<Competition 학습 과정에서의 교훈>

Competition 진행 1 일차

진행한 상황

1. 직접 데이터를 탐색하여 EDA 진행 -> 데이터의 구조는 Incorrect, correct, None 으로 구성되어 있으며 더 자세한 구성은 대외비
2. Dataset.ipynb 를 통해 Train Data 를 분류하는 과정을 생각
3. Pre-trained 된 VGG Network 를 이용해 Image 분류

아쉬운 점 & 개선할 점

1. EDA 진행 사항을 일일이 손으로 하였는데 이것을 더 시각화 툴을 이용해서 진행해보고 싶다
2. 처음에 생각을 잘 못해서 Train Data 에 대한 전처리에서 삽질을 많이 했다 하지만 전처리는 내일 강의에서 배울 내용이므로 순차적으로 강의를 따라가 모델 성능을 차근차근 올리자고 생각했다. 그랬기에 오늘 EDA 에 집중하지 않고 시간을 날린 것을 반성하게 되었다.

Competition 진행 3일차

진행한 상황

1. 전날 오류가 생겼던 Model 부분에서 Pre - trained 된 Resnet 을 불러와 현재 주어진 Dataset 과의 연결을 성공했습니다.
2. Dataset 을 만드는 과정에서 Random Crop, Resize, Flip, Center Crop 등 다양한 기법을 적용했습니다.
3. 기존 Data 를 80% : 20%의 비율로 Train Set 과 Valid Set 으로 구분했습니다.
4. Model 이 문법 오류 없이 학습을 하고 정확도와 loss 를 계산하는 과정을 구현했습니다.

아쉬운 점 & 개선할 점

1. Model 이 모든 Data 에 대해 한 개의 Labeling 만 한 것을 확인했습니다. 이것이 학습 과정에서의 문제인지 Dataset 을 만드는 과정에서 생긴 문제이지 파악하지 못해 진행하지 못 한 점이 아쉽습니다.
2. Dataset 을 만드는 과정에서 밝기 조절, CutMix 등 다양한 기법을 생각했지만 앞선 오류를 수정하느라 진행하지 못한 점이 아쉽습니다.

Competition 진행 5일차

진행한 상황

1. 전날 오류가 생겼던 모든 Data 에 대해 한 개의 Label 로만 예측하는 원인을 파악했습니다. 이는 Image 에 Transform 을 적용하는 과정에서 ToPILImage()를 사용해 생긴 문제로 파생되어 Dataset 안에서 transform 할 때 적용한 np.uint8 등을 제거하면서 해결해 Model 예측이 나오게 했습니다.
2. Tranfer Learning 을 통해 기존 Pretrained 된 Vgg11, Resnet18, Resnet50, Resnet101 을 사용했습니다. 그 결과 VGG 와 Resnet 에서는 유의미한 차이가 있는 것을 발견했고 Resnet 사이에서는 큰 차이가 없었습니다. Resnet101 을 사용해 정확도를 77%까지 올렸습니다.
3. Dataset 을 만드는 과정에서 RandomCrop 과 ColorJitter 를 이용해 성능을 개선했습니다.
4. Normalize 에서 기존에 존재했던 대중적인 값을 버리고 이번 dataset 에 맞는 값을 사용했습니다.]
5. Learning rate 를 고려한 결과 0.01 은 너무 빨랐으며 1e-05 는 느렸기에 1e-04 를 사용했습니다.

아쉬운 점 & 개선할 점

1. Pretrained Model 과 지금 진행하는 Classification 에 관해서는 Task 가 유사하지 않기에 학습을 하다가 한계가 생겨 성능이 올라가지 않는 현상을 확인했습니다. 또 다른 Model, 또는 Resnet 안을 Dropout 등을 이용해 구조를 바꿔야 할 것입니다.
2. 19000 의 Train Data 이므로 충분하지 않아 이 Data 에 Overfitting 이 발생하는 현상이 발견되었습니다. Data 를 늘릴 고민을 해야 될 것 같습니다.
3. Age 에 관해 50 대와 60 대는 큰 차이가 없지만 이것을 다른 Label 로 구분해야 해 생기는 예측 오류가 있었습니다. 나이 구분을 잘하기 위한 노력을 어떻게 해야 할지 고민해 봐야 합니다.

Competition 진행 8일차

진행한 상황

1. OpenCv 에 있는 Age Detection 을 적용해 보았습니다. Age 만을 뽑아내는 것은 성공했지만 기존 Opencv 에 있는 Model 이 서양인에 맞춰져 있다는 점 그리고

Model 자체적으로 성능이 좋지 않았기에 젊은 나이에서는 예측이 맞았지만 나이가 들수록 오류가 심한 것을 확인할 수 있었습니다.

2. 5-KFold 를 적용해 본 뒤 성능 향상이 가능성이 보여 Stratified KFold 를 적용해 모델을 구성했습니다. Stratified KFold 특성 상 Train 에 Ground Truth 도 같이 표시해 주어야 하기에 Dataset 을 만드는 코드를 수정했습니다.
3. SGD 와 Adam 의 차이점을 알 수 없기에 Adam 으로 진행했습니다.

아쉬운 점 & 개선할 점

1. KFold 적용을 통해 Overfitting 이 내려가 Validation 성능이 올라갔지만 Class 불균형으로 인한 F1 score 는 좋지 못한 점이 아쉽습니다.
2. Age Detection 모델을 불러왔지만 성능에 큰 향상이 없었습니다. 전통적으로 Age Detection 은 어려운 문제라고 생각해 개선 방법이 어려운 것 같습니다.
3. Stratified KFold 를 시도해 볼 것입니다.

Wrap Up Report

<기술적인 도전>

LB 점수 0.7038, 123등

검증(Validation) 전략

제공된 Dataset에 대해 Stratified KFold 5를 적용했습니다. 이것을 적용하기 위해 Dataset Class에서 해당 데이터의 label도 따로 모아 후에 학습할 때 넘겨줬습니다.

사용한 모델 아키텍처 및 하이퍼 파라미터

1. 아키텍처: Resnet101 & Resnet 50
    A. LB 점수 : 0.7100
    B. training time augmentation

        RandomCrop((330,220)), ColorJitter(brightness=0.1, contrast=0.2, saturation=0, hue=0), Normalize([0.560, 0.524, 0.501], [0.233, 0.243, 0.246])

C. img_size = 330 X 220

D. 추가 시도

- Batch Size를 처음에는 256으로 시작했다가 최종적으로 32를 적용해 Local minimum에 빠지는 문제를 해결해 Accuracy를 올렸습니다.
- Learning rate를 0.01, 1e-04, 1e-05등 다양하게 적용을 해 가장 나은 값이 1e-04라는 것을 알게 되었습니다.
- Optimizer 부분에서 SGD와 Adam을 두고 여러 번 반복 시도를 하였지만 둘의 큰 차이가 없다는 결론을 내렸습니다.

2. OpenCV Age Detection

A. 세가지 분류에 대해 Mask Detection, Gender는 높은 확률로 맞췄지만 Age에 대해 성능이 좋지 않았던 점에서 시도하게 되었습니다.

B. OpenCV에 있는 Age Detection을 적용해 Images의 나이를 예측했습니다. 이 과정은 먼저 MTCNN에 있는 Pretrained 모델을 불러와 Face Detection을 통해 얼굴의 위치를 찾게 됩니다.

C. 찾은 얼굴의 위치를 Box 배열로 전달해 OpenCV는 전달받은 Face Image를 Age_net을 이용해 나이를 예측합니다.

D. age_list = ['(0 ~ 2)','(4 ~ 6)','(8 ~ 12)','(15 ~ 20)', '(25 ~ 32)','(38 ~ 43)','(48 ~ 53)','(60 ~ 100)'] 나이 분포 List는 이렇게 되어있으므로 이 Labeling을 적절히 이용해 이번 Classification Model에 적용했습니다.

E. 하지만 이 Model은 서양 얼굴에 대해 예측을 잘하지만 지금 분류하는 동양 얼굴에 대해서는 실제 나이보다 10~20살 정도 어리게 예측하는 문제가 있으며 정확도가 높지 않았습니다.

3. 앙상블(Ensemble) 방법

각 KFold 모델 마다 나온 예측 Label을 K로 나눠 ans에 저장한 뒤 최종적으로 Test가 끝나면 해당 Image에 대한 예측 분포에서 Argmax를 이용한 가장 큰 값을 정답으로 사용하였습니다.

4. 시도했으나 잘 되지 않았던 것들

1. Overfitting을 피하기 위해 Label Smoothing을 적용해 보았으나 성능 개선이 좋지 않았습니다.
2. Random Augmentation을 하고 싶어 적용을 해보았으나 적절한 각 Augmentation에 대한 확률 분포를 알 수 없어 개선이 되지 않았습니다.
3. Resnet Fully Connect 부분에서 Dropout을 적용해 보았으나 실질적인 성능 개선은 이뤄지지 않았습니다.
4. 처음에 Pretrained Model을 쓰고 싶지 않아 혼자서 스스로 VGG11을 구현해 학습을 시켜봤으나 Data와 Model 양측 문제로 성능이 나오지 않았습니다.
5. 수평, 수직으로 뒤집기와 Resize후 CenterCrop 등 다양한 방식을 시도하였지만 눈에 띄는 성과는 없었습니다.