딥러닝/클라우드

2차과제 레포트

32183164 이석현

Dankook University

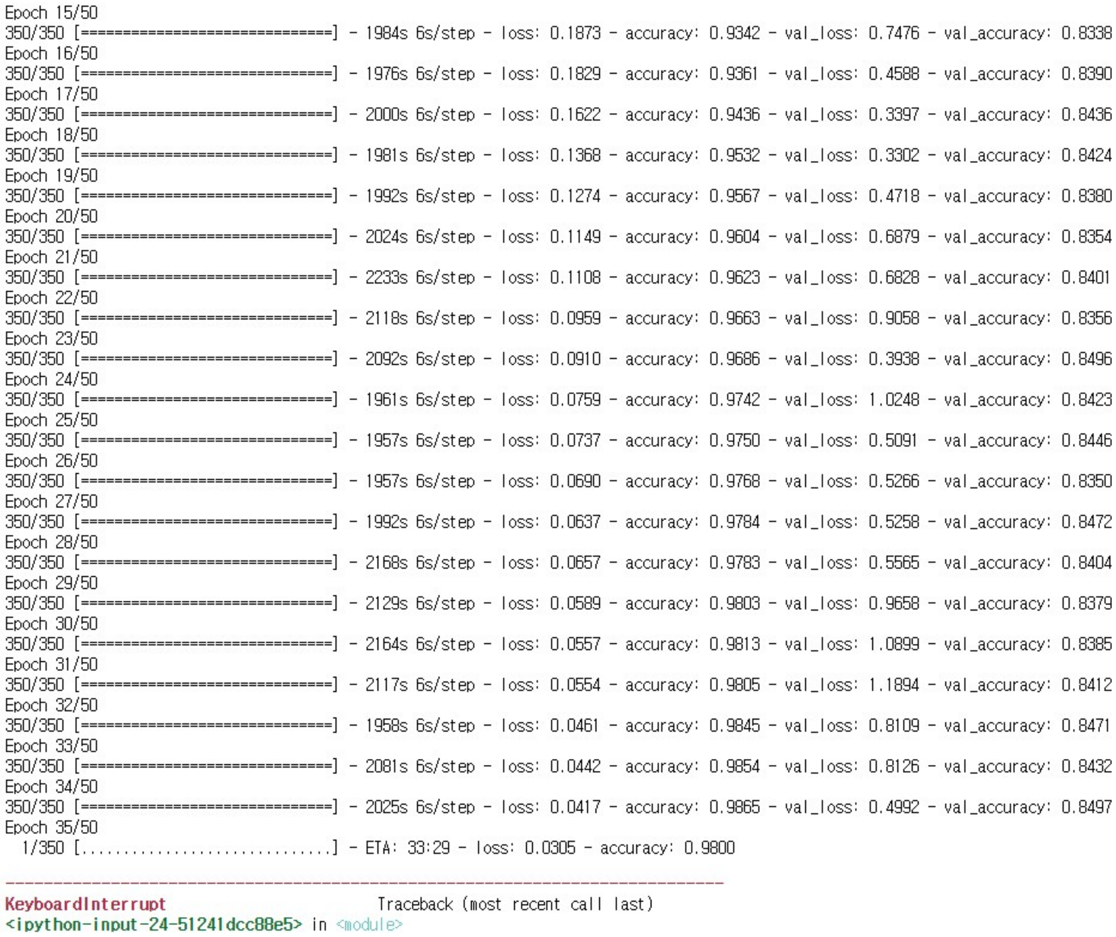
2020 Fall

딥러닝/클라우드 2차과제 레포트

Seokhyeon Lee(이석현) 32183164

목차

1. 실행 화면 캡처



```
Epoch 15/50
350/350 [==============================] - 1984s 6s/step - loss: 0.1873 - accuracy: 0.9342 - val_loss: 0.7476 - val_accuracy: 0.8338
Epoch 16/50
350/350 [==============================] - 1976s 6s/step - loss: 0.1829 - accuracy: 0.9361 - val_loss: 0.4588 - val_accuracy: 0.8390
Epoch 17/50
350/350 [==============================] - 2000s 6s/step - loss: 0.1622 - accuracy: 0.9436 - val_loss: 0.3397 - val_accuracy: 0.8436
Epoch 18/50
350/350 [==============================] - 1981s 6s/step - loss: 0.1368 - accuracy: 0.9532 - val_loss: 0.3302 - val_accuracy: 0.8424
Epoch 19/50
350/350 [==============================] - 1992s 6s/step - loss: 0.1274 - accuracy: 0.9567 - val_loss: 0.4718 - val_accuracy: 0.8380
Epoch 20/50
350/350 [==============================] - 2024s 6s/step - loss: 0.1149 - accuracy: 0.9604 - val_loss: 0.6879 - val_accuracy: 0.8354
Epoch 21/50
350/350 [==============================] - 2233s 6s/step - loss: 0.1108 - accuracy: 0.9623 - val_loss: 0.6828 - val_accuracy: 0.8401
Epoch 22/50
350/350 [==============================] - 2118s 6s/step - loss: 0.0959 - accuracy: 0.9663 - val_loss: 0.9058 - val_accuracy: 0.8356
Epoch 23/50
350/350 [==============================] - 2092s 6s/step - loss: 0.0910 - accuracy: 0.9686 - val_loss: 0.3938 - val_accuracy: 0.8496
Epoch 24/50
350/350 [==============================] - 1961s 6s/step - loss: 0.0759 - accuracy: 0.9742 - val_loss: 1.0248 - val_accuracy: 0.8423
Epoch 25/50
350/350 [==============================] - 1957s 6s/step - loss: 0.0737 - accuracy: 0.9750 - val_loss: 0.5091 - val_accuracy: 0.8446
Epoch 26/50
350/350 [==============================] - 1957s 6s/step - loss: 0.0690 - accuracy: 0.9768 - val_loss: 0.5266 - val_accuracy: 0.8350
Epoch 27/50
350/350 [==============================] - 1992s 6s/step - loss: 0.0637 - accuracy: 0.9784 - val_loss: 0.5258 - val_accuracy: 0.8472
Epoch 28/50
350/350 [==============================] - 2168s 6s/step - loss: 0.0657 - accuracy: 0.9783 - val_loss: 0.5565 - val_accuracy: 0.8404
Epoch 29/50
350/350 [==============================] - 2129s 6s/step - loss: 0.0589 - accuracy: 0.9803 - val_loss: 0.9658 - val_accuracy: 0.8379
Epoch 30/50
350/350 [==============================] - 2164s 6s/step - loss: 0.0557 - accuracy: 0.9813 - val_loss: 1.0899 - val_accuracy: 0.8385
Epoch 31/50
350/350 [==============================] - 2117s 6s/step - loss: 0.0554 - accuracy: 0.9805 - val_loss: 1.1894 - val_accuracy: 0.8412
Epoch 32/50
350/350 [==============================] - 1958s 6s/step - loss: 0.0461 - accuracy: 0.9845 - val_loss: 0.8109 - val_accuracy: 0.8471
Epoch 33/50
350/350 [==============================] - 2081s 6s/step - loss: 0.0442 - accuracy: 0.9854 - val_loss: 0.8126 - val_accuracy: 0.8432
Epoch 34/50
350/350 [==============================] - 2025s 6s/step - loss: 0.0417 - accuracy: 0.9865 - val_loss: 0.4992 - val_accuracy: 0.8497
Epoch 35/50
  1/350 [..............................] - ETA: 33:29 - loss: 0.0305 - accuracy: 0.9800

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-24-51241dcc88e5> in <module>
```

34번째 epoch 종료 후 keyboard interrupt 했습니다.

딥러닝/클라우드 2차과제 레포트

Seokhyeon Lee(이석현) 32183164

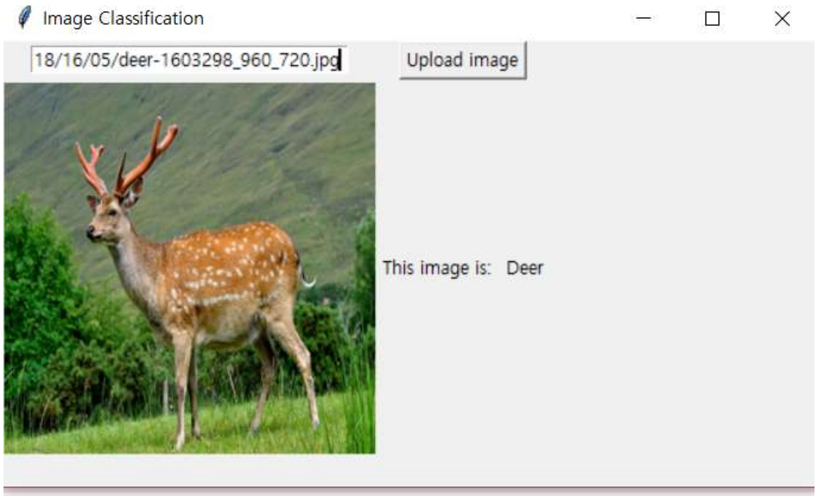
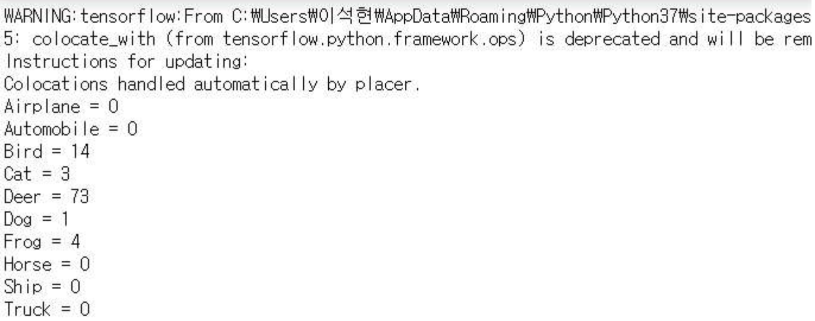


Image Classification

18/16/05/deer-1603298_960_720.jpg

This image is: Deer



```
WARNING:tensorflow:From C:₩Users₩이석현₩AppData₩Roaming₩Python₩Python37₩site-packages
5: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be rem
Instructions for updating:
Colocations handled automatically by placer.
Airplane = 0
Automobile = 0
Bird = 14
Cat = 3
Deer = 73
Dog = 1
Frog = 4
Horse = 0
Ship = 0
Truck = 0
```

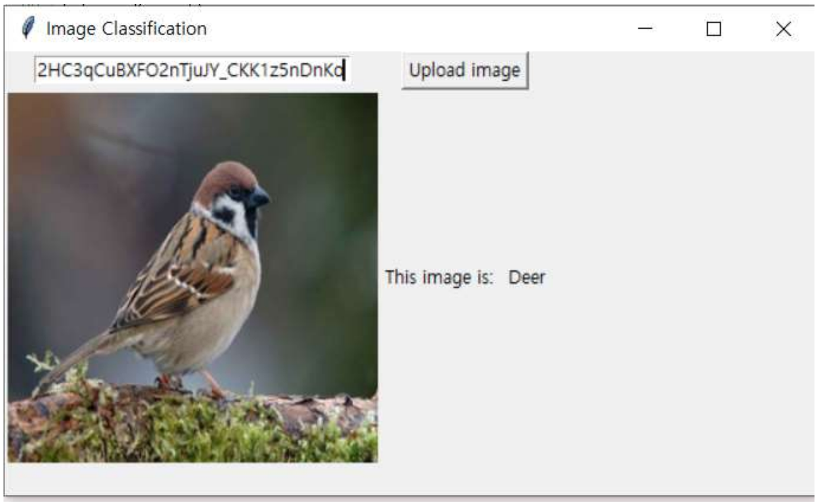
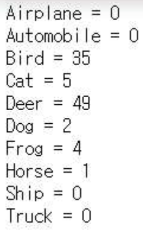


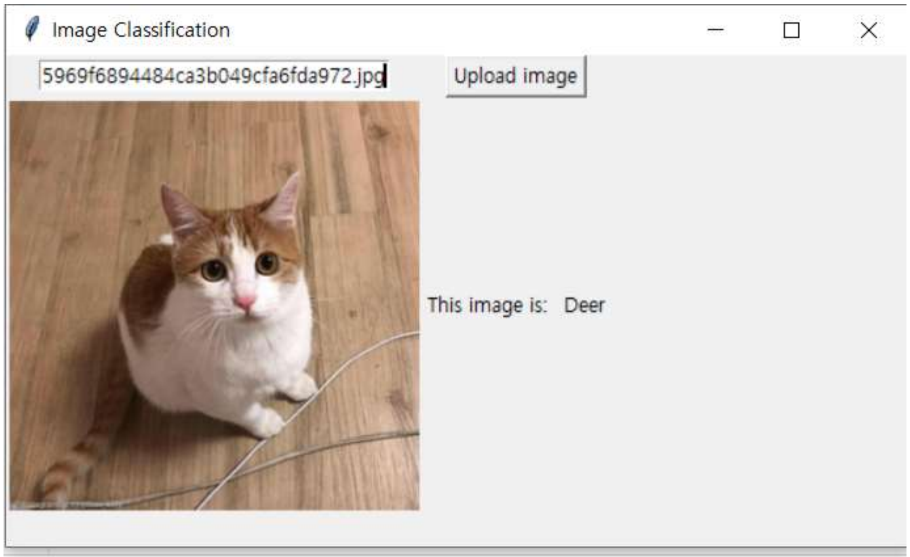
Image Classification

2HC3qCuBXFO2nTjuJY_CKK1z5nDnKo

This image is: Deer



```
Airplane = 0
Automobile = 0
Bird = 35
Cat = 5
Deer = 49
Dog = 2
Frog = 4
Horse = 1
Ship = 0
Truck = 0
```



5969f6894484ca3b049cfa6fda972.jpg

This image is: Deer

```
Airplane = 0
Automobile = 0
Bird = 24
Cat = 6
Deer = 57
Dog = 2
Frog = 6
Horse = 1
Ship = 0
Truck = 0
```

2. 소감

처음에 epoch 를 50 번으로 설정하고 시행하다가 34 번째 epoch 가 되어서야 validation accuracy 가 18 번째부터 줄어들고 있었다는 것을 확인하고 중단했습니다. Epoch 를 17 로 하고 딥러닝을 돌리면 최적의 값이 나올 것이지만 한 epoch 당 한 시간이 넘게 소요되기에 시간관계상 시도하지 못하였습니다. 결과적으로 이미지 분류에서도 overfitting 되어 deer 의 값이 계속 높게 나오는 문제가 발생하여 아쉬움이 남습니다.

Transfer learning 으로 이전에 개발되어 있는 모델을 이용해 새로운 학습 모델을 만들면서 이런 방식도 존재한다는 것에 흥미를 느꼈고, 그래서인지 재미있게 모델을 설계할 수 있었습니다. 다만 아쉬운 점이 있다면, 한 모델을 개발하는데 걸리는 러닝타임이 길어, 다양한 시도를 해보지 못한 아쉬움이 남습니다. 또한 tkinter 를 이용해 GUI 를 구현했는데 이 패키지를 오랜만에 사용하다 보니 낯선 부분이 있어 시간이 꽤 걸렸고, 하이퍼링크를 입력 받아 사진을 다운받고 업로드 하는데 생각보다 많은 오류가 발생해 이를 수정하는데 어려움이 있었습니다. 그래도 결국 오류를 해결해 프로그램을 동작 시킬 수 있어 뿌듯합니다.

3. 모델 개발용 소스코드

```
import numpy as np
import pandas as pd
from sklearn.utils.multiclass import unique_labels
import os

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline
import itertools
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

from keras import Sequential
from keras.applications import VGG19
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import SGD,Adam
from keras.callbacks import ReduceLROnPlateau
from keras.layers import Flatten,Dense,BatchNormalization,Activation,Dropout
from keras.utils import to_categorical
from keras.datasets import cifar10

#develop model
(train_X,train_y),(test_X,test_y)=cifar10.load_data()
train_X,val_X,train_y,val_y=train_test_split(train_X,train_y,test_size=.3)
train_y=to_categorical(train_y)
val_y=to_categorical(val_y)
test_y=to_categorical(test_y)

train_generator = ImageDataGenerator(rotation_range=2,
horizontal_flip=True, zoom_range=.1 )
val_generator = ImageDataGenerator(rotation_range=2,
horizontal_flip=True, zoom_range=.1)
test_generator = ImageDataGenerator(rotation_range=2, horizontal_flip=
True, zoom_range=.1)

train_generator.fit(train_X)
val_generator.fit(val_X)
test_generator.fit(test_X)

lrr= ReduceLROnPlateau(monitor='val_acc', factor=.01, patience=3,
min_lr=1e-5)

base_model_1 =
VGG19(include_top=False,weights='imagenet',input_shape=(32,32,3),classes
=train_y.shape[1])

model_1= Sequential()
model_1.add(base_model_1)
model_1.add(Flatten())
model_1.add(Dense(1024,activation=('relu'),input_dim=512))
model_1.add(Dense(512,activation=('relu')))
model_1.add(Dense(256,activation=('relu')))
#model_1.add(Dropout(.3))
model_1.add(Dense(128,activation=('relu')))
#model_1.add(Dropout(.2))
model_1.add(Dense(10,activation=('softmax')))

batch_size= 100
epochs=50
learn_rate=.001

sgd=SGD(lr=learn_rate,momentum=.9,nesterov=False)
adam=Adam(lr=learn_rate, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0, amsgrad=False)
model_1.compile(optimizer=sgd,loss='categorical_crossentropy',metrics=['accuracy'])
model_1.fit_generator(train_generator.flow(train_X,train_y,batch_size=batch_size), epochs=epochs, steps_per_epoch=train_X.shape[0]//batch_size, validation_data=val_generator.flow(val_X,val_y,batch_size=batch_size),validation_steps=250, callbacks=[lrr], verbose=1)

#save model
model_1.save('cifar10_model.h5')
model_json = model_1.to_json()
with open("model.json", "w") as json_file :
    json_file.write(model_json)
model_1.save_weights("model.h5")
```

4. 이미지 분류 SW 소스 코드

```
from tkinter import *
import urllib.request
import os

from PIL import ImageTk, Image

def img_processing():
    from keras.models import model_from_json
    json_file = open("model.json", "r")
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
    loaded_model.load_weights("model.h5")
    loaded_model.compile(loss="binary_crossentropy", optimizer="rmsprop", metrics=['accuracy'])

    import cv2
    im = cv2.imread('test.jpg')

    im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
    im = cv2.resize(im, (32,32))

    in_size = 32*32*3

    import numpy as np
    im = im.reshape(-1,32,32,3)/ 255

    r = loaded_model.predict(im)

    res = r[0]
    labels = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog',
'Frog', 'Horse', 'Ship', 'Truck']
    for i, acc in enumerate(res) :
        print(labels[i], "=", int(acc*100))
    return labels[res.argmax()]

def download_img():
    url = txt.get()
    #print(url)
    os.system("curl " + url + " > test.jpg")
    open_img()

def open_img():
    x = 'C:/Users/이석현/딥러닝/test.jpg'
    img = Image.open(x)
    img = img.resize((250, 250), Image.ANTIALIAS)
    img = ImageTk.PhotoImage(img)
    panel = Label(root, image = img)
    panel.image = img
    panel.grid(row = 2)
    result = img_processing()
    lbl3 = Label(root, text=result)
    lbl3.grid(row=2,column=2)

root = Tk()
root.title("Image Classification")
root.geometry("550x300+300+150")
root.resizable(width = True, height = True)
txt = Entry(root, width=30)
txt.grid(row=1)

btn = Button(root, text ='Upload image', command = download_img).grid(row = 1, column=1, columnspan = 4)
lbl2 = Label(root, text="This image is: ")
lbl2.grid(row=2,column=1)

root.mainloop()
```