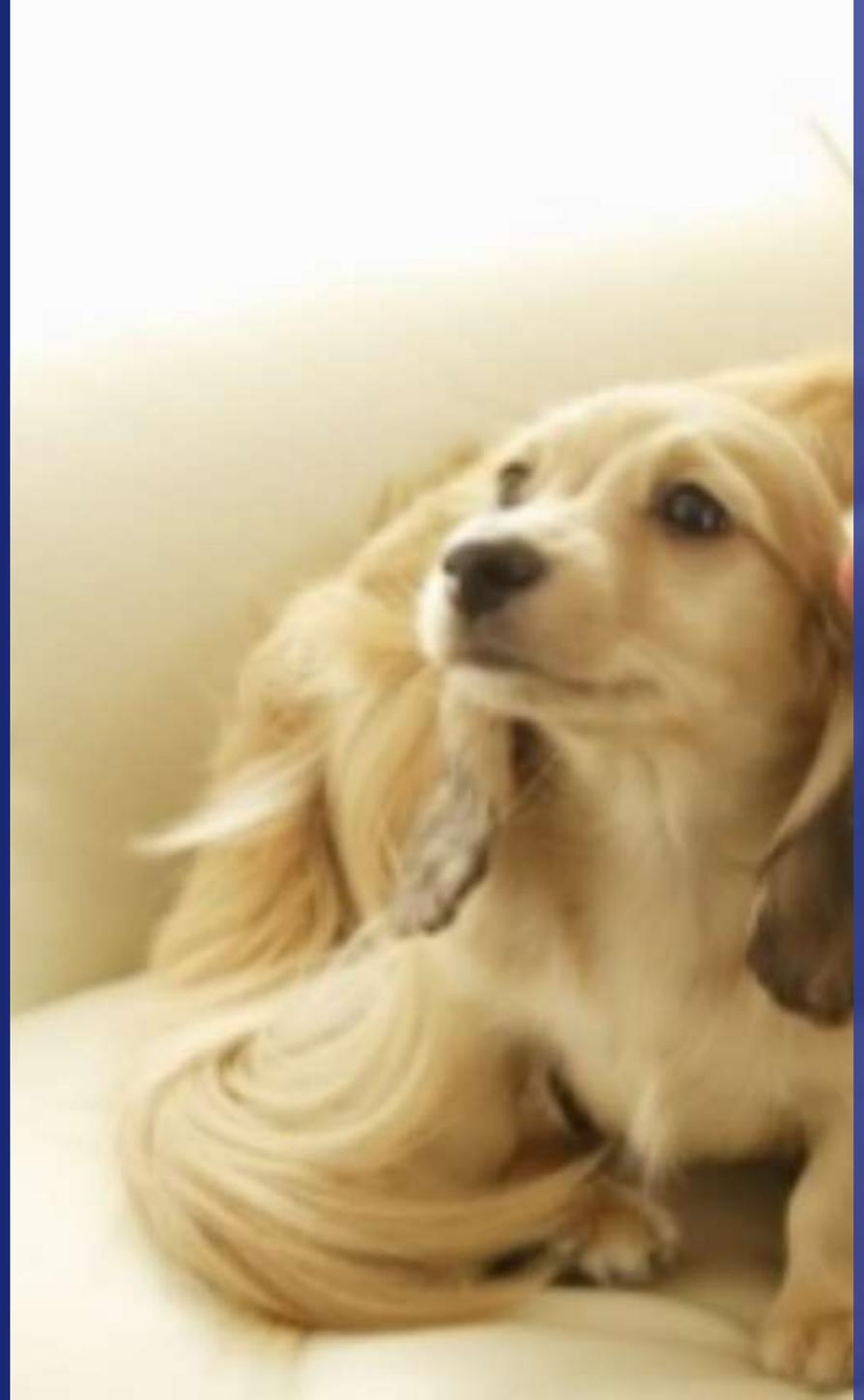


Dog BreeD Fractionation

KYUNG BOK UNIVERSITY

4조

이준호 권혁준 정진우
이길헌 김동현 조범수 변
경수

KYUNG BOK UNIVERSITY

CONTENTS

01. 개요

개요

이 프로그램을 만든 이유

요즘은 많은 사람들이 반려견에 관심이 많기 때문입니다.

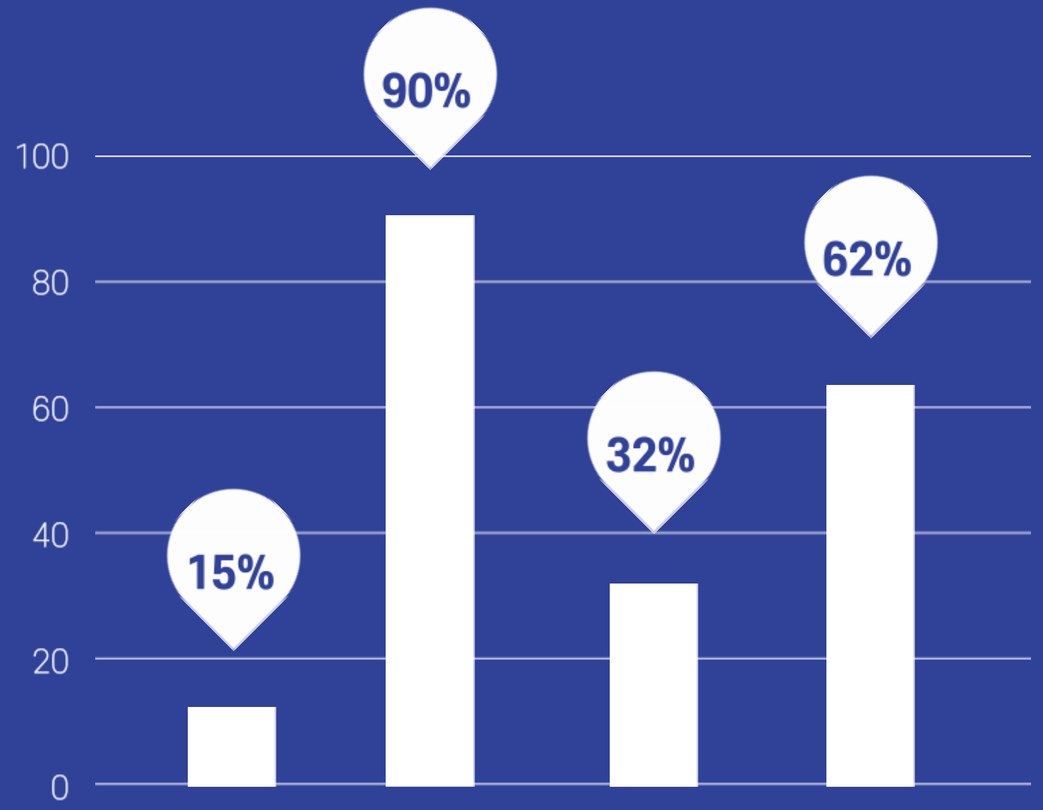
하지만 품종이 워낙 다양해졌기 문에 품종에 대해 잘 모르는 사람도 많아졌습니다.

그래서 저희는 품종 분류프로그램을 만들어 보자 생각했습니다.

02. 프로젝트 소개

프로젝트 소개

강아지 품종 분류 프로그램 입니다.



AI 학습,훈련

프로그램에서 AI를 학습 및 훈련을 시켜줍니다.

원하는 사진 입력

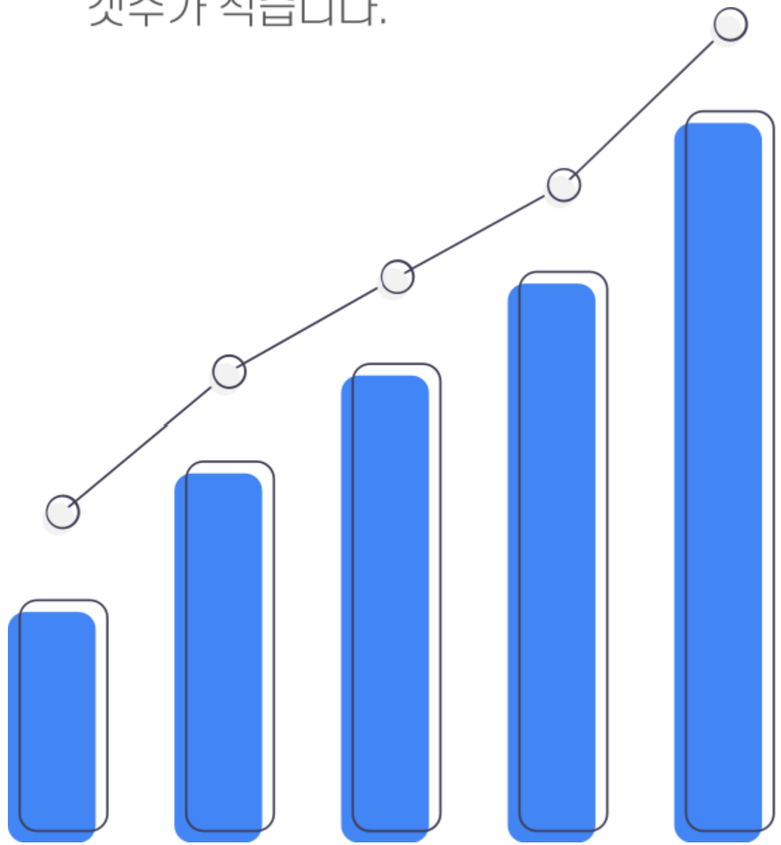
품종이 알고싶은 강아지의 사진을 넣어줍니다.

품종 예측 및 정확도

어떤 품종인지 예측하여 대답을해줍니다 정확도 또한 알려줍니다.

프로젝트 소개

아직은 초반이라 정확도가 엄청 뛰어나거나 분류가능한 품종의 갯수가 적습니다.



더 많은 품종

지금 보다 더 많은 품종을 분류할 수있도록 할 예정입니다.

높은 정확도

더 많은 사진들을 학습시켜 정확도를 높일 예정입니다.

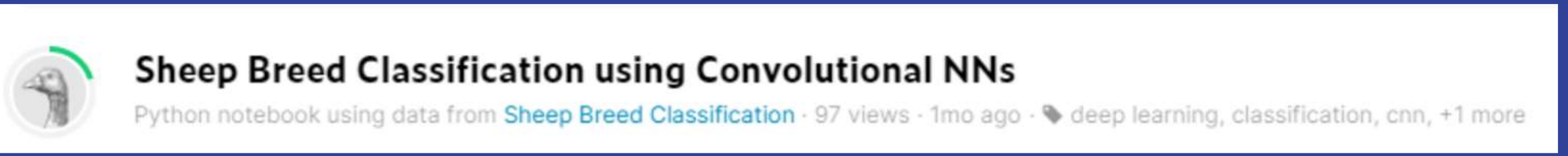
쉬운 사진넣기

아직은 원하는 사진을 넣는것에 있어 복잡함이 있습니다.
코드를 더 잘 바꾸어 쉽게 넣을수 있게할 예정입니다.

03. 코드 설명

참고한 코드

양 품종 분류 코드를 참고하였습니다.



kaggle 에서 찾은 양 품종 분류프로그램 코드입니다.

https://www.kaggle.com/frederikolsen/sheep-breed-classification-using-convolutional-nns

양사진을 강아지로 대체하고 안의 코드들을 불필요한것은 지우고 필요한것은 더하는식으로 참고하였습니다.

notebook8 Draft saved

File Edit View Run Add-ons Help

Run All

[84]:

```
# This Python 3 environment comes with many helpful analytics librari
# It is defined by the kaggle/python Docker image: https://github.com
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import random
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img

# Input data files are available in the read-only "../input/" directo
# For example, running this (by clicking run or pressing Shift+Enter)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/labels/labels.csv
/kaggle/input/dog-breed/dog/h/h37.jpg
/kaggle/input/dog-breed/dog/h/h66.jpg
/kaggle/input/dog-breed/dog/h/h44.jpg
/kaggle/input/dog-breed/dog/h/h51.jpg
/kaggle/input/dog-breed/dog/h/h58.jpg
/kaggle/input/dog-breed/dog/h/h17.jpg
/kaggle/input/dog-breed/dog/h/h68.jpg
/kaggle/input/dog-breed/dog/h/h2.jpg
/kaggle/input/dog-breed/dog/h/h55.jpg
/kaggle/input/dog-breed/dog/h/h100.jpg
```

코드 설명

Kaggle

저희는 Kaggle에서 실행시켰습니다.
다른 곳에서는 아직 실행을 못시켜봤습니다.

기본적인 프로그램 import

내부에서 써야할것들을 import 해줍니다.

사진 넣기

필요한 사진을 kaggle 디렉토리에 넣어놨기 때문에
디렉토리에서 사진을 받아줍니다.

다음과 같이 사진이 다운되는것을 볼수있습니다.

```
#name of the directories
os.listdir("../input/dog-breed/dog")
```

```
['h', 'd', 'p', 's', 'w']
```

디렉토리 읽어오기

디렉토리 리스트들을 가져옵니다
5개의 디렉토리가 있습니다.

디렉토리 리네임

디렉토리들의 이름을 알기쉽게 바꿔줍니다.

```
doberman = "/kaggle/input/dog-breed/dog/d/"
husky = "/kaggle/input/dog-breed/dog/h/"
poodle = "/kaggle/input/dog-breed/dog/p/"
siba = "/kaggle/input/dog-breed/dog/s/"
welshcogi = "/kaggle/input/dog-breed/dog/w/"
```

데이터 전처리

데이터 셋을 만들어주고 그 안에 들어가는
사진들의 사이즈를 (150, 150) 로 바꿔줍니다.

```
from tqdm import tqdm
x = []
y = []

def create_dataset(dirname,breedname):
    for i in tqdm(os.listdir(dirname)):
        path = os.path.join(dirname,i)
        try:
            img = cv2.imread(path)
            img = cv2.resize(img,(150,150))
        except:
            continue

        x.append(img)
        y.append(breedname)
    return x,y
```

```
x,y = create_dataset(doberman,"doberman")
x,y = create_dataset(husky,"husky")
x,y = create_dataset(poodle,"poodle")
x,y = create_dataset(siba,"siba")
x,y = create_dataset(welshcogi,"welshcogi")
```

```
100%|██████████| 106/106 [00:00<00:00, 309.99it/s]
100%|██████████| 100/100 [00:00<00:00, 338.16it/s]
100%|██████████| 100/100 [00:00<00:00, 172.58it/s]
100%|██████████| 106/106 [00:00<00:00, 331.00it/s]
100%|██████████| 100/100 [00:00<00:00, 318.16it/s]
```

```
plt.figure(figsize = (12,7))
for i in range(10):
    indx = random.randint(0,len(y))
    img = x[indx]
    plt.subplot(2,5,i+1)
    plt.subplots_adjust(hspace=0.3)
    plt.imshow(img)
    plt.xlabel(y[indx])

plt.tight_layout()
plt.show()
```

데이터 셋 만들기

디렉토리내의 사진들을 데이터셋으로 만들어줍니다.

사진 확인하기

데이터셋에 넣은 사진들을 랜덤하게 10개 가져와
잘 들어갔는지 확인해줍니다.

```
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
le = LabelEncoder()
y = le.fit_transform(y)
y = to_categorical(y)
```

```
#divide our dataset into train & test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

```
img_size=150
x_train = np.array(x_train)/255.0
x_test = np.array(x_test)/255.0

x_train = x_train.reshape(-1,img_size,img_size,3)
y_train = np.array(y_train)

x_test = x_test.reshape(-1,img_size,img_size,3)
y_test = np.array(y_test)
```

모델링 하기

모델링을 시작해줍니다.

인코딩하기

CNN 모델은 문자열값을 이해하지못합니다.
그래서 우리는 목표값을 숫자 값으로 변환한다음
to_categorical 을 이용하여 해당값을 하나의 인코딩값
으로 변환해줍니다.

```
from tensorflow.keras.applications.vgg19 import VGG19
vgg = VGG19(weights = "imagenet",include_top = False,input_shape=(150,150,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/v
80142336/80134624 [==============================] - 1s 0us/step
```

```
for layer in vgg.layers:
    layer.trainable = False
```

```
model2 = Sequential()
model2.add(vgg)
model2.add(Flatten())
model2.add(Dense(5,activation = "softmax"))

model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 4, 4, 512)         20024384
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 40965
=================================================================
Total params: 20,065,349
Trainable params: 40,965
Non-trainable params: 20,024,384
_________________________________________________________________
```

완성된 모델입니다.

vgg 라는 모델을 가져왔습니다.

학습 시키기

총 10번의 학습을 시켜주었습니다.
오답률은 줄어들고 정답률은 늘어나는것을 볼수있습니다.

```
Epoch 1/10
13/13 [==============================] - ETA: 0s - loss: 1.4293 - accuracy: 0.4205
Epoch 00001: val_accuracy improved from -inf to 0.55340, saving model to vgg19.h5
13/13 [==============================] - 69s 5s/step - loss: 1.4293 - accuracy: 0.4205 - val_loss: 1.0621 - val_accuracy: 0.5534
Epoch 2/10
13/13 [==============================] - ETA: 0s - loss: 0.6730 - accuracy: 0.7555
Epoch 00002: val_accuracy improved from 0.55340 to 0.64078, saving model to vgg19.h5
13/13 [==============================] - 69s 5s/step - loss: 0.6730 - accuracy: 0.7555 - val_loss: 0.8583 - val_accuracy: 0.6408
 Epoch 10/10
 13/13 [==============================] - ETA: 0s - loss: 0.0684 - accuracy: 0.9976
 Epoch 00010: val_accuracy did not improve from 0.76699
 13/13 [==============================] - 69s 5s/step - loss: 0.0684 - accuracy: 0.9976 - val_loss: 0.6366 - val_accuracy: 0.7670
```

```
loss2,accuracy2 = model2.evaluate(x_test,y_test)
print(f"Loss for vgg19: {loss2}")
print(f"Accuracy for vgg19: {accuracy2}")
```

```
4/4 [==============================] - 10s 2s/step - loss: 0.6366 - accuracy: 0.7670
Loss for vgg19: 0.6365630030632019
Accuracy for vgg19: 0.7669903039932251
```

```
print(classification_report(y_test_vgg,y_pred))
```

	precision	recall	f1-score	support
0	0.94	0.80	0.86	20
1	0.70	0.93	0.80	15
2	0.96	0.92	0.94	24
3	0.56	0.65	0.60	23
4	0.75	0.57	0.65	21
accuracy			0.77	103
macro avg	0.78	0.77	0.77	103
weighted avg	0.78	0.77	0.77	103

확인해보기

학습이 마무리 된 후 정답률과 오답률을 보기쉽게 나타내줍니다.

학습 시키기

디렉토리 별 학습량을 보여줍니다.

디렉토리 명

0 = Doberman
1 = Husky
2 = Poodle
3 = Shiba
4 = Welshcorgi

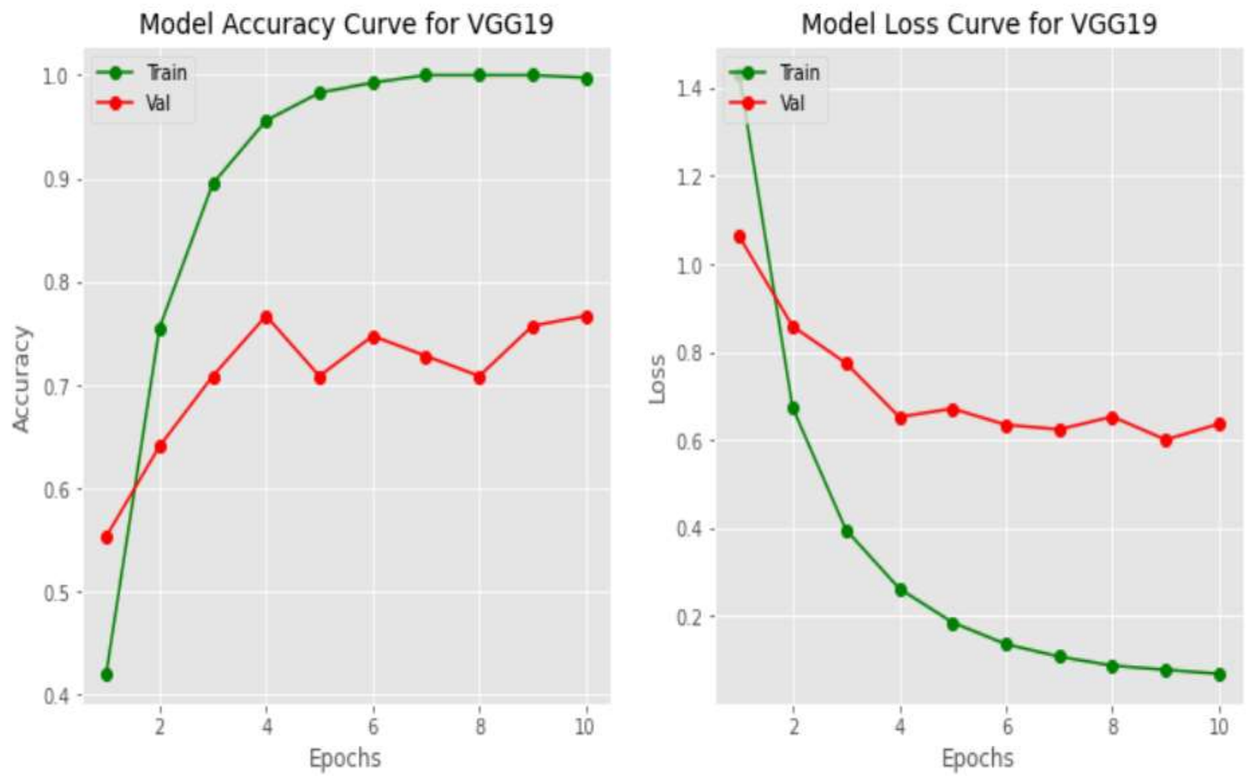
```python
def plot_learning_curve_for_vgg(history,epochs):
    plt.style.use("ggplot")
    plt.figure(figsize=(12,6))
    epochs = np.arange(1,epochs+1)
    plt.subplot(1,2,1)
    plt.plot(epochs,history.history["accuracy"],"go-")
    plt.plot(epochs,history.history["val_accuracy"],"ro-")
    plt.title("Model Accuracy Curve for VGG19")
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.legend(["Train","Val"],loc = "upper left")

    plt.subplot(1,2,2)
    plt.plot(epochs,history.history["loss"],"go-")
    plt.plot(epochs,history.history["val_loss"],"ro-")
    plt.title("Model Loss Curve for VGG19")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend(["Train","Val"],loc = "upper left")
    plt.show()
```

모델이 구버전이라 그런지 정답률과 오답률이 처참합니다...

시각화

한눈에 알아보기 쉽게 시각화 표를 만들어줍니다.

```python
plot_learning_curve_for_vgg(history2,10)
```

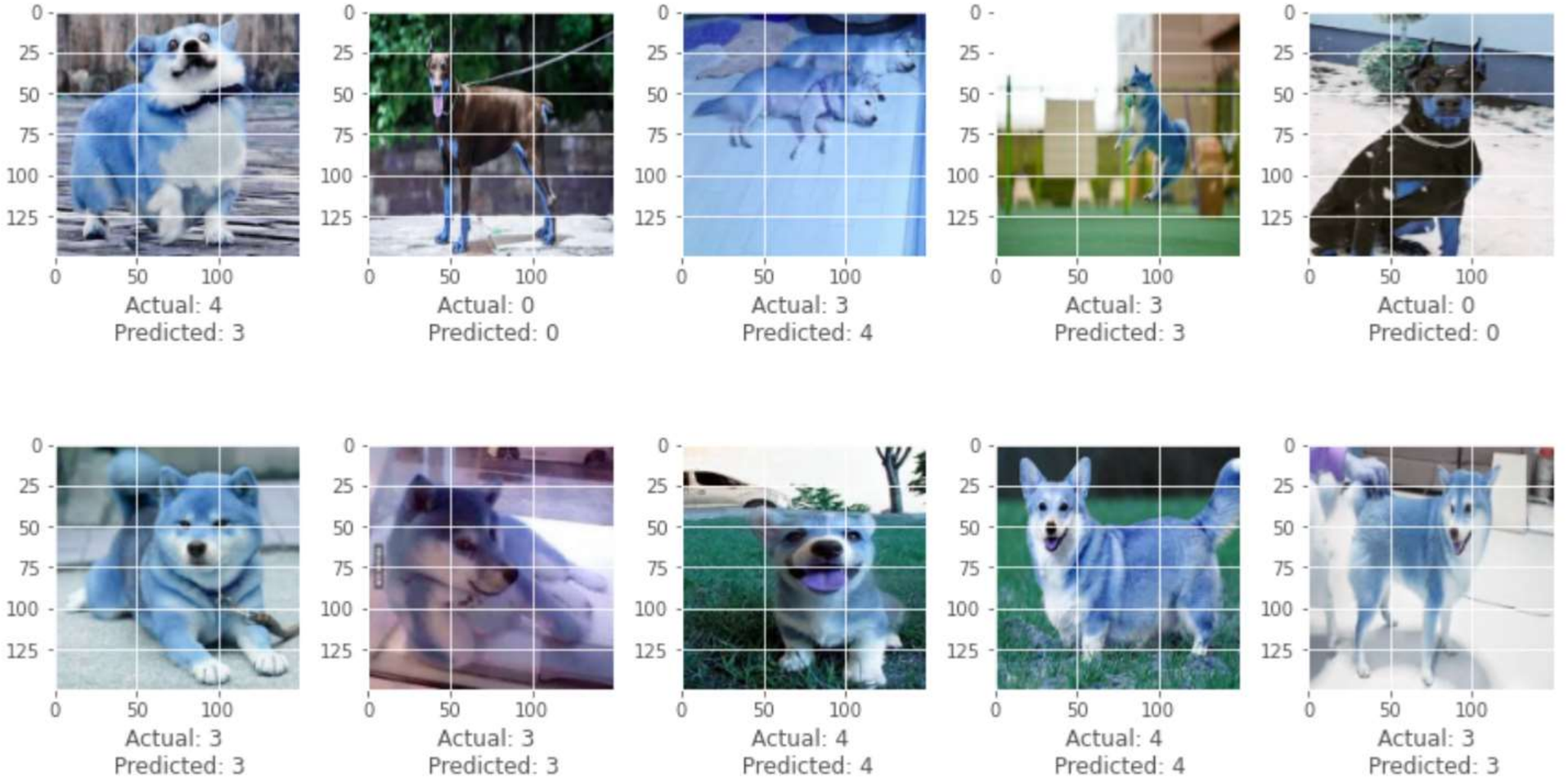


```python
plt.figure(figsize=(12,7))
for i in range(10):
    indx = random.randint(0,len(y_test_vgg))
    plt.subplot(2,5,i+1)
    plt.subplots_adjust(hspace=0.3)
    plt.imshow(x_test[indx])
    plt.xlabel(f"Actual: {y_test_vgg[indx]} \n Predicted: {y_pred[indx]}")

plt.tight_layout()
plt.show()
```

예측해보기

랜덤하게 사진을 뽑아와 어떤 품종인지 예측시켜보았습니다.

정답률이 낮아서그런지 실패한것들도 꽤나 많습니다.



디렉토리 명

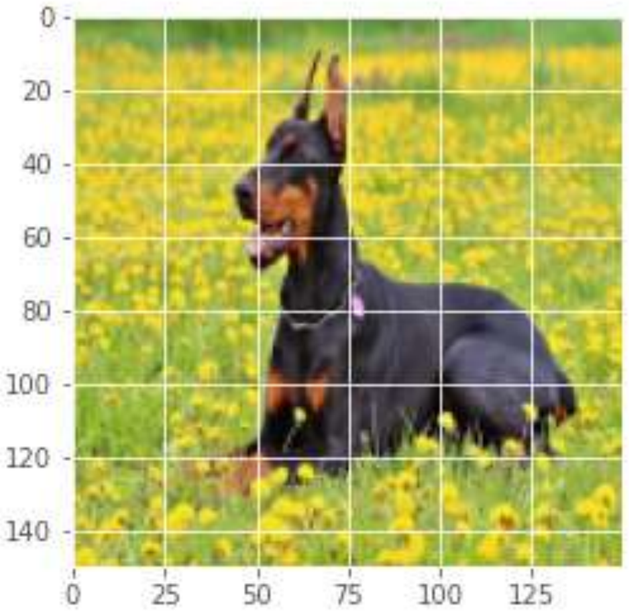
0 = Doberman
1 = Husky
2 = Poodle
3 = Shiba
4 = Welshcorgi

```
img = imread("/kaggle/input/ddtest/4.jpg")

plt.imshow(img)
plt.show()

img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model2.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "이 강아지의 품종은 {} 이고 정확도는 {:.2f} 퍼센트가 나왔습니다."
    .format(class_names[np.argmax(score)], 100 * accuracy2)
)
```

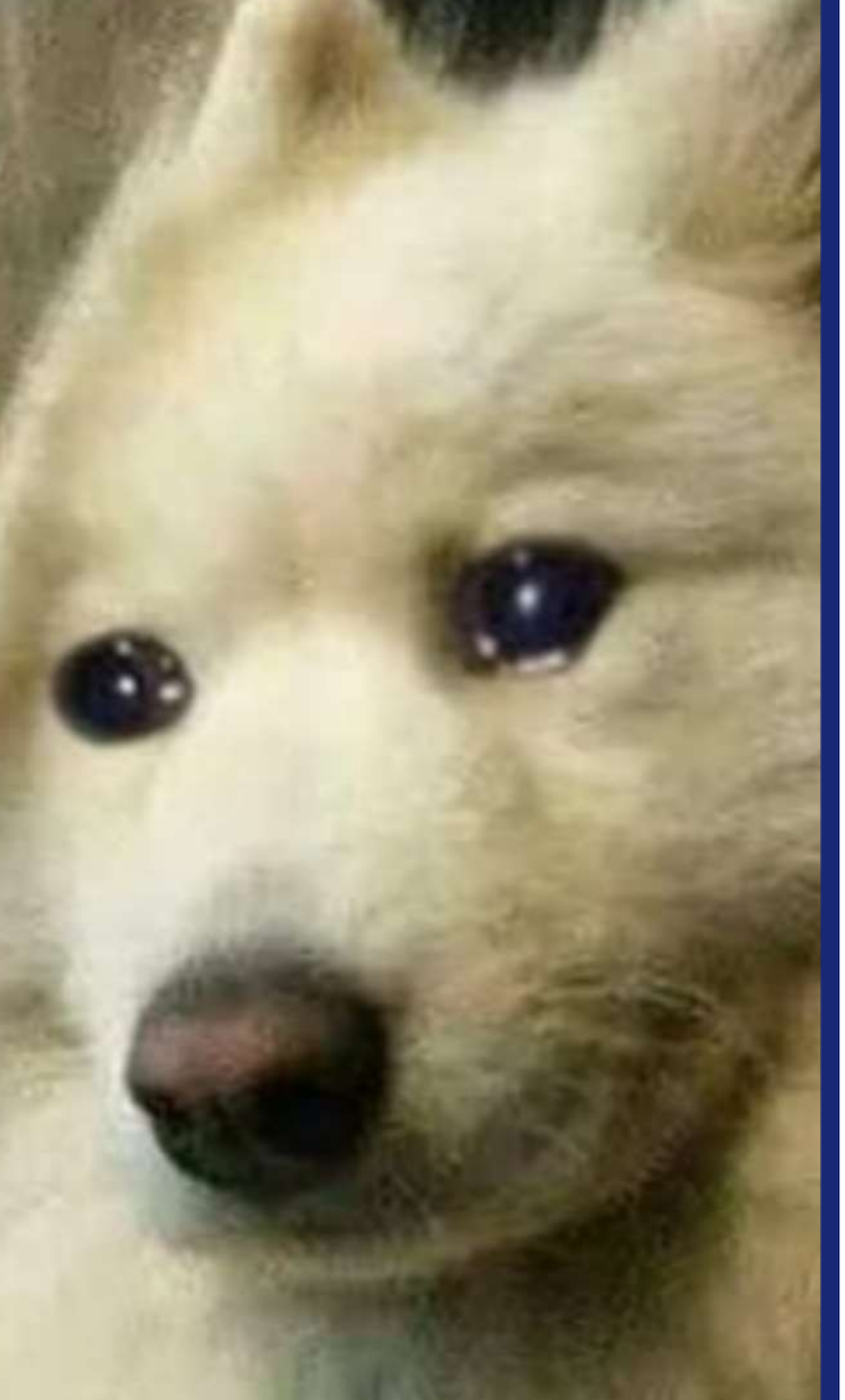
품종은 맞췄습니다.
하지만 모델의 문제인지 학습의 문제인지
정확도는 약 81퍼센트가 나왔습니다.

사진 직접 넣어보기

저희가 사진을 직접골라서 하나 넣어 예측시켜보았습니다.
사진은 도베르만 입니다.




```
이 강아지의 품종은 Doberman 이고 정확도는 81.55 퍼센트가 나왔습니다.
```



다음번에는...

정확도를 높이는 방법

다른 모델

모델을 최신모델을 찾아 바꿔보기

더 많은 데이터

사진의 양을 지금보다 더많은 양을 넣어보기

학습량 늘리기

학습량을 늘려 보기

Thank You