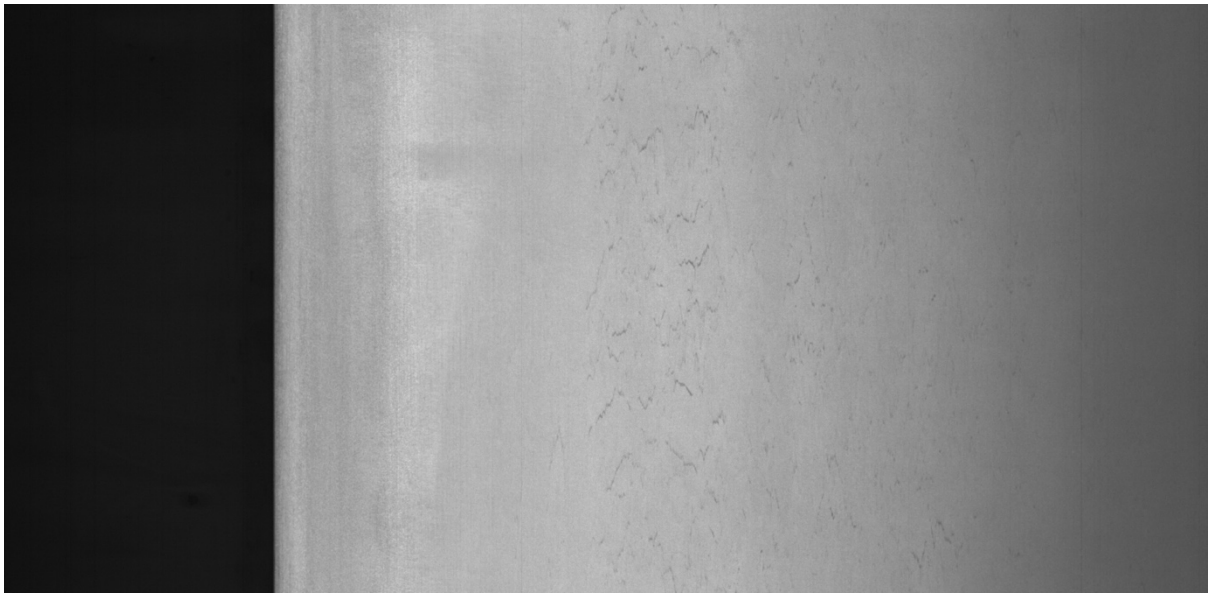
문제개요

금속 표면 데이터의 결함 부분을 잘라내고 라벨링 하여 결함의 종류를 분류하는 모델 학습

데이터 예시1



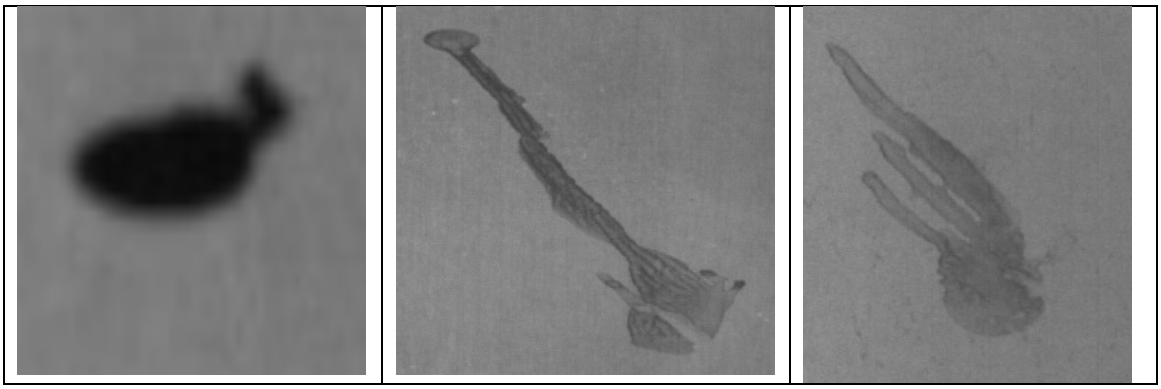
데이터 예시2

```
"img_01_425502900_00018.jpg": {
    "filename": "img_01_425502900_00018.jpg",
    "width": 2048,
    "height": 1000,
    "anno": [
        {
            "label": "crescent_gap",
            "bbox": [
                892,
                801,
                1101,
                1000
            ]
        }
    ]
```


데이터의 면적이 넓어 학습에 방해가 되기 때문에 결함 라벨과 결함 위치 bbox 정보가 담김 JSON 파일을 통해 라벨링과 이미지 크롭의 필요성을 느낌.

1. 데이터 전처리 진행

1-1 전처리가 완료되어 결함 부분만 잘라낸 모습



1-2 도중 발생한 오류

첫 시도에는 이미지 크롭과 같은 전처리 없이 그대로 커스텀 데이터셋에 넣어 라벨과 bbox 정보를 가져오려는 시도를 해보았으나 아래와 같이 한 이미지에 결함 종류가 두 가지 이상 있는 경우가 있어 이 방식은 불가능

하다고 판단.


```
"img_01_425008500_00874.jpg": {
    "filename": "img_01_425008500_00874.jpg",
    "width": 2048,
    "height": 1000,
    "anno": [
        {
            "label": "crescent_gap",
            "bbox": [
                720,
                342,
                1026,
                922
            ]
        },
        {
            "label": "welding_line",
            "bbox": [
                1061,
                621,
                2048,
                698
            ]
        }
```



```
, 108), 28: ('o', 1549), 29: ('i', 12), 30: ('l', 1608), 31: ('_', 194), 32: ('o', 1560), 33: ('i', 275), 34: ('l', 1608), 35: ('_', 383), 36: ('o', 1549), 37: ('i', 23), 38: ('l', 1615), 39: ('_', 66), 40: ('o', 1696), 41: ('i', 1), 42: ('l', 1750), 43: ('_', 45)}
{0: ('w', 71), 1: ('a', 517), 2: ('i', 944), 3: ('s', 629), 4: ('s', 1128), 5: ('i', 27), 6: ('l', 1997), 7: ('k', 620), 8: ('s', 300), 9: ('i', 33), 10: ('l', 1027), 11: ('k', 962), 12: ('s', 164), 13: ('i', 603), 14: ('l', 851), 15: ('k', 919), 16: ('s', 613), 17: ('i', 378), 18: ('l', 1135), 19: ('k', 624), 20: ('o', 1697), 21: ('i', 618), 22: ('l', 1750), 23: ('_', 694), 24: ('i', 1622), 25: ('n', 7), 26: ('c', 2006), 27: ('l', 108), 28: ('o', 1549), 29: ('i', 12), 30: ('l', 1608), 31: ('_', 194), 32: ('o', 1560), 33: ('i', 275), 34: ('l', 1608), 35: ('_', 383), 36: ('o', 1549), 37: ('i', 23), 38: ('l', 1615), 39: ('_', 66), 40: ('o', 1696), 41: ('i', 1), 42: ('l', 1750), 43: ('_', 45)}
{0: ('i', 792), 1: ('n', 812), 2: ('c', 839), 3: ('l', 859), 4: ('s', 1128), 5: ('i', 27), 6: ('l', 1997), 7: ('k', 620), 8: ('s', 300), 9: ('i', 33), 10: ('l', 1027), 11: ('k', 962), 12: ('s', 164), 13: ('i', 603), 14: ('l', 851), 15: ('k', 919), 16: ('s', 613), 17: ('i', 378), 18: ('l', 1135), 19: ('k', 624), 20: ('o', 1697), 21: ('i', 618), 22: ('l', 1750), 23: ('_', 694), 24: ('i', 1622), 25: ('n', 7), 26: ('c', 2006), 27: ('l', 108), 28: ('o', 1549), 29: ('i', 12), 30: ('l', 1608), 31: ('_', 194), 32: ('o', 1560), 33: ('i', 275), 34: ('l', 1608), 35: ('_', 383), 36: ('o', 1549), 37: ('i', 23), 38: ('l', 1615), 39: ('_', 66), 40: ('o', 1696), 41: ('i', 1), 42: ('l', 1750), 43: ('_', 45)}
{0: ('c', 715), 1: ('r', 432), 2: ('e', 972), 3: ('s', 1000), 4: ('w', 899), 5: ('e', 647), 6: ('l', 2048), 7: ('d', 737), 8: ('s', 300), 9: ('i', 33), 10: ('l', 1027), 11: ('k', 962), 12: ('s', 164), 13: ('i', 603), 14: ('l', 851), 15: ('k', 919), 16: ('s', 613), 17: ('i', 378), 18: ('l', 1135), 19: ('k', 624), 20: ('o', 1697), 21: ('i', 618), 22: ('l', 1750), 23: ('_', 694), 24: ('i', 1622), 25: ('n', 7), 26: ('c', 2006), 27: ('l', 108), 28: ('o', 1549), 29: ('i', 12), 30: ('l', 1608), 31: ('_', 194), 32: ('o', 1560), 33: ('i', 275), 34: ('l', 1608), 35: ('_', 383), 36: ('o', 1549), 37: ('i', 23), 38: ('l', 1615), 39: ('_', 66), 40: ('o', 1696), 41: ('i', 1), 42: ('l', 1750), 43: ('_', 45)}
{0: ('w', 487), 1: ('a', 425), 2: ('t', 644), 3: ('e', 671), 4: ('w', 899), 5: ('e', 647), 6: ('l', 2048), 7: ('d', 737), 8: ('s', 300), 9: ('i', 33), 10: ('l', 1027), 11: ('k', 962), 12: ('s', 164), 13: ('i', 603), 14: ('l', 851), 15: ('k', 919), 16: ('s', 613), 17: ('i', 378), 18: ('l', 1135), 19: ('k', 624), 20: ('o', 1697), 21: ('i', 618), 22: ('l', 1750), 23: ('_', 694), 24: ('i', 1622), 25: ('n', 7), 26: ('c', 2006), 27: ('l', 108), 28: ('o', 1549), 29: ('i', 12), 30: ('l', 1608), 31: ('_', 194), 32: ('o', 1560), 33: ('i', 275), 34: ('l', 1608), 35: ('_', 383), 36: ('o', 1549), 37: ('i', 23), 38: ('l', 1615), 39: ('_', 66), 40: ('o', 1696), 41: ('i', 1), 42: ('l', 1750), 43: ('_', 45)}
{0: ('p', 94), 1: ('u', 114), 2: ('n', 217), 3: ('c', 191), 4: ('c', 1981), 5: ('r', 5), 6: ('e', 2048), 7: ('s', 107), 8: ('s', 300), 9: ('i', 33), 10: ('l', 1027), 11: ('k', 962), 12: ('s', 164), 13: ('i', 603), 14: ('l', 851), 15: ('k', 919), 16: ('s', 613), 17: ('i', 378), 18: ('l', 1135), 19: ('k', 624), 20: ('o', 1697), 21: ('i', 618), 22: ('l', 1750), 23: ('_', 694), 24: ('i', 1622), 25: ('n', 7), 26: ('c', 2006), 27: ('l', 108), 28: ('o', 1549), 29: ('i', 12), 30: ('l', 1608), 31: ('_', 194), 32: ('o', 1560), 33: ('i', 275), 34: ('l', 1608), 35: ('_', 383), 36: ('o', 1549), 37: ('i', 23), 38: ('l', 1615), 39: ('_', 66), 40: ('o', 1696), 41: ('i', 1), 42: ('l', 1750), 43: ('_', 45)}
(vtuber) link@gim-yeongjaeui-MacBookAir CODE %
```

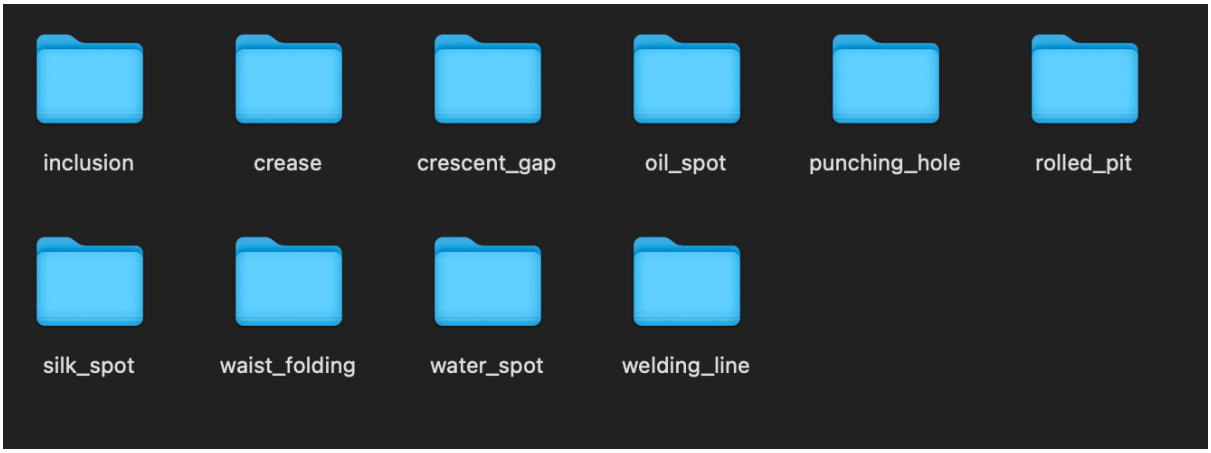
```python
def label_split(file_path):
    dict = {}
    image_path = glob.glob(os.path.join(file_path, '*.jpg'))
    image_name = os.listdir(file_path)
    for i in image_name:
        count = 0
        for anno in metal_info[i]['anno']:
            label = anno.get('label')
            bbox = anno.get('bbox')
            dict[count] = label, bbox
            count += 1
            print(dict)
```

문제 출력 디버그 콘솔 터미널 주석

```
{0: ('waist_folding', [71, 517, 944, 629]), 1: ('silk_spot', [1128, 27, 1997, 620]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
{0: ('inclusion', [792, 812, 839, 859]), 1: ('silk_spot', [1128, 27, 1997, 620]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
{0: ('crescent_gap', [715, 432, 972, 1000]), 1: ('silk_spot', [1128, 27, 1997, 620]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
{0: ('crescent_gap', [715, 432, 972, 1000]), 1: ('welding_line', [899, 647, 2048, 737]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
{0: ('water_spot', [487, 425, 644, 671]), 1: ('welding_line', [899, 647, 2048, 737]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
{0: ('punching_hole', [94, 114, 217, 191]), 1: ('welding_line', [899, 647, 2048, 737]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
{0: ('punching_hole', [94, 114, 217, 191]), 1: ('crescent_gap', [1981, 5, 2048, 107]), 2: ('silk_spot', [300, 33, 1027, 962]), 3: ('silk_spot', [164, 603, 851, 919]), 4: ('silk_spot', [613, 378, 1135, 624]), 5: ('oil_spot', [1697, 618, 1750, 694]), 6: ('inclusion', [1622, 7, 2006, 108]), 7: ('oil_spot', [1549, 12, 1608, 194]), 8: ('oil_spot', [1560, 275, 1608, 383]), 9: ('oil_spot', [1549, 23, 1615, 66]), 10: ('oil_spot', [1696, 1, 1750, 45])}
(vtuber) link@gim-yeongjaeui-MacBookAir CODE %
```

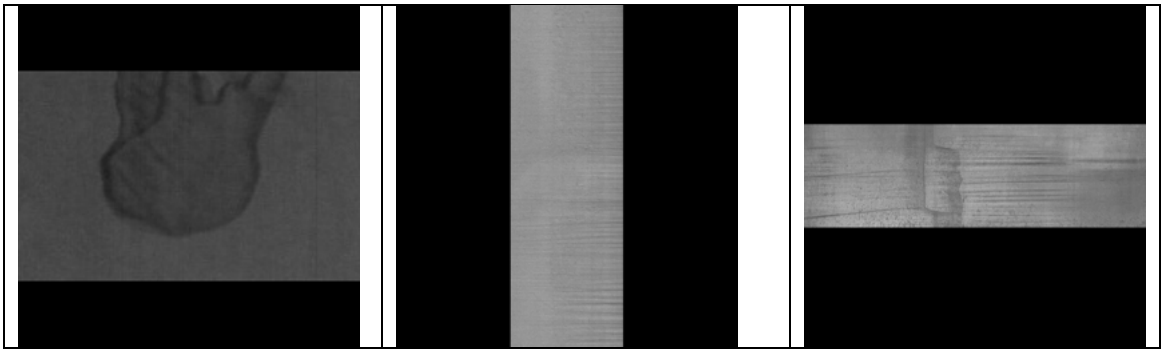
JSON 문법에 대한 미숙지로 도중 다양한 오류가 발생한 모습

이후 커스텀 데이터셋에서 직접 정보를 얻는 것이 아니라 데이터 전처리 과정에서 라벨 별로 폴더를 만드는 방법을 통해 문제 해결.



1-3 패딩 작업

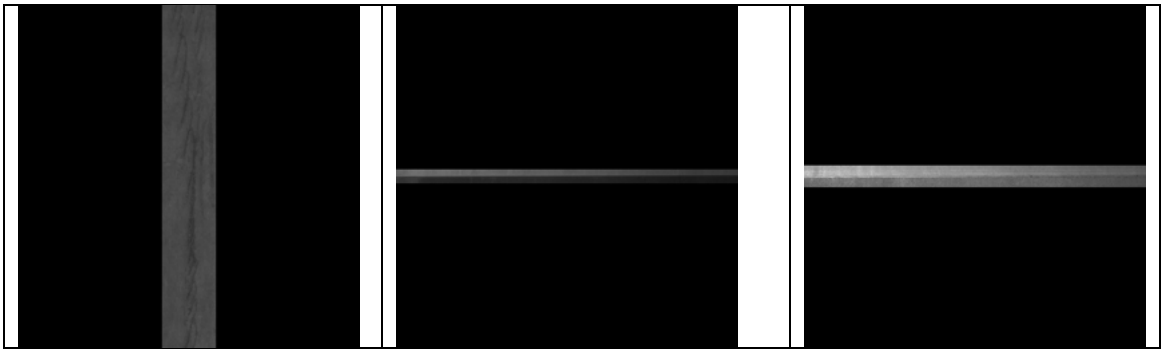
이미지 크롭을 마치고 이미지 크기를 256x256에 맞게 패딩을 붙이는 작업을 실행.



패딩 작업이 완료된 모습

1-3특이사항

패딩작업을 마치고 보니 애초에 결함 부분이 아주 작거나 얇은 선 종류인 경우 데이터 부분보다 패딩의 면적이 더 크다는 것을 발견



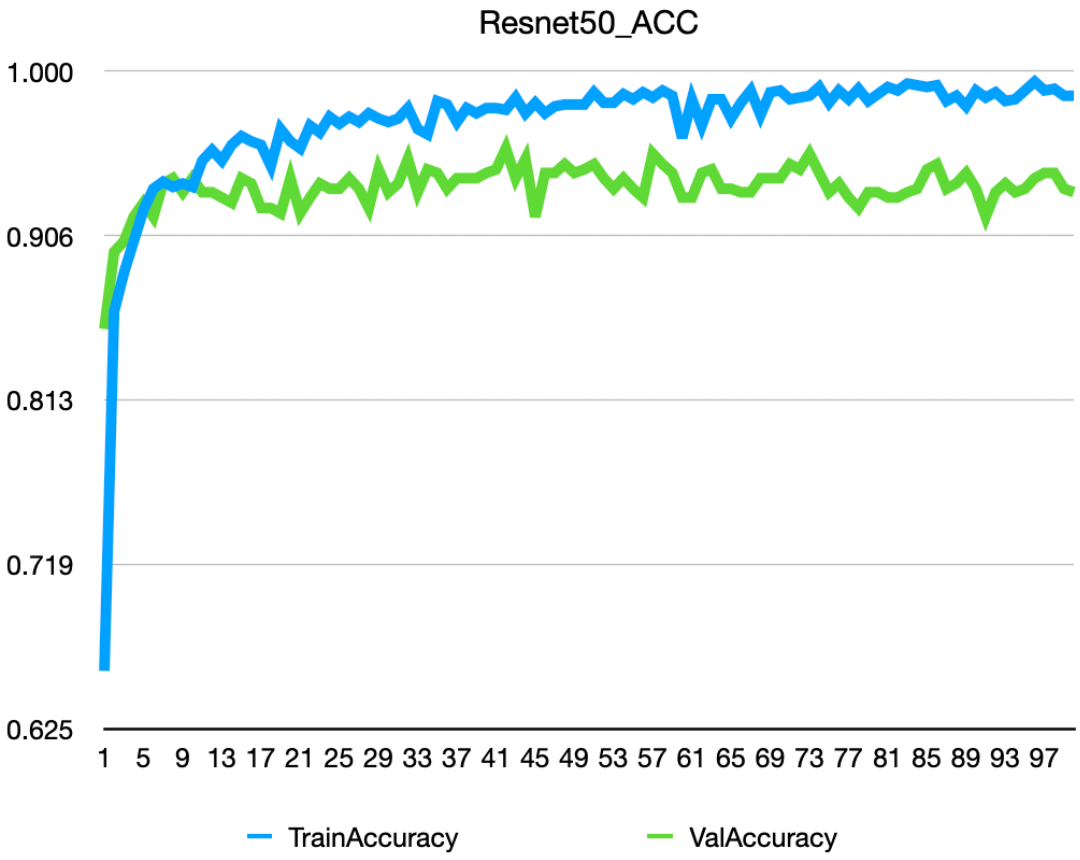
데이터 부분보다 패딩 부분이 더 많은 데이터

➔ 크롭만 된 데이터와 패딩까지 된 데이터 모두 학습 시켜본 결과 정확도와 손실에 큰 차이가 없다는 것을 확인함.

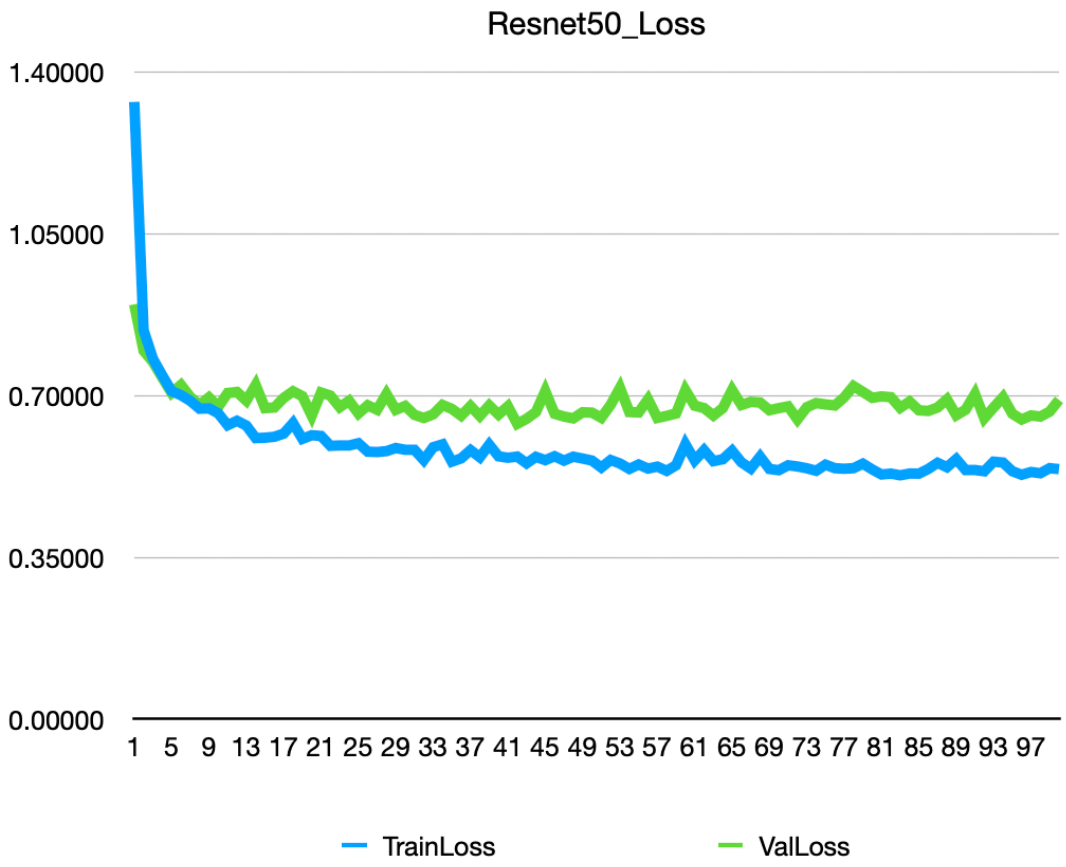
2. 학습진행

모델은 Resnet50, Swin_t, EfficientNet_b4 이 세 가지를 사용하였고, 옵티마이저와 손실함수는 AdmaW와 LabelSmoothingCrossEntropy로 통일 lr 0.001 batch_size는 각각 모델에 맞게 조정하면서 학습 진행.

3. 학습 결과

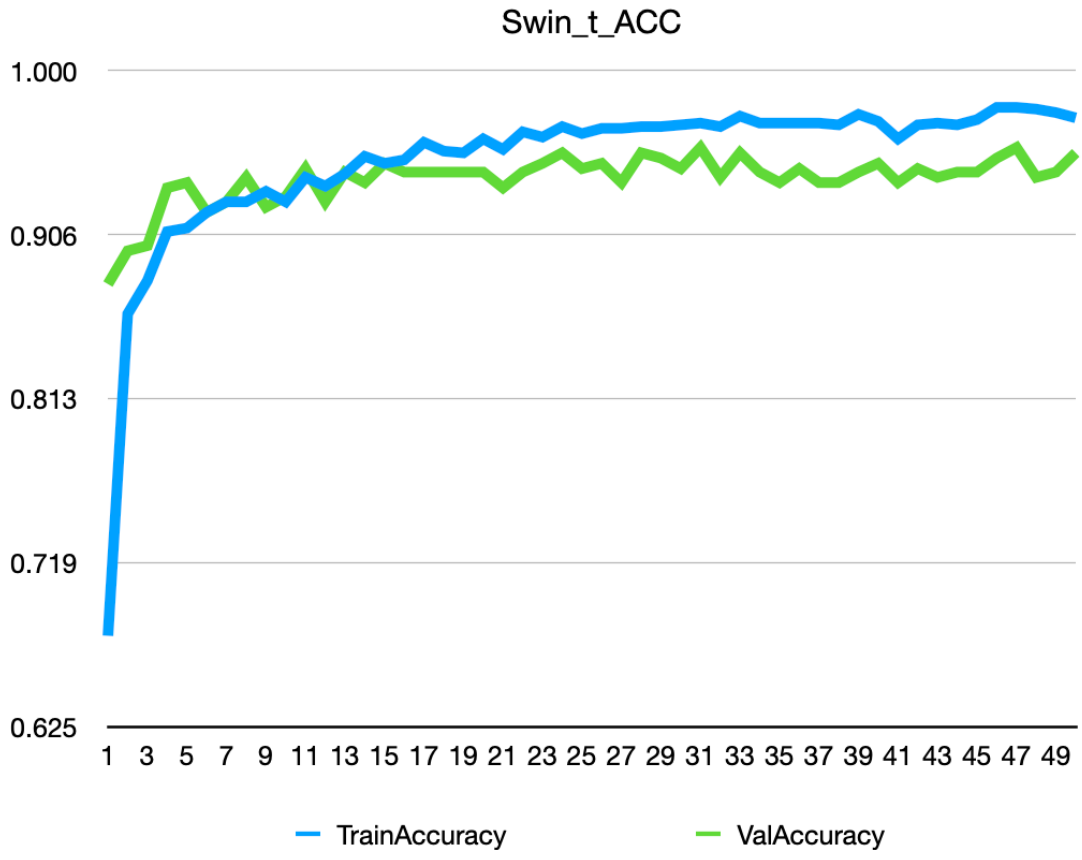


Resnet50 정확도

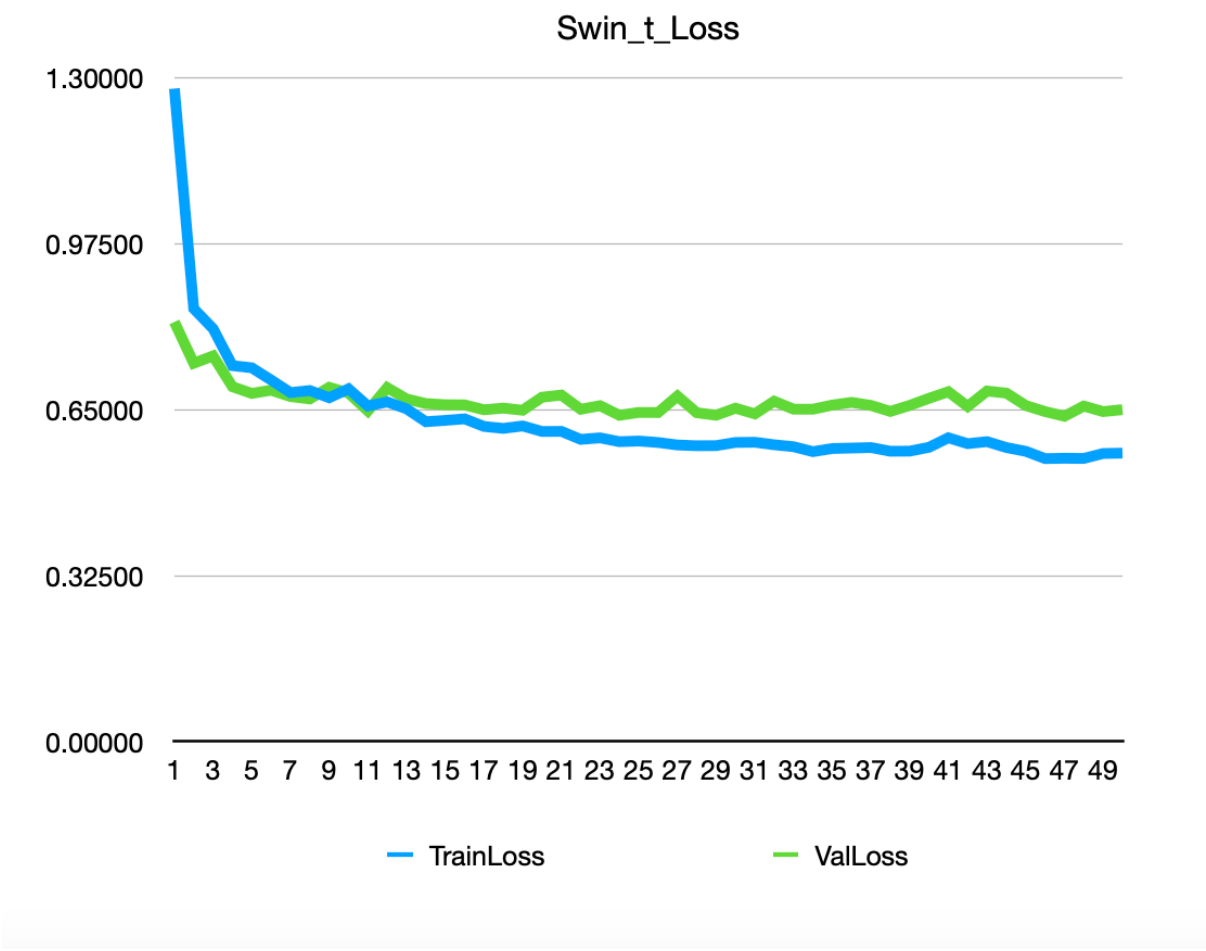


Resnet50 Loss

첫 에포크를 제외하면 정확도와 손실 모두 좋은 것을 확인 할 수 있음. 이후 진행되는 모델 학습에서는 에포크를 50으로 줄이기로 결정.

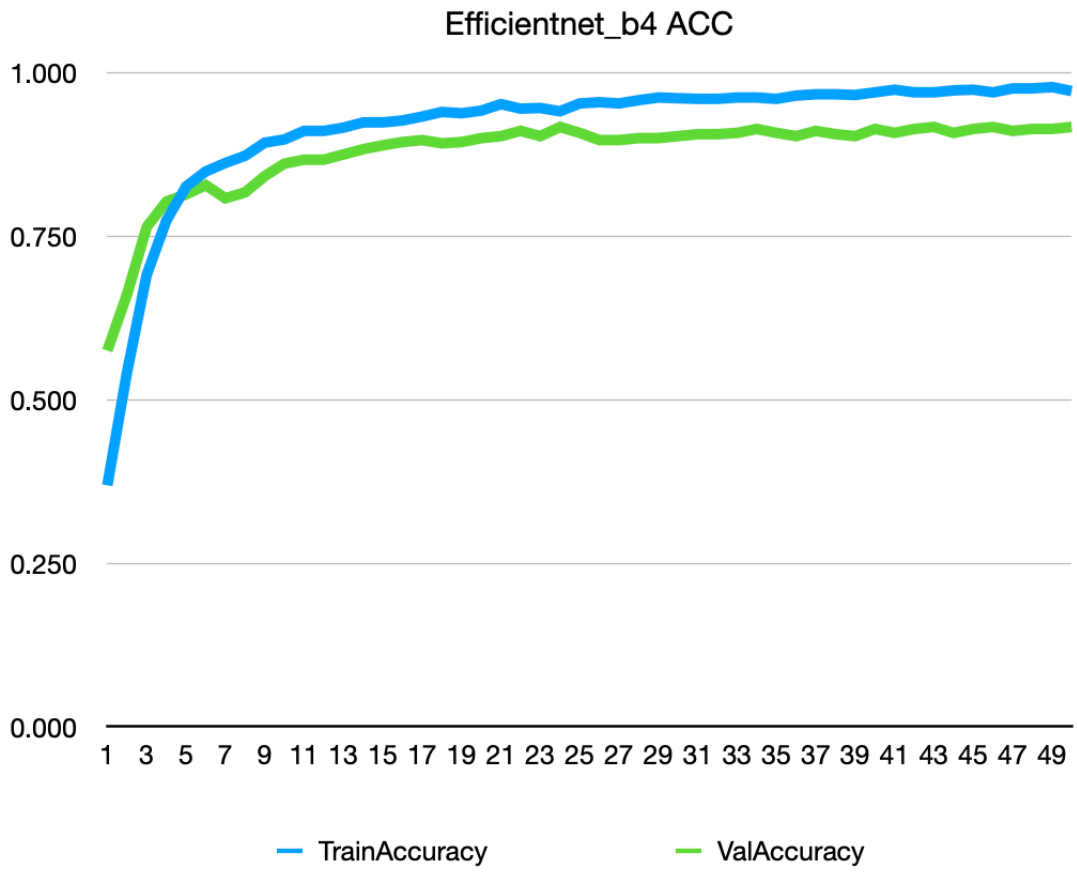

Swin_t_ACC
1.000
0.906
0.813
0.719
0.625
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
TrainAccuracy
ValAccuracy

Swin_t 정확도

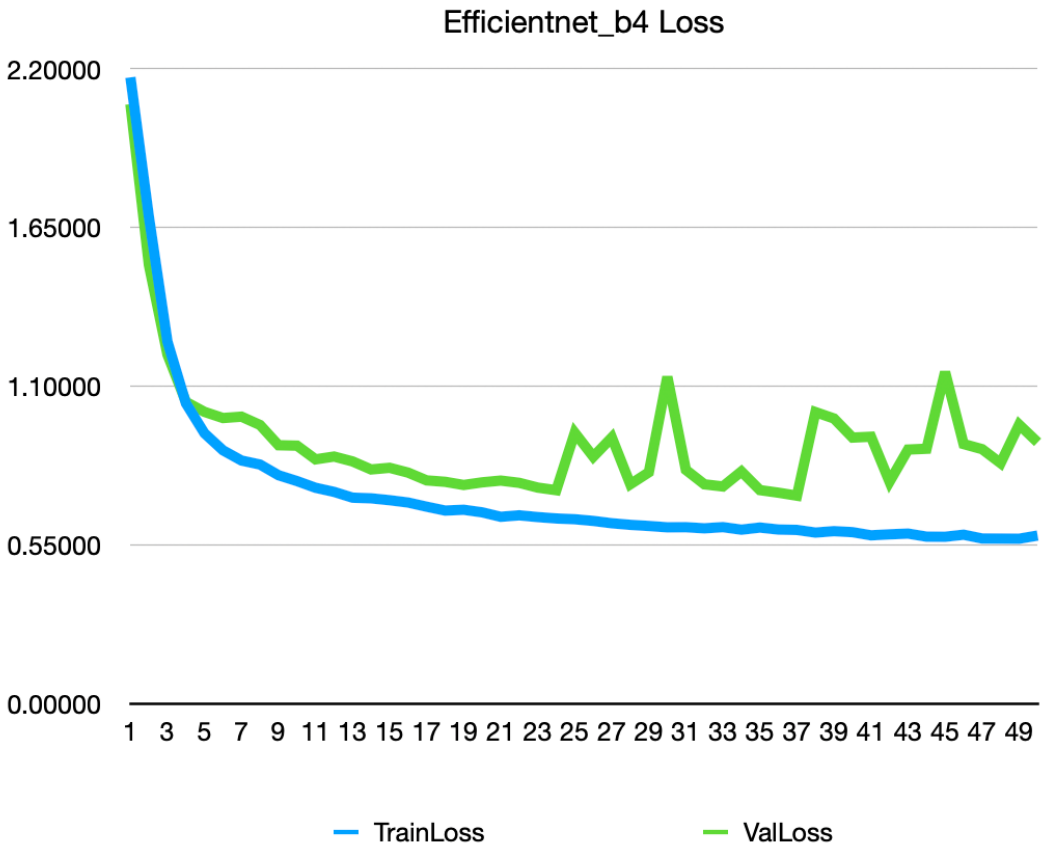


Swin_t Loss

Swin_t 모델도 Resnet50과 비슷하게 첫 에포크에서는 낮은 정확도를 보여줬지만 이후 학습에서는 점점 나아지는 모습을 보여줌.



Efficientnet_b4 정확도



Efficientnet_b4 Loss

스터디 학습에서 가장 좋지 못한 결과를 냈던 Efficientnet 이었던 만큼 우려가 컸지만 초반을 제외하면 세 모델 중에 가장 안정적인 정확도를 보여줌. 하지만 25 에포크를 넘어가는 시점에서 Val Loss가 조금 튀는 모습을 보여주기도 함.

테스트 결과

```
C:\Miniconda\envs\vtuber\python.exe "C:/Users/labadmin/Desktop/New folder/0116/test.py"
Effi_Adam Acc >> 0.9232954545454546

Process finished with exit code 0
```

Resnet50 테스트 정확도 0.923

```
C:\Miniconda\envs\molru\python.exe "C:/Users/labadmin/Desktop/New folder/0116/test.py"
Effi_Adam Acc >> 0.9176136363636364

Process finished with exit code 0
```

Swin_t 테스트 정확도 0.917

```
C:\Miniconda\envs\molru\python.exe "C:/Users/labadmin/Desktop/New folder/0116/test.py"
C:\Miniconda\envs\molru\lib\site-packages\torchvision\models\_utils.py:208: UserWarning:
  warnings.warn(
C:\Miniconda\envs\molru\lib\site-packages\torchvision\models\_utils.py:223: UserWarning:
  warnings.warn(msg)
Effi_Adam Acc >> 0.9261363636363636

Process finished with exit code 0
```

Efficientnet 데스트 정확도 0.926