



# Bitirme Projesi: İşaret Dilini Metne Çevirme

# İÇİNDEKİLER

- 1- Projenin Amacı
- 2- projenin Kullanılabileceği Alanlar
- 3- Projenin Veri Seti Hakkında Genel Bilgi
- 4- Proje Kodlarının Açıklanması
- 5- proje sonucu
- 6-kaynak
- 7- Bitiş



## 1- Projenin Amacı

- Bu projede, işitme engelli bireylerin kullandığı işaret dilini görsel verilerden anlamak ve bu işaret dilini metin formuna dönüştürmek için yapay zeka ve derin öğrenme tekniklerini kullanıldı. CNN, görüntü verilerini işlemek için etkili bir derin öğrenme mimarisi olduğundan, işaret dilini tanımak için kullanıldı. Mimari, işaret dilini tanımak ve anlamak için özelleştirir, görsel verilerden öznitelikler çıkararak bu öznitelikleri metin verisine dönüştürür
- Projede amaçlanan şey, işaret dilini kullanan bireylerin iletişimini geliştirmek ve onlara daha geniş bir iletişim ağı sağlamaktır. Bu teknoloji, işitme engelli bireylerin günlük hayatta daha rahat iletişim kurmalarına yardımcı olur. Görsel verilerden doğrudan metin çıkarılması, işitme engelli bireylerin anlaması ve kendilerini ifade etmeleri için önemli bir adımdır.

## 2- Projenin kullanım alanları

- 1- Eğitim
- 2- Sağlık Hizmetleri
- 3- Toplumsal Katılım
- 4- Dijital İletişim
- 5- Acil Durumlar ve Hizmetler

### 3- Veri Seti Hakkında Genel Bilgi

#### 1- Veri Setinin İçeriği:

Veri seti Türk işaret dili alfabesi baz alınarak hazırlanmıştır.  
29 harften oluşan veri setinde her harf için 150 fotoğraf kullanılmıştır.



Ö harfi



K harfi



Ç harfi



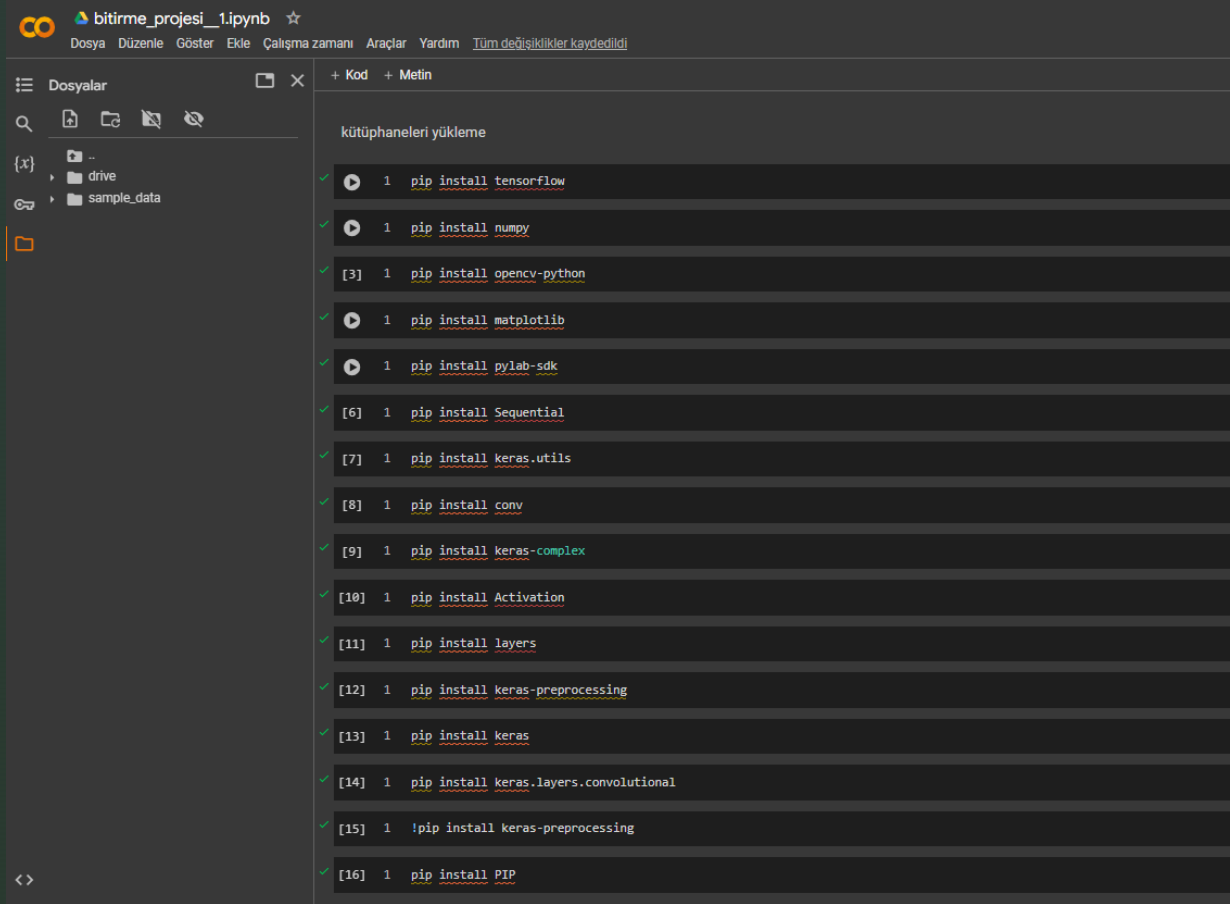
## 2- Veri Setinin Kodları:

Veri seti hazırlanırken kullanılan kod:

```
1 import os
2 import cv2
3
4 DATA_DIR="./data" #dosya yolu
5
6 if not os.path.exists(DATA_DIR):# data dosyası yoksa oluşturulur.
7     os.makedirs(DATA_DIR)
8
9 number_of_class=29 # sınıf sayısı
10 data_size=150 # her sınıftaki görüntü sayısı
11
12 cap=cv2.VideoCapture(0)
13 for j in range(number_of_class):
14     if not os.path.exists(os.path.join(DATA_DIR,str(j))): #DATA_DIR içine sınıf sayısı kadar klasör oluşturur
15         os.makedirs(os.path.join(DATA_DIR,str(j)))
16
17     print("data sınıfları {}".format(j))
18
19     while True:
20         _,frame=cap.read() #cap.read() ile bir video çerçevesi okunur,cv2.putText fonksiyonuyla ekrana "hazır olunca q bas" şeklinde bir metin eklenir.
21         cv2.putText(frame,"hazır olunca q bas",(100,50),cv2.FONT_HERSHEY_SIMPLEX,1.3,(0,0,255),3)
22
23         cv2.imshow("frame",frame)
24         if cv2.waitKey(25) & 0xFF==ord("q"):
25             break
26
27     counter=0
28
29     while counter<data_size:
30         _,frame=cap.read()
31         cv2.imshow("frame",frame)
32         cv2.waitKey(25)
33         cv2.imwrite(os.path.join(DATA_DIR,str(j),"{}.jpg".format(counter)),frame)
34
35         counter+=1
36
37 cap.release()
38 cv2.destroyAllWindows()
39
```

# 4- Proje Kodlarının Açıklanması

-Gerekli kütüphaneleri dahil edildi



The screenshot shows a Jupyter Notebook titled "bitirme\_projesi\_1.ipynb". The left sidebar displays a file explorer with a folder named "sample\_data". The main area shows a list of 16 pip install commands, each preceded by a green checkmark and a play button icon, indicating successful execution. The commands are as follows:

```
1 pip install tensorflow
1 pip install numpy
[3] 1 pip install opencv-python
1 pip install matplotlib
1 pip install pylab-sdk
[6] 1 pip install Sequential
[7] 1 pip install keras.utils
[8] 1 pip install conv
[9] 1 pip install keras-complex
[10] 1 pip install Activation
[11] 1 pip install layers
[12] 1 pip install keras-preprocessing
[13] 1 pip install keras
[14] 1 pip install keras.layers.convolutional
[15] 1 !pip install keras-preprocessing
[16] 1 pip install PIP
```

Tensorflow import edildi  
Verilerin dosya yolu eklendi  
Veriler train ve test olarak ayrıldı

veriyi yükleme

✓  
13  
sn.

```
[17] 1 import tensorflow
      2 from tensorflow.keras.preprocessing.image import ImageDataGenerator
      3
      4 base_dir="/content/drive/MyDrive/bitirme_projesi/data"
      5
      6 train_datagen=ImageDataGenerator(rescale=1./255,validation_split=0.1)
      7 test_datagen=ImageDataGenerator(rescale=1.255,validation_split=0.1)
      8
      9 train_datagen=train_datagen.flow_from_directory(base_dir,target_size=(500,500),subset="training",batch_size=2)
     10 test_datagen=test_datagen.flow_from_directory(base_dir,target_size=(500,500),subset="validation",batch_size=2)
     11
     12
     13
     14
```

Found 3924 images belonging to 29 classes.  
Found 435 images belonging to 29 classes.

4350 veriden: 3924 tanesi train için , 435 tanesi test(validation) için ayrıldı



Verilerimizin doğru bir şekilde oluğunu görmek için matplotlib kütüphanesinden yararlanıp görselleştiriyoruz

görselleştirme


```
1 import matplotlib.pyplot as plt
2
3
4 for _ in range(5):
5     img,label=test_datagen.next()
6     print(img.shape)
7     img=img.astype("uint8")
8     plt.imshow(img[0])
9     print(label[0])
10    plt.show()
11
12
```

(2, 500, 500, 3)

(2, 500, 500, 3)

(2, 500, 500, 3  
10 0 0 0 0

(2, 500, 500, 3)



(2, 500, 500, 3)

## sıralı(sequential) model oluştumu

```

1  import tensorflow as tf
2  import numpy as np
3  import pylab as pl
4  from keras import backend as k
5  import matplotlib.pyplot as plt
6  import keras.utils as np_utils
7  from keras.models import Sequential
8  from keras.layers import Conv2D, MaxPooling2D
9  from keras.layers import Dense, Dropout, Activation, Flatten
10 from tensorflow.keras import layers, activations
11
12 import tensorflow as tf
13 """optimizer=tf.keras.optimizers.Adamax(learning_rate=0.001)"""
14 loss=tf.keras.losses.CategoricalCrossentropy()
15 model=Sequential()
16
17
18 model.add(layers.Conv2D(filters=32,activation="relu", kernel_size=(5,5),input_shape=(500,500,3)))
19 model.add(layers.MaxPooling2D(2,2))
20
21 model.add(layers.Conv2D(filters=64,activation="relu", kernel_size=(3,3)))
22 model.add(layers.MaxPooling2D(2,2))
23
24 model.add(layers.Conv2D(filters=128,activation="relu", kernel_size=(2,2)))
25 model.add(layers.MaxPooling2D(2,2))
26
27
28
29 model.add(layers.Flatten())
30
31 model.add(layers.Dense(64,activation="relu"))
32 model.add(layers.Dense(128,activation="relu"))
33 model.add(layers.Dense(128,activation="relu"))
34 model.add(layers.Dense(64,activation="relu"))
35 model.add(layers.Dense(29,activation="softmax"))
36
37 model.compile(optimizer="adam",loss=loss,metrics=["accuracy"])
38
39 model.summary()

```

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 496, 496, 32)	2432
max_pooling2d_3 (MaxPooling2D)	(None, 248, 248, 32)	0
conv2d_4 (Conv2D)	(None, 246, 246, 64)	18496
max_pooling2d_4 (MaxPooling2D)	(None, 123, 123, 64)	0
conv2d_5 (Conv2D)	(None, 122, 122, 128)	32896
max_pooling2d_5 (MaxPooling2D)	(None, 61, 61, 128)	0
flatten_1 (Flatten)	(None, 476288)	0
dense_5 (Dense)	(None, 64)	30482496
dense_6 (Dense)	(None, 128)	8320
dense_7 (Dense)	(None, 128)	16512
dense_8 (Dense)	(None, 64)	8256
dense_9 (Dense)	(None, 29)	1885

Total params: 30571293 (116.62 MB)  
 Trainable params: 30571293 (116.62 MB)  
 Non-trainable params: 0 (0.00 Byte)

modelimizi test ediyoruz

```
1 result=model.fit(train_datagen,epochs=5,verbose=1, validation_data=test_datagen)
```

```
Epoch 1/5  
1962/1962 [=====] - 788s 400ms/step - loss: 3.3607 - accuracy: 0.0441 - val_loss: 14.4357 - val_accuracy: 0.1149  
Epoch 2/5  
1962/1962 [=====] - 64s 33ms/step - loss: 2.7253 - accuracy: 0.2077 - val_loss: 165.2851 - val_accuracy: 0.6207  
Epoch 3/5  
1962/1962 [=====] - 66s 34ms/step - loss: 0.2167 - accuracy: 0.9353 - val_loss: 96.5039 - val_accuracy: 0.9000  
Epoch 4/5  
1962/1962 [=====] - 63s 32ms/step - loss: 0.0563 - accuracy: 0.9906 - val_loss: 209.5965 - val_accuracy: 0.8897  
Epoch 5/5  
1962/1962 [=====] - 64s 33ms/step - loss: 0.1232 - accuracy: 0.9727 - val_loss: 76.5852 - val_accuracy: 0.9149
```

%91

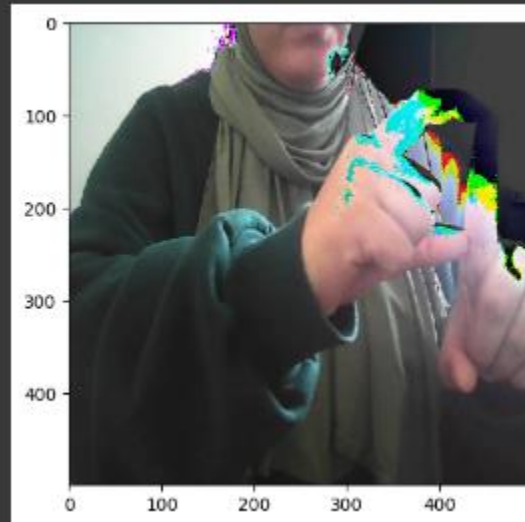
Model.fit() fonksiyonu ile train\_datagen kullanılarak 5 epochsluk bir eğitim yapılmaktadır, validation\_data= test datagen ise eğitilmiş modelin performansını değerlendirir.eğitim sırasında kullanılamaz, öğrenme sürecine dahil edilmez.



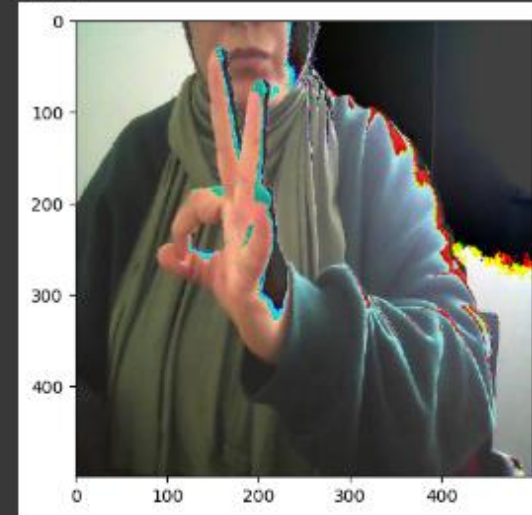
test verisi seti üzerinden test etmek

```
1 print(test_datagen.class_indices)
2 for _ in range(5):
3     img, label = test_datagen.next()
4     model.predict(img)
5     img = img.astype("uint8")
6     np.argmax(a[0])
7     plt.imshow(img[0])
8     if np.argmax(a[0]) == 0:
9         print("-- A --")
10    if np.argmax(a[0]) == 1:
11        print("-- B --")
12    if np.argmax(a[0]) == 2:
13        print("-- C --")
14    if np.argmax(a[0]) == 3:
15        print("-- C --")
16    if np.argmax(a[0]) == 4:
17        print("-- D --")
18    if np.argmax(a[0]) == 5:
19        print("-- E --")
20    if np.argmax(a[0]) == 6:
21        print("-- F --")
22    if np.argmax(a[0]) == 7:
23        print("-- G --")
24    if np.argmax(a[0]) == 8:
25        print("-- G --")
26    if np.argmax(a[0]) == 9:
27        print("-- H --")
28    if np.argmax(a[0]) == 10:
29        print("-- I --")
30    if np.argmax(a[0]) == 11:
31        print("-- I --")
32    if np.argmax(a[0]) == 12:
33        print("-- J --")
34    if np.argmax(a[0]) == 13:
35        print("-- K --")
36    if np.argmax(a[0]) == 14:
37        print("-- L --")
38    if np.argmax(a[0]) == 15:
39        print("-- M --")
40    if np.argmax(a[0]) == 16:
41        print("-- N --")
42    if np.argmax(a[0]) == 17:
43        print("-- O --")
44    if np.argmax(a[0]) == 18:
45        print("-- O --")
46    if np.argmax(a[0]) == 19:
47        print("-- P --")
48    if np.argmax(a[0]) == 20:
49        print("-- R --")
50    if np.argmax(a[0]) == 21:
51        print("-- S --")
52    if np.argmax(a[0]) == 22:
53        print("-- S --")
54    if np.argmax(a[0]) == 23:
55        print("-- T --")
56    if np.argmax(a[0]) == 24:
57        print("-- U --")
58    if np.argmax(a[0]) == 25:
59        print("-- U --")
60    if np.argmax(a[0]) == 26:
61        print("-- V --")
62    if np.argmax(a[0]) == 27:
63        print("-- V --")
64    if np.argmax(a[0]) == 28:
65        print("-- Z --")
66
67 plt.show()
68
69
```

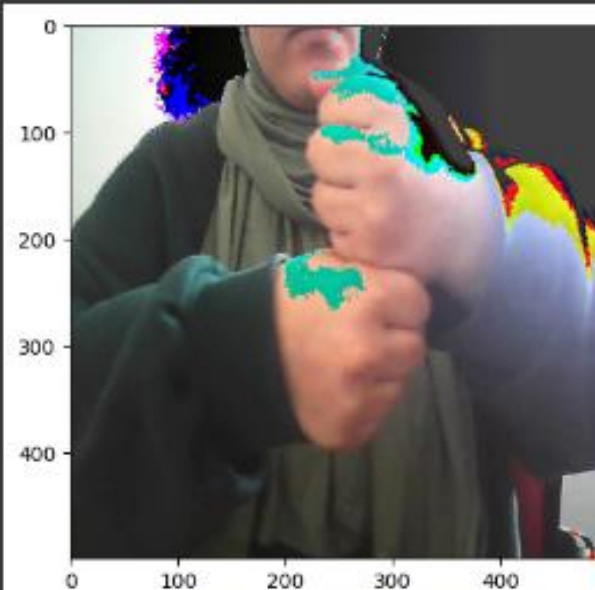
1/1 [=====] - 0s 30ms/step  
-- D --



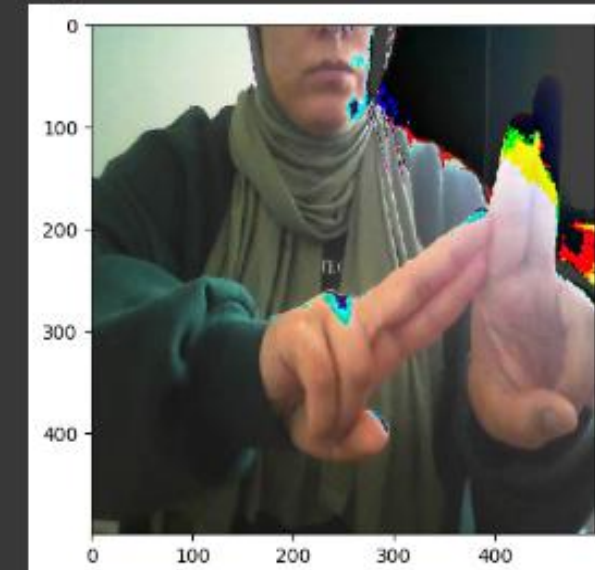
{'A': 0, 'B': 1, 'C': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7  
1/1 [=====] - 0s 84ms/step  
-- 0 --



1/1 [=====] - 0s 4ms/step  
-- G --



1/1 [=====] - 0s 21ms/step  
-- K --



Tahmin ve gerçek değerleri resim olarak değil, sayısal olarak göstermesini istiyorum

tahmin ve gerçek değerlerini karşılaştıralım



```
1 test_a=model.predict(test_datagen)
2 t=[]
3 print(test_datagen.class_indices)
4 for i in test_a:
5     t.append(np.argmax(i))
6
7 x=zip(t,test_datagen.labels)
8 for i,j in x:
9     print(f" tahmin:{i}   gerçek:{j}")
```



```
218/218 [=====
{'A': 0, 'B': 1, 'C': 2, 'Ç': 3, 'D': 4}
tahmin:2 gerçek:0
tahmin:15 gerçek:0
tahmin:25 gerçek:0
tahmin:20 gerçek:0
tahmin:14 gerçek:0
tahmin:8 gerçek:0
tahmin:1 gerçek:0
tahmin:10 gerçek:0
tahmin:17 gerçek:0
tahmin:20 gerçek:0
tahmin:7 gerçek:0
tahmin:0 gerçek:0
tahmin:23 gerçek:0
tahmin:2 gerçek:0
tahmin:20 gerçek:0
tahmin:20 gerçek:1
tahmin:12 gerçek:1
tahmin:1 gerçek:1
tahmin:3 gerçek:1
tahmin:12 gerçek:1
tahmin:4 gerçek:1
tahmin:6 gerçek:1
tahmin:1 gerçek:1
tahmin:9 gerçek:1
tahmin:27 gerçek:1
tahmin:19 gerçek:1
tahmin:4 gerçek:1
tahmin:16 gerçek:1
tahmin:28 gerçek:1
tahmin:4 gerçek:1
tahmin:7 gerçek:2
tahmin:13 gerçek:2
tahmin:17 gerçek:2
tahmin:6 gerçek:2
tahmin:7 gerçek:2
tahmin:4 gerçek:2
tahmin:4 gerçek:2
tahmin:3 gerçek:2
tahmin:13 gerçek:2
tahmin:1 gerçek:2
tahmin:7 gerçek:2
tahmin:10 gerçek:2
tahmin:19 gerçek:2
tahmin:28 gerçek:2
tahmin:7 gerçek:2
```

```

tahmin:28 gerçek:3
tahmin:8 gerçek:3
tahmin:0 gerçek:3
tahmin:3 gerçek:3
tahmin:28 gerçek:3
tahmin:25 gerçek:3
tahmin:14 gerçek:3
tahmin:19 gerçek:3
tahmin:16 gerçek:3
tahmin:11 gerçek:3
tahmin:10 gerçek:3
tahmin:23 gerçek:3
tahmin:16 gerçek:3
tahmin:19 gerçek:3
tahmin:13 gerçek:3
tahmin:21 gerçek:4
tahmin:5 gerçek:4
tahmin:9 gerçek:4
tahmin:23 gerçek:4
tahmin:11 gerçek:4
tahmin:18 gerçek:4
tahmin:15 gerçek:4
tahmin:4 gerçek:4
tahmin:16 gerçek:4
tahmin:21 gerçek:4
tahmin:25 gerçek:4
tahmin:12 gerçek:4
tahmin:26 gerçek:4
tahmin:14 gerçek:4
tahmin:17 gerçek:4
```



bitirme\_projesi\_1.ipynb ☆

Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi

Dosyalar



{x}



drive



MyDrive



resul python dersi

Colab Notebooks

aygöl python dersi

ayşe python dersi

batuhan python d...

bitirme\_projesi

data

veriler

data.pickle

esra python dersi

hsd logo

irem c dersi

lütü c dersi

nurcan c dersi

python

python kodları

selçuk python dersi

şule python dersi

sample data

sing\_language\_model

+ Kod + Metin

MODEL KAYDETME

```
[31] 1 model.save("sing_language_model/") #modeli kaydettik
```

kaydettiğimiz modeli tekrar baştan eğitmeden test edebiliyoruz

```
1 model.evaluate(train_datagen)
```

```
1962/1962 [=====] - 38s 20ms/step - loss: 0.0094 - accuracy: 0.9977  
[0.009408536367118359, 0.997706413269043]
```

# kaynakça

1- <https://www.youtube.com/watch?v=P9r6XwiD4w8>

2- <https://www.youtube.com/@yasarniyazogluai>

3-

<https://www.udemy.com/share/1024gy3@gbSEjllV1M3imPyPGBSQQjBsNNoxNJQCX51M2U-Me8G1SxB3HS3HW94r6SAH6kCgYw==/>

4-<https://www.kaggle.com/code/sayakdasgupta/sign-language-classification-cnn-99-40-accuracy>

Dinlediğiniz için  
teşekkürler

