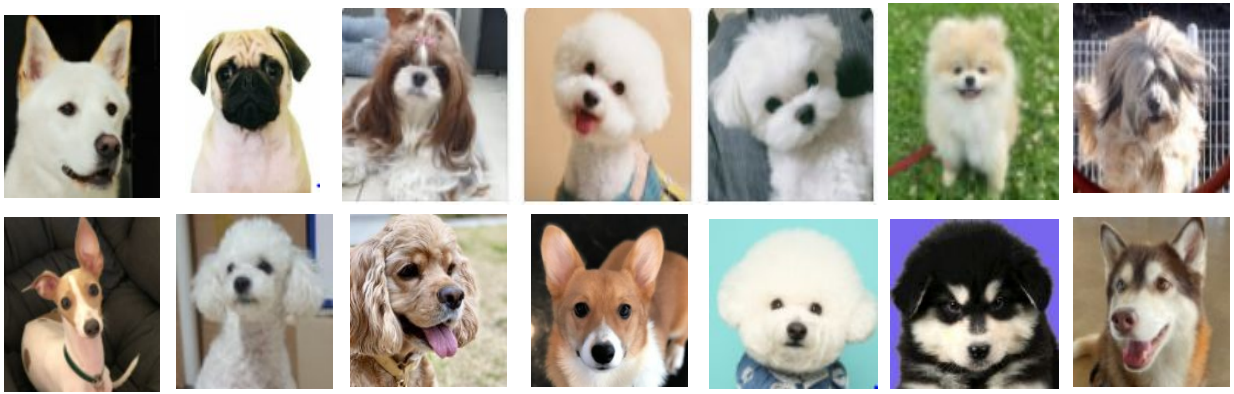
강아지 품종 이미지 분류

4조 - 김현나, 손보영, 이종헌, 전영욱, 허권

14종 강아지 품종 이미지 분류모델



14종의 강아지 품종을 예측하는 이미지 분류모델을 만들고자 한다.

1. 코카스파니엘
2. 푸들
3. 그레이하운드
4. 말티즈
5. 퍼그
6. 비숑
7. 진도개
8. 삽살개
9. 시베리안 허스키
10. 말라뮤트
11. 닥스훈트
12. 웰시코기
13. 리트리버
14. 포메라니안

Workflow


Image Crawling
데이터 수: 11,289,
출처: NAVER, GOOGLE
Preprocessing
Resizing,
Zero centering,
Gray scale
Modeling
ResNet50,
Transfer learning

Image Crawling

1. site:
   google image, naver image

2. method:
   selenium

3. result:
   label = 14개
   data = 11,289개
   (train set : test set = 0.8 : 0.2)

```python
options = webdriver.ChromeOptions()
options.add_argument('--headless')  # Head-less 설정
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome('chromedriver', options=options) #chrome driver 설치한 경로 작성
driver.get("https://search.naver.com/search.naver?where=image&sm=tab_jum") #구글 이미지 검색 url
element = WebDriverWait(driver, 5)  #켜질때의 대기시간
#암묵적 대기 -> webdriverwait : 켜질때까지 기다리고 parsing해야해서 이런처리하는것

elem = driver.find_element_by_name("query") #구글검색창 선택
name = "리트리버" # 검색할 검색어 입력
elem.send_keys(name) #검색창에 검색할 내용 넣기
elem.send_keys(Keys.RETURN) #검색할 내용 넣고 enter 치는것


SCROLL_PAUSE_TIME = 2 #scroll 내려가는 시간
# Get scroll height
last_height = driver.execute_script("return document.body.scrollHeight") #browser의 높이를 javascript로 찾음


while True:
    # Scroll down to bottom
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") #browser 끝까지 scroll 내림
    # Wait to load page
    time.sleep(SCROLL_PAUSE_TIME)
    # Calculate new scroll height and compare with last scroll height
    new_height = driver.execute_script("return document.body.scrollHeight")
    if new_height == last_height:
        try:
            driver.find_element_by_css_selector(".mye4qd").click() #'결과 더보기'가 뜨는 경우 이를 클릭해라
        except:
            break
    last_height = new_height
```

```python
images = driver.find_elements_by_css_selector("._image._listImage")
count = 1
for image in images:
    image.click()
    imgUrl = driver.find_element_by_xpath("/html/body/div[3]/div[2]/div/div[1]/section[2]/div/div[2]/div/div/div[1]/div[1]/div/div/div[1]/div[1]/img").get_attribute("src")
    opener=urllib.request.build_opener()
    opener.addheaders=[('User-Agent','Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36')] # PC의 User-Agent값 입력
    urllib.request.install_opener(opener)
    urllib.request.urlretrieve(imgUrl, "/content/drive/MyDrive/KDT/offline/mini project3-image/리트리버/naver"+str(count) + ".jpg") # name 이름 폴더 생성후 해당 폴더에 이미지 저장
    count = count + 1

driver.close()
```

Preprocessing

- crawling한 이미지의 사이즈가 제각각임을 확인.
- resizing 작업을 통해 모든 이미지의 사이즈를 동일하게 만듦.
- Zero centering
- Gray scale

Modeling

- ResNet50 model
  -pre-trained model

- Transfer learning
  -fine tuning(conv layer 일부 재학습 + 분류기 학습)
  -conv layer 12층, 15층, 30층 일부 재학습

Modeling result

	Int 처리 150*150 epoch 20	150*150 epoch 10	150*150 epoch 20	200*200 epoch 10	200*200 epoch 20	220*220 epoch 8	220*220 epoch 9
train accuracy	0.9728	0.9260	0.9667	0.9387	0.9678	0.9312	0.9300
test accuracy	0.8189	0.8136	0.8091	0.8441	0.8406	0.8539	0.8512
after fine tuning				0.8565			

*conv layer(15층) + 분류층 학습
*batch size = 50
*gray scale 처리

1. epoch 수를 늘렸다고 무조건 accuracy가 증가하는건 아니었음.
2. 이미지 크기가 커짐에따라 정확도가 높아지는것을 확인.
3. 이미지 크기를 늘리고 epoch수를 줄여도 accuracy가 높은것을 통해 크기의 영향이 훨씬크다는걸 보여줌.
4. ram에 부담이 안되는 크기에 accuracy가 가장 컸던 200*200 epoch 10만 fine tuning실행

Modeling result

	Conv layer(12층) + 분류층 학습, Dropout(0.5) 2번 사용, epoch 10	Conv layer(12층) + 분류층 학습, Dropout(0.5) 1번 사용, epoch 10	Conv layer(30층) + 분류층 학습, Dropout(0.5) 1번 사용, epoch 15
train accuracy	0.7837	0.9633	0.9922
test accuracy	0.8667	0.8940	0.8887
after fine tuning	0.8760	0.8920	0.8920

*data int 화 처리
*image size 180 * 180
*batch size =128
*gray scale 처리 안함

Gray scale 처리를 안하고 바로 color image를 사용하고 drop out 수, conv layer 학습하는 층 수를 변경했더니 accuracy를 최대 89% 까지 증가시킴

개선할 수 있는 부분

Model 을 바꿔도 accuracy가 같다는 건 데이터 측면에서 손봐야 할 것.

1. Overfitting 방지: augmentation을 통해 data양 늘리기, FC층 변경
2. Dataset 품질 개선: 잘못된 데이터 삭제

Issue & Solution

issue 1)

image를 crawling해서 가져오면 각자 사이즈들이 다르기때문에 resize를 해주는데 이때 224*224로 통일시켰더니 image가 너무 커서 ram이 터짐

solution)

-import gc, 쓸모없는 변수 삭제 등으로 메모리 확보했으나 -> 소용 x

-image pixel숫자가 float형이어서 image array를 int로 바꿔서 해보기(약간의 손실이 있을 순 있지만 그래도 용량 줄이기위해) -> 소용 x

-image size를 224*224가 아닌 좀 더 작은 size로 해보기

Issue & Solution

issue 2)

resize시 ‘1차원 → 3차원 → resize’ 를 하려고했는데 data가 너무커서 오류발생

solution)

repeat 이전 1차원에서 resize를 하고나서 그 다음 3차원으로 하는 순으로 하니 해결됨.

Issue & Solution

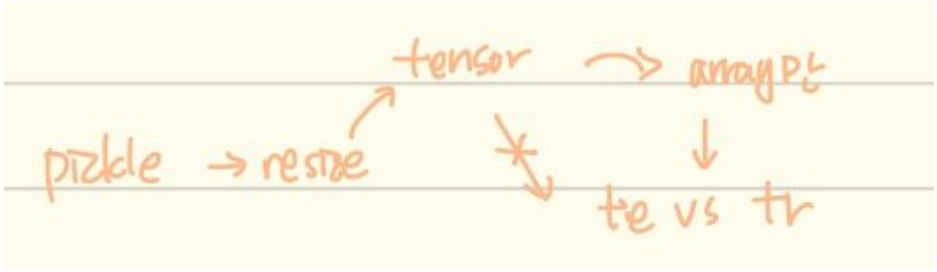
issue 3)
pickle을 resize시킨것으로 train & test split시 오류가 남.

solution)

불러온 pickle을 다시 resize를 한다면 tensor형식으로 저장됨.

train set & test set으로 넣을땐 array형태만 가능.

그러므로 tensor 형식으로 저장된걸 array형식으로 형변환을해야 train&test로 split 가능.



감사합니다