<코스포 x 데이콘 자동차 충돌 분석 AI경진대회>

20196783 좌대현

Abstract

본 대회는 불균형이 심한 데이터였고 불균형한 데이터 예측까지 잘 해야 높은 점수가 나오는 Macro F1 score를 평가지표로 사용하였다. 이에 따라 불균형한 데이터 속에서도 학습을 잘하도록 다음과 같은 방법을 적용했다.

1. Task 별 모델 학습(task 별로 예측하여 합치는 방식)
2. 데이터 증강
3. Stratified K-fold
4. 손실함수 클래스 별 가중치 부여
5. 동영상 분류를 이미지 분류로 변환 적용
6. Semi-supervised learning
7. DCGAN을 활용한 이미지 생성

그 결과 Semi-supervised learning, GAN은 성공적인 결과를 내지는 못했지만 다양한 방법들을 통해 기존 Baseline score 0.19/0.21에서 0.68/0.68까지 성능을 끌어올릴 수 있었다.

1. Task

본 대회는 <코스포 x 데이콘 자동차 충돌 분석 AI경진대회>로 블랙박스 영상을 보고 충돌 여부, 날씨, 시간을 분류하는 video classification 문제이다. 총 label은 13개이지만 각 라벨은 crash, ego-Involve, weather, timing이라는 4개의 항목이 합쳐진 결과이며 다음과 같다.

crash	ego-Involve	weather	timing	label
No	-	-	-	0
Yes	Yes	Normal	Day	1
Yes	Yes	Normal	Night	2
Yes	Yes	Snowy	Day	3
Yes	Yes	Snowy	Night	4
Yes	Yes	Rainy	Day	5
Yes	Yes	Rainy	Night	6
Yes	No	Normal	Day	7
Yes	No	Normal	Night	8
Yes	No	Snowy	Day	9
Yes	No	Snowy	Night	10
Yes	No	Rainy	Day	11
Yes	No	Rainy	Night	12

대회 평가지표는 Macro F1 score로 각 클래스마다 F1 score 계산 후 평균내는 것이다. 따라서 정확도와 달리 빈도가 낮은 데이터에 대해서도 예측을 잘해야 한다. 이 task를 위해 어떤 모델을 쓰고 어떤 기법을 적용했는지 살펴보겠다.

2. Data

먼저 대회 데이터는 모두 차량 안에서 찍힌 블랙박스 영상 데이터로 초당 10프레임, 총 5초로 50프레임으로 이루어져 있다. Label은 총 13개로 이루어져 있지만 label의 분포는 매우 불균형하다.

```
train_df['label'].value_counts()
```

```
label
0     1783
1      318
7      317
3       78
2       51
9       34
11      33
8       30
5       28
4       13
12       6
10       4
6        3
Name: count, dtype: int64
```

또한 영상 확인 결과, 잘못 지정된 label이 많아 직접 잘못 지정된 label에 대해 다시 label을 지정해주었다. 몇 개의 영상은 애매한 듯 보여 삭제하여 모델의 일반화 성능을 올리고자 하였다.

3. Train process

사용 모델은 아래와 같다.

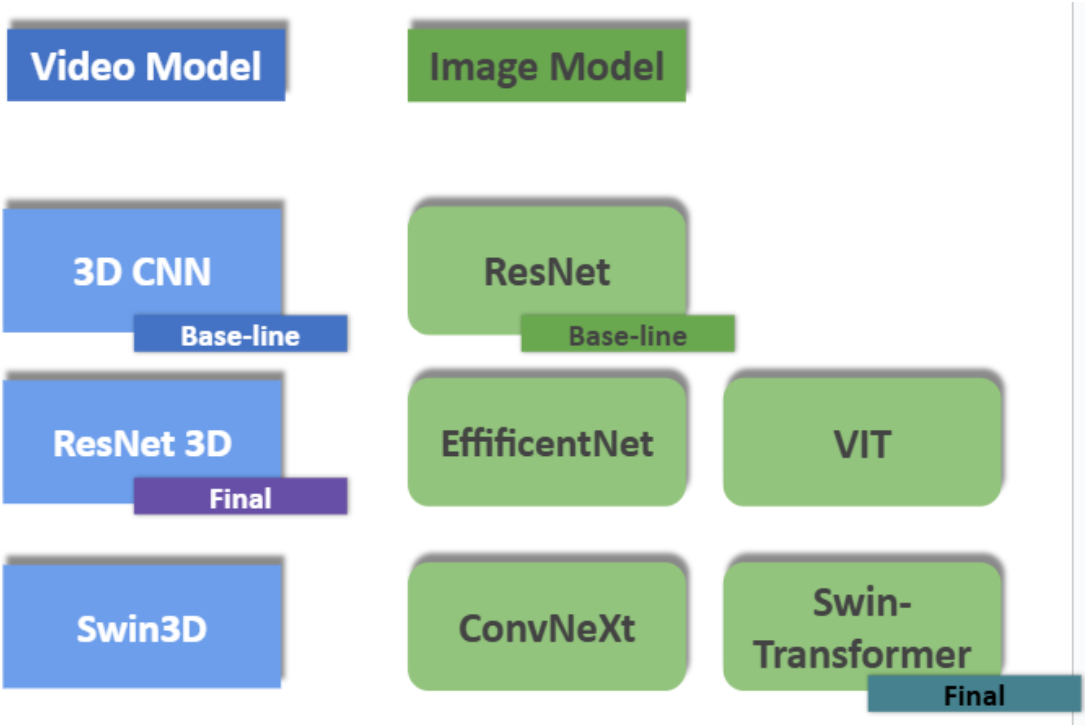

Image Model을 쓰는 이유와 방법은 추후 설명하겠지만 4개의 crash, ego-Involve, weather, timing 각각에 대해 예측하였지만 weather의 경우 성능이 낮기도 하였고 단편적인 프레임만을 보고도 예측하는데 무리가 없을 것이라 생각하여 weather에 Image Model을 사용하였다.

3-1. Base Model: 3D CNN

```python
class BaseModel(nn.Module):
    def __init__(self, num_classes=13):
        super(BaseModel, self).__init__()
        self.feature_extract = nn.Sequential(
            nn.Conv3d(3, 8, (1, 3, 3)),
            nn.ReLU(),
            nn.BatchNorm3d(8),
            nn.MaxPool3d(2),
            nn.Conv3d(8, 32, (1, 2, 2)),
            nn.ReLU(),
            nn.BatchNorm3d(32),
            nn.MaxPool3d(2),
            nn.Conv3d(32, 64, (1, 2, 2)),
            nn.ReLU(),
            nn.BatchNorm3d(64),
            nn.MaxPool3d(2),
            nn.Conv3d(64, 128, (1, 2, 2)),
            nn.ReLU(),
            nn.BatchNorm3d(128),
            nn.MaxPool3d((3, 7, 7)),
        )
        self.classifier = nn.Linear(1024, num_classes)
```

베이스라인 모델은 Conv3d 4개가 포함된 feature extractor와 Linear층 1개의 classifier로 이루어져 있다. Baseline Model을 13개 label에 대해 학습 후 제출한 결과 0.19/0.21로 좋지 않은 성능을 보였다. 모델이 단순하기도 하지만 12번, 10번, 6번 label의 경우 학습 data가 10개도 되지 않아 학습이 제대로 이루어지기 힘든 것으로 보인다. 실제로 rainy이면서 night인 학습 영상은 12개밖에 없었지만 4개의 task로 나눠 학습한다면 rinay에 대해 83개의 영상, night에 대해 105개의 영상을 학습할 수 있게 되고 각 모델은 좀 더 세분화된 task로 시간을 걸리지만 더 좋은 성능을 보일 것이라 생각하였다.

3-2 Method 1: 4개 task로 나눠 각각 모델 학습

Test F1: 0.484 / 0.590

각 task 성능 지표로는 모든 class에 대해 잘 맞춰야 하므로 Val F1 score를 사용하였다.

- Crash
  ResNet 3D-18을 사용했는데 Val F1 0.99326으로 거의 대부분 잘 맞추는 것으로 나타났다.

- Ego-Involve

  ResNet 3D-18과 Swin 3D 모델을 사용해봤고 ResNet 3D-18 모델의 Val F1이 0.90359로 더 높은 성능을 보였다.

- Weather

  Weather의 경우 다음과 같이 0(normal)이 대부분이고 1(snow), 2(rainy)는 거의 없다.

```
label
0    634
1    171
2     83
Name: count, dtype: int64
```

  ResNet 3D-18과 Swin 3D 모델을 학습해봤을 때 Val f1이 0.78857이 나왔고 모델의 더 나은 일반화를 위해 Horizontalflip, Rotation, ColorJitter, Crop 등 다양한 증강을 적용하였다. 또한 데이터 class 분포와 동일하게 fold를 나누는 Stratified K-fold를 사용하여 데이터 불균형 문제를 일부 해결하여 Val F1을 0.88831까지 끌어올렸다. 실험 과정에서 모델의 classifier만 학습하거나 기존 classifier에서 층을 더 붙이는 등 여러 시도를 해봤지만 성능에 큰 변화는 없었고 기존 사전학습모델을 그대로 사용해 전이학습을 진행하였다.

- Timing

  timing에도 마찬가지로 ResNet 3D-18과 Swin 3D 모델을 학습하였고 마찬가지로 Stratified K-fold와 다양한 증강을 적용하여 ResNet 3D-18 모델로 Val F1 0.95678을 달성하였다. Timing은 day(낮), night(밤)으로 이루어진 이진분류로 밝기로 어렵지 않게 모델이 분류할 수 있는 문제이다. 모델 학습 후 직접 test 영상에 대해 예측한 것을 50개 정도 확인 결과 모두 잘 판단하는 것으로 보였다. 하지만 낮, 밤을 나누는 경계 부분은 애매하여 사람이 판단하기도 애매하였고 이 부분은 모델 역시 학습이 어려운 부분이라 Val F1이 0.95선까지 나온 것으로 보인다.

  Method1을 적용해본 결과 ego-Involve와 weather의 성능이 약간 아쉬웠고 ego-Involve 성능을 올리기 위해 비슷한 task인 crash와 ego를 묶어 학습하였다.

3-2 Method 2: 3개 task로 나눠 각각 모델 학습

Test F1: 0.559 / 0.628

Weather와 timing에 대해서는 Method 1 모델과 동일하게 사용

- crash+ego
  crash와 ego를 묶어 학습하고 튜닝하였더니 Val F1이 0.96029로 좋은 성능을 보였고 실제 제출 시에도 4개의 task로 나누는 것 보다 3개의 task로 나누는 것이 더 좋은 성능을 보였다.

3-3 Method 3: weather를 Image classification으로 변환

Test F1: 0.725 / 0.607

Weather에서의 성능 역시 아쉬웠기에 다른 방법을 생각하여 보았다. Weather는 한 프레임만 보고도 분류가 가능할 것이라고 생각하였고 Image classification으로 변환하였다.

- Weather
  Image classification을 위해 영상에서 프레임을 추출해 저장하였고 모든 프레임을 학습에 사용시 프레임간 간격이 0.1초로 너무 비슷하다고 생각하여 최대한 간격을 두고 0번째, 25번째, 49번째 프레임을 학습에 사용하였다. 각 프레임간 간격은 2.5초로 이미지가 완전 같지 않고 데이터가 증강되는 효과도 있었다.

  또한 모델의 일반화와 불균형한 데이터에 대해서도 잘 학습할 수 있도록 데이터 증강과 loss function에 class별 weight를 부여해주었다. 최종적으로 증강은 Horizontal Flip, Rotation, ColorJitter, Crop, Random Erase을 적용하였고 weight을 주었을 때 Test F1이 더 잘 나왔다. 아무래도 weather 데이터가 많이 불균형하여 weight를 주었을 때 대회 평가지표인 Macro F1 score가 더 잘 나오는 것 아닐까 생각하였다.

ResNet 50	Data augmentation	Weight	Val F1	Test F1
1st	X	x	0.82641	-
2nd	Horizontal Flip, Rotation, ColorJitter, Crop, Random Erase	x	0.87193	0.467 / 0.575
4th	**2nd** + Vertical Flip - ColorJitter	x	0.82904	-
6th	**2nd** - Random Erase	**o**	0.82786	0.518 / 0.734
7th	**2nd**	**o**	0.86038	0.618 / 0.723

3-4 Semi-supervised learning

이러한 방법들을 적용해 성능이 올랐지만 추가적인 성능 향상을 위해 weather에 대해 분류가 되어있지 않은 label0 데이터를 사용하고자 하였다. Label0은 다음과 같이 crash에 대한 정보가 없기에 성능이 낮은 weather에 잘 활용하면 성능 향상의 여지가 크다고 판단하여 새로운 방법론을 생각해 보았다.

crash	ego-involve	weather	timing	label
No	-	-	-	0

Semi-supervised learning은 구현을 못해 다음과 같은 방법으로 진행했다.

1. label 0에 대해 성능이 가장 좋은 Weather Model로 inference하여 weather에 대한 예측 확률을 뽑는다.

2. 예측된 확률이 0.8 이상인 것들만 분류한다.

3. snow, rainy에 대한 데이터가 적고 학습이 안 되었기에 snow, rainy 데이터만 가져와 사용한다.

2번 threshold를 0.8로 정한 이유는 0.5로 정할 시 애매하거나 오분류된 데이터가 많아질 것으로 예상하였다. 이런 데이터를 추가해 학습할 시 오히려 학습에 혼동을 주어 좋지 않은 결과가 나올 것이라 생각했다.

추가된 데이터 확인 결과 snow에 대해서는 잘 예측했으나 rainy에 대해서는 밤인 경우가 대부분이었다. 때문에 학습에 사용해도 그렇게 성능이 좋아지는 모습은 보이지 않았다. 실패한 이유는 첫째, 1번에서 분류기로 사용한 모델 역시 snow, rainy가 적었기에 snow, rainy에 대해 완전하게 잘 학습됐다고 보기는 힘들었던 것 같다. 둘째, 이미지로 분류하다 보니 rainy의 경우 모델이 비가 내리는 모습을 통해 판단하지 못하고 젖은 땅과 차 전면유리에 맺힌 빗방울을 통해 파악하는 것 같았다. 밤의 경우 땅이 어두운 색깔이라 noraml인지 rainy인지 혼동하는 것 같았고 대부분의 추가된 rainy는 밤이었다. 따라서 추가된 데이터에 대해서도 그대로 사용하지 못하고 직접 확인을 하여 라벨을 재지정하고 학습을 진행해주었다.

3-5 Ensemble

Weather의 경우 데이터가 불균형하고 아래처럼 각 모델마다 0, 1, 2에 대한 비율이 꽤나 차이나는 것을 볼 수 있다. 따라서 모델이 robust하도록 가장 성능 좋은 5개 모델로 Hard voting을 진행해주었다.

```
1 # ResNet 3D-18
2 df1['label'].value_counts()
```

```
label
0    1518
1     188
2      94
Name: count, dtype: int64
```

```
1 # VIT
2 df2['label'].value_counts()
```

```
label
0    807
2    658
1    335
Name: count, dtype: int64
```

```
1 # ConvNeXt
2 df5['label'].value_counts()
```

```
label
0    1036
2     507
1     257
Name: count, dtype: int64
```

```
1 # ResNet50
2 df3['label'].value_counts()
```

```
label
0    1486
1     185
2     129
Name: count, dtype: int64
```

```
# Swin
df4['label'].value_counts()
```

```
label
0    1186
2     390
1     224
Name: count, dtype: int64
```

Hard voting을 적용한 결과 Test F1이 0.60/0.66으로 큰 차이는 없었고 weather에 대해 모델마다 너무 다른 결과가 나와서 합쳐지는게 그렇게 좋아지지 않았나 싶기도 하다. 또한 3개의 crash+ego, weather, timing이 합쳐지다 보니 2개의 task에 대해 맞춰도 1개의 task에서 틀리면 오답이기에 3개의 task가 합쳐지는 과정에서 운적인 요소도 포함되기에 정확한 성능 변화를 알기는 힘들었다.

4. Final Model

결과적으로 crash+ego와 timing에 대해서는 Swin 3D 보다 ResNet 3D-18이 성능이 좋았고, weather에 대해서는 Swin-base를 사용하였고 적용한 기법 및 하이퍼파라미터는 다음과 같다.

- **CFG**
  - **crash+ego**
    - Img_size: 128, batch_size: 4, lr: 1e-5
    - augmentation : Horizontal Flip, Rotate, Brightness, Gauss Noise
  - **weather**
    - Img_size: 224, batch_size: 32, lr: 1e-5
    - augmentation : Horizontal Flip, Rotate, ColorJitter, Crop, RandomErase
    - Label weight
    - Semi-supervised Learning
  - **timing**
    - Img_size: 128, batch_size: 16, lr: 1e-5
    - augmentation : Horizontal Flip, Rotate, Brightness, Gauss Noise
    - k-fold

최종적으로 Public, Private에서 가장 우수한 그리고 Public, Private에서 모두 우수한 모델이 나왔고 사용 모델과 성능은 다음과 같다.

- Public 1st Model
  - Model : ResNet 3D-18 + Swin-base
  - Public : 0.725
  - Private : 0.607
- Private 1st Model
  - Model : ResNet 3D-18 + ResNet50
  - Public : 0.618
  - Private : 0.723
- Robust Model
  - Model : ResNet 3D-18 + Swin-base
  - Public : 0.687
  - Private : 0.688

대회 제출 모델과 달리 실제 우리가 가장 좋다 생각한 모델은 Robust Model로 한쪽에 대해 높은 것 보다 Public, Private 모두에 대해 높은 강건한 모델을 선택하였다.

5. 아쉬운 점

- 원본 학습 데이터가 오분류 된 것이 많아 직접 확인 후 label을 재지정 해줬어도 정확한 분류 기준을 알 수 없기에 성능하락이 있었을 것으로 예상

- weather에 대한 추가 데이터셋을 찾았지만 비슷한 블랙박스 영상이 없었음

- 생성형 모델(DCGAN)을 통해 이미지를 생성해봤지만 학습에 사용할 정도의 이미지가 나오지 않았음

- task를 3개로 나눠서 하였지만 대회 제출은 3개를 합쳐서 제출하고 합쳐진 것에 대한 성능이 나오기에 각 task에 대한 성능을 알 수 없었던 점이 아쉽다.

- 추후 semi-supervised learning을 end-to-end로 완전하게 적용해보고 싶다.

아쉬운 부분도 있지만 결론적으로 965명 중 8등이라는 좋은 성적을 냈다는 점에서 뿌듯하다. 대회를 진행하며 비디오 분류, 이미지 분류의 여러 모델, 증강 기법 등 다양한 기법을 사용하여 성능을 올리고 직접 코드를 구현해봤다는 점에서 성장할 수 있었던 좋은 기회였던 것 같다.

마지막으로 한학기 동안 이미지 딥러닝 수업을 다양한 컴퓨터 비전 분야에 대해 접할 수 있어서 좋았습니다. 수강 인원이 7명으로 적었지만 그 덕분에 잘하는 분들과 함께 소수로 수업들으며 배울 수 있었던 것 같습니다. 이 수업으로 코드 실습, 논문 발표, 프로젝트를 진행하고 직접 적용하며 많이 배운 것 같습니다. 한학기 동안 소수 인원 상대로 열정적으로 강의해주신 교수님께도 감사드립니다.