



Modul Praktikum Kecerdasan Buatan



Daftar Isi

Daftar Isi	1
Data Sintetik	2
Data Augmentasi	3
Overfitting	4
Undefitting	5
Implementasi	5
Contoh	8
Source Online	11



TensorFlow – Augmentasi

Data Sintetik

Data sintetik adalah data buatan yang meniru observasi dunia nyata. Biasanya digunakan untuk melatih model machine learning ketika data nyata sulit atau mahal untuk didapatkan. Sebagai contoh data sintetik membuat data dengan karakteristik yang diperoleh dari objek nyata tetapi tidak menggambarkan secara langsung.



Berdasarkan komposisinya, terdapat 2 jenis data sintetik yaitu:

- **Partial Type** adalah data yang terdapat data sintetik dan data asli
- **Full Type** adalah data yang hanya terdapat data sintetik

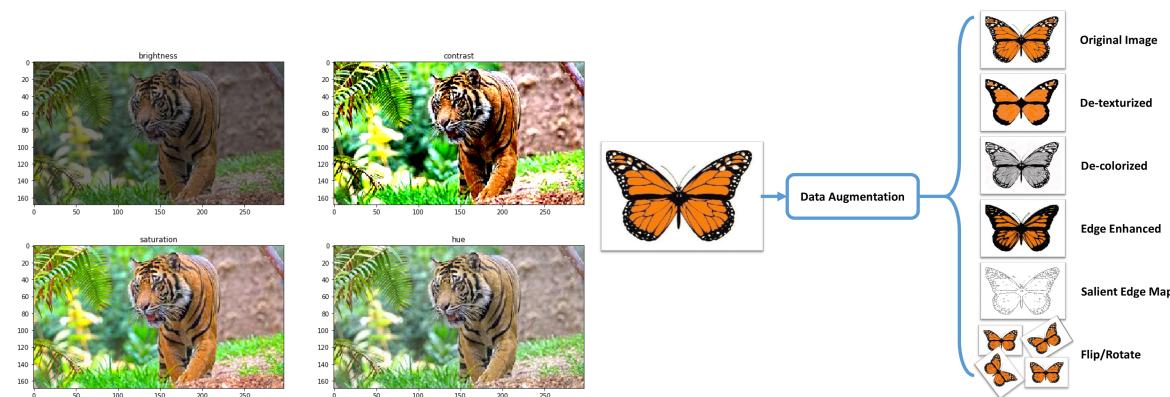
Data sintetik biasa digunakan karena hal-hal berikut.

- Cost and time efficiency ⇒ efisien waktu dan dana
- Exploring rare data ⇒ sejumlah data sulit didapatkan
- Privacy issues resolved ⇒ data sensitif
- Easy labelling and control ⇒ teknikal efficiency



Data Augmentasi

Data augmentasi adalah proses meningkatkan jumlah data dengan membuat data point baru dari data yang ada secara artifisial. Proses meliputi perubahan kecil pada data atau penggunaan model deep learning untuk membuat data point. Secara sederhana data augmentasi adalah proses modifikasi (augmentasi) dataset dengan data tambahan.



Data augmentasi digunakan dalam banyak pendekatan deep learning seperti object detection, image classification, image recognition, natural language understanding, semantic segmentation, dan masih banyak lagi. Data augmentasi meningkatkan performa dan hasil model deep learning dengan membuat instance yang baru dan beragam untuk train set.

Terdapat banyak metode data augmentasi pada image processing yang dapat dilakukan. Berikut merupakan beberapa metode data augmentasi yang terkenal.

- Position Augmentation
 - Center Crop
 - Random Crop
 - Random Vertical Flip
 - Random Horizontal Flip
- Color Augmentation
 - Brightness
 - Contrast
 - Saturation
- Random Augmentation
 - Random Rotation
 - Resize
 - Random Affine
 - Zooming
- Grayscale
 - Grayscale
 - Adding noise

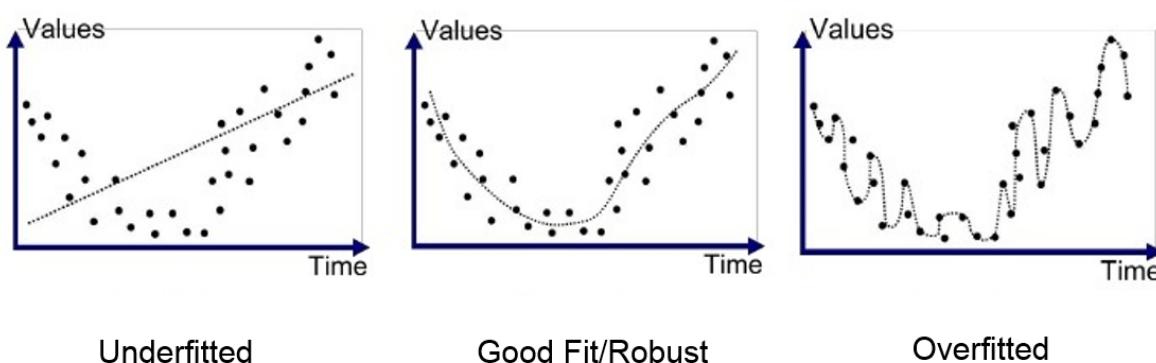
Untuk melakukan data augmentasi dapat digunakan model seperti Adversarial training, GANs, Neural style transfer, Reinforcement learning. Salah satu packages yang biasa digunakan adalah ImageDataGenerator dari Keras.



Overfitting

Overfitting adalah konsep dalam data science yang terjadi ketika model statistik sangat cocok dengan data training. Ketika hal ini terjadi, model tidak dapat bekerja dengan baik pada data yang belum pernah dilihat. Hal ini bertentangan dengan salah satu prinsip machine learning yaitu generalization.

Overfitting dapat terjadi ketika model di-training terlalu lama pada data sampel atau arsitektur model terlalu kompleks. Hal ini membuat model ikut mempelajari noise atau informasi yang tidak relevan pada dataset. Kondisi ini membuat model terlalu memahami, mengingat noise data training, dan tidak dapat men-generalisasi data baru. Kondisi ini membuat model tidak dapat melakukan klasifikasi atau prediksi.



Berikut adalah indikator terjadinya overfitting

- Low error rates and High variance
- Low error rates (train) and High error rates (val)

Cara mengatasi overfitting:

- **Early stopping.** Menghentikan training sebelum model belajar tentang noise
- **Train with more data.** Menambah jumlah data training
- **Data augmentation.** Menambahkan data dengan noise
- **Feature Selection.** Menyederhanakan model
- **Regularization.** Identifikasi dan mengurangi noise data



Underfitting

Underfitting adalah konsep dalam data science ketika model tidak dapat mengidentifikasi relasi antara variable input dan output secara akurat. Hal ini terjadi ketika model terlalu simpel, kurang input feature, terlalu banyak regularization atau training yang dilakukan terlalu sebentar.

Berikut adalah indikator terjadinya underfitting

- High bias and low variance
- High error rate (train&val)

Cara mengatasi underfitting:

- **Decrease regularization.** Mengurangi regularisasi
- **Increase the duration of training.** Meningkatkan durasi training
- **Feature selection.** Menambah kompleksitas model

Implementasi

```
tf.keras.preprocessing.image.ImageDataGenerator()
```

Membuat batch data gambar tensor dengan data augmentasi secara real-time.

Parameter:

- featurewise_center=False
 - Boolean. Mengatur rata-rata input ke 0
- samplewise_center=False
 - Boolean. Mengatur setiap rata-rata sampel ke 0
- featurewise_std_normalization=False
 - Boolean. Membagi input dengan std dataset
- samplewise_std_normalization=False
 - Boolean. Membagi setiap input dengan std nya
- zca_whitening=False
 - Boolean. Menggunakan-nya
- zca_epsilon=1e-06
 - Epsilon
- rotation_range=0
 - Int. Range derajat untuk perputaran random
- width_shift_range=0.0
 - Float Pecahan dari lebar total, jika < 1, pixel jika >= 1
 - 1-D array. Random elemen



- Int. Jumlah pixel
- height_shift_range=0.0
 - Float Pecahan dari lebar total, jika < 1, pixel jika >= 1
 - 1-D array. Random elemen
 - Int. Jumlah pixel
- brightness_range=None
 - Tuple atau list, 2 float. Range nilai perubahan brightness
- shear_range=0.0
 - Float. Intensitas shear
- zoom_range=0.0
 - Float [lowwer, upper]
 - Range. random
- channel_shift_range=0.0
 - Float. Random perubahan channel
- fill_mode='nearest'
 - 'constant', 'nearest', 'reflect', 'wrap'
- cval=0.0
 - Float/Int. Nilai yang digunakan untuk nilai diluar batasan (fill_mode='constant')
- horizontal_flip=False
 - Boolean. Random
- vertical_flip=False
 - Boolean. Random
- rescale=None
 - Int/Float. Nilai scale
- preprocessing_function=None
 - Fungsi yang akan digunakan pada setiap input
- data_format=None
 - 'channels_last' (samples, height, width, channels)
 - 'channels_first' (samples, channels, height, width)
- validation_split=0.0
 - Float. Pecahan data untuk validasi (0-1)
- interpolation_order=1
- dtype=None
 - Tipe data dari array yang akan terbuat

**<datagen_name>.flow_from_directory()**

Mengambil path ke direktori dan membuat batch data augmentasi

- directory
 - String. Path
- target_size=(256, 256)
 - Tuple(Int). (height, width)
- color_mode='rgb'
 - 'grayscale', 'rgb', 'rgba',
- classes=None
 - List. Nama label
- class_mode='categorical'
 - 'categorical' 2D one-hot encoded labels
 - 'binary' 1D binary labels
 - 'sparse' 1D integer labels
 - 'input' autoencoders
- batch_size=32
 - Int. Ukuran batches data
- shuffle=True
 - Boolean. Mengacak data atau tidak
- seed=None
 - Int. Seed random yang digunakan
- save_to_dir=None
 - String. Path tempat menyimpan data yang diaugmentasikan
- save_prefix="
 - String. Prefix dari nama file yang disimpan pada save_to_dir
- save_format='png'
 - 'png', 'jpeg', 'bmp', 'pdf', 'ppn', 'gif', 'tif', 'jpg'
- follow_links=False
 - Boolean. Mengikuti symlink di dalam direktori class
- subset=None
 - String. Subset data ('training' atau 'validation')
- interpolation='nearest'
 - String. resample data jika target size berbeda dari gambar input
 - 'nearest', 'bilinear', 'bicubic', 'lanczos', 'box', 'hamming'
- keep_aspect_ratio=False
 - Boolean. mengubah ukuran gambar ke target_size tanpa distorsi ratio (crop)



Contoh

1. Siapkan dataset, dataset yang kita gunakan yaitu dataset publik kaggle. Dataset dapat di download pada [link ini](#).

The screenshot shows a file explorer interface. At the top, it says "Villains (5 directories)". Below that is a section titled "About this directory" with the text "20 images in each directory. Each with different sizes.". Underneath are five folder icons, each labeled with a villain's name and "20 files": "Darth Vader", "Green Goblin", "Joker", "Thanos", and "Venom".

2. Import library yang dibutuhkan.

```
import tensorflow as tf

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

from keras.preprocessing.image import ImageDataGenerator
```

3. Tampilkan dataset yang akan digunakan

```
class_names = ['Vader', 'Green Goblin', 'Joker', 'Thanos', 'Venom']

for i in range(5):

    paths = os.path.join('Villains', class_names[i], class_names[i]+' 1.jpg')

    if class_names[i]=='Vader':

        vader = 'Darth Vader'

        paths = os.path.join('Villains', vader, class_names[i]+' 1.jpg')

    image = plt.imread(paths)

    plt.subplot(3, 1, 1)
```



```
plt.imshow(image)

plt.show()
```

4. Buat datagen untuk mendeklarasikan ImageDataGenerator

```
IMG_PATH = 'Villains'

img_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=0.45,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

5. Kemudian ambil dataset melalui datagen yang telah dibuat

```
img_generator = img_datagen.flow_from_directory(
    IMG_PATH,
    target_size=(150,150),
    batch_size=32,
    shuffle=True,
    class_mode='categorical',
    save_to_dir='augmented',
    save_prefix='aug' ,
```



```
    save_format='jpg'  
)
```

6. Buat Model Sequential dengan Convolutional Layer

```
model = tf.keras.Sequential([  
  
    tf.keras.layers.Conv2D(256, (3,3), activation='relu',  
    input_shape=[150,150,3]),  
  
    tf.keras.layers.MaxPooling2D(2,2),  
  
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),  
  
    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(128, activation='relu'),  
  
    tf.keras.layers.Dense(5, activation='softmax')  
])
```

7. Compile model menggunakan optimizer Adam, loss function Categorical_crossentropy

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

8. Latih model dengan masukan data generator sebagai inputan data dan atur epoch sampai 10 kali

```
hasil = model.fit(img_generator, epochs=10)
```



Source Online

1. [Data Sintetik](#)
2. [Data Augmentasi](#)
3. [Overfitting](#)
4. [Underfitting](#)