**Tujuan**

Tahap awal sebelum kita membangun sebuah model machine learning adalah mendefinisikan *problem statement* yang ingin kita selesaikan. Pada tahap ini kita menentukan apa masalah yang ingin diselesaikan dan bagaimana implementasi dari model jaringan saraf tiruan dapat menyelesaikan masalah tersebut. Setelah kita memahami masalah, kita dapat mengembangkan model jaringan saraf tiruan sebagai sebuah solusi.

Pada latihan kali ini kita akan membuat sebuah model untuk mengklasifikasi gambar sebuah kamar dan memprediksi apakah kamar tersebut rapi atau berantakan. Pada akhir latihan Anda dapat menguji model yang telah dibuat dengan menggunakan gambar kamar Anda sendiri. Keren, bukan?

**Tahapan Latihan**

Tentunya machine learning selalu membutuhkan data. Pada tahap awal kita perlu memahami dataset yang kita miliki terlebih dahulu. Beberapa hal yang perlu diketahui adalah format dari data, jumlah sampel, dan berapa jumlah label. Selain itu, kita juga perlu memastikan apakah dataset tersebut merupakan data kontinu (masalah regresi) atau data diskrit (masalah klasifikasi).

Dataset yang kita gunakan memiliki 192 sampel data latih yang terdiri dari 96 sampel gambar ruangan rapi dan 96 sampel gambar ruangan berantakan.

[](https://www.dicoding.com/academies/184/tutorials/8527?from=8522)

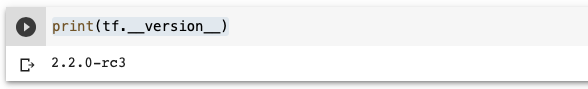
Tahapan latihan kali ini adalah:

1. Memastikan TensorFlow yang digunakan di Google Colab adalah versi di atas 2.0.
2. Mengunduh dataset dan melakukan *extract file* dengan metode *unzip*.
3. Menampung direktori setiap kelas pada direktori train dan validasi ke dalam variabel.
4. *Pre-processing* data dengan *image augmentation*.
5. Mempersiapkan data latih yang akan dipelajari oleh model.
6. Membangun arsitektur model dengan Convolutional Neural Network (CNN).
7. *Compile* dan latih model dengan model.compile dan model.fit hingga mendapatkan akurasi yang diinginkan.
8. Menguji model yang telah dibuat dengan menggunakan gambar yang belum dikenali oleh model.

**Codelab**

Hal pertama yang perlu dilakukan adalah**memastikan** bahwa versi **TensorFlow** yang Anda gunakan adalah versi 2 ke atas.

1. import tensorflow as tf
2. print(tf.\_\_version\_\_)



Tahap selanjutnya adalah mempersiapkan dataset yang akan digunakan. Anda dapat mengunduh dataset tersebut dengan menggunakan perintah berikut.

1. !wget --no-check-certificate \
2. https://dicoding-academy-assets.sgp1.cdn.digitaloceanspaces.com/184/messy-vs-clean-room.zip \
3. -O /tmp/messy\_vs\_clean\_room.zip

Kode di bawah ini berfungsi untuk mengekstrak data yang sebelumnya kita unduh. Lalu kita mendefinisikan nama direktori untuk data latih dan data validasi.

1. # melakukan ekstraksi pada file zip
2. import zipfile,os
3. local\_zip = '/tmp/messy\_vs\_clean\_room.zip'
4. zip\_ref = zipfile.ZipFile(local\_zip, 'r')
5. zip\_ref.extractall('/tmp')
6. zip\_ref.close()
8. base\_dir = '/tmp/images'
9. train\_dir = os.path.join(base\_dir, 'train')
10. validation\_dir = os.path.join(base\_dir, 'val')

Setelah Anda jalankan kode di atas, perhatikanlah, direktori data latih dan data validasi masing-masing memiliki sub-direktori *clean* dan *messy*. Setiap sub-direktori menyimpan gambar yang sesuai dengan nama sub-direktori tersebut. Jadi, pada sub-direktori ‘**clean**’ terdapat gambar-gambar ruangan yang rapi dan pada sub-direktori ‘**messy**’ terdapat gambar-gambar ruangan yang berantakan.

1. os.listdir('/tmp/images/train')
2. os.listdir('/tmp/images/val')



Langkah selanjutnya, kita akan menerapkan ImageDataGenerator untuk data latih dan data validasi. ImageDataGenerator merupakan sebuah fungsi yang sangat berguna untuk mempersiapkan data latih dan data validasi. Beberapa kemudahan yang disediakan ImageDataGenerator antara lain, **preprocessing data**, **pelabelan sampel otomatis**, dan **augmentasi gambar**.

Augmentasi gambar merupakan sebuah teknik yang dapat digunakan untuk memperbanyak data latih dengan cara menduplikasi gambar yang telah ada dengan menambahkan variasi tertentu. Anda akan mempelajari lebih dalam terkait teknik ini pada kelas “Belajar Pengembangan Machine Learning”. Anda juga dapat melihat detail mengenai augmentasi gambar menggunakan ImageDataGenerator pada [tautan](https://keras.io/preprocessing/image/) berikut.

Kode berikut menunjukkan proses augmentasi gambar pada setiap sampel di dataset.

1. from tensorflow.keras.preprocessing.image import ImageDataGenerator
3. train\_datagen = ImageDataGenerator(
4. rescale=1./255,
5. rotation\_range=20,
6. horizontal\_flip=True,
7. shear\_range = 0.2,
8. fill\_mode = 'nearest')
10. test\_datagen = ImageDataGenerator(
11. rescale=1./255)

Selanjutnya, siapkan data latih dan validasi dari kumpulan data gambar yang di-load dalam memori melalui fungsi flow() berikut.

1. train\_generator = train\_datagen.flow\_from\_directory(
2. train\_dir,  # direktori data latih
3. target\_size=(150, 150),  # mengubah resolusi seluruh gambar menjadi 150x150 piksel
4. batch\_size=4,
5. # karena ini merupakan masalah klasifikasi 2 kelas, gunakan class\_mode = 'binary'
6. class\_mode='binary')
8. validation\_generator = test\_datagen.flow\_from\_directory(
9. validation\_dir, # direktori data validasi
10. target\_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 piksel
11. batch\_size=4, # karena ini merupakan masalah klasifikasi 2 kelas gunakan class\_mode = 'binary'
12. class\_mode='binary')

Setelah data siap, kita bisa membangun model Convolutional Neural Network (CNN). Pembuatan model CNN pada keras mirip dengan pembuatan model Multi Layer Perceptron (MLP) yang dibahas pada modul sebelumnya. Perbedaannya terdapat pada empat lapis layer konvolusi dan max pooling.

Anda tentu masih ingat bahwa fungsi dari layer konvolusi adalah untuk mengekstraksi atribut pada gambar. Sedangkan layer max pooling berguna untuk mereduksi resolusi gambar sehingga proses pelatihan model lebih cepat. Nah, pada model CNN, proses klasifikasi gambar hanya berfokus pada atribut-atribut unik yang membedakan tiap kategori. Sehingga, teknik ini dinilai lebih optimal dibandingkan hanya menggunakan model MLP yang membedakan tiap kategori dengan melihat keseluruhan piksel-piksel pada gambar.

1. model = tf.keras.models.Sequential([
2. tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(150, 150, 3)),
3. tf.keras.layers.MaxPooling2D(2, 2),
4. tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
5. tf.keras.layers.MaxPooling2D(2,2),
6. tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
7. tf.keras.layers.MaxPooling2D(2,2),
8. tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
9. tf.keras.layers.MaxPooling2D(2,2),
10. tf.keras.layers.Flatten(),
11. tf.keras.layers.Dense(512, activation='relu'),
12. tf.keras.layers.Dense(1, activation='sigmoid')
13. ])

Usai membuat model, kita bisa menggunakan fungsi summary() untuk melihat summary dari arsitektur model yang telah kita buat.

1. model.summary()

Berikut merupakan hasil summary dari arsitektur model yang telah dibuat.

1. Model: "sequential"
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Layer (type)                 Output Shape              Param #
4. =================================================================
5. conv2d (Conv2D)              (None, 148, 148, 32)      896
6. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
7. max\_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
8. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
9. conv2d\_1 (Conv2D)            (None, 72, 72, 64)        18496
10. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
11. max\_pooling2d\_1 (MaxPooling2 (None, 36, 36, 64)        0
12. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
13. conv2d\_2 (Conv2D)            (None, 34, 34, 128)       73856
14. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
15. max\_pooling2d\_2 (MaxPooling2 (None, 17, 17, 128)       0
16. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
17. conv2d\_3 (Conv2D)            (None, 15, 15, 512)       590336
18. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
19. max\_pooling2d\_3 (MaxPooling2 (None, 7, 7, 512)         0
20. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
21. flatten (Flatten)            (None, 25088)             0
22. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
23. dense (Dense)                (None, 512)               12845568
24. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
25. dense\_1 (Dense)              (None, 1)                 513
26. =================================================================
27. Total params: 13,529,665
28. Trainable params: 13,529,665
29. Non-trainable params: 0

Anda mungkin bertanya-tanya, bagaimana membaca model summary di atas? Jangan khawatir, yuk kita bahas bersama.

Berdasarkan hasil summary di atas, model yang kita buat terdiri dari empat lapis Convolutional dan MaxPoling layer, sebuah flatten layer, serta dua buah dense layer. Ingatlah bahwa dense layer terakhir merupakan output layer. Pada kasus klasifikasi biner, output model merupakan angka tunggal antara 0 dan 1. Sehingga, kita set dense layer terakhir = 1. Sementara itu, kolom “Param #” berisi informasi mengenai jumlah parameter pada tiap layer.

Selanjutnya, kolom “Output Shape” berisi informasi ukuran output yang dihasilkan tiap layer. Jika diperhatikan, ukuran input gambar yang telah didefinisikan sebelumnya adalah sebesar (150, 150). Tapi pada convolutional layer pertama, setiap satu input gambar akan menghasilkan ukuran output (148, 148) sebanyak 32 gambar. Ukuran tersebut berkurang karena kita menggunakan filter dengan ukuran (3, 3) dengan jumlah filter sebanyak 32 filter. Sehingga, tiap satu input gambar akan menghasilkan 32 gambar baru dengan ukuran (148, 148).

Kemudian, resolusi tiap gambar akan diperkecil dengan tetap mempertahankan informasi pada gambar menggunakan MaxPoling layer yang berukuran (2, 2). Hal ini  akan menghasilkan ukuran output gambar sebesar (74, 74). Nah, proses tersebut juga berlaku untuk Convolutional dan MaxPoling layer yang lain.

Berikutnya, mari perhatikan flatten layer. Output dari MaxPoling layer terakhir yang terdiri dari 512 gambar dengan ukuran (7, 7) akan diubah ke dalam bentuk array 1D (tensor 1D). Hal ini  akan menghasilkan output berukuran (25088).

Nah, output tersebut kemudian masuk ke dalam dense layer pertama yang memiliki 512 neuron. Sehingga, ia akan menghasilkan output dengan ukuran (512). Selanjutnya, output ini akan masuk pada dense layer kedua yang memiliki 1 neuron sehingga akan menghasilkan output dengan ukuran (1). Output dari layer terakhir inilah yang digunakan sebagai hasil akhir model untuk kasus klasifikasi biner.

Selamat! Sampai di sini, Anda sudah memahami penjelasan mengenai parameter-parameter dalam model summary. Mari kita lanjutkan proses pembuatan model ke tahap berikutnya.

Setelah membuat arsitektur model CNN, tahap selanjutnya adalah melakukan *compile* model tersebut menggunakan fungsi compile(). Pada tahap ini, kita juga menentukan**loss function** serta **optimizer** yang akan digunakan. Loss function yang digunakan pada kasus klasifikasi biner adalah "binary\_crossentropy". Selain itu, optimizer yang digunakan  pada kasus ini adalah "Adam optimizer". Adam optimizer dipilih karena mudah diterapkan, lebih efisien secara komputasi dan kebutuhan memori yang lebih kecil.

1. # compile model dengan 'adam' optimizer loss function 'binary\_crossentropy'
2. model.compile(loss='binary\_crossentropy',
3. optimizer=tf.optimizers.Adam(),
4. metrics=['accuracy'])

Nah, tahap terakhir dari pembuatan model adalah proses yang disebut sebagai model fitting. Ia merupakan proses untuk melatih model pada data masukan dan label yang bersesuaian. Pada proses ini, kita memasukkan data latih pada jaringan Neural Network yang telah kita buat sebelumnya.

Hal yang harus didefinisikan pada tahap ini adalah loss function dan optimizer. Kemudian, kita mulai proses pelatihan model dengan memanggil fungsi fit().

Dengan menggunakan ImageDataGenerator, kita tidak perlu memasukkan parameter gambar dan labelnya. Image data generator secara otomatis melabeli gambar sesuai dengan direktorinya. Sebagai contoh,  sebuah gambar yang terdapat di direktori *clean*, akan diberi label “clean” oleh ImageDataGenerator secara otomatis.

1. # latih model dengan model.fit
2. model.fit(
3. train\_generator,
4. steps\_per\_epoch=25,  # berapa batch yang akan dieksekusi pada setiap epoch
5. epochs=20, # tambahkan epochs jika akurasi model belum optimal
6. validation\_data=validation\_generator, # menampilkan akurasi pengujian data validasi
7. validation\_steps=5,  # berapa batch yang akan dieksekusi pada setiap epoch
8. verbose=2)

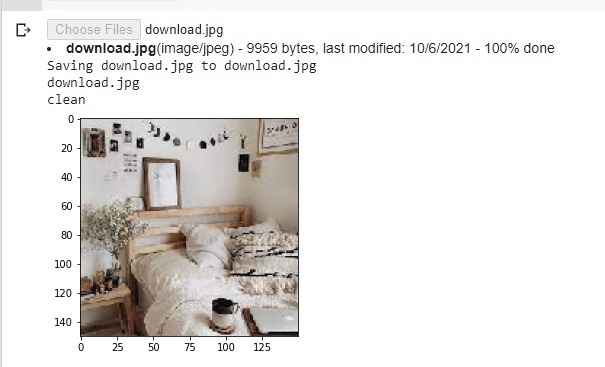
Sampai di sini, proses training telah selesai. Anda telah berhasil membuat model machine learning dengan CNN untuk mengklasifikasi gambar ruangan yang bersih dan berantakan. Selamat!

Setelah berhasil membuat model, Anda tentu ingin menguji model tersebut untuk memprediksi gambar baru (gambar yang belum dikenal oleh model sebelumnya). Potongan program berikut memungkinkan kita secara interaktif memilih sebuah gambar. Kemudian, gambar tersebut akan diolah terlebih dahulu sebelum dimasukkan ke model untuk diprediksi.

Sebelum menerapkan kode ini, pastikan Anda telah memiliki contoh gambar yang akan diuji. Anda dapat mencari dan mengunduh gambar tersebut dari mesin pencari Google. Atau, Anda dapat menggunakan gambar dari [tautan](https://drive.google.com/drive/folders/1_k9DSTSld-9kcJ_aW3vBLkdm3lWYbaze?usp=sharing) berikut.

1. import numpy as np
2. from google.colab import files
3. from tensorflow.keras.preprocessing import image
4. import matplotlib.pyplot as plt
5. import matplotlib.image as mpimg
6. %matplotlib inline
8. uploaded = files.upload()
10. for fn in uploaded.keys():
12. # predicting images
13. path = fn
14. img = image.load\_img(path, target\_size=(150,150))
16. imgplot = plt.imshow(img)
17. x = image.img\_to\_array(img)
18. x = np.expand\_dims(x, axis=0)
19. images = np.vstack([x])
21. classes = model.predict(images, batch\_size=10)
22. print(fn)
23. if classes==0:
24. print('clean')
25. else:
26. print('messy')

Berikut merupakan contoh tampilan hasil prediksi dari model yang sebelumnya telah kita latih. Keren bukan? Silakan foto kamar Anda sendiri dan lihat hasil prediksi dari model yang telah Anda buat.

[](https://www.dicoding.com/academies/184/tutorials/8527?from=8522)