

# [영상처리와딥러닝] 기말프로젝트 보고서

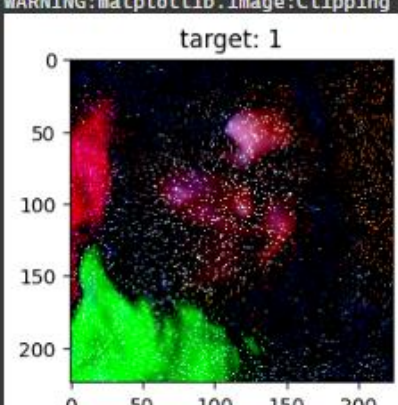
스마트 IoT전공 20217137 강슬기

## 1. 데이터 전처리 및 증강

학습 데이터셋은 Fake 2500 장, Real 2500 장으로 크기가 작아서 데이터 증강(Data Augmentation)을 통해 학습 데이터의 다양성을 높이는 전략을 사용했습니다. Albumentations api 레퍼런스들을 하나씩 살펴보고 첫 학습에서 활용해보면 좋을 것 같은 transforms 을 적용해보았습니다. HorizontalFlip(좌우반전), Rotate(회전), RandomRain(비내리는 효과), RandomShadow(데이터에 도형 그림자 추가), GaussNoise(가우시안 노이즈), ISONoise(ISO 노이즈추가), MotionBlur(모션 블러) 등 다양한 변형을 적용했습니다. 초기 테스트에서는 데이터 증강 효과를 확인하기 위해 스케줄러나 weight decay 를 설정하지 않고 학습을 진행하였습니다. 초기 학습에서는 Baseline 코드에 있는 ResNet18 모델을 사용하였으며, Adam Optimizer 를 활용하여 8 에포크 동안 학습을 진행했습니다. 학습 결과, test set 에서 스코어 0.92800 가 나왔습니다. 첫 학습을 진행한 후 Train 과 Test 데이터를 자세히 살펴보니 초기 증강 기법이 우리의 data set 에는 과하다고 판단하여, 이후 학습부터는 과도하지 않게 적절히 데이터 증강 기법을 조정하였습니다.

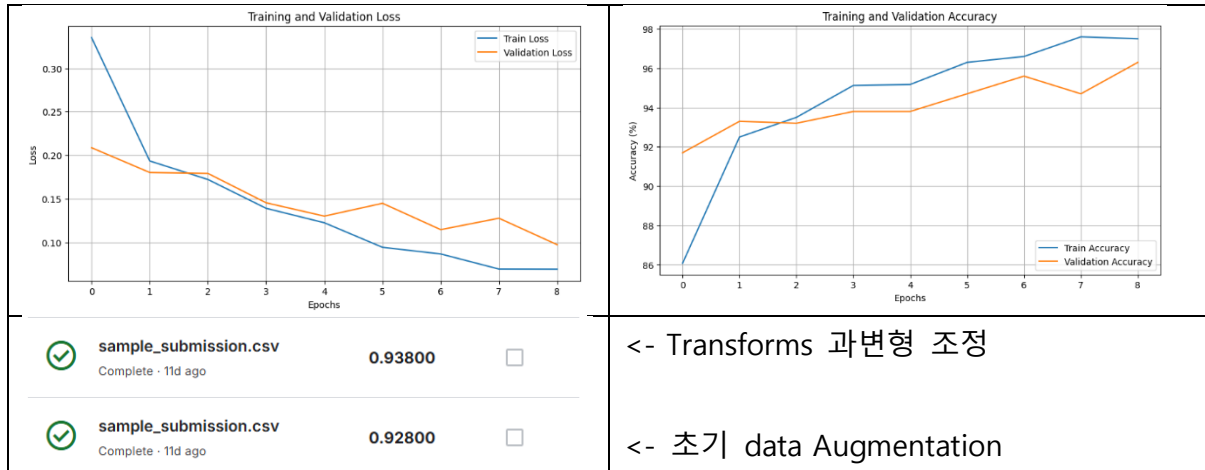
첫 학습

과한 Transforms 기법 조정

<pre>[6] from torchvision import transforms from torchvision.datasets import ImageFolder import albumentations  # 본인의 학습 전략에 맞는 transform 사용 # https://albumentations.ai/docs/examples/example/ # https://albumentations.ai/docs/api_reference/full_reference/  transforms_train = albumentations.Compose([     albumentations.Resize(224, 224),     albumentations.HorizontalFlip(p=0.5),     albumentations.Rotate(limit=30, p=0.5),     albumentations.RandomRain(p=0.5),     albumentations.RandomShadow(p=0.5),     albumentations.GaussNoise(var_limit=(10.0, 50.0), p=0.5),     albumentations.ISONoise(color_shift=(0.01, 0.05), intensity=(0.1, 0.5), p=0.5),     albumentations.MotionBlur(blur_limit=3, p=0.3),     albumentations.Normalize(), ])  transforms_val = albumentations.Compose([     albumentations.Resize(224, 224),     albumentations.Normalize() ])</pre>	<pre>from torchvision import transforms from torchvision.datasets import ImageFolder import albumentations  transforms_train = albumentations.Compose([     albumentations.Resize(224, 224),     albumentations.HorizontalFlip(p=0.5),     albumentations.Rotate(limit=30, p=0.5),     albumentations.RandomShadow(p=0.5),     albumentations.GaussNoise(var_limit=(10.0, 50.0), p=0.7),     albumentations.Normalize(), ])  transforms_val = albumentations.Compose([     albumentations.Resize(224, 224),     albumentations.Normalize() ])</pre>
	

## 2. 모델 선택 및 실험

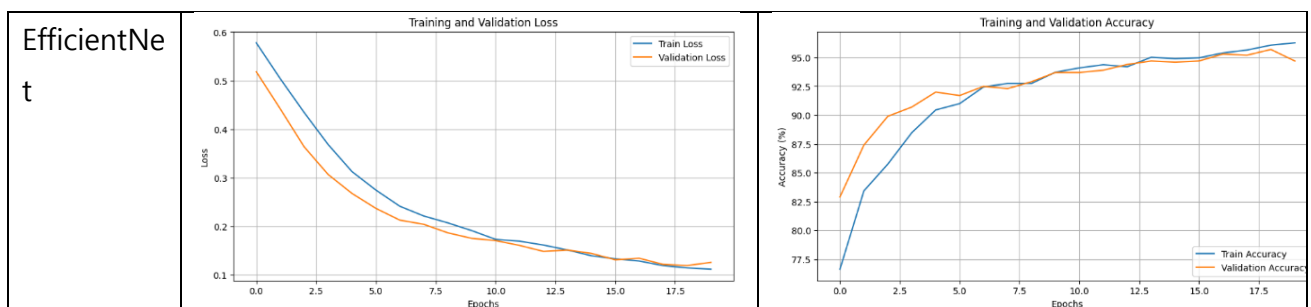
첫 학습은 baseline code 의 예제로 있던 ResNet18 모델로 학습하였습니다. (8 epoch, learning\_rate= 5e-5, optimizer=Adam)

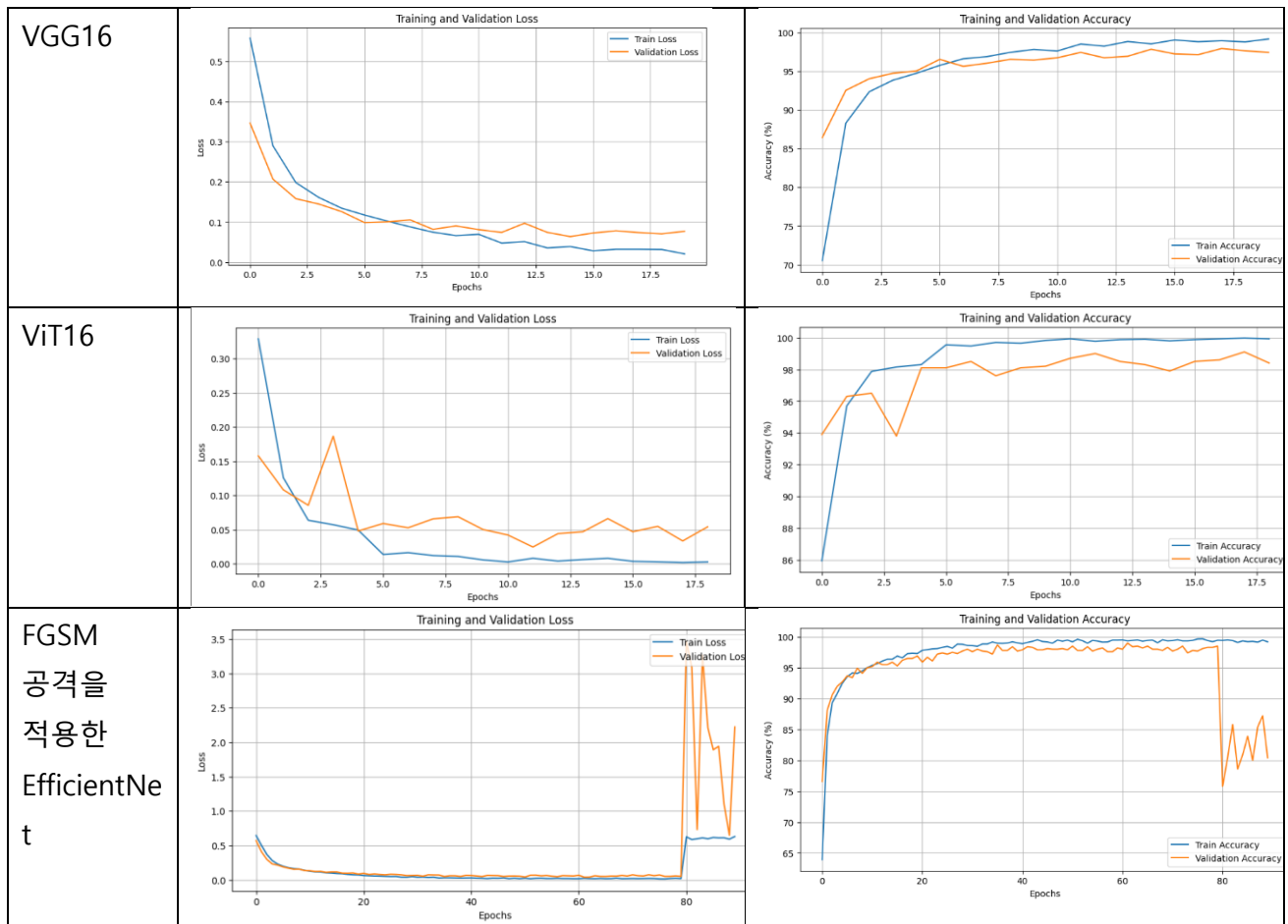


위 결과를 통해 데이터 전처리(데이터증강)이 어느정도 잘 작동하는 것을 확인하였습니다. Pre trained 된 모델을 활용할 수 있다는 점을 고려하여 다른 모델들을 사용했을 때 성능 차이를 비교해보기 위해 EfficientNet 과 ViT(Vision Transformer)를 추가로 학습했습니다.

- EfficientNet\_B0: 적은 데이터셋에서도 좋은 성능을 보이는 모델로 알려져 있기에 학습 시 early stopping 과 weight decay 기법을 추가하여 과적합을 방지하고 일반화 성능을 향상시키는 데 중점을 두었습니다.
- ViT: 이미지 패치를 입력으로 받아 전처리된 데이터를 효과적으로 학습할 수 있습니다. ViT 모델은 한 번도 사용해보지 못했어서 본 대회를 통해 ViT 가 어느정도 성능을 가지는지 궁금하여 사용해봤습니다.
- VGG : 데이터셋이 작기 때문에 너무 복잡한 모델 대신 단순한 구조의 VGG 모델 사용.
- EfficientNet\_v2\_s: 기존 EfficientNet 에서 Fused-MBConv 를 사용하여 속도를 높이고 계산량을 줄인 모델으로, 제한된 리소스 내에서 빠르게 학습 가능하여 사용해봤습니다.

Optimizer 는 Adam 을 사용하였고, Learning Rate 는 모든 모델에 1e-5 를 사용, EfficientNet 을 사용할 때부터 ReduceLROnPlateau 스케줄러를 적용하여 validation loss 가 개선되지 않을 경우 학습률을 동적으로 감소시켰습니다. weight decay 또한 1e-5 으로 설정하여 학습하였습니다. 과적합을 방지하기 위해 validation loss 개선이 없는 경우 학습을 조기 종료하였습니다.





위 3 개의 모델 학습 결과 validation 에서의 loss 나 acc 가 비슷하다는 것을 알 수 있습니다.

EfficientNet 모델을 학습할 때 FGSM(Fast Gradient Sign Method) 를 적용하여 학습해 보았습니다. 적대적 데이터를 생성해서 모델의 예측을 방해하기 위해 노이즈(perturbation)를 추가하여 학습하였습니다.

```


patience = 10 # 열리 스탑 기준 (최대 10 에포크 동안 개선 없을 시 종료)
early_stop_counter = 0 # 성능 개선이 없을 때 증가
epsilon = 0.01 # FGSM에서 최대 노이즈 perturbation 크기

# FGSM 어택
def FGSM_fun(model, input_data, target_label, epsilon):
    input_data.requires_grad = True # 그래디언트 계산
    output = model(input_data) # 모델로 입력 데이터 예측
    loss = torch.nn.CrossEntropyLoss()(output, target_label) # 예측값과
    model.zero_grad()
    loss.backward()

    # FGSM으로 적대적 데이터 생성
    adversarial_data = input_data + epsilon * input_data.grad.sign() #
    adversarial_data = torch.clamp(adversarial_data, 0, 1) # 0 1 데이터
    return adversarial_data

```

FGSM은 입력 데이터의 그래디언트를 계산하고, 그래디언트의 부호(sign)을 사용해서 입력 데이터에 epsilon 크기의 노이즈를 추가합니다. 원본 데이터와 적대적 데이터를 모델에 모두 입력하여 손실을 계산하고, 이 손실 값을 더한 총 손실을 기준으로 모델의 가중치를 업데이트하는 방식입니다.


**sample\_submission\_efficientNet\_FGSM.csv**
0.87200  
 Complete · 3h ago

적대적 공격을 적용하여 학습한 모델 또한 다른 모델들과 비슷한 validation 정확률을 보이고 있지만, csv 제출 결과 스코어는 0.87200 으로 적대적 공격을 적용하지 않고 학습한 모델보다 못한 결과를 보였습니다.

Submission and Description	Public Score <sup>①</sup>	Select
<b>sample_submission_efficientNet.csv</b> Complete · 42m ago <b>적대적 공격을 제외하고 57epoch 학습된 EfficientNet</b>	0.94600	<input type="checkbox"/>
<b>sample_submission_efficientNet_FGSM.csv</b> Complete · 1h ago <b>FGSM 적대적 공격 추가한 20epoch 학습된 EfficientNet</b>	0.87200	<input type="checkbox"/>
<b>sample_submission_efficientnet_v2_s_norm_121...</b> Complete · 4h ago <b>40epoch 학습 EfficientNet_v2_s 모델</b>	0.96400	<input type="checkbox"/>
<b>sample_submission_vgg_1210_2320.csv</b> Complete · 4h ago <b>20epoch 학습 VGG</b>	0.94600	<input type="checkbox"/>
<b>sample_submission_EfficientNet_norm_1210_21...</b> Complete · 16h ago <b>Test set에 transform을 적용x normal, 40epoch 학습 EfficientNet</b>	0.96200	<input type="checkbox"/>
<b>sample_submission_EfficientNet_Purification_1...</b> Complete · 16h ago <b>Test set에 불러 transform을 추가, 40epoch 학습 EfficientNet</b>	0.96000	<input type="checkbox"/>
<b>sample_submission_EfficientNet_Purification_1...</b> Complete · 16h ago <b>Test set에 노이즈를 죽이기 위해 불러 transform을 추가한 EfficientNet</b>	0.95200	<input type="checkbox"/>
<b>sample_submission_ViT_Purification_1210_2043...</b> Complete · 16h ago <b>Test set에 노이즈를 죽이기 위해 불러 transform을 추가한 ViT</b>	0.96000	<input type="checkbox"/>
<b>sample_submission_ViT_1210_2001.csv</b> Complete · 17h ago <b>Data 증강, Weight decay, scheduler 적용된 ViT</b>	0.96400	<input type="checkbox"/>
<b>sample_submission_efficientNet_1210_0110.csv</b> Complete · 1d ago <b>Data 증강, Weight decay, scheduler 적용된 EfficientNet</b>	0.96200	<input type="checkbox"/>
<b>sample_submission_efficientNet_1210_0056.csv</b> Complete · 2d ago <b>Data 증강, Weight decay 적용된 EfficientNet</b>	0.96600	<input checked="" type="checkbox"/>
<b>sample_submission_efficientnet.csv</b> Complete · 2d ago <b>Data 증강만 적용된 EfficientNet</b>	0.94400	<input type="checkbox"/>
<b>sample_submission_model_ensemble.csv</b> Complete · 1h ago	0.97400	<input checked="" type="checkbox"/>

test set 의 스코어를 보더라도 세 모델이 비슷한 성능을 내고 있는 것을 알 수 있습니다. 제일 높은 스코어를 기록한 0.96600 의 EfficientNet 모델이며, 이 모델은 weight decay 과 Data Augmentation 기법만 적용되고 learning scheduler 가 적용되지 않은 모델입니다.

```
# 모델 불러오기
model_path = [
    '/content/model/efficientNet_learning_param_41.pt',
    '/content/model/efficientNet_param_7.pt',
    '/content/model/param_28_ViT_241210_1958.pt',
    '/content/model/vgg_best_loss_param_14.pt',]

ids = []
targets = []

for k, (image, image_id) in enumerate(test_loader):
    with torch.no_grad():
        ensemble_output = []
        for m_path in model_path:
            # 모델 불러오기
            model = torch.load(m_path, weights_only=False).to(device)
            model.eval()

            output = model.forward(image.to(device))
            ensemble_output.append(output)

        # 확률 평균 계산함
        avg_output = sum(ensemble_output) / len(ensemble_output)
        target = avg_output.argmax(1).item()

        ids.append(image_id[0])
        targets.append(target)

submission = pd.DataFrame({'ID': ids, 'label': targets})
submission.to_csv('./sample_submission_model_ensemble.csv', index = False)
```

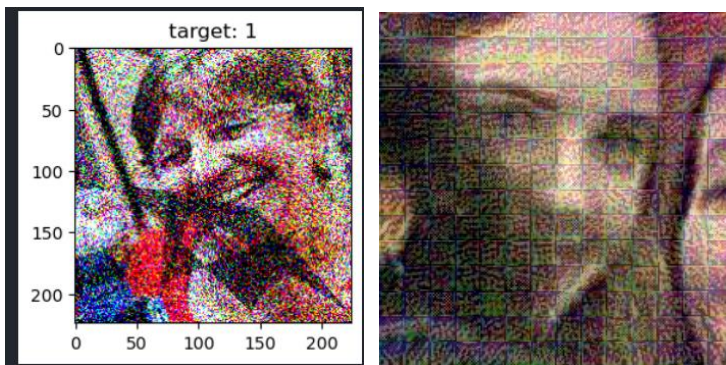
다른 네트워크 구조의 모델들의 성능이 비슷하기 때문에, 최종적으로 모델 앙상블 전략을 사용했습니다. 정확률이 가장 높았던 weight\_decay 를 적용한 EfficientNet 모델과, 앞의 모델에 learning scheduler 을 적용한 EfficientNet 모델, ViT 모델과, VGG 모델 총 4 개의 결과들을 평균내는 방식으로 앙상블을 수행했으며, 이를 통해 0.97400 으로 제일 제출했던 파일 중 제일 높은 스코어를 달성하게 되었습니다.

### 3. 결론

초기 실험에서는 ResNet18 을 사용해 데이터 증강 기법의 효과를 확인하였고, 이후 EfficientNet 과 ViT, VGG 모델을 활용하여 동일한 설정에서 네트워크의 성능을 비교하였습니다. 데이터 증강, learningrate scheduler, weight decay, early stopping 등의 다양한 기법들이 모델 성능에 중요한 역할을 한다는 것을 확인할 수 있었습니다. 특히나 EfficientNet 모델에서 제일 높았던 성능을 보여주었으며 데이터 증강 및 weight decay 기법이 적절히 적용된 결과라고 판단됩니다.

데이터 증강, weight decay 만 적용하여 학습한 EfficientNet 모델이 성능이 제일 좋은 것을 보면, 적대적 공격이 적용된 현재 데이터셋과 최대한 비슷하게 Albumentations 의 가우시안 노이즈의 transform 을 적용한 것의 영향이 크다고 생각합니다.

위의 데이터 증강 기법을 적용하면 아래 사진과 같은 이미지로 적용이 되게 됩니다. 이는 Train 이나 Test 셋에 적용된 이미지와 비슷하게 적용되어 모델이 학습하고 추론하는데 큰 문제가 없었다고 생각합니다.



적대적 공격(Adversarial Training)을 적용한 EfficientNet 모델의 경우, 기존 데이터 증강 및 학습 기법에 추가적으로 모델을 성능을 높이려고 하였으나, validation 결과에서는 기대했던 만큼의 성능 개선을 보이지 않았습니다. 이미 Albumentations 의 가우시안 노이즈 transform 이 적용되어 적대적 공격이 현재 데이터셋의 특성과 일치하지 않아서 생긴 것이라고 생각합니다.

마지막으로 적용한 EfficientNet, ViT, VGG 등의 모델을 결합한 앙상블 전략은 스코어가 비슷한 개별적인 모델의 한계를 보완하면서 0.97400 이라는 스코어를 기록했습니다. 이 결과를 통해 다양한 모델의 특성을 활용한 앙상블 기법이 복잡한 데이터셋에서 모델의 완성도를 높이기 위한 필수적인 전략임을 확인할 수 있었습니다.