<Transfer Learning>

= 이미 방대한 데이터를 이용해 훈련된 모델을 이용해, 원래 학습된 목적과는 다른 분류/예측에 사용하는 것

예를 들면, ImageNet이라는 대회에서, 방대한 데이터를 이용해 1000개의 결과값으로 분류하는 학습 과정을 이미 마친 MobileNet을 이용하여, 고양이와 강아지 사진을 주면 고양이인지 강아지인지 분류하는 것.

>> 이미 학습된 모델을 사용하므로, 훈련 시간도 빨라지고, 정확도도 훨씬 높아짐!

주의할 점

1. Output Layer를 교체해야 함

이미 훈련된 모델 MobileNet은 1000개의 결과값으로 분류하지만, 우리가 분류하고자 하는 결과의 개수는 2개 -> Output layer를 떼내고, 대신 유닛 개수가 2개인 dense layer를 추가하기

\*\* Tensorflow Hub에는 이미 훈련된 모델에서 Output layer만 떼낸 모델이 많이 저장되어 있음!

1. Input Size를 맞추어야 함

이미 훈련된 모델 mobileNet은 224 x 224 사이즈의 이미지로 훈련하였으므로, 우리가 사용할 때에도 같은 크기의 이미지를 인풋값으로 넣어줘야 함

1. 이미 훈련된 부분은 Freeze해야 함

모델에서 이미 훈련된 부분의 weight, bias는 바꿀 필요가 없음! 우리가 새로 추가한 output layer만 바꾸면 됨

그런데, 만약 이미 훈련된 부분까지 포함해서 모델을 다시 훈련시킬 경우, 처음에는output layer가 갖는 weight는 랜덤으로 초기값 설정되어 loss가 굉장히 큰 값을 갖게 되고, 그 때문에 이미 훈련을 마친 부분의 weight까지 크게 바뀌게 됨

\*\* Output layer만 떼낸 모델 = Feature extractor! (마지막 레이어에는 이미지를 분류하는 데 필요한 특징들만 담겨있기 때문! 여기서 분류만 하지 않은 것)

\*\*Convolution 과정을 그림으로 보기

<https://towardsdatascience.com/understanding-your-convolution-network-with-visualizations-a4883441533b>

Convolution 진행된 과정을 살펴보면, 각 필터의 weight가 조정되면서 몇 개는 그림의 윤곽선을 찾는 역할을 하고, 몇 개는 그림의 위치를 찾는 역할을 하고, 몇 개는 꽃의 중심을 찾거나 특정 부분만을 인식한다는 걸 볼 수 있음!

특히 초기 Convolution layer에서 두드러지게 나타나는데, 이는

“초기 Convolution layer는 이미지의 전체적인 특징을 잡고, 뒤쪽 Convolution layer로 갈수록 훈련 데이터에서만 볼 수 있는 세부적인 특징들을 기억하기 때문”

>> transfer learning을 할 때, 초기 Conv레이어의 weight는 고정시키고, 뒤쪽 Conv레이어의 weight는 훈련시키는 것이 좋음!

(초기 Conv레이어가 인식하는 그림의 전체적인 특징은 방대한 데이터를 가질수록 성능이 좋아지지만, 실제 세부적인 분류를 하는 기준이 되는 뒤쪽 Conv레이어는 우리가 분류할 이미지에 특화된 것이 좋기 때문)