**LMP 자료 조사**

|  |  |
| --- | --- |
| Company Name | ROMiserable Co., Ltd. |
| Team | Lyft Motion Prediction |
| Document Version | 0.1 |
| Last Update | 2025년 3월 6일 목요일 오후 8:57:15 |
| Last Author | Min Su Je |

**<목차>**

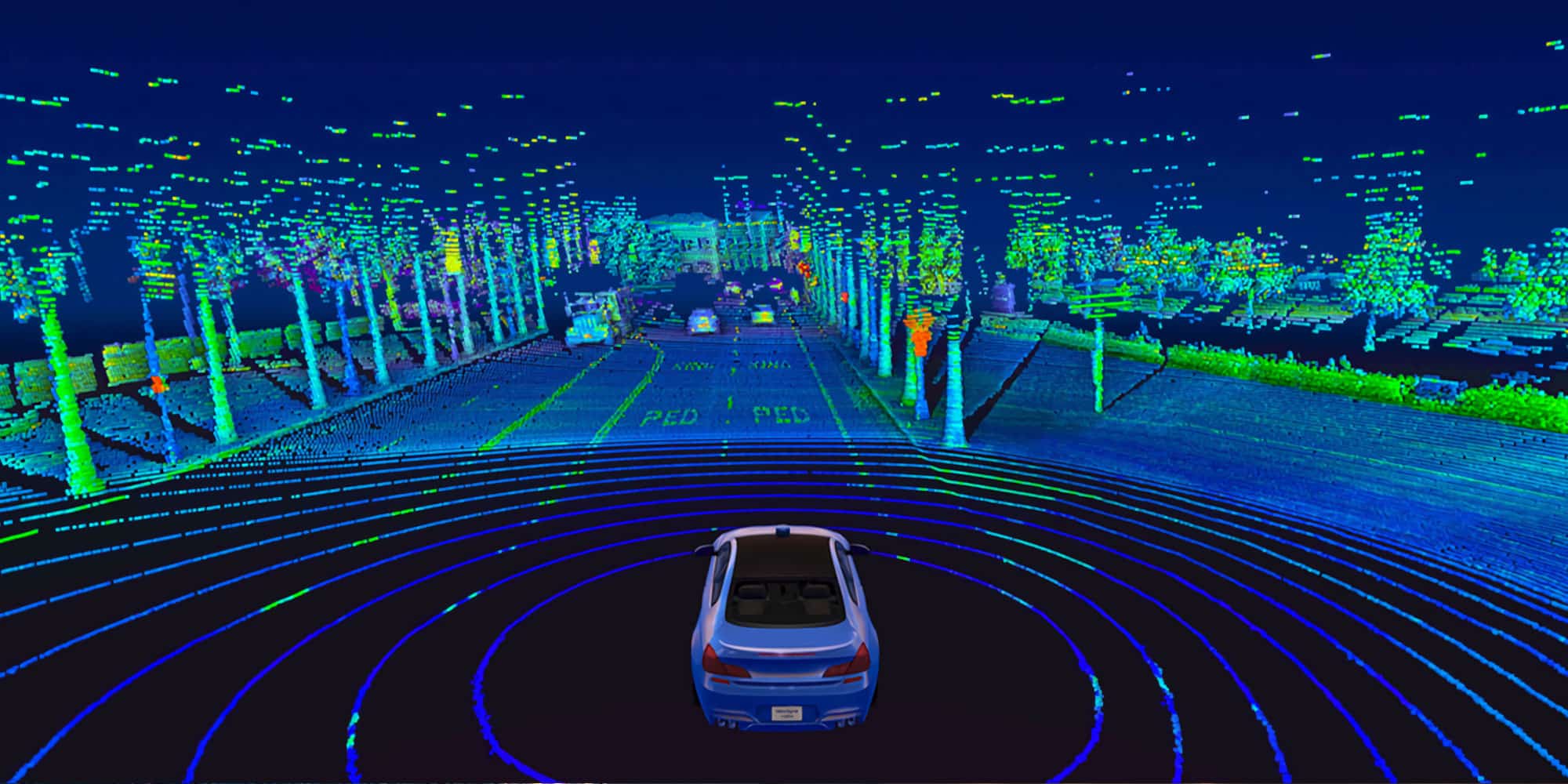
1. Data
2. 관련 자료 조사
3. **Data**

해당 문서는 motion prediction과 관련 되어 자료를 정리하여 모아 두기 위해 작성 된 것으로 작성자 이외로 자료 수정을 금지 합니다. 이하 아래 문서는 말을 최대한 줄여서 작성 하오니 양해 부탁 드립니다.  
파란글은 수정 될 수 있음을 표시(예 확정을 못한 사항)

1. **참조 데이터**

해당 프로젝트는 kaggle의 프로젝트를 통해 학습 및 자료를 받아 사용함.  
https://www.kaggle.com/competitions/lyft-motion-prediction-autonomous-vehicles  
또한 프로젝트는 L5Kit의 데이터 셋을 활용하여 작업하며 해당 자료는 4년전쯤이 마지막 업데이트.  
https://github.com/woven-planet/l5kit?tab=readme-ov-file  
https://woven-planet.github.io/l5kit/  
데이터는 Palo Alto에서 운행되는 Toyota AV에서 Woven이 기록한 1000시간 분량의 지각 출력. 이 데이터는 [zarr 형식](data\_format.md)을 사용하여 30초 단위로 저장  
또한 주석이달린 HD지도 활용(자세한 사항은 아래 ㅁ 참조)

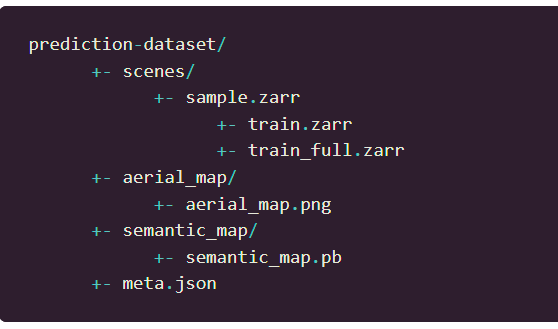
1. **데이터 가공**



데이터는 LiDAR sensor(3d+카메라)를 가공하여 물체 감지및 신호 정보 물체의 속도 가속도 등을 가공하여 zarr 형식으로 데이터를 제공해 줌.  


그리고 구글의 지도 이미지 데이터인 TIFF와 그에대한 자료 TFW파일을 제공 받으며, 해당 데이터를 semantic\_map.pb 데이터로 더 자세한 자료로 제공 받음.

1. **데이터 셋**



위와 같은 구조로 데이터를 받아 볼 수 있으며 데이터 셋을 읽어 와보면 아래와 같은 데이터 셋을 받을 수 있음.

1) Scene: ([ 0, 248], 'host-a116', 1576254970817669888, 1576254995817669888)

*- [0, 248] : scene의 인덱스 범위 씬은 25초 길이의 연속된 데이터 조각을 의미*

*- host-a116 : 데이터를 수집한 차량 id*

*- 1576254970817669888, 1576254995817669888 씬의 시작과 끝을 나타내는 Unix 타임 스탬프(1970년 1월 1일 00:00:00 UTC부터의 초 단위 경과 시간을 나타내는 정수)*

2) Frame: (1576254971002364806, [ 0, 56], [0, 0], [ 541.62249756, -2402.70581055, 288.62399292],

[[-0.42925233, -0.90311611, 0.01112122], [ 0.90317595, -0.42916176, 0.00966374],

[-0.00395468, 0.0141926 , 0.99989146]])

*- 1576254971002364806 : 프레임의 타임스탬프*

*- [0, 56]: 프레임 내의 객체 인덱스 범위로, 이 프레임에는 57개의 객체가 포함되어 있음을 나타냄*

*- [0, 0] : 프레임 내의 라벨 인덱스 밤위(라벨이 존재하지 않음을 의미)*

*- [ 541.62249756, -2402.70581055, 288.62399292] : 3d 프레임 위치 좌표*

*- [[-0.42925233, -0.90311611, 0.01112122], [0.90317595, -0.42916176, 0.00966374], [-0.00395468, 0.0141926, 0.99989146]]: 프레임 회전 행렬로 차량의 자세를 나타냄*

3) Agent: ([ 545.47155762, -2411.68945312], [4.427541 , 1.8168873, 1.6035007], 2.0373902, [0., 0.], 1,

[0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

*- [545.47155762, -2411.68945312]: 에이전트의 2D 위치*

*- [4.427541, 1.8168873, 1.6035007]: 에이전트의 크기를 나타내는 3D 치수*

*- 2.0373902: 에이전트의 속도*

*- [0., 0.]: 에이전트의 가속도*

*- 1: 에이전트의 라벨(1은 보통 차량이라고 함)*

*- [0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]: 에이전트의 속도 및 가속도에 대한 추가 정보로, 각 요소는 특정 속성이나 상태를 나타냄*

*- 속도\_x: 에이전트의 x축 방향 속도 (m/s)*

*- 속도\_y: 에이전트의 y축 방향 속도 (m/s)*

*- 속도\_z: 에이전트의 z축 방향 속도 (m/s)*

*- 가속도\_x: 에이전트의 x축 방향 가속도 (m/s²)*

*- 가속도\_y: 에이전트의 y축 방향 가속도 (m/s²)*

*- 가속도\_z: 에이전트의 z축 방향 가속도 (m/s²)*

*- 회전 속도\_x: 에이전트의 x축 회전 속도 (rad/s)*

*- 회전 속도\_y: 에이전트의 y축 회전 속도 (rad/s)*

*- 회전 속도\_z: 에이전트의 z축 회전 속도 (rad/s)*

*- 선형 속도: 에이전트의 선형 속도 크기 (m/s)*

*- 선형 가속도: 에이전트의 선형 가속도 크기 (m/s²)*

*- 선형 속도 표준 편차: 선형 속도의 표준 편차 (m/s)*

*- 선형 가속도 표준 편차: 선형 가속도의 표준 편차 (m/s²)*

*- 선형 속도 신뢰도: 선형 속도에 대한 신뢰도 점수 (0~1)*

*- 선형 가속도 신뢰도: 선형 가속도에 대한 신뢰도 점수 (0~1)*

*- 회전 속도 표준 편차: 회전 속도의 표준 편차 (rad/s)*

*- 회전 속도 신뢰도: 회전 속도에 대한 신뢰도 점수 (0~1)*

4) Traffic Light Face: ('m/rO', '0E3j', [0., 0., 1.])

*- 'm/rO(face\_id) : 트래픽 라이트 페이스(전구)의 고유 식별자*

*- '0E3j'(traffic\_light\_id) : 신호등의 고유 식별자의 고유 식별자*

*- [0., 0., 1.](traffic\_light\_face\_status) : 차례대로 빨간불 노란불 초록불*

5) TFW(TIFF world File)

*- 0.25 : 픽셀 크기 (X 방향)*

*- 0 : 회전 값 (X 방향)*

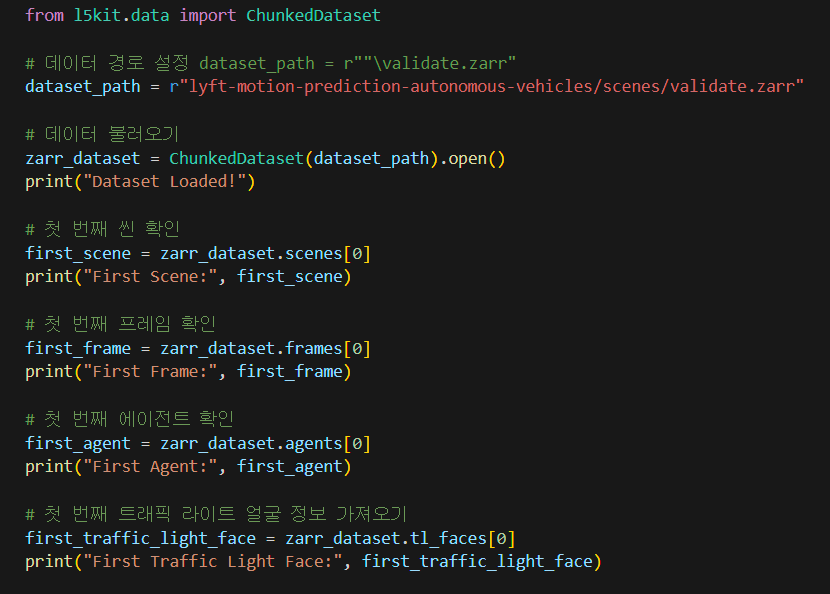
*- 0 : 회전 값 (Y 방향)*

*- -0.25 : 픽셀 크기 (Y 방향)*

*- -44628913.4205 : X 좌표*

*- 14701276.1451 : Y 좌표*

데이터 읽는 코드는 아래 와 같음



1. **Model**

**1. 모델 분석**

1) agents예제 모델(ResNet) 분석

fromtorchvision.models.resnetimportresnet50

1-1) ResNet의 특징

ResNet(Residual Network)은 잔차 연결(Residual Connection)을 활용하여 모델이 깊어 질수록(레이어 개수 증가-> 더 복잡한 패턴 학습) 발생하는 기울기 소실 문제를 해결 한 모델임. 또한 이미지 분류 및 컴퓨터 비전 작업에서 자주 사용 되는 모델이며 대표적인 CNN(Convolutional Neural Networks: 합성곱 신경망)모델 중에 하나임.  
ResNet은 뒤에 번호를 붙여 모델의 규모등을 표시하는데 아래와 같은 표를 작성 해볼수 있음

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 모델 | 레이어 | 블록구조 | 파라미터 | 연산량(FLOPs) | 특징 |
| ResNet18 | 18 | Basic Block | 11.7M | 1.8GFLOPs | 가볍고 연산량 적음 |
| ResNet34 | 34 | Basic Block | 21.8M | 3.6GFLOPs | 18보다 깊음 |
| ResNet50 | 50 | Botteneck Block | 25.6M | 4.1GFLOPs | 성능 속도 균형적 |
| ResNet101 | 101 | Botteneck Block | 44.5M | 7.8GFLOPs | 깊은 학습 가능 |
| ResNet152 | 152 | Botteneck Block | 60.2M | 11.3GFLOPs | 가장 깊은 모델 |

M(100만), 1 GFLOPs = 10억 (1,000,000,000) FLOPs -> FLOPs= 부동 소수점 연산 횟수 모델이 한번 추론시 수행하는 곱샘과 덧셈 연산의 총개수

Basic Block 의 흐름  
입력 -> 3x3 Conv -> ReLU -> 3x3 Convop -> Skip Connection(입력과 더하기) -> ReLU

Bottleneck Bock의 흐름  
입력 1x1Conv(채널 축소) -> ReLU -> 3x3 Conv(특징 추출) -> ReLU -> 1x1 Conv(채널 복구) -> Skip Connection -> ReLU

ReLU(f(x) = max(0,x))

x가 0보다 크면 x를 출력하고 x가 0보다 작으면 0을 나타낸다. 이는 기울기가 죽어버리고 신경망 구조가 0 이되어 죽어버린 ReLU라고 불리는 Dead ReLU 문제가 발생 할 수 있음  
ResNet에서는 기본적으로 ReLU과정 이전에Batch Normalization(BN)을 적용 하여 값이 평균 0, 분산 1로 정규화됨. -> 뉴런의 출력값이 너무 크거나 작아지는것 방지하는 역할일정 범위로 유지됨. -> Dead ReLU에 대해 너무 커지거나 작아지는 문제를 완화 시켜 줄 수 있음.

Dead ReLU 상황이 발생 할 수 있음 에도 사용 되는 이유는 계산의 효율성이 좋고 일부 신경망이 비활성화 되는 희소성이 발생하여 노이즈들은 무시하고 더 효율 적인 학습이 가능.

최종적으로 50~151의 모델이 목표에 가장 어울린다고 볼수 있다고 볼 수 있음.

단 50은 예제와 같은 것에 사용하기 유용 할 것으로 보이나 실질 적으론 101 이나 152 등을 사용 하는 것이 유용 할 것으로 보임.

1-2) 수학적 해석

Bottleneck Bock입력에 대한 수식적 변화과정을 나타 내고자 한다.

에 대한 Conv(Convlution) 연산은

: 연산 결과 출력 값  
 : 입력 이미지에서 i,j 위치 에서 m,n 만큼 이동한 값

: 커널의 각 값

: 커널의 모든 값을 합산

조금 더 구체적인 수식으로 가자면

입력 채널이 256이라는 가정을 하고 하겠음.

라는 이미지 입력이 들어오면 1x1 Convolution (채널 축소) : 커널을 적용 하고 편향을 더함  
채널이 256 에서 채널 축소 됨 보통 축소는 1/4로 줄어 들며 ResNet에서 256입력이면 64로 출력함

을 사용하여 3x3 Convolution(특징 추출) :

입력을 64 채널로 받고 출력 되는 채널은 64채널로 입출력의 채널의 수가 같다.

채널 복구 단계을 적용 하고 편향

입력 받은 64의 채널을 기존의 256 채널로 복구 한다(마지막의 Skip Connection을 진행 하기위해)

마지막 단계 Skip Connection는 에 입력 x를 더함.

최초 데이터 x 256채널과 채널 복구 단계의 데이터를 더함.

최종 출력은 z

채널 축소와 특징 추출 채널 복구를 통해 만든 데이터를 처음 입력 x와 더하여 잔차 학습을 적용된다고 해석 할 수 있음.

Skip Connection 중에 채널이 맞지 않다면 채널을 같게 맞춰 줘야함.

아래는 수정이 필요한 자료 아직 미완.

2) 이외 모델

딥러닝에 사용되는 모델의 수는 여러가지가 있다. 선형회귀나 옛 필터 종류들도 하나의 딥러닝 모델로 볼 수 있지만 모델들은 점차 진화 하여 크게 어떤 데이터를 어떤 출력을 만들어 낼 것인가에 따라 모델군으로 크게 분류하고 데이터에 알맞은 모델링을 선택

2-1) AlexNet

5개의 합성 곱층

3개의 완전 연결층

ReLU 활성화 함수 사용

Y=f(W\*X+b)

cnn의 기본 구조를 제시한 모델로 간단한 구조로 빠른 연산이 가능 하나 최근 모델 치고 성능이 낮으며 네트워크가 얕은편에 속함

최종적으로 깊이가 얕고 성능자체가 낮아 부적합함,

2-2) VGG((16,19가 있음)

깊은 신경망을 통해 표현력을 가지고 3x3 합성곱 필터를 다층으로 쌓아 깊은 네트워크를 구성한 모델

Y=f(W\*X+b)

간결한 구조로 깊은 신경망을 활용하는데에 성능이 좋으나 연산량이 많고 메모리 사용량 역시 많음

최종적으로 연산량이 많아 실시간 예측이 어려움으로 부적합

2-3) EffcientNet

2-4) DenseNet

2-5) MobileNet

1. **2**
2. **3**
3. **4**
4. **5**
5. **6**
6. **7**
7. **8**
8. **9**
9. **0**