Laporan Proyek Machine Learning
"Traffic Sign Recognition"

Anggota Kelompok 2:
C14190009 / Gloria I. Endhy
C14190027 / Rebeka Dea L.
C14190073 / Pacquita Mahamaya

Pembagian kerja:

Nama Anggota	Bagian
Gloria I. Endhy	- Mengisi percobaan - Hasil pengujian dan analisa - Mencari dataset bersama - sama - Memodifikasi code bersama - sama - Membuat kesimpulan bersama - sama
Rebeka Dea L.	- Mengisi laporan bagian percobaan - Mencari dataset bersama-sama - Memodifikasi code bersama-sama - Membuat kesimpulan bersama
Pacquita Mahamaya	- Mempelajari cara kerja algoritma neural network - Mencari dataset bersama-sama - Memodifikasi code bersama-sama - Membuat kesimpulan bersama

Teori yang digunakan

Sumber : Apa itu Convolutional Neural Network? | by QOLBIYATUL LINA | Medium

Convolutional Neural Networks Explained (CNN Visualized)

Teori yang kami gunakan adalah Convolutional Neural Network (CNN).Untuk kemudahan penjelasan, beberapa istilah yang akan dipakai yaitu sebagai berikut:

- Channel => nilai dari setiap pixel. Misalnya jika suatu gambar diklasifikasi melalui pencahayaannya saja, gambar dianggap mempunyai 1 channel yang bernilai antara 0-225 dimana 0 relatif tergelap. Jika suatu gambar diklasifikasi melalui RGB, maka gambar tersebut mempunyai 3 channel.
- Kernel/filter=> mini matrix yang berukuran lebih kecil dari resolusi input.
- Feature map => array baru yang menyimpan angka-angka hasil pemindaian kernel terhadap gambar
- Abstraksi => proses penggabungan kernel yang lebih kompleks dengan memanfaatkan feature map yang terbuat sebelumnya.

Secara garis besar, arsitektur dari CNN dapat dibagi menjadi 2 bagian besar, yaitu :

1) Feature Extraction Layer, terbagi menjadi Convolutional Layer dan Pooling Layer

Merupakan lapisan dimana gambar di encoding menjadi features yang berupa angka-angka yang mempresentasikan gambar tersebut. 1 Convolutional layer bisa mempunyai beberapa kernel dan feature maps.

2) Fully-connected Layer

Secara umum, algoritma CNN adalah sebagai berikut :

1) **Memecah gambar menjadi gambar-gambar yang memiliki resolusi lebih kecil (Convolutional Layer)**
2) **Memasukkan setiap gambar yang lebih kecil ke small neural network (Convolutional Layer)**
   Setiap gambar kecil dijadikan input untuk menghasilkan sebuah representasi fitur sederhana. Pada proses ini, kernel akan bergerak sepanjang gambar serta memindainya dengan operasi dot product antara matriks gambar dan matriks kernel.Proses ini dilakukan untuk semua bagian dari masing-masing gambar kecil dengan kernel yang sama. Dengan demikian setiap bagian gambar akan memiliki faktor pengali yang sama (weight sharing). Jika ada sesuatu dari gambar yang menarik, maka bagian itu akan ditandai sebagai object of interest.
3) **Menyimpan hasil dari masing-masing gambar kecil ke dalam feature maps (Convolutional Layer)**
   Setelah selesai memindai gambar, output yang dihasilkan oleh kernel akan disimpan ke feature map. Pada awa
   ]\lnya, tipe kernel yang digunakan lebih simpel dan cenderung geometris untuk mendeteksi sudut, garis, serta pola yang simpel.
4) **Downsampling (Pooling Layer)**
   Pada p3u\oiulzz.,mnbvcr oses ini, bagian feature maps yang tidak penting akan dibuang sehingga kalkulasi kernel di layer-layer berikutnya bisa lebih cepat seiring dengan berkurangnya ukuran setiap image. Tipe pooling yang digunakan adalah max pooling dimana kernel digeser sepanjang feature maps dan nilai pixel terbesar per kernel yang akan disimpan di feature map yang baru.
   Langkah 1- 4 akan diulang untuk membentuk abstraksi. Dengan demikian semakin bertambahnya convolutional layer, kernel yang akan terbentuk menjadi semakin kompleks.
5) **Membuat prediksi (Fully Connected Layer)**
   Inputan pada layer ini adalah gabungan-gabungan features map yang sudah digabung" menjadi satu dan membentuk pola . Pola ini kemudian akan digabung sehingga dapat mengklasifikasi gambar inputan awal, yang pada percobaan ini adalah traffic sign.

Dataset yang digunakan :

1. German Traffic Sign Dataset : GTSRB - German Traffic Sign Recognition Benchmark | Kaggle. Dataset ini terdiri 43 kategori dengan 51 ribu gambar. Untuk percobaan kali ini, kami mengambil maksimal 400 gambar per kategori.
2. Chinese Traffic Sign Dataset : Traffic Sign Recogntion Database (ia.ac.cn). Dataset ini terdiri dari 58 kategori dengan 6 ribu gambar. Karena kategori yang ada lebih banyak namun jumlah gambar lebih sedikit, hanya beberapa kategori yang diambil dari dataset ini. Pada percobaan penggabungan dua dataset, 200 gambar diambil dari dataset China

dan 200 gambar diambil dari dataset Jerman. Jika dalam kategori terdapat kurang dari 200 gambar, maka sisanya akan diambil dari dataset Jerman untuk mencapai jumlah 400.

3. Data test yang kami kumpulkan sendiri dari internet berjumlah 420 gambar yang berisi gambar traffic sign dari berbagai negara.

Percobaan :

1. Menggunakan dataset traffic sign Jerman

   Step 1: Load dataset traffic sign Jerman

```
! pip install kaggle
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets download meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
! unzip gtsrb-german-traffic-sign.zip
```

   Step 2: Import library yang diperlukan

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import pathlib
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import accuracy_score
import  PIL
from PIL import  ImageEnhance , ImageOps , Image
from matplotlib import pyplot
from keras.models import Sequential
from tensorflow.keras.preprocessing import image
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, array_to_img, load_img
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.models import Sequential
from keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.optimizers import Adam
```

   Step 3: Setting untuk gambar plot-nya

```
plt.rcParams["figure.figsize"] = (16,10) #Make the plots bigger by default
plt.rcParams["lines.linewidth"] = 2 #Setting the default line width
plt.style.use("ggplot")
```

Step 4: Proses balancing dataset

Pada dataset traffic sign jerman terdapat 2 folder yaitu Train dan Test. Folder Train yang awalnya kami rencanakan untuk digunakan rupanya tidak sama jumlah gambarnya per kategori. Oleh sebab itu, kami perlu melakukan proses balancing dataset. Proses balancing dataset dimulai dengan cara menambahkan data dari folder Test ke folder Train sehingga total dataset menjadi lebih banyak.

Step 4.1: Membaca file Test.csv

```
data_tambahan = pd.read_csv('Test.csv', delimiter=',', na_values=".")
```

```
data_tambahan.head()
```

	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
0	53	54	6	5	48	49	16	Test/00000.png
1	42	45	5	5	36	40	1	Test/00001.png
2	48	52	6	6	43	47	38	Test/00002.png
3	27	29	5	5	22	24	33	Test/00003.png
4	60	57	5	5	55	52	11	Test/00004.png

Step 4.2: Memindahkan file dari folder Test ke folder Train sesuai kategorinya

```
[17] import shutil

     files = [data_tambahan['Path'].tolist(),data_tambahan['ClassId'].tolist()]

     print(files[0][0])
     i=0
     for f in data_tambahan['Path']:
       shutil.move(f, 'Train/'+str(files[1][i]))
       i+=1
```

```
Test/00000.png
```

Step 4.3: Mencari tahu jumlah kategori terkecil
Dikarenakan jumlah kategori terkecil adalah 270 gambar, kami memutuskan untuk menggunakan 400 gambar tiap kategori agar dataset dapat menjadi lebih seimbang antar kategori.

```
min=len(os.listdir('Train/'+str(1)))
for i in range(43):
  if(min>len(os.listdir('Train/'+str(i)))):
    min=len(os.listdir('Train/'+str(i)))
print(min)
```

```
270
```

Step 5: Menentukan train dataset, image size, dan jumlah kategori
Untuk dataset training, kami menggunakan isi folder Train. Ukuran gambar pada tiap dataset adalah 30x30. Pada dataset Traffic sign jerman terdapat 43 kategori/class.

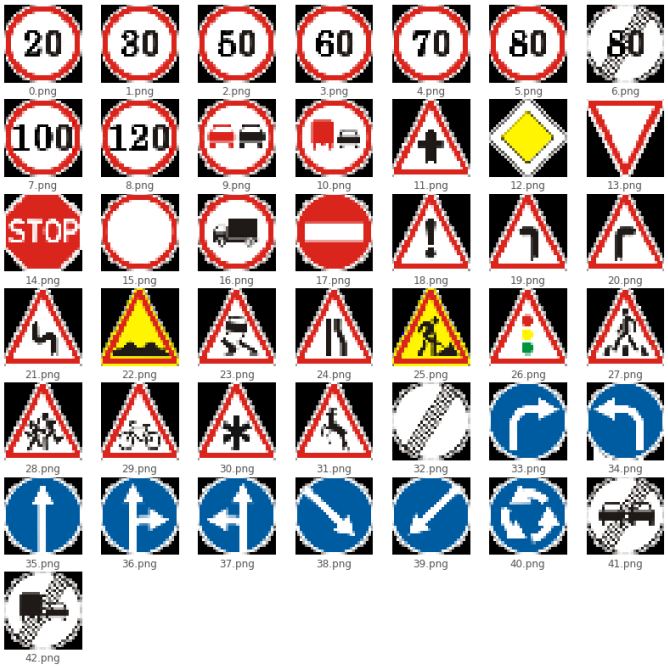
```
train_path = 'Train'
IMG_HEIGHT = 30
IMG_WIDTH = 30
```

```
# Number of Classes
NUM_CATEGORIES = len(os.listdir(train_path))
NUM_CATEGORIES
```

```
43
```

Step 6: Memvisualisasikan jenis-jenis traffic sign yang ada
Kami menggunakan isi folder Meta yang berisi tiap gambar di kategori yang berbeda.

```
# Visualizing all the different signs
plt.figure(figsize=(14,14))
for i in range(NUM_CATEGORIES):
    plt.subplot(7, 7, i+1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    img = load_img('Meta/'+str(i)+'.png', target_size=(IMG_WIDTH, IMG_HEIGHT))
    plt.imshow(img)
    plt.xlabel(str(i)+'.png')
plt.show()
```

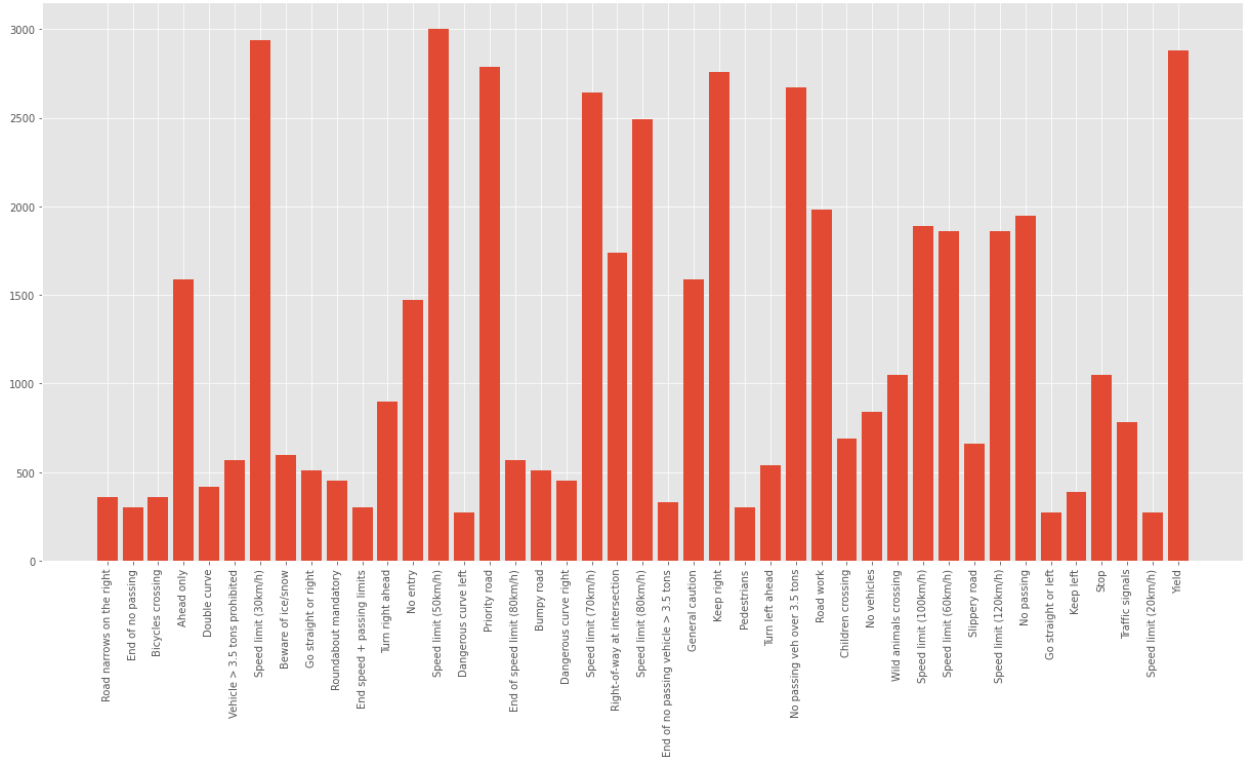


Step 7: Membuat list class/kategori traffic sign
Pada list classes ini total ada 43 kategori.

```
# Label Overview
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Vehicle > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
            30:'Beware of ice/snow',
            31:'Wild animals crossing',
            32:'End speed + passing limits',
            33:'Turn right ahead',
            34:'Turn left ahead',
            35:'Ahead only',
            36:'Go straight or right',
            37:'Go straight or left',
            38:'Keep right',
            39:'Keep left',
            40:'Roundabout mandatory',
            41:'End of no passing',
            42:'End of no passing vehicle > 3.5 tons' }
```

Step 8: Memvisualisasikan jumlah gambar pada tiap kategori
Dari plot tersebut, dapat dilihat bahwa walaupun sudah menggabungkan isi folder Test ke isi folder Train, jumlah dataset per kategori tetap tidak balance. Oleh sebab itu, kami memutuskan untuk menggunakan 400 gambar per kategori.

```
folders = os.listdir(train_path)


train_number = []
class_num = []

for folder in folders:
    train_files = os.listdir(train_path + '/' + folder)
    train_number.append(len(train_files))
    class_num.append(classes[int(folder)])



plt.figure(figsize=(21,10))
plt.bar(class_num, train_number)
plt.xticks(class_num, rotation='vertical')
plt.show()
```

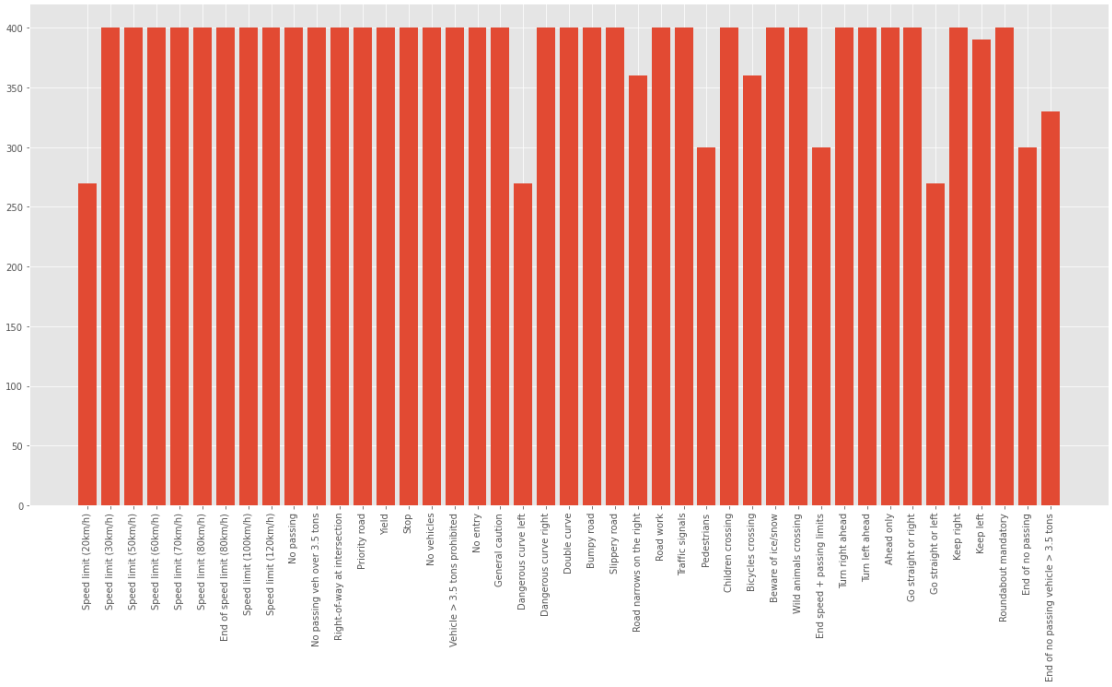


Step 9: Membuat fungsi load_data
Fungsi load_data berguna untuk melakukan load data dari file-file pada folder Train sehingga menghasilkan 2 list yaitu, images dan labels. List images digunakan untuk

menyimpan data image tiap file sedangkan list labels digunakan untuk menyimpan label/kategori tiap file. Pada fungsi ini, juga menghasilkan plot yang memvisualisasikan hasil balancing dataset.

```
def load_data(data_dir):
    images = list()
    labels = list()
    jml_data = list()
    class_num  = list()
    for category in range(NUM_CATEGORIES):
        categories = os.path.join(data_dir, str(category))
        class_num.append(classes[category])
        counter=0
        for img in os.listdir(categories):
            if(counter<400):
              img = load_img(os.path.join(categories, img), target_size=(IMG_HEIGHT, IMG_WIDTH))
              image = img_to_array(img)
              images.append(image)
              labels.append(category)
              counter+=1
            else:
              break
        jml_data.append(counter)
    print(len(jml_data))
    plt.figure(figsize=(21,10))
    plt.bar(class_num, jml_data)
    plt.xticks(class_num, rotation='vertical')
    plt.show()
    return images, labels
```

```
images, labels = load_data(train_path)
```



Step 10: Mengubah list labels menjadi categorical, Membagi dataset menjadi data train dan data test, dan Melakukan normalisasi pada data train dan data test
Dataset dibagi menjadi 30% untuk data test dan 70% untuk data train.

```
labels = to_categorical(labels)
```

```
x_train, x_test, y_train, y_test = train_test_split(np.array(images), labels, test_size=0.3)
```

```
x_train/=255
x_test/=255
print('x_train shape:',x_train.shape)
print('Number of images in x_train',x_train.shape[0])
print('Number of images in x_test',x_test.shape[0])
```

```
x_train shape: (11445, 30, 30, 3)
Number of images in x_train 11445
Number of images in x_test 4905
```

Step 11: Membuat Model CNN

Model CNN yang kami gunakan terdiri dari 3 layer.

```
input_shape=( IMG_HEIGHT, IMG_WIDTH, 3)
```

```
model = Sequential()

# First Convolutional Layer
model.add(Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=input_shape))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))

# Second Convolutional Layer
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))

# Third Convolutional Layer
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))

model.add(Flatten())
model.add(Dense(units=64, activation='relu'))
model.add(Dense(NUM_CATEGORIES, activation='softmax'))

# Compiling the model


lr = 0.001
epochs = 30
model.compile(loss='categorical_crossentropy',optimizer="adam",metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 32)        896

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 dropout (Dropout)           (None, 14, 14, 32)        0

 conv2d_1 (Conv2D)           (None, 12, 12, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 6, 6, 64)          0

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 43)                2795

=================================================================
Total params: 124,715
Trainable params: 124,715
Non-trainable params: 0
```

Step 12: Melakukan training data dan menyimpan history-nya

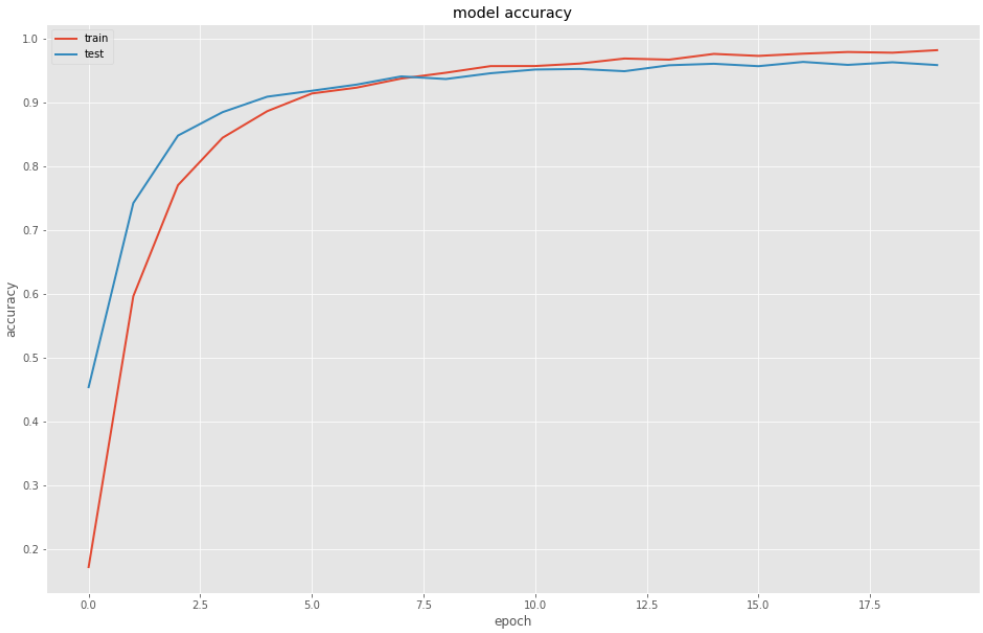
```
history = model.fit(x_train, y_train,validation_split=0.3,epochs=40)
```

Step 13: Evaluasi model dengan data test dan Membuat plot history akurasi model

```
loss, accuracy = model.evaluate(x_test, y_test)

print('test set accuracy: ', accuracy * 100)
```

```
486/486 [==============================] - 7s 13ms/step - loss: 0.0666 - accuracy: 0.9883
test set accuracy:  98.82973432540894
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Step 14: Evaluasi model dengan data test dari internet

```
loss, accuracy = model.evaluate(images2, labels2)

print('test set dari internet accuracy: ', accuracy * 100)
```

```
14/14 [==============================] - 0s 10ms/step - loss: 2209.3918 - accuracy: 0.4238
test set dari internet accuracy:  42.38095283508301
```

2. Menggunakan dataset traffic sign gabungan China dan Jerman
   Dalam step ini kami hanya mengubah dan menambah beberapa dari coding Jerman sebelumnya :
   Step 1 : Memasukan dataset tambahan yaitu dataset China (TSRD) ke dalam project

```
! mkdir 'TSRD-Test Annotation'
! mkdir 'TSRD-Train Annotation'
! mkdir 'TSRD-Test'
! mkdir 'tsrd-train'
```

```
[ ] ! unzip 'TSRD-Test Annotation.zip' -d 'TSRD-Test Annotation'

    unzip:  cannot find or open TSRD-Test Annotation.zip, TSRD-Test Annotation.zip.zip or TSRD-Test Annotation.zip.ZIP.
```

```
[ ] ! unzip 'TSRD-Train Annotation.zip' -d 'TSRD-Train Annotation'

    unzip:  cannot find or open TSRD-Train Annotation.zip, TSRD-Train Annotation.zip.zip or TSRD-Train Annotation.zip.ZIP.
```

```
[ ] ! unzip 'TSRD-Test.zip' -d 'TSRD-Test'

    unzip:  cannot find or open TSRD-Test.zip, TSRD-Test.zip.zip or TSRD-Test.zip.ZIP.
```

Step 2 : Membaca dataset

```
[ ] data_china = pd.read_csv('TSRD-Train Annotation/TsignRecgTrain4170Annotation.txt', delimiter=';', na_values=".",header=None)
```

Step 3 : Kami ingin melihat setiap kategori yang dimiliki jadi kami memisah dataset ke dalam kategori, karena dalam dataset yang kami gunakan ini belum terbagi dalam kategori, lalu mengambil 1 gambar dari setiap kategori.

```
[ ]  file_unique = []
     for i in range(58):
       sudah_ada = 0
       for j in range(len(data_china[7])):

         if(data_china[7][j]==i and sudah_ada==0):
           file_unique.append(data_china[0][j])
           sudah_ada=1
```

Lalu kami ingin melihat gambar dari setiap kategori itu, dengan load gambar yang sebelumnya sudah kami simpan namanya.

```
[ ]  # Visualizing all the different signs in the china dataset
     plt.figure(figsize=(14,14))
     index_sebelum = 0
     for i in range(58):
         plt.subplot(8, 8, i+1)
         plt.grid(False)
         plt.xticks([])
         plt.yticks([])
         img = load_img('tsrd-train/'+str(file_unique[i]), target_size=(120, 120))
         plt.imshow(img)
         plt.xlabel(str(i))
     plt.show()
```

Step 4 : Membuat folder yang akan diisi gabungan dataset, lalu membuat folder sesuai banyak kategori

```
[ ]  ! mkdir 'new_mix_train_fixed'
```

```
[ ]  for i in range(43):
       os.makedirs("new_mix_train_fixed/"+str(i), exist_ok=True)
```

Step 5 : Untuk pengambilan data berbeda dengan coding yang hanya menggunakan dataset Jerman sebelumnya, jadi mengambil 200 dari setiap kategori dari dataset Jerman, lalu dimasukan ke dalam folder yang sudah dibuat.

```
[ ]  import shutil
     for c in range(43):
       count=0
       for f in (os.listdir('Train/'+str(c))):
         if(count<200):
           shutil.copy('Train/'+str(c)+'/'+f, 'new_mix_train/'+str(c))
           count+=1
         else:
           break
```

Step 6 : Karena tidak semua dataset China sama dengan dataset Jerman, kami hanya mengambil data yang sama dengan kategori di Jerman, jadi kami akan membuat list dimana kami dapat mengetahui kategori China mana yang sama dengan kategori di Jerman. Sebagai contoh kategori 1 Jerman sama dengan kategori 2 di China.

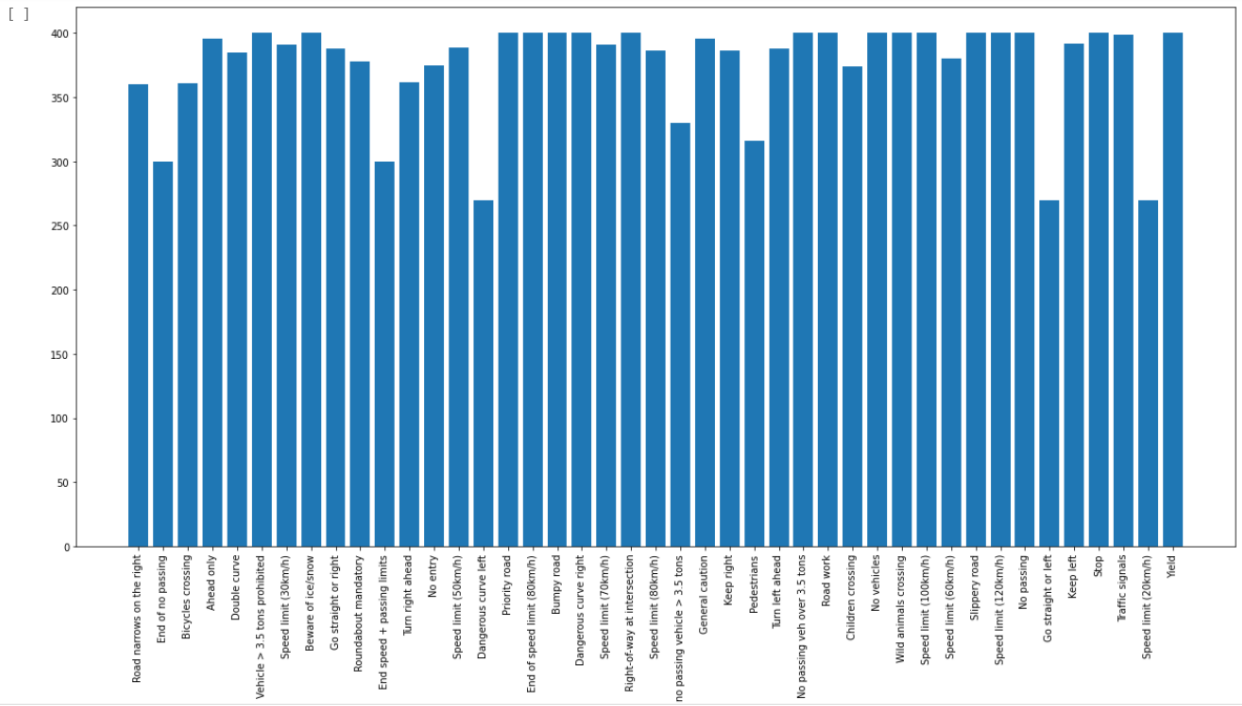
```
[ ]  id_china =[2,4,5,6,7,53,55,34,46,48,33,35,37,36,24,22,21,20,26,25,27]
     id_jerman=[1,2,3,4,5,15,17,18,21,25,26,27,28,29,33,34,35,36,38,39,40]
```

Step 7 : Mengambil 200 data dari setiap kategori di China sesuai list yang sudah dibuat.

```
[ ]  import shutil
     for f in range(len(data_china[0])):
       counter=0
       for x in range(len(id_china)):
         if(data_china[7][f]==id_china[x] and counter<200):
           shutil.copy('tsrd-train/'+str(data_china[0][f]), 'new_mix_train_fixed/'+str(id_jerman[x]))
           counter+=1
           break
```

Step 8 : Lalu untuk melihat grafik jumlah dataset yang sudah diambil, kurang lebih sama seperti di coding percobaan sebelumnya.

```
[ ]  train_path = 'new_mix_train_fixed'
     folders = os.listdir(train_path)


     train_number = []
     class_num = []

     for folder in folders:
         train_files = os.listdir(train_path + '/' + folder)
         train_number.append(len(train_files))
         class_num.append(classes[int(folder)])



     plt.figure(figsize=(21,10))
     plt.bar(class_num, train_number)
     plt.xticks(class_num, rotation='vertical')
     plt.show()
```



Step 9 : Load data, sama dengan percobaan sebelumnya

```
def load_data(data_dir):


    images = list()
    labels = list()
    for category in range(NUM_CATEGORIES):
        categories = os.path.join(data_dir, str(category))
        for img in os.listdir(categories):
            img = load_img(os.path.join(categories, img), target_size=(IMG_HEIGHT, IMG_WIDTH))
            image = img_to_array(img)
            images.append(image)
            labels.append(category)

    return images, labels
```

```
[ ]  images, labels = load_data(train_path)
```

Step 10 : Lalu dilakukan train dan pemodelan dengan cara yang sama dengan percobaan sebelumnya

Step 11 : Lalu dilakukan evaluasi, dan didapatlah akurasi sebesar 95,7 untuk data test dari dataset dan akurasi sebesar 44,5 untuk data test dari internet.

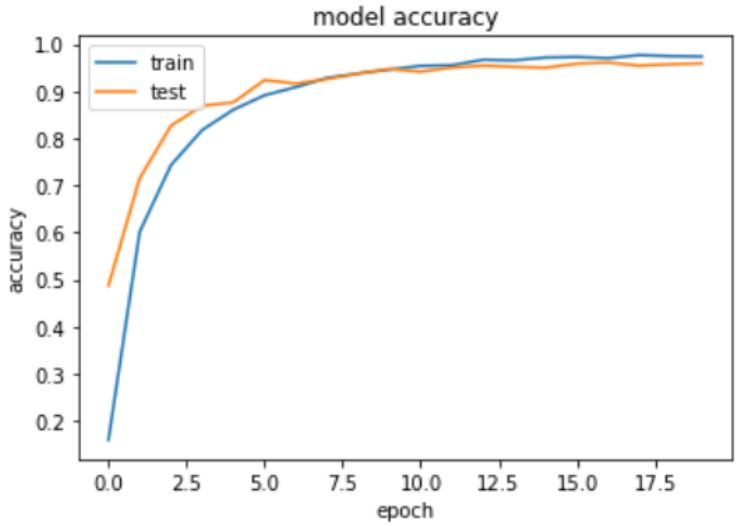
```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
152/152 [==============================] - 2s 13ms/step - loss: 0.1764 - accuracy: 0.9572
test set accuracy:  95.72314023971558
```

```
loss, accuracy = model.evaluate(images2, labels2)

print('test set dari internet accuracy: ', accuracy * 100)
```

```
14/14 [==============================] - 0s 12ms/step - loss: 1394.7095 - accuracy: 0.4452
test set dari internet accuracy:  44.52380836009979
```

```
[ ]  plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('model accuracy')
     plt.ylabel('accuracy')
     plt.xlabel('epoch')
     plt.legend(['train', 'test'], loc='upper left')
     plt.show()
```



3. Penggantian epoch
   Epoch 20, 30, 40

Epoch 20 :

- Pada percobaan yang menggunakan dataset Jerman, didapatkan akurasinya 96,8

```
history = model.fit(x_train, y_train,validation_split=0.3,epochs=20)
```

```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
154/154 [==============================] - 2s 14ms/step - loss: 0.1333 - accuracy: 0.9680
test set accuracy:  96.7991828918457
```

- Pada percobaan yang menggunakan dataset Jerman dan China, didapatkan akurasinya 95,7

```
[ ]  history = model.fit(x_train, y_train,validation_split=0.3,epochs=20)
```

```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
152/152 [==============================] - 2s 13ms/step - loss: 0.1764 - accuracy: 0.9572
test set accuracy:  95.72314023971558
```

Epoch 30 :

- Pada percobaan yang menggunakan dataset Jerman, didapatkan akurasinya 96,9

```
[ ]  history = model.fit(x_train, y_train,validation_split=0.3,epochs=30)
```

```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
154/154 [==============================] - 2s 12ms/step - loss: 0.1251 - accuracy: 0.9688
test set accuracy:  96.88073396682739
```

- Pada percobaan yang menggunakan dataset Jerman dan China, didapatkan akurasinya 97,7

```
[ ]  history = model.fit(x_train, y_train,validation_split=0.3,epochs=30)
```

```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
155/155 [==============================] - 2s 11ms/step - loss: 0.1243 - accuracy: 0.9768
test set accuracy:  97.67583012580872
```

Epoch 40 :

- Pada percobaan yang menggunakan dataset Jerman, didapatkan akurasinya 97,3

```
[ ]  history = model.fit(x_train, y_train,validation_split=0.3,epochs=40)
```

```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
154/154 [==============================] - 2s 12ms/step - loss: 0.1461 - accuracy: 0.9729
test set accuracy:  97.28848338127136
```

- Pada percobaan yang menggunakan dataset Jerman dan China, didapatkan akurasinya 98,8

```
[ ]  history = model.fit(x_train, y_train,validation_split=0.3,epochs=40)
```

```
[ ]  loss, accuracy = model.evaluate(x_test, y_test)

     print('test set accuracy: ', accuracy * 100)
```

```
486/486 [==============================] - 7s 14ms/step - loss: 0.0666 - accuracy: 0.9883
test set accuracy:  98.82973432540894
```

4. Percobaan perbedaan jumlah data training

   Percobaan ini bertujuan untuk mengetahui bagaimana pengaruh jumlah data training terhadap akurasi model.

   Step 1: Melakukan sorting pada data agar dapat mengetahui jumlah data per kategori secara berurutan sehingga mempermudah untuk membagi grup kategori.

```
[65] sorted_class = [x for _,x in sorted(zip(train_number,class_num))]
     sorted_train_number = sorted(train_number)
     sorted_data = {'Kategori':sorted_class,'Jumlah Data':sorted_train_number}
     df_sorted_data = pd.DataFrame(sorted_data)
     print(df_sorted_data.to_string(index=False))
```

```
 Kategori  Jumlah Data
        0          210
       19          210
       37          210
       27          240
       32          240
       41          240
       42          240
       24          270
```

Step 2: Mengelompokkan kategori dan memberikan batas jumlah data pada tiap grup kategori. Kami membagi menjadi 3 grup kategori yaitu sebagai berikut:

- Grup 1 berisi 15 kategori dengan jumlah data 210 per kategori
- Grup 2 berisi 14 kategori dengan jumlah data 420 per kategori
- Grup 3 berisi 14 kategori dengan jumlah data 1320 per kategori

Step 2.1: Membuat folder grup kategori dan mengisi folder tersebut dengan isi data gambar grup kategori.

```
! mkdir 'Grup 1'
! mkdir 'Grup 2'
! mkdir 'Grup 3'
```

```
import shutil
counter=0
for folder in sorted_class:
    if(counter<15):
      os.makedirs(os.path.dirname('Grup 1/'+str(folder)), exist_ok=True)
      shutil.copytree('Train/'+str(folder),'Grup 1/'+str(folder))
      counter+=1
    elif(counter<29):
      os.makedirs(os.path.dirname('Grup 2/'+str(folder)), exist_ok=True)
      shutil.copytree('Train/'+str(folder),'Grup 2/'+str(folder))
      counter+=1
    else:
      os.makedirs(os.path.dirname('Grup 3/'+str(folder)), exist_ok=True)
      shutil.copytree('Train/'+str(folder),'Grup 3/'+str(folder))
      counter+=1
```

Step 3: Membuat fungsi load_data

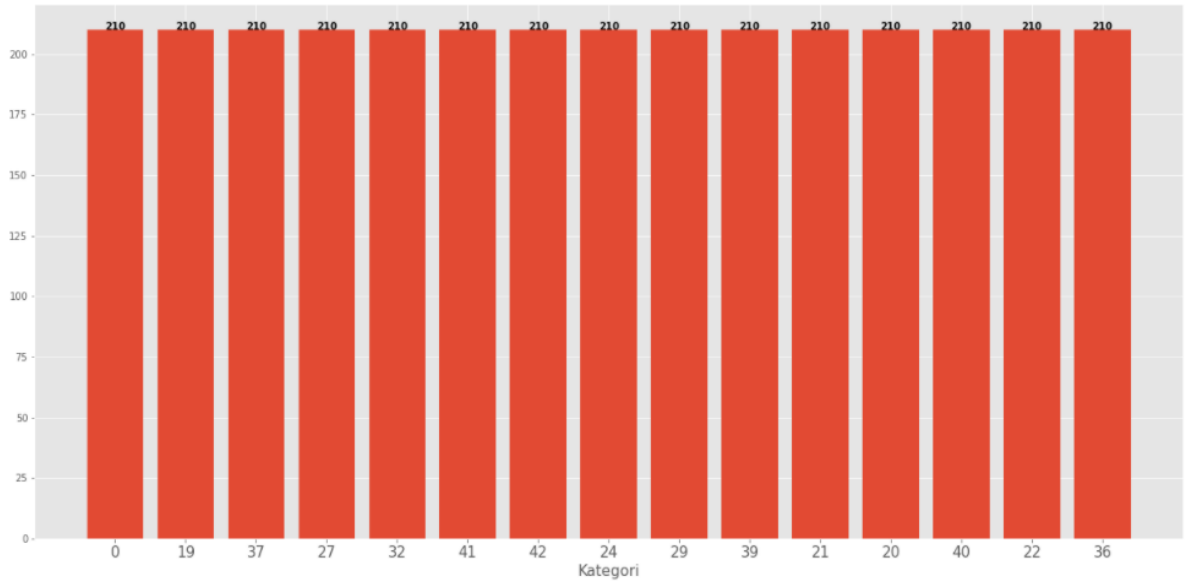
Fungsi load_data berguna untuk melakukan load data dari file-file pada path folder yang di input sehingga menghasilkan 2 list yaitu, images dan labels. List images digunakan untuk menyimpan data image tiap file sedangkan list labels digunakan untuk menyimpan label/kategori tiap file. Pada fungsi ini, juga menghasilkan plot yang memvisualisasikan jumlah data yang di load per kategori.

```
def load_data(data_dir, data_num, group):
    images = list()
    labels = list()
    jml_data = list()
    class_num  = list()
    group_class=list()

    if(group==-1):
      group_class=list(map(str,range(0,43)))
    elif(group==1):
      group_class=sorted_class[:15]
    elif(group==2):
      group_class=sorted_class[15:29]
    else:
      group_class=sorted_class[29:43]

    for category in group_class:
        categories = os.path.join(data_dir, str(category))
        class_num.append(category)
        counter=0
        for img in os.listdir(categories):
            if(counter<data_num or data_num==-1):
              img = load_img(os.path.join(categories, img), target_size=(IMG_HEIGHT, IMG_WIDTH))
              image = img_to_array(img)
              images.append(image)
              labels.append(category)
              counter+=1
            else:
              break
        jml_data.append(counter)
    print("Jumlah Kategori dalam " + data_dir + ": "+str(len(jml_data)))
    fig, ax = plt.subplots(figsize=(21,10))
    ax.bar(class_num, jml_data)
    plt.xticks(class_num, rotation='horizontal',fontsize=15)
    plt.xlabel("Kategori",fontsize=15)
    for i, v in enumerate(jml_data):
        ax.text(i,v, str(v), ha='center', fontweight='bold')
    plt.show()
    return images, labels
```

Step 4: Melakukan load data untuk data training dan data test (data test dari data training yang di-split dan juga data test yang kami cari sendiri dari internet)

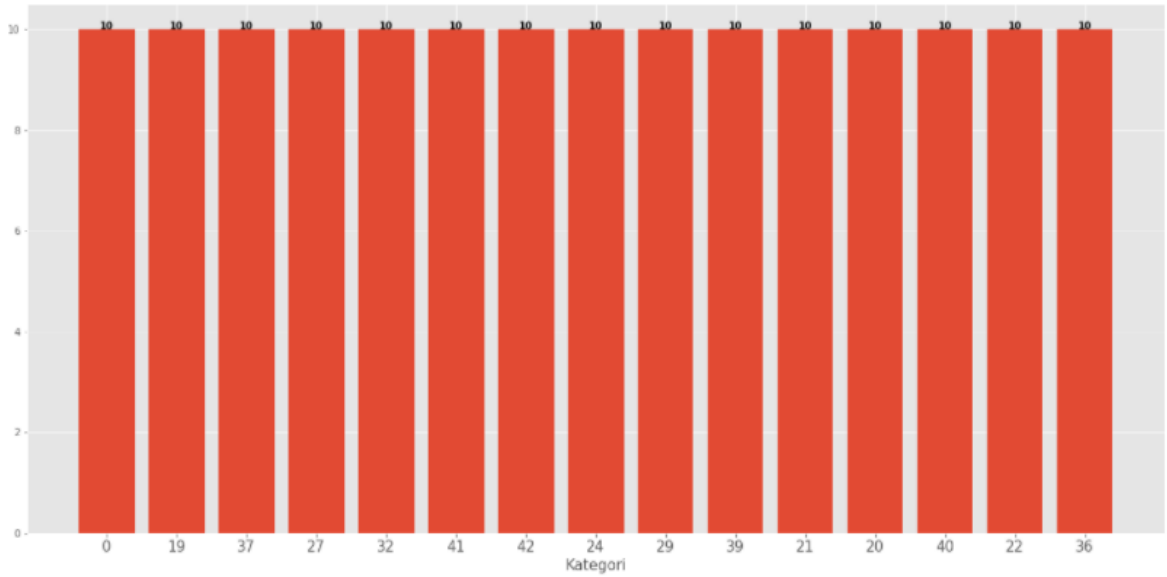
```
[127] images, labels = load_data('Grup 1',210,1)
```

```
Jumlah Kategori dalam Grup 1: 15
```




```
[149] images2, labels2 = load_data('gdrive/MyDrive/Data test Grup 1 (dari internet)',-1,1)
```

```
Jumlah Kategori dalam gdrive/MyDrive/Data test Grup 1 (dari internet): 15
```

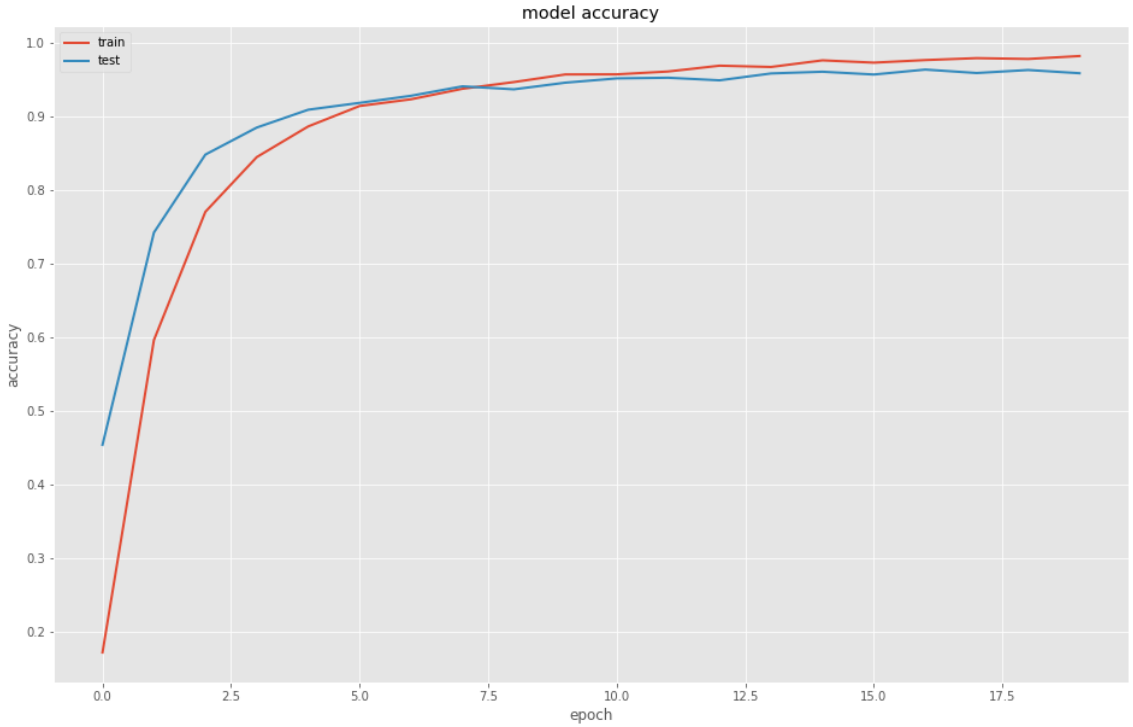


Step 5: Membuat model, melakukan training, dan evaluasi model sama seperti percobaan yang sebelumnya.

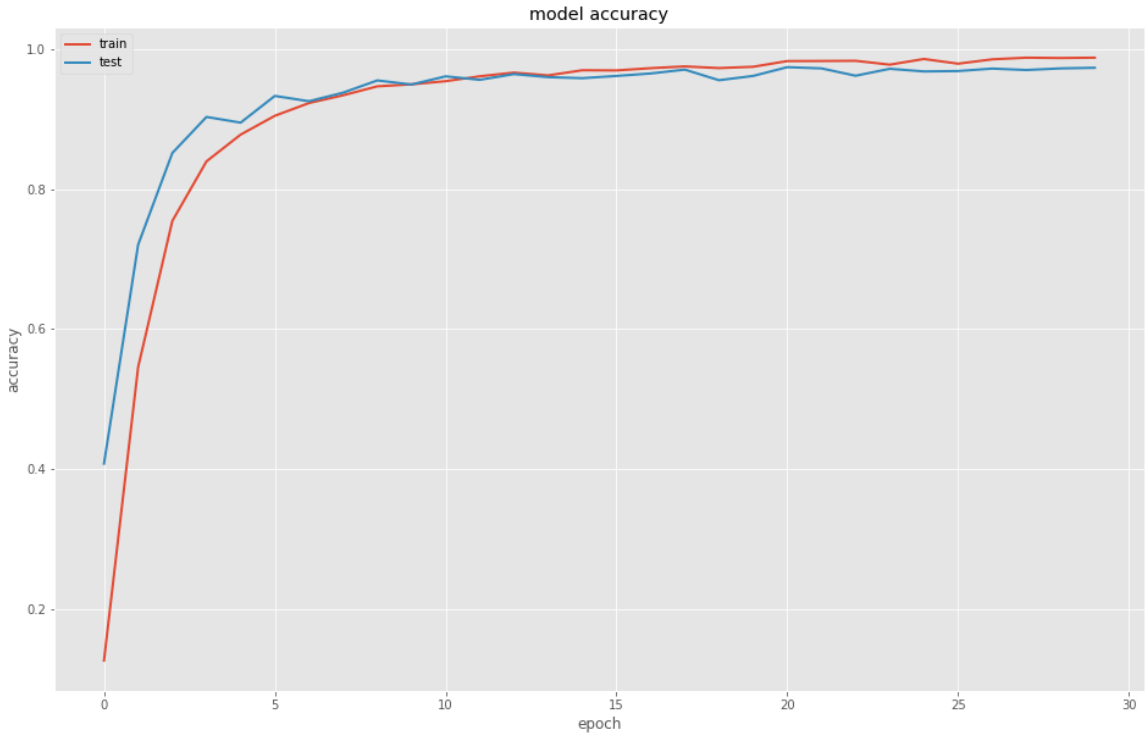
Hasil pengujian dan analisa :

Grafik Akurasi:

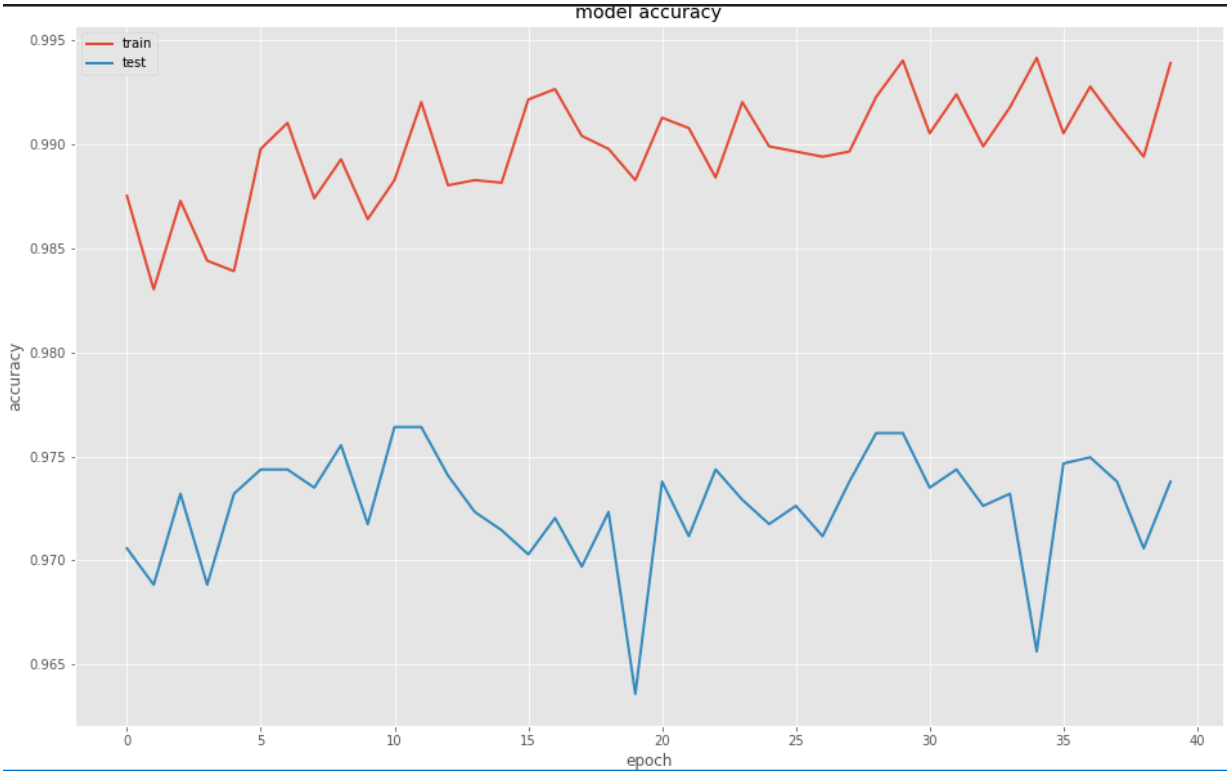
1. Percobaan dengan dataset Jerman sebagai data training
    a. Epoch 20



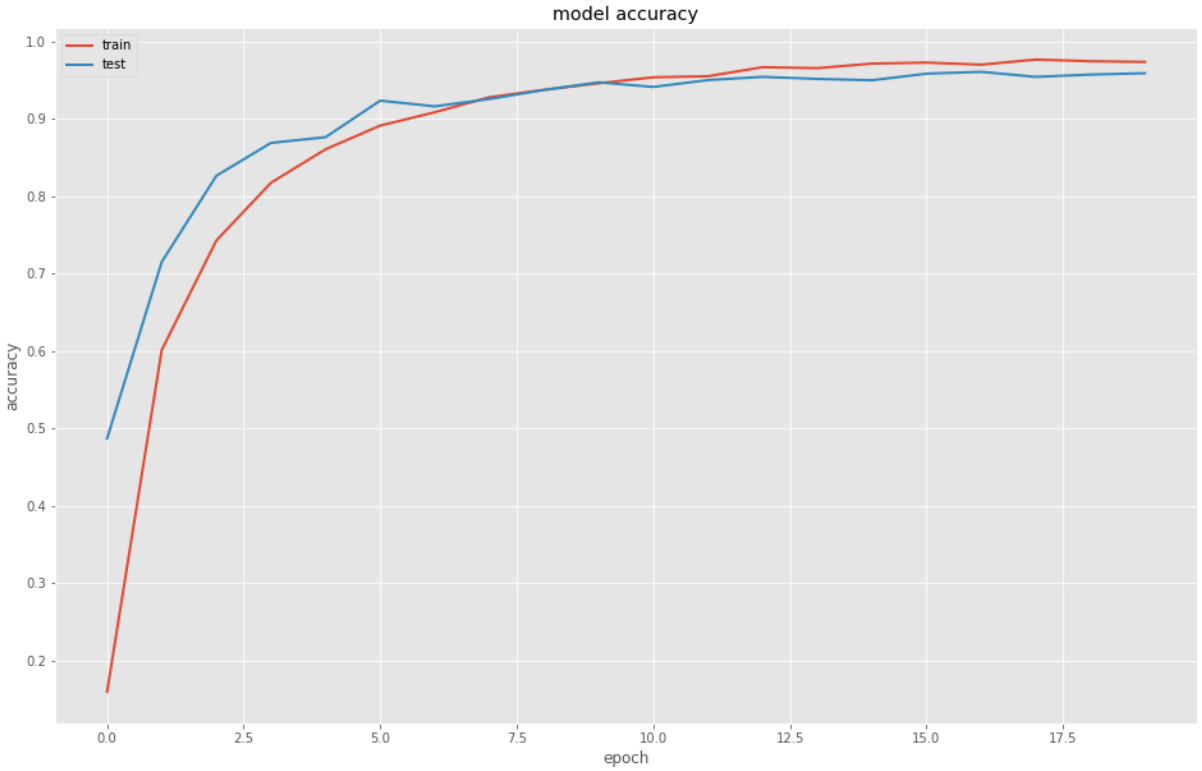
    b. Epoch 30



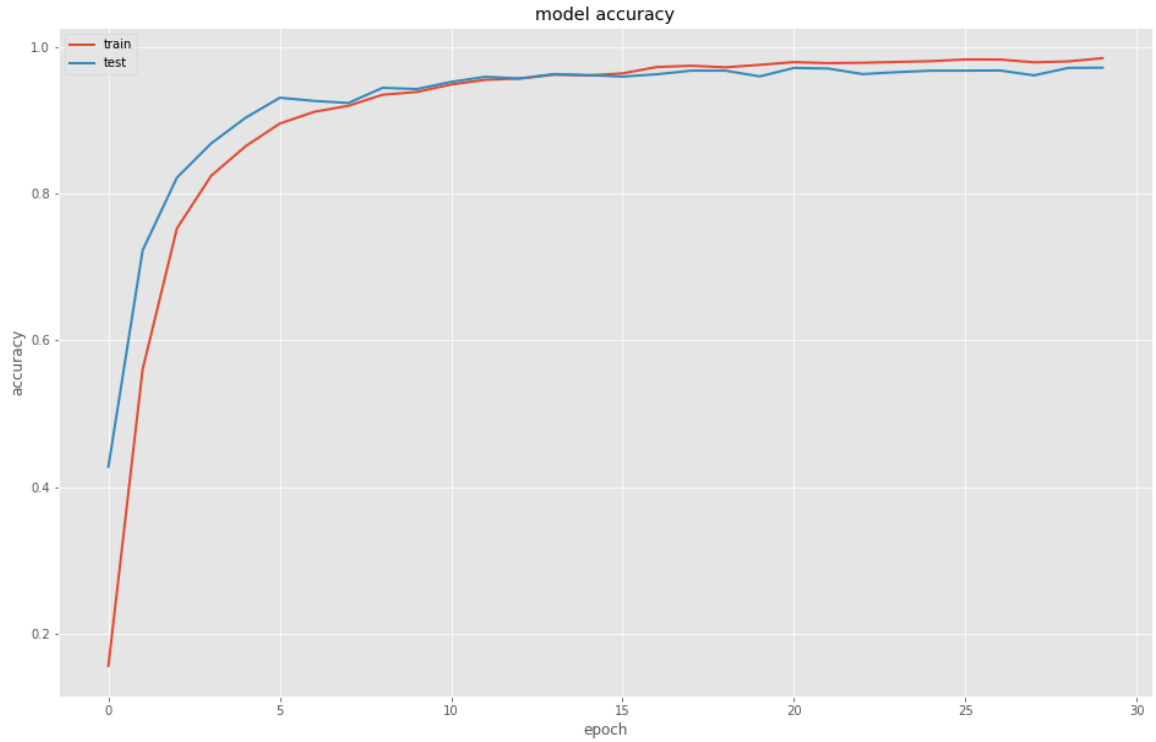
c. Epoch 40



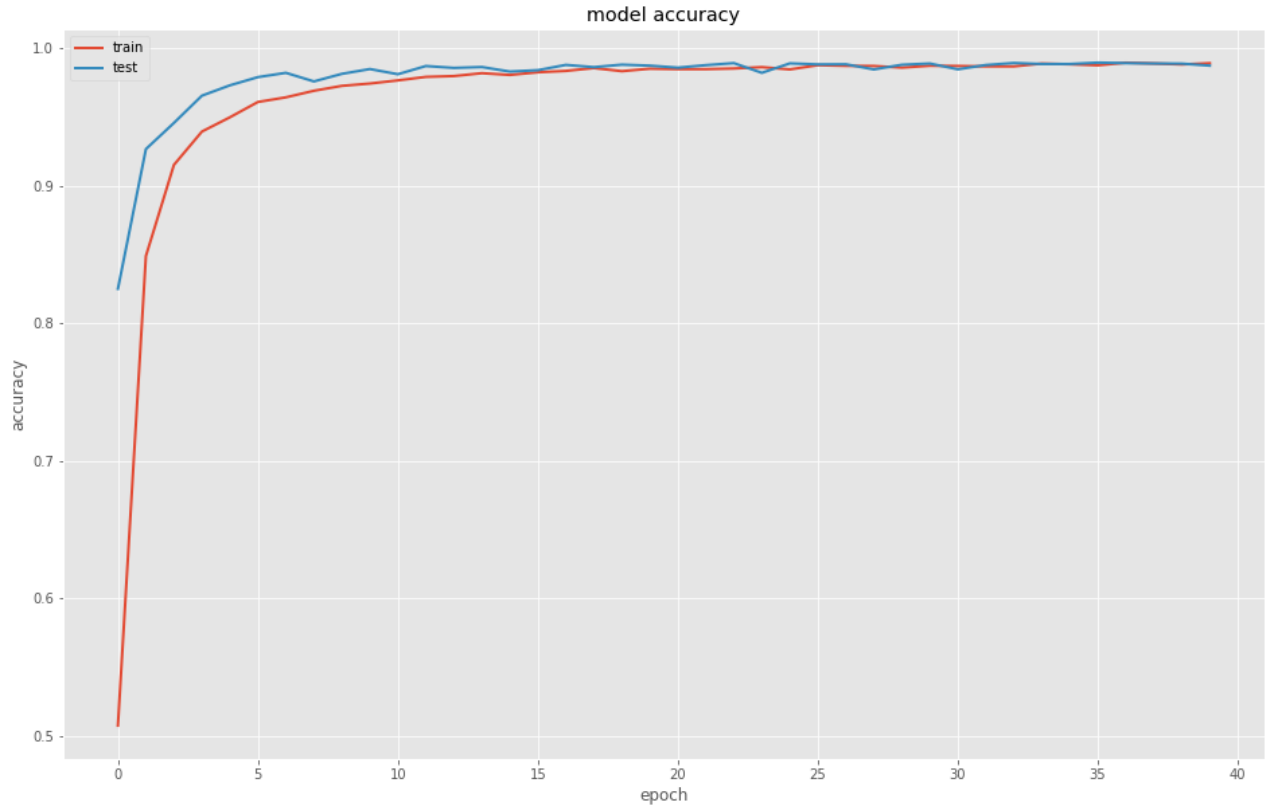
2. Percobaan dengan dataset Jerman-China sebagai data training
   a. Epoch 20



b. Epoch 30

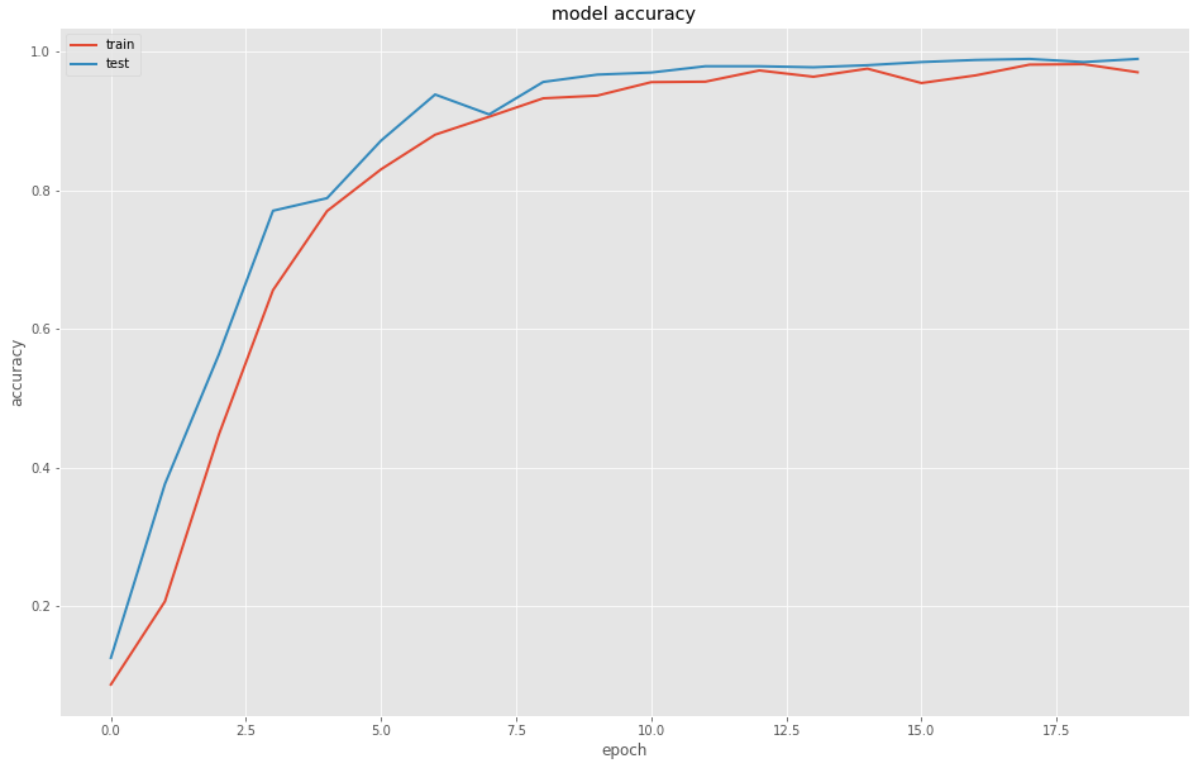


c. Epoch 40

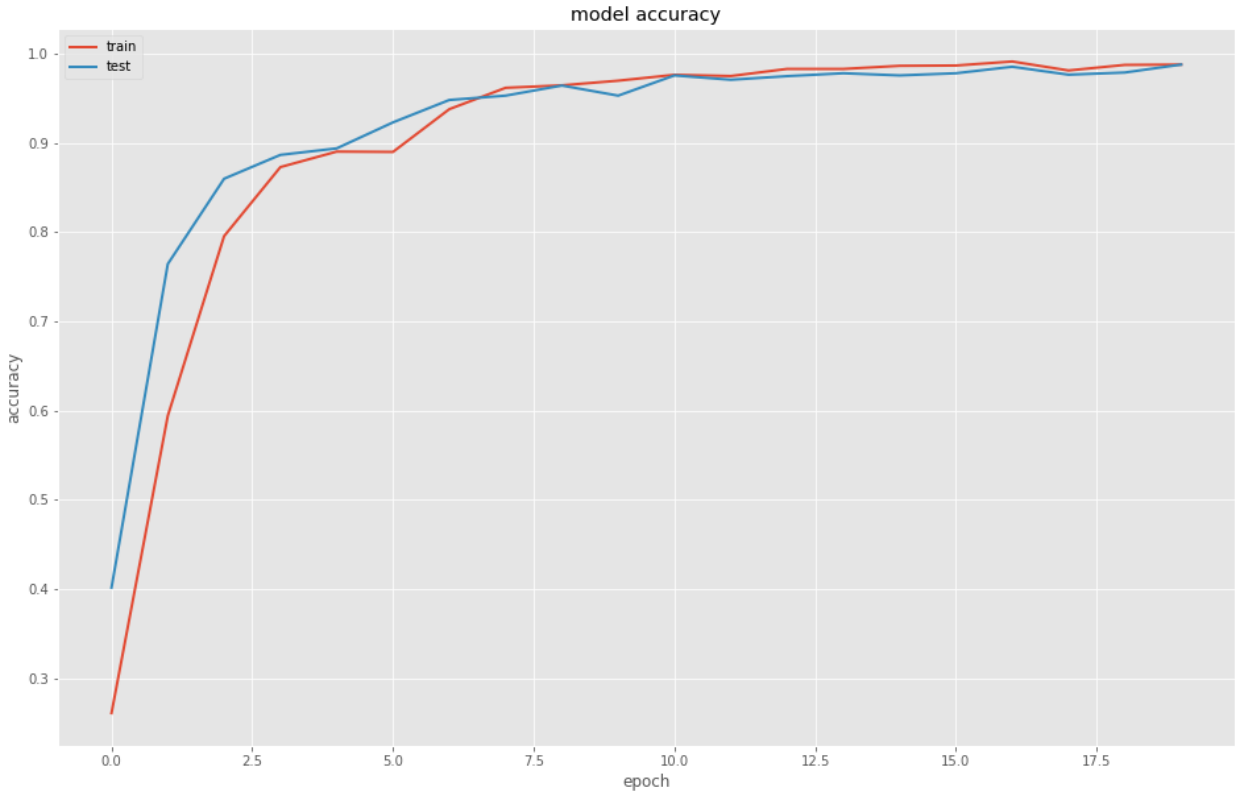


3. Percobaan perbedaan jumlah data training

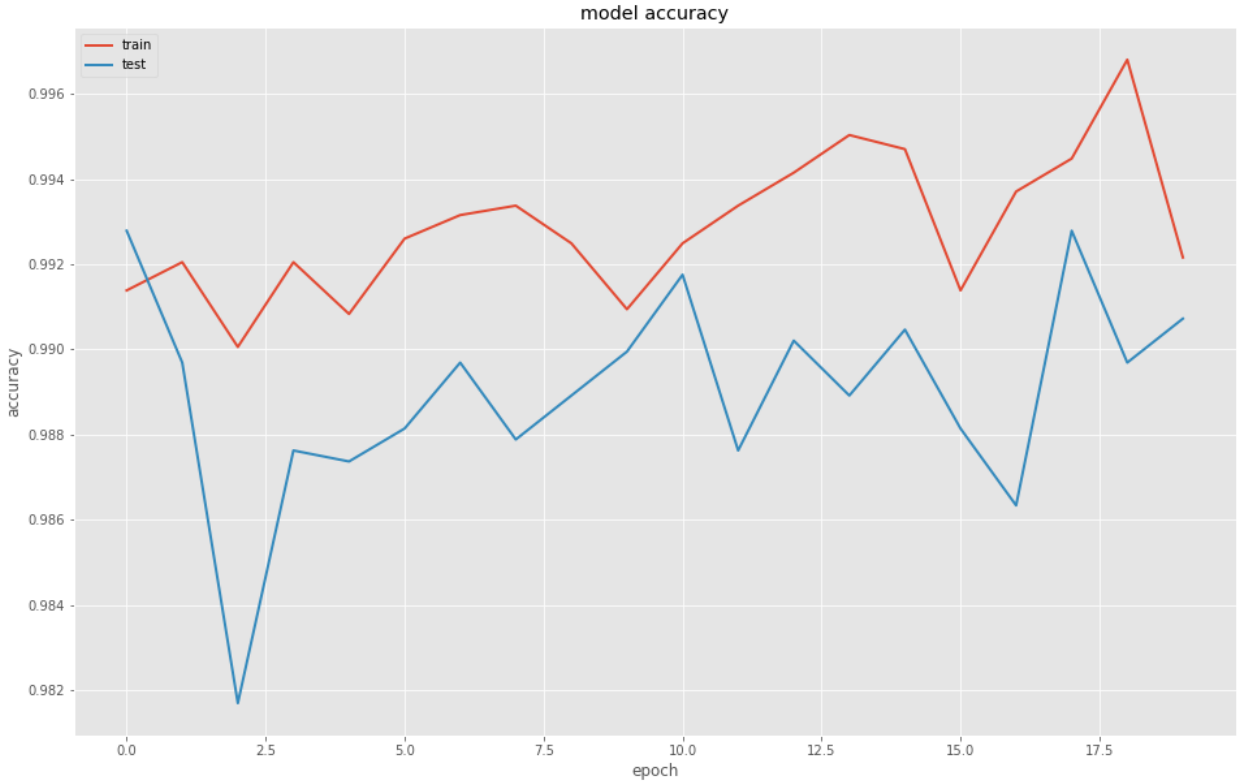
a. Grup 1 (15 kategori dengan jumlah data 210 per kategori)



b. Grup 2 (14 kategori dengan jumlah data 420 per kategori)



c. Grup 3 (14 kategori dengan jumlah data 1320 per kategori)



Tabel Akurasi Hasil Percobaan antara data Jerman saja dengan data Jerman-China:

No	Percobaan	Epoch	Akurasi dengan data test dari dataset	Akurasi dengan data test dari internet
1	Dataset Jerman	20	96.7991828918457	42.38095283508301
2	Dataset Jerman	30	96.88073396682739	41.428571939468384
3	Dataset Jerman	40	97.28848338127136	40.714284777641296
4	Dataset Jerman dan China	20	95.72314023971558	44.52380836009979
5	Dataset Jerman dan China	30	97.67583012580872	45.47618925571442
6	Dataset Jerman dan China	40	98.82973432540894	48.8095223903656

Tabel Akurasi Hasil Percobaan Perbedaan Jumlah Data Training:

No	Percobaan	Akurasi dengan data test dari dataset	Akurasi dengan data test dari internet
1	Grup 1 (15 kategori dengan jumlah data 210 per kategori)	98.51852059364319	38.66666555404663
2	Grup 2 (14 kategori dengan jumlah data 420 per kategori)	98.8095223903656	49.23076927661896
3	Grup 3 (14 kategori dengan jumlah data 1320 per kategori)	99.22438859939575	63.57142925262451

Analisa:

1) Untuk evaluasi model dengan data test dari dataset, penggunaan dataset Jerman saja menghasilkan akurasi yang lebih tinggi dibandingkan dengan menggunakan dataset Jerman yang digabung dengan dataset China.

   Dari analisa kami hal ini terjadi karena kemungkinan dari adanya perbedaan bentuk dari traffic sign Jerman dan China meskipun masih sama kategorinya, dan karena kami mengambil 200 data / kategori dari setiap dataset, tetapi dari dataset China sendiri tidak semuanya sampai 200 data.
2) Untuk evaluasi model dengan data test dari internet, penggunaan dataset Jerman saja menghasilkan akurasi yang lebih rendah dibandingkan dengan menggunakan dataset Jerman yang digabung dengan dataset China.

   Dari analisa kami hal ini terjadi karena pada percobaan dengan dataset Jerman dan China, terdapat lebih banyak variasi pada data trainingnya daripada hanya dengan dataset Jerman saja. Hal ini menyebabkan percobaan dengan dataset Jerman-China lebih tinggi akurasinya karena data test dari internet juga memiliki lebih banyak variasi.
3) Untuk evaluasi model dengan data test dari dataset, penggunaan Epoch yang lebih banyak meningkatkan akurasi.

   Dari analisa kami hal ini terjadi karena semakin bertambah epoch maka semakin detail mesin mempelajari data training. Karena evaluasi model dilakukan dengan data test dari dataset, maka variasi antara dataset training dengan dataset test tidak begitu berbeda. Oleh sebab itu, evaluasi model pada kedua percobaan yang kami lakukan sama-sama meningkat seiring bertambahnya epoch.
4) Untuk evaluasi model dengan data test dari internet, pada penggunaan dataset Jerman saja, semakin banyak epoch yang digunakan semakin turun akurasi yang didapatkan.

Sedangkan pada penggunaan dataset Jerman dan China, semakin banyak epoch yang digunakan semakin tinggi akurasi yang didapatkan.

Dari analisa kami hal ini terjadi karena mesin yang di training dengan menggunakan dataset Jerman saja hanya dapat mempelajari gambar yang kurang bervariasi sehingga ketika epoch semakin bertambah, detail-detail pada gambar yang sebenarnya tidak diambil dikenali menjadi bagian dari model sehingga akurasi pun menurun ketika model dievaluasi dengan menggunakan data test yang lebih bervariasi. Sebab semakin banyak epoch maka dataset training yang kurang bervariasi tersebut akan dipelajari semakin detail. Sedangkan, mesin yang di training dengan menggunakan dataset Jerman-China mempelajari data yang lebih bervariasi. Oleh sebab itu, semakin bertambah epoch, semakin banyak variasi yang ia pelajari sehingga ketika model dievaluasi dengan data test yang bervariasi, akurasinya meningkat.

5) Pada percobaan perbedaan jumlah data training, dapat dilihat terdapat perbedaan akurasi antara Grup 1, Grup 2, dan Grup 3. Model yang di training dengan data dari Grup 3 memiliki tingkat akurasi yang lebih tinggi dibanding dua grup lainnya. Mesin akan menjadi lebih pintar jika data yang ia pelajari lebih banyak. Dalam hal ini data training merupakan data yang dipelajari mesin. Oleh sebab itu, semakin banyak jumlah data training maka semakin tinggi akurasi modelnya.

Kesimpulan :

Secara general, semakin banyak epoch yang dilakukan maka kompleksitas model yang di generate akan semakin meningkat (terbukti pada percobaan yang data testnya berasal dari dataset yang sama). Semakin tinggi kompleksitas model, maka kemampuannya untuk mengenali suatu gambar juga akan semakin meningkat. Performance dari model ini akan terus meningkat hingga mencapai 1 titik, lalu akan mulai menurun sehingga kondisi ini dinamakan overfitting. Pada percobaan antara data Jerman dan data Jerman-China, overfitting terjadi saat model yang di generate dari dataset Jerman dites ke gambar traffic sign yang kami temukan dari internet. Selain itu, dari percobaan perbedaan jumlah data training, dapat disimpulkan bahwa semakin banyak jumlah data training maka semakin tinggi akurasi modelnya.