멀티모달 AI 기반 포켓몬 이미지 및 캡션 분석 모델 개발

윤성헌

INDEX

목차

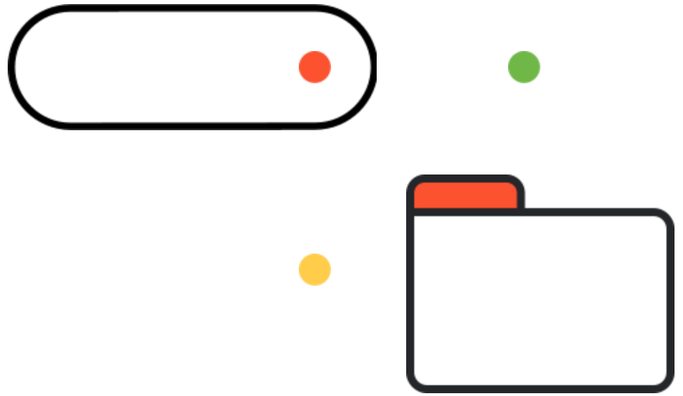
01

인트로 (Introduction)

1.프로젝트 배경 및 문제 정의

2. 프로젝트 목표

3. 기대효과



프로젝트 배경 및 문제 정의

- 기존의 이미지 분류는 주로 이미지 자체의 정보만 사용해서 진행되는 경우가 많음
- 하지만 현실은 이미지와 함께 텍스트 설명이 같이 존재하는 경우가 많음
- 이미지만 단독으로 처리하는 모델보다 이미지와 텍스트를 동시에 이해하고 처리하는 멀티모달(Multimodal) AI 모델이 대세
- 그렇기에 이미지와 캡션 데이터를 함께 학습하는 모델의 경험 필요성 느낌

프로젝트 목표

- 포켓몬 이미지와 그에 따른 캡션(설명 문장)을 같이 분석하는 멀티모달 AI 모델을 만드는 것이 목표
- 이미지 특징 추출에는 사전 학습된 EfficientNetB0 모델을 활용
- 텍스트는 문장 토큰화를 통해 LSTM 신경망으로 처리해 두 정보를 결합

기대 효과

- 이 모델이 성공하면 이미지와 텍스트의 의미적 정보를 연결하여 둘 사이의 관계성을 높임
- 자동으로 이미지에 태그를 붙이거나, 원하는 이미지를 더 정확히 찾는 검색 시스템 개발에 활용
- 전자상거래나 SNS 같은 다양한 서비스에서 콘텐츠를 더 똑똑하게 분류하고 추천하는 데 도움

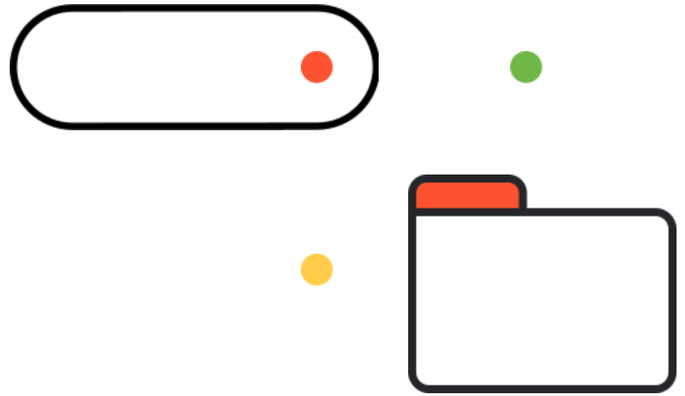
02

데이터 준비 (Data Preparation)

1.데이터셋 소개

2. 데이터셋 처리 코드

3. 데이터 전처리(Preprocessing)



presentation_photo

데이터셋 소개

- 포켓몬 이미지와 해당 이미지의 설명 텍스트(캡션) 사용
- 이미지와 캡션 데이터 불러와 실제 존재하는 이미지와 유효한 캡션만 선별
- 모든 이미지는 224x224 크기로 리사이징
- 1025개의 이미지-캡션 쌍으로 구성

데이터셋 처리 코드

이미지 데이터셋 처리 코드

```
import tensorflow as tf
from tensorflow import keras


# 이미지 크기 지정 및 데이터 증강 레이어 정의
img_size = (224, 224)
data_augmentation = keras.Sequential([
    keras.layers.RandomFlip("horizontal"), # 좌우 반전
    keras.layers.RandomRotation(0.1), # 10% 내외 회전
    keras.layers.RandomZoom(0.1), # 10% 내외 확대/축소
])
```

텍스트 데이터셋 코드

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequences


# 1. 토크나이저 생성 및 학습
# OOV(Out-of-Vocabulary) 토큰은 단어 사전에 없는 단어를 처리하기 위함
tokenizer = Tokenizer(num_words=5000, oov_token="<OOV>")
tokenizer.fit_on_texts(df['caption'])

# 2. 캡션을 정수 시퀀스로 변환
sequences = tokenizer.texts_to_sequences(df['caption'])

 # 3. 패딩 처리로 문장 길이 통일
# 가장 긴 문장 길이 기준
max_seq_len = max(len(seq) for seq in sequences)
padded_seqs = pad_sequences(sequences, maxlen=max_seq_len,
padding='post')
```

데이터 전처리 (Preprocessing)

특징 융합 및 분류

- 이미지와 텍스트에서 나온 특징 연결.
- 이어서 Dense layer를 여러 개 거쳐 최종 sigmoid 활성화 함수로 이진 분류 결과 출력

이미지 브렌치

- 224 X 224 RGB 이미지
- EfficientNetB0 모델을 사용해 이미지 특징을 추출
- 추출된 특징은 GAP를 통해 벡터화 압축되고, Batch Normalization과 Dropout으로 정규화 및 과적합 방지 처리

텍스트 브렌치

- 입력: 토큰화 및 패딩된 캡션 시퀀스 (최대 길이 고정)
- Embedding 층에서 단어를 128차원 벡터 공간에 임베딩
- LSTM 층으로 문장의 순차적 특성과 의미를 파악
- Batch Normalization과 Dropout으로 정규화 및 과적합 방지 처리

Dense Layer: 신경망에서 입력 값을 받아서, 모든 입력을 각각의 가중치와 곱하고, 모두 더한 뒤, sigmoid 활성화 함수를 거쳐 출력 값 생성

sigmoid: 딥러닝에서 자주 쓰이는 활성화 함수 중 하나, 입력 값을 0과 1 사이의 실수로 변환

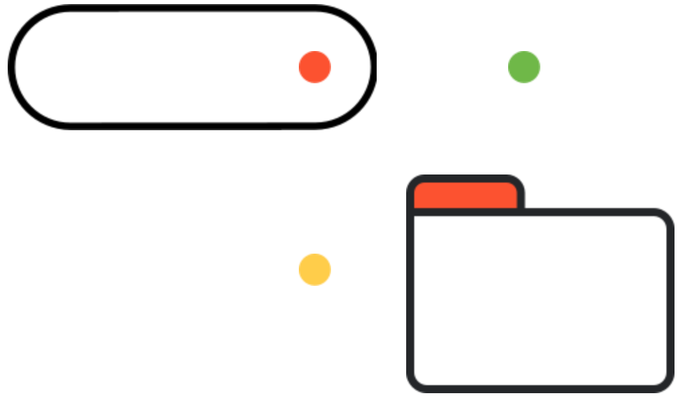
03

모델링 (Modeling)

1.전체 아키텍처

2. 핵심 기술 및 모델 선정 이유

3. 학습 환경



전체 아키텍처

이미지 입력 → EfficientNetB0 → GAP → BN → Dropout

\

→ Concatenate → Dense → BN → Dropout → Output

/

텍스트 입력 → Embedding → LSTM → BN → Dropout

EfficientNetB0: 이미지에서 특징을 추출하는 CNN(합성곱 신경망) 모델

GAP (Global Avarage Pooling): CNN 출력(다차원층 데이터)을 1차원 벡터로 압축

BN(Batch Normalization): 학습을 안정화하고 속도를 높이기 위해 정규화
정규화: 신경망이 학습할 때 각 층에 입력되는 데이터 분포를 일정하게 맞춰줌

Dropout: 과적합 방지를 위해 일부 뉴런을 무작위로 꺼줌

Embedding: 텍스트를 컴퓨터가 이해할 수 있는 숫자 벡터로 변환

LSTM: 문장의 순서를 고려해 정보를 압축

Dense Layer: 신경망에서 입력 값을 받아서, 모든 입력을 각각의 가중치와 곱하고,
모두 더한 뒤, sigmoid 활성화 함수를 거쳐 출력 값 생성

sigmoid: 딥러닝에서 자주 쓰이는 활성화 함수 중 하나, 입력 값을 0과 1 사이의 실수로 변환

핵심 기술 및 모델 선정 이유

EfficientNetB0

- 이미지에서 특징을 추출하는 CNN(합성곱 신경망) 모델
- 적은 계산량으로도 높은 정확도를 달성
- 사전학습 가중치 활용으로 빠른 수렴과 좋은 특징 추출 가능

LSTM

- 순차 데이터(문장)의 순서와 관계를 잘 반영하는 RNN 계열 모델
- 긴 문맥 정보도 기억해 자연어 처리에 효과적
- 캡션과 같은 텍스트의 의미를 파악하는 데 적합

학습 환경

- 손실 함수: binary_crossentropy (이진 분류 문제에 적합)
- 최적화 알고리즘: Adam
  (적응적 학습률과 모멘텀을 결합해 빠르고 안정적인 학습 지원)
- 학습률: 1e-4 (0.0001)
- 배치 크기: 32
- 에포크 수: 최대 30회
- 데이터 증강: 이미지에 랜덤 뒤집기, 회전, 줌 적용해
  과적합 방지 및 일반화 능력 향상
- 환경: TensorFlow 2.17 및 Keras 기반, CPU 계산 환경

```
# 모델 컴파일: 학습 환경 설정
model.compile(
    # 최적화: Adam, 학습률: 0.0001
    optimizer=keras.optimizers.Adam(learning_rate=1e-4),
    loss='binary_crossentropy',  # 손실 함수
    metrics=['accuracy']  # 평가 지표
)
```

```
# 모델 학습 시작
history = model.fit(
    train_dataset, epochs=30, # 최대 에포크 수
    batch_size=32, # 배치 크기
    validation_data=val_dataset,  #검증 할 데이터 지정
    callbacks=[early_stop]  # 조기종료
)
```

Binary_crossentropy: 확률 분포와 실제 레이블(0 또는 1) 간의 불일치를 로그 함수로 계산

Adam: 딥러닝에서 가장 널리 쓰이는 최적화 알고리즘, 학습률과 모멘텀 기법을 결합한 방식

모델링 처리 코드

이미지 브랜치 코드

```
# 이미지 입력 정의
# 3: 이미지 색상 채널
img_input = keras.Input(shape=(224,224,3),
name="image_input")


# EfficientNetB0 모델 로드 (사전 학습된 가중치 사용)
base_cnn = keras.applications.EfficientNetB0(
    include_top=False,     #특징추출기_사용
    weights='imagenet',      #ImageNet_가중치
   input_tensor=img_input
)


# 이미지 특징 추출 파이프라인
# 차원 축소
x_img =
keras.layers.GlobalAveragePooling2D()(base_cnn.output)
# 정규화
x_img = keras.layers.BatchNormalization()(x_img)
# 과적합 방지
x_img = keras.layers.Dropout(0.3)(x_img)
```

텍스트 브랜치 코드

```
# 텍스트 입력 정의
text_input = keras.Input(shape=(max_seq_len,),
name="text_input") vocab_size = len(tokenizer.word_index) +
1 # 단어 사전 크기


# 텍스트 특징 추출 파이프라인
# 단어 임베딩 128차원의 숫자 벡터
x_text = keras.layers.Embedding(vocab_size, 128,
mask_zero=True)(text_input)
# 문맥 이해 256차원의 벡터로 출력
x_text = keras.layers.LSTM(256)(x_text)
# 정규화
x_text = keras.layers.BatchNormalization()(x_text)
# 과적합 방지
x_text = keras.layers.Dropout(0.3)(x_text)
```

특징 융합 및 최종 출력 코드

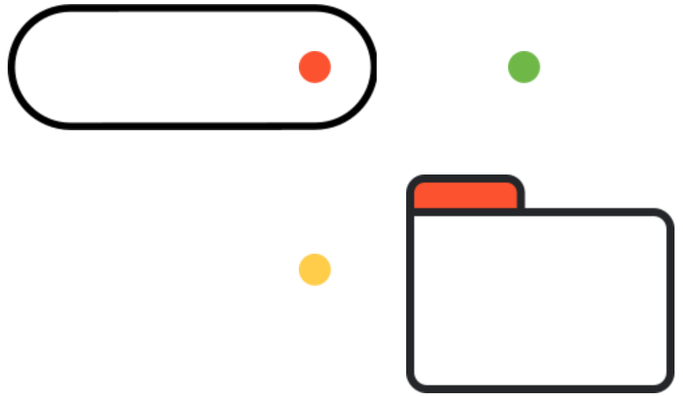
```
# 1. 이미지와 텍스트 특징 연결(Concatenate)
x = keras.layers.concatenate([x_img, x_text])


# 2. 융합된 특징을 기반으로 최종 분류
# 256 뉴런과 활성화 함수 사용하여 비선형성 추가
x = keras.layers.Dense(256, activation='relu')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Dropout(0.3)(x)


# 3. 최종 출력 (이진 분류: 매칭 O/X)
# 입력 값을 0과 1 사이의 실수로 변환
output = keras.layers.Dense(1, activation='sigmoid',
name="output")(x)


# 완성된 모델 정의
 model = keras.Model(inputs=[img_input,text_input],
outputs=output)
```

04

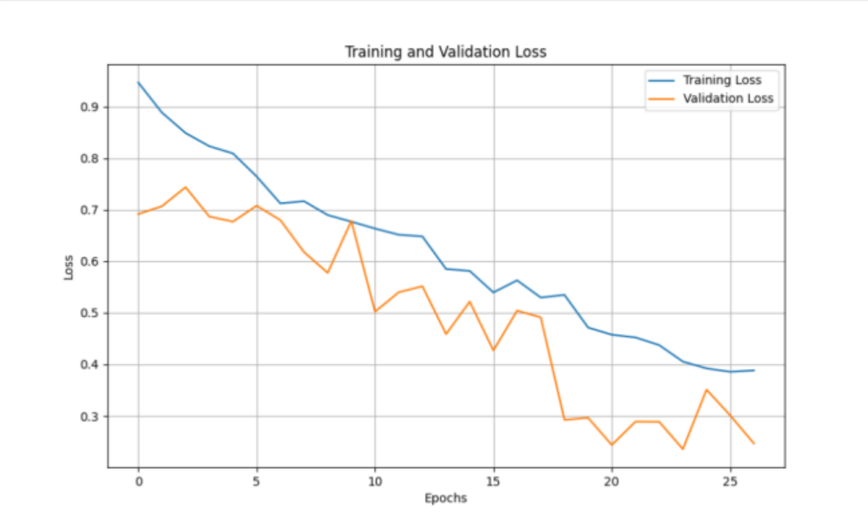
학습 결과 및 성능 평가 (Results & Evaluation)

1.학습 과정 시각화

2. 학습 모델 테스트 결과



학습 과정 시각화 - 손실(Loss) 그래프

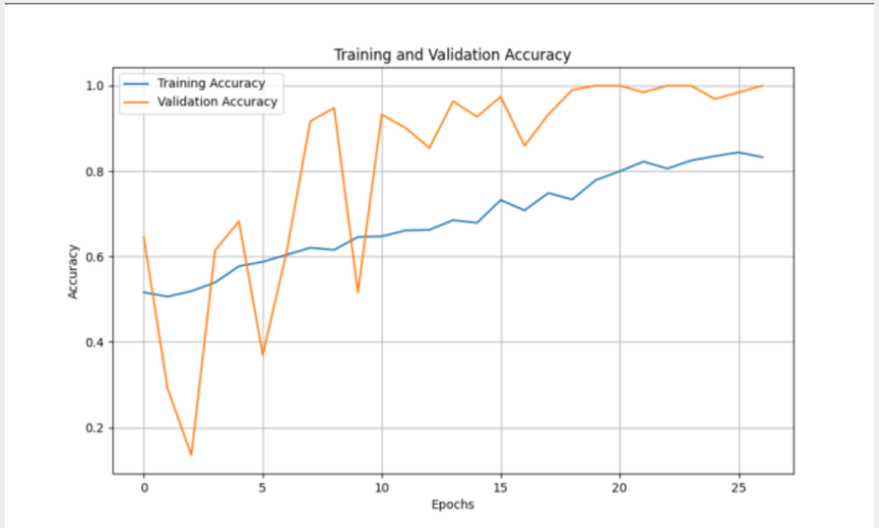


- 훈련 손실: 점진적 감소 후 0.4 근처 안정화
- 검증 손실: 0.25까지 점진적 감소
- 손실 추세: 두 손실 모두 감소

훈련 손실: 학습에 사용하는 데이터(훈련 데이터셋)에 대해 계산된 손실값

검증 손실: 본 적 없는 별도의 데이터(검증 데이터셋)에 대해 계산된 손실값

학습 과정 시각화 - 정확(Accuracy) 그래프



- 훈련 정확도: 지속 상승 → 정확도: 약 83%
- 검증 정확도: 변동 있음 → 최종 100% 근접
- 정확도 추세: 검증 정확도가 훈련 정확도를 상회

훈련 정확도: 학습에 직접 사용하는 데이터(훈련 데이터셋)에 대한 정답률

검증 정확도: 별도의 데이터(검증 데이터셋)에 대한 정답률

학습 모델 테스트 결과

매치 데이터 이미지

출력 결과

- 이미지와 캡션 입력에 대해 약 87.5% 확률로 긍정 클래스 예측
- 멀티모달 입력의 연관성 판단에 성공적
- 실제 응용 서비스(매칭, 추천 등) 가능성 높음

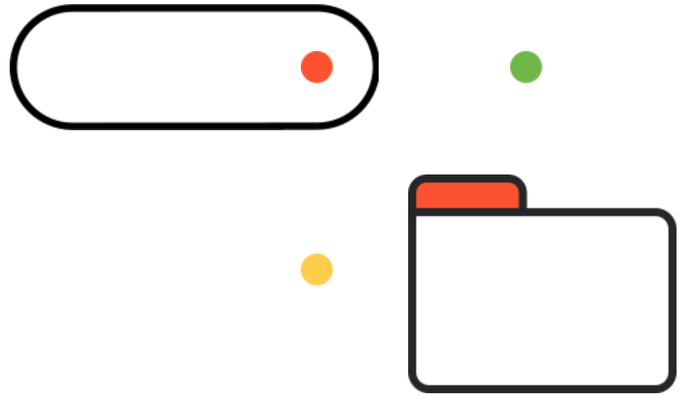
05

결론 및 향후 계획 (Conclusion & Future Work)

1. 프로젝트 요약 및 의의

2. 한계점

3. 향후 계획



프로젝트 요약 및 의의

- 멀티모달 AI 구현
  (이미지 + 텍스트)
- 멀티모달 학습 가능성 확인
- 이미지 검색, 자동 태깅 등
  응용 기반 마련

한계점

- 데이터 양 제한
- 일반화 성능 개선 필요
- 라벨 불균형

향후 계획

- 모델이 얼마나 잘 맞추는지와,
  분류 기준이 되는 확률 값을 다시
  점검해서, 결과를 더 믿을 수
  있도록 추가로 확인
- 프로젝트 적용 및
  피드백 반영

일반화: 새로운 데이터에 대해서도 잘 맞는 모델로 개선

라벨 불균형: 클래스(라벨)의 데이터 개수가 서로 크게 차이 나는 상태