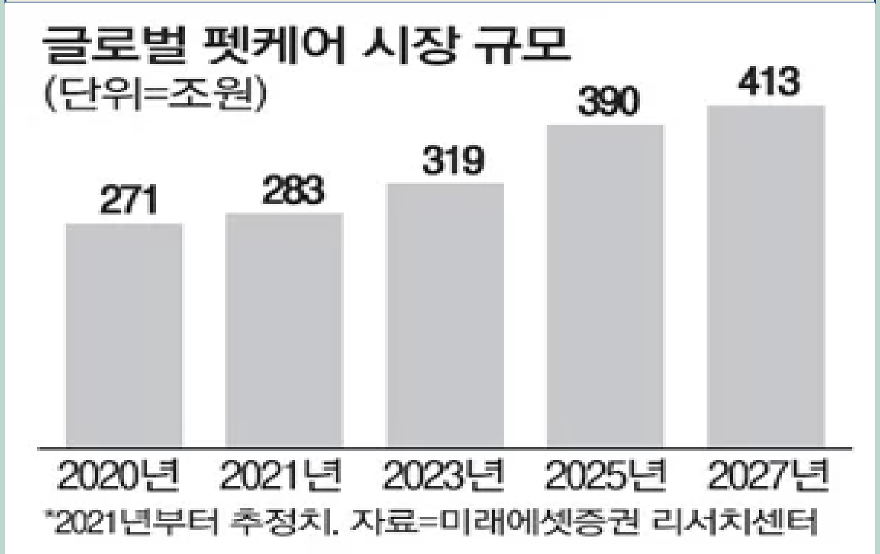
반려동물 질병 진단

pet-mily

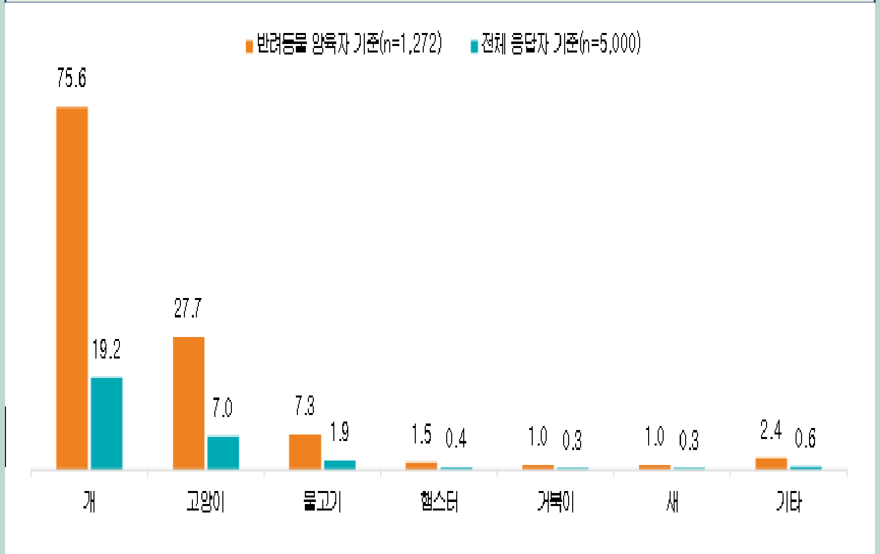
목차

주제 선정 이유

글로벌 펫케어 시장 규모와 전망

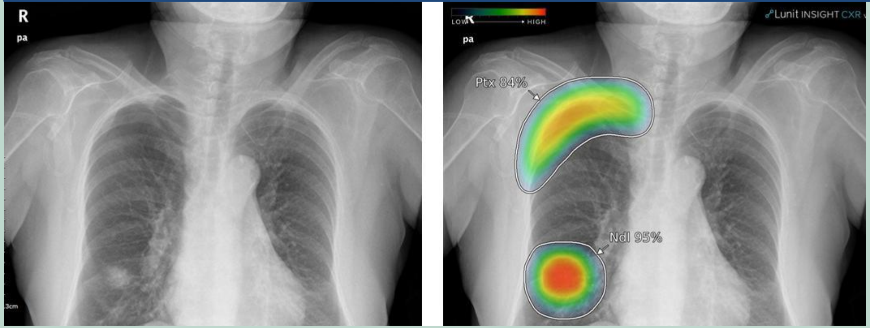


2022년 동물 보호에 대한 국민의식 조사 결과 발표에 따른 2022년 국내 반려동물 비율



주제 선정 이유

의료 AI를 통해서 폐질환을 찾는 영상판독

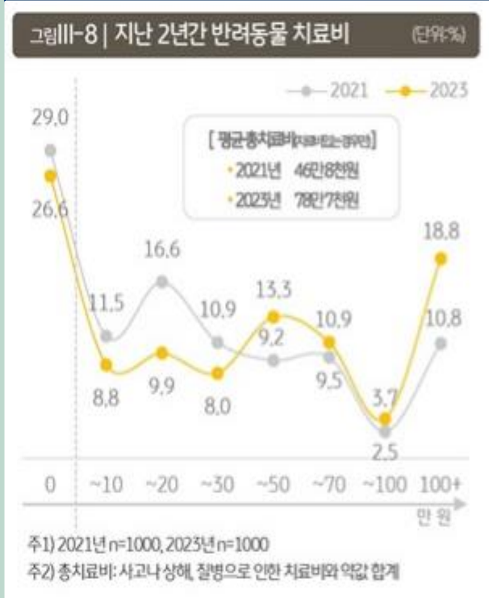


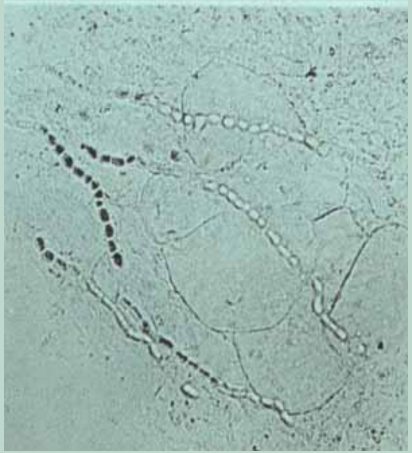
		영상의학과 의사 단독 판독	인공지능 시스템 보조 판독
초응급 질환 (기흉, 기복증, 대동맥 박리)	진단 정확도	29.2%(7/24)	70.8%(17/24)
	판독 대기 소요 시간	3371 초	640 초
응급 질환 (폐렴, 폐부종, 활동성 결핵, 간질성 폐질환, 폐결절, 흉수, 종격동 종양, 늑골 골절)	진단 정확도	78.2%(244/312)	82.7%(258/312)
	판독 대기 소요 시간	2127 초	1840 초
비응급 질환/ 정상	진단 정확도	91.4%(801/876)	93.8%(822/876)
	판독 대기 소요 시간	2815 초	3267 초

▲ 응급실 모의 판독 실험 결과. 출처: 서울대병원

주제 선정 이유

2021 ~ 2023년 동안 반려동물 치료비 & 연령별 치료비 비율





인수공통감염병 링웜(Ringworm)

주제 선정 이유

2021 ~ 2023년 반려동물 치료비 지출처 비율

데이터 셋

반려동물 안구질환

구분	촬영위치	장비	질환	중증도	이미지 수
반려견	표면	검안경 및 일반 카메라	안검염	유	8,600
				무	8,600
			안검종양	유	6,000
				무	6,000
			안검 내반증	유	12,000
				무	12,000
			유루증	유	12,000
				무	12,000
			색소침착성각막염	유	8,800
				무	8,800
			궤양성	무	8,600
			각막질환	상	8,600
				하	8,600
			핵경화	유	12,000
				무	12,000
			결막염	유	12,000
				무	12,000
			비궤양성 각막질환	상	6,000
				하	6,000
				무	6,000
	내면		백내장	초기	8,600
				비성숙	8,600
				성숙	8,600
				무	8,600
		안구 초음파	백내장	유	8,600
				무	8,600
	후면	안저 카메라 안구 초음파	유리체변성	상	10,000
				하	10,000
				무	10,000

반려동물 피부질환

구분	장비	증상	증상 여부	이미지 수
반려견	일반 카메라 스마트폰 카메라 (피부질환 이미지)	구진,플라크	유증상	40,600
			무증상	40,600
		비듬,각질,상피성잔고리	유증상	67,300
			무증상	67,300
		태선화,과다색소침착	유증상	67,300
			무증상	67,300
		농포,여드름	유증상	15,000
			무증상	15,000
		미란,궤양	유증상	15,000
			무증상	15,000
		결절,종괴	유증상	15,000
			무증상	15,000
	장비	질병	질병구분	이미지 수
	현미경용 디지털카메라 (cytology)	감염성 피부염	감염성	3,000
			비감염성	3,000

구분	장비	증상	증상 여부	이미지 수
반려묘	일반 카메라 스마트폰 카메라 (피부질환 이미지)	농포,여드름	유증상	8,460
			무증상	8,460
		비듬,각질,	유증상	8,460
		상피성잔고리	무증상	8,460
		결절,종괴	유증상	8,460
			무증상	8,460
	장비	질병	질병구분	이미지 수
	현미경용 디지털카메라 (cytology)	감염성 피부염	감염성	1,420
			비감염성	1,420

데이터 전처리

데이터 샘플링

```python
import os
import random
import shutil

source_folder = 'C:/kkm/pet_eyes'

train_folder = 'C:/kkm/pet_eyes/train'
val_folder = 'C:/kkm/pet_eyes/val'

image_extensions = ['.jpg', '.jpeg', '.png']

split_ratio = 0.8

disease_folders = [folder for folder in os.listdir(source_folder) if os.path.isdir(os.path.join(source_folder, folder))]

os.makedirs(train_folder, exist_ok=True)

os.makedirs(val_folder, exist_ok=True)

# 각 질병 폴더 별로 분할 작업
for folder_name in disease_folders:

    disease_folder = os.path.join(source_folder, folder_name)


    disease_files = os.listdir(disease_folder)
    image_files = [file for file in disease_files if any(file.lower().endswith(ext) for ext in image_extensions)]


    random.shuffle(image_files)
    split_index = int(len(image_files) * split_ratio)
    train_files = image_files[:split_index]
    val_files = image_files[split_index:]


    for train_file in train_files:
        src_path = os.path.join(disease_folder, train_file)
        dst_path = os.path.join(train_folder, train_file)
        shutil.copy(src_path, dst_path)


    for val_file in val_files:
        src_path = os.path.join(disease_folder, val_file)
        dst_path = os.path.join(val_folder, val_file)
        shutil.copy(src_path, dst_path)
```

- 질병 폴더안에 json 파일과 같은 이름의 이미지 파일 몇개인지 추출
- 8대2 비율로 train,val 폴더 생성

label 전처리

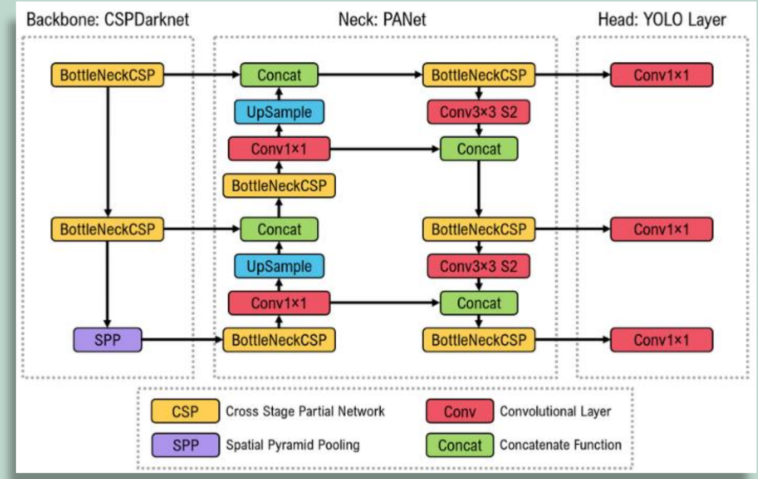
```python
# yolo에 맞게 바운딩박스 수치 수정하는 함수
def convert_bbox_to_yolo(label_bbox, image_width, image_height):
    x1, y1, x2, y2 = map(float, label_bbox)

    bbox_width = x2 - x1
    bbox_height = y2 - y1

    x_center = (x1 + bbox_width / 2) / image_width
    y_center = (y1 + bbox_height / 2) / image_height
    bbox_width = bbox_width / image_width
    bbox_height = bbox_height / image_height

    return [x_center, y_center, bbox_width, bbox_height]

# json파일 txt로 바꾸는 함수
def convert_json_to_txt(json_folder_path, txt_folder_path, class_label):
    for filename in os.listdir(json_folder_path):
        if filename.endswith('.json'):
            json_path = os.path.join(json_folder_path, filename)
            txt_path = os.path.join(txt_folder_path, f'{os.path.splitext(filename)[0]}.txt')

            with open(json_path, 'r') as f:
                data = json.load(f)

            image_width, image_height = map(int, data['images']['meta']['width_height'])
            label_bbox = data['label']['label_bbox']
            yolo_bbox = convert_bbox_to_yolo(label_bbox, image_width, image_height)

            with open(txt_path, 'w') as txt_file:
                txt_file.write(f'{class_label} {" ".join([format(coord, ".6f") for coord in yolo_bbox])}\n')
```

- json파일 x, y, width, height 좌표추출
- yolo라벨 형식에 맞게 txt파일 생성

모델설명: YOLOv5

<모델특징>

1. 높은 FPS
2. bbox를 이용한 객체 탐지



<모델구조>

- Backbone - CSPDarknet
  - cnn 학습능력강화
  - 연산 bottleneck 제거
  - 메모리 cost감소

- Neck - PANet
  - 낮은레벨의 피처와와 높은레벨의 피처를 섞어서 성능 향상 시킴

- Head - YOLO layer
  - Neck으로 부터 추출된 피처를 바운딩박스 파라미터, 객체 확률로 반환

yolov5 사용코드

1. github 다운

```
%cd /content
!git clone https://github.com/ultralytics/yolov5.git
%cd /content/yolov5/
!pip install -r requirements.txt

import torch
from IPython.display import Image, clear_output
```

2. yolov5s.yaml

```
# Parameters
nc: 7  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]],  # 9
  ]

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

3. train

```
!python train.py --img 640 --batch 8 --epochs 20 ₩
--data /content/drive/MyDrive/project/고인메리호/yolov5/labels.yaml ₩
--weight /content/drive/MyDrive/project/고인메리호/yolov5/runs/train/exp32/weights/last.pt
```

<Hyper Parameter>

```
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

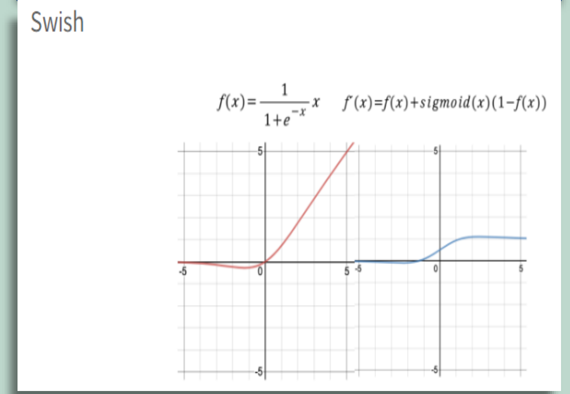
모델 결과 - yolov5

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 25/25 [02:04<00:00, 4.97s/it]
all	396	396	0.329	0.813	0.435	0.429

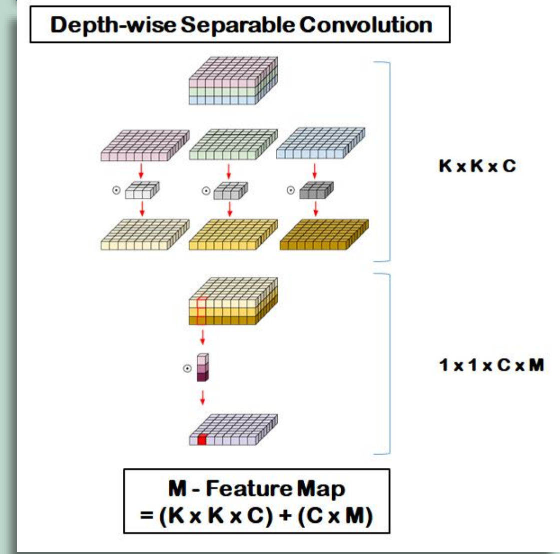
모델설명: EfficientNetB0

<모델특징>

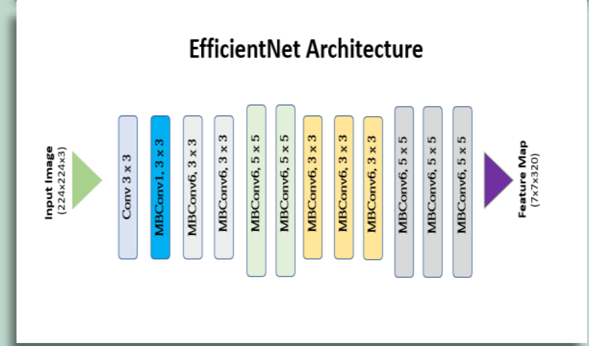
1. Swish Activation 으로 구성된 MBConv 블록 사용



2. MBConv 블록에 Depthwise Separable Convolution 이 포함되어 있어 계산량과 파라미터 수를 줄임



3. 530만개의 파라미터를 사용해서 컴퓨팅 리소스를 절약



EfficientNetB0 사용코드

데이터 샘플링

```python
if not data_already_created:
    # 데이터세트를 분할 및 복사하는 코드는 'if not data_already_created' 블록 내에 위치
    for classification_dir in classification_dirs:
        for category, base_path in [('negative', negative_base_path), ('positive', positive_base_path)]:
            src_dir = base_path + classification_dir
            files = os.listdir(src_dir)

            random.shuffle(files)

            train_count = int(split_ratio[0] * len(files))
            test_count = int(split_ratio[1] * len(files))

            train_files = files[:train_count]
            test_files = files[train_count:train_count + test_count]
            val_files = files[train_count + test_count:]

            train_classification_path = train_base_path + classification_dir + '/'
            test_classification_path = test_base_path + classification_dir + '/'
            val_classification_path = val_base_path + classification_dir + '/'



            # 데이터를 복사합니다.
            for dataset, dataset_path, dataset_files in [('train', train_classification_path, train_files),
                                                          ('test', test_classification_path, test_files),
                                                          ('val', val_classification_path, val_files)]:
                dest_dir = os.path.join(dataset_path, category)
                os.makedirs(dest_dir, exist_ok=True)
                for file in dataset_files:
                    shutil.copy(os.path.join(src_dir, file), os.path.join(dest_dir, file))

# 나머지 코드 (EfficientNet 모델 생성, 적합 및 훈련)는 이전과 동일합니다.


# 강아지 양성 폴더 파일 갯수 확인
# import os

path = '/content/drive/MyDrive/kijae_yunho_hwijae/dog/test'

# Function to count files recursively in a directory
def count_files(directory):
    file_count = 0
    for root, dirs, files in os.walk(directory):
        file_count += len(files)
    return file_count

# Get a list of all subdirectories in the path
subdirectories = [os.path.join(path, folder) for folder in os.listdir(path) if os.path.isdir(os.path.join(path, folder))]

# Print the count of files in each subdirectory
for subdir in subdirectories:
    file_count = count_files(subdir)
    print(f"Number of files in {subdir}: {file_count}")
```

- train, test, val 폴더를 만들어서 데이터를 저장

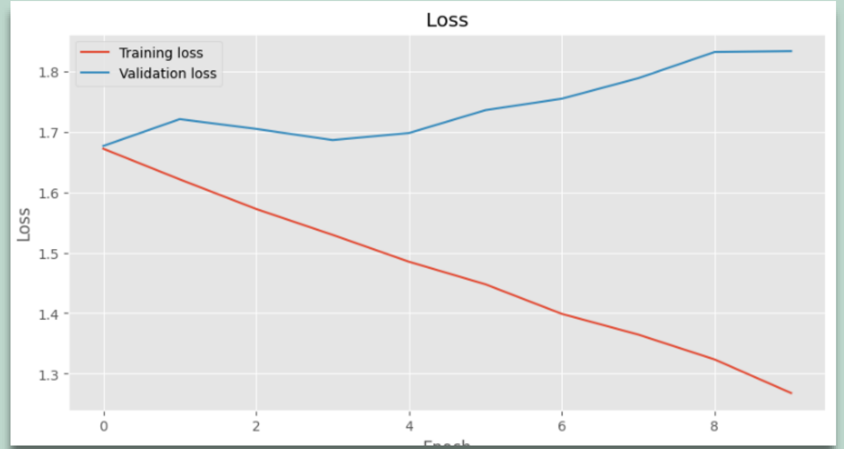
모델 훈련

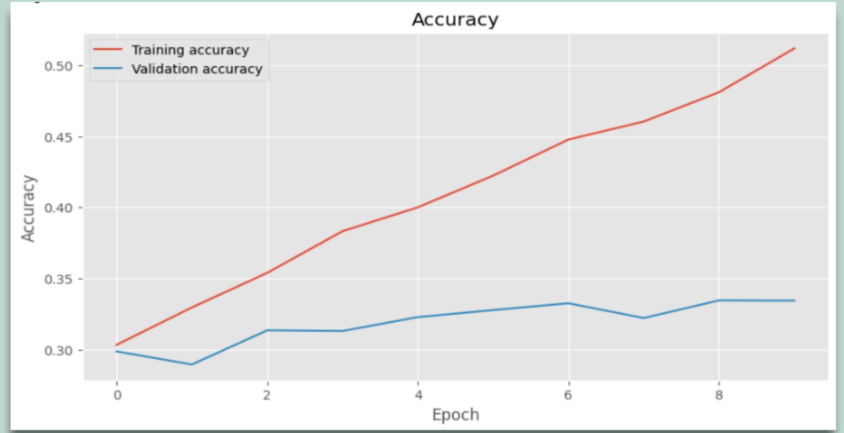
```python
num_classes = len(classification_dirs)
base_model = EfficientNetB0(weights="imagenet", include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(num_classes, activation="softmax")(x)

model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable = False

optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=["accuracy"])

BATCH_SIZE = 32

from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest",
)

train_generator = train_datagen.flow_from_directory(
    train_base_path,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode="categorical"
)

val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
val_generator = val_datagen.flow_from_directory(
    val_base_path,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode="categorical"
)

TRAIN_STEPS = train_generator.__len__()
VAL_STEPS = val_generator.__len__()

print(f"Updated steps_per_epoch: {TRAIN_STEPS}")
print(f"Updated validation_steps: {VAL_STEPS}")


# 모델 체크포인트
checkpoint_filepath = "/content/drive/MyDrive/disease_classifier_checkpoint.h5"
checkpoint = ModelCheckpoint(
    checkpoint_filepath, monitor="val_loss", verbose=1, save_best_only=True, mode="min", save_weights_only=True
)
```

- imagenet으로 pretrain된 efficinetnetb0를 불러옴
- 이미지 사이즈즈 224*224, batch_size : 32 epoch:10

모델 결과 - EfficientNetB0

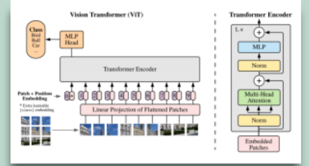
```
Epoch 1/10
448/448 [==============================] - 1110s 2s/step - loss: 1.6719 - accuracy: 0.3034 - val_loss: 1.6765 - val_accuracy: 0.2988
Epoch 2/10
448/448 [==============================] - 1132s 3s/step - loss: 1.6212 - accuracy: 0.3297 - val_loss: 1.7208 - val_accuracy: 0.2896
Epoch 3/10
448/448 [==============================] - 1114s 2s/step - loss: 1.5723 - accuracy: 0.3540 - val_loss: 1.7047 - val_accuracy: 0.3136
Epoch 4/10
448/448 [==============================] - 1114s 2s/step - loss: 1.5295 - accuracy: 0.3833 - val_loss: 1.6862 - val_accuracy: 0.3132
Epoch 5/10
448/448 [==============================] - 1130s 3s/step - loss: 1.4851 - accuracy: 0.4001 - val_loss: 1.6979 - val_accuracy: 0.3229
Epoch 6/10
448/448 [==============================] - 1128s 3s/step - loss: 1.4478 - accuracy: 0.4224 - val_loss: 1.7357 - val_accuracy: 0.3279
Epoch 7/10
448/448 [==============================] - 1142s 3s/step - loss: 1.3989 - accuracy: 0.4477 - val_loss: 1.7546 - val_accuracy: 0.3327
Epoch 8/10
448/448 [==============================] - 1145s 3s/step - loss: 1.3646 - accuracy: 0.4604 - val_loss: 1.7885 - val_accuracy: 0.3223
Epoch 9/10
448/448 [==============================] - 1174s 3s/step - loss: 1.3235 - accuracy: 0.4810 - val_loss: 1.8319 - val_accuracy: 0.3347
Epoch 10/10
448/448 [==============================] - 1169s 3s/step - loss: 1.2680 - accuracy: 0.5118 - val_loss: 1.8332 - val_accuracy: 0.3345
```



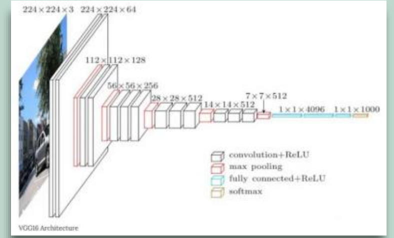


한계점

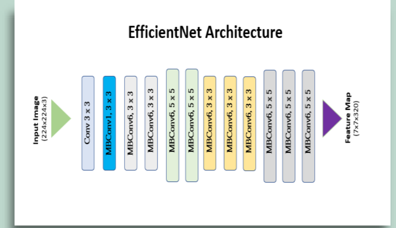
1. 많은 이미지 분류모델 사용



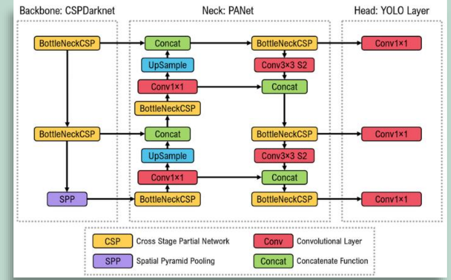
<vision transformer>



<VGG 16>



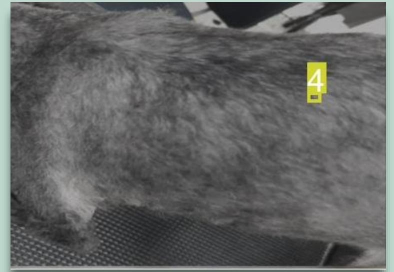
<EfficientNetB0>



<Yolo>

2. 데이터 바운딩 박스 좌표

<안구 바운딩박스>

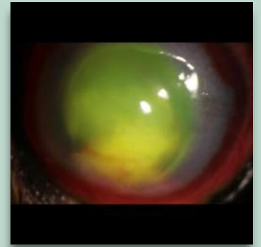


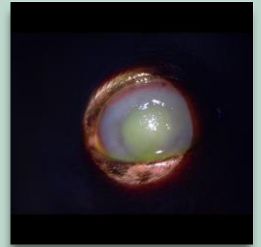
<피부 바운딩박스>

한계점

3. 주어진 데이터의 한계

1) 다른 질병이지만 비슷한 특성을 지니고 있다.


궤양성각막질환


비궤양성각막질환


색소침착성각막염

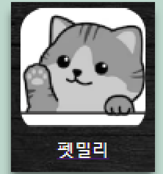
2) 같은 질병이지만 공통된 특성이 없다.

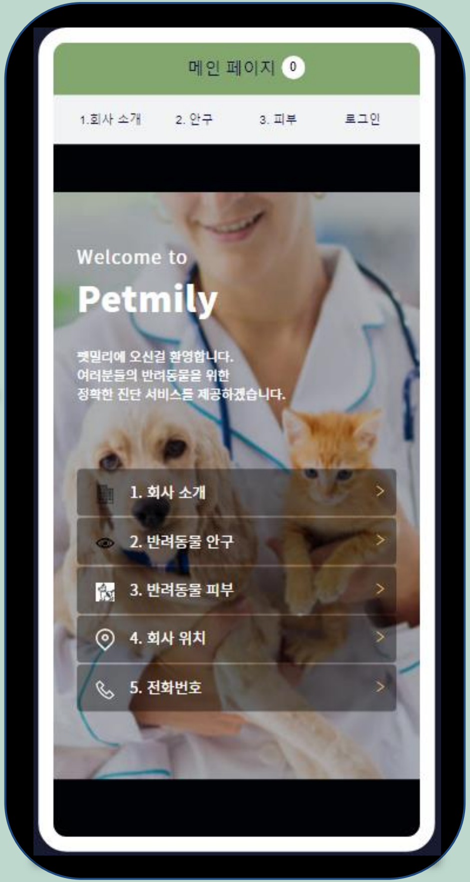

유루증


유루증


유루증

활용방안

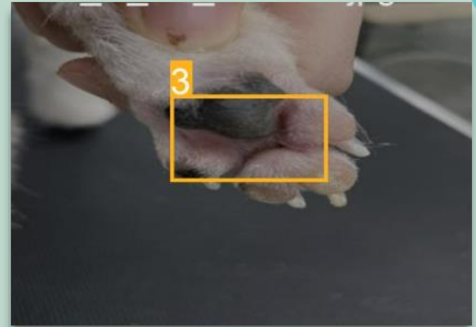




안구 질병 분류

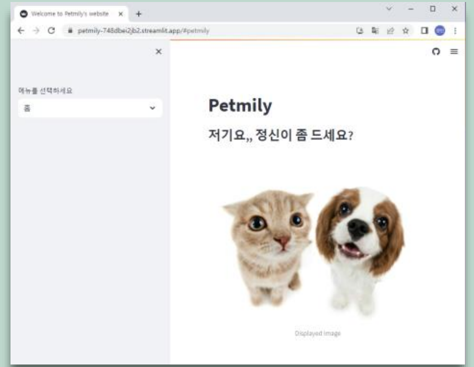
ex) 백내장(초기) 탐지

피부 질병 분류

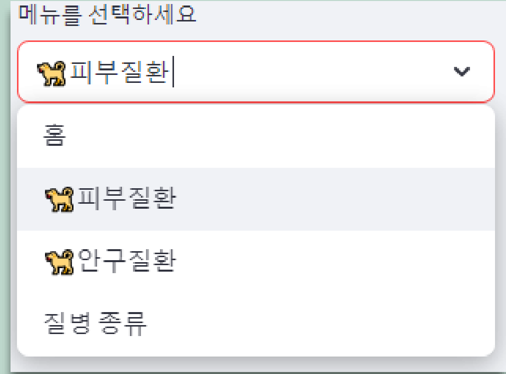


ex) 과다 색소침착 탐지

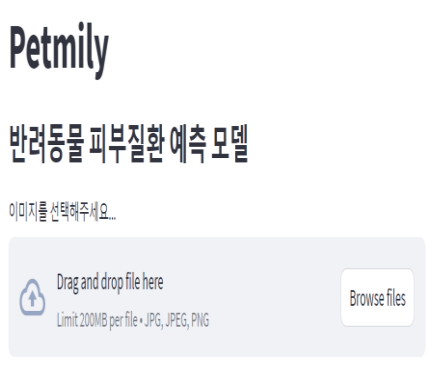
활용방안

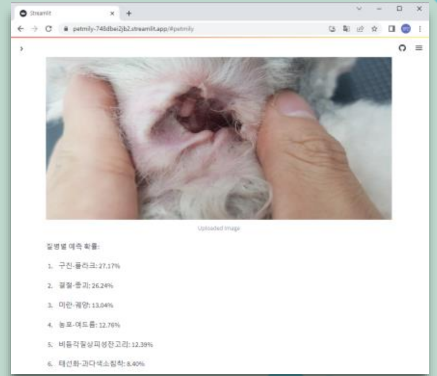


<펫밀리 홈페이지>



<분류 질병 선택>





<질병 분류>

감사합니다

펫밀리