Computer Vision
기말고사 프로젝트

Jaesung Lee

curseor@cau.ac.kr

2025. 06. 11.

개요

- 시멘틱 세그멘테이션 모델을 1개 구현하여 제출(6p 참조)
- 중간고사 이후 수업에서 사용하였던 5개의 데이터셋에 대해 1회 실험 성능을 측정(5p 데이터셋 상세 설명 참조)
- 최종 점수는 **준비한 1개의 모델**에 대해 5개의 데이터셋에 대한 평균으로 산정(3p 상세 평가 기준 참조)
- 평가지표는 예측 마스크와 Ground Truth (GT) 마스크 간의 IoU이며 이는 실험 코드에서 제공(7p 평가 지표 설명 참조)
- 데모 당일에는 **최종 제출 모델 1개**에 대한 실행 여부 확인 후 제출
  - 최종적으로 모델 1개만 제출
  - 깃허브 private 리파지토리에 제안 모델 소스 코드 업로드 후 조교 입회하에 오류 없이 실행 여부 체크
- 제약 조건
  - **사전학습 모델 활용 불가**
  - **파라미터 수는 3.0M 미만**
  - **데이터 증강 불가**

❖ 중간 고사와 다르게 데모시 하이퍼파라미터 튜닝과정이 없으며 상세 훈련 세팅은 실험 코드 참고

평가 기준

➢ 컴퓨터비전 기말 프로젝트의 평가 기준은 아래와 같음(미제출시 0점)

<table>
<tr><th>항 목</th><th>평가 상세 (비중, 점수)</th><th colspan="2">설 명</th></tr>
<tr><td rowspan="2">최종 성능 평가
(80%, 32점)</td><td>모델 파라미터 수</td><td>• 제출 모델의 모델 파라미터 수가 적을 수록 높은 순위를 부여하여 순위 기준으로 1.0점 ~ 0.2점 차등 점수 부여</td><td rowspan="2">최종 성능 점수 =
$32 \times$
$\sqrt{(\text{파라미터 점수}) \times (\text{정확도 점수})}$
(4p 참조)</td></tr>
<tr><td>정확도</td><td>• 파라미터 수 제약(3M) 기준으로 통과시 A 그룹, 미통과시 B 그룹
• 제출 모델 정확도 기준으로 각 그룹 내 순위 결정
• B 그룹은 A 그룹 마지막 순위에 이어 정확도 최종 순위 할당
• 정확도 최종 순위 기준으로 1.0점 ~ 0.2점 차등 점수 부여</td></tr>
<tr><td rowspan="2">그 외 (20%, 8점)</td><td>모델 개선 점수 (10%, 4점)</td><td colspan="2">• 모델 및 훈련 변경에 따라 1, 2, 4점의 점수 부여
  • 4점: 추가 모듈 부착 혹은 직접 구현한 모듈을 부착한 경우
  • 2점: 레이어 개수 조절, 커널 수 조절 등 마이너한 조정
  (옵티마이저, 러닝레이트와 그와 관련된 하이파라미터 수정의 경우에는 1점 처리)
  • 1점: 모델 변경 없이 제출
❖ 14p에 1, 2, 4점에 대한 세부 기준 및 예시 명시</td></tr>
<tr><td>데모 점수 (10%, 4점)</td><td colspan="2">• 데모 제출물 오류 발생 시 1점씩 감점(지각, 코드 오류 등)</td></tr>
<tr><td colspan="4">총계 (100%, 40점)</td></tr>
</table>

평가 기준

- 성능 평가 기준은 파라미터 점수와 정확도 점수의 적절한 트레이드 오프 달성을 할 수 있도록 아래와 같이 기하 평균을 통해 설정됨.

$$\text{최종 성능 점수} = 32 \times \sqrt{(\text{파라미터 점수}) \times (\text{정확도 점수})}$$

- 아래 예시 각 점수 조합별 계산 결과 테이블 참조

정확도 점수 \ 파라미터 점수	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.2	0.20	0.24	0.28	0.32	0.35	0.37	0.40	0.42	0.45
0.3	0.24	0.30	0.35	0.39	0.42	0.46	0.49	0.52	0.55
0.4	0.28	0.35	0.40	0.45	0.49	0.53	0.57	0.60	0.63
0.5	0.32	0.39	0.45	0.50	0.55	0.59	0.63	0.67	0.71
0.6	0.35	0.42	0.49	0.55	0.60	0.65	0.69	0.73	0.77
0.7	0.37	0.46	0.53	0.59	0.65	0.70	0.75	0.79	0.84
0.8	0.40	0.49	0.57	0.63	0.69	0.75	0.80	0.85	0.89
0.9	0.42	0.52	0.60	0.67	0.73	0.79	0.85	0.90	0.95
1.0	0.45	0.55	0.63	0.71	0.77	0.84	0.89	0.95	1.00

학습 및 평가 데이터셋

- 중간고사 이후 수업에서 사용하였던 5개의 데이터셋에 대해 성능을 측정
- Train, Validation, Test Set은 각 데이터셋별로 고정되어있으며 이를 사용하여 각 데이터셋 1회 반복 실험을 수행
- 데이터의 세부 내용은 해당 주차의 파일을 참고

no.	Dataset name	Domain	Image size	#Classes	# Images	
					Total	Train:Val:Test
1	PASCAL VOC 2012	General	256 × 256	21	2,913	1,747:583:583
2	ETIS-LaribPolypDB	Medical	256 × 256	2	196	118:39:39
3	CVPPP2017-LSC	Plant	256 × 256	2	810	972:324:324
4	CFD	Crack	256 × 256	2	118	70:24:24
5	CarDD	Car Damage	256 × 256	2	518	310:104:104

논문 리스트

- 다음은 참고용 모델이며 이외에 다른 모델 구현 및 튜닝 가능
- 각 모델은 제약조건인 3.0M 미만이 아닌 모델들도 있으니 제출시 유의

no.	Model name	Title	Year	Venue	# Parameters (M)	code	Citations (Google Scholar 25.05.27.)
1	SeaFormer-T++	SeaFormer++: Squeeze-Enhanced Axial Transformer for Mobile Visual Recognition	2025	International Journal of Computer Vision	1.8M	https://github.com/fudan-zvg/SeaFormer	26
2	LCNet-3-7	Lightweight Context-Aware Network Using Partial-Channel Transformation for Real-Time Semantic Segmentation	2024	IEEE Transactions on Intelligent Transportation Systems	0.5M	https://github.com/lztjy/LCNet	46
3	PIDNet-S	PIDNet: A Real-Time Semantic Segmentation Network Inspired by PID Controllers	2023	CVPR	7.6M	https://github.com/XuJiacong/PIDNet	519
4	DDRNet-23-Slim	Deep Dual-Resolution Networks for Real-Time and Accurate Semantic Segmentation of Traffic Scenes	2023	IEEE Transactions on Intelligent Transportation Systems	5.7M	https://github.com/ydhongHIT/DDRNet	449
5	SeaFormer-T	SeaFormer: Squeeze-enhanced Axial Transformer for Mobile Semantic Segmentation	2023	ICLR	1.7M	https://github.com/fudan-zvg/SeaFormer	201
6	SegFormer	SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers	2021	NeurIPS	3.7M	https://github.com/NVlabs/SegFormer	6,382
7	DFANet-B	DFANet: Deep Feature Aggregation for Real-Time Semantic Segmentation	2019	CVPR	4.8M	https://github.com/huaifeng1993/DFANet	831
8	LEDNet	LEDNet: A lightweight encoder-decoder network for real-time semantic segmentation	2019	IEEE international conference on image processing (ICIP)	0.9M	https://github.com/xiaoyufenfei/LEDNet	479
9	ESPNet	ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation	2018	ECCV	0.4M	https://github.com/sacmehta/ESPNet	1,130
10	DeepLabV3+ (ResNet-101)	Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation	2018	ECCV	59.3M	https://github.com/jfzhang95/pytorch-deeplab-xception	19,749
11	SegNet	SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation	2017	IEEE transactions on pattern analysis and machine intelligence	29.4M	https://github.com/vinceecws/SegNet_PyTorch	22,466
12	ERFNet	ERFNet: Efficient Residual Factorized ConvNet for Real-time Semantic Segmentation	2017	IEEE Transactions on Intelligent Transportation Systems	2.1M	https://github.com/Eromera/erfnet_pytorch	1,661
13	ENet	ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation	2016	Arxiv	0.4M	https://github.com/yassouali/pytorch-segmentation/blob/master/models/enet.py	3,114
14	UNet	U-Net: Convolutional Networks for Biomedical Image Segmentation	2015	MICCAI	31.2M	https://github.com/milesial/Pytorch-UNet	111,472

평가 지표 설명

- 정확도 평가는 각 테스트 이미지별로 Intersection over Union (IoU)를 통해 집계됨.
  - $\mathrm{IoU}(P, G) = \frac{|P \cap G|}{|P \cup G|}$ P는 예측 박스의 픽셀 집합, G는 Ground Truth 박스의 픽셀 집합

- 데이터셋 D_i 의 평균 IoU:

$$\mathrm{IoU}_{D_i} = \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{|P_{i,j} \cap G_{i,j}|}{|P_{i,j} \cup G_{i,j}|}$$

- 서로 다른 5 개의 데이터셋의 평균 loU:

$$\mathrm{IoU}_{\mathrm{avg}} = \frac{1}{5} \sum_{i=1}^{5} \mathrm{IoU}_{D_i} = \frac{1}{5} \sum_{i=1}^{5} \left(\frac{1}{N_i} \sum_{j=1}^{N_i} \frac{|P_{i,j} \cap G_{i,j}|}{|P_{i,j} \cup G_{i,j}|} \right)$$

- $\mathrm{IoU}_{\mathrm{avg}}$ 가 제출 모델의 정확도 평가 점수로 활용됨.

실험 코드

- competition_main.ipynb
  - 실제 평가시 활용될 실험용 코드
  - 제출하지 않음.
  - student_id 변수 수정 후 에러 없이 작동 완료되는지 확인 필수

```
[1]: 1 student_id = '2023110214'
```

실험 코드

- Model_Test.ipynb
  - competition_main.ipynb에 잘 작동하는지 확인하기 위한 간소화된 테스트 코드
  - 붉은 색 표시된 부분을 수정하여 제안 모델을 불러와서 서로 다른 클래스 수를 테스트
  - 제출하지 않음.

[1]:

```python
from models.submission_2023110214.submission_2023110214 import submission_2023110214


import torch

num_classes_list = [1, 2, 4, 8, 20, 21]  # 여러 개의 number of classes를 테스트

for n in num_classes_list:
    model = submission_2023110214(in_channels=3, num_classes=n)
    print(f"Testing with number of classes = {n}")
    model.train()
    output = model(torch.rand((16, 3, 256, 256)))
    expected_shape = (16, n, 256, 256)
    if output.shape == expected_shape:
        print(f"Train mode: The test has passed (output.shape={output.shape})")
    else:
        print(f"Train mode: The test has failed (output.shape={output.shape}, expected={expected_shape})")

    model.eval()
    output = model(torch.rand((16, 3, 256, 256)))
    if output.shape == expected_shape:
        print(f"Eval mode: The test has passed (output.shape={output.shape})")
    else:
        print(f"Eval mode: The test has failed (output.shape={output.shape}, expected={expected_shape})")
```

Last executed at 2025-05-27 18:57:22 in 9.62s

```
Testing with number of classes = 1
Train mode: The test has passed (output.shape=torch.Size([16, 1, 256, 256]))
Eval mode: The test has passed (output.shape=torch.Size([16, 1, 256, 256]))
Testing with number of classes = 2
Train mode: The test has passed (output.shape=torch.Size([16, 2, 256, 256]))
Eval mode: The test has passed (output.shape=torch.Size([16, 2, 256, 256]))
Testing with number of classes = 4
Train mode: The test has passed (output.shape=torch.Size([16, 4, 256, 256]))
Eval mode: The test has passed (output.shape=torch.Size([16, 4, 256, 256]))
Testing with number of classes = 8
Train mode: The test has passed (output.shape=torch.Size([16, 8, 256, 256]))
Eval mode: The test has passed (output.shape=torch.Size([16, 8, 256, 256]))
Testing with number of classes = 20
Train mode: The test has passed (output.shape=torch.Size([16, 20, 256, 256]))
Eval mode: The test has passed (output.shape=torch.Size([16, 20, 256, 256]))
Testing with number of classes = 21
Train mode: The test has passed (output.shape=torch.Size([16, 21, 256, 256]))
Eval mode: The test has passed (output.shape=torch.Size([16, 21, 256, 256]))
```

제출용 모델 코드

- models/{학번} 모듈
  - 실제 제안 모델을 포함하도록 수정 가능한 부분
  - models/{학번}/{학번}.py 내부에 {학번} 클래스를 포함해야 함.
  - 그 외에 models/{학번} 내부에 모델을 위한 코드들을 다수 추가 가능
  - Model_Test.ipynb에서 학번만 수정하여 실행하였을 때 문제없이 동작하도록 구조를 작성해야 함.
  - 메인 모델 클래스는 in_channels와 num_classes 변수를 입력받을 수 있도록 해야함. 그외 하이퍼파라미터는 입력 불가
  - num_classes는 출력 노드를 의미하며 데이터 클래스가 2개일 경우 1, 아닐 경우 데이터 클래스와 출력 노드 수는 동일함.
  - **작성하여 제출함.**

```python
import torch
import torch.nn as nn

class submission_2023110214(nn.Module):
    def __init__(self, in_channels=1, num_classes=1):
        # - in_channels: 입력 영상의 채널 수를 지정합니다. 반드시 함수 인자로 받아야 합니다.
        # - num_classes: 클래스(출력 채널) 수를 지정합니다. 반드시 함수 인자로 받아야 합니다.
        #                바이너리 세그멘테이션의 경우 노드를 1개 갖도록 하므로 num_classes를 1로 가정합니다.
        super().__init__()
        init_features=32
        features = init_features
        self.encoder1 = self.block(in_channels, features)
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.encoder2 = self.block(features, features * 2)
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.encoder3 = self.block(features * 2, features * 4)
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)
        self.encoder4 = self.block(features * 4, features * 8)
        self.pool4 = nn.MaxPool2d(kernel_size=2, stride=2)
```

제출용 학습 하이퍼파라미터 코드

- training_args.py
  - 3개 주요 학습 하이퍼파라미터 선언으로 수정 가능한 부분
  - Optimizer와 lr scheduler, loss function을 선언
  - 아래 예시를 그대로 사용 가능
  - **작성하여 제출함.**

- requirements.txt
  - 필요 라이브러리 및 버전 작성
  - **작성하여 제출함.**

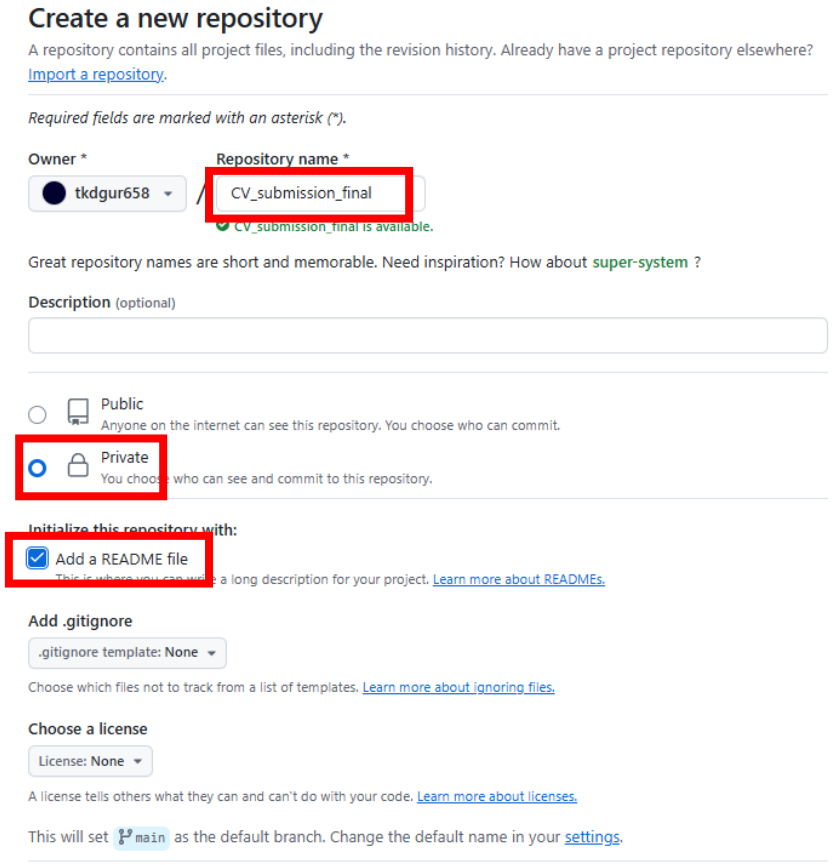
- 그 외에 파이썬 버전 변경 등 requirements.txt로 달성 어려운 이슈가 필요할 경우 깃허브 README.md에 명령어 상세 기술

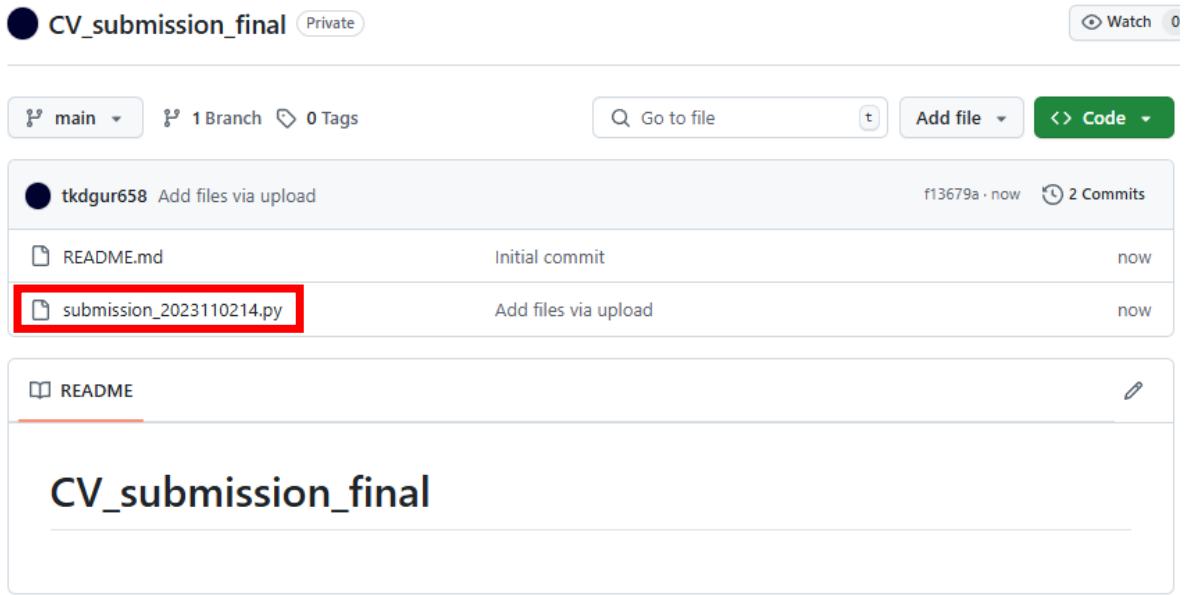
```python
import torch
import torch.nn.functional as F
def Make_Optimizer(model):
    # TODO: 모델의 파라미터와 원하는 하이퍼파라미터(learning rate 등)를 사용하여
    #       optimizer(예: torch.optim.Adam, torch.optim.SGD 등)를 생성 및 반환하도록 작성하세요.
    return optimizer
def Make_LR_Scheduler(optimizer):
    # TODO: optimizer와 원하는 학습률 스케줄링 전략(예: CosineAnnealingLR, StepLR 등)을 사용하여
    #       lr_scheduler를 생성 및 반환하도록 작성하세요.
    return lr_scheduler

def Make_Loss_Function(number_of_classes):
    # TODO: 클래스 수에 따라 적절한 loss function(예: CrossEntropyLoss 등)을
    #       생성 및 반환하도록 작성하세요.
    return loss_function
```

```python
# 예시용 (제출시 삭제)
def Make_Optimizer(model):
    return torch.optim.AdamW(model.parameters(), lr=1e-3, weight_decay=1e-4)

def Make_LR_Scheduler(optimizer):
    return torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max = 1e-6, eta_min = 30)

def Make_Loss_Function(number_of_classes):
    class DiceCELoss:
        def __init__(self, weight=0.5, epsilon=1e-6, mode='multiclass'):
            self.weight = weight
            self.epsilon = epsilon
            self.mode = mode
```

제출 및 데모 가이드라인

- 깃허브 private 리파지토리에 models/{학번} 내부의 파일들 모두 업로드 (6월 17일 23시 59분까지)
  - tkdgur658@gmail.com로 collaborator 초대
- 데모 당일 1명씩 돌아가며 코드 동작 여부를 조교와 함께 확인 후 문제 없을시 종료
- 이슈사항 발생 인원은 위 체크 종료 후 논의





보고서 제출

- 기말고사 프로젝트에서 선정한 논문의 선정 동기와 제출한 모델에 대한 설명을 포함한 보고서를 제출합니다.
  - 분량 : 본문 한글 공백 제외 500자 이상 1,000 미만(영단어 동일)
  - 포함 내용
    - 논문 선정 동기
    - 제출 모델 설명
    - 4p 및 13p의 모델 개선 점수를 위한 근거 설명(누락시 평가서 제외)
  - 데드라인 : 6월 17일 23시 59분
  - 제출 방식 : 포탈상에서 PDF 제출

Appendix (모델 개선 점수 세부 기준)

1. 4점 해당 항목
   - 모듈 추가 또는 구조적 개선이 명확히 드러나야 4점으로 인정
   - 중간고사와 마찬가지로 별도 보고서 과제에 설명 필수

2. 2점 해당 항목
   - 깊이(레이어 개수), 너비(각 층의 커널 개수), 커널 크기, 스트라이드, 패딩 크기, 풀링 크기, 드롭아웃 비율 등의 PyTorch 혹은 선정한 논문에서 **사전 정의된 스칼라 값의 단순 변경**
   - 이외에 단순 공식 PyTorch에서 제공하는 특정 구현을 그대로 가져와 기존의 구현과 대체하거나 추가하는 방식(torch.nn, torch.functional 등)
     - 예시
       - 업샘플링, 다운샘플링 방식 변경: Max Pool, Average Pool, Strided Convolution, Global Average Pooling, Transposed Conv, Interpolation, Unpooling 등
       - 정규화 방식 변경: Batch Normalization, Layer Normalization, Instance Normalization, Group Normalization 등
       - 활성화 함수변경: ReLU, LeakyReLU, ELU, GELU, Swish, Sigmoid, Tanh 등

3. 1점 해당 항목
   - 아래 항목의 수정은 2점이 아닌 1점에 해당하는 수정사항이며 이들을 변경없이 제출하여도 동일하게 1점
     - 옵티마이저 및 그에 해당하는 하이퍼파라미터
     - 러닝레이트, 러닝레이트 스케줄러 및 그에 해당하는 하이퍼파라미터