AnimaR

-강아지 이미지학습을 통한 품종 별 특징 제공 및 반려동물에 관한 정보제공 -

팀명 : DoGtor
팀원 : 송동익
신성보
정지현

목차

Ⅰ. 배경 및 목적

1. 프로젝트 개요

반려동물을 가족처럼 대하는 펫 팸족(Pet+Family), 반려동물을 사람처럼 대하는 펫 휴머니제이션(Pet+Humanization)과 같은 신조어 생성을 통해 반려동물에 대한 높은 사회적 관심이 높아졌음을 알 수 있다. 또한 반려동물을 양육 의사가 있는 가구와, 실제 양육하는 가구의 수도 꾸준히 증가하여 국가차원에서 '펫 티켓(Pet+Ettiquet)' 즉 반려인과 반려동물 그리고 비 반려인이 함께 공존할 수 있도록 지자체별로 교육을 제공하는 등 반려동물과 함께하는 삶의 질 향상을 추구하고 있다.

이러한 사회적 현상에도 불구하고 여전히 반려인들은 반려동물을 양육 시 필요한 정보를 동물, 유튜브, 여러 웹사이트 검색 등을 통해 정보를 얻기 위한 노력이 필요하다.

따라서 대표적인 반려동물인 반려견을 기준으로 이미지 분석을 통해 품종 별 특성 정보 제공, 블로그를 이용한 양육과정 기록, 위치 기반 서비스를 통한 동물병원, 약국 정보를 제공한다. 이에 따라 반려견을 좀 더 이해하고 위치기반 서비스를 이용해 주변 동물병원과 약국을 쉽게 검색할 수 있도록 한다.

웹사이트 이용 시 하나의 웹사이트에서 반려동물에 관한 지식, 가게 등의 정보를 쉽게 얻어 반려인에게 정보검색 시 편의를 제공한다. 또한 블로그를 이용해 반려동물과의 일상생활을 기록하고, 태그기능을 위해 반려동물의 취향, 건강상태 등을 정리해 둘 수 있다.

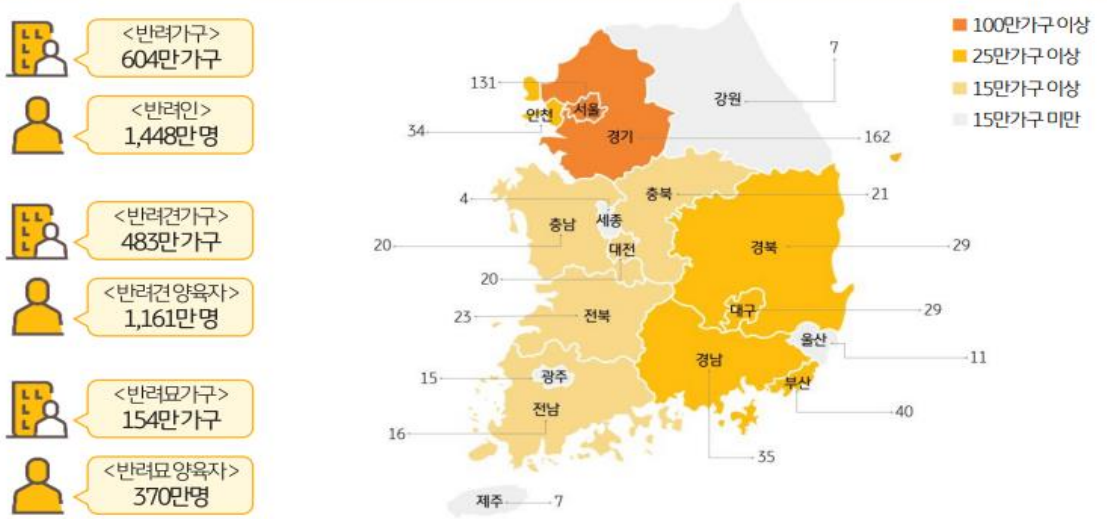
이후 강아지 품종 별 성격에 따른 훈련 법, 대표적인 질병 등의 정보 제공, 블로그의 태그 기능 및 게시글 데이터를 수집해 태그를 이용한 공통 관심사의 블로그 글 열람하고, 리뷰 서비스를 도입하여 병원 후기 게시글을 활용하여 병원 평점 나아가 산책, 미용, 식당 등 반려동물과 함께 생활할 수 있는 장소에 대한 추천 서비스를 제공할 예정이다.

이를 통해 반려인들은 반려동물의 특성, 취향 등을 인지하여 반려생활 기간동안 서로의 유대감을 높여 함께하는 삶의 질을 높일 수 있도록 돕는다.

2. 반려 동물 양육 현황 및 문제인식

2-1. 반려동물 양육 현황

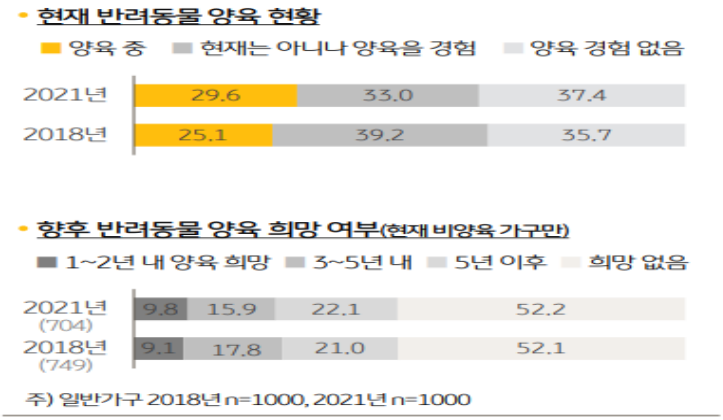
그림 I-1 | 한국의 지역별 반려가구 현황 (단위: 만가구)



※통계청 <2019 인구주택총조사>, 농림축산식품부 동물등록정보 데이터를 가공한 요약정보(2019년 말 기준), 전국 20세 이상 남녀 1,000명을 대상으로 한 설문조사(2020년 말 기준) 결과를 기초 자료로 활용해 추정. 데이터 속성과 조사 방법 관련 세부 내용은 참고자료를 참조

2020년 기준 한국에서 반려동물을 기르는 '반려가구'는 604만 가구로 전체 가구의 29.7%를 차지하고, 반려 인은 1,448만 명으로 '반려인 1500만시대'를 눈 앞에 두고 있다.

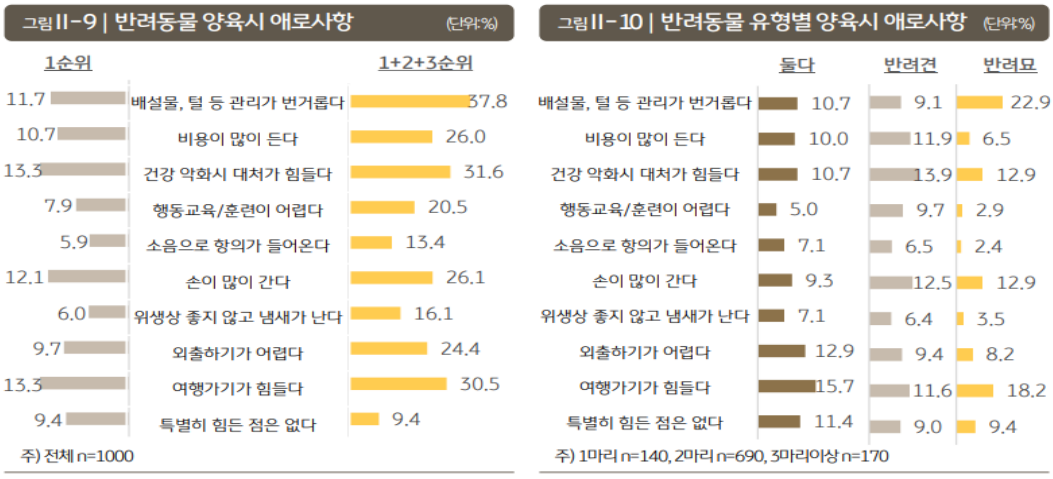
그림 I-3 | 반려동물 양육 현황 및 향후 양육의향 (단위:%)



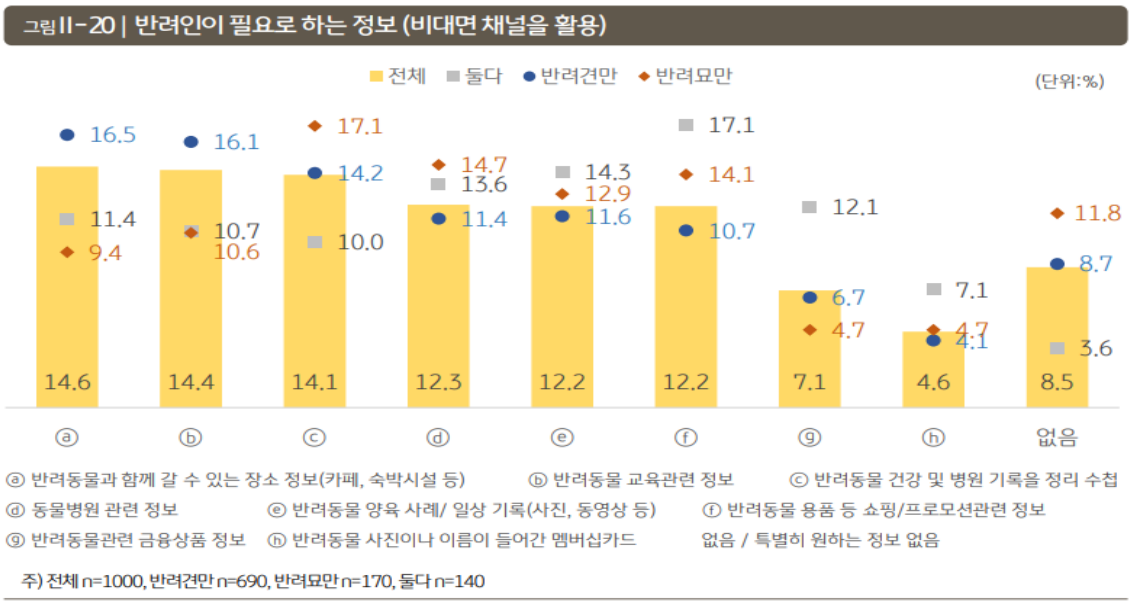
주) 일반가구 2018년 n=1000, 2021년 n=1000

현재 반려동물을 기르지 않는 가구 중 '향후 개나 고양이를 키워보고 싶다'고 응답한 비율은 47.8%로, 2018년 47.9%과 유사한 수준을 보였다. 양육 시기에 대해서는 '향후 1~2년 내 양육 희망' 9.8%, '향후 3~5년 내 양육 희망' 15.9%, '향후 5년 이후 양육 희망' 22.1%로 나타나 반려인이 점차 증가할 것으로 예상된다.

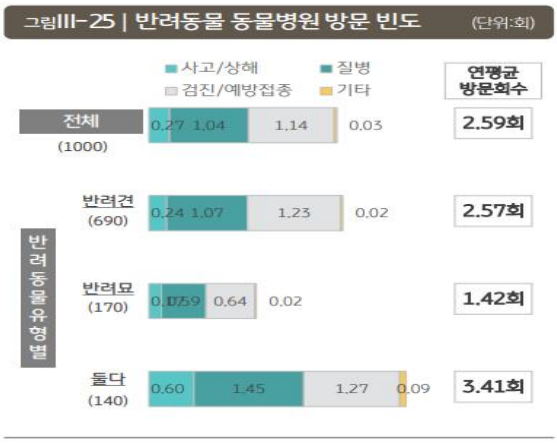
2-2. 반려동물 양육 가구의 문제 인식

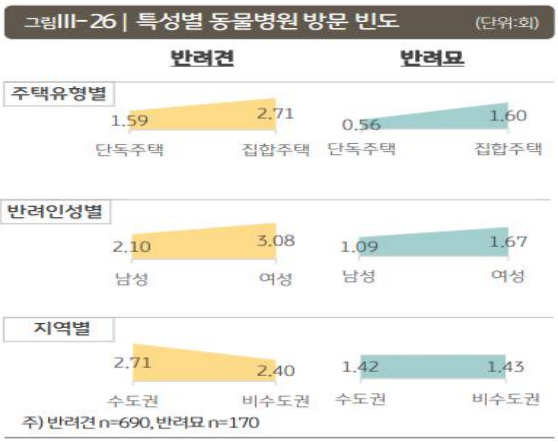


반려가구가 반려동물을 기르면서 느끼는 불편 사항 중 '반려동물이 아플 때 대처가 힘들다'가 2위를 차지했다. 이를 선택한 경우 '반려동물이 아플 때 대처가 힘들다'(13.3%), '가 1위를 차지했다. 그 중 반려견을 기르는 가구는 '반려견이 아플 때 대처가 힘들다' (13.9%), '로 1위를 차지했다.

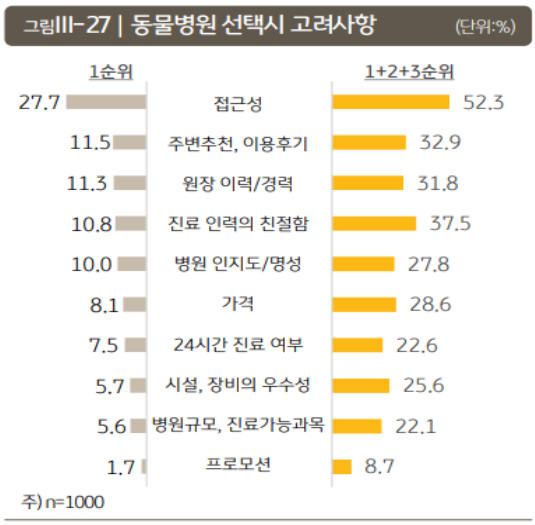


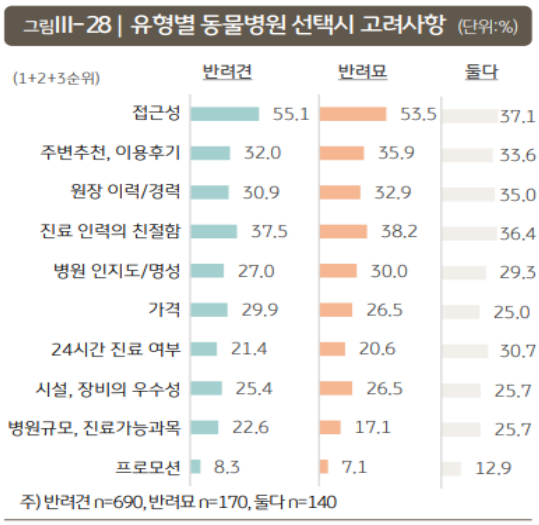
반려인이 업체 홈페이지나 앱과 같은 비 대면 채널을 통해 얻고 싶은 정보 세가지는 '카페나 숙박 시설 등 반려동물과 함께 갈 수 있는 장소 정보', '반려동물 교육 정보', '반려동물 건강 관리와 병원 기록을 정리하는 수첩'으로 나타났다. 반려견을 기르는 반려인의 경우 '반려동물과 외출할 수 있는 장소'와 '반려동물 교육 정보'를 원하는 반면, 반려묘를 기르는 반려인의 경우 '반려동물 건강 관리와 병원 기록을 정리하는 수첩'을 필요로 했다.





반려동물이 동물병원을 방문한 횟수는 전체적으로 연간 2.59회였다. 동물병원을 방문한 이유를 보면 '검진이나 예방 접종'을 위해 동물병원을 방문한 경우가 연 1.14회로 가장 많았고, '질병'에 의한 경우도 연 1.04회로 비교 적 높은 정도였다.

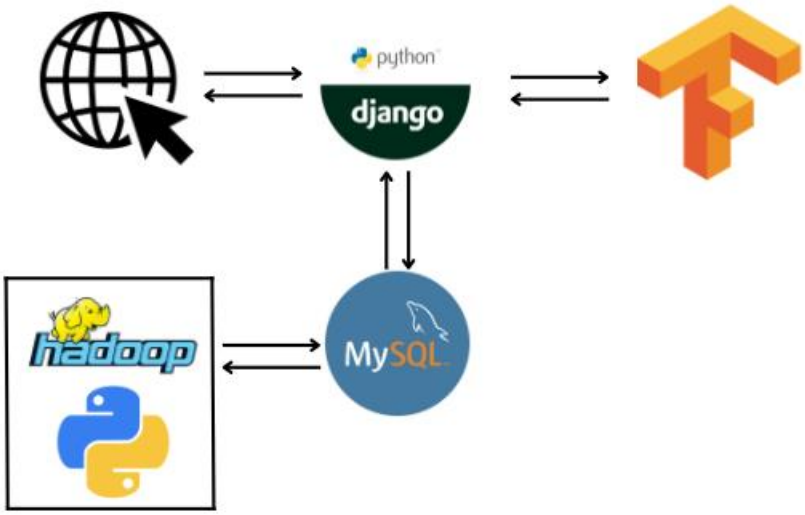




동물병원을 선택할 때 가장 중요하게 여기는 요소는 '접근성'이었다. 동물병원을 선택할 때 중요하게 고려하는 사항(1+2+3순위)은 '접근성'으로 52.3%가 가장 중요한 요소로 꼽았다.

Ⅱ. 프로젝트 추진 과정

1. 시스템 구성



- 운영체제 → Windows 10 / Windows 11 / Ubuntu : 18.04
- 데이터 분석 → Python : 3.9.7
- 데이터 분석, 전 처리 → Numpy : 1.20.3 / Pandas : 1.3.4 / Hadoop : 2.10.1
- 머신 러닝 → Tensorflow : 2.8.0
- 웹 개발 프레임워크 → Django : 4.0.4
- 데이터베이스 → MySQL : 8.0.28

2. 데이터 설명

- 여러 품종의 강아지 이미지를 학습시켜, 강아지 품종을 궁금해하는 사용자들에게 해당 품종이 무엇인지 알려준다. 품종을 이미 알고 있는 경우 품종 결과에 따라 제공되는 특성 정보를 제공한다.

• 사용자 현위치를 기준으로 주변 동물병원과 동물의약품을 판매하는 약국 정보를 제공하기 위해 업체명, 위치 좌표를 사용한다.

2-1. 데이터 전 처리

2-1-1. 강아지 이미지 파일 전 처리

2-1-1-1. 파일 분류

• 파일 분류 진행과정

사용하려고 했던 Dataset은 Kaggle에 있는 Dog Breed를 주제로한 데이터 셋이었으며 캐글 사이트에서 제공한 용량의 정보는 약 20gb였다.

다운을 받고 압축을 풀어보니 하나의 폴더에 대략 12만장의 Dog Breed File이 있었고, 용량은 총 16gb였다.

같이 동봉되어 있는 csv 파일을 통해 확인 결과 아래 사진과 같이 이미지 파일에 따른 견 종 정보가 Labeling 되어있었다.

DogId	Breed	Image	ImageNumber
6533	AM STAFF CROSS	65db56cd-18fd-425e-a329-61587d753e81.jpg	1
6533	AM STAFF CROSS	745544ba-e959-4307-b556-b68f0b7fb292.jpg	2
6533	AM STAFF CROSS	bfa602dc-076b-41fc-9894-65565d571ee6.jpg	3
6533	AM STAFF CROSS	918a84e7-24c9-4081-a112-d36ca7105998.jpg	4
6533	AM STAFF CROSS	0e84b274-c943-492f-ac67-4ce23df8b33a.jpg	5
6533	AM STAFF CROSS	9e8ad757-97b0-4a76-b13c-787206c41255.jpg	6
6533	AM STAFF CROSS	5a161a31-ad4c-409e-91f9-4391d536ed69.jpg	7
6533	AM STAFF CROSS	1d7df69c-36da-4033-a4c8-a04eaa113f5b.jpg	8
6562	KELPIE CROSS	98668ef5-0c34-4ffd-8376-cdd461c2ee6d.jpg	1
6562	KELPIE CROSS	f52921cf-8c0e-4a21-8f5f-b89d5813abb3.jpg	2
6562	KELPIE CROSS	1f823388-ec83-414b-94d1-3a4971001798.jpg	3
6562	KELPIE CROSS	cd695a53-224a-42f2-9b4f-67554a5e848a.jpg	4
6562	KELPIE CROSS	7729efb1-8227-4254-96c8-5d758899e217.jpg	5

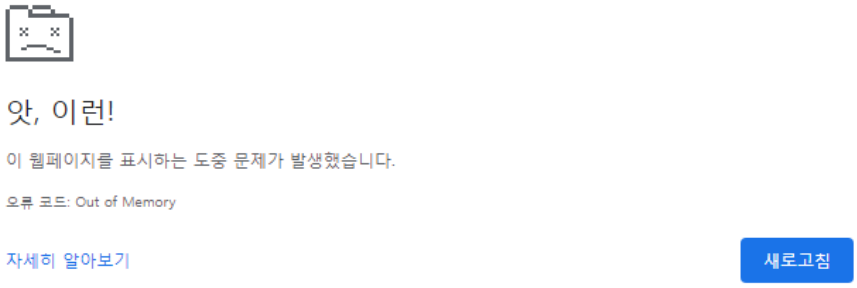
```
data_dir = "dataset/dog_breed_photos/dog_breed_photos/"
images = []
labels = []
```

```
type(df)
```

```
pandas.core.frame.DataFrame
```

```
for file in df['Image']:
    image = np.array(Image.open(data_dir+file))
    images.append(image)
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13720/332068703.py in <module>
      1 for file in df['Image']:
----> 2     image = np.array(Image.open(data_dir+file))
      3     images.append(image)

MemoryError: Unable to allocate 1.83 MiB for an array with shape (800, 800, 3) and data type uint8
```



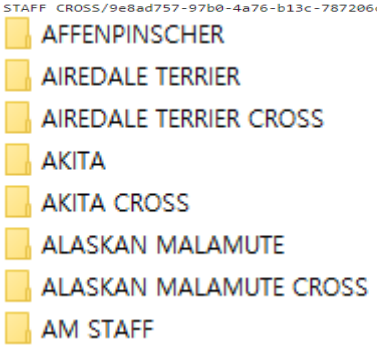
해당 코드로 이미지를 한 장씩 불러 images라는 리스트에 담으려고 했으나, 16gb라는 큰 사이즈로 Memory Error가 발생하였다.
그래서 csv file에 파일명이 이미 label이 매칭이 되어있었으므로, 각 파일에 해당하는 label디렉토리를 만들어 이미지파일을 분류하기로 했다.
이때 shutil, os 라이브러리를 사용하였으며, shutil은 파일과 파일모음에 대한 여러가지 연산을 제공하는데 특히 파일 복사와 삭제를 지원하는 함수가 제공이 된다.
os 라이브러리는 운영체제에서 제공되는 여러 기능을 python에서 수행 할 수 있게 해주는 라이브러리이다.
os.mkdir 을 통하여 디렉토리를 생성하기로 했다.
그리고 shutil.move 를 통하여 os.mkdir 로 생성된 디렉토리에 path 경로에 있는 파일들을 새롭게 지정한 path1이라는 새로운 경로로 옮기기로 했다.

```
# path : 현재 사진들이 있는 경로
# path1 : 새로 사진들을 옮길 경로
path = 'C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/'
path1= 'C:/data/pythonworkspace/Final/spdataset/'
```

```
# path1의 경로에 label 명으로 각 디렉토리를 생성한다
img_type = '.jpg'

for a in range(len(df['Breed'])):
    try:
        os.mkdir(path1 + df.Breed[a+1])
    except:
        pass
```

```
# Dataframe의 파일 이미지 이름으로 그에 해당하는 생성된 label의 디렉토리로 이미지를 옮긴다.
for a in range(len(df['Breed'])):
    filename = df['Image'][a]
    src = path
    dir = path1 + df['Breed'][a]+'/'
    try:
        shutil.move(src + filename, dir + filename)
        print(src + filename+","+dir + filename)
    except:
        pass
print("##copy end##")
```

```
C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/65db56cd-18fd-425e-a329-61587d753e81.jpg,C:/data/p
ythonworkspace/Final/spdataset/AM STAFF CROSS/65db56cd-18fd-425e-a329-61587d753e81.jpg
C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/745544ba-e959-4307-b556-b68f0b7fb292.jpg,C:/data/p
ythonworkspace/Final/spdataset/AM STAFF CROSS/745544ba-e959-4307-b556-b68f0b7fb292.jpg
C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/bfa602dc-076b-41fc-9894-65565d571ee6.jpg,C:/data/p
ythonworkspace/Final/spdataset/AM STAFF CROSS/bfa602dc-076b-41fc-9894-65565d571ee6.jpg
C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/918a84e7-24c9-4081-a112-d36ca7105998.jpg,C:/data/p
ythonworkspace/Final/spdataset/AM STAFF CROSS/918a84e7-24c9-4081-a112-d36ca7105998.jpg
C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/0e84b274-c943-492f-ac67-4ce23df8b33a.jpg,C:/data/p
ythonworkspace/Final/spdataset/AM STAFF CROSS/0e84b274-c943-492f-ac67-4ce23df8b33a.jpg
C:/data/pythonworkspace/Final/dataset/dog_breed_photos/dog_breed_photos/9e8ad757-97b0-4a76-b13c-787206c41255.jpg,C:/data/p
ythonworkspace/Final/spdataset/AM STAFF CROSS/9e8ad757-97b0-4a76-b13c-787206c41255.jpg
```



<그림5. 새롭게 생성된 디렉토리들>

각 Label 디렉토리가 생성이 되었으며 이미지 파일들이 매칭된 label 디렉토리로 모두 이동을 하였다.
딥 러닝을 하기위한 Train, Test, Valid 로 이미지 파일들을 split을 해야 하기에 splitfolders라는 라이브러리를 사용하였다.
해당 라이브러리는 input, output 디렉토리 설정과, 랜덤 시드 값, train/test/valid 비율 값을 정해주면 train/test/valid 디렉토리를 생성하여 설정한 비율에 맞게 이미지 파일들을 분배한다.

```
splitfolders.ratio('C:/data/pythonworkspace/Final/spdataset/',
                   output='C:/data/pythonworkspace/Final/spdatasetdl/', seed=55, ratio=(0.6,0.2,0.2))
```

```
Copying files: 117254 files [08:32, 228.59 files/s]
```

딥 러닝에 사용할 train, test, valid set를 사용할 수 있게 되었다.

```
train_dir = 'spdatasetdl/train/'
test_dir = 'spdatasetdl/test/'
valid_dir = 'spdatasetdl/val/'
```

```
BATCH_SIZE = 100
image_height = 224
image_width = 224
```

```
train = ImageDataGenerator(rescale=1./255, rotation_range=10, width_shift_range=0.1,
                           height_shift_range=0.1, shear_range=0.1, zoom_range=0.1)
train_generator = train.flow_from_directory(train_dir,
                                            target_size=(image_height,image_width),
                                            color_mode='rgb',
                                            batch_size=BATCH_SIZE,
                                            seed=1, shuffle=True,
                                            class_mode="categorical")
valid = ImageDataGenerator(rescale=1.0/255.0)
valid_generator = valid.flow_from_directory(valid_dir,
                                            target_size=(image_height,image_width),
                                            color_mode='rgb',
                                            batch_size=BATCH_SIZE,
                                            seed=7, shuffle=True,
                                            class_mode="categorical")

test = ImageDataGenerator(rescale=1.0/255.0)
test_generator = test.flow_from_directory(test_dir,
                                            target_size=(image_height,image_width),
                                            color_mode='rgb',
                                            batch_size=BATCH_SIZE,
                                            seed=7, shuffle=True,
                                            class_mode="categorical")
```

```
Found 70251 images belonging to 276 classes.
Found 23342 images belonging to 276 classes.
Found 23661 images belonging to 276 classes.
```

2-1-1-2. 이미지 전 처리 및 증가

```
BATCH_SIZE = 100
image_height = 224
image_width = 224
train_dir = 'standforddata2/train/'
test_dir = 'standforddata2/test/'
valid_dir = 'standforddata2/val/'
```

```
train = ImageDataGenerator(rescale=1./255, rotation_range=15, width_shift_range=0.2, height_shift_range=0.1,
                           shear_range=0.3, zoom_range=0.3,horizontal_flip=True)

train_generator = train.flow_from_directory(train_dir,
                                            target_size=(image_height,image_width),
                                            color_mode='rgb',
                                            batch_size=BATCH_SIZE,
                                            seed=1, shuffle=True,
                                            class_mode="categorical")
```

사이즈가 큰 이미지는 관련성이 있거나, 또는 없는 많은 정보가 포함되어 있을 수 있다.

입력이 클수록 네트워크에서 처리해야하는 매개변수가 더 많아야 하기 때문에 입력의 크기가 중요했다. 그리고 매개변수가 많을수록 더 좋은 컴퓨팅 성능이 필요하며, CNN의 경우엔 매개변수가 많고 샘플이 충분하지 않을 경우 과적합의 문제가 발생할 수 있고 그렇게 될 경우 더 많은 데이터가 필요 할 수 있기에 일반적으로 사용하는 이미지의 크기인 (224, 224)로 설정을 하고, 사용할 이미지를 전부 (224,224)로 Resizing 하였다.
이후, 1pixel의 rgb값을 가진 dataset으로 변환하였고, 최대값인 255로 나누어 0 ~ 1 값을 가지는 스케일링 작업을 하였다.
그리고 데이터 셋 변경으로 인하여 (아래 3-2에 기술) 전체적인 파일의 개수가 줄어듦에 따라 이미지 전 처리 과정에서 위와 같은 코드를 통해 데이터를 증가시켰다.

ImageDataGenerator 의 클래스의 인수를 활용하였고, 인수는 아래와 같다.

Rotation_range : 이미지 회전범위,
값이 15면 0~15도 범위 내에서 임의로 회전

Width_shift_range : 그림을 수평으로 랜덤하게 평행 이동시키는 범위
값이 0.2면 전체 넓이가 100을기준으로 20픽셀 내외로 이미지를 좌우 이동

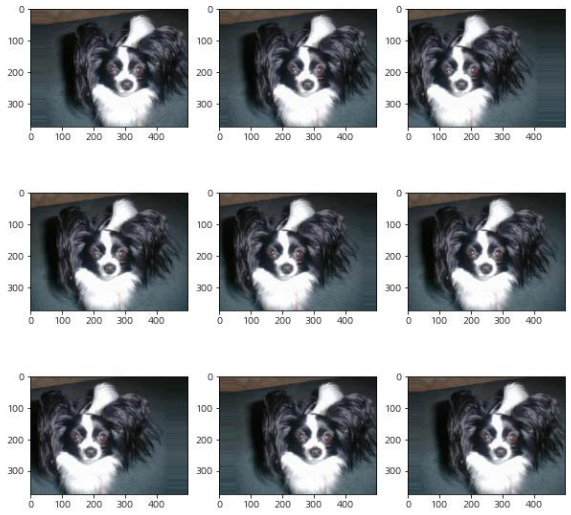
Height_shift_range : 그림을 수직으로 랜덤하게 평행 이동시키는 범위
값이 0.2면 전체 넓이가 100을기준으로 10픽셀 내외로 이미지를 좌우 이동

Shear_range : 원본 이미지를 임의로 변형시키는 범위
값이 0.3이면 0.3라디안 내외로 시계 반대방향으로 이미지 변환

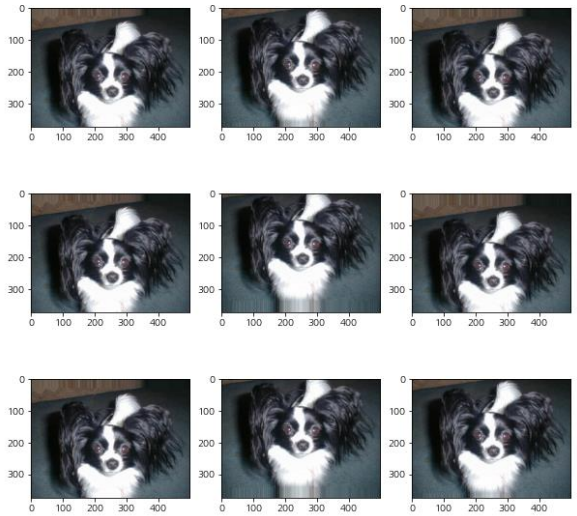
Zoom_range : 임의 확대/축소 범위.
값이 0.3이면 0.7 ~ 1.3 배의 크기로 이미지를 변환

Horizontal_flip : 이미지를 랜덤으로 가로로 뒤집는다.

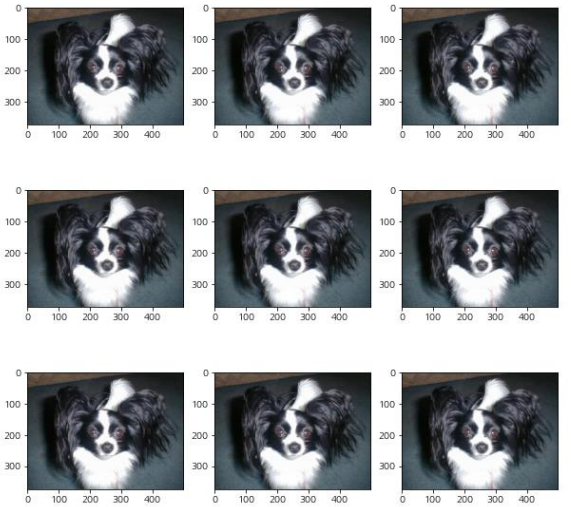
```
data_gen= ImageDataGenerator(width_shift_range=0.2)
data_iter = data_gen.flow(img_data, batch_size=1)
fig = plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    batch = data_iter.next()
    image = batch[0].astype('uint16')
    plt.imshow(image)
plt.show()
```



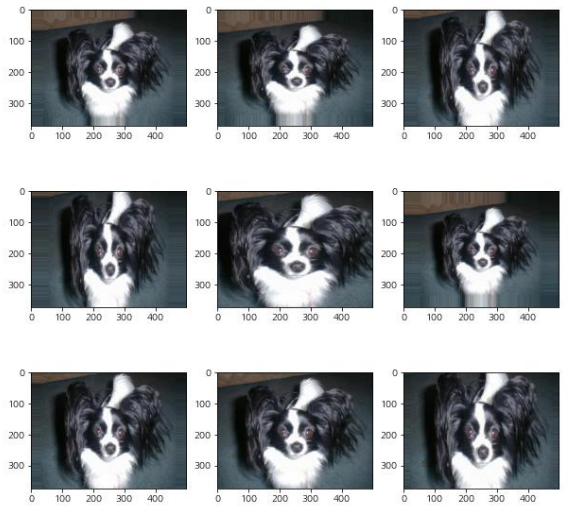
```
data_gen= ImageDataGenerator(height_shift_range=0.1)
data_iter = data_gen.flow(img_data, batch_size=1)
fig = plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    batch = data_iter.next()
    image = batch[0].astype('uint16')
    plt.imshow(image)
plt.show()
```



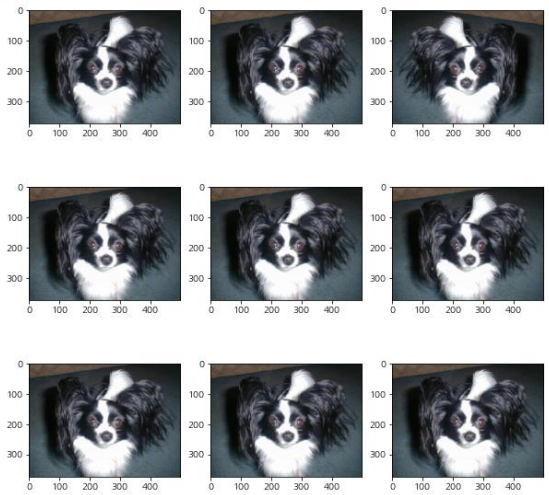
```
data_gen= ImageDataGenerator(shear_range=0.3)
data_iter = data_gen.flow(img_data, batch_size=1)
fig = plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    batch = data_iter.next()
    image = batch[0].astype('uint16')
    plt.imshow(image)
plt.show()
```



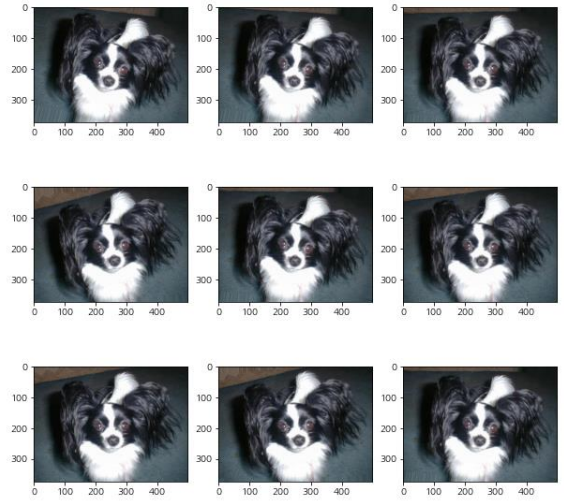
```
data_gen= ImageDataGenerator(zoom_range=0.3)
data_iter = data_gen.flow(img_data, batch_size=1)
fig = plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    batch = data_iter.next()
    image = batch[0].astype('uint16')
    plt.imshow(image)
plt.show()
```



```
data_gen= ImageDataGenerator(horizontal_flip=True)
data_iter = data_gen.flow(img_data, batch_size=1)
fig = plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    batch = data_iter.next()
    image = batch[0].astype('uint16')
    plt.imshow(image)
plt.show()
```



```
data_gen= ImageDataGenerator(rotation_range=15)
data_iter = data_gen.flow(img_data, batch_size=1)
fig = plt.figure(figsize=(10,10))
for i in range(9):
    plt.subplot(3,3,i+1)
    batch = data_iter.next()
    image = batch[0].astype('uint16')
    plt.imshow(image)
plt.show()
```



2-1-2. 동물병원 및 약국데이터

• 로컬데이터 사이트의 동물병원, 약국 인허가 데이터를 CSV파일 형태로 사용했다.

해당 CSV 파일에는 영업상태, 사업장명, 도로명 주소, 좌표정보(위도,경도)외 32개의컬럼과 영업, 폐업, 휴업 정보가 포함되어 있었다.

사용할 4개의 컬럼인 영업상태, 사업장명, 도로명 주소, 좌표정보(위도,경도)을 제외한 나머지 컬럼을 제거하고, 위,경도 좌표정보는 중부원점 좌표계(GRS80)를 기준으로 한 좌표 값이기에 실제 지도나 GPS에서 사용하는 좌표정보(WGS84)로 변환하기위해 Python의 Pyproj 라이브러리를 활용하여 변환을 하였다.

```
def project_array(coord, p1_type, p2_type):
    """
    좌표계 변환 함수
    - coord: x, y 좌표 정보가 담긴 NumPy Array
    - p1_type: 입력 좌표계 정보 ex) epsg:5179
    - p2_type: 출력 좌표계 정보 ex) epsg:4326
    """
    p1 = pyproj.Proj(init=p1_type)
    p2 = pyproj.Proj(init=p2_type)
    fx, fy = pyproj.transform(p1, p2, coord[:, 2], coord[:, 3])
    return np.dstack([fx, fy])[0]
```

```
# 좌표계 정보 설정
p1_type = "epsg:2097"
p2_type = "epsg:4326"
```

```
result = project_array(coord, p1_type, p2_type)
result
```

• • •

```
df['mapx']=result[:,0]
df['mapy']=result[:,1]
```

	번호	사업장명	좌표정보(x)	좌표정보(y)	mapx	mapy
0	1	일곡동물병원	190562.9092	189333.0171	126.894285	35.203171
1	2	우리동물병원	191120.4200	184715.8661	126.900458	35.161559
2	3	참좋은 동물병원	188403.3482	189881.7854	126.870562	35.208094
3	4	김앤강 동물병원	192235.3200	187547.2500	126.912668	35.187089
4	5	빛고을 동물의료센터	192695.3842	186661.0310	126.917727	35.179105
...	...	...	...	...	...	...
4831	4832	연산동물의료센터	389916.4819	188188.0447	129.082605	35.174963
4832	4833	원 동물병원	238737.9236	352447.6661	127.431251	36.672479
4833	4834	우리동물병원	188528.4554	228849.9720	126.871386	35.559323
4834	4835	소담동물병원	418279.6423	361268.6617	129.441248	36.727679
4835	4836	도그스타 동물병원	196493.9366	418242.4068	126.958374	37.266121

변환한 좌표 값을 가지고 테스트를 하기위해 구글지도에서 해당 좌표를 입력하여 실제로 변환이 잘되었는 확인 작업을 했다

실제 마커를 이용하여 표시한결과 약 260m의 차이를 보였으며, 판단결과 변환한 값으로 정보를 제공하기에는 사용자의 혼란을 야기할 수 있기에, 다른 방법으로 전처리가 필요했다.

검색해본 결과 Geocoder라는 프로그램이 있었으며, 해당 프로그램은 하루에 10000건의 Data를 무료로 변환할 수 있었으며, 도로명주소를 기반으로 좌표 값을 변환해 주기에, 해당 파일로 좌표 값을 추출했다.

그림 2와 그림3을 비교해보면 새롭게 변환한 주소의 값과 실제 위치한 좌표와 거의 차이가 없음을 확인하였고, 해당 프로그램을 이용하여 좌표 값의 전처리를 하였다.

3. 인공지능 모델

딥 러닝에 대해서 배울 시간이 적었으며, 각 Layer를 어떻게 활용할지에 대한 정보가 너무 적었다. 또한 합성곱 신경망 기반의 딥러닝 모델을 제대로 훈련시키기 위해선 많은 양의 데이터가 필요하지만 구하기도 쉽지 않기에, 전이학습을 사용할 수밖에 없었다.

전이학습을 사용하게 되면 비교적 적은 수의 데이터로도 원하는 값을 얻을 수 있기 때문이다

전이 학습 중에서도 마지막 완전연결층만 학습하는 특성 추출기법을 사용했다.

또한 딥 러닝의 가중치는 Imagenet에서 사전 훈련된 가중치인 'imagenet'을 사용했다.

3-1. 인공지능 모델 후보 선정

• Resnet : 신경망은 깊이가 깊을수록 성능이 좋아지다가 일정한 단계에 이르면 오히려 성능이 나빠지는데 이것을 보완한 모델이다. 기본적으로 VGGNet의 VGG구조를 뼈대로 하고 있다.

• VGGNet (VGG19) : VGGNet은 합성곱 층의 파라미터 개수를 줄이고 훈련 시간을 개선하려고 탄생 한 모델이며, 네트워크 계층의 총 개수에 따라 여러 유형이 있다. (VGG16, VGG19)

• Xception: 인셉션 모듈은 특징을 효율적으로 추출하기 위해 1x1, 3x3, 5x5의 합성곱 연산을 각각 수행한다. 그리고 인셉션 모듈은 과적합이나, 기울기 소멸문제를 비롯한 학습 시간 지연과 연산 속도 등의 문제를 해결할 수 있다. Xception은 이러한 인셉션 모듈을 단순화했다

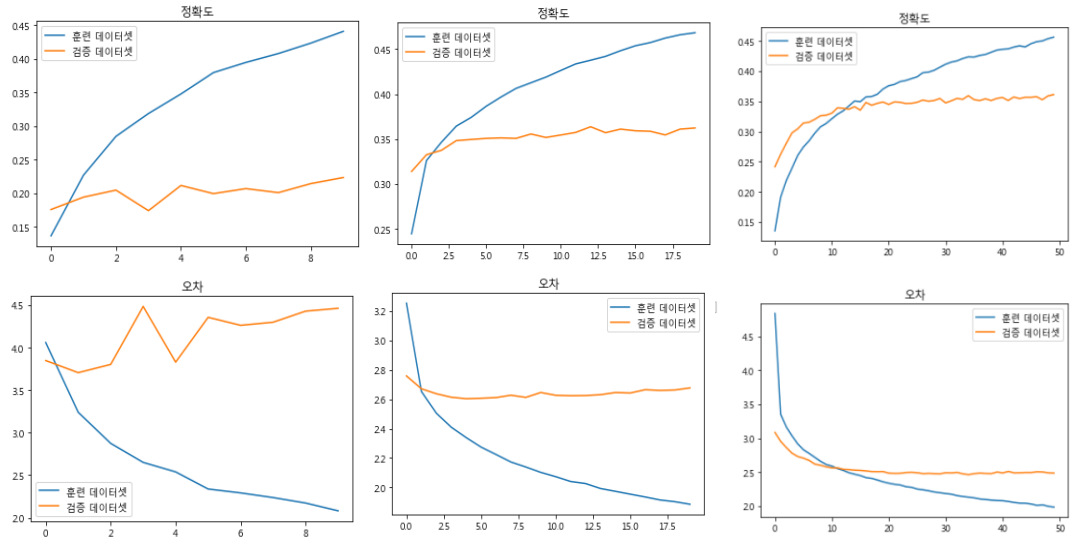
3-2 인공지능 모델 별 학습 결과 및 최종 모델 선정

• 각 후보 별 처리 과정

데이터 셋의 변경.

처음에 사용했던 Dataset은 약 10만개의 사진이었으며 용량은 약 16기가 바이트정도를 차지했었다.

학습시간은 1에폭당 10분이 넘을 정도로 학습 완료시까지 상당한 시간을 필요로 했으나 정확도는 0.5를 넘지 못 했다.

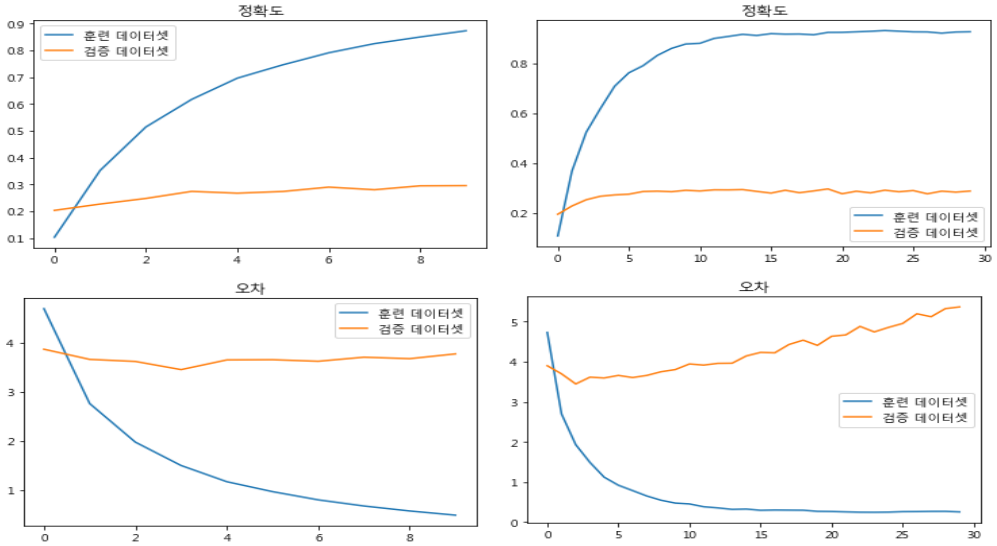


이미지를 좀 더 확인해본 결과 모델링에 혼란을 줄 수 있는 사진들 또한 발견이 되었다.


따라서 학습시간, 모델의 정확도, 오차를 고려하였을 때 현재의 데이터셋을 사용하는 것은 무리라고 판단이 되어 데이터셋을 변경하였다.

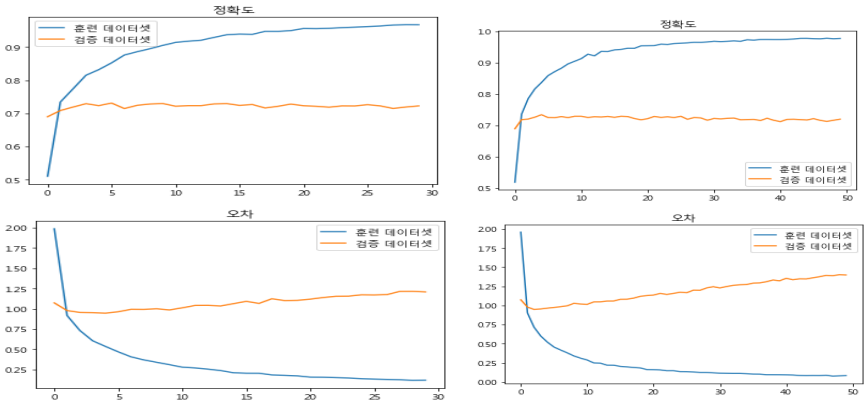
모델 별 훈련/검증 결과 그래프

- **VGG16** : 학습 시 훈련 데이터셋에 대한 정확도는 꾸준히 올라간 반면, 검증 데이터셋에 대한 정확도는 향상이 저조하였다



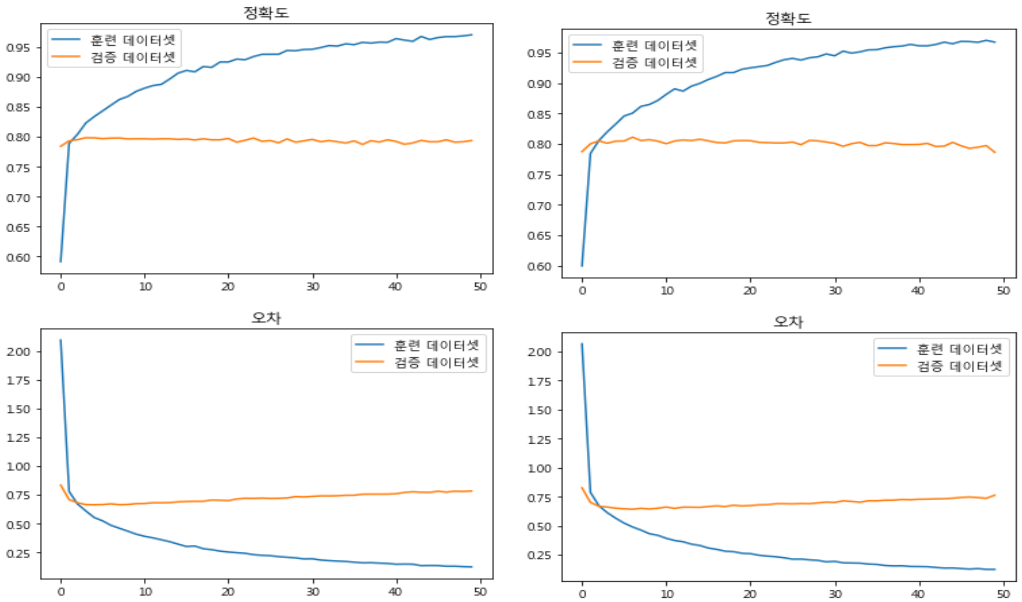
<그림1 VGG16 학습결과 그래프>

- **Resnet** : Xception과 훈련/검증 데이터셋의 정확도와 오차의 흐름이 비슷하나 검증 데이터의 정확도가 약 8% 차이가 났다. 1에폭당 Xception과의 차이는 130초,120초로 Resnet을 사용한 것이 10초 더 걸렸다. 학습시킨 모델로 Test Data Set을 Evaluate한 결과 정답률은 약 0.72를 보였다.



훈련 데이터셋의 정확도는 Xception, VGG16과는 크게 차이가 없었다. 다만 검증, 테스트 데이터셋의 정확도에서 VGG16과는 꽤나 많은 차이를 보였으며, Xception하고는 대략 0.07~0.08정도의 값의 차이를 보였다. 정확도차이는 크게 없었으나 오차인 loss에서 값의 차이가 대략 0.4 ~0.5의 차이가 발생했다.

- **Xception** : 훈련 데이터셋의 정확도는 0.95를 상회 하였으며, 검증 데이터셋의 정확도 또한 0.8정도로 꽤 안정적인 수치를 보여주고 있다.
  테스트 데이터셋으로 evaluate 했을 때도 대략 0.78~0.8 사이의 값을 보이며 오차 또한 0.8로 검증 데이터셋과 비슷한 흐름으로 가는 것을 확인했다. 따라서 **최종 모델**로 선정했다.



최종모델(Xception)

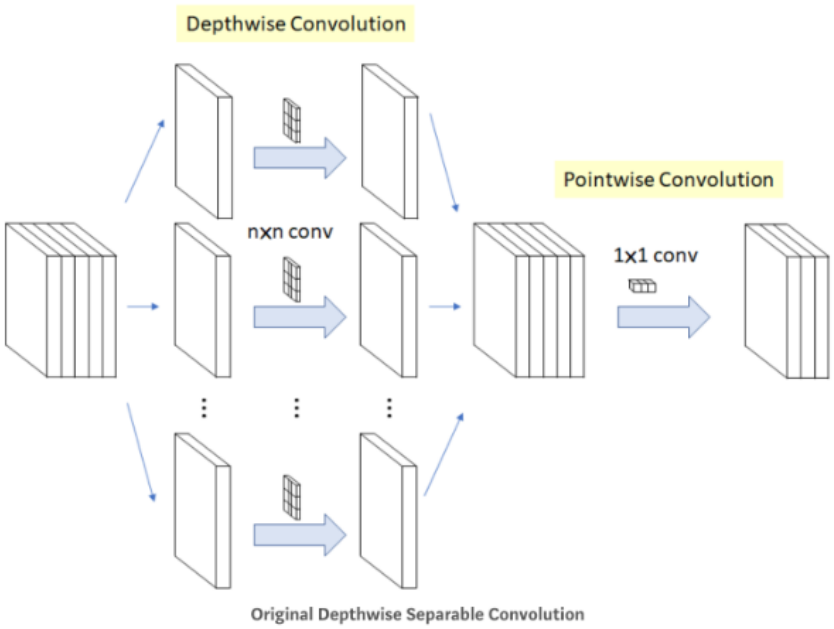
- 목적 : 연산 량과 parameter의 개수를 줄여서, 큰 이미지 인식을 고속화 한다.
- 장점 : VGG처럼 네트워크의 구조가 간단해서 활용도가 높다

실제 사용한 Layer

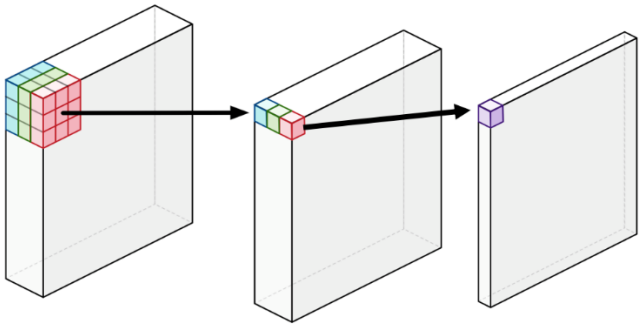
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 7, 7, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 121)               247929

=================================================================
Total params: 21,109,409
Trainable params: 247,929
Non-trainable params: 20,861,480
_________________________________________________________________
```

깊이별 분리 합성곱 (Depthwise Separable Convolution)

- 채널(깊이)별로 나누어 합성곱을 한다. 일반 합성곱과 결과는 같지만 아래와 같이 2단계로 진행한다.



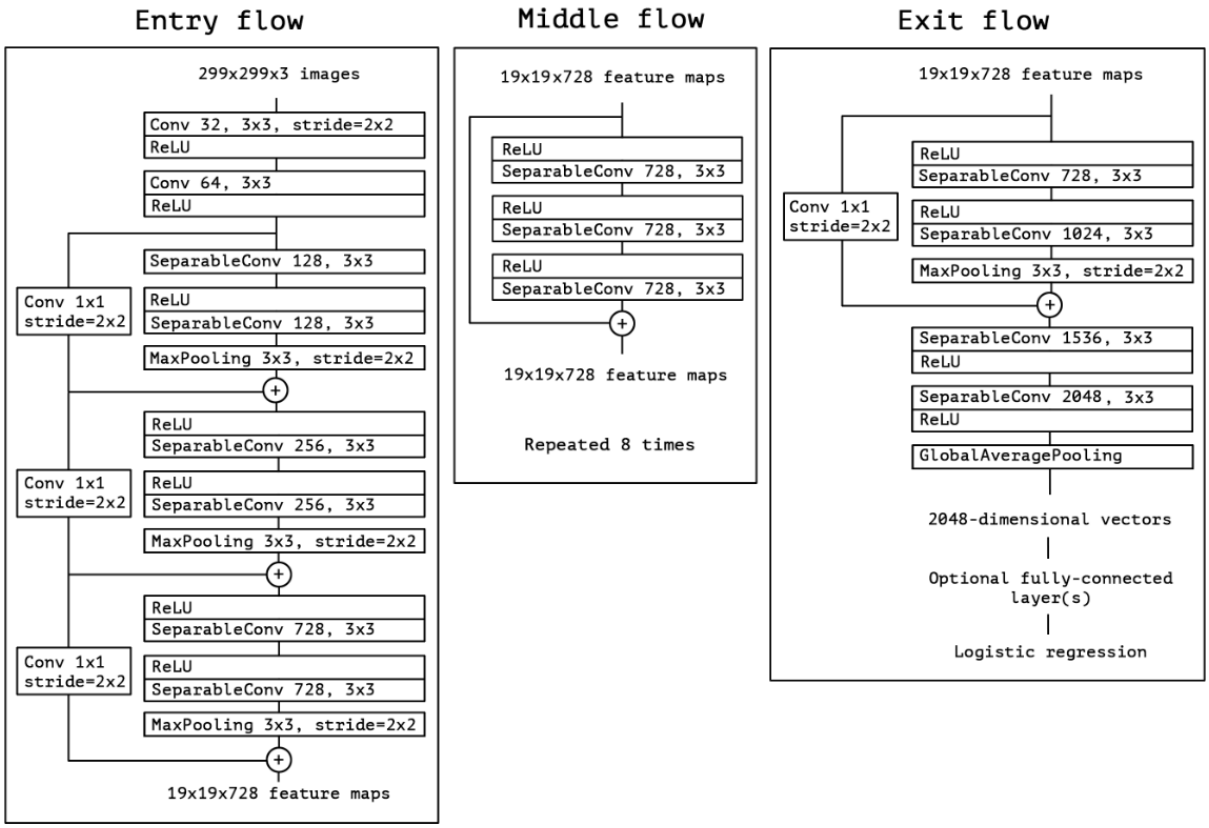
- 1단계 : Channel-wise n x n spatial convolution
- 인풋으로 5개의 채널이 들어오면 5개의 n x n conv를 따로 진행하여 합친다.
- 2단계 : Pointwise Convolution
- 채널의 개수를 줄이기 위한 방법으로 사용된다



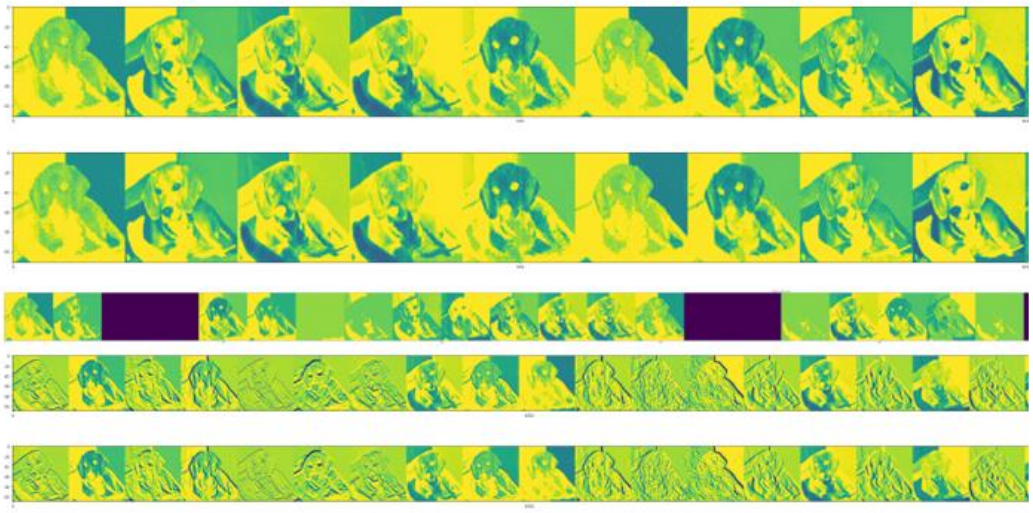
-

- Xception으로 하나의 이미지 file을 넣게 되면 Xception의 Layer에서 각 Feature들을 Extraction 하기위해 아래 사진과 같이 각 layer마다 이미지 처리를 하기 시작한다.

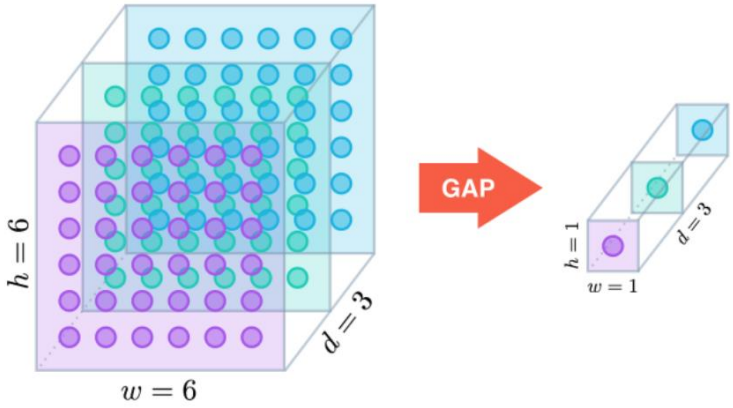
구조



- **[Entry Flow]**
  - 인풋: 229 x 229 x 3
  - > 모든 convolutional layer 다음에는 batch normalization(배치정규화)을 사용한다
  - > 2번 normal convolution (3x3) -> 필터의 갯수: 32 -> 64
  - > Residual Network 가 합쳐진 Inception Module 3번
- **[Middle Flow]**
  - > 반복되는 단순한 모델: 필터의 개수와 width/height 는 바뀌지 않음
  - > ReLU -> Separable Conv -> Separable Conv 8번 반복
- **[Exit Flow]**
  - > Filter의 개수를 늘린다음 -> Maxpooling -> 2번 separable convolution -> Global Average Pooling -> Optional Fully-Connected -> Logistic Regression



- GAPpooling2D 직전에 총 2048개의 사진값이 나오게 되며, GAPpooling2D 레이어를 지나면서 해당 Feature Map 상의 Node값들의 평균을 뽑아낸다.



4. 웹사이트 구현

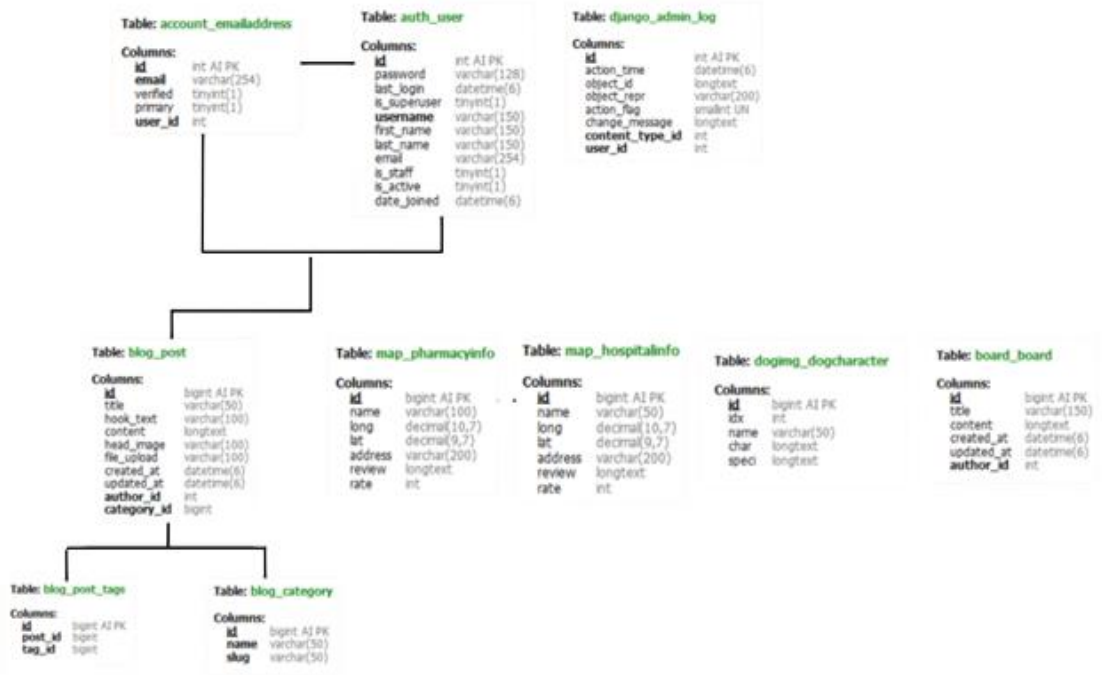
Django과 AWS 클라우드 컴퓨팅을 이용하여 웹 사이트 서버를 구축했고
데이터베이스는 MY SQL을 사용했다.

4-1. 각 프레임워크 사용 이유

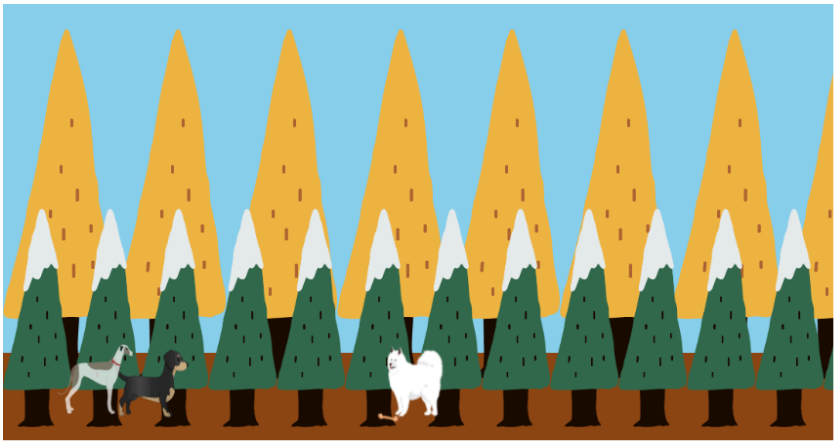
- **Django 사용 이유:**
  1. Python 언어 기반으로 Python이 가진 모든 기능과 라이브러리들을 사용할 수 있다
  2. Django가 제공하는 라이브러리 (로그인 + 회원가입, 인증, CORS, data parsing 등)를 이용해 빠르게 웹 페이지를 만들 수 있다.
  3. Django로 웹사이트 구축 시 데이터베이스를 특별한 코딩 없이 연동시켜 주기 때문에 태그, 게시글, 품종 정보 등 다양한 데이터베이스를 사용해야 하여 작업량 감소와 웹사이트 개발 시간 단축에 도움을 준다.

- **AWS 클라우드 컴퓨팅 사용 이유:**
  1. 클라우드 기술을 이용해 서버를 관리하여, 물리적 하드웨어의 제약없이 사용할 수 있다.
  2. 비용이 저렴하다.

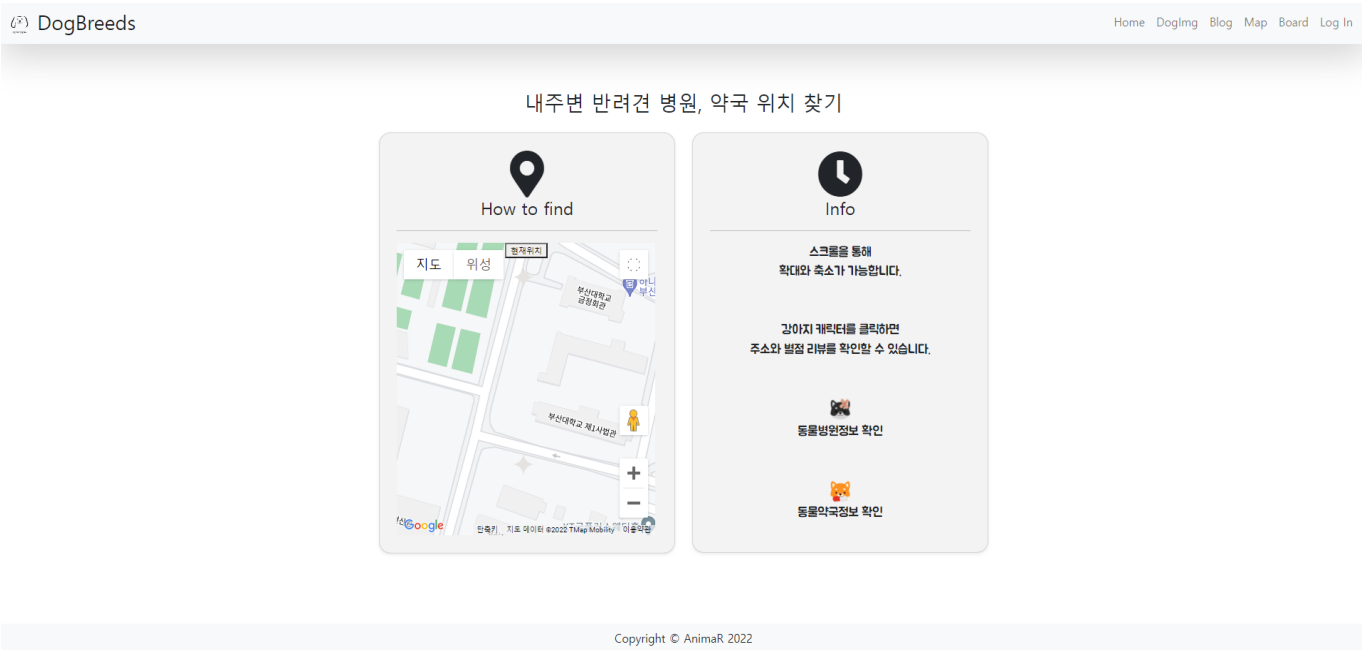
4-2. 데이터베이스 테이블 명세서



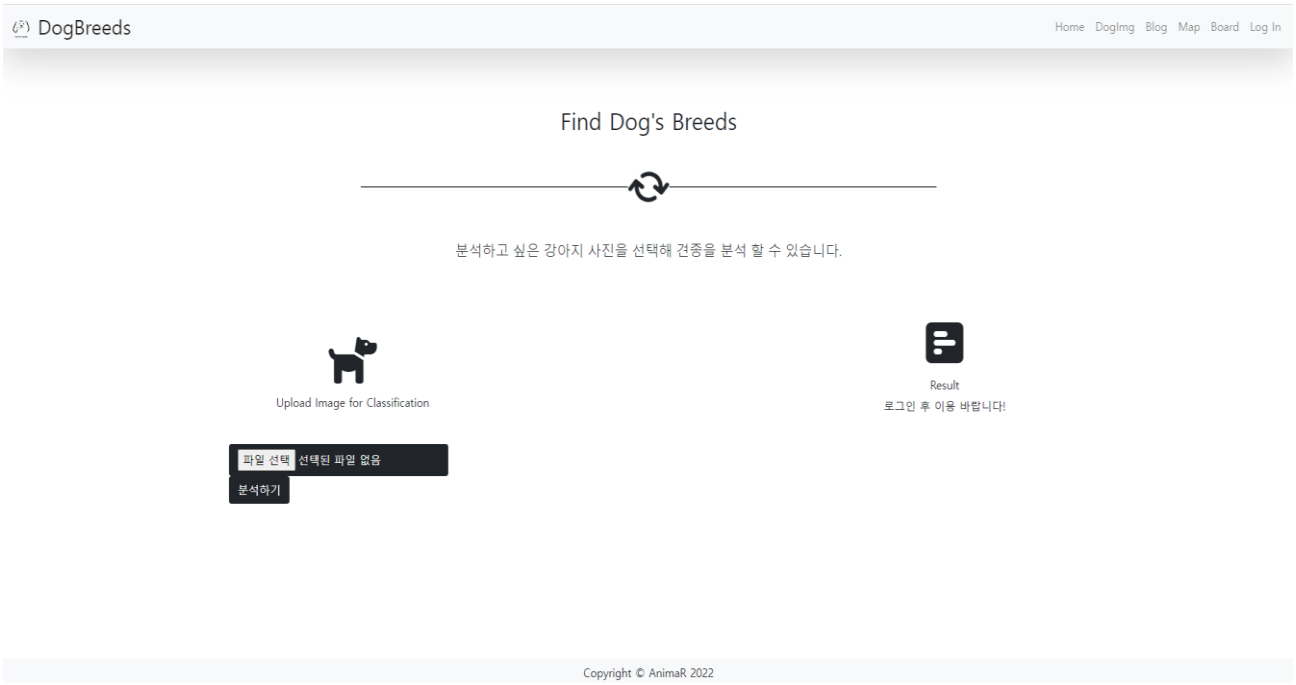
4-3. 웹사이트 구성 시 고려사항



└> 딱딱한 웹사이트가 아닌 친근한 웹사이트의 이미지로 보여지도록 움직임을 이용하여 메인구상



└> 처음 사용하는 사용자에게 친절하게 사용법을 제공할 수 있도록 설명서 추가 작성

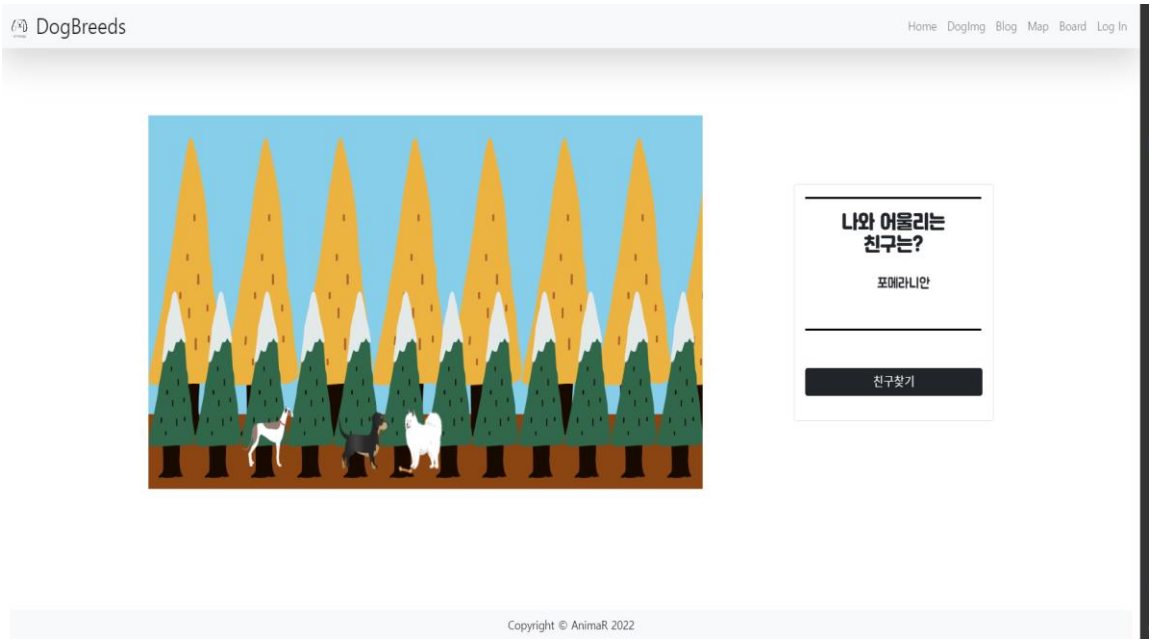


└> 핵심 기술인 견 종 분석 서비스에는 로그인 시 분석 가능하도록 구현

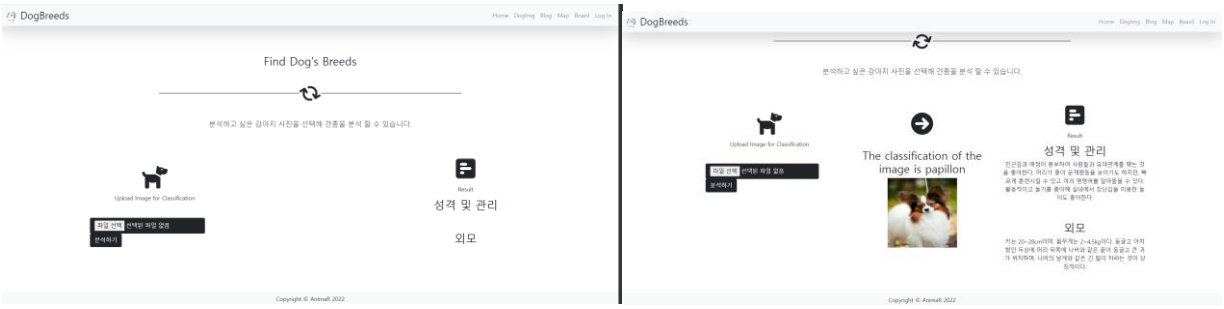
└> 되도록이면 사용자가 지식 서비스를 이용할 수 있도록 구현(로그아웃 상태에서도 게시판을 볼 수 있다.

Ⅳ. 결론

1. 프로젝트 결과

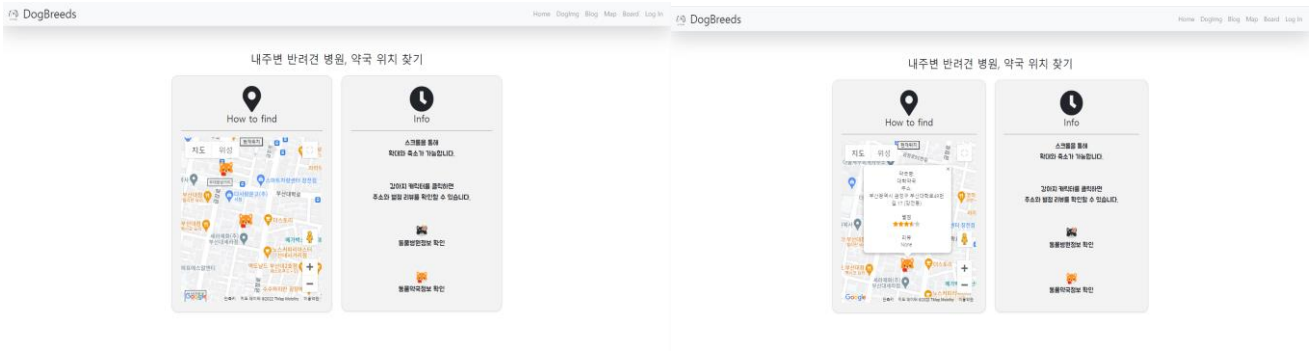


학습된 모델을 사용자가 사용하기 쉽도록 브라우저를 통해 서비스를 제공한다.



강아지 사진을 업로드하면 강아지 사진과 일치하는 견 종을 보여준다.

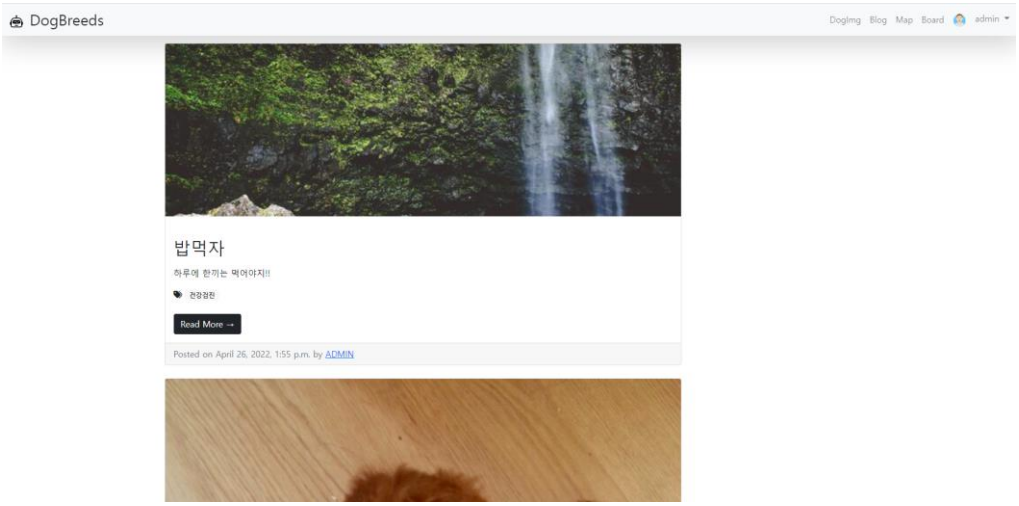
특성을 통해 강아지의 성격, 신체적 특징을 알 수 있다.



<사진: 맵 페이지 현재위치 기반으로 주변 병원, 약국 마커 이미지 사진>

맵 페이지에서 병원 또는 약국을 클릭하면 지도에 사용자의 현 위치 기준으로 가까운 곳의 병원 또는 약국 마커가 나타난다.

마커를 누르면 해당 업체의 정보, 평점 및 리뷰를 알 수 있다.



<사진: 블로그 태그 기능과 게시글 페이지 사진>

블로그에 자유양식으로 글을 쓰고 태그를 남긴다.

2. 향후 계획 및 기대 효과

2-1. 향후 계획

2-1-1. 정보 제공의 확장

- 강아지 품종과 특성을 단순히 제공하는 것에서 확장하여 성격 별 훈련방법, 반려인의 성격과 환경을 설문조사를 통해 강아지 품종 별 특징과 매칭하여 강아지와 반려인의 궁합도를 수치화해서 보여준다.

- 반려동물 양육을 처음 시작하는 반려인들을 위한 기초적인 배변훈련, 산책 방법 등 생활에 필요한 기본정보를 제공하는 웹페이지를 추가하여 나의 반려동물에게 맞는 방법을 찾을 수 있도록 돕는다.

2-1-2. 블로그를 통한 정보 수집 및 활용 방안

- 블로그의 태그와 게시판 카테고리 기능을 연동하여 건강검진 또는 건강 관련 태그의 내용을 데이터베이스에 저장하여 이상 증상이 일정기간 지속된 경우, 간편 건강 기록 수첩의 기능을 추가하여 병원에 건강 기록 데이터를 전송할 수 있도록 서비스를 확장한다. 이를 통해 반려동물의 건강을 보다 체계적으로 관리 할 수 있게 돕는다.

- 산책, 미용, 식당과 같은 태그를 가진 게시글의 장소를 데이터베이스에 저장하고 해당 데이터의 좌표 값을 이용하여 위치기반 서비스를 통해 주변에 반려동물과 함께 생활할 수 있는 장소의 정보를 제공한다. 이용 후 리뷰와 평점을 입력 받아 업체에 대한 만족도를 평가한다. 반려견과 반려인이 만족했던 업체 유형을 분석하여 추천하는 서비스로 확장시킨다.

2-2 기대효과

많은 반려인들이 반려동물과 함께하는 시간동안 어떻게 양질의 삶을 줄 것인가에 대한 끝없는 고민을 한다.
AnimaR 서비스를 통해 기본적인 강아지의 품종 별 특성부터 나의 반려동물의 특성으로 구체화하여 스쳐 지나갈 수 있는 정보를 기록하고 저장하여 데이터화 시킬 수 있다.
이를 통해 반려인이 반려견의 성격, 취향, 건강상태 등 반려동물에 대한 지속적인 관심을 데이터화 하여 정리해줌으로 체계적인 반려동물 케어를 할 수 있도록 돕는다.
향후 반려 견 외 다른 반려동물로 서비스를 확대하여, 반려동물 양육가구들이 반려동물과의 생활을 즐겁게 보낼 수 있게 한다.

3. 프로젝트 후기

송 동익: 항상 프로젝트를 진행하면서 느끼는 것이 데이터셋의 중요성이었다. 이번에도 역시 잘못된 데이터셋으로 인해서 딥러닝 모델의 학습 정확도, 오차가 너무나도 안 좋게 나왔었고, 데이터셋이 많다고 좋은 것이 아닌, 어떤 특징을 중점으로 많이 데이터셋을 모았는지에 대해서 중요성을 느끼게 되었다.
또한 장고라는 새로운 프레임 워크를 사용하면서, 프레임워크에 대해서 또 세계관이 넓어졌으며, 어떠한 새로운 환경에 맞닥뜨리더라도, 헤쳐 나갈 수 있다는 자신감을 얻게 되었다.

신 성보 : 데이터 셋은 다 같이 찾고 분업화를 진행하였습니다. 그 과정에서 이미지의 데이터도 검증 과정이 중요하다는 것을 다시 한번 확인하였습니다.
충분히 검증하고 학습을 진행했음에도 불구하고 학습과정에는 데이터 셋이 적합하지 않아 각각 분업 화하는 과정에서도 동시에 데이터 셋 검색 과정을 동시에 진행하게 되어서 조금 힘들었습니다. 웹 프런트 디자인과 백 엔드 부분을 같이 진행하면서 시간이 부족함을 많이 느꼈습니다.
마음은 며칠이면 끝낼 거 같았지만 오류가 나는 경우는 구글링을 통해 오류를 잡는다고 예상시간보다 더 오래 걸렸습니다.
다음에는 같은 오류가 발생한다면 능숙하게 다룰 수 있어 자신감을 얻는 좋은 프로젝트가 되었습니다.

정 지현 : 사회적 이슈에서 아이디어를 얻어 기술을 구현할 수 있도록 데이터 셋을 찾고, 실제 구현해보며 현실화가 가능한 아이디어인지 판단할 수 있는 좋은 경험이었습니다. 또한 진행과정에서 데이터의 문제점이 들어나 보강을 위해 데이터 셋을 추가로 찾는 작업을 하며, 검증과정의 중요성 또한 느낄 수 있었습니다.
파이썬 웹 프레임워크인 장고를 사용해보며 배우지 않은 프로그래밍 언어도 노력을 통해 학습화 할 수 있다는 자신감을 얻었습니다. 부족한 부분은 팀원들과 채워 나가며 정해진 시간 내에 효율적으로 결과를 낼 수 있게 서로 배려하며 프로젝트를 진행한 뜻깊은 경험을 할 수 있었습니다.

Ⅳ. 참고사이트 및 문헌

- KB금융지주 2021 한국 반려동물보고서
- KOSIS 국가 통계 포털 반려가구 통계자료
- 네이버 뉴스

- 데이터 수집
  - 강아지 이미지 - https://www.kaggle.com/
    - https://www.google.com/
    - https://www.kaggle.com/datasets/amandam1/120-dog-breeds-breed-classification
  - 동물병원, 약국 정보 - https://localdata.go.kr/
  - 동물 특성 정보 - https://ko.wikipedia.org/wiki

- 구글 맵 API
  - https://developers.google.com/maps

- 위키독스
  - https://wikidocs.net/