

WTech Yapay Zeka Eđitimi Bitirme Projesi


28.04.2024

Melisa YASAK

Github reposu: https://github.com/MelisaYasak/WTech_YZE/tree/main/bitirmeProjesi_CNN

VERİ SETİ

Veri seti Kaggle platformundan indirildi.

 SAMUEL CORTINHAS · UPDATED A YEAR AGO

▲ 109 New Notebook Download (68 MB) ● ⋮

Cats and Dogs image classification

Binary classification between cats and dogs

[Data Card](#) [Code \(61\)](#) [Discussion \(1\)](#) [Suggestions \(0\)](#)

About Dataset

Over a 1000 images of cats and dogs scraped off of google images. The problem statement is to build a model that can classify between a cat and a dog in an image as accurately as possible.

Image sizes range from roughly 100×100 pixels to 2000×1000 pixels.

Image format is jpeg.

Duplicates have been removed.

Usability ⓘ

8.75

License


[CC0: Public Domain](#)

Expected update frequency

Never

Tags

Earth and NatureImageBeginnerClassificationCNN



Veri Seti içerisinde 350 kedi, 350 köpek olmak üzere 700 adet görsel bulunmaktadır. Her görselin boyutu birbirinden farklıdır. Görseller RGB formatındadır. Veri seti aşağıdaki görseldeki gibi düzenlenmiş ve oluşturulan klasöre yüklendi. Modelin ne ürettiğini denemek üzere test klasörü içerisine görsel eklendi.

```
▼ data
  > cats
  > dogs

  > test
```

MODEL OLUŞTURMA AŞAMASI

Model Kurulumu

Gerekli kütüphaneler import edildikten sonra model kurulumu yapıldı. Doğru sınıflandırma yapan model özeti sağdaki görselde görülmektedir.

Modeli kurarken

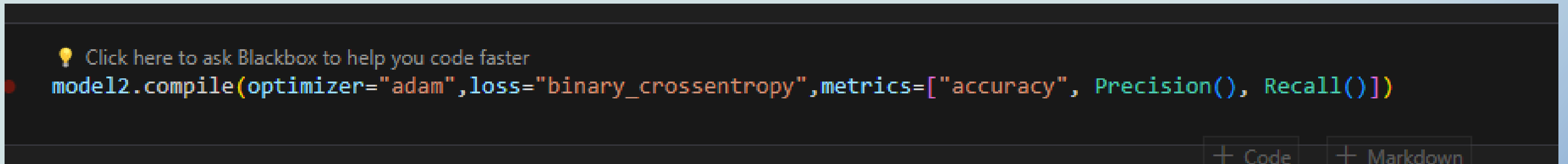
- Konvülasyon katmanlarında filitrelerin boyut farklılıkları ,
- Havuzlama katmanlarının boyutları deneme çalışmaları sonucunda belirlendi.
- Ezberlemenin önüne geçilmek için dropout katmanı sayısı için denemeler yapıldı.
- Kullanılan makinenin memory'i hatası oluşmaması için giriş boyutu (240, 240,3) olarak seçildi.
- Ara katmanların aktivasyon fonksiyonu 'ReLu', son katmanın ise 'Softmax' olarak belirlendi.
- Son katmanın sınıf sayısı 2 olduğundan nöron sayısı 2 olarak ayarlandı.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 238, 238, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 119, 119, 64)	0
conv2d_1 (Conv2D)	(None, 117, 117, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 58, 58, 128)	0
conv2d_2 (Conv2D)	(None, 56, 56, 256)	295168
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 256)	0
flatten (Flatten)	(None, 200704)	0
dense (Dense)	(None, 256)	51380480
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 2)	130

=====
Total params: 51792578 (197.57 MB)
Trainable params: 51792578 (197.57 MB)
Non-trainable params: 0 (0.00 Byte)

COMPILE İŞLEMİ



A screenshot of a code editor interface. At the top, there is a light blue banner with a lightbulb icon and the text "Click here to ask Blackbox to help you code faster". Below this, a line of Python code is shown: `model2.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy", Precision(), Recall()])`. The code is color-coded: "model2.compile" is blue, "optimizer" is orange, "adam" is orange, "loss" is orange, "binary_crossentropy" is orange, "metrics" is blue, and the list of metrics is in green. At the bottom right of the code editor, there are two buttons: "+ Code" and "+ Markdown".

- Optimizer çoğunlukla iyi sonuç ürettiğinden ‘Adam’ olarak seçildi.
- Loss Fonksiyonu ikili sınıflandırma yapıldığından ‘Binary CrossEntropy’ olarak seçildi.
- Model değerlendirme metriği olarak ‘Doğruluk (Acc), Presicion (Hassasiyet) ve Recall (Duyarlılık)’ metrikleri seçildi. Modelin F1 skorunu öğrenebilmek amacıyla presicion ve recall metrikleri de seçildi.

GÖRSELLERİN DÜZENLENMESİ

💡 Click here to ask Blackbox to help you code faster

```
data_dir = r'C:\Users\melis\WTech_YZE\bitirmeProjesi_CNN\image_classification_with_cnn\data'
```

```
datagen = ImageDataGenerator(  
    rescale=1./255,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True,  
    validation_split=0.2  
)
```

```
validation_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = datagen.flow_from_directory(  
    data_dir,  
    target_size=(240, 240),  
    batch_size=32,  
    class_mode='binary'  
)
```

```
validation_generator = validation_datagen.flow_from_directory(  
    data_dir,  
    target_size=(240, 240),  
    batch_size=32,  
    class_mode='binary'  
)
```

✓ 0.0s

Veri arttırımı yapmak, modeli genelleştirmek için,

- rescale: Görüntü pikselleri 0 ile 1 arasında yeniden ölçeklendirildi.
- shear_range: Görüntüleri yatay ve dikey eksenler boyunca eğilmesi veya kayması için belirlendi.
- zoom_range: Görüntüleri yakınlaştırmak için kullanıldı.
- horizontal_flip: Görüntülerin yatay eksen etrafında rastgele çevrilme işlemi için True olarak ayarlandı.

Son olarak verinin %20'si doğrulama geri kalan %80'i eğitim için kullanmak üzere validation_split değeri 0.2 olarak ayarlandı.

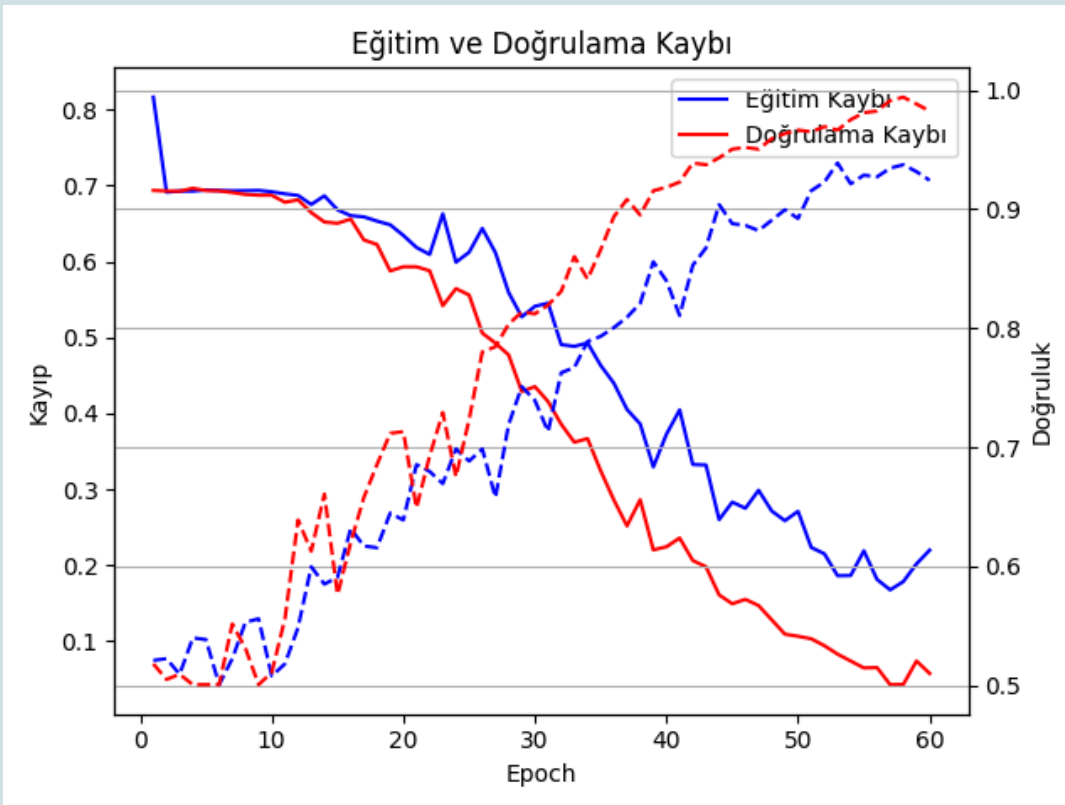
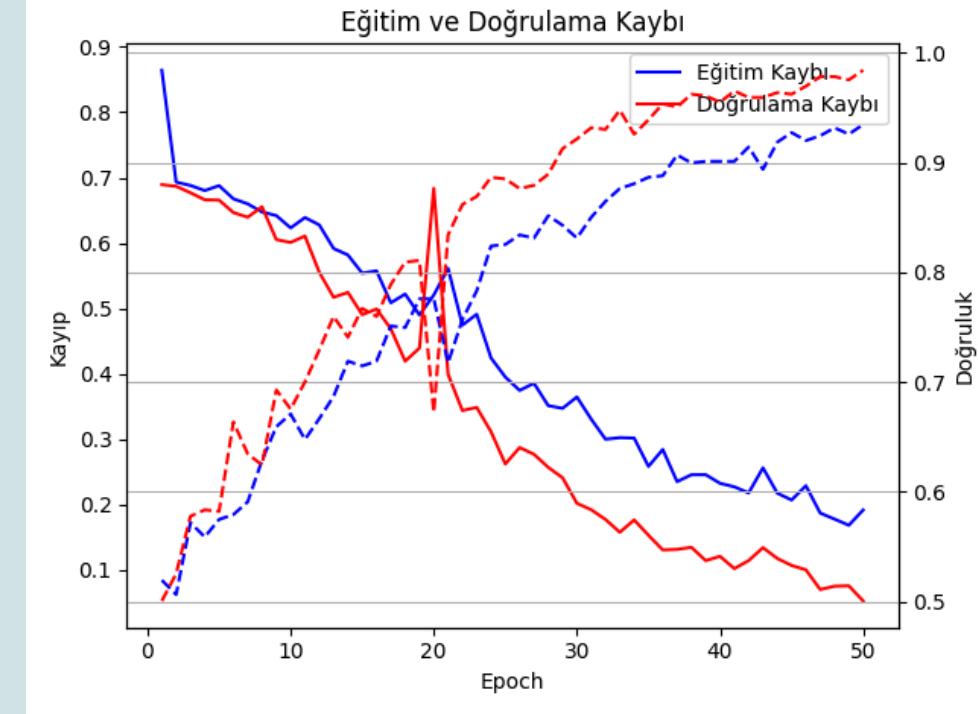
Eğitim ve doğrulama için görselin boyutu modelin girişindeki gibi (240,240) olarak ayarlandı ve

Model Eđitimi

```
Click here to ask Blackbox to help you code faster
history2 = model2.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=60,
    validation_data=validation_generator,
    validation_steps=len(validation_generator)
)
```

2 Epoch'lu deneme eđitimi
yapıldı.

50 epoch'lu eđitim yapıldı. F1
score 0.9345, acc 0.9345



60 Epochlu model
eđitildi

loss: 0.2200

accuracy: 0.9243

F1 score: 0.9243

Bu model test verisinde
en iyi çalışan ve FAST
API'de kullanılan
modeldir.

150 epoch'lu eđitim yapıldı.

Ezberleme görüldü. Bu
eđitimin grafiđine göre epoch
deđrinin 60'larda olması
gerektiđine karar verildi. Ek
olarak bu ařamadan sonra bir
dropout daha eklendi modele.

SON İŞLEMLER

💡 Click here to ask Blackbox to help you code faster

```
precision_values = history3.history['precision_2']  
recall_values = history3.history['recall_2']
```

💡 Click here to ask Blackbox to help you code faster

```
last_f1_score = 2 * (precision_values[-1] * recall_values[-1]) / (precision_values[-1] + recall_values[-1])  
last_f1_score
```

💡 Click here to ask Blackbox to help you code faster

```
len(precision_values)
```

💡 Click here to ask Blackbox to help you code faster

```
max_f1_score = 2 * (max(precision_values) * max(recall_values)) / (max(precision_values) + max(recall_values))  
print(f"max F1 score: {max_f1_score}\nrecall epoch değeri: {recall_values.index(max(recall_values))}\npresicion epoch değeri:{precision_values.index(max(precision_values))}")
```

💡 Click here to ask Blackbox to help you code faster

```
model.save('models/dog-cat2.h5')
```

💡 Click here to ask Blackbox to help you code faster

```
model = load_model('models/dog-cat2.h5')
```

```
test_image = image.load_img('test\dog_4.jpg', target_size=(240,240))  
test_image = image.img_to_array(test_image)  
test_image = np.expand_dims(test_image, axis=0) / 255.0  
result = model.predict(test_image)  
print(result)  
sorted_result_indexes = np.argsort(result[0])[::-1]  
sorted_result = result[0][sorted_result_indexes]
```

```
print(sorted_result[0])
```

✓ 4.6s

1/1 [=====] - 0s 188ms/step

[[0.27242315 0.72757685]]

0.72757685

FAST API KODLANMASI

```
💡 Click here to ask Blackbox to help you code faster
from fastapi import FastAPI
from fastapi import File, UploadFile
from tensorflow.keras.preprocessing import image
import numpy as np
import io
from tensorflow.keras.models import load_model

app = FastAPI()

@app.get("/home")
def home():
    return {"message": "Wtech Yapay Zeka Eğitimi Bitirme Projesi!"}

@app.get("/about")
def about():
    return {"message": "Melisa YASAK"}

@app.post("/predict_image/")
async def predict_image(file: UploadFile = File(...)):
    load_ann = load_model(r'C:\Users\melis\WTech_YZE\bitirmeProjesi_CNN\image classification with cnn\models\dog-cat2.h5')
    contents = await file.read()
    img = image.load_img(io.BytesIO(contents), target_size=(240, 240))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    prediction = load_ann.predict(img_array)
    predicted_class_index = np.argmax(prediction[0])

    class_labels = ["Kedi", "Köpek"]

    prediction = class_labels[predicted_class_index]
    return {"prediction": prediction}
```


GET /home Home

GET /about About

POST /predict_image/ Predict Image

Parameters Cancel Reset

No parameters

Request body required multipart/form-data

file * required

string(\$binary) dog_35.jpg

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict_image/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@dog_35.jpg;type=image/jpeg'
```

Request URL

http://127.0.0.1:8000/predict_image/

Server response

Code	Details
200	<div><p>Response body</p><pre>{ "prediction": "köpek" }</pre>Download</div> <div><p>Response headers</p><pre>content-length: 22</pre></div>

TEŞEKKÜRLER

Melisa YASAK