자율주행의 안전성을 높이는 딥러닝 기반 차선 검출

2ternals

팀장 : 손예린

팀원 : 김창현, 권희정, 어우섭, 이해성, 최정수

발표일 : 2022. 3. 25

목차 a table of contents

#Introduction

Introduction

Team Members

팀장 : 손예린
팀원 : 김창현, 권희정, 이우섭, 이해성, 최정수

Goal

안정성 문제를 개선하기 위해 컴퓨터비전과 딥러닝 CNN 모델을 활용하여 주행 중 빠르고 정확하게 차선을 인식 할 수 있는 프로그램 개발

Dataset

AI Hub에서 제공하는 자율주행 데이터
다양한 조건과 환경의 12개 도로주행 영상

Skills

Numpy	Scipy	OpenCV	TensorFlow	Matplotlib
10%	15%	30%	40%	5%

Work Rule

Python Style Guide : PEP8
Team Blog : https://2ternals.tistory.com
GitHub : https://github.com/ddunddunni/self-driving-2ternals

Roles and Responsibilities

손예린(PM)	권희정(DA)	김창현(TA)	이우섭(TA)	이해성(TA)
프로세스 총괄 일정관리 논문 및 자료 검색 데이터 전처리 모델 구축 및 테스트	데이터 수집 데이터 전처리 모델 구축 발표	데이터 수집 논문 및 자료 검색 모델 구축 발표자료 작성	논문 및 자료 검색 모델 성능 테스트 하이퍼파라미터 튜닝	논문 및 자료 검색 모델 성능 테스트 하이퍼파라미터 튜닝

Work Schedule

02.14

기획서 작성

역할분담, 개요,
추진방향 설정

02.15

계획서 작성

프로젝트 진행 과정, 데이터 샘플,
알고리즘, 평가 목록 설정

02.16

기획서&계획서 발표

피드백 기초로
기획서&계획서 수정

02.18

AI Hub

데이터 수집

교통표지판, 신호등,
차량, 차선 등의 이미지
데이터 수집

02.20

데이터 레이블링

02.22

초기 모델 구축

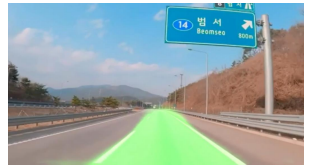
테두리 검출
이미지 변환
Noise Reduction

03.07

모델링&학습

CNN을 통해 여러 물체
들을 인식하도록 학습

03.10



이미지 & 영상 평가

이미지 & 영상 데이터를
통해 항목마다
인식하는 정도 평가

03.18

모델 수정

인식률이 낮은 객체
또는 환경을 파악한
후 모델 수정

03.22

발표자료 작성

#Early Model

Early Model

초기 모델 구성

BGR to HLS → Sobel Edge → Perspective Transformation

데이터

AI Hub 도로주행영상 (https://aihub.or.kr/aidata/8007)

결과

장점 : 고속도로, 실선 차선에서 비교적 정확한 차선 인식

문제점 : 터널구간, 점선 차선, 앞 차량과의 간격이 좁은 경우 차선인식 어려움

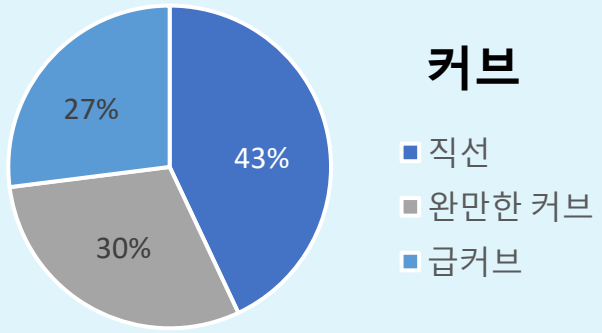
#Dataset

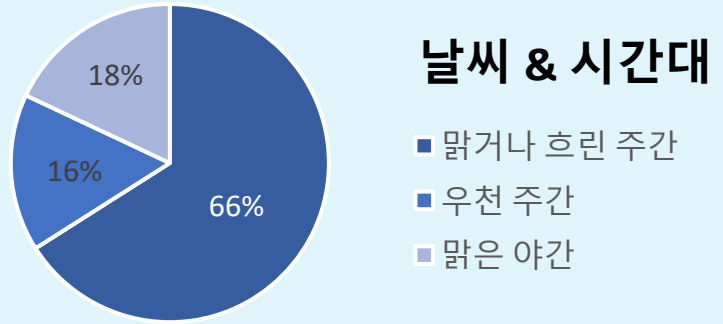
Dataset

학습 데이터

12개 비디오에서 추출된 1280x720 사이즈의 이미지 21,054장
→ 160 x 80 크기로 resize

데이터 구성





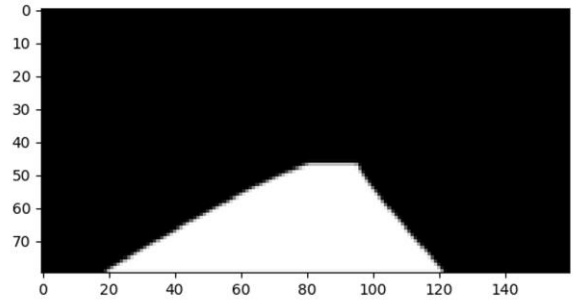
Preprocessing

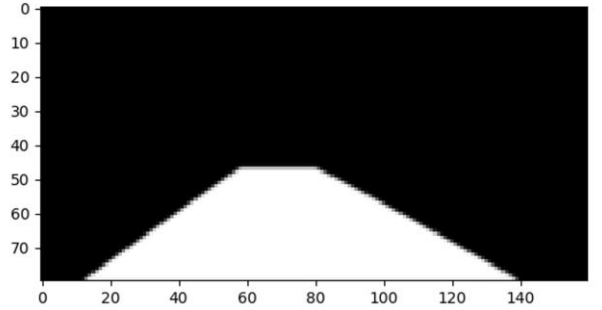
완만한 커브 &
맑거나 흐린 주간

직선 & 맑은 야간

원본 이미지

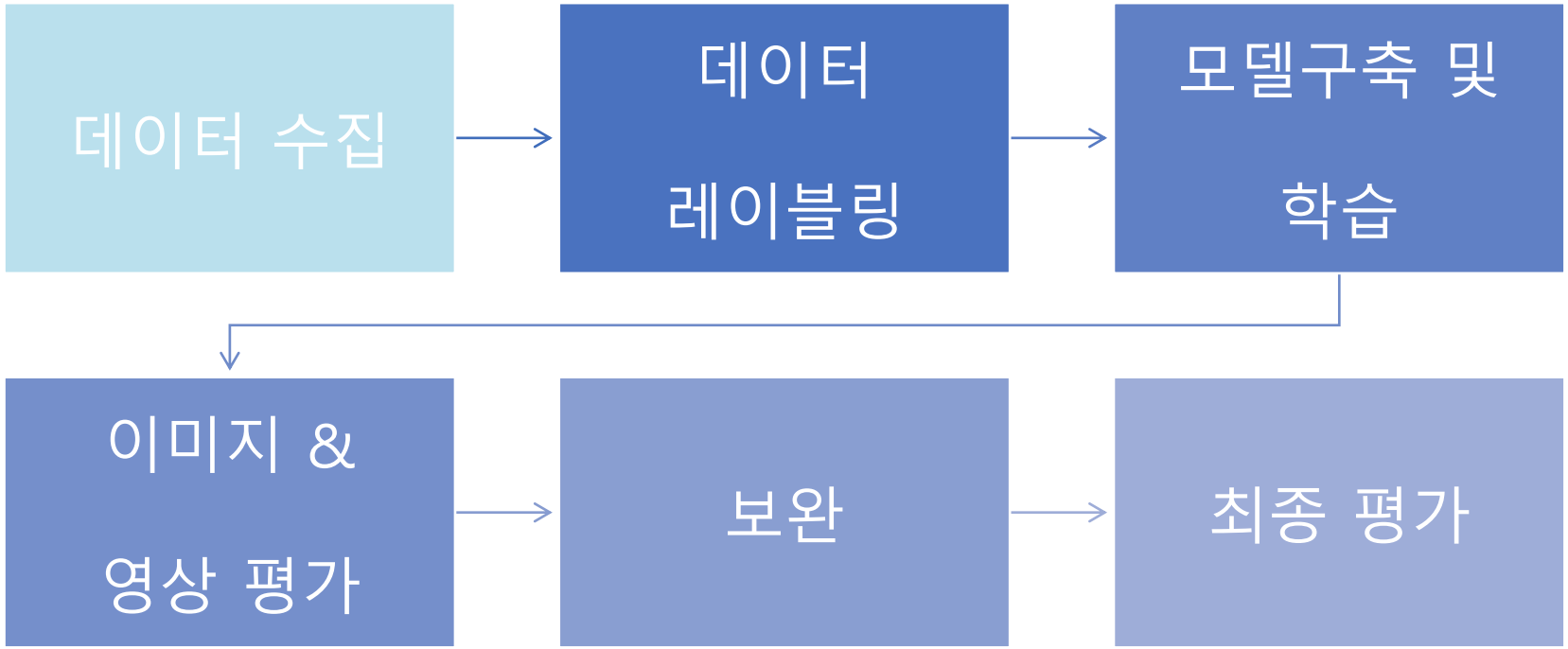
Segmentation Map
(라벨)





#Process

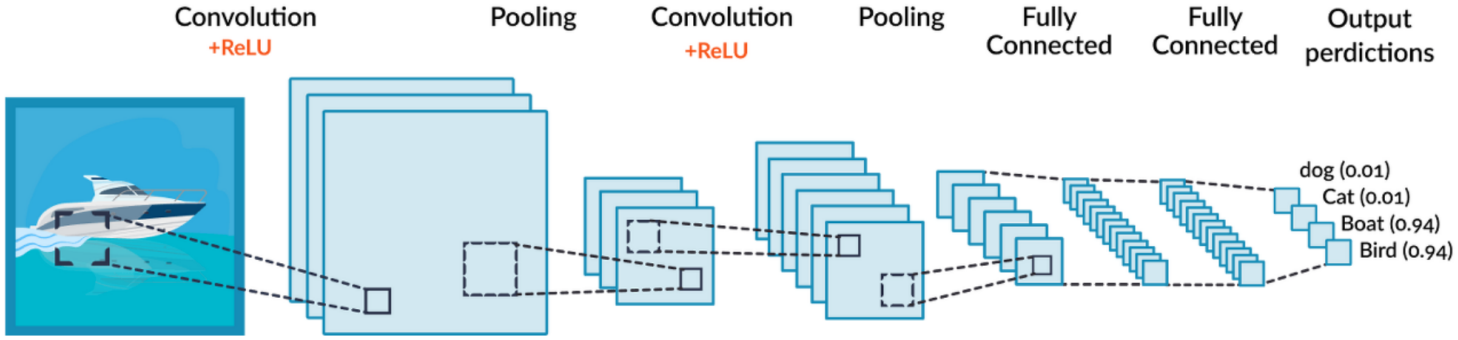
Process



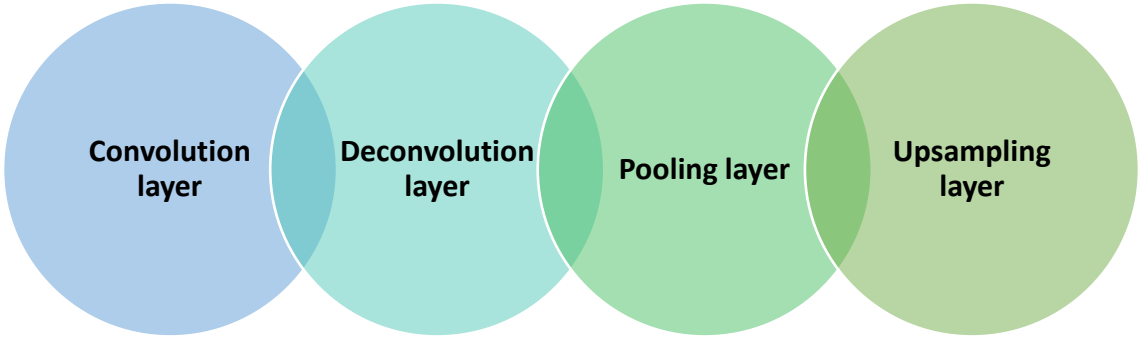
#Modeling

Modeling

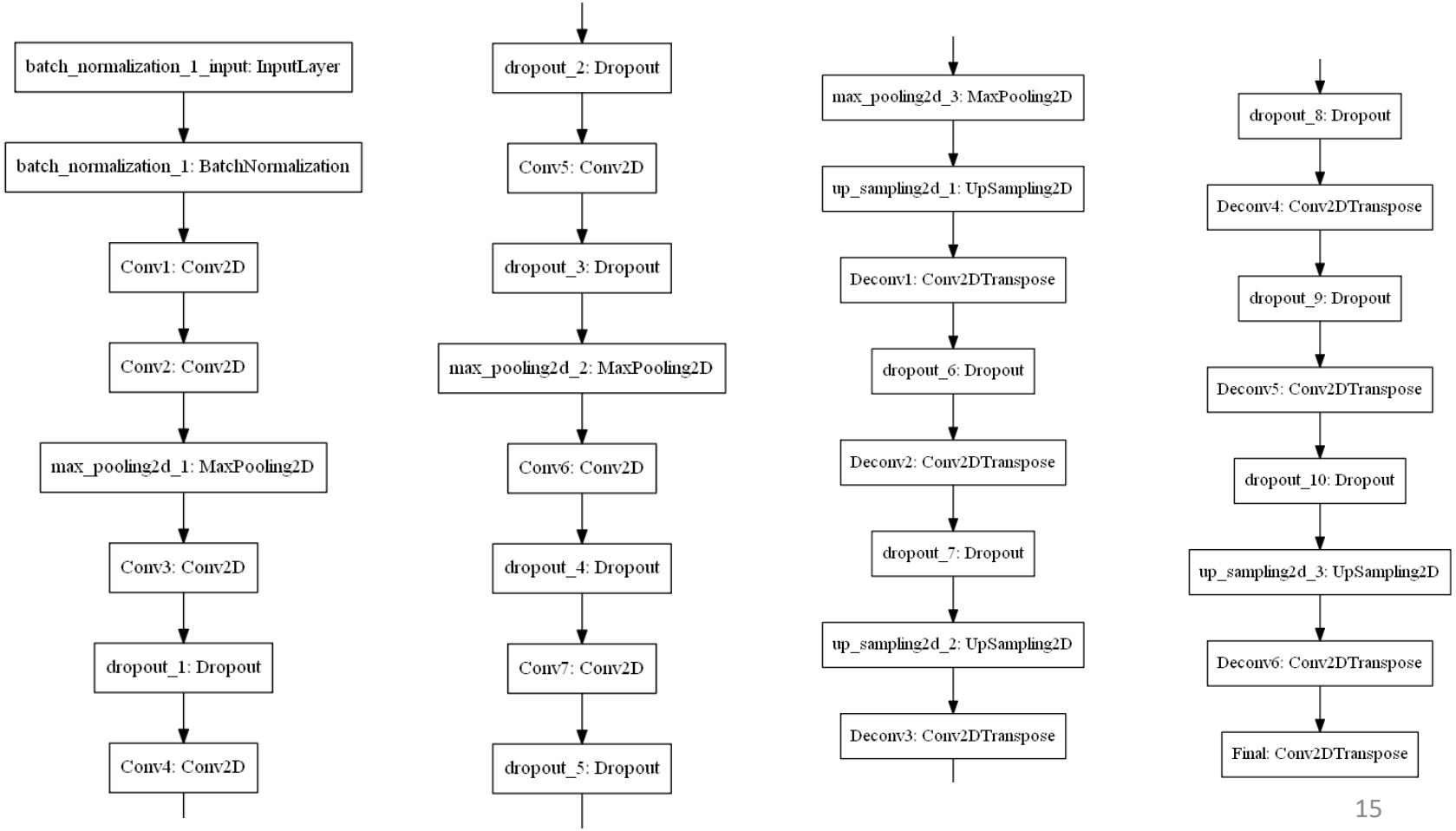
CNN 네트워크 구조



CNN 모델 구성



Modeling


batch_normalization_1_input: InputLayer
batch_normalization_1: BatchNormalization
Conv1: Conv2D
Conv2: Conv2D
max_pooling2d_1: MaxPooling2D
Conv3: Conv2D
dropout_1: Dropout
Conv4: Conv2D
dropout_2: Dropout
Conv5: Conv2D
dropout_3: Dropout
max_pooling2d_2: MaxPooling2D
Conv6: Conv2D
dropout_4: Dropout
Conv7: Conv2D
dropout_5: Dropout
max_pooling2d_3: MaxPooling2D
up_sampling2d_1: UpSampling2D
Deconv1: Conv2DTranspose
dropout_6: Dropout
Deconv2: Conv2DTranspose
dropout_7: Dropout
up_sampling2d_2: UpSampling2D
Deconv3: Conv2DTranspose
dropout_8: Dropout
Deconv4: Conv2DTranspose
dropout_9: Dropout
Deconv5: Conv2DTranspose
dropout_10: Dropout
up_sampling2d_3: UpSampling2D
Deconv6: Conv2DTranspose
Final: Conv2DTranspose

Modeling

CNN 모델 구성	
Convolution layer	7개
Deconvolution layer	6개
Pooling layer	3개
Upsampling layer	3개

```python
def create_model(input_shape, pool_size):
    model = Sequential()
    model.add(BatchNormalization(input_shape=input_shape))

    # Conv Layer 1
    model.add(Conv2D(8, (3, 3), padding='valid', strides=(1, 1), activation='relu', name='Conv1'))

    # Conv Layer 2
    model.add(Conv2D(16, (3, 3), padding='valid', strides=(1, 1), activation='relu', name='Conv2'))

    # Pooling 1
    model.add(MaxPooling2D(pool_size=pool_size))

    # Conv Layer 3
    model.add(Conv2D(16, (3, 3), padding='valid', strides=(1, 1), activation='relu', name='Conv3'))
    model.add(Dropout(0.2))
```

⋮

```python
    # Pooling 3
    model.add(MaxPooling2D(pool_size=pool_size))

    # Upsample 1
    model.add(UpSampling2D(size=pool_size))

    # Deconv 1
    model.add(Conv2DTranspose(64, (3, 3), padding='valid', strides=(1, 1), activation='relu', name='Deconv1'))
    model.add(Dropout(0.2))
```

⋮

```python
    # Upsample 3
    model.add(UpSampling2D(size=pool_size))

    # Deconv 6
    model.add(Conv2DTranspose(16, (3, 3), padding='valid', strides=(1, 1), activation='relu', name='Deconv6'))

    # 마지막 layer - 1채널
    model.add(Conv2DTranspose(1, (3, 3), padding='valid', strides=(1, 1), activation='relu', name='Final'))

    return model
```

Modeling

CNN 모델 구성	
Convolution layer	7개
Deconvolution layer	6개
Pooling layer	3개
Upsampling layer	3개

```python
def main():
    # training 이미지, 라벨 로드
    train_images = pickle.load(open("full_CNN_train.p", "rb"))
    labels = pickle.load(open("full_CNN_labels.p", "rb"))

    train_images = np.array(train_images)
    labels = np.array(labels)

    # 라벨 정규화
    labels = labels / 255

    # shuffle & split
    train_images, labels = shuffle(train_images, labels)
    X_train, X_val, y_train, y_val = train_test_split(train_images, labels, test_size=0.1)

    batch_size = 128
    epochs = 20
    pool_size = (2, 2)
    input_shape = X_train.shape[1:]

    # create_model 함수 호출
    model = create_model(input_shape, pool_size)
```

⋮

```python
    model.compile(optimizer='Adam', loss='mse', metrics=['mae'])

    early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
    history = model.fit_generator(datagen.flow(X_train, y_train, batch_size=batch_size),
                                  steps_per_epoch=len(X_train)/batch_size, epochs=epochs, verbose=1,
                                  validation_data=(X_val, y_val), callbacks=[early_stop])

    model.trainable = False
    model.compile(optimizer='Adam', loss='mse', metrics=['mae'])
```

Training Result

검증 척도(Metrics) :

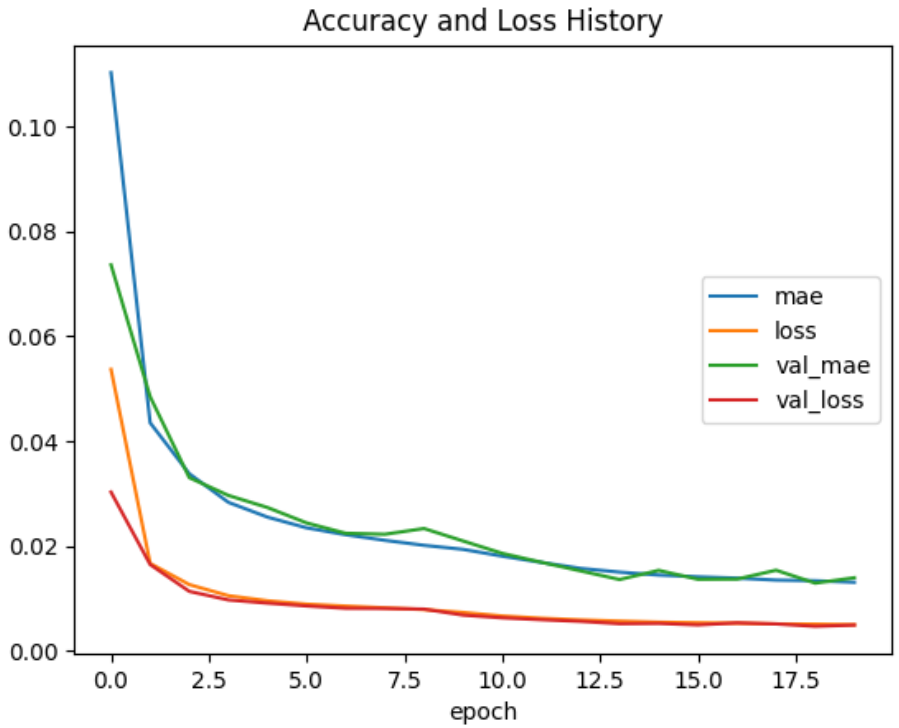
평균제곱오차(MSE)

Optimizer : Adam

Epoch : 18

Validation loss :

0.0057



#Test

Image Test

장점

- 어두운 터널구간에서 차선인식 가능
- 차선 변경 시 빠른 인식 가능
- 앞 차량과의 간격이 좁은 경우에도 차선인식 가능
- 커브 구간에서 유연한 인식 가능

Image Test

문제점 :

- 도로가 아닌 부분을 차선으로 인식하는 잡음 발생
- 차선이 표시되지 않은 구간에서 옆 차선 침범

Video Test

#Conclusion

Conclusion

성과

- 기존 연구의 대부분을 차지했던 필터와 원근변환을 이용한 모델보다 딥러닝 모델의 반응속도와 범용성이 높음을 확인함
- Transfer Learning을 사용하지 않고도 약 99.94%의 정확도를 달성함

한계

- 제한적인 학습 데이터에 최적화된 모델을 다양한 밝기와 복잡한 도로사정을 가진 현실에 그대로 적용하기 어려움
- 학습 데이터를 다양화하는 것을 통해 해결 가능할 것으로 예상됨

보완방향

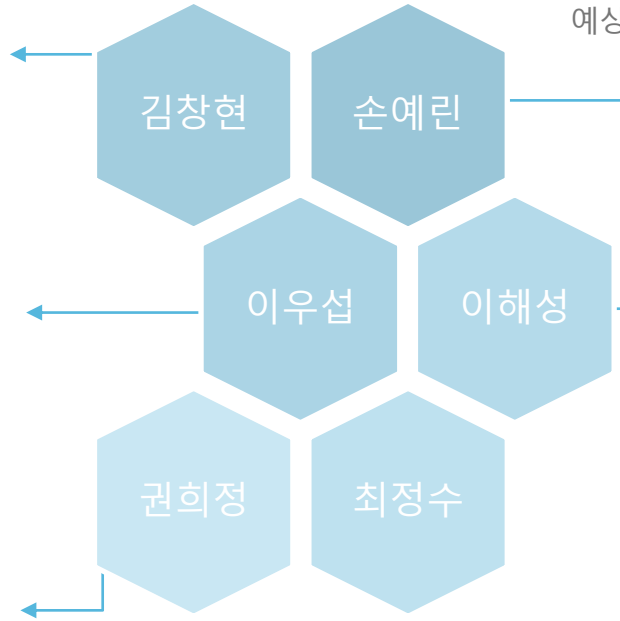
- 노이즈 및 옆 차선 침범 문제를 ROI 지정을 통해 보완하는 것이 필요함
- 차선이 명시되지 않은 교차로 구간에서의 혼동을 피하기 위해 차선 예측 알고리즘을 통한 보완이 필요함

#Review

Review

머지않은 미래에 현실화 될 자율주행기술의 일부 기능을 직접 구현해 보는 것이 재미있었고, 관련 논문들을 찾아보면서 최신 연구 동향을 파악할 수 있었던 값진 기회였다.
라이다와 같은 센서 데이터를 사용하지 않고 컴퓨터비전만으로 자율주행 프로그램을 만들기 위해서는 CNN 외에 앞 차량과의 간격을 계산하거나 노면의 특성 등을 고려하여 차선을 예상할 수 있게 하는 알고리즘이 필요할 것 같다.

프로젝트를 통해 객체를 인식하는 방법에 대해서 여러 알고리즘을 알고 구현하게 되었다. 또한 인식하는 것에 있어서 장애요인, 고려해야할 사항도 깨닫게 되는 시간이었다.



다양한 객체인식 모델에 대해 알게 되었으며 모델의 학습 정확도와 실제 모델이 객체인식을 하는 정확도가 여러가지 요인으로 인해 차이가 생길 수 있다는 것을 알게 되는 시간이었다.

객체인식이 생각보다 쉽지 않았지만 여러 모델을 통해 다양한 방법이 있다는 것을 알았고 정확도를 높이기 위한 학습이 많이 필요하다는 것을 알았다.

자율주행에 필요한 여러 요소와 방법들을 알게되었다. 또한 객체인식에 대해 좀 더 알게되는 좋은 계기였던것 같다.

Thank You