SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERİN ÖĞRENME VE EVRİŞİMLİ SİNİR AĞLARI DERSİ

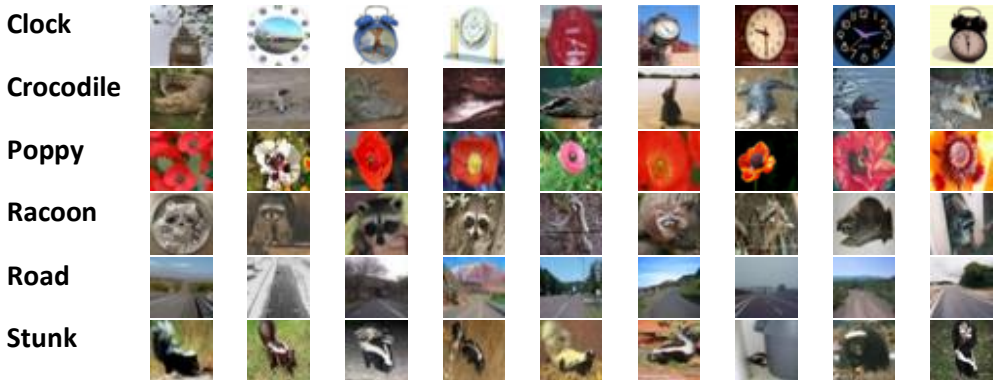
CIFAR100 VERİ TABANINDAN 6 SINIF İÇİN CONVNET AĞI EĞİTME

Öğrenci Numarası: B181210378

Öğrenci Adı: Sevim Suna Kalaycı

Grup: 1A

Ödevde cifar100 veri setinden 6 adet sınıf seçerek eğitim yapmamız gerekiyordu. Benim numaram üzerinden kullanacağım sınıflar 22 clock, 27 crocodile, 62 poppy, 66 racoon, 68 road, 75 stunk idi.




```
import keras
from keras.datasets import cifar100
import os, shutil
import matplotlib.pyplot as plt
import numpy as np
import os
from keras.datasets import cifar100
from keras.models import Sequential
from keras.layers import Dense,Flatten
from keras.layers import Conv2D, MaxPooling2D,Activation
from keras import layers
from keras.utils import to_categorical
from keras import optimizers
import numpy as np
import matplotlib.pyplot as plt
(x_train, y_train), (x_test, y_test) = cifar100.load_data()
```

Gerekli olacağını düşündüğüm bütün kütüphaneleri yükledim. Cifar100 verisetini “from keras.datasets import cifar100” ile cifar100 verisetini çağırdım. Önce dataset isimli klasörü daha sonra test ve train klasörlerini oluşturdum.

```
os.mkdir('dataset')
os.mkdir('dataset\\train')
os.mkdir('dataset\\test')
```

```
path=os.path.join('dataset\\train',str(22))
os.mkdir(path)
path=os.path.join('dataset\\train',str(27))
os.mkdir(path)
path=os.path.join('dataset\\train',str(62))
os.mkdir(path)
path=os.path.join('dataset\\train',str(66))
os.mkdir(path)
path=os.path.join('dataset\\train',str(68))
os.mkdir(path)
path=os.path.join('dataset\\train',str(75))
os.mkdir(path)
path=os.path.join('dataset\\test',str(22))
os.mkdir(path)
path=os.path.join('dataset\\test',str(27))
os.mkdir(path)
path=os.path.join('dataset\\test',str(62))
os.mkdir(path)
path=os.path.join('dataset\\test',str(66))
os.mkdir(path)
path=os.path.join('dataset\\test',str(68))
os.mkdir(path)
path=os.path.join('dataset\\test',str(75))
os.mkdir(path)
```

Benim numaram için seçilen sınıfları tek tek numaralarına göre yükleme gerçekleştirdim. Önce trainleri oluşturdum daha sonrasında testleri oluşturdum.

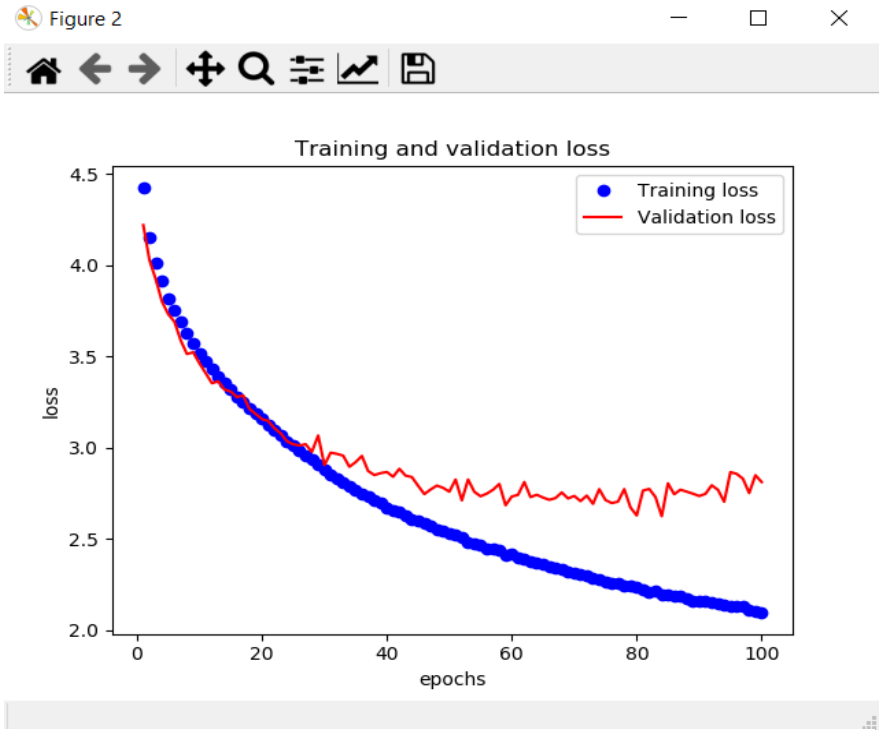
Cifar10 için yaptığımız yüklemelerde 50000 train ve 10000 test görüntüleri vardı ve yüklemek için range 50000 ve 10000 olarak ayarlamıştık. Cifar100 her sınıf için 500 train görüntüsü ve 100 test görüntüsü gerekli idi. Test ve train rangelerini o şekil ayarladım. Ancak ona göre o aralıktaki görüntüler geldi sınıf başı yaklaşık 3 veya 4 görüntü elde ettim. O zaman range klasör içindeki sayı değil görüntüleri verisetindeki sırası için gerekli olduğunu anladı. 50000 ve 10000 range ayarlayınca bütün sınıflarım için 500 train ve 100 test elde etmiş oldum.

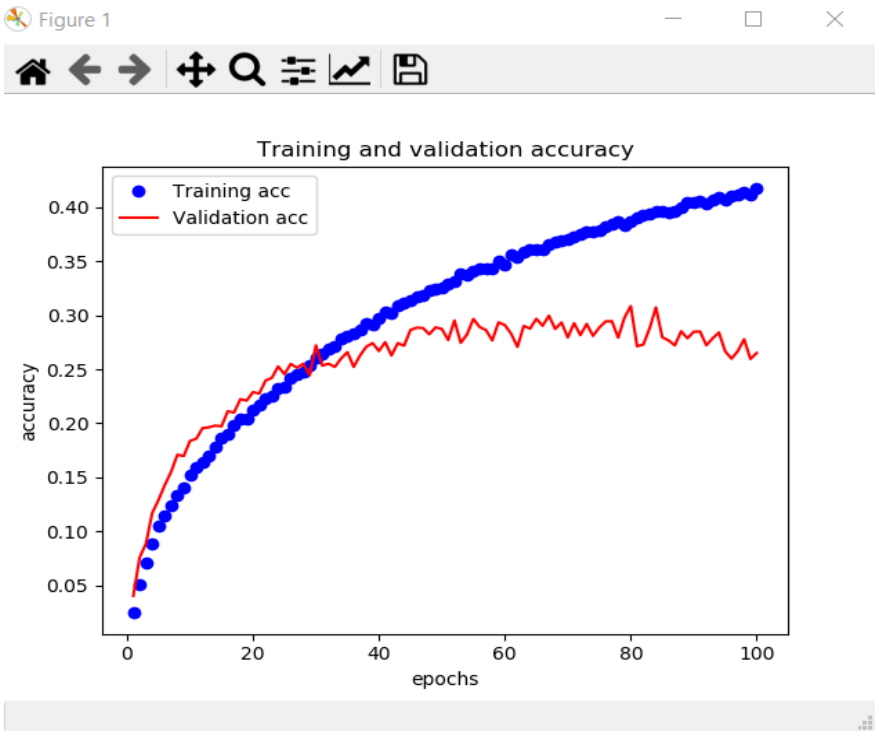
```
for i in range(50000):
    if int(y_train[i]) == 22:
        path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_train[i])
    if int(y_train[i]) == 27:
        path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_train[i])
    if int(y_train[i]) == 62:
        path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_train[i])
    if int(y_train[i]) == 66:
        path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_train[i])
    if int(y_train[i]) == 68:
        path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_train[i])
    if int(y_train[i]) == 75:
        path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_train[i])
for i in range(10000):
    if int(y_test[i]) == 22:
        path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_test[i])
    if int(y_test[i]) == 27:
        path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_test[i])
    if int(y_test[i]) == 62:
        path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_test[i])
    if int(y_test[i]) == 66:
        path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_test[i])
    if int(y_test[i]) == 68:
        path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_test[i])
    if int(y_test[i]) == 75:
        path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
        plt.imsave(path,x_test[i])
```

Tekrar şekillendirme yaptım. Bu konuda hocamızın önceki örneklerinden aldım. Daha sonra katagorize ettim. Başta 6 sınıf olduğu için 6 girmemiz gerektiğini düşündüm. Ancak asıl kendi veriseti sınıfı 100 adet olduğu için 100 girmemiz gerektiğini anladım.

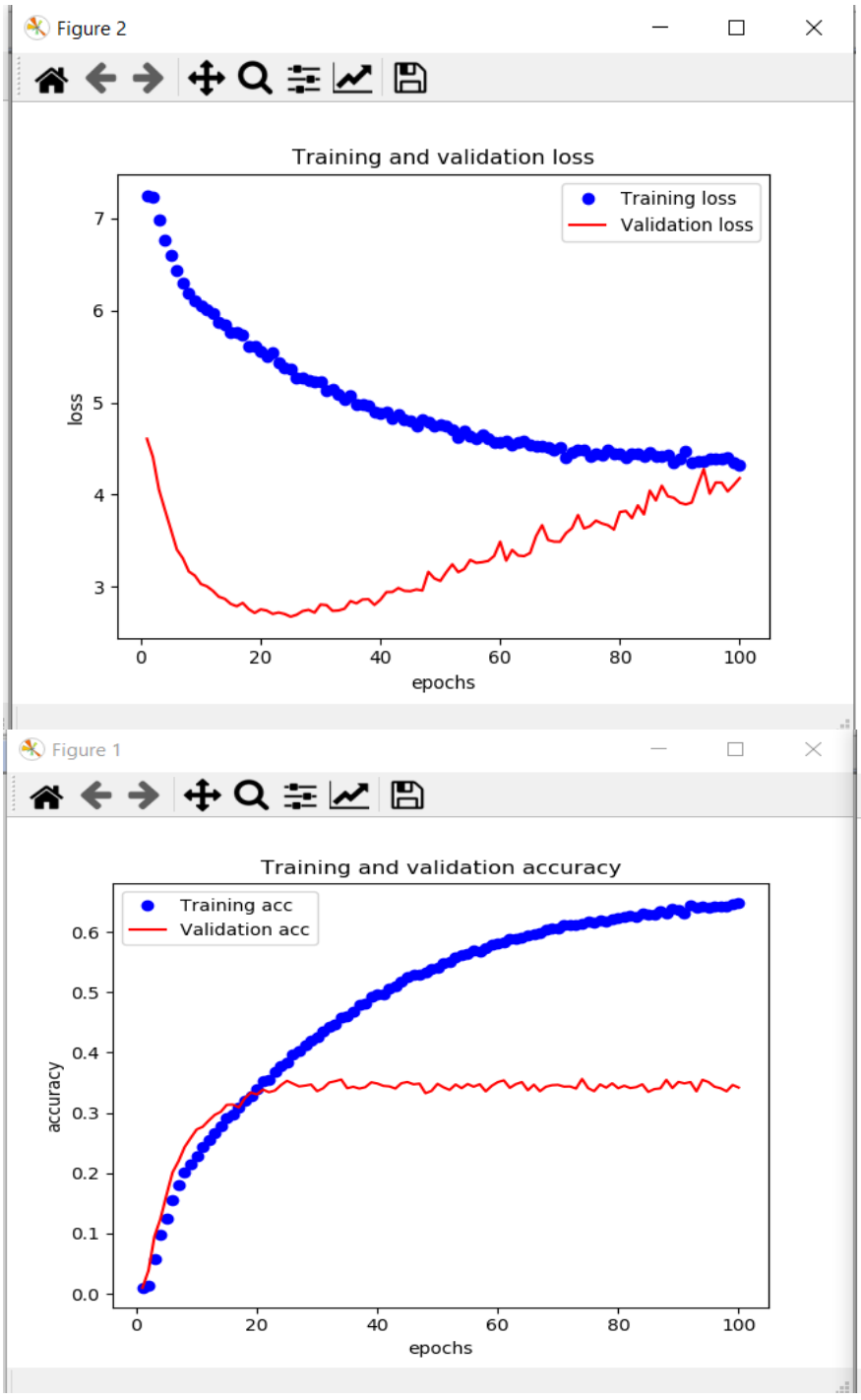
```
x_train=x_train.astype('float32')/255.0
x_test=x_test.astype('float32')/255


y_train=to_categorical(y_train,100)
y_test=to_categorical(y_test,100)
```

Ödev için ilk istenilen 3 veya 6 adet konvolüsyon katmanı, tam bağlantılı katman ise 2-3 olacaktı. 1 2 deneme yaptım. İlk denememde %60 altında oran çıktı ve 3 konvolüsyon katmanı ve 2 dense katmanı kullanmıştım. Birkaç deneme sonucu 6 konvolüsyon ve 3 dense katmanı kullandım. 6 katmandan ilk ve sonuncusu sigmoid kullandım. Araştırmalarda başlangıç ve sonuç katmanı olarak kullanılması gerektiği söyleniyordu. Ara katmanlarda relu kullandım. O da hızlı olmasını amaçladım. Bunun sonucu %99 civarı başarı olarak gösterdi ancak figürlerden anladığım kadarı ile ezberleme olmuş.





Daha sonra katmanların sonlarına dropout kullandım. Dropout kullanarak alttaki grafikleri elde etmiş oldum. Bu tabloda loss üzerinde yaklaşımları yakalasak da istenilen şekilde sonuçlanmadığını görüyorum. Ezberleme üzerinde bir artı sağlanamamış.



Son istenilen veri zenginleştirme ile tekrar ağı tekrar derledim. Bu seçenek üzerine dropoutları kaldırdım. Son dense katmanını 6 olarak ayarladım. Sadece veri zenginleştirme uyguladım.

```
def fonk(img):
    plt.imshow(img)
    plt.show()
    return img


from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
      #preprocessing_function=fonk,
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      vertical_flip=True,
      fill_mode='nearest')


train_dir='C:\\Users\\sevim\\Desktop\\Derin\\dataset\\train'
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(32, 32),
        batch_size=20,
        class_mode='categorical')

test_dir='C:\\Users\\sevim\\Desktop\\Derin\\dataset\\test'
test_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(32, 32),
        batch_size=20,
        class_mode='categorical')
```

Başarım %75 oranında gözükmekte idi. Diğer işlemlerden daha düşük bir başarı oranı oldu ancak tablodan çıkardığım sonuca göre ezberlemeden çok öğrenme işlemi gerçekleşmişti. Loss ve accuracyler istenilen seviyede yakın. Sadece ir aralıkta keskin çıkışalar olmuş ama onları genel sonuçlara baktığımda göz ardı ettim.

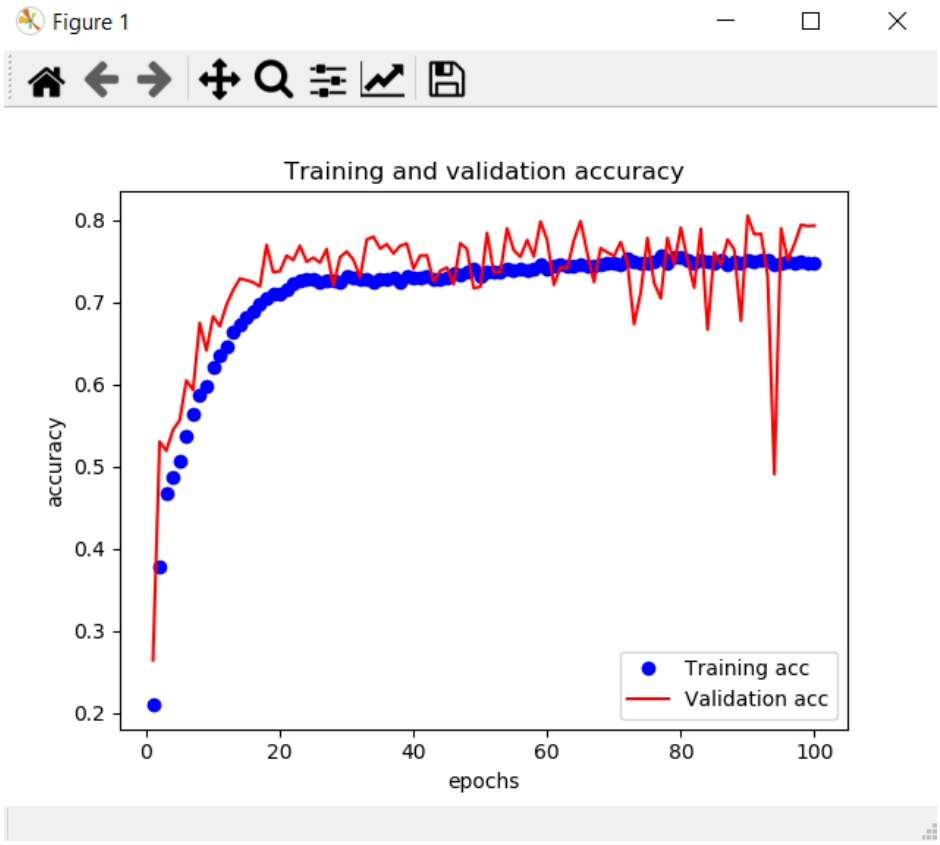


Figure 1
Training and validation accuracy
accuracy
epochs
Training acc
Validation acc
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
20
40
60
80
100

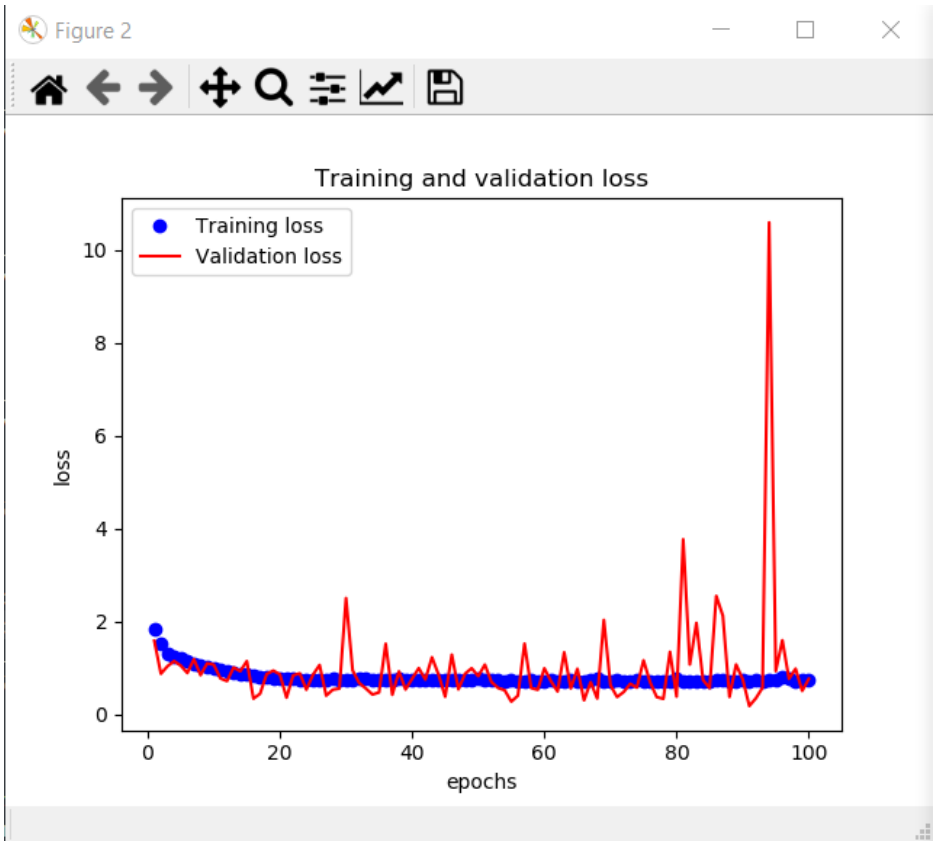


Figure 2
Training and validation loss
loss
epochs
Training loss
Validation loss
0
2
4
6
8
10
0
20
40
60
80
100

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

Sonuçları tabloya yazdırmak için kaydettim.

```
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()

plt.show()
```

Sonuçları yazdırdım. Grafiklere döküp ekrana vermesini sağladım.

```
import keras
from keras.datasets import cifar100
import os, shutil
import matplotlib.pyplot as plt
import numpy as np
from keras.datasets import cifar100
from keras.models import Sequential
from keras.layers import Dense,Flatten
from keras.layers import Conv2D, MaxPooling2D,Activation
from keras import layers
from keras.utils import to_categorical
from keras import optimizers


(x_train, y_train), (x_test, y_test) = cifar100.load_data()

# os.mkdir('dataset')
# os.mkdir('dataset\\train')
```

```
# os.mkdir('dataset\\test')

# path=os.path.join('dataset\\train',str(22))
# os.mkdir(path)
# path=os.path.join('dataset\\train',str(27))
# os.mkdir(path)
# path=os.path.join('dataset\\train',str(62))
# os.mkdir(path)
# path=os.path.join('dataset\\train',str(66))
# os.mkdir(path)
# path=os.path.join('dataset\\train',str(68))
# os.mkdir(path)
# path=os.path.join('dataset\\train',str(75))
# os.mkdir(path)
# path=os.path.join('dataset\\test',str(22))
# os.mkdir(path)
# path=os.path.join('dataset\\test',str(27))
# os.mkdir(path)
# path=os.path.join('dataset\\test',str(62))
# os.mkdir(path)
# path=os.path.join('dataset\\test',str(66))
# os.mkdir(path)
# path=os.path.join('dataset\\test',str(68))
# os.mkdir(path)
# path=os.path.join('dataset\\test',str(75))
# os.mkdir(path)

# for i in range(50000):
#     if int(y_train[i]) == 22:
#         path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
#         plt.imsave(path,x_train[i])
#     if int(y_train[i]) == 27:
#         path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
#         plt.imsave(path,x_train[i])
#     if int(y_train[i]) == 62:
#         path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
#         plt.imsave(path,x_train[i])
#     if int(y_train[i]) == 66:
#         path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
#         plt.imsave(path,x_train[i])
#     if int(y_train[i]) == 68:
#         path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
#         plt.imsave(path,x_train[i])
#     if int(y_train[i]) == 75:
#         path='dataset/train/'+str(int(y_train[i]))+'/'+str(i)+'.png'
#         plt.imsave(path,x_train[i])
# for i in range(10000):
#     if int(y_test[i]) == 22:
```

```
#       path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
#       plt.imsave(path,x_test[i])
#    if int(y_test[i]) == 27:
#       path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
#       plt.imsave(path,x_test[i])
#    if int(y_test[i]) == 62:
#       path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
#       plt.imsave(path,x_test[i])
#    if int(y_test[i]) == 66:
#       path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
#       plt.imsave(path,x_test[i])
#    if int(y_test[i]) == 68:
#       path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
#       plt.imsave(path,x_test[i])
#    if int(y_test[i]) == 75:
#       path='dataset/test/'+str(int(y_test[i]))+'/'+str(i)+'.png'
#       plt.imsave(path,x_test[i])

x_train=x_train.astype('float32')/255.0
x_test=x_test.astype('float32')/255


y_train=to_categorical(y_train,100)
y_test=to_categorical(y_test,100)

model=Sequential()
model.add(layers.Conv2D(32,
                        (3,3),
                        activation='sigmoid',
                        padding='same',
                        input_shape= (32, 32, 3)))
model.add(layers.Conv2D(32,
                        (3,3),
                        padding='same',
                        activation='relu'))


#model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64,
                        (3,3),
                        padding='same',
                        activation='relu'))

model.add(layers.Conv2D(64,
                        (3,3),
                        padding='same',
                        activation='relu'))
```

```
model.add(layers.Conv2D(64,
                 (3,3),
                 padding='same',
                 activation='relu'))
model.add(layers.MaxPool2D())
model.add(layers.Conv2D(64,
                 (3,3),
                 padding='same',
                 activation='sigmoid'))
model.add(layers.MaxPool2D())

#model.add(layers.Dropout(0.5))

model.add(layers.Flatten())
model.add(layers.Dense(512,activation='relu'))
model.add(layers.Dense(512,activation='relu'))
model.add(layers.Dense(6,activation='softmax'))
#model.add(layers.Dropout(0.50))
model.summary()


model.compile(loss='categorical_crossentropy',
          optimizer=optimizers.RMSprop(lr=1e-4),
          metrics=['acc'])

def fonk(img):
   plt.imshow(img)
   plt.show()
   return img


from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    #preprocessing_function=fonk,
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest')


train_dir='C:\\Users\\sevim\\Desktop\\Derin\\dataset\\train'
train_generator = train_datagen.flow_from_directory(
```

```
        train_dir,
        target_size=(32, 32),
        batch_size=20,
        class_mode='categorical')

test_dir='C:\\Users\\sevim\\Desktop\\Derin\\dataset\\test'
test_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(32, 32),
        batch_size=20,
        class_mode='categorical')

# history=model.fit(x_train,
#               y_train,
#               epochs=100,
#               validation_data=(x_test,y_test)
#               )

history = model.fit_generator(
    train_generator,
    steps_per_epoch=1000,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=100)

model.save('cifar100_model1.h5')
model.save_weights('cifar100_weights1.h5')



acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']



np.save('history1.npy',(acc,val_acc,loss,val_loss))
(acc,val_acc,loss,val_loss)=np.load('history1.npy')

epochs = range(1,len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
```

```
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()

plt.show()
```

Yukarıdaki tabloda ödevin tam halini yazdım. Ödevde öncelik olarak hocamızın örneklerinden yardım aldım.