktivasi untuk digunakan. Jika Anda tidak menentukan apa pun, tidak ada aktivasi yang diterapkan (lihat keras.activations).

Contoh

```
tf.keras.layers.Conv2D(
    32,
    (3,3),
    strides=(1, 1),
    padding='valid',
    activation=’relu’,
```

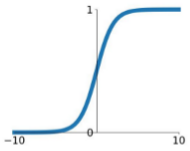


Activation

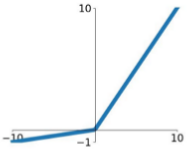
Activation Layer Activation Layer terletak sebelum melakukan *Pooling Layer* dan setelah melakukan *Convolution Layer*. Pada lapisan ini, terjadi proses pengubahan nilai nilai *feature map* pada jarak tertentu tergantung pada *Activation Function* yang dipakai. Tujuan utamanya yaitu untuk meneruskan nilai yang memperlihatkan *dominant feature* dari gambar inputan. Terdapat beberapa *Activation Function* yang kerap dipakai untuk *Convolutional Network*. Berikut adalah beberapa *activation function* tersebut.

Activation Functions

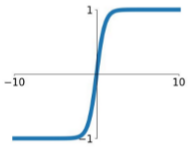
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

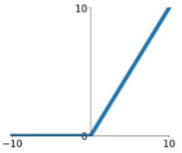


tanh
$\tanh(x)$

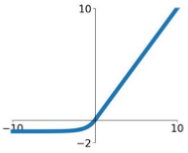


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$

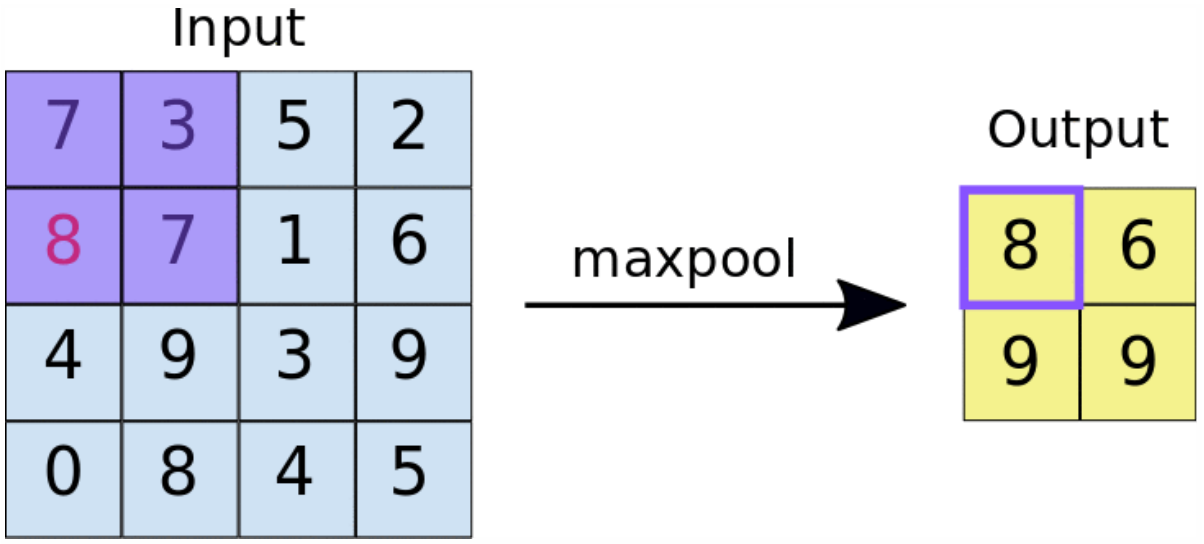


ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

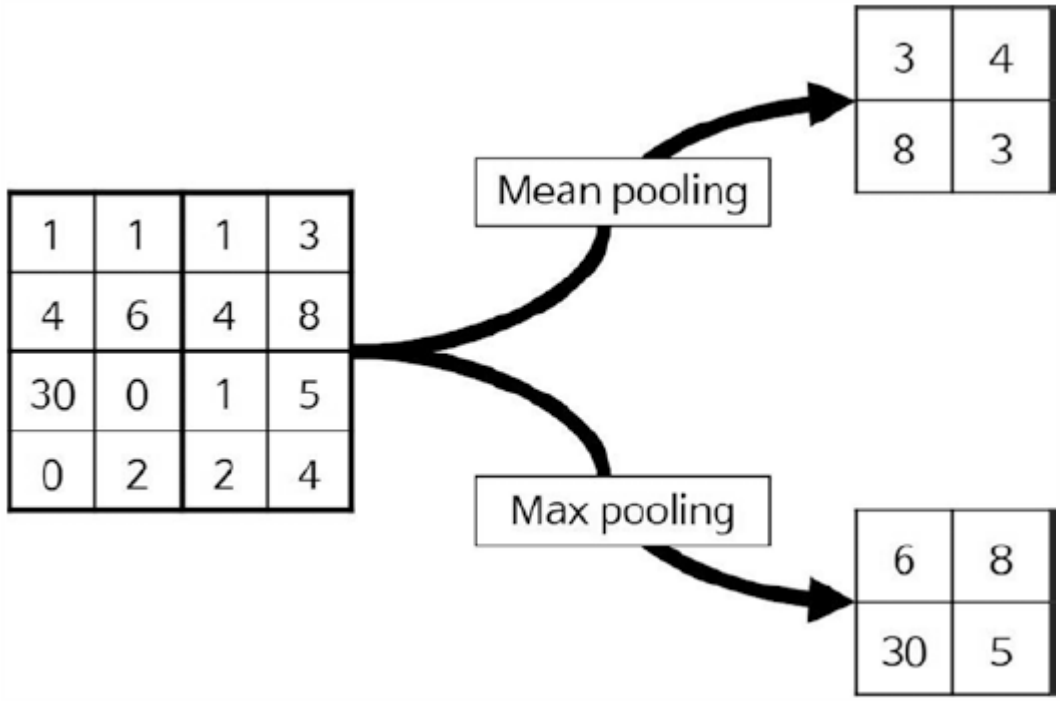


Pooling

Pooling Layer pada *Neural Network*, pada umumnya setelah proses konvolusi selesai, maka akan dilakukan proses *pooling*. Apa itu *pooling*?. *Pooling* adalah proses mengurangi ukuran/resolusi pada gambar akan tetapi tetap mempertahankan informasi penting pada gambar tersebut. *Max pooling* merupakan salah satu contoh *pooling*. *Max pooling* ini bekerja dengan cara mengambil satu piksel pada area dengan luas tertentu pada gambar. Hasil *max pooling* adalah menciptakan gambar baru. Berikut adalah contoh *max pooling* dengan ukuran 2x2 pixel dan gambar yang dilakukan *max pooling* berukuran 4x4 pixel. Hasilnya akan membuat sebuah gambar baru dengan ukuran 2x2.



Terdapat beberapa macam *pooling* lain seperti *max pooling, min pooling, mean pooling, sum pooling*, dll Gambar dibawah ini menunjukan contoh *max pooling* dan *min pooling* pada gambar yang sama.



Di tensorflow kita menggunakan **tf.keras.layers.MaxPooling2D** untuk max pooling, dan **tf.keras.layers.AveragePooling2D.** Parameter yang digunakan untuk menggunakan fungsi ini biasanya hanya menentukan pool_size. pool_size sendiri berisi bilangan bulat atau tuple dari 2 bilangan bulat, ukuran jendela yang akan diambil maksimum. (2, 2) akan mengambil nilai maksimal pada jendela penyatuan 2x2. Jika hanya satu bilangan bulat yang ditentukan, panjang jendela yang sama akan digunakan untuk kedua dimensi.

Contoh

```
tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(pool_size=(2,2))
```

Practice

Contoh pertama menggunakan remote dataset

1. import library yang dibutuhkan

```
import tensorflow as tf

import tensorflow_datasets as tfds

import numpy as np

import matplotlib.pyplot as plt
```

2. Ekstrak dataset dari tfds, kali ini kita menggunakan fashion_mnist. Split data menjadi 3 bagian, train, validation, dan test

```
(train, validation, test), info = tfds.load('fashion_mnist', split=['train[:80%]','train[80%:]','test'], as_supervised=True, with_info=True)

print(info)
```

3. Tampilkan sampel dari dataset

```
tfds.show_examples(train, info)
```

4. Transformasikan dataset yang telah diekstrak seperti normalize, tentukan batch size, dan prefetch

```
def normalize_img(image, label):

    return tf.cast(image, tf.float32)/255 , label



train = train.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)

train = train.cache()

train = train.batch(128)

train = train.prefetch(tf.data.AUTOTUNE)
```

```
validation = validation.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)

validation = validation.cache()

validation = validation.batch(128)

validation = validation.prefetch(tf.data.AUTOTUNE)


test = test.map(normalize_img, num_parallel_calls=tf.data.AUTOTUNE)

test = test.cache()

test = test.batch(128)

test = test.prefetch(tf.data.AUTOTUNE)
```

5. Susun layer sequential menggunakan convolution layer seperti berikut

```
model = tf.keras.Sequential([

        tf.keras.layers.Conv2D(128, (3,3), activation='relu', input_shape=[28, 28, 1]),

    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(256, activation='relu'),

    tf.keras.layers.Dense(10, activation='softmax')

])


model.summary()
```

6. compile model yang telah dibuat

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

7. Lalu latih model dengan epoch sebanyak 10 epoch

```
hasil = model.fit(train, epochs=10, validation_data=validation)
```

8. Model yang telah di latih di evaluasi dengan data test

```
model.evaluate(test)
```

9. Tampilkan line chart untuk melihat pertumbuhan accuracy pada saat model dilatih

```
plt.plot(hasil.history['accuracy'])

plt.plot(hasil.history['val_accuracy'])

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend(['training accuracy', 'validation accuracy'])

plt.show()
```

Source Online

1. https://kotakode.com/blogs/2707/Convolutional-Neural-Network-(CNN)
2. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D