Deep Learning Algorithm

인공지능 플랫폼 과정

늑대 vs 허스키

CNN 알고리즘 분석 테스트

팀 2

이도규

윤연석

강진성

01

데이터 수집

구글 이미지로 부터 이미지 데이터 가져오기

02

데이터 학습

이미지 데이터를 전처리하기
CNN알고리즘에 학습하기

03

데이터 평가

데이터 평가 및 프로젝트 결론

Insert a sub-title

데이터 학습

데이터 구분

전처리 코드에 대한 설명

01. 데이터 수집

주제 선정 및 수집방법

비교 이미지 선정

개 이미지로 검색하면 치와와 등의 사진이
늑대와 형태가 확연하게 차이가 나서
늑대와 닮은 대표 견종 중 시베리안 허스키 채택

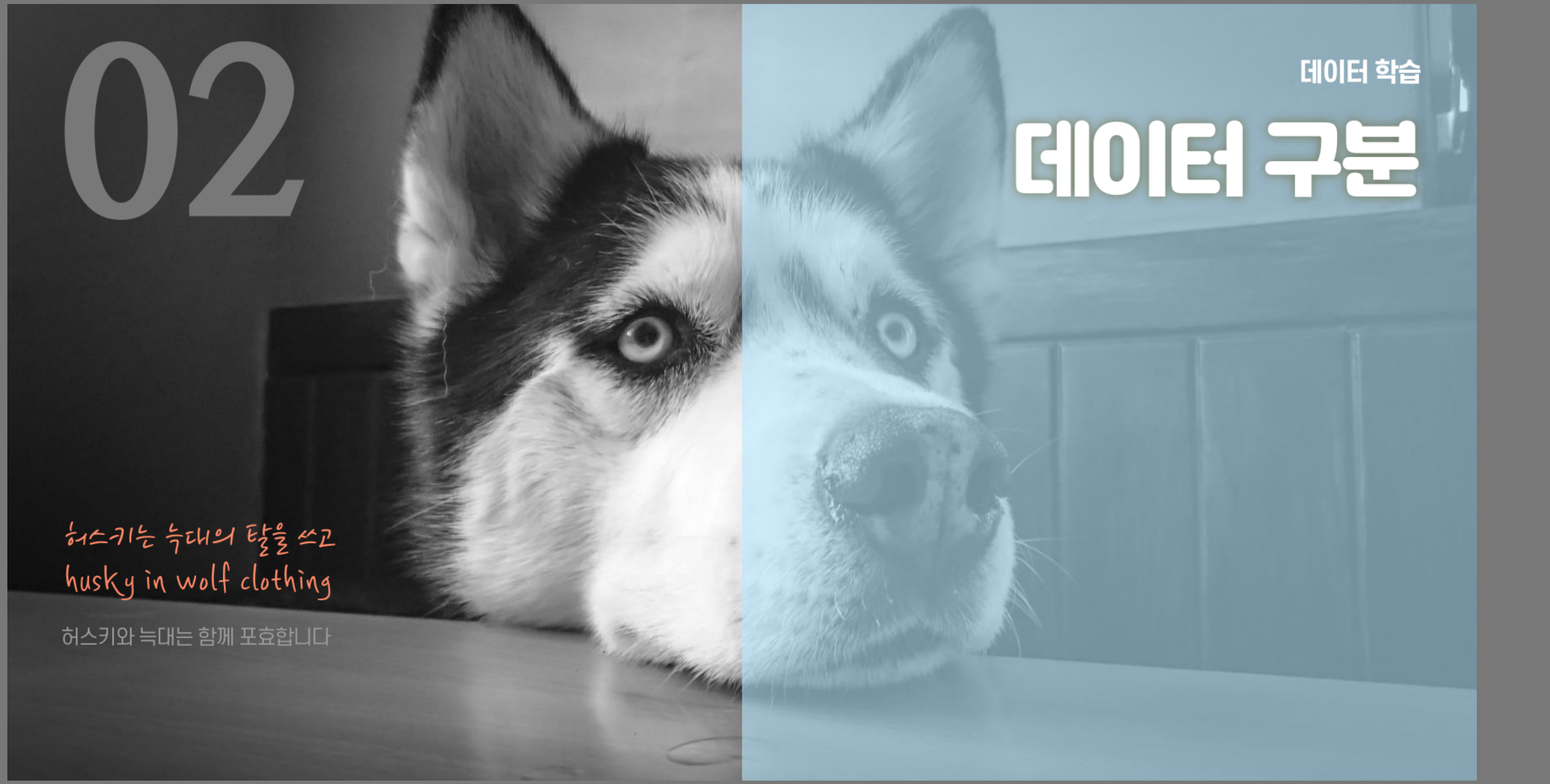
구글 이미지 검색어

구글 이미지에서 1회 검색 시 조회되는 이미지 수가 많지 않음
검색어 변경은 효과 적음 / 다양한 나라의 단어로 검색
— 늑대, wolf, wolf face, волк, 狼 등
— 시베리안 허스키, Siberian husky, Сибирский хаски, シベリアンハスキー 등

이미지 선정 및 분류

사람 등 다른 개체와 함께 있거나 사진이 아닌 그림, 일러스트, 인형
과 같은 부적절한 이미지를 모두 삭제하고 두 폴더로 구분

```python
kw = 'волк'  # wolf 러시아어
# シベリアンハスキー 일본어
# Husky sibérien
# 늑대 한국어
# wolf 영어

elem = driver.find_element(By.ID, 'APjFqb')
elem.send_keys(kw)
elem.send_keys(Keys.RETURN) # 엔터키
time.sleep(2) # 로딩 시간

# 스크롤을 끝까지 내리는 코드
SCROLL_PAUSE_TIME = 2
last_height = driver.execute_script("return document.body.scrollHeight")

while True:
    driver.execute_script('window.scrollTo(0, document.body.scrollHeight);')
    time.sleep(SCROLL_PAUSE_TIME)

    new_height = driver.execute_script('return document.body.scrollHeight')
    if new_height == last_height:
        try:
            time.sleep(SCROLL_PAUSE_TIME)
            driver.find_element(By.CSS_SELECTOR, "input.mye4qd").click()
        except:
            break
    last_height = new_height

    time.sleep(1)

    # 더 보기 버튼 클릭
    try:
        more_btn = driver.find_element(By.CSS_SELECTOR, '#islmp > div > div > div > div > div.C5Hr4 > div.K414
        more_btn.click()
    except:
        pass

elems = driver.find_elements(By.CSS_SELECTOR, '#islrg > div.islrc > div div > a.FRuiCf.islib.nfEiy > div.fR600

for i, elem in enumerate(elems):
    try:
        elem.click()
        time.sleep(1)
        elem = driver.find_element(By.XPATH, '//*[@id="Sva75c"]/div[2]/div[2]/div[2]/div[2]/c-wiz/div/div/div/
        img_url = elem.get_attribute('src')
        print(img_url)
        file_name = r'C:\\Python\\Pandas\\deeplearning\\husky\\' + img_url.split('/')[-1]
        urlretrieve(img_url, file_name)
        print(file_name)
    except:
        pass
```



02

데이터 학습

데이터 구분

허스키는 늑대의 탈을 쓰고
husky in wolf clothing

허스키와 늑대는 함께 포효합니다

02. 데이터 학습

데이터 폴더 구조 및 학습 데이터 선정

- Dog_Wolf_Training
  - Origin
    - Dogs
    - Dogs_origin
    - Wolfs
    - Wolfs_origin
  - test
    - Dogs
    - Wolfs
  - Train
    - Dogs
    - Wolfs
  - validation
    - Dogs
    - Wolfs

Dogs : 허스키 수정본

Dogs_origin : 허스키 원본

Wolfs : 늑대 수정본

Wolfs_origin : 늑대 원본

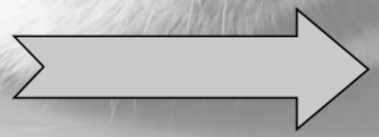
test : 테스트용 이미지

Train : 트레이닝용 이미지

validation : 확인용 이미지

```
wolf_data_train_start = 0
wolf_data_train_end = 250

wolf_data_validation_start = 250
wolf_data_validation_end = 300

wolf_data_test_start = 300
wolf_data_test_end = 350
```

이미지 데이터 개수



```
# 이미지 데이터를 변형처리하는 객체
train_datagen = ImageDataGenerator(rescale = 1/255)
test_datagen = ImageDataGenerator(rescale = 1/255)

# 이미지 데이터를 가져오는 객체

# 데이터를 불러오는 객체
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='binary'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150,150),
    batch_size=20,
    class_mode='binary'
)
```

✓ 0.0s

```
Found 500 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
```

02. 데이터 학습

학습 네트워크 구성

```
# 3. 모델 구성하기
from tensorflow.keras.preprocessing.image import ImageDataGenerator

model = Sequential()

model.add(Conv2D(32,(3,3), activation='relu', input_shape=(150,150,3)))
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(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 148, 148, 32)      896

max_pooling2d_19 (MaxPooli   (None, 74, 74, 32)        0
ng2D)

conv2d_24 (Conv2D)           (None, 72, 72, 64)        18496

max_pooling2d_20 (MaxPooli   (None, 36, 36, 64)        0
ng2D)

conv2d_25 (Conv2D)           (None, 34, 34, 128)       73856

max_pooling2d_21 (MaxPooli   (None, 17, 17, 128)       0
ng2D)

flatten_7 (Flatten)          (None, 36992)             0

dense_41 (Dense)             (None, 128)               4735104

dense_42 (Dense)             (None, 64)                8256

dense_43 (Dense)             (None, 32)                2080

dense_44 (Dense)             (None, 16)                528

dense_45 (Dense)             (None, 1)                 17

=================================================================
Total params: 4839233 (18.46 MB)
Trainable params: 4839233 (18.46 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

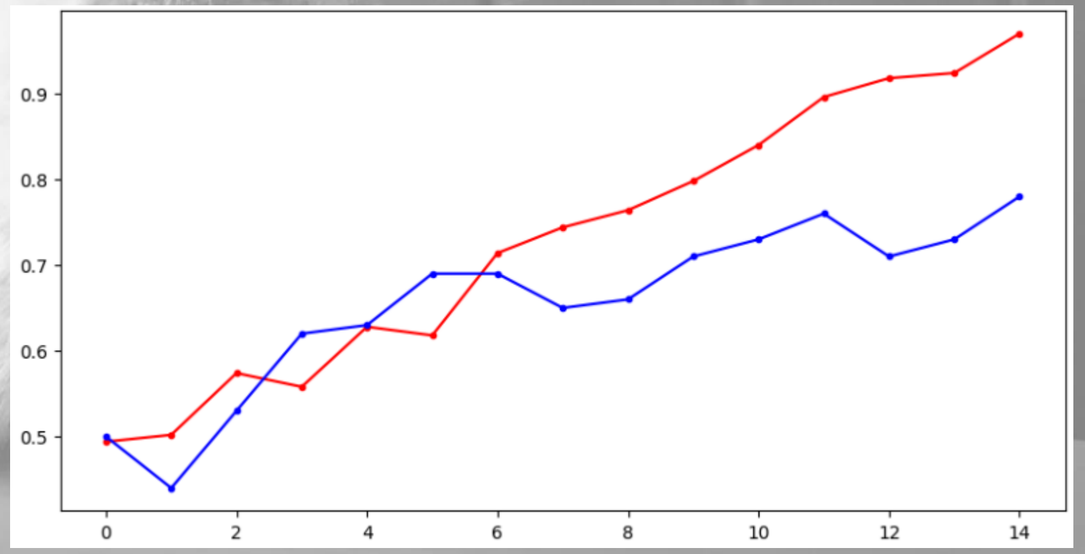
02. 데이터 학습

학습 과정 - 기본

```
history = model.fit(train_generator,
                    epochs=15,
                    validation_data=test_generator)
```

```
loss: 0.7050 - accuracy: 0.4940 - val_loss: 0.6928 - val_accuracy: 0.5000
loss: 0.6934 - accuracy: 0.5020 - val_loss: 0.6928 - val_accuracy: 0.4400
loss: 0.6789 - accuracy: 0.5740 - val_loss: 0.7343 - val_accuracy: 0.5300
loss: 0.6942 - accuracy: 0.5580 - val_loss: 0.6762 - val_accuracy: 0.6200
loss: 0.6938 - accuracy: 0.6280 - val_loss: 0.6821 - val_accuracy: 0.6300
loss: 0.6784 - accuracy: 0.6180 - val_loss: 0.6226 - val_accuracy: 0.6900
loss: 0.5812 - accuracy: 0.7140 - val_loss: 0.5740 - val_accuracy: 0.6900
loss: 0.5378 - accuracy: 0.7440 - val_loss: 0.5928 - val_accuracy: 0.6500
loss: 0.4846 - accuracy: 0.7640 - val_loss: 0.6163 - val_accuracy: 0.6600
loss: 0.4374 - accuracy: 0.7980 - val_loss: 0.6076 - val_accuracy: 0.7100
loss: 0.3527 - accuracy: 0.8400 - val_loss: 0.6387 - val_accuracy: 0.7300
loss: 0.2521 - accuracy: 0.8960 - val_loss: 0.6563 - val_accuracy: 0.7600
```

```
fig = plt.figure(figsize=(10,5))
plt.plot(hist.history['accuracy'], marker='.', color='r')
plt.plot(hist.history['val_accuracy'], marker='.', color='b')
```



02. 데이터 학습

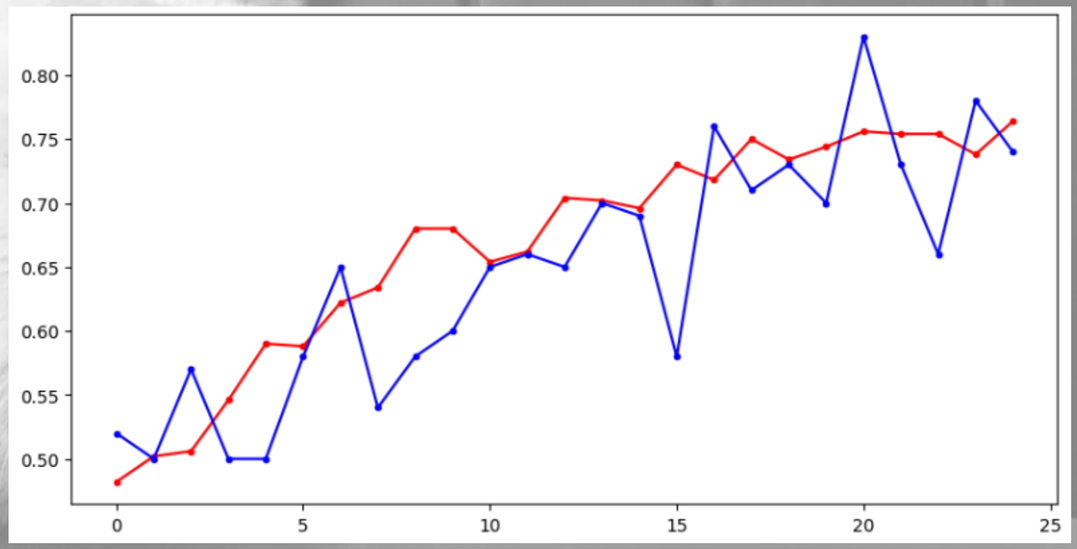
학습 과정 - 데이터 증폭

```python
# 이미지를 변형 및 증폭하는 객체
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip = True,
    vertical_flip = True)
```

```python
hist = model.fit(
    train_generator,
    epochs=25,
    validation_data = validation_generator
)
```

```python
fig = plt.figure(figsize=(10,5))
plt.plot(hist.history['accuracy'], marker='.', color='r')
plt.plot(hist.history['val_accuracy'], marker='.', color='b')
```

```
loss: 0.6208 - accuracy: 0.6800 - val_loss: 0.9230 - val_accuracy: 0.6000

loss: 0.6253 - accuracy: 0.6540 - val_loss: 0.7241 - val_accuracy: 0.6500

loss: 0.6060 - accuracy: 0.6620 - val_loss: 0.6773 - val_accuracy: 0.6600



loss: 0.5090 - accuracy: 0.7380 - val_loss: 0.5819 - val_accuracy: 0.7800

loss: 0.4933 - accuracy: 0.7640 - val_loss: 0.6222 - val_accuracy: 0.7400
```



03

늑대 개
Wolf Dog

늑대는 숲을 우둔하게 하는 [illegible]을 깨고, 개는 [illegible]
을 따스하게 하는 따뜻한 [illegible]입니다.

데이터 평가

평가 및 결론

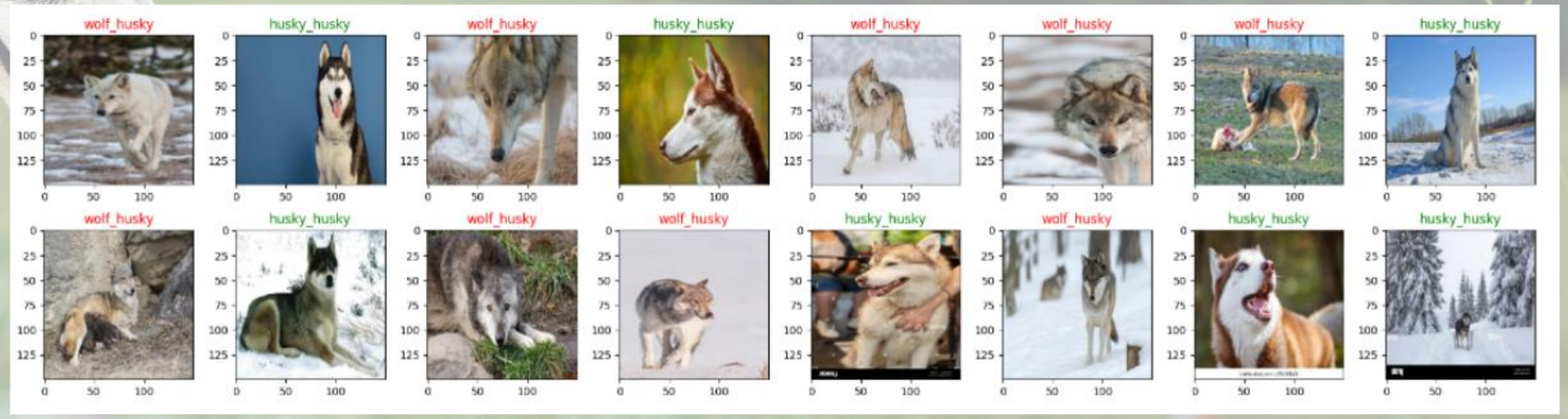
03. 데이터 평가

예측 데이터

기본 모델

예측 테스트 결과 7/16

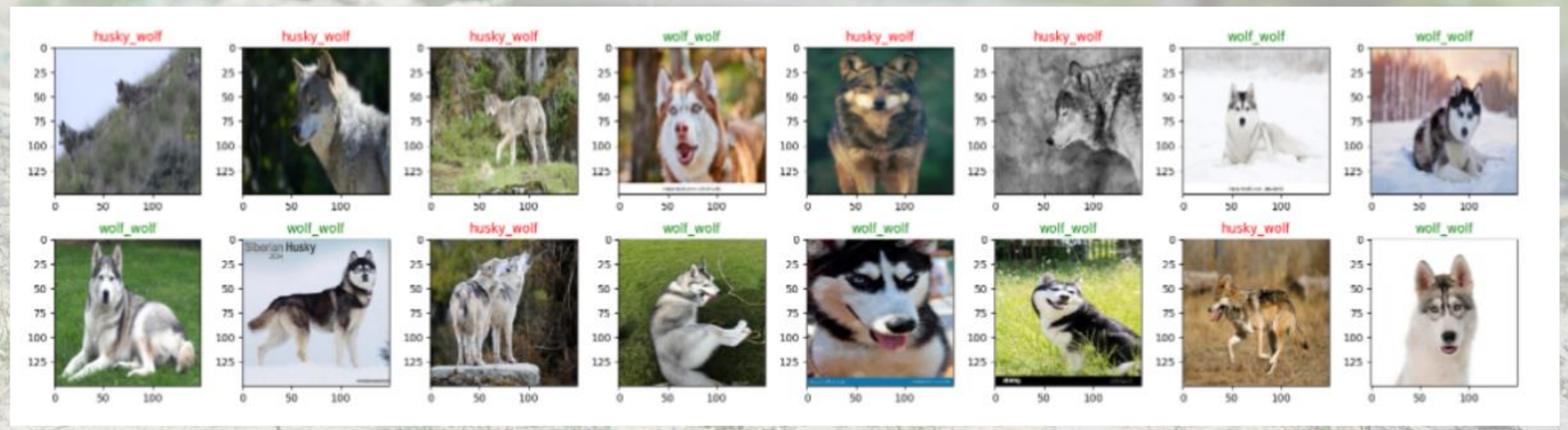
loss: 0.0364 / val_loss: 2.4409
accuracy: 0.9880 / val_accuracy: 0.6500



증폭 모델

예측 테스트 결과 9/16

loss: 0.4933 / val_loss: 0.6222
accuracy: 0.7640 / val_accuracy: 0.7400



03. 데이터 평가

결론

각각 350개의 데이터로 늑대와 허스키를 분류

CNN을 돌린 결과 75% 확률로 학습이 완료되었음을 확인

추후 개선 사항

이미지 학습 방법에 대하여 padding / cropping / resize 등의 적절한 방안 연구

과적합 우려, 적절한 네트워크 구성 연구

이미지 가공 처리

감사합니다.
Q/A