딥러닝 이해

기말프로젝트 보고서

20174119 한지훈

기말프로젝트 주제: CNN을 이용한 이미지 분류

7개 브랜드의 자동차 이미지를 학습한 모델을 이용하여 자동차 이미지만으로 어떤 브랜드의 자동차인지 분류하도록 하는 프로젝트

다양한 자동차 이미지파일이 필요하고 그 데이터를 기반으로 학습할 모델을 설계하는것이 중요하다.

데이터는 kaggle에 있는 데이터셋 (https://www.kaggle.com/datasets/kshitij192/cars-image-dataset) 을 사용하였고 모델은 CNN을 사용하였다.

더 효율적인 모델의 학습을 위해 데이터를 전처리 하는 과정을 거쳤다.

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True

)

이미지의 픽셀 값을 0과 1 사이로 조정하여 정규화하고 회전은 최대20도까지 좌우 및 상하로 이동시킬 수 있는 범위는 최대 20%로 설정하고 이미지변형 최대 20%까지만 허용 확대는 최대 20%까지 수평으로 뒤집는 것을 허용하여 데이터를 전처리하였다.

test\_datagen = ImageDataGenerator(rescale=1. / 255)

테스트 데이터는 픽셀값만 정규화하여 전처리 하였다.

train\_generator = train\_datagen.flow\_from\_directory(

train\_data\_dir,

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical'

)

# 테스트 데이터 로딩 및 전처리

test\_generator = test\_datagen.flow\_from\_directory(

test\_data\_dir,

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical'

)

# 검증 데이터 로딩 및 전처리

val\_generator = test\_datagen.flow\_from\_directory(

test\_data\_dir,

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False

)

이미지를 리사이즈하고 다중클래스분류문제로 설정한다. 검증데이터의 경우 순서를 섞을 필요가없기때문에 shuffle=false로 설정한다.

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_height, img\_width, 3)))

model.add(MaxPooling2D((2, 2)))

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

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(train\_generator.num\_classes, activation='softmax'))

Sequantial모델을 생성하고 Conv2D 레이어를 추가한다. 필터수는 32개 크기는 (3,3) 활성화함수는 'relu' 입력이미지는 150x150크기의 컬러 이미지로 지정한다.

그리고  MaxPooling2D 레이어를 추가하여 최대풀링연산을 수행하게하고 이 과정을 통해 이미지 크기를 줄인다. 그다음으로도 Conv2D 레이어를 3번 더 추가하고 MaxPooling2D 레이어도 마찬가지로 더 추가한다.

그리고 과적합문제를 해결하기 위해 dropout을 이용하여 일부뉴런을 비활성화한다.

ptimizer = Adam(learning\_rate=0.001)

model.compile(optimizer=optimizer, loss='categorical\_crossentropy', metrics=['accuracy'])

def lr\_schedule(epoch):

lr = 0.001

if epoch > 100:

lr \*= 0.1

elif epoch > 50:

lr \*= 0.5

return lr

# 모델 학습

lr\_scheduler = LearningRateScheduler(lr\_schedule)

history = model.fit(

train\_generator,

epochs=num\_epochs,

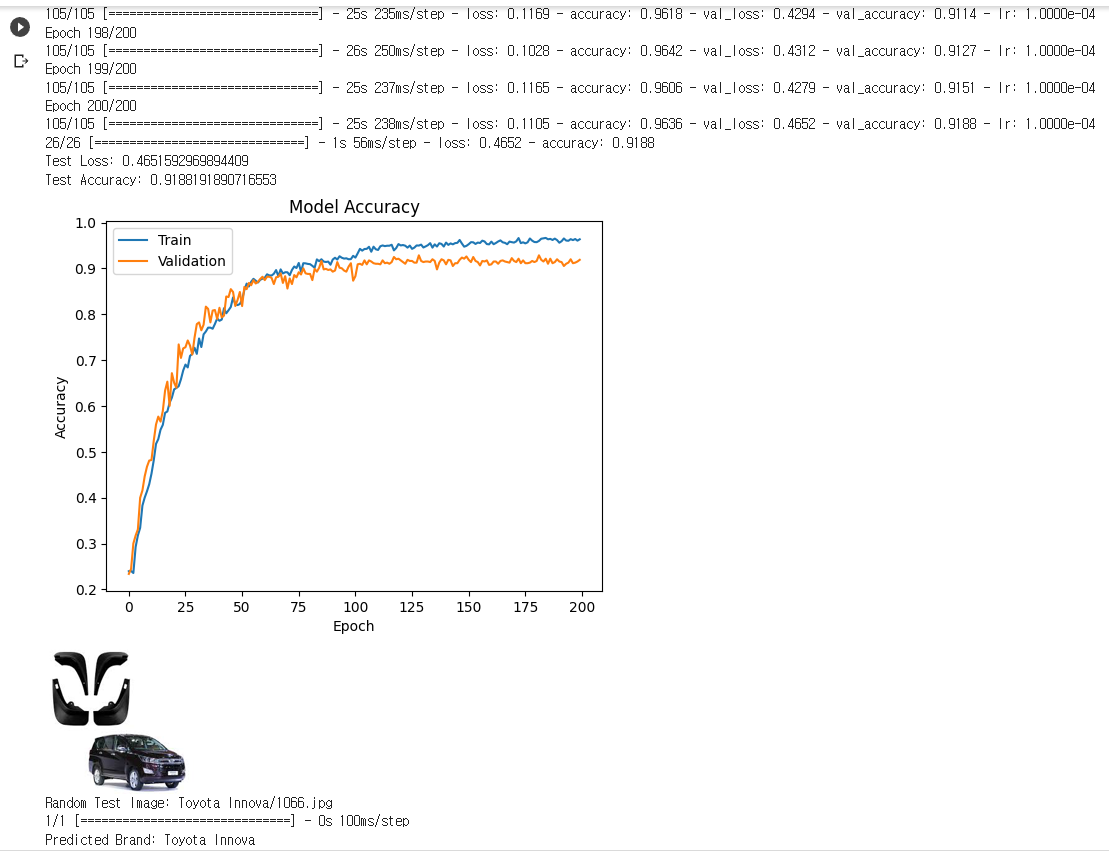
callbacks=[lr\_scheduler],

validation\_data=val\_generator

)

그리고 옵티마이저는 adam으로 설정하고 categorical crossentropy로 설정하여 다중클래스 분류문제를 해결할 수 있도록한다. 그리고 평가지표는 정확도로 설정하고 epoch에따라 학습률을 변화시켜 모델의 성능을 개선시킨다. Epoch가100을 넘을경우 학습률을 0.1배 50을 넘을경우 0.5배 하는방식으로 학습률을 변화시킨다.

그 후 테스트 데이터로 모델의 성능을 평가하고 이를 그래프를 통해 시각화한다.



테스트 데이터보다 훈련데이터에서 정확도가 높긴하지만 테스트 데이터도 0.9이상의 정확도를 내는 것을 알 수 있다. 훈련데이터의 경우150epoch부터 테스트 데이터의 경우 105epoch부터 정확도에 큰 변화가 없다는것을 알 수 있다.

영상링크

[Untitled5 ipynb Colaboratory Chrome 2023 06 15 23 08 43](https://youtu.be/OhuJI41p7hg)

[](https://youtu.be/OhuJI41p7hg)

학습 과정은 제외하고 학습한 모델을 이용하여 인터넷에서 따로 다운받은 이미지들로 예측을 실행하는 영상이다.

전체코드

import os

import numpy as np

from PIL import Image

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import LearningRateScheduler

import matplotlib.pyplot as plt

# 데이터 경로 설정

train\_data\_dir = '/content/Cars Dataset/train'

test\_data\_dir = '/content/Cars Dataset/test'

img\_width, img\_height = 150, 150

batch\_size = 32

num\_epochs = 200

# 데이터 생성기 생성 및 전처리 설정

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True

)

test\_datagen = ImageDataGenerator(rescale=1. / 255)

# 훈련 데이터 로딩 및 전처리

train\_generator = train\_datagen.flow\_from\_directory(

train\_data\_dir,

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical'

)

# 테스트 데이터 로딩 및 전처리

test\_generator = test\_datagen.flow\_from\_directory(

test\_data\_dir,

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical'

)

# 검증 데이터 로딩 및 전처리

val\_generator = test\_datagen.flow\_from\_directory(

test\_data\_dir,

target\_size=(img\_height, img\_width),

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False

)

# CNN 모델 정의

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_height, img\_width, 3)))

model.add(MaxPooling2D((2, 2)))

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

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(train\_generator.num\_classes, activation='softmax'))

# 모델 컴파일

optimizer = Adam(learning\_rate=0.001)

model.compile(optimizer=optimizer, loss='categorical\_crossentropy', metrics=['accuracy'])

# 학습률 스케줄링을 위한 함수

def lr\_schedule(epoch):

lr = 0.001

if epoch > 100:

lr \*= 0.1

elif epoch > 50:

lr \*= 0.5

return lr

# 모델 학습

lr\_scheduler = LearningRateScheduler(lr\_schedule)

history = model.fit(

train\_generator,

epochs=num\_epochs,

callbacks=[lr\_scheduler],

validation\_data=val\_generator # 검증 데이터 지정

)

# 모델 평가

test\_loss, test\_accuracy = model.evaluate(test\_generator)

print('Test Loss:', test\_loss)

print('Test Accuracy:', test\_accuracy)

# 정확도 변화 그래프 출력

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()

# 랜덤으로 테스트 이미지 선택 및 예측

random\_index = np.random.randint(len(test\_generator.filenames))

random\_image = test\_generator.filenames[random\_index]

random\_image\_path = os.path.join(test\_data\_dir, random\_image)

random\_label = test\_generator.classes[random\_index]

class\_labels = list(test\_generator.class\_indices.keys())

image = Image.open(random\_image\_path).resize((img\_height, img\_width))

image.show()

print('Random Test Image:', random\_image)

image\_array = np.array(image)

print('Predicted Brand:', class\_labels[np.argmax(model.predict(np.expand\_dims(image\_array, axis=0)))])