1P : 프로젝트 소개, 팀원 소개

2P : 목차 제목만 읽기

3P :

혹시 여러분들은 평소에 맛집을 어떻게 찾으시나요? 보통 특별한 관심이 없는 분들은 이렇게 인스타나 네이버 등에 맛집을 검색하는 분들이 많습니다. 이러한 경우에는 광고성 컨텐츠로 인해 맛집을 제대로 찾지 못하는 경우가 빈번합니다. 조금 더 꼼꼼하게 알아보시는 분들은 이렇게 특정 맛집의 개별피드를 하나씩 확인하고, 방문자 리뷰 등을 통해 진짜 맛집인지를 따져봅니다. 더 나아가서 맛집에 관심이 많은 분들은 특정 어플을 사용하거나, 찐 맛집을 찾는 본인만의 방법을 가지고 있습니다. 하지만 그럼에도 불구하고 정보를 공개하지 않는, 맛있어 보이는 사진만 공개하는 맛집 초고수들은 따라갈 수 없죠.

4P :

요즘들어 특히 심해지는 혐핫 신드롬이란 것이 있습니다. 매체가 발달하면서 유명세와 함께 그 가치가 떨어지는 것이 싫은 사람들이 늘어나는 현상인데, 그 때문에 SNS에 자랑은 하지만, 여기가 어디인지는 표현하지 않는 글이 많다고 합니다. 이렇게 해시태그에 알려주지 않는다는 얘기까지 쓰는 경우들도 많다는데, 그래서 저희 팀은 이런 사진만 보고도 맛집을 알아낼 수 있는 방법이 없을까? 라는 생각에 이미지 분석 기술을 이용한 프로젝트를 시작하였습니다.

5P:

프로젝트는 다음과 같이 진행하였습니다. 우선 학습할 음식 사진들을 모으는 Data Collection을 진행하고, 여러 각도로 촬영된 음식사진을 구분해야하기 때문에 Data Augmentation을 통해 모델 성능을 높이고, 이미지 분류로 개발된 고성능의 Pretrained Model을 활용한 Transfer Learning을 진행한 뒤 마지막으로 앙상블을 통해 프로젝트를 완료했습니다.

6P:

프로젝트를 설명하기에 앞서, 팀 프로젝트로 진행하기 위한 깃허브에 대해 잠깐 설명드리고자 합니다. 저희가 사내에서는 ECM과 같은 시스템을 통해 서로 문서를 공유하여 작업을 진행할 수 있었지만, 이번 프로젝트를 할 때 코드나 데이터를 공유하는 방법을 찾아야만 했습니다. 때문에 저희는 깃허브를 이용하였는데요, 여기서 Git이란, 파일의 변경사항을 관리하는 시스템입니다. 이 변경사항들을 GitHub라는 허브를 통해 업로드하고 관리할 수 있으며, 이 시스템은 개인 PC나 Docker에 폴더를 지정하여 관리하는 방식으로 굉장히 편리하게 파일을 공유할 수 있습니다.

7P:

프로젝트 시작을 위한 데이터 수집 단계입니다. 저희는 인터넷에 있는 음식점에서 실제로 사람들이 찍어 올린 사진들을 수집했습니다. 웹 사이트는 Static과 Dynamic의 두 가지 종류가 존재합니다. Static은 웹 사이트가 변화하지 않는 형태이지만 Dynamic은 사용자가 마우스를 올려놓는 등의 행동을 하였을 때 표시하는 정보가 달라집니다. 실제로 저희가 데이터를 수집하기 위해 사용한 인스타그램, 구글, 네이버, 다음은 기본적으로 Static한 웹 사이트지만, 이렇게 코드를 입력하여 사이트에 접속하였을 때 스크랩 할 수 있는 이미지는 굉장히 제한적이라, 스크롤을 내려 동적으로 데이터를 수집해야 했습니다. 동적 Scrap 기술을 활용하여 보이는 영상과 같이 자동으로 원하는 음식점의 수많은 사진들을 수집하는 코드를 구현했습니다.

8P:

하지만 저희가 수집한 사진들은 실제로 새로 찍힌 사진에 대한 이미지 분석을 하는데에는 한계가 있습니다. 사람들은 정해진 각도나 위치에서 사진을 찍지 않기 때문에, 그러한 상황들을 학습하기 위한 데이터 Augmentation이 필요합니다. 여기 보이는 고양이는 저희 팀의 문박사님의 고양이인데요, 이렇게 수집된 기존 사진에서 회전을 하고, 모델에 맞게 Crop하는 방법으로 여러 각도에서의 데이터를 만들어낼 수 있었습니다. 이 Data Augmentation 과정은, Train Test Split 이후에 진행했습니다. 저희는 음식점 Class 당 최소 300장 이상의 사진을 수집하여 각 Class별로 Train 180장, Valid 60장, Test 60장씩 나눈 다음, Augmentation을 이용하여 이미지를 늘린 뒤, 뒤에 사용할 Tensorflow 이미지 데이터셋을 위한 flow\_from\_directory를 통해 class별로 데이터를 나누었습니다. 이렇게 생성된 데이터는 총 Train 52020개, Valid와 Test 17340개가 수집되었습니다.

9P:

이렇게 수집된 데이터를 학습시킬 CNN 모델은 Transfer Learning을 활용하였습니다. 총 4개의 모델을 사용했으며, 그 중 성능이 가장 잘 나왔던 2개의 모델을 소개하고자 합니다.

EfficientNetV2는 21년에 나온 EfficientNet에서 Train 효율을 높인 모델입니다. 큰 이미지에 대한 학습을 위해 EfficientNet에서는 이미지 크기에 따라 정규화를 하고, Depthwise Convolution의 느린 학습속도를 개선하기 위해 Fused convolution을 활용하며, 기존 EfficientNet에서 사용하는 compound scaling을 non-uniform하게 이용하여 학습속도를 개선한 모델입니다. 여기 보이는 표와 같이 다른 무거운 모델들과 비교하였을 때, 성능과 학습속도가 모두 좋은 모델임을 알 수 있습니다.

다음은 Xception입니다. 이 모델 Inception 모듈을 개선한 모델입니다. Depthwise Convolution을 Pointwise Convolution을 먼저 한 뒤 진행함으로써 cross-channel과 spatial에 대한 correlation을 독립적으로 계산하기 좋은 구조로 만들어 Depthwise Separable Convolution을 구현합니다.

10P:

이러한 Transfer Learning을 총 4개 모델에 17class에 대해 진행하였습니다. 각 모델은 전체 Weight를 업데이트하고, Class 개수에 맞춰 FC Layer를 수정하였습니다. 또한 가장 마지막 Layer 뒤에는 Softmax를 붙여 Voting을 하는데 사용하였습니다. 저희가 사용한 최종 모델은 Weighted Voting을 활용한 앙상블 모델입니다. 이렇게 이미지가 들어오면 각 모델에서 Softmax로 확률값을 내고, 이 값들에 각 모델이 가진 Valid Accuracy를 곱해서 Voting하여 이미지를 분류합니다.

11P:

앙상블 Voting에 대해서 잠깐 말씀드리자면, Hard Voting, Soft Voting, Weighted Voting 세 가지로 나눌 수 있습니다. Hard Voting의 경우, 각 모델이 내놓은 결과에 대해 1표씩 투표를 하여 결정하는 방법이고, Soft Voting은 각 모델에서 내놓은 Class별 확률값을 더해서 Voting을 합니다. 저희가 프로젝트에 사용한 Weighted Voting의 경우, 각 모델에서 내놓은 확률값에 각 모델의 성능을 가중치로 두어 Voting을 하는 방식입니다. 여기서의 가중치는 모델의 성능이 아닌 다른 가중치도 사용될 수 있습니다.