1. Proses pembutan submission
2. Mecari dataset di situs Kaggle, lalu medapa tkan di link : https://www.kaggle.com/datasets/landrykezebou/vcor-vehicle-color-recognition-dataset
3. Mendownload API token kaggle saya
4. Membuat proyek submission pertama dengan tema pendidikan, dan memilih machine learning klasifikasi gambar
5. Membuat program machine learning klasifikasi gambar di colab google

Berikut proses membuat programnya:

1. Install tensorflow

import tensorflow as tf

1. Upload file API kaggle kita

!chmod 600 /content/kaggle.json

1. Download dataset menggunakan API kita

! KAGGLE\_CONFIG\_DIR=/content kaggle datasets download -d landrykezebou/vcor-vehicle-color-recognition-dataset

1. Extraksi file dataset

import zipfile, os, shutil

#ekstraksi file dataset

local\_zip = '/content/vcor-vehicle-color-recognition-dataset.zip'

zip\_ref = zipfile.ZipFile(local\_zip, 'r')

zip\_ref.extractall('/file')

zip\_ref.close()

1. Membuat Direktori dataset

os.mkdir('/tmp/color\_car')

1. Membagi direktori dataset menjadi train (untuk data training) dan val (untuk data validation)

base\_dir = '/tmp/color\_car'

train\_dir = os.path.join(base\_dir, 'train')

validation\_dir = os.path.join(base\_dir, 'val')

1. Membagi direktori masing masing warna pada direktori train

train\_beige\_dir = os.path.join(train\_dir, 'beige')

train\_black\_dir = os.path.join(train\_dir, 'black')

train\_blue\_dir = os.path.join(train\_dir, 'blue')

train\_brown\_dir = os.path.join(train\_dir, 'brown')

train\_yellow\_dir = os.path.join(train\_dir, 'yellow')

train\_gold\_dir = os.path.join(train\_dir, 'gold')

train\_green\_dir = os.path.join(train\_dir, 'green')

train\_grey\_dir = os.path.join(train\_dir, 'grey')

train\_orange\_dir = os.path.join(train\_dir, 'orange')

train\_pink\_dir = os.path.join(train\_dir, 'pink')

train\_purple\_dir = os.path.join(train\_dir, 'purple')

train\_red\_dir = os.path.join(train\_dir, 'red')

train\_silver\_dir = os.path.join(train\_dir, 'silver')

train\_tan\_dir = os.path.join(train\_dir, 'tan')

train\_white\_dir = os.path.join(train\_dir, 'white')

1. Membagi direktori warna pada direktori validation

validation\_beige\_dir = os.path.join(validation\_dir, 'beige')

validation\_black\_dir = os.path.join(validation\_dir, 'black')

validation\_blue\_dir = os.path.join(validation\_dir, 'blue')

validation\_brown\_dir = os.path.join(validation\_dir, 'brown')

validation\_yellow\_dir = os.path.join(validation\_dir, 'yellow')

validation\_gold\_dir = os.path.join(validation\_dir, 'gold')

validation\_green\_dir = os.path.join(validation\_dir, 'green')

validation\_grey\_dir = os.path.join(validation\_dir, 'grey')

validation\_orange\_dir = os.path.join(validation\_dir, 'orange')

validation\_pink\_dir = os.path.join(validation\_dir, 'pink')

validation\_purple\_dir = os.path.join(validation\_dir, 'purple')

validation\_red\_dir = os.path.join(validation\_dir, 'red')

validation\_silver\_dir = os.path.join(validation\_dir, 'silver')

validation\_tan\_dir = os.path.join(validation\_dir, 'tan')

validation\_white\_dir = os.path.join(validation\_dir, 'white')

1. Membuat direktori train, validation dan warna warna

os.mkdir(train\_dir)

os.mkdir(validation\_dir)

os.mkdir(train\_beige\_dir)

os.mkdir(train\_black\_dir)

os.mkdir(train\_blue\_dir)

os.mkdir(train\_brown\_dir)

os.mkdir(train\_yellow\_dir)

os.mkdir(train\_gold\_dir)

os.mkdir(train\_green\_dir)

os.mkdir(train\_grey\_dir)

os.mkdir(train\_orange\_dir)

os.mkdir(train\_pink\_dir)

os.mkdir(train\_purple\_dir)

os.mkdir(train\_red\_dir)

os.mkdir(train\_silver\_dir)

os.mkdir(train\_tan\_dir)

os.mkdir(train\_white\_dir)

os.mkdir(validation\_beige\_dir)

os.mkdir(validation\_black\_dir)

os.mkdir(validation\_blue\_dir)

os.mkdir(validation\_brown\_dir)

os.mkdir(validation\_yellow\_dir)

os.mkdir(validation\_gold\_dir)

os.mkdir(validation\_green\_dir)

os.mkdir(validation\_grey\_dir)

os.mkdir(validation\_orange\_dir)

os.mkdir(validation\_pink\_dir)

os.mkdir(validation\_purple\_dir)

os.mkdir(validation\_red\_dir)

os.mkdir(validation\_silver\_dir)

os.mkdir(validation\_tan\_dir)

os.mkdir(validation\_white\_dir)

1. Memecah file dengan jumlah data validation 20% (dari data training) dan data training dengan train\_test\_split()

from sklearn.model\_selection import train\_test\_split

train\_beige, validation\_beige = train\_test\_split(os.listdir('/file/train/beige'), test\_size=0.2, random\_state=True)

train\_black, validation\_black = train\_test\_split(os.listdir('/file/train/black'), test\_size=0.2, random\_state=True)

train\_blue, validation\_blue = train\_test\_split(os.listdir('/file/train/blue'), test\_size=0.2, random\_state=True)

train\_brown, validation\_brown = train\_test\_split(os.listdir('/file/train/brown'), test\_size=0.2, random\_state=True)

train\_yellow, validation\_yellow = train\_test\_split(os.listdir('/file/train/yellow'), test\_size=0.2, random\_state=True)

train\_gold, validation\_gold = train\_test\_split(os.listdir('/file/train/gold'), test\_size=0.2, random\_state=True)

train\_green, validation\_green = train\_test\_split(os.listdir('/file/train/green'), test\_size=0.2, random\_state=True)

train\_grey, validation\_grey = train\_test\_split(os.listdir('/file/train/grey'), test\_size=0.2, random\_state=True)

train\_orange, validation\_orange = train\_test\_split(os.listdir('/file/train/orange'), test\_size=0.2, random\_state=True)

train\_pink, validation\_pink = train\_test\_split(os.listdir('/file/train/pink'), test\_size=0.2, random\_state=True)

train\_purple, validation\_purple = train\_test\_split(os.listdir('/file/train/purple'), test\_size=0.2, random\_state=True)

train\_red, validation\_red = train\_test\_split(os.listdir('/file/train/red'), test\_size=0.2, random\_state=True)

train\_silver, validation\_silver = train\_test\_split(os.listdir('/file/train/silver'), test\_size=0.2, random\_state=True)

train\_tan, validation\_tan = train\_test\_split(os.listdir('/file/train/tan'), test\_size=0.2, random\_state=True)

train\_white, validation\_white = train\_test\_split(os.listdir('/file/train/white'), test\_size=0.2, random\_state=True)

1. Mengcopy data bagian train dan validation ke direktori masing masing

#mengcopy data bagian train

for filename in train\_beige:

shutil.copy(os.path.join('/file/train/beige', filename), os.path.join(train\_beige\_dir, filename))

for filename in train\_black:

shutil.copy(os.path.join('/file/train/black', filename), os.path.join(train\_black\_dir, filename))

for filename in train\_blue:

shutil.copy(os.path.join('/file/train/blue', filename), os.path.join(train\_blue\_dir, filename))

for filename in train\_brown:

shutil.copy(os.path.join('/file/train/brown', filename), os.path.join(train\_brown\_dir, filename))

for filename in train\_yellow:

shutil.copy(os.path.join('/file/train/yellow', filename), os.path.join(train\_yellow\_dir, filename))

for filename in train\_gold:

shutil.copy(os.path.join('/file/train/gold', filename), os.path.join(train\_gold\_dir, filename))

for filename in train\_green:

shutil.copy(os.path.join('/file/train/green', filename), os.path.join(train\_green\_dir, filename))

for filename in train\_grey:

shutil.copy(os.path.join('/file/train/grey', filename), os.path.join(train\_grey\_dir, filename))

for filename in train\_orange:

shutil.copy(os.path.join('/file/train/orange', filename), os.path.join(train\_orange\_dir, filename))

for filename in train\_pink:

shutil.copy(os.path.join('/file/train/pink', filename), os.path.join(train\_pink\_dir, filename))

for filename in train\_purple:

shutil.copy(os.path.join('/file/train/purple', filename), os.path.join(train\_purple\_dir, filename))

for filename in train\_red:

shutil.copy(os.path.join('/file/train/red', filename), os.path.join(train\_red\_dir, filename))

for filename in train\_silver:

shutil.copy(os.path.join('/file/train/silver', filename), os.path.join(train\_silver\_dir, filename))

for filename in train\_tan:

shutil.copy(os.path.join('/file/train/tan', filename), os.path.join(train\_tan\_dir, filename))

for filename in train\_white:

shutil.copy(os.path.join('/file/train/white', filename), os.path.join(train\_white\_dir, filename))

#mengcopy data bagian validation

for filename in validation\_beige:

shutil.copy(os.path.join('/file/train/beige', filename), os.path.join(validation\_beige\_dir, filename))

for filename in validation\_black:

shutil.copy(os.path.join('/file/train/black', filename), os.path.join(validation\_black\_dir, filename))

for filename in validation\_blue:

shutil.copy(os.path.join('/file/train/blue', filename), os.path.join(validation\_blue\_dir, filename))

for filename in validation\_brown:

shutil.copy(os.path.join('/file/train/brown', filename), os.path.join(validation\_brown\_dir, filename))

for filename in validation\_yellow:

shutil.copy(os.path.join('/file/train/yellow', filename), os.path.join(validation\_yellow\_dir, filename))

for filename in validation\_gold:

shutil.copy(os.path.join('/file/train/gold', filename), os.path.join(validation\_gold\_dir, filename))

for filename in validation\_green:

shutil.copy(os.path.join('/file/train/green', filename), os.path.join(validation\_green\_dir, filename))

for filename in validation\_grey:

shutil.copy(os.path.join('/file/train/grey', filename), os.path.join(validation\_grey\_dir, filename))

for filename in validation\_orange:

shutil.copy(os.path.join('/file/train/orange', filename), os.path.join(validation\_orange\_dir, filename))

for filename in validation\_pink:

shutil.copy(os.path.join('/file/train/pink', filename), os.path.join(validation\_pink\_dir, filename))

for filename in validation\_purple:

shutil.copy(os.path.join('/file/train/purple', filename), os.path.join(validation\_purple\_dir, filename))

for filename in validation\_red:

shutil.copy(os.path.join('/file/train/red', filename), os.path.join(validation\_red\_dir, filename))

for filename in validation\_silver:

shutil.copy(os.path.join('/file/train/silver', filename), os.path.join(validation\_silver\_dir, filename))

for filename in validation\_tan:

shutil.copy(os.path.join('/file/train/tan', filename), os.path.join(validation\_tan\_dir, filename))

for filename in validation\_white:

shutil.copy(os.path.join('/file/train/white', filename), os.path.join(validation\_white\_dir, filename))

1. Image Augmentasi

Image augmentasi adalah teknik untuk memperbanyak data latih tanpa harus mencari data baru. Dengan reslale 1/255, rotation\_range 20, horizontal flip bernilai True, shere\_range 0,2 , fill\_mode ‘wrap’.Programnya sebagai berikut

#Menerapkan Augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

horizontal\_flip=True,

shear\_range = 0.2,

fill\_mode = 'wrap')

test\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

horizontal\_flip=True,

shear\_range = 0.2,

fill\_mode = 'wrap')

1. Image Data Generator

imageDataGenerator() berfungsi untuk memproses data sebelum di load, data yang diproses ialah data latih, dengan target size 150x150, ukuran keseluruhan 32, class mode categorical. berikut programnya

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(150, 150),

batch\_size=32,

class\_mode='categorical')

validation\_generator = test\_datagen.flow\_from\_directory(

validation\_dir,

target\_size=(150, 150),

batch\_size=32,

class\_mode='categorical')

1. Modeling

Pembuatan model menggunakan Keras Sequential, disetial layernya memakai activasi bernilai rule, dikarnakan inputan gambar berupa 2D, dengan output dari model berjumlah 15. berikut progranya :

#membuatan model CNN (Convolutional Neural Network)

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3,3), activation='relu',input\_shape=(150,150,3)),

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

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

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

tf.keras.layers.Dropout(0.3),

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

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

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

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

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(512, activation='relu'),

tf.keras.layers.Dropout(0.5),

tf.keras.layers.Dense(15, activation='softmax')

])

#menampilkan model yang dibuat

model.summary()

1. Compile model

Proses compile, memakai perameter :

1. loss = 'categorical\_crossentropy'
2. optimizer=tf.optimizers.Adam(),
3. metrics=['accuracy'])

berikut programnya:

#mengcompile model yang telah dibuat

model.compile(loss='categorical\_crossentropy',

optimizer=tf.optimizers.Adam(),

metrics=['accuracy'])

1. membuat callback

Callback dibuat dengan tujuan menghindari overfit

Berikut programnya :

"""Membuat callback"""

#menerapkan callback

class Callback(tf.keras.callbacks.Callback):

def on\_epoch\_end(self, epoch, logs={}):

if(logs.get('accuracy') > 0.80):

print('stop => accuracy sudah mencapai 80%')

self.model.stop\_training = True

callbacks = Callback()

1. Melatih Model

#latih model dengan model.fit

riwayat = model.fit(

train\_generator,

steps\_per\_epoch=60,

epochs=30,

validation\_data=validation\_generator,

validation\_steps=27,

verbose = 2,

callbacks=[callbacks]

)

1. Test Model

Data test diambil dari hasil upload oleh pengguna

# Commented out IPython magic to ensure Python compatibility.

from IPython.core import history

import numpy as np

from google.colab import files

from keras.preprocessing import image

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

# %matplotlib inline

uploaded = files.upload()

for fn in uploaded.keys():

# predicting images

path = fn

img = image.load\_img(path, target\_size=(150,150))

imgplot = plt.imshow(img)

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

images = np.vstack([x])

classes = model.predict(images, batch\_size=10)

print(fn)

if classes[0][0] == 1:

print('beige')

elif classes[0][1] == 1:

print('black')

elif classes[0][2] == 1:

print('blue')

elif classes[0][3] == 1:

print('brown')

elif classes[0][4] == 1:

print('gold')

elif classes[0][5] == 1:

print('green')

elif classes[0][6] == 1:

print('grey')

elif classes[0][7] == 1:

print('orange')

elif classes[0][8] == 1:

print('pink')

elif classes[0][9] == 1:

print('purple')

elif classes[0][10] == 1:

print('red')

elif classes[0][11] == 1:

print('silver')

elif classes[0][12] == 1:

print('tan')

elif classes[0][13] == 1:

print('white')

elif classes[0][14] == 1:

print('yellow')

1. Evaluasi

Evaluasi menampilkan grafik hasil latih dari model klasifikasi gambar, grafiknya disajikan berupa grafik accuracy dan garafik loss

import matplotlib.pyplot as plt

plt.plot(riwayat.history['accuracy'])

plt.plot(riwayat.history['val\_accuracy'])

plt.title('accuracy')

plt.ylabel('accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'val'], loc='upper right')

plt.show()

plt.plot(riwayat.history['loss'])

plt.plot(riwayat.history['val\_loss'])

plt.title('Loss Plot')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'val'], loc='upper right')

plt.show()

1. Run all program
2. Download program dengan format (.pynb)
3. Download program dengan format (.py)
4. Buat laporan dengan struktur penulisan yang sesuai dari perintah decoding dengan format (.md)
5. Masukan ke tiga file ke folder lalu kompres ke format (.zip)
6. Upload file ke Submission decoding
7. Referrensi pembuatan submission

<https://medium.com/@hklard/jenis-jenis-performance-metrics-yang-wajib-kamu-ketahui-24d2f1de093d>

<https://ichi.pro/id/panduan-colab-informatif-untuk-memuat-set-data-gambar-dari-github-kaggle-dan-mesin-lokal-129514346641950>

1. Mengapa saya tidak melakukan plagiasi

bahwa kode yang saya tuliskan tidak sama dengan yang dituliskan oleh siswa lainnya, atau sama dengan referensi yang ada diluar modul Dicoding, dikarnakan kode yang saya buat itu hasil referensi dari kode saya sebelumya yang sudah di acc saat tugas submission Proyek Pertama: Membuat Model NLP dengan TensorFlow. Jadi sangat yakin saya tidak melakukan plagiasi.