



2024-1 파란학기 기업제안 프로젝트

팀 노브레이크

-프로젝트 성과발표-



백승훈 (교통 20)

박종일 (건설 18)

이현성 (교통 20)

김정엽 (교통 20)

목차



2024-1 파란학기 기업제안 프로젝트

01

프로젝트 목적

02

주제 관련 설문조사

03

프로젝트 과정

04

추후 목표



도로상에서는 **교통사고 및 고장으로 인한 정차, 정체로 인한 서행, 보행자 출몰, 낙하물 등 다양한 돌발상황이 발생**하고 있음. 이러한 돌발상황의 발생 시 빠른 시간 내에 감지한다면 지능형 교통체계(Intelligent Transport Systems, ITS), 차세대 지능형 교통체계(Cooperative-ITS, C-ITS) 등을 통해서 후방 차량에게 신속하게 전파할 경우 **추가적인 교통사고 등을 예방**할 수 있음.

따라서, 본 과제에서는 **고속도로 상에서 발생하는 돌발상황을 정의하고, 정의된 돌발상황 중 일부를 감지할 수 있는 AI 알고리즘을 개발**하고자 함. 이러한 돌발상황 감지 알고리즘 개발을 위해서 먼저, 도로를 주행하는 차량 내 부착된 단말기에서 수집된 이미지 데이터를 전처리하고, 기존에 부여된 라벨을 수정 및 보완하고, CNN 등을 이용한 감지 알고리즘을 개발한 후, 개발된 알고리즘의 성능을 평가하고자 함.

왜 구글폼을 제작하였는가?



도로 상의 돌발상황을 정의하기에 앞서
비전공자가 생각하는 돌발상황에 대한 궁금증 생겨
간단한 설문조사를 통해 돌발상황에 대한 인식 파악



구글폼에 작성한 질문들



교통사고 상황이 돌
발상황에 포함된다
고 생각하는지?

고장차량/ 도로시설물
파손이 돌발상황에 포함
된다고 생각하는지?

위험물질 유출/ 동물출현
상황이 돌발상황에 포함된
다고 생각하는지?

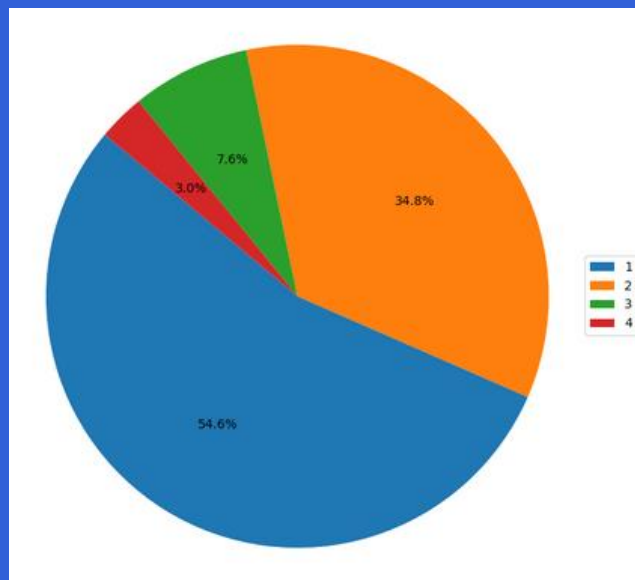
02 주제 관련 설문조사



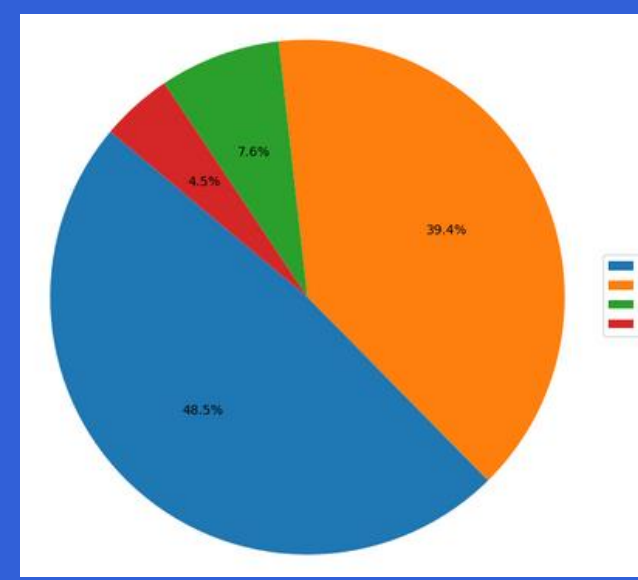
2024-1 파란학기 기업제안 프로젝트

설문조사 결과 및 해석

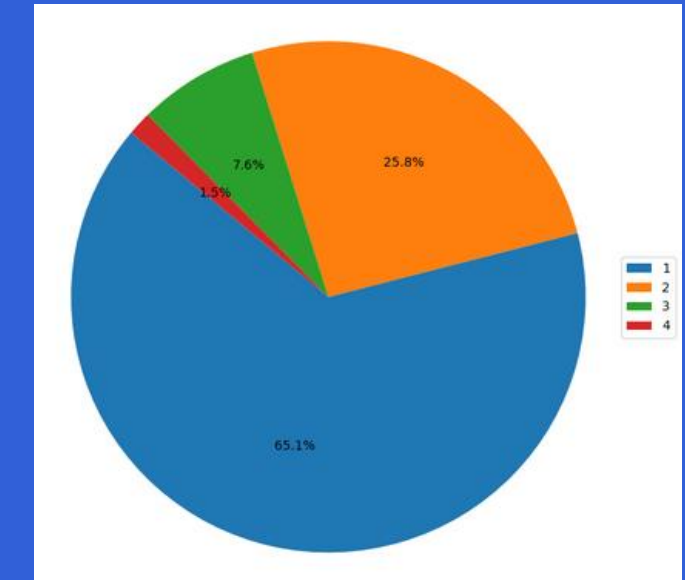
교통사고 상황



고장차량 / 도로시설물 파손



위험물질 유출/ 동물출현 상황



1. 매우 적합하다
2. 약간 적합하다
3. 그냥 그렇다
4. 약간 적합하지 않다.

구글폼 결과에 대한 해석

1. 비전공자들도 돌발상황에 대한 정의 확실
2. 위험물질 / 동물출현을 가장 큰 돌발상황으로 판단
3. 고장차량/ 도로시설물 파손을 제일 혼동



프로젝트 과정



2024-1 파란학기 기업제안 프로젝트

1. 이미지 라벨링

2.CNN & ResNet &YOLO

3.최적의 하이퍼파라미터 튜닝

4.사전학습 모델링

5.전처리 과정

6.Test data 수집

7.viT 모델링

03 프로젝트 과정

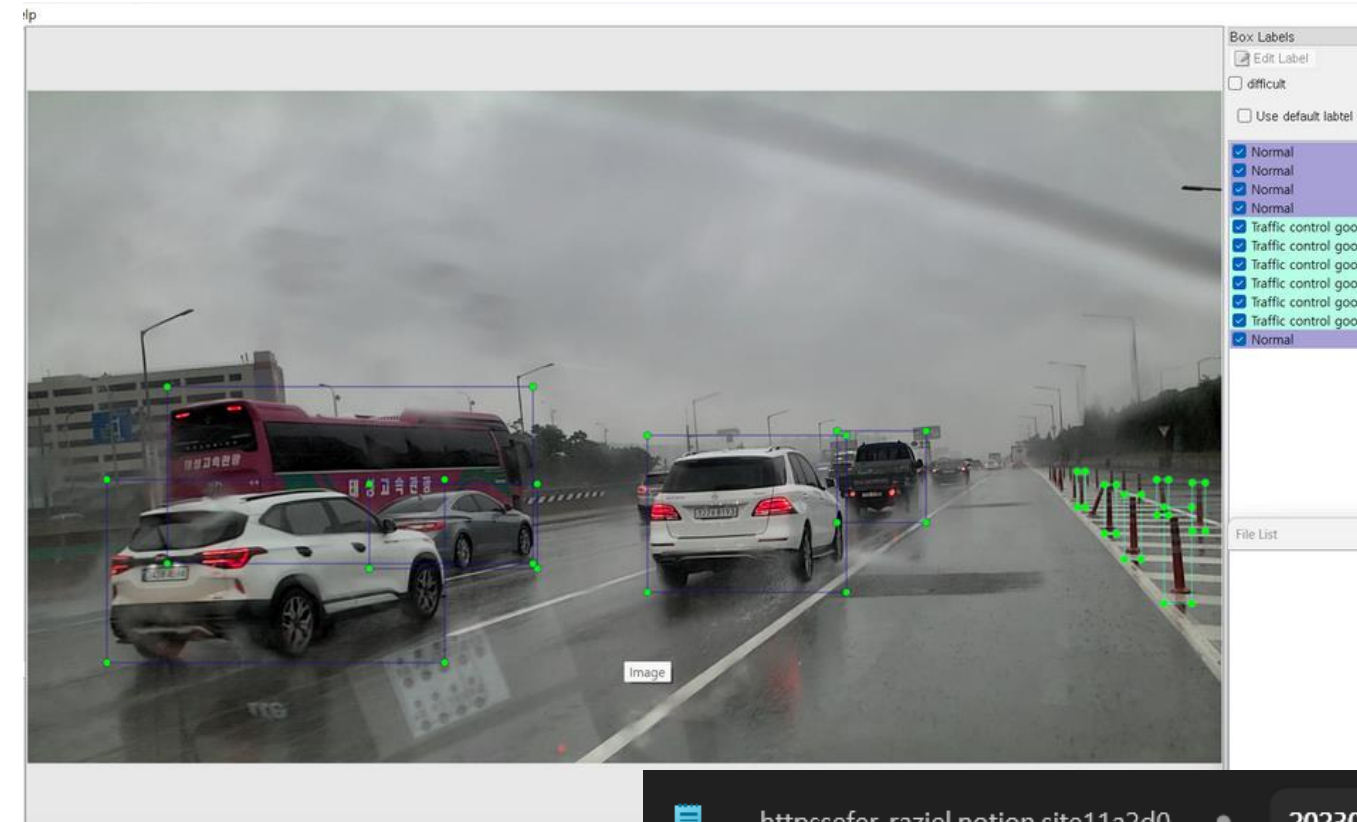


2024-1 파란학기 기업제안 프로젝트

1. 이미지 라벨링

데이터 클래스

1. 사고/고장상황
2. 정상상황
3. 도로 위 보행자
4. 도로 위 낙하물
5. 공사 중
6. 정체상황



```
httpssefer-raziel.notion.site11a2d0 202309_87a56263d1be467b89b14c × +
파일 편집 보기

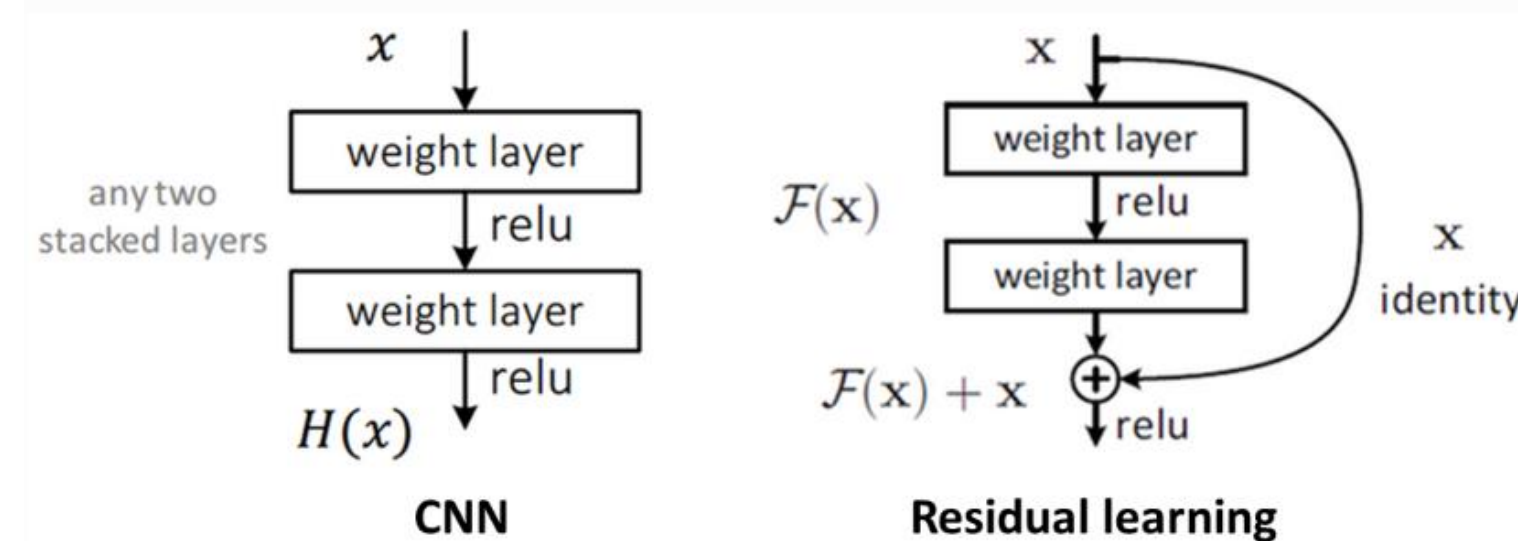
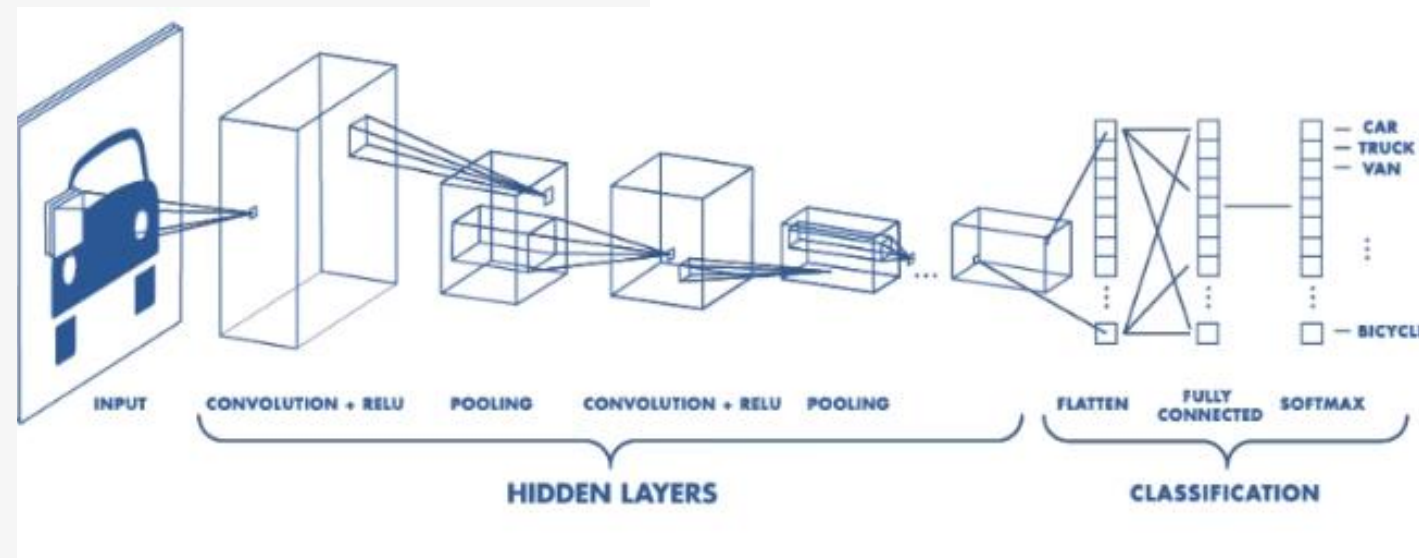
3
6 0.776823 0.555093 0.024479 0.091667
5 0.859375 0.602315 0.038542 0.180556
5 0.918229 0.625463 0.040625 0.226852
7 0.935417 0.768056 0.104167 0.167593
0 0.498698 0.615278 0.096354 0.110185
0 0.720052 0.556481 0.033854 0.055556
```


03 프로젝트 과정

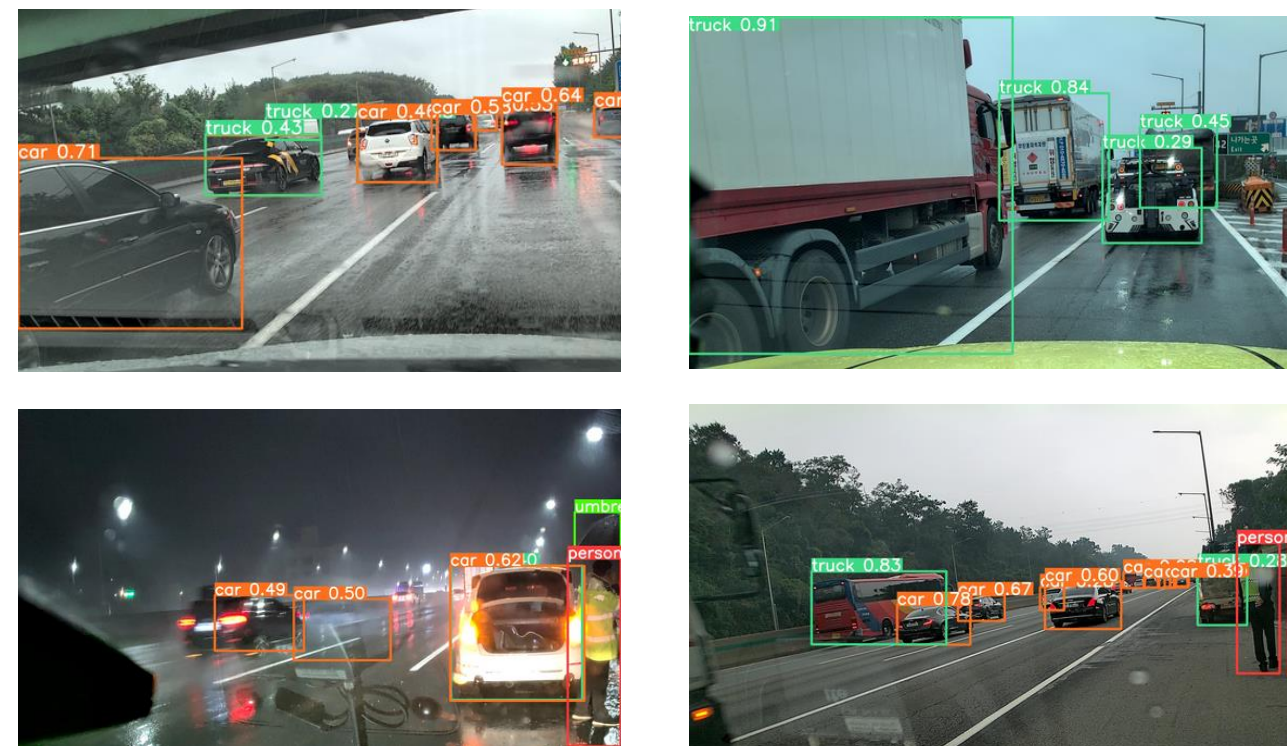


2024-1 파란학기 기업제안 프로젝트

2. CNN & ResNet & YOLO

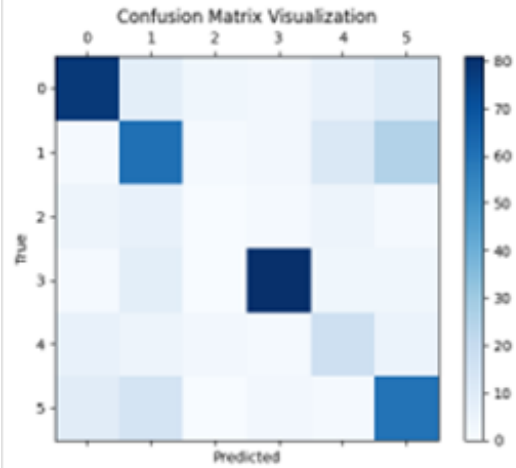
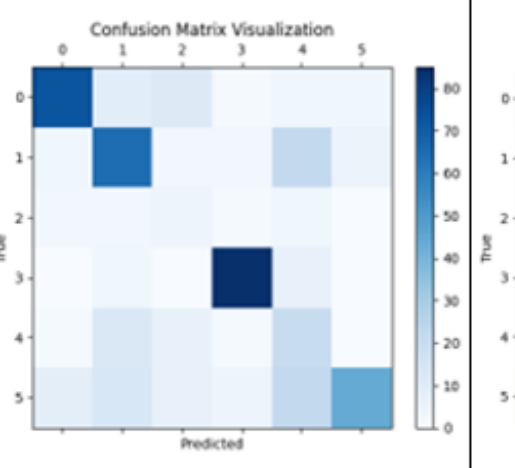
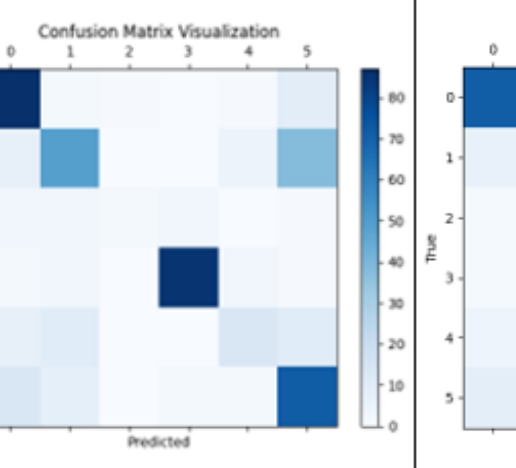
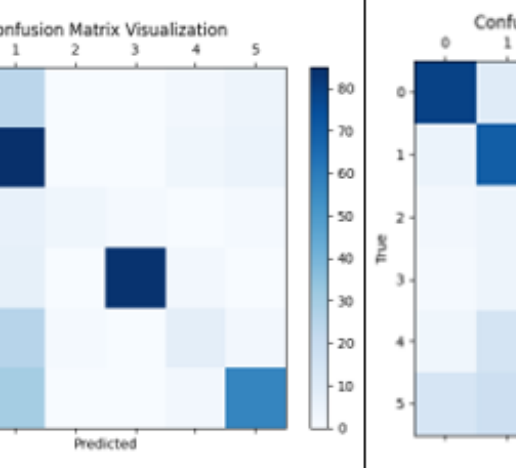
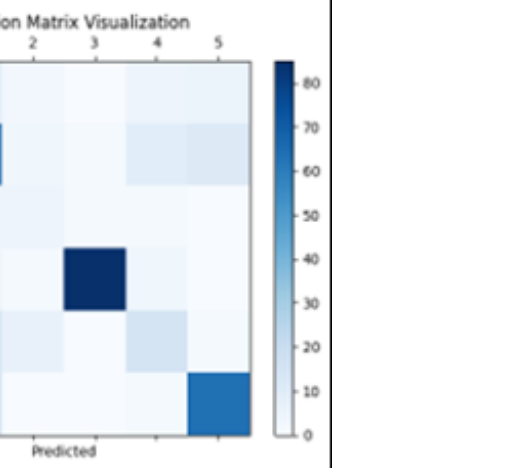
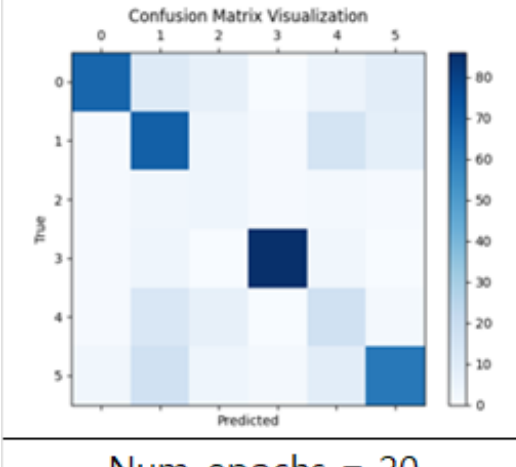
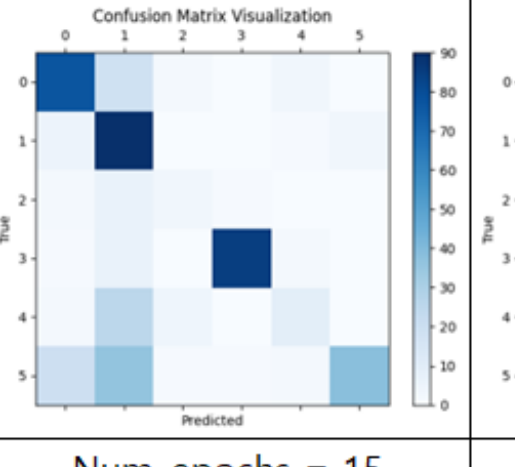
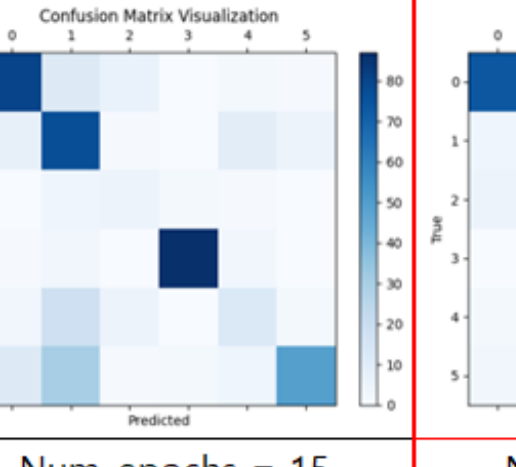
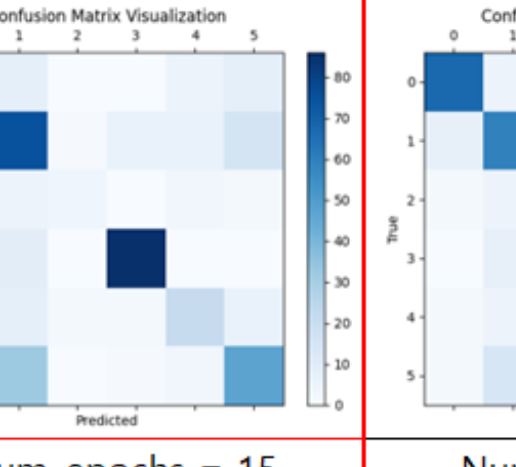
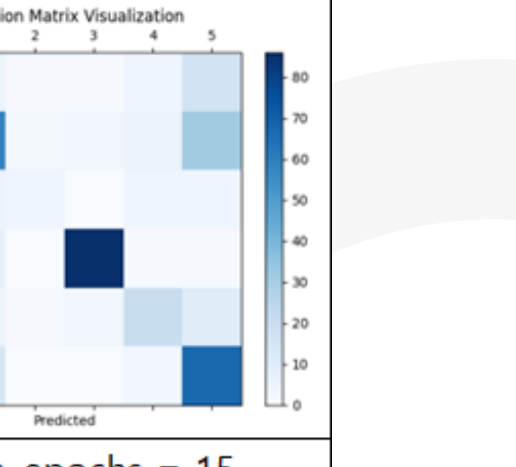


YOLO 알고리즘



3. 최적의 하이퍼파라미터 튜닝

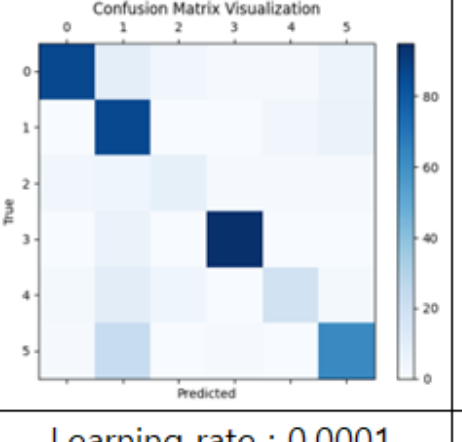
CNN & ResNet 파라미터 조정 결과 표

				
Num_epochs = 10 Learning rate = 0.001 Batch_size = 32 Optimizer = ADAM F1-score : 0.55	Num_epochs = 10 Learning rate = 0.001 Batch_size = 32 Optimizer = RMSprop F1-score : 0.58	Num_epochs = 10 Learning rate = 0.0001 Batch_size = 32 Optimizer = RMSprop F1-score : 0.60	Num_epochs = 10 Learning rate = 0.0001 Batch_size = 32 Optimizer = Adam F1-score : 0.61	Num_epochs = 15 Learning rate = 0.0001 Batch_size = 32 Optimizer = Adam F1-score : 0.63
				
Num_epochs = 20 Learning rate = 0.0001 Batch_size = 32 Optimizer = Adam F1-score : 0.61	Num_epochs = 15 Learning rate = 0.0001 Batch_size = 64 Optimizer = Adam F1-score : 0.58	Num_epochs = 15 Learning rate = 0.0001 Batch_size = 32 Optimizer = RMSprop F1-score : 0.61	Num_epochs = 15 Learning rate = 0.001 Batch_size = 32 Optimizer = Adam F1-score : 0.63	Num_epochs = 15 Learning rate = 0.001 Batch_size = 32 Optimizer = RMSprop F1-score : 0.62

<CNN 파라미터 정리표>

3. 최적의 하이퍼파라미터 튜닝

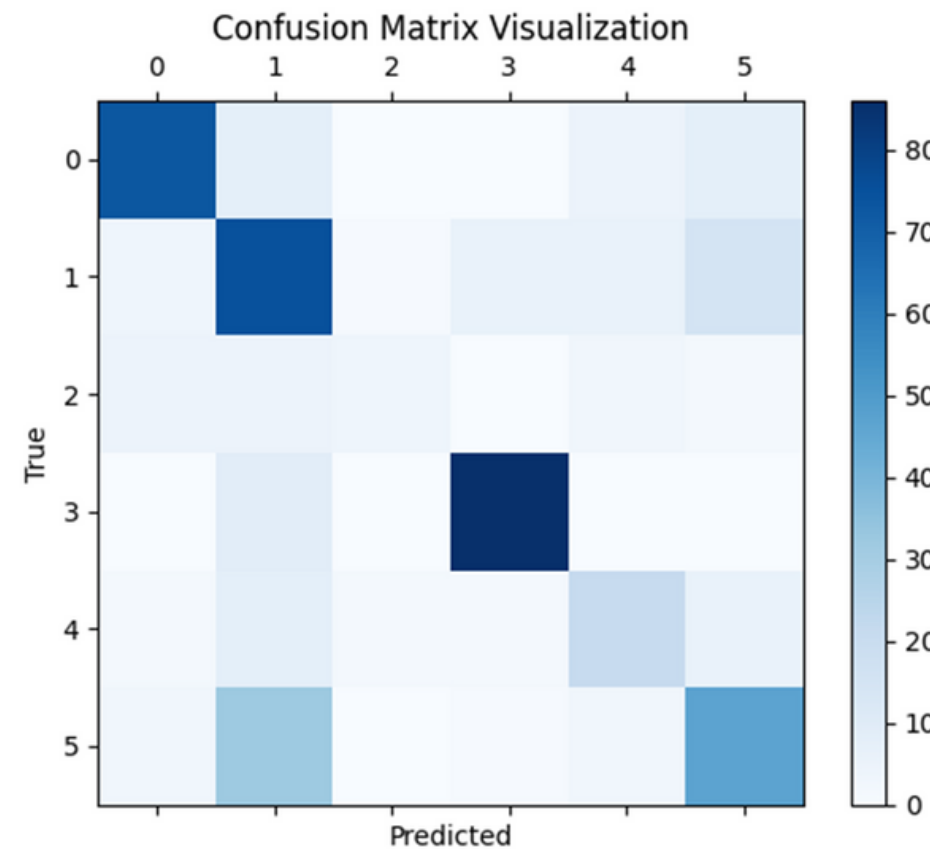
CNN & ResNet 파라미터 조정 결과 표

			
Learning rate : 0.001 Epochs : 10 Optimizer: Adam Batch size : 32 F1 score: 0.64 Resnet 34	Learning rate : 0.001 Epochs : 10 Optimizer: AdamW Batch size : 32 F1 score: 0.55 Resnet 34	Learning rate : 0.001 Epochs : 10 Optimizer: RMSprop Batch size : 32 F1 score: 0.51 Resnet 34	Learning rate : 0.0001 Epochs : 10 Optimizer: Adam Batch size : 32 F1 score: 0.67 Resnet 34
			
Learning rate : 0.0001 Epochs : 15 Optimizer: Adam Batch size : 32 F1 score: 0.75 Resnet 34	Learning rate : 0.0001 Epochs : 15 Optimizer: Adam Batch size : 32 F1 score: 0.74 Resnet 50	Learning rate : 0.0001 Epochs : 10 Optimizer: Adam Batch size : 64 F1 score: 0.73 Resnet 34	Learning rate : 0.0001 Epochs : 15 Optimizer: Adam Batch size : 64 F1 score: 0.70 Resnet 34

<ResNet 파라미터 정리표>

3. 최적의 하이퍼파라미터 튜닝

CNN & ResNet
최적의 파라미터
조정 결과



CNN

learning rate : 0.0001

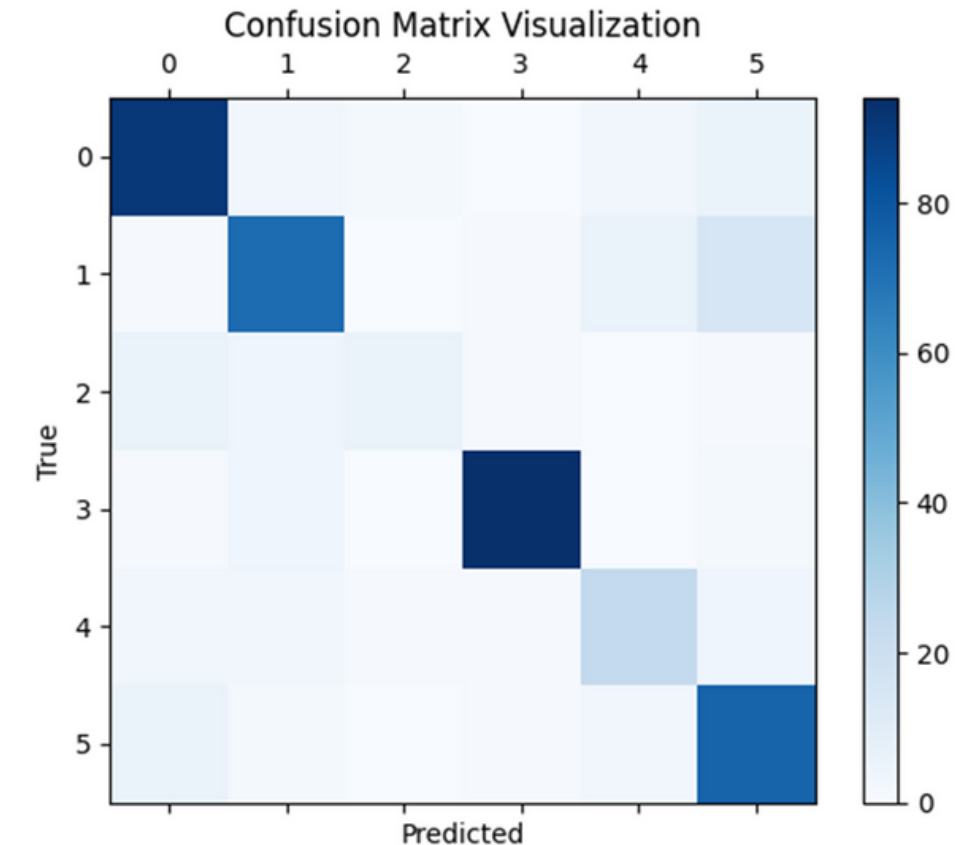
epochs : 15

optimizer : adam

batch size : 32

best accuracy : 70.1357

f1-score : 0.63



ResNet 34

learning rate : 0.0001

epochs : 15

optimizer : adam

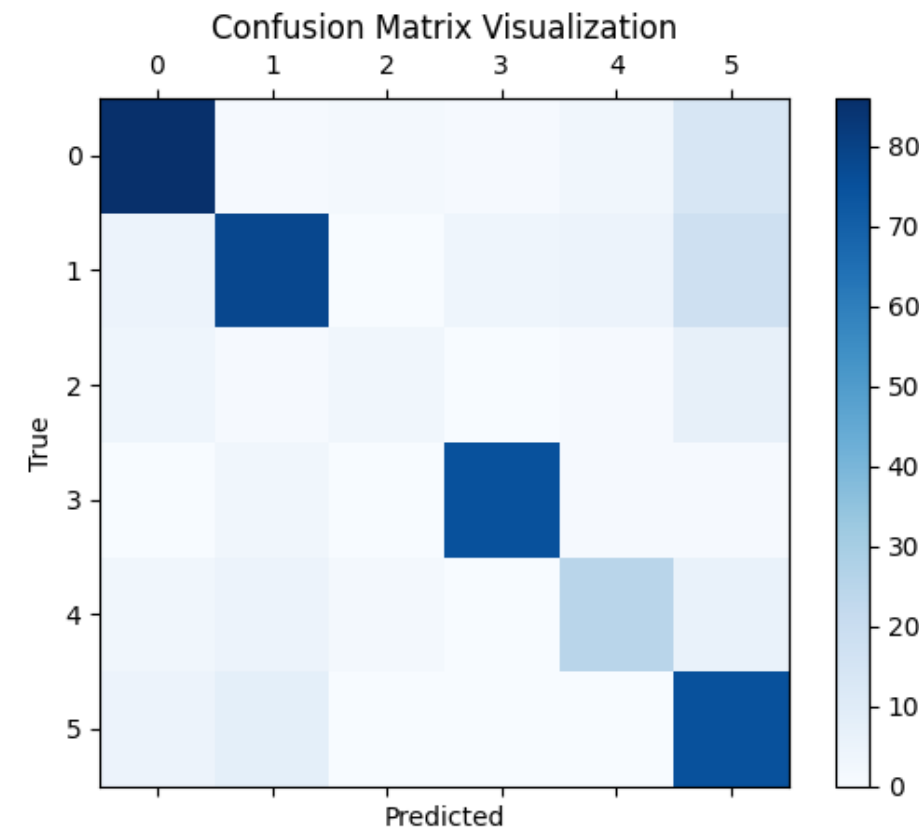
batch size : 32

best accuracy : 83.48

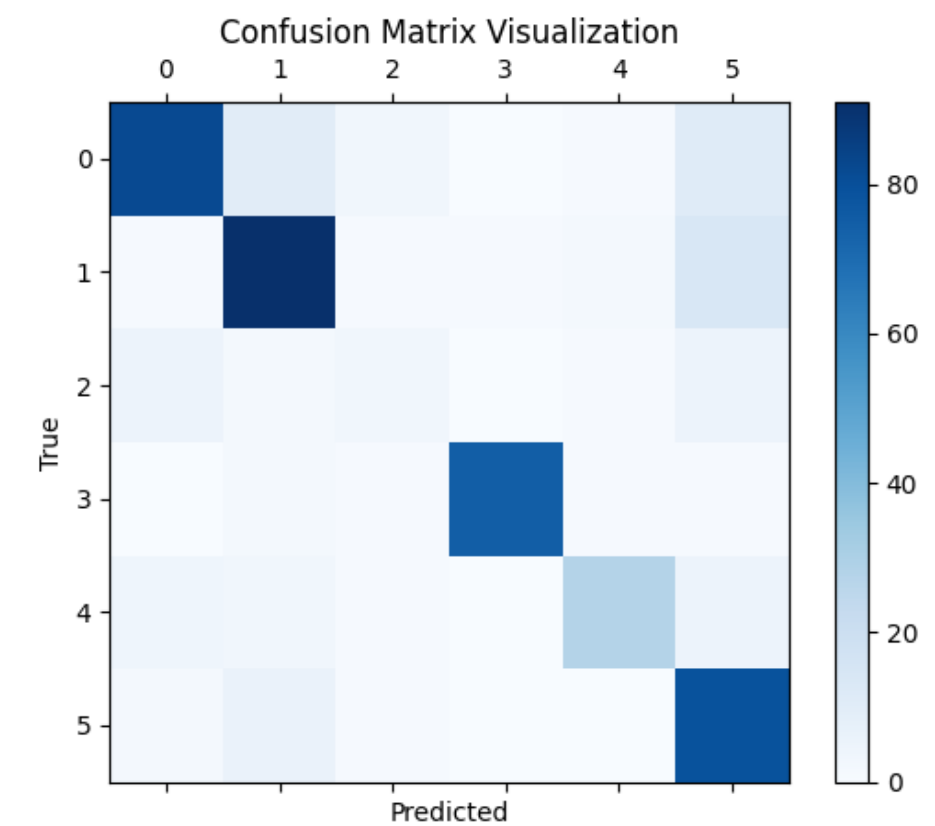
f1-score : 0.75

4. 사전학습 모델링

Google Net &
Mobile Net 결과



Google Net
f1-score : 0.69
best accuracy : 78.0543



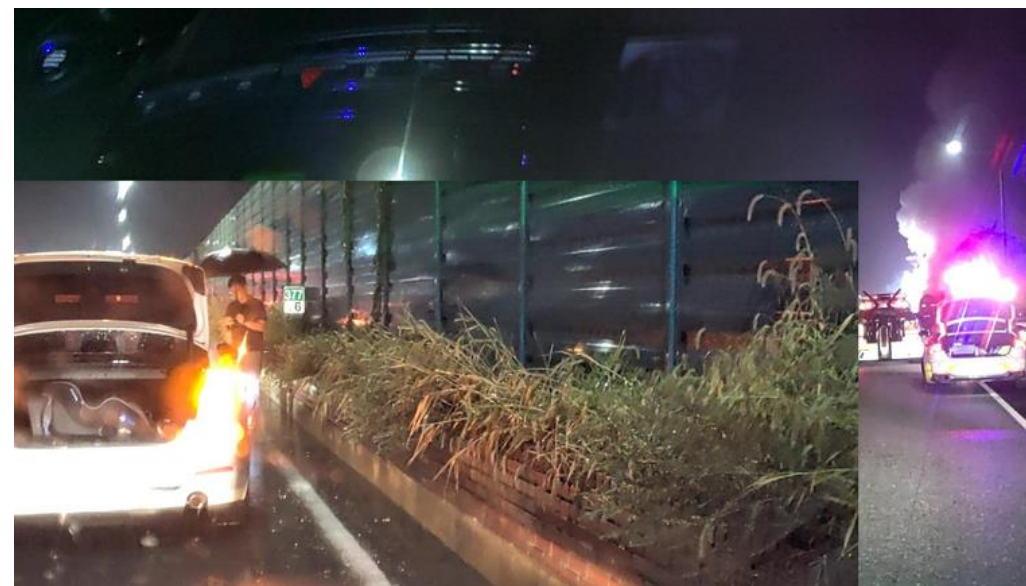
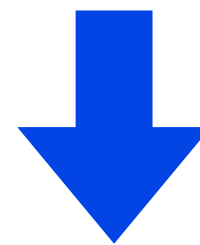
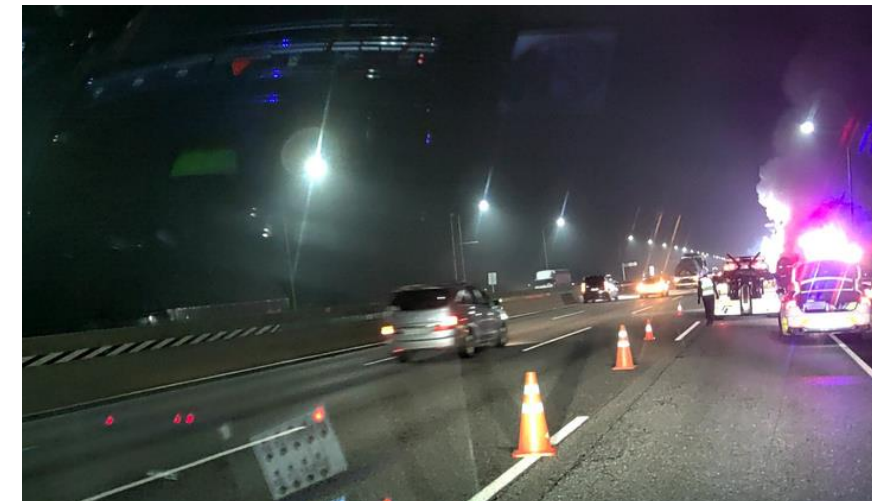
Mobile Net
f1-score : 0.73
best accuracy : 82.5791

5. 전처리 과정

CNN

최적의 파라미터 조정
데이터 전처리 및 증강

CutMix

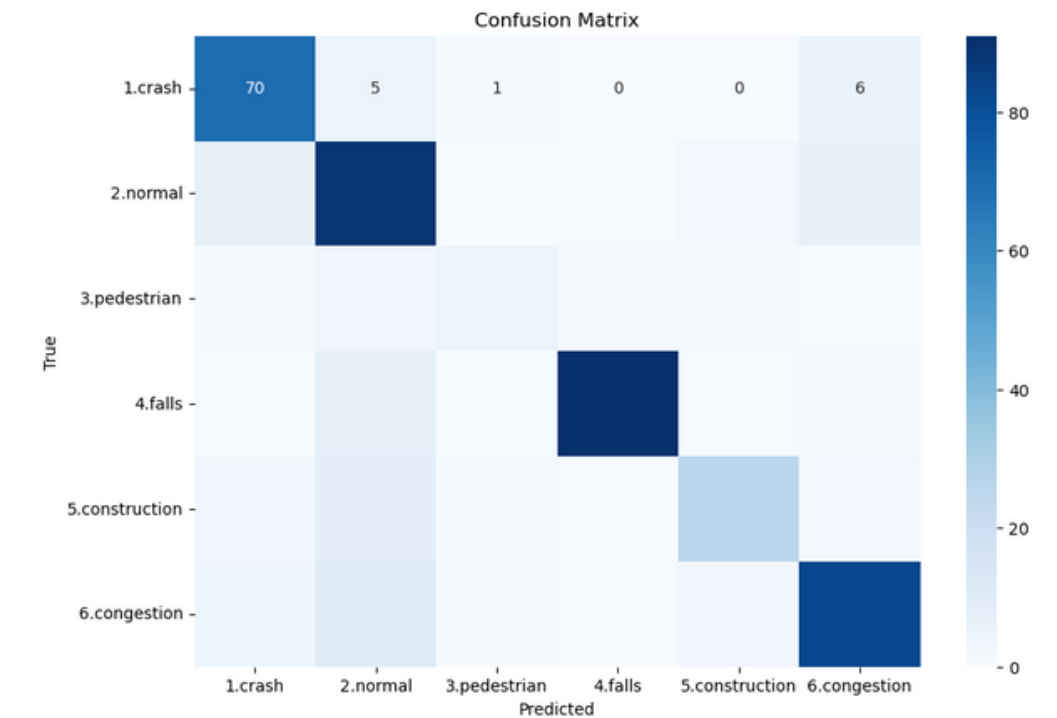
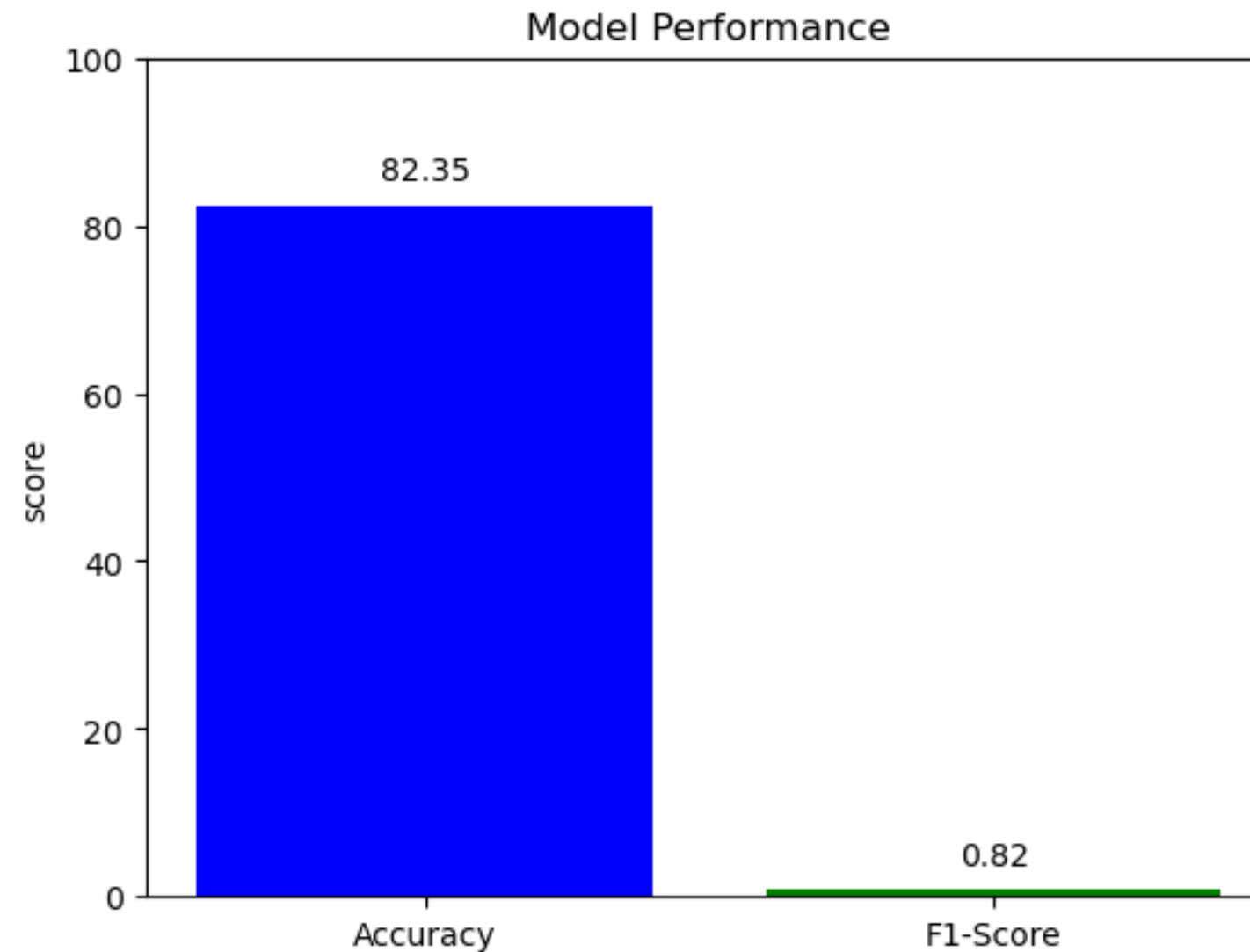


Epoch 10, Loss: 0.39856713784463477
Accuracy after epoch 10: 69.00452488687783%
F1-Score after epoch 10: 0.6683402502567904
Epoch 11, Loss: 0.3305983529849486
Accuracy after epoch 11: 70.81447963800905%
F1-Score after epoch 11: 0.6960972700582972
Epoch 12, Loss: 0.28072667663747614
Accuracy after epoch 12: 72.62443438914028%
F1-Score after epoch 12: 0.714408582457888
Epoch 13, Loss: 0.16714200535506912
Accuracy after epoch 13: 70.13574660633485%
F1-Score after epoch 13: 0.6860800608608072
Epoch 14, Loss: 0.1310290094521461
Accuracy after epoch 14: 70.81447963800905%
F1-Score after epoch 14: 0.6965224937587582
Epoch 15, Loss: 0.11168557761067693
Accuracy after epoch 15: 71.26696832579185%
F1-Score after epoch 15: 0.703486814230387

5. 전처리 과정

Efficient Net

최적의 파라미터 조정+
데이터 전처리



Epoch 15, Loss: 0.060348898264099705

Accuracy: 82.35%

F1-Score: 0.8236

Confusion Matrix:

```
[[70  5  1  0  0  6]
 [ 8 89  0  0  2  7]
 [ 1  3  5  1  1  0]
 [ 0  8  0 91  0  1]
 [ 3  9  1  0 26  2]
 [ 4 11  1  0  3 83]]
```


03 프로젝트 과정



2024-1 파란학기 기업제안 프로젝트

6. Test data 수집



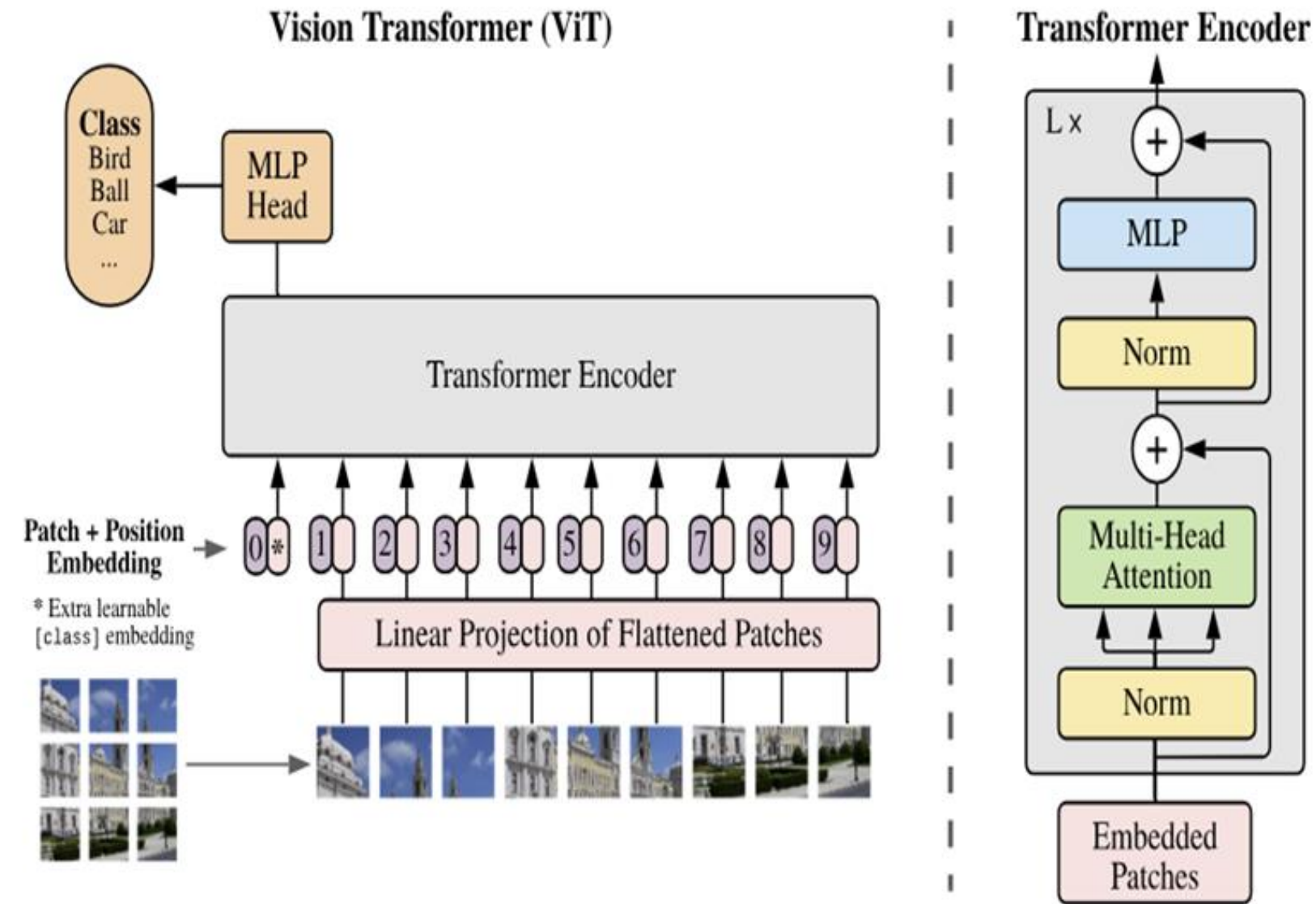
Test Data를 **비오는 상황**에 직접 수집하여
Robust 모델을 만들고자 함

7. ViT 모델링

SOTA

-State Of The Art

<ViT> 모델의 원리



<Vision Transformer (ViT) 모델의 전체 구조>

1. 패치 및 위치 정보 임베딩:

각 이미지 패치가 벡터로 평탄화 -> 각 패치의 위치 정보를 포함하는 위치 임베딩이 추가 -> Transformer의 입력으로 사용

2. Transformer 인코더로 전달:

Multi-Head Attention: 각 패치가 이미지의 다른 패치와 어떻게 상호 작용하는지 학습

정규화 (Norm) 및 MLP (Multi-Layer Perceptron): 패치의 특징을 further process하여 더 복잡한 표현을 학습

3. MLP Head로 전달:

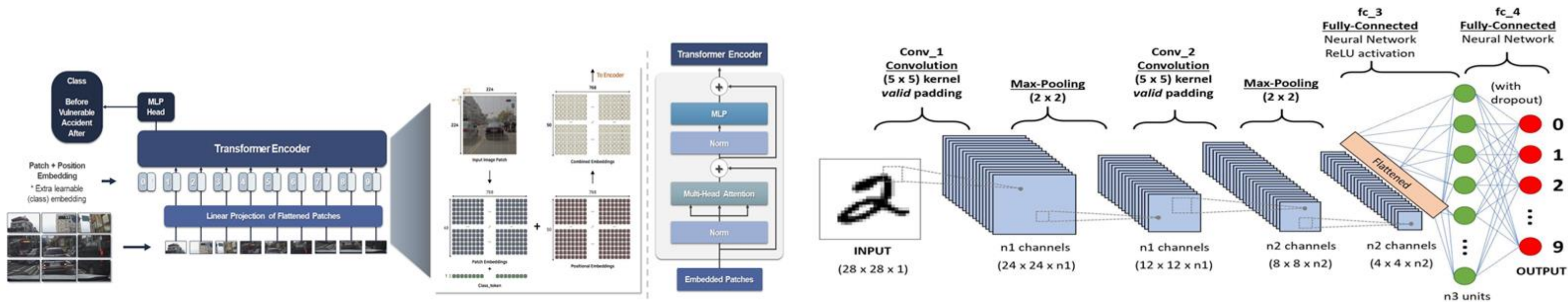
최종 Transformer 인코더 레이어의 출력을 MLP 헤드로 전달 -> 최종적인 분류 결정

4. 클래스 예측:

최종 클래스 레이블(객체명)로 변환

7. ViT 모델링

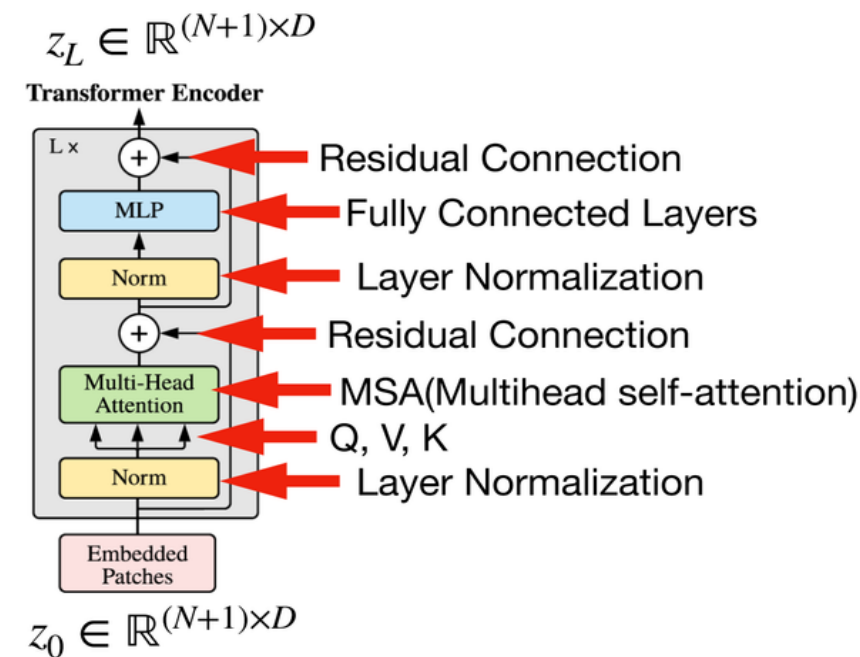
ViT vs CNN



특징	ViT (Vision Transformer)	CNN (Convolutional Neural Network)
처리방식	자연어 처리 방식을 기반으로 하여, 이미지를 패치로 나눈 후 각 패치를 토큰으로 취급하여 학습	연속적인 합성곱 레이어가 이미지의 지역적 특성을 추출
학습 맥락	전체 이미지의 맥락을 통합적으로 학습, 전체 이미지 구조에 대한 이해 증진	주로 지역적 특징에 집중
병렬처리	Multi-Head Attention을 통한 효율적인 병렬 처리, 빠른 학습 가능	병렬 처리는 제한적, 대개 연속적인 연산이 필요
성능	이미지 전체에 대한 관계와 맥락을 파악하여 더 정교한 패턴 인식 가능	간단한 패턴 및 특징 추출에 효과적, 복잡한 맥락에서는 제한적
적용측면	대규모 이미지 데이터셋에서 뛰어난 성능 발휘, 복잡한 장면 이해에 유리	기본적인 이미지 분류 및 객체 인식에서 강력하지만, 전체적인 맥락 파악은 떨어짐

7. ViT 모델링

ViT 모델의 프로젝트 적용



val_loss: 0.6264208436012269, val_acc: 0.8201286196708679
training_loss: 0.640850231051445

ViT 모델 성능 결과

→ val_acc가 **0.82**로 앞선 모델들과 비교하여 뛰어난 성능을 보임.

1. 데이터 로딩 및 전처리:

- 이미지 데이터 크기를 28x28로 조정한 후 텐서로 변환
- 데이터셋을 훈련 세트와 검증 세트로 나눔.
- DataLoader를 사용하여 배치 단위로 데이터를 로드

2. 모델 구성:

- EmbeddingLayer: 입력 이미지를 패치로 나누고, 각 패치를 임베딩 벡터로 변환, 클래스 토큰 추가, 위치 정보 인코딩
- MultiheadAttention: 여러 헤드를 사용하여 다양한 측면에서 입력을 attention
- FeedForwardBlock: 피드포워드 신경망 블록으로 임베딩 벡터를 변환
- ViT: 비전 트랜스포머 블록을 구성하여 입력 데이터를 변환
- TransformerEncoder: 여러 개의 트랜스포머 블록을 쌓아 심층 신경망을 구성

3. 학습 및 평가:

- ViT_Encoder: PyTorch Lightning을 사용하여 학습, 검증, 테스트 단계를 정의
- 조기 종료 콜백을 설정하여 검증 손실이 개선되지 않으면 학습을 중단
- 트레이너를 사용하여 모델을 학습하고 평가

4. 테스트 데이터 확인:

- 테스트 데이터 로더에서 레이블의 형태를 출력하여 데이터가 제대로 로드되었는지 확인

TensorFlowLite 시스템 포팅 관련 연구 진행

>> 저희 단말기에서는 TensorFlowLite 모델로 돌리고있는데,

아주대학교에서 어떤 알고리즘과 프레임워크를 사용해서 모델을 학습시킬지 모르지만,

학습시킨 모델을 TensorFlowLite모델로 변환하여 사용한다면 아래 예제와 가이드문서로 안드로이드 어플리케이션에 충분히 포팅 할 수 있을것으로 생각됩니다.

*참고로 저희 모델 변환과정입니다

Darknet 프레임워크 학습 -> darknet 프레임워크 모델 (*.weights 모델) -> keras (*.h5)-> TFLite

TFLite 모델을 android 공식 가이드문서입니다.

문서 : https://www.tensorflow.org/lite/guide?hl=ko&_gl=1*_1mefxgc*_up*MQ..*_ga*_NDEwOTY4OTM0LjE3MTUwNDQ4Njk.*_ga_W0YLR4190T*_MTcxNTA0NDg2OC4xLjAuMTcxNTA0NDg2OC4wLjAuMA..

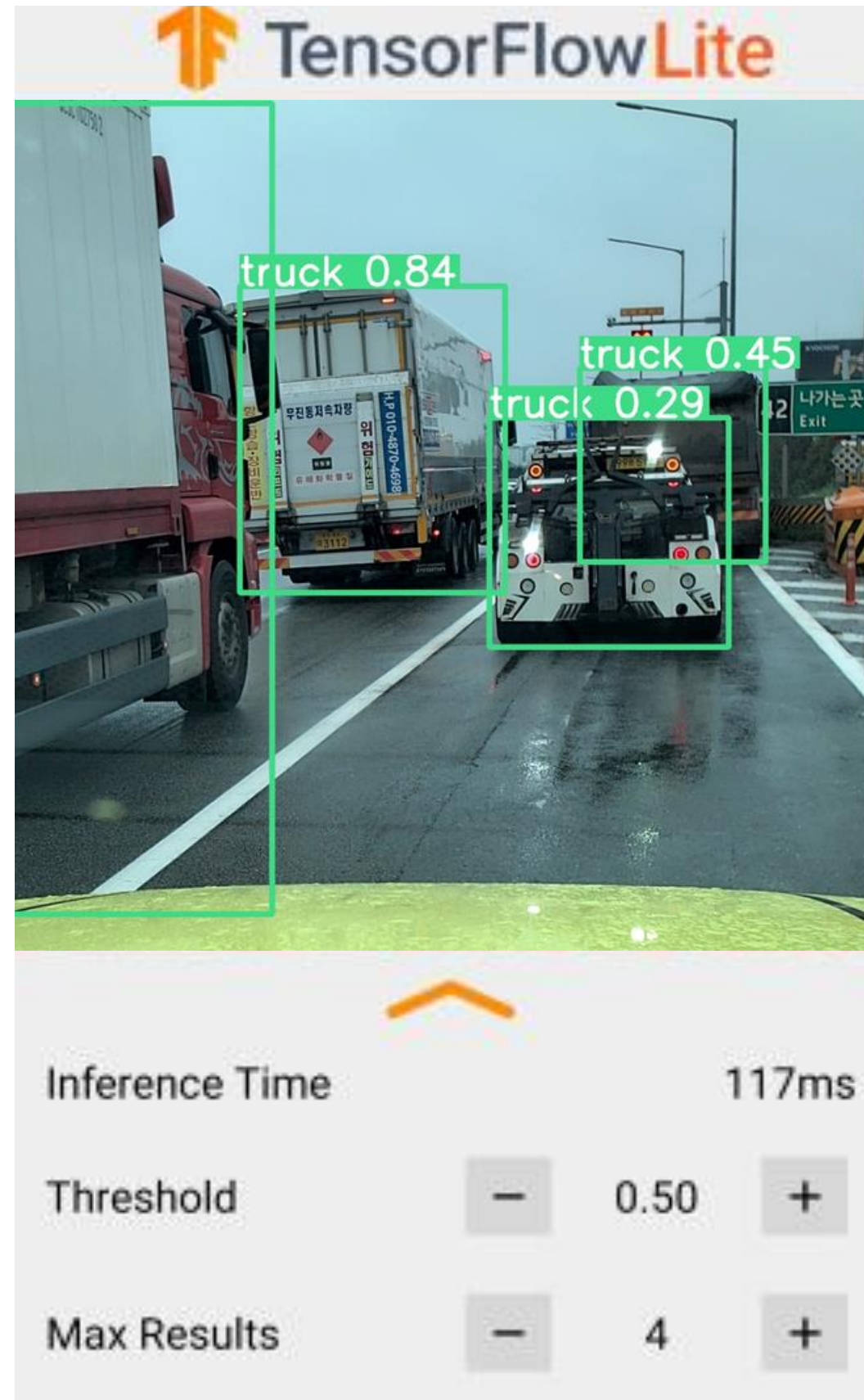
예제 : <https://www.tensorflow.org/lite/examples?hl=ko>

직접 기업에 메일을 보내 어떤 방법으로
시스템 포팅을 하는지 자문을 구함

기업에서 TensorFlowLite
모델 공식문서 및 **기업 자체 모델 변환 과정**
제공

우리는 이에 대한 내용을
직접 구현하는 것을 추후 프로젝트 방향성으로
설정하여, 공식 문서의 튜토리얼을 진행하였음

TensorFlowLite Fine Tuning



- 현재 딥러닝 모델을 이용하여, 고속도로에서의 돌발상황을 감지하는 알고리즘의 정확도 60%를 넘기는 목표는 달성이 되었음
- 결론적으로, 우리의 과제가 현실 세계에서 활용되기 위해서 Edge Device에 탑재되어야 하는 것은 필수적
- 사전 학습된 TensorFlowLite COCO SSD 모바일넷 v1 튜토리얼을 진행하여 **경량화 모델에 대한 연구를 진행중 이고 제공받은 데이터와 직접 수집한 데이터를 사용하여, 안드로이드 디바이스에서 Fine Tuning된 모델을 구축하는 것이 최종 목표**



2024-1 파란학기 기업제안 프로젝트

감사합니다