

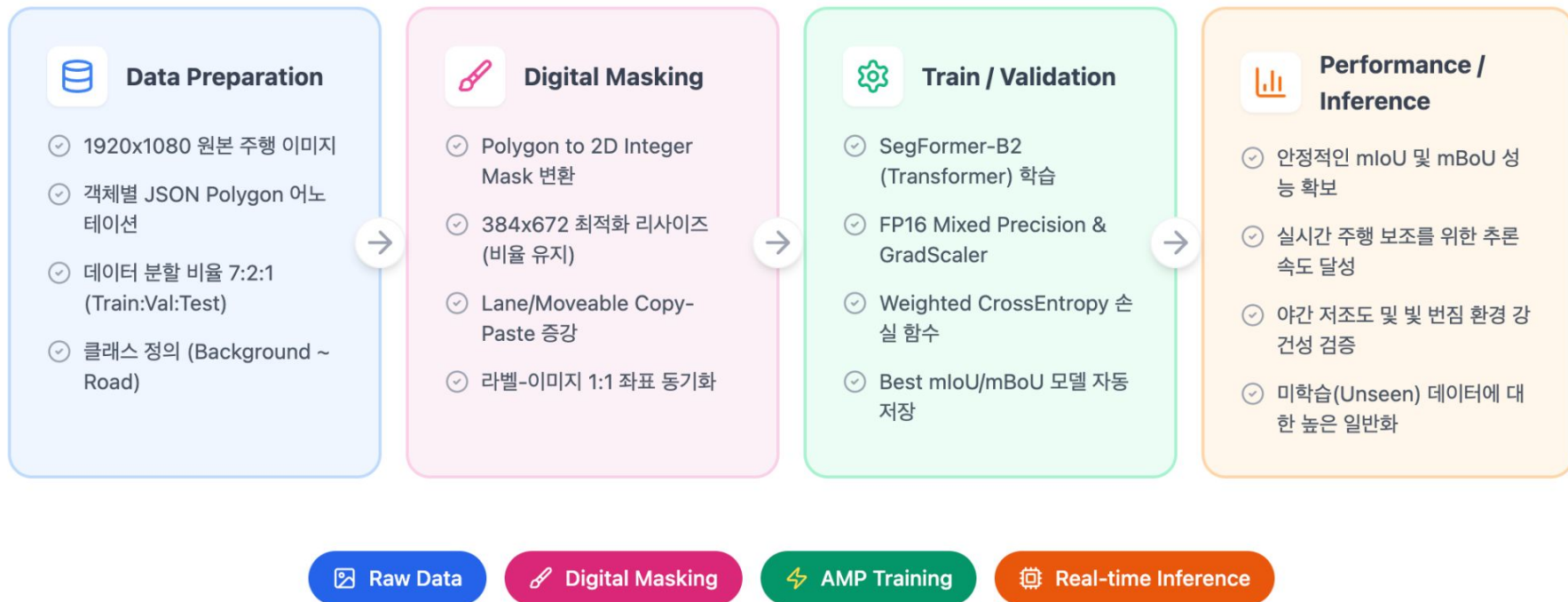
# Loss Zero

야간 안전 주행을 위한 고정밀·실시간 다중 클래스 시멘틱  
세그멘테이션 모델 연구

## 프로젝트 목표 및 핵심과제

- 본 프로젝트는 야간 오토바이 주행 시 발생하는 빛 번짐과 저조도 문제를 해결하기 위해, 학습 단계에서 경험하지 못한 새로운 주행 프레임에서도 주변 가동 객체를 **95% 이상의 확률로 정확히 식별**하는 것을 목표로 합니다.
- 주행 환경의 특수성을 고려하여 모델의 추론 속도(FPS)와 정확도 간의 최적 균형을 확보함으로써 실시간 주행 안전성을 극대화합니다.

# 파이프라인



# 모델 선택 가이드 (1): CNN 계열 모델 분석 및 한계

## 실험 모델군 분석 (DeepLabV3+ vs U-Net)

기존 연구 및 팀 내 분석을 통해 도출된 CNN 기반 모델의 특징은 다음과 같습니다.

### 1. DeepLabV3+ (ResNet50 Backbone)

- a. 강점: ASPP(다중 스케일 분석)와 Decoder를 통해 뭉개진 경계선을 복구하는 능력이 탁월함.
- b. 약점: 모델이 무거워 실시간 주행(High FPS)에 불리하며, 야간의 빛 번짐(Halo) 노이즈를 개별 픽셀 단위로만 판단하려는 경향이 있음

### 2. U-Net

- a. 입문용으로 적합하나, 복잡한 도로 환경의 다중 클래스 분류에는 정교함이 부족함..

# 모델 선택 가이드 (2): CNN 계열 모델 분석 및 한계

## 주요 기술적 통찰 (Best Selection)

기존 연구 및 팀 내 노트북 분석을 통해 도출된 CNN 기반 모델의 특징은 다음과 같습니다.

### 1. Loss Function

- a. 야간 도로의 핵심인 '이동 물체'와 차선 비중이 작기 때문에 **Weighted CrossEntropy** 도입이 필수적임.

### 2. Backbone

- a. 깊은 층의 **ResNet50**이 야간의 복합적인 시각 특징 추출에 유리함을 확인.

### 3. 결론

- a. CNN 계열 중 최선은 **DeepLabV3+**이나, 야간 주행의 특수성(빛 번짐, 실시간성)을 해결하기엔 구조적 한계가 존재하고, 실시간성 빠른 추론에는 부적합함.

# 모델 선택 가이드 (3): 왜 SegFormer(Transformer)인가?

"판을 읽는 전략가" Transformer, 야간 주행의 노이즈를 맥락으로 극복하다.

CNN vs Transformer 사고방식의 차이

	CNN (DeepLabV3+, ResNet)	Transformer (SegFormer)
인식 방식	국소적 ( <b>Local</b> ): 주변 픽셀 위주 분석	전역적 ( <b>Global</b> ): 이미지 전체 맥락 파악
야간 대응	빛 번짐 노이즈를 장애물로 오인 가능성 높음	<b>Attention 매커니즘</b> 으로 주변 환경을 고려해 판단
연산 효율	이미지 크기 증가 시 연산량 폭증	계층적( <b>Hierarchical</b> ) 구조로 속도와 성능 최적화

# 모델 선택 가이드 (4): 왜 SegFormer(Transformer)인가?

## SegFormer를 선택한 결정적 이유

기존 연구 및 팀 내분석을 통해 도출된 모델의 특징은 다음과 같습니다.

1. **노이즈 강건성**: 전조등 빛 번짐으로 인해 픽셀 정보가 손실되어도, 주변의 도로/차선 맥락(Self-Attention)을 이용해 가동 객체를 정확히 추론.
2. **압도적 속도**: MiT(Mix-Transformer) 인코더는 ResNet50급 성능을 내면서도 훨씬 가벼워 실시간 **FPS 확보**에 최적.
3. **확장성**: **Cityscapes** 사전학습 가중치를 활용하여 **95%** 정확도 도달 시간을 단축하고, 6개 클래스(배경, 라이더, 본인 바이크, 가동 객체, 차선, 도로) 구분에 최적화됨

# 최종 모델 설계 및 구현 전략 (Proposed Solution)

성능과 속도의 최적 밸런스, SegFormer-B2 기반의 고효율 아키텍처

최종 모델 명세: **SegFormer-B2**

- Backbone: **MiT-B2** (계층적 트랜스포머 구조로 다양한 크기의 장애물 포착)
- Optimization: **AdamW** (Weight Decay를 통한 오버피팅 방지 및 안정적 학습)
- Speed-Up: **Mixed Precision (FP16)** 기술을 적용하여 메모리 절약 및 속도 향상.



# 최종 모델 설계 및 구현 전략 (Proposed Solution)

## 구현 디테일 (Engineering)

1. **Transfer Learning:** nvidia/segformer-b2-finetuned-cityscapes 모델을 기반으로 우리 데이터셋(6 Classes)에 맞게 **Fine-tuning**.
2. **Label Mapping:** 자율주행 알고리즘 확장을 고려한 Standard 6 Class 체계 구축.

Mask Labeling 이미지크기에 맞추어서 똑같은 W, H

5	5	5
5	1	1
5	1	1

- **학습 원리:** 모델이 예측한 픽셀별 점수와 **라벨링 배열**을 대조해 오차를 교정하며, 최적의 클래스 번호를 짚어내는 **통계적 연결 고리**를 완성함.

# EDA - 클래스 불균형 (Class Imbalance) 분석

전체 픽셀에서 각 클래스가 차지하는 비중을 시각화 했습니다.

 Undrivable: 42.9%

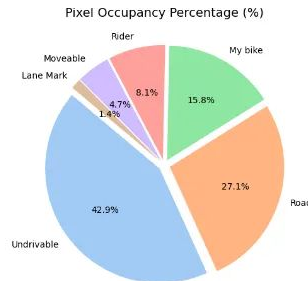
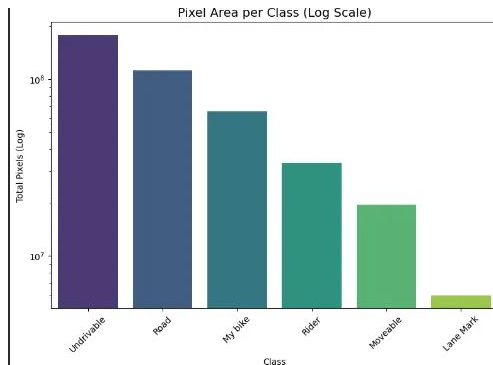
 Road: 27.1%

 Bike: 15.7%

 Rider: 8%

 Moveable: 4.7%

 ✨ Lane Mark: 1.4%



--- 클래스별 분포 요약 ---

Class	Percentage
Undrivable	42.886086
Road	27.109400
My bike	15.797043
Rider	8.061059
Moveable	4.707659
Lane Mark	1.438754

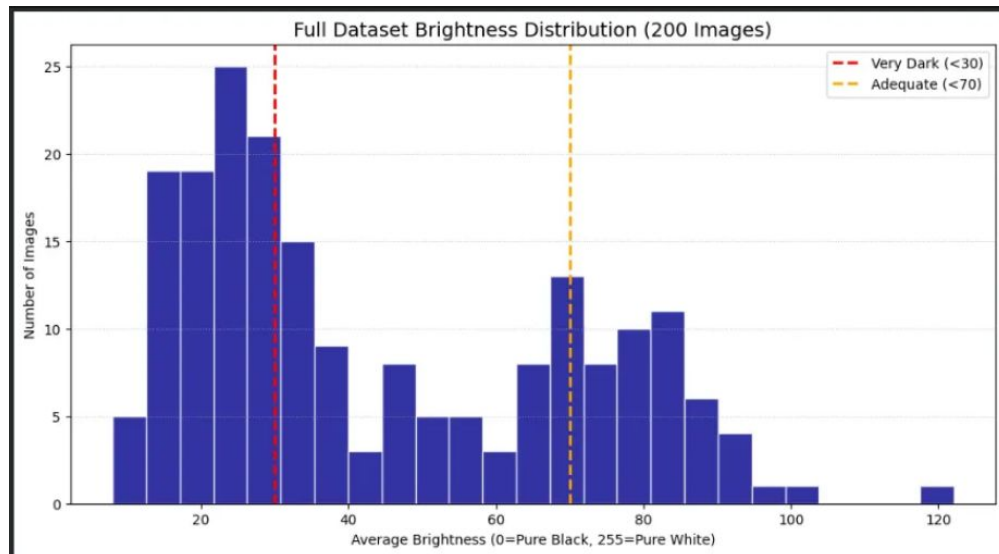
# EDA - 클래스 불균형 개선 방안

```
# 🏴 클래스별 가중치 설정 (Class Weights)
weights = torch.tensor([
    3.0,    # Rider: 5.0 → 3.0 (중요하지만 과하지 않게)
    1.5,    # My bike: 2.0 → 1.5 (내 오토바이는 너무 잘 맞이니 조금 더 낮춤)
    6.0,    # Moveable: 10.0 → 6.0 (절반으로 줄여서 부담 완화)
    12.0,   # Lane Mark: 20.0 → 12 (여전히 제일 강력하지만, 20배는 너무 가혹했음)
    1.0,    # Road: 1.0 (기준점 유지)
    0.8     # Undrivable: 0.5 → 0.8 (배경을 너무 무시해서 도로 경계가 무너지는 것 방지)
], dtype=torch.float).to(CFG['device'])
```

# EDA - 야간 밝기 분석

이미지 밝기 분포를 확인하여, 너무 어두운 이미지에서도 특징을 잡을 수 있을지 검토 했습니다.

```
# YUV 색공간의 Y(밝기) 채널 평균 계산
y_channel = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)[: , : , 0]
brightness_list.append(np.mean(y_channel))
```



--- [전수조사 통계 보고서] ---

1. 평균 밝기: 44.41
2. 중앙값 밝기: 34.07
3. 최저/최고 밝기: 7.99 / 122.05
4. 밝기 30 미만 이미지 비중: 42.5%

[결정] 저조도 이미지가 많습니다. CLAHE 또는 감마 보정 전처리가 필수적입니다.

# 데이터 전처리 - 증강

## 1. 전처리 솔루션 : CLAHE 적용

단순히 밝기를 올리면 야간 노이즈도 같이 커져서 이미지가 깨집니다. CLAHE(Contrast Limited Adaptive Histogram Equalization)를 사용하여 어두운 부분의 대비만 지능적으로 높아 집니다.

Sample 3 [Original] (Index: 11)



Sample 3 [CLAHE Applied]



# 데이터 전처리 - 증강

**albumentations** 라이브러리를 사용했습니다.

RandomBrightnessContrast



RandomGamma



GaussNoise



# 데이터 전처리 - 증강

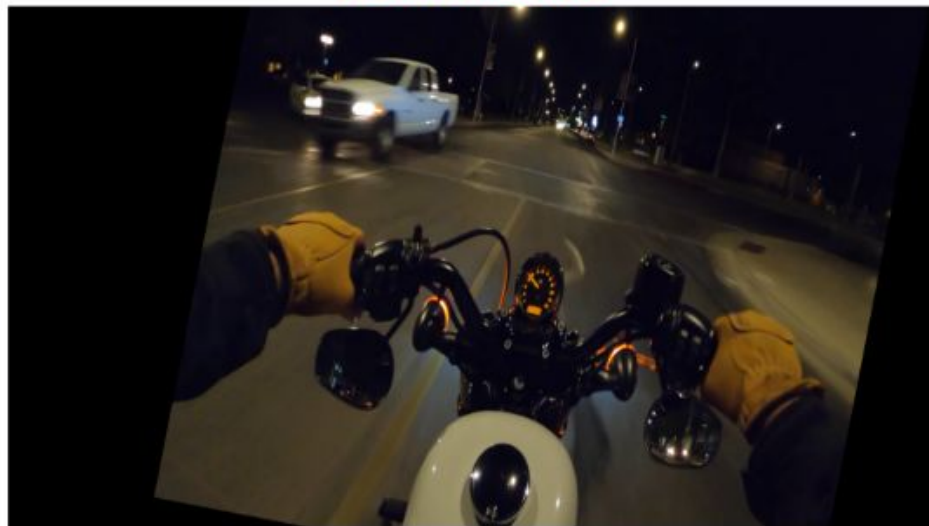
좌우 반전

1. Horizontal Flip



이동/크기/회전

2. ShiftScaleRotate



# 데이터 정규화 (Normalization) 재검토

처음 사용된 정규화 수치는 Cityscapes의 ImageNet 기반 평균/표준편차를 사용 중이었습니다.

```
A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225))
```

야간 주행 데이터셋은 일반적인 낮 이미지보다 평균 밝기값이 훨씬 낮습니다.

전체 야간 데이터셋의 Mean/Std를 직접 계산해서 적용해 보았습니다.



# 데이터 정규화 (Normalization) 재검토

## 데이터셋 전체 Mean/Std 계산하기

모든 이미지를 한 번씩 읽으면서 채널별(R, G, B) 합계와  
제곱 합을 구하여 정확한 통계치를 계산합니다.

```
# 4. 채널별 합계 및 제곱합 누적
sum_ += np.mean(image, axis=(0, 1))
sum_sq += np.mean(image**2, axis=(0, 1))
pixel_count += 1

# 최종 평균 및 표준편차 계산
mean = sum_ / pixel_count
std = np.sqrt((sum_sq / pixel_count) - (mean**2))
```

🌟 [최종 결과] 순수 원본 데이터 통계치  
Mean: (0.176, 0.169, 0.191)  
Std: (0.197, 0.195, 0.195)

# XAI 시각화 grad CAM 사용

모델이 작동하는 방식

합리적인지 ?

이미지의 '어디'를 보고

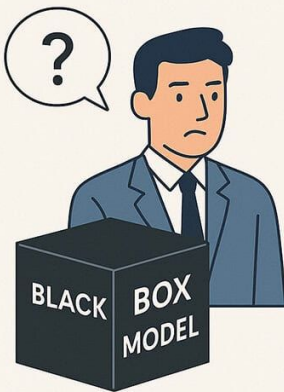
판단했는지 ?

왜 그런 결과가 나왔는지 ?

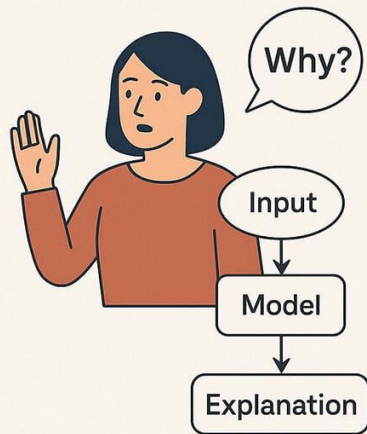
설명하는 단계

XAI는 **eXplainable AI**의 약자로,  
👉 설명 가능한 인공지능을 뜻합니다.

## Need for Explainable-by-Design ML Methods

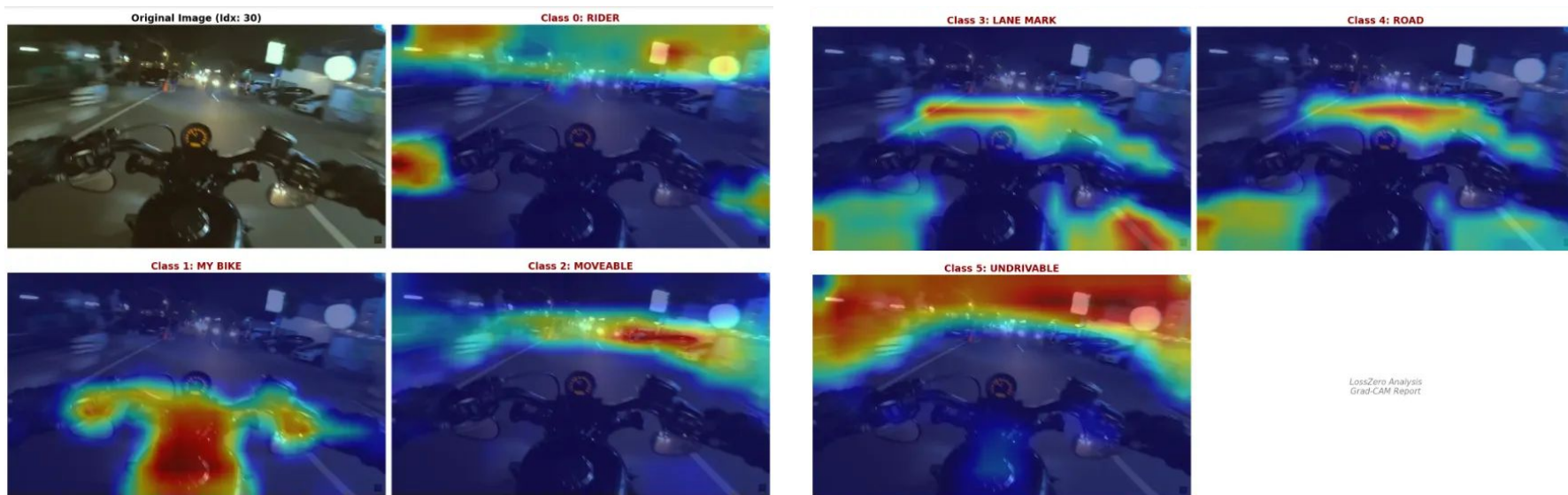


Prediction



# XAI 시각화 grad CAM 사용

모델이 학습을 마친 후 최고의 성능을 낸 mIoU 모델 (.pth)을 저장하게 되는데 그 모델을 불러와서 히트맵을 그려게 시켜 보았습니다.



# 성능 개선 전략 (1): 시행착오와 문제 분석

핵심 요약: "단순 **Crop**과 리사이즈의 한계, 차선(Lane) 정보 손실의 원인이 되다."

## 1차 시도: 효율성 중심의 학습 (Failure)

- 방법: 352 x 352 Random Crop 및 기본 리사이즈
- 문제점:
  - a. 야간 도로는 차선 등 세밀한 객체의 비중이 매우 낮음. 무작위 Crop 시 차선 정보가 잘려 나가거나 데이터 불균형이 심화됨.
- 결과: 차선 IoU 0.1728, Boundary IoU 0.0472 (심각한 인식 저하)

## 2차 시도: 데이터 증강 및 크기 확대 (Trial)

- 방법: 482 x 482 확대, Random Crop 배제, 특정 클래스(Moveable, Lane) 50% 확률 Copy-Paste 적용.
- 분석: 데이터 노출 빈도는 늘었으나, 원본(1920 x 1080) 대비 왜곡된 화면 비율로 인해 미세 객체가 뭉개지는 현상 지속. 큰 폭의 지표 개선 실패.

# 성능 개선 전략 (2): 최종 최적화 및 결과

핵심 요약: "화면 비율 최적화와 전수 Copy-Paste를 통한 성능 비약적 향상"

최종 전략: 비율 고정 리사이즈 & 전수 학습 (Final Solution)

1. **Aspect Ratio Optimization:** 원본 비율을 반영한 384 x 672 리사이즈 적용. 픽셀 왜곡을 최소화하여 차선의 선예도 유지.
2. **Lane-Centric Augmentation:** 모든 학습 이미지에 차선을 강제로 Copy-Paste 하여 모델이 차선의 패턴을 완벽히 학습하도록 유도.

# 성능 개선 전략 (3): 최종 최적화 및 결과

차선 (Lane Mark) 지표 대폭 상승: IoU **0.17** → **0.32** (약 88% 향상), Boundary IoU **0.04** → **0.12** (약 200% 향상).

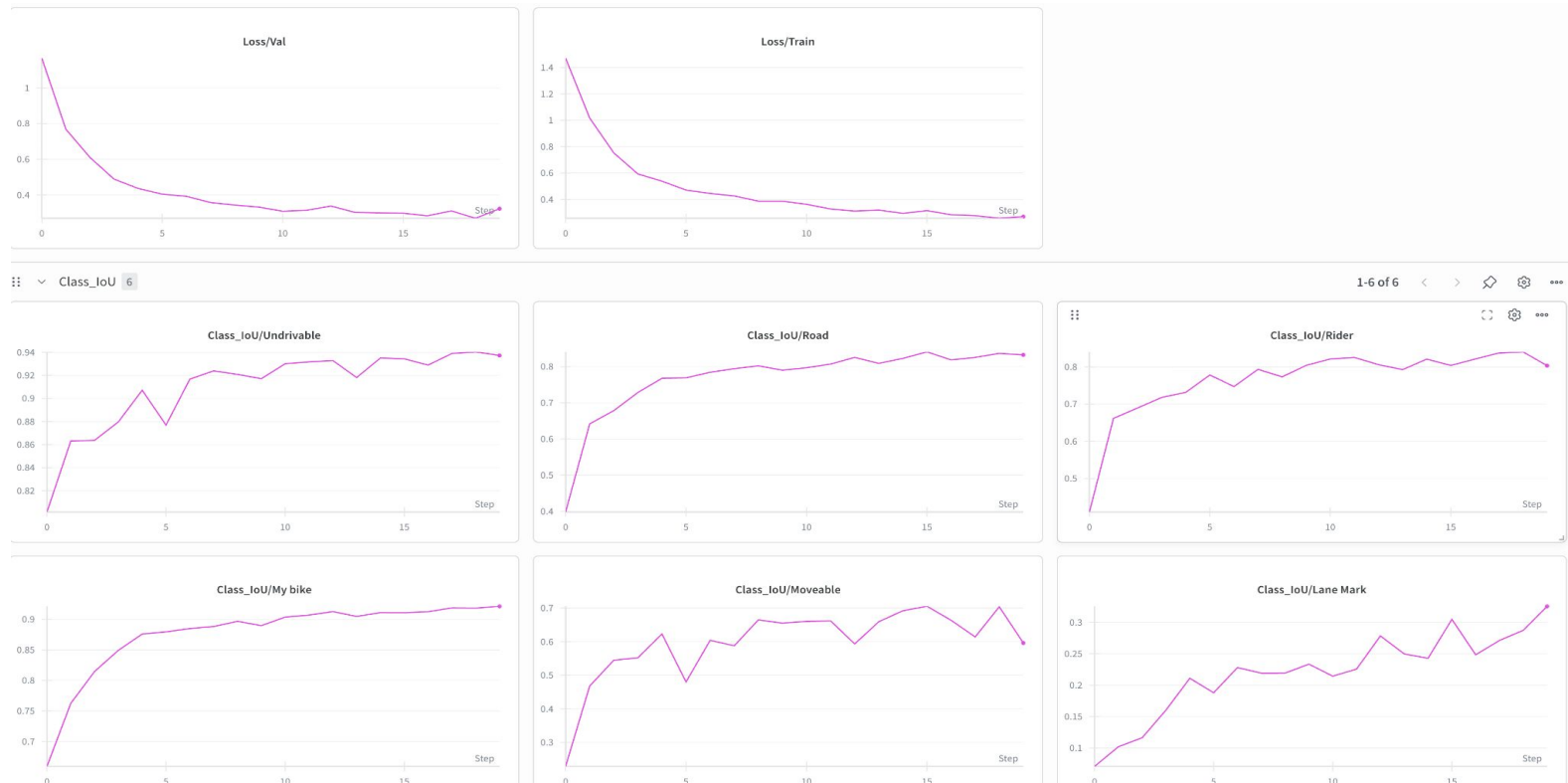
- 1. 전체 모델 성능: mIoU **0.7545** | mBoU: **0.1557** 달성 (안정적인 객체 분별력 확보).
- 2. 문제점: Halo현상과 크기의 다형성, 움직임 때문에 mBoU점수에 치명적임

평가 항목 (Metric)	Category-specific IoU (영역 정확도)	Boundary IoU (경계선 정확도)
Rider	0.8032	0.1218
My bike	0.9216	0.2632
Moveable	0.5958	0.1040
Lane Mark	0.3256	0.1299
Road	0.8325	0.2155
Undrivable	0.9373	0.1001

Train Loss: 0.2692  
Val Loss: 0.3225

# Model Train/Validation 지표

WanDB



# Post-Processing : DenseCRF(Todo)

- SegFormer-B2 output에 DenseCRF를 적용 : mIoU/mBoU 높이기
- DenseCRF 이란?
  - "이웃 픽셀은 비슷해야 한다"는 조건부 확률 그래프 모델
- DenseCRF 추론 과정 (단계별 시각화) : SegFormer → CRF iteration → 최종 결과
- 핵심 정리

1. SegFormer-B2는 빠르고 정확  
→ 하지만 경계가 약간 흐림
2. DenseCRF는 경계를 정제  
→ 픽셀 간 관계성을 학습 (mBoU 향상)
3. 조합의 강점  
SegFormer-B2 속도 + DenseCRF 정확도  
= 실시간 고정확도 세그멘테이션

- 개선 예시 (AI 제안)

모델	Train Loss	mIoU	mBoU
SegFormer-B2	0.2570	0.7545	0.1470
+ DenseCRF	0.2570	0.7812	0.2234
개선율	-	+3.6%	+52.1%



# Hybrid Loss를 통한 정밀도 극대화(Todo)

현재 **Weighted CrossEntropy**를 사용 중이지만, 여기에 경계선 강조 손실(Edge-aware Loss)이나 Boundary Loss' 를 추가로 결합(Hybrid)하여 현재 15% 수준인 mBoU를 끌어올려야 합니다.

$$Total\ Loss = L\_WCE\ (\text{전체 파악}) + L\_Boundary(\text{테두리 정밀}) + L\_Attention(\text{집중도})$$

이 로스들을 섞어서(Total Loss) 학습하면 모델이 다음과 같이 변합니다.

- **Edge-aware Loss:** "색깔이 확 변하는 테두리를 더 세밀하게 관찰해!"
- **Boundary Loss:** "정답 선에서 1픽셀이라도 벗어나면 벌점을 더 세게 줄 거야!"
- **Attention-Guided:** "대충 보지 말고, 작고 얇은 차선을 뚫어지게 쳐다봐!"

# 모델 평가 개요 및 학습 전략

- 학습 설정
  - 이미지 : 학습(70%), 검증(20%), 테스트(10%)
  - Optimizer , learning rate, batch size, epoch 수
  - Loss function : Cross-Entropy
- 학습 전략: 학습 과정 중 **mIoU**와 **Boundary IoU(mBoU)** 지표에서 각각 **Best**를 기록한 두 가지 모델 파라미터를 저장
- 평가 프로세스: 저장된 두 모델(Best mIoU, Best mBoU)의 성능을 테스트 세트로 각각 평가
- 주요 지표: Latency, Confusion Matrix

# [평가 결과 1] 모델 정보 및 실시간성 분석

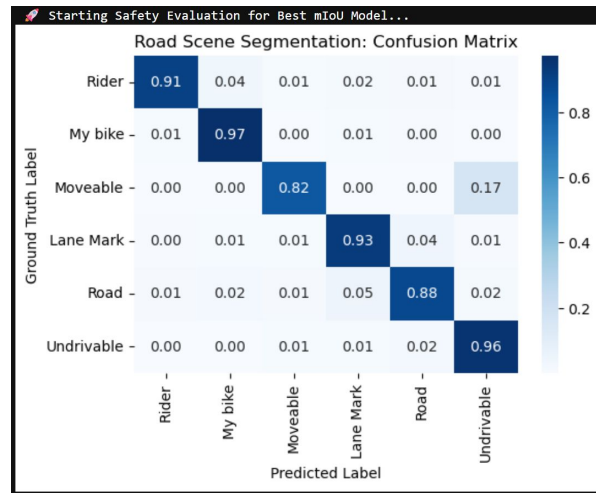
- 공통 : Parameters 27.35M , MACs 55.88G , GFLOPs 111.75G
- 추론 성능 비교 (LMS 환경) :
  - Best mIoU 모델: Latency **52.13ms** / **19.18 FPS**
  - Best mBoU 모델: Latency **52.48ms** / **19.05 FPS**

- **MACs** Multiply-Accumulate Operations
- **GFLOPs** Giga Floating Point Operations

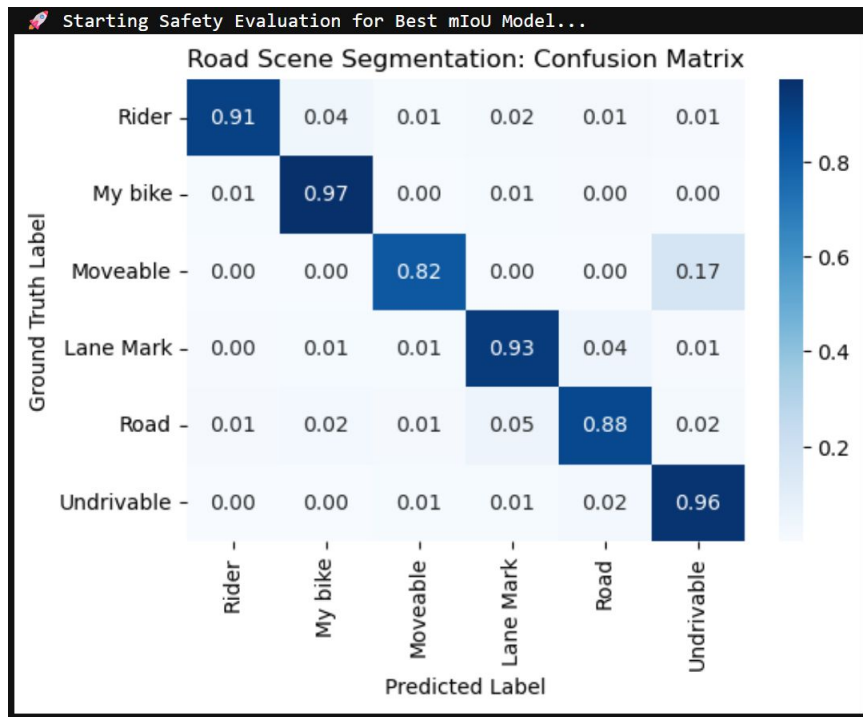
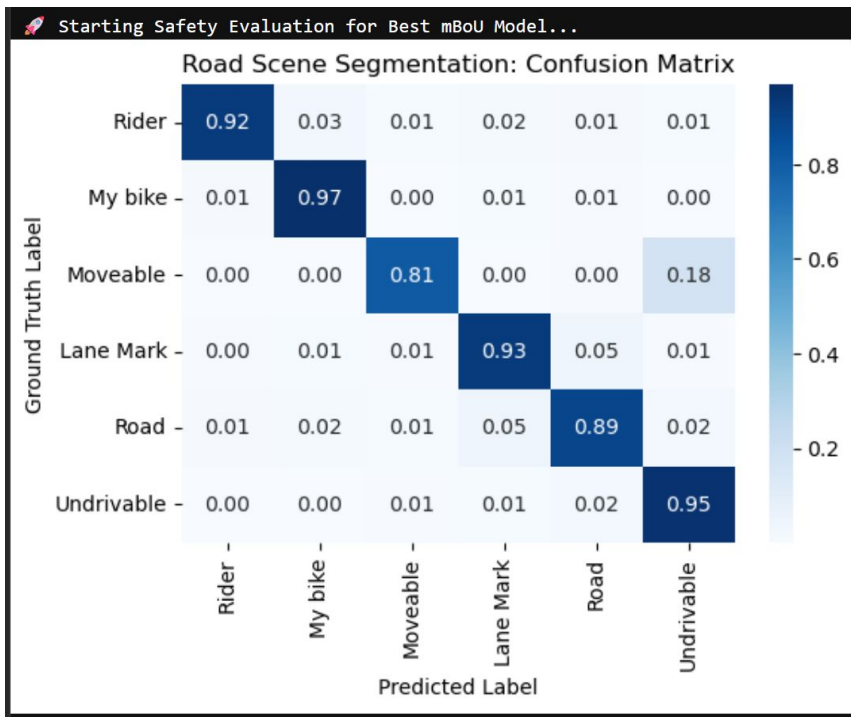
## [평가 결과 2-1] Safety-Critical Analysis (종합 분석)

1. F1-Score 0.88 ~ 0.95의 안정적인 분할 성능
2. Moveable ↔ Undrivable 오분류
  - Moveable 클래스의 재현율이 0.82로 가장 낮음. Moveable의 17%를 Undrivable로 오인
  - 이동 가능한 장애물(적치물, 보행자 등)을 주행 불가능한 배경으로 처리
  - Undrivable의 정밀도 (Precision) 저하
3. Lane Mark & Road 간의 혼동:
  - Road의 5%가 Lane Mark로, Lane Mark의 4%가 Road로 오분류

클래스	재현율 (Recall)	정밀도 (Precision)	F1-Score
Rider	0.91	0.98	0.94
My bike	0.97	0.93	0.95
Moveable	0.82	0.95	0.88
Lane Mark	0.93	0.91	0.92
Road	0.88	0.93	0.90
Undrivable	0.96	0.82	0.88



## [평가 결과 2-2] Safety-Critical Analysis (종합 분석)



# 향후 과제: 고도화된 실시간 HUD 시스템 구축

## 1. 연구의 한계점 및 극복 방안

- 현재: 개발자 관점의 정성적 분석(Grad-CAM) 및 정량 지표(mIoU) 확보 완료.
- 향후: 라이더(사용자) 관점의 직관적인 HUD(Head UP display) 시뮬레이션 구현 필요.

## 2. 핵심 구현 항목 (Obstacle & Lane)

- 지능형 장애물 감지 (Dynamic Bounding):
  - 픽셀 단위 인식을 넘어, 이동 객체(차량, 보행자)를 실시간 박스 처리하여 고해상도 시야 내 가시성 확보.
- 정밀 차선 가이드 (Smart Path Projection):
  - 야간에 보이지 않는 차선을 디지털 라인으로 투사하여 차선 이탈 방지 보조 기능 고도화.

## 3. Safety-First 실전 적용

- 리스크 기반 알람 시스템:
  - AI 판단 수치(1.36%의 충돌 위험 등)에 따라 위험도가 높을 경우 HUD 색상 변경 및 경고음 연동

감사합니다