머신러닝

최종 보고서

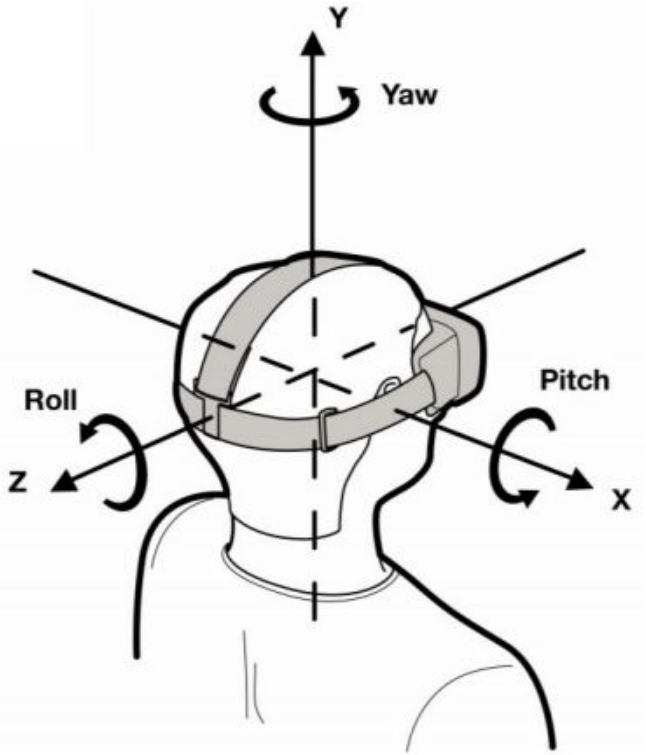
2020. 12. 21

목차

개요

목표

- 현재와 과거의 VR 사용자의 움직임 데이터를 이용해서 미래의 움직임을 예측한다.



데이터

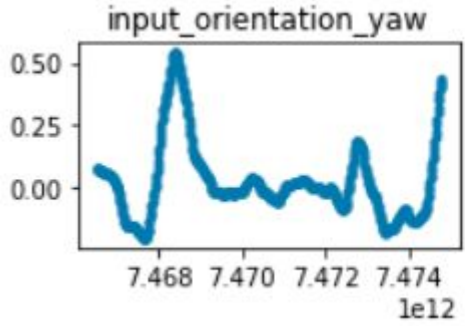
- 사용할 수 있는 데이터는 현재와 과거의 머리 각도 yaw, pitch, roll과, 각속도, 가속도 센서 값 등이 있다.
- 해당 데이터를 통해 300ms 미래에 사용자의 머리 각도 yaw, pitch, roll 값을 예측한다.

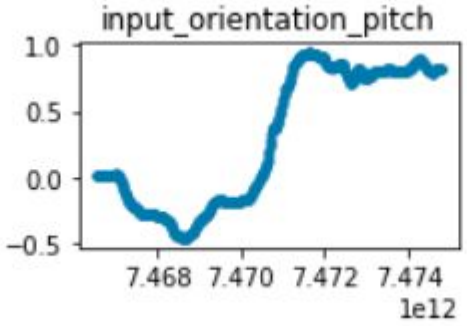
데이터 구조

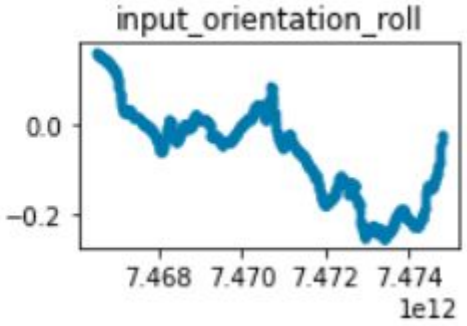
데이터는 시계열 데이터로 시간에 따라서 변하는 위치나 속도, 가속도와 센서 데이터 등을 가지고 있다.

각도 단위는 radian, 시간 단위는 tick으로 되어있다.

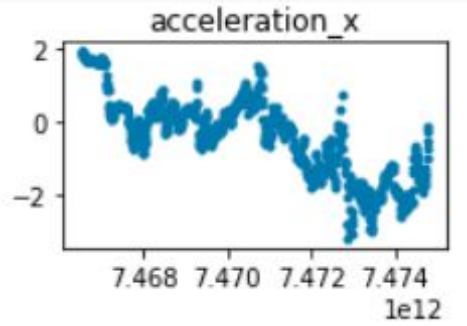
사용자의 움직임 각도 데이터

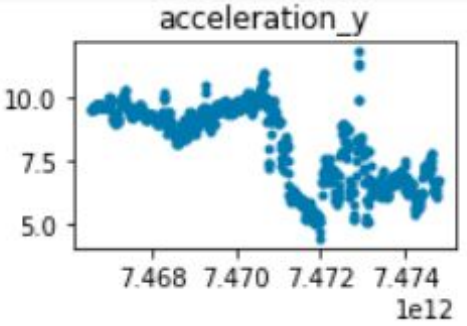


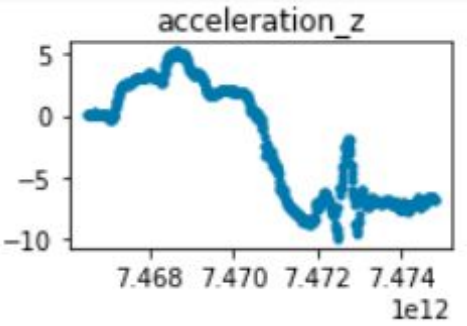




사용자의 움직임 가속도 데이터





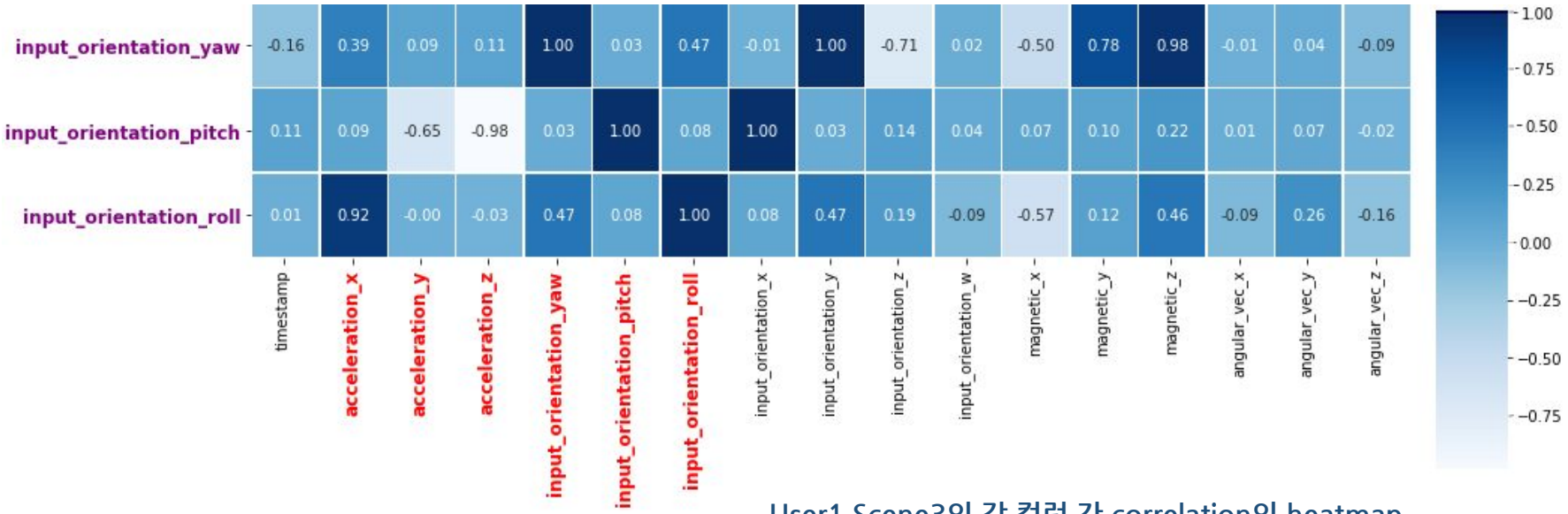


상관관계 분석

User1 Scene3의 motion data에서 모든 column간의 서로의 correlation을 구한 테이블이다.

구하고자 하는 input_orientation_[yaw, pitch, roll] 값과 상관관계가 큰 acceleration_[x, y, z] 데이터로 총 6가지 데이터를 입력으로 사용했고, input_orientation_[yaw, pitch, roll] 3개의 출력 값을 갖는다.

관련성이 떨어지는 데이터는 많으면 많을수록 overfit할 확률이 증가하며, 실험적으로 다른 column들로 성능 향상이 적다는 것을 확인하였기 때문에 다른 column은 사용하지 않았다.

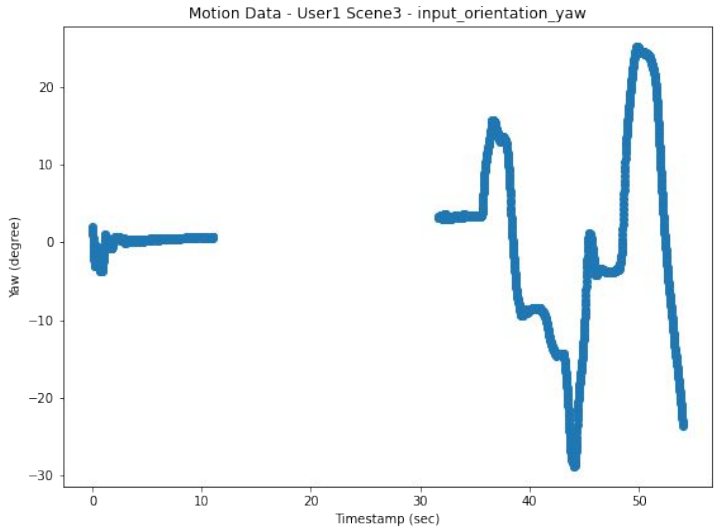


User1 Scene3의 각 컬럼 간 correlation의 heatmap.
절대값이 클 수록 관계가 크다.

끊어진 데이터 처리

모든 데이터의 초반에 수십여초 동안 끊어지는 부분이 존재한다.

끊어진 부분의 index를 수동을 찾아서 초반 1000~2000 timestamp의 데이터는 사용하지 않았다.



User1, Scene3의 yaw 데이터 초반부
데이터가 중간에 끊어진 것을 볼 수 있다.

데이터 전처리

단위 변환

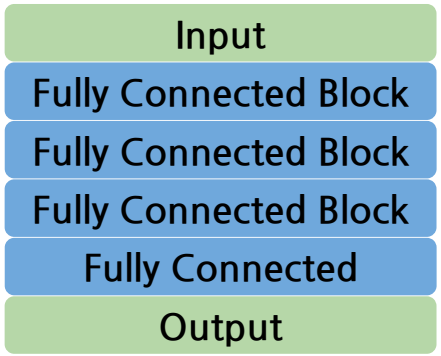
연산 편의성을 위해 각도에서 radian을 degree로, time을 tick 에서 seconds 로 변환했다.

데이터 전처리

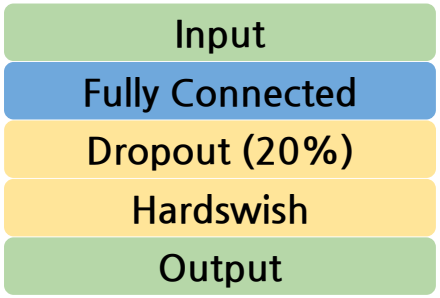
각각의 user와 scene에 대해 끊어지거나 잘못된 데이터의 위치를 수동으로 확인해서 제거하였다.
나머지를 절반으로 나눠서 앞부분은 train, 뒷부분은 test용으로 나누었다.
그리고 사용하지 않는 column을 삭제해서 새롭게 데이터셋을 만들었다.

MLP Based Network

MLP(Multi-Layer Perceptron)와 activation function과 dropout 기반의 모델 구조이다.
입력 sequence를 1차원 벡터로 flatten해준 뒤, MLP를 통과하여 결과를 낸다.



MLP 기반 모델의 구조

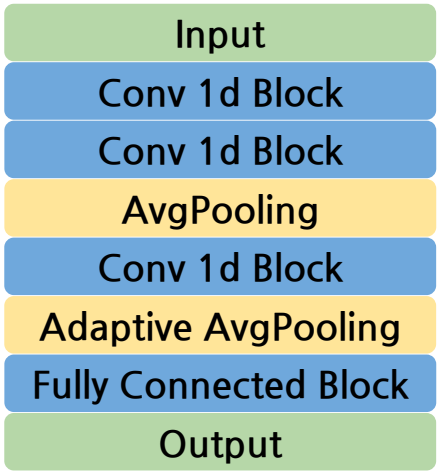


Fully Connected Block의 구조

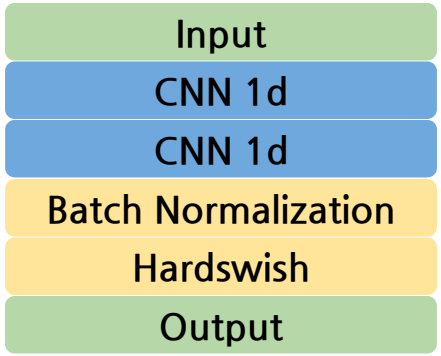
CNN Based Network

1차원 Convolutional Neural Network를 기반으로 작성된 모델이다.
yaw, pitch, roll와 acceleration 총 6개의 채널의 입력을 받아서 3개의 값을 출력한다.

사용하는 데이터셋은 max 값의 의미가 이미지 데이터에 비해 적기 때문에
MaxPooling이 아니라 AvgPooling을 사용했다.



CNN 기반 모델의 구조

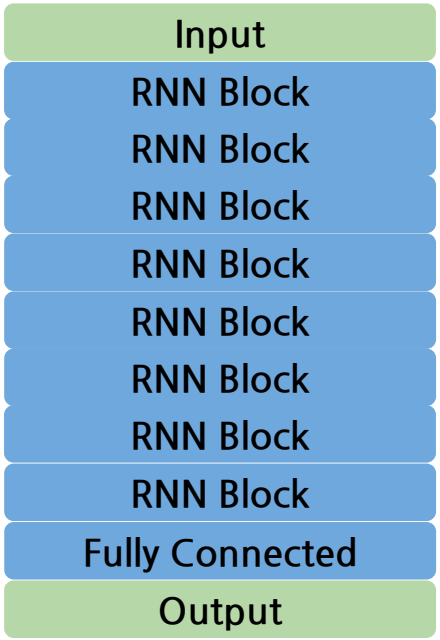


Conv 1d Block의 구조

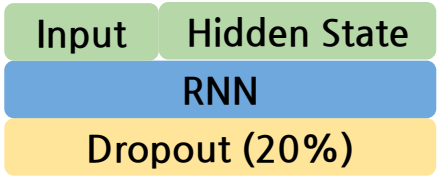
Stacked RNN Network

RNN Block을 8 layer 쌓은 Stacked RNN 구조이다.
RNN 대신에 LSTM과 GRU으로도 테스트를 했으나, 실험적으로 RNN이 가장 결과가 좋았으며,
Bidirectional 모드에서 가장 결과더 좋았다.

모델의 일반화 성능을 높이기 위해 각 RNN 블러마다 20%의 dropout layer를 추가해주었다.



Stacked RNN

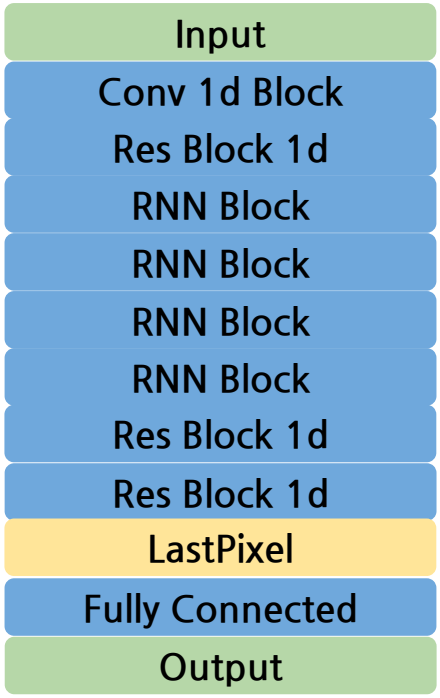


RNN Block 구조

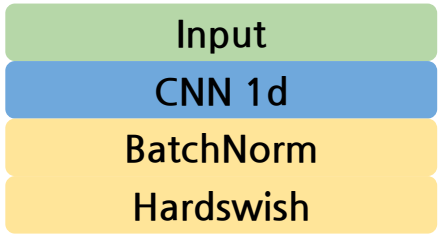
CRNNC

Timescale에서의 해석력을 높여주기 위해 Stacked RNN 구조의 모델 앞뒤에 CNN layer를 추가해주었다.
모델이 너무 깊어져서 overfit되는 것을 막기 위해 Stacked RNN의 깊이를 4층으로 줄여주었고, dropout을 제거하고, bidirectional 모드를 해제 해주었다.

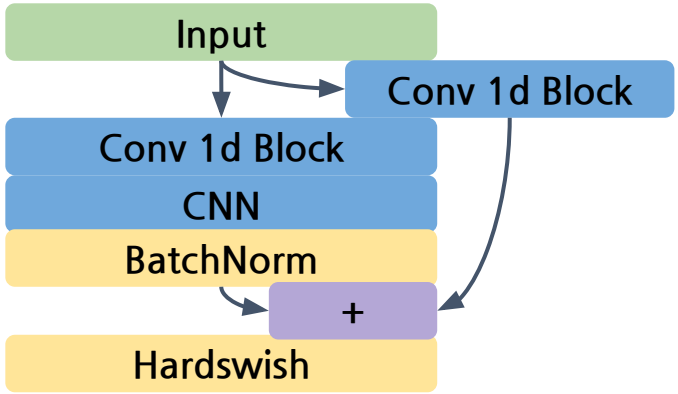
ConvBlock 부분에서 Grouped CNN으로 서로 다른 차원의 데이터인 yaw, pitch, roll과 acceleration을 따로 연산해주었다.
CNN의 출력 데이터에서 마지막 timestamp의 값만 선택해서 Fully Connected Layer에 넘겨준다.



CRNNC



Conv 1d Block 구조



Res Block 1d 구조

실험 설계

사용한 데이터

- 사용한 User는 1~7까지의 모든 user를 사용했다.
- Scene은 3번 scene만 사용했다.
- 각 User 별로 전반부 약 50%는 train데이터, 후반부 50%는 test 데이터로 사용했다.

데이터 입출력 포멧

- 기존 20초(120 timestamp)의 6가지 값(yaw, pitch, roll, acceleration)을 입력으로 하여 300ms(18timestamp)뒤의 1 timestamp의 값 3가지(yaw, pitch, roll)을 예측한다.
- 즉 입력 shape는 (batch_size, 120, 6) 이고, 출력 shape는 (batch_size, 3) 이다.

데이터 생성

- 현재 timestamp i에서 입력을 timestamp (i ~ i + 120)으로, 출력을 (i + 120 + 18)으로 한다.
- i를 1씩 증가시켜가면서 그 다음 번째 데이터를 생성한다.

실험 설계

모델 학습

120 timestamp 사이즈의 데이터를 모델에 입력으로 줘서 300ms 이후의 1 timestamp 사이즈의 값을 얻는다. 이 값을 실제 300ms 이후의 데이터 1 timestamp와 비교하여 loss를 계산한다.

MSE(Mean Squared Error) Loss

Loss 함수는 일종의 regression 문제이므로 MSE를 사용했다.

RAdam Optimizer

Adam optimizer보다 학습에 유리하다고 평가받고 있는 RAdam optimizer를 사용해서 모델을 학습시켰다.

평가 방법

MAE (Mean Absolute Error)

예측된 yaw, pitch, roll 데이터를 각각 실제 데이터의 차이의 절대값의 평균을 구했다.
즉, yaw, pitch, roll 3개의 MAE 값을 구한다.

RMS (Root Mean Square)

3개의 MAE 값의 제곱의 평균의 제곱근을 구한다.

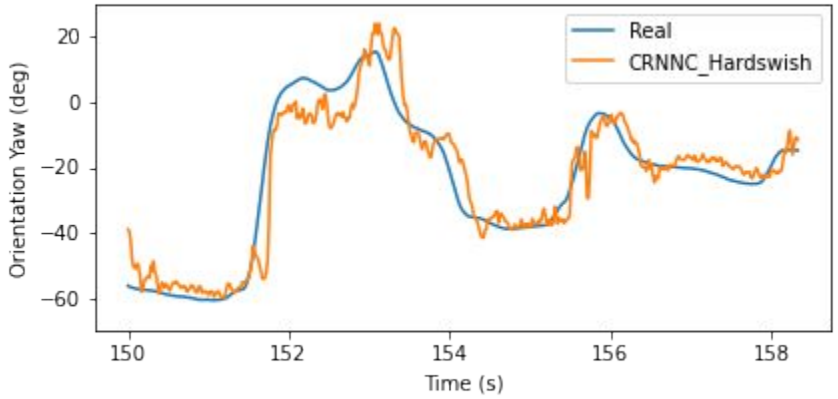
99 Percentile

모든 테스트 데이터에서 구한 RMS 값 중에서 99 백분위수를 구한다.

GAN 실험 방법

기존 실험 방법은 1 timestamp만을 예측하기 때문에 time scale에서 consistency가 떨어질 수 있다는 단점이 있다.

위의 단점을 해결하기 위해 예측 데이터가 현실적인가를 판단하는 discriminator를 추가하여 GAN 학습 방법을 통해 기존 모델과 유사한 모델을 학습시켰다.



실제 데이터와 예측 데이터
예측 데이터가 잔떨림을 더 많이 포함하고 있다.

GAN 실험 설계

입출력 데이터

입력으로는 기존과 동일하게 120 timestamp 사이즈의 데이터를 입력으로 사용하고, 출력 데이터는 입력 데이터보다 18 timestamp 미래의 120 timestamp 사이즈의 데이터이다.
그러므로 출력된 120 timestamp 가운데 앞의 102 timestamp는 입력 데이터에 포함되어있으므로 실제로 예측하는 것은 뒤의 18 timestamp 뿐이다.

Discriminator 학습

120 timestamp의 입출력 데이터를 discriminator가 real 데이터인지, fake 데이터인지 판별하고 real loss와 fake loss를 구한다. 해당 loss 값을 RAdam optimizer를 통해 discriminator 학습한다.

Generator 학습

120 timestamp의 출력 중에서 실제로 예측된 값인 뒤의 18 timestamp만으로 generator loss를 계산한다.
StockGAN에서 한 것과 같이 generator loss를 0.05의 가중치로, fake loss를 0.95의 가중치로 weighted sum을 하여 loss를 구한다. 해당 loss값을 RAdam optimizer를 통해 generator를 학습한다.

GAN 실험 설계

Loss 함수

Discriminator와 generator의 optimizer는 RAdam을 사용했다.
Loss 함수는 generator는 MSE, discriminator는 Binary Cross Entropy를 사용했다.

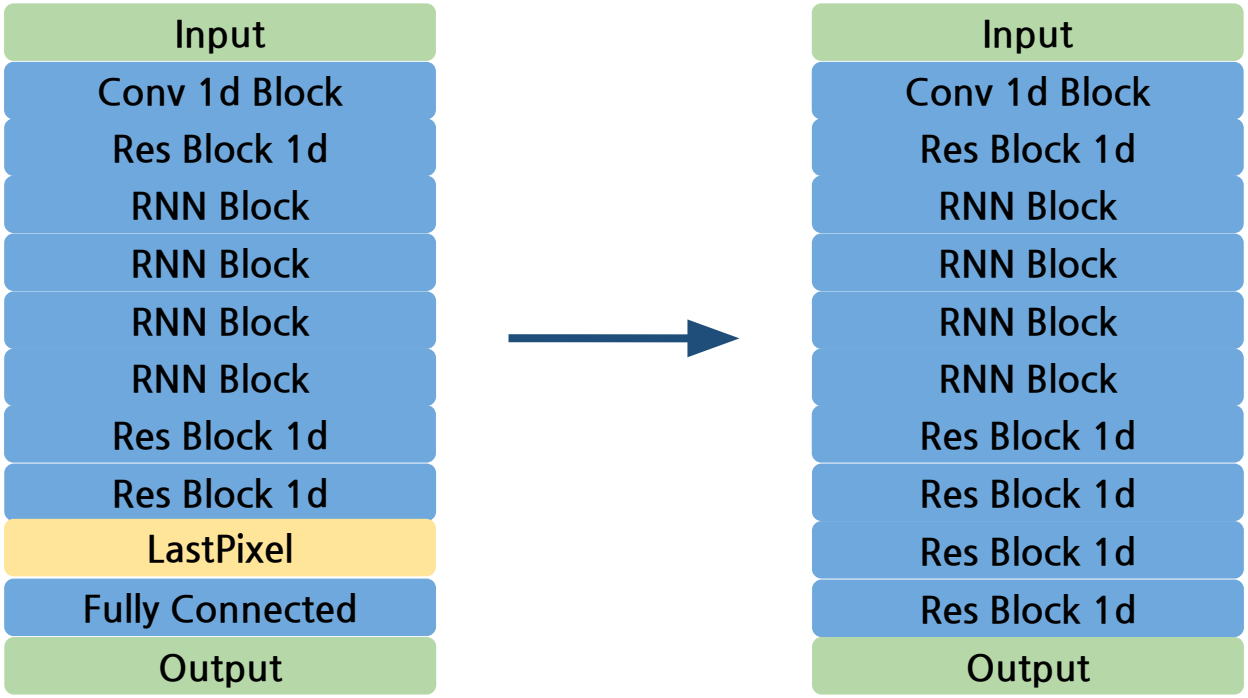
평가 방법

기존의 평가방법과 동일하게 yaw, pitch, roll의 MAE와 RMS, 99%tile을 구한다.
하지만 출력 데이터의 timestamp 사이즈가 18이므로 기존의 평가지표와의 형평성을 고려하여 loss를 구할 때와는 다르게 가장 마지막 18번 째 값에 대해서만 평가 지표를 계산한다.

CRNNC_GAN

기존의 1 timestamp의 출력을 내는 CRNNC와 달리, 여러 timestamp의 출력을 내기 위해 CRNNC의 모델 뒤쪽의 fully connected layer를 제거하고, Residual Block을 두 층 추가해주었다.

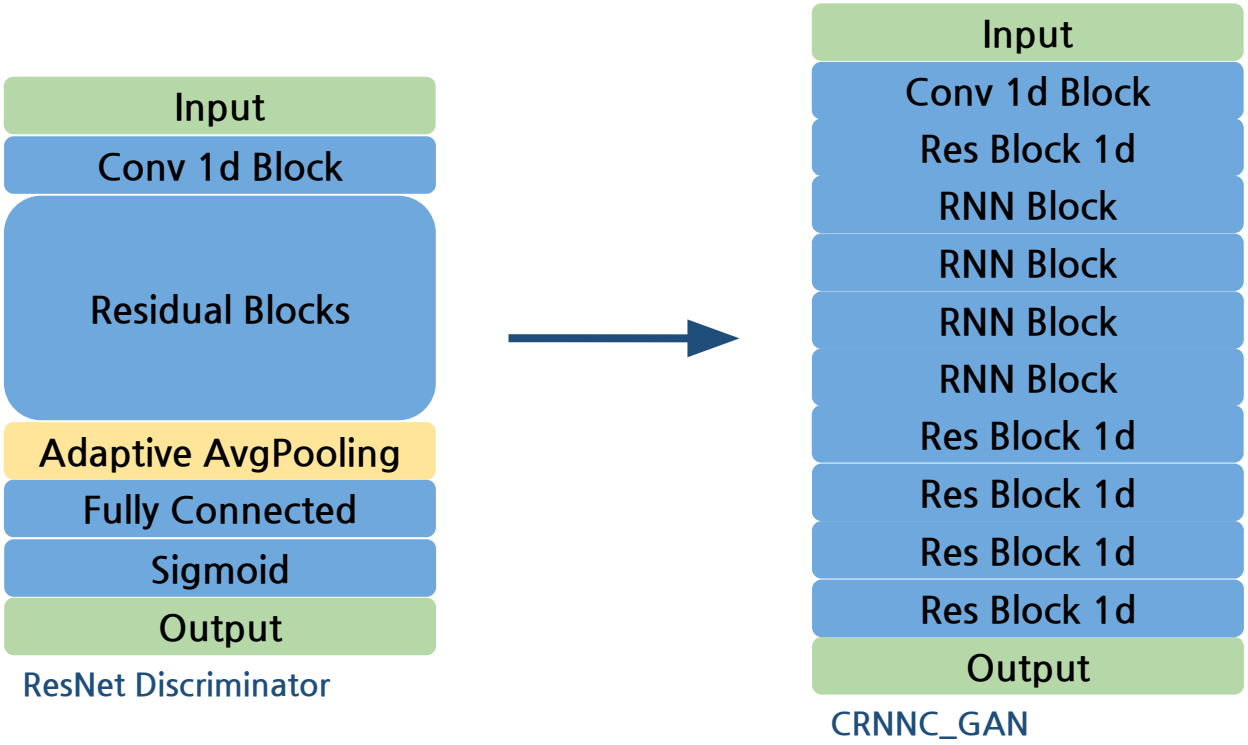
그리고 Residual Block에서 stride를 제거하여 입출력의 timestamp 사이즈가 동일해지도록 설정하였다.



ResNet Discriminator

일반적인 ResNet과 동일하게 Convolution Block과 Residual Block을 쌓아서 입력 데이터의 feature를 추출하고, Adaptive Average Pooling으로 데이터를 flatten 해준 뒤 Fully Connected Layer를 통해 하나의 값을 추출한다. 마지막으로 Sigmoid 함수를 통해 출력 값을 확률로 나타내어준다.

Residual Block은 ResNet15와 동일한 layer 사이즈로 설정해주었고, 모든 CNN은 1차원 CNN으로 치환해주었다.



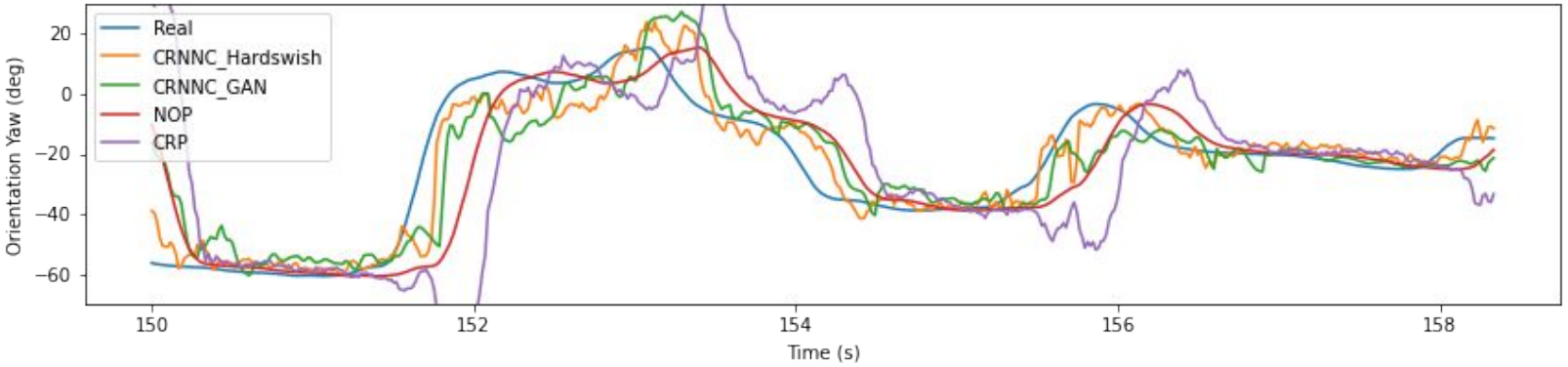
실험 결과

총 50 epoch 동안 validation 99%ile이 가장 좋은 epoch의 결과를 정리했다.

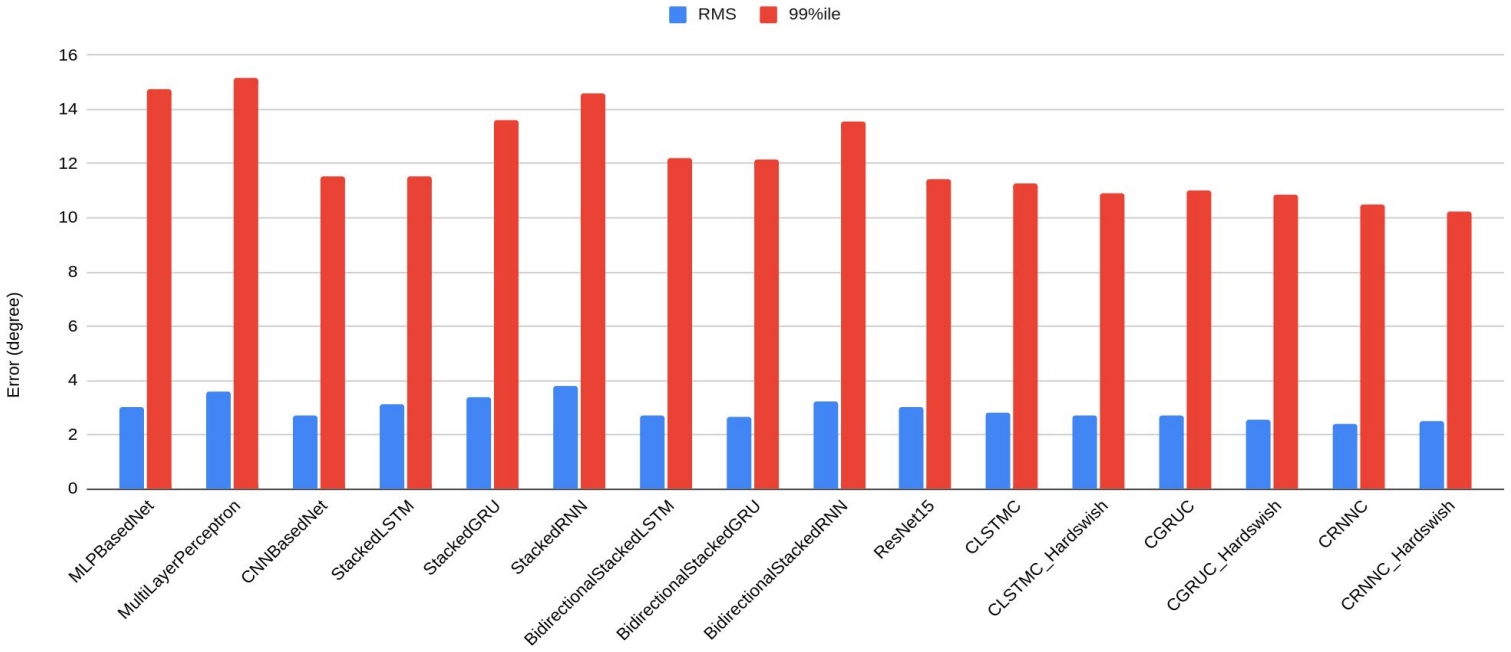
네트워크 구조 별 실험 결과 표

Network	Window-Size	Yaw	Pitch	Roll	RMS	99%ile
NOP	120	3.7757	1.2366	0.7040	2.3296	11.7335
CRP	120	7.4529	2.6030	1.5381	4.6435	22.9184
MLPBasedNet	120	3.6352	1.8897	0.9207	2.4244	9.5912
CNNBasedNet	120	4.6257	2.0408	1.2223	3.0031	10.9916
StackedLSTM	120	3.1227	1.7970	2.6377	2.5780	9.0086
StackedGRU	120	3.2675	1.8109	2.3453	2.5466	9.0403
StackedRNN	120	4.0451	2.3106	2.3353	3.0086	10.5136
BidirectionalStackedLSTM	120	3.2579	1.4815	1.0731	2.1572	9.7233
BidirectionalStackedGRU	120	3.1924	1.5377	1.0206	2.1290	9.6778
BidirectionalStackedGRU	120	4.0607	1.7552	1.8414	2.7665	9.8455
CLSTMC	120	3.1303	1.4297	0.9110	2.0553	8.5173
CLSTMC_Hardswish	120	2.9449	1.2244	0.6959	1.8847	8.5575
CGRUC	120	3.1314	1.4687	0.9334	2.0683	8.0942
CGRUC_Hardswish	120	2.8357	1.1794	0.7691	1.8279	8.0447
CRNNC	**120**	**3.0384**	**1.3095**	**0.7874**	**1.9636**	**8.0119**
CRNNC_Hardswish	**120**	**2.7567**	**1.3074**	**1.0968**	**1.8719**	**7.9650**
CRNNC_Hardswish_GAN	**120**	**3.4839**	**1.3917**	**0.7253**	**2.2061**	**8.4906**

- Head Orientation Yaw 예측 에러 스냅샷



- 네트워크 구조 별 예측 에러 비교 결과



결론

Conclusion

Convolutional Neural Network, Recurrent Neural Network 등을 사용하여 움직임 예측 모델을 만들었고, 실제 데이터를 통해 예측모델의 동작을 확인할 수 있었다.
그리고 GAN 학습 방법을 도입하여 성능을 끌어올릴 수 있었다.

Future Works

추후에는 학습이 불안정한 GAN 학습 방법을 개선하고, user와 scene의 정보를 사용하여 각각의 user와 scene에 특화된 결과를 예측하는 모델을 만들 수 있겠다.

참고자료

- **RAdam**
  Liyuan Liu et al, On the Variance of the Adaptive Learning Rate and Beyond, 2019

- **ResNet**
  Kaiming He et al, Deep Residual Learning for Image Recognition, 2015

- **StockGAN**
  Kang Zhang et al, Stock Market Prediction Based on Generative Adversarial Network, 2018

- **Deep RNN Framework**
  Bo Pang et al, Deep RNN Framework for Visual Sequence Application, 20181

Thank You