Master's Thesis Progress Presentation

User Position Prediction in 6-DoF Mixed Reality Applications using Recurrent Neural Networks

Oleksandra Baga

Freie Universität Berlin
Matrikelnummer 5480722
Master Computer Science
E-Mail: oleksandra.baga@gmail.com

Problem statement and introduction

01

The goal of the research.
Used hardware and software

Dataset and data exploration

02

Obtaining, analyze and preprocessing of 6-DoF dataset

Models

03

Analyzed RNN variants

Evaluation

04

Evaluation metrics

Experiments

05

Results of experiments with different RNN variants

Analyze

06

Discussion of the obtained results and steps for finishing the research

Problem statement and introduction

01

The goal of the research.
Used hardware and software

02 Dataset and data exploration

Models 03

04 Evaluation

Experiments 05

06 Analyze

Introduction

This thesis focuses on **evaluation of the approaches for the prediction of human head position in 6 degrees of freedom** (6-DoF) in VR Applications.

The goal of research is a **reduction of Motion-to-Photon (M2P) latency** for a given look-ahead time (LAT) in order to reduce the **network and computational delays** when transfering volumetric video data to the client from server.

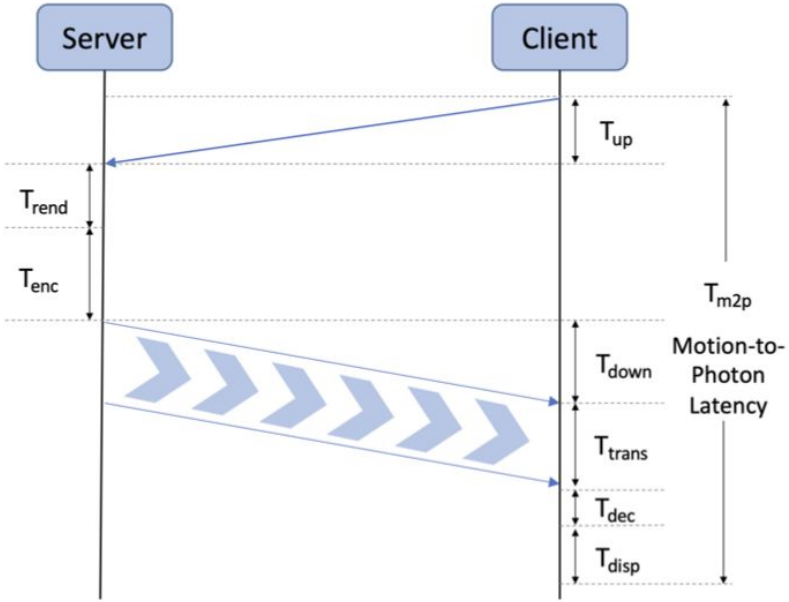


Fig.1. Components of the motion-to-photon latency for a remote rendering system.

Picture source: Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality.

Problem statement

The correct and fast head movement prediction is a key to provide a smooth and comfortable user experience in VR environment during head-mounted display (HDM) usage.

The VR hardware was once extremely expensive but recent years became more broadly accessible and the 6-DoF VR headset designed for the end-user were released.

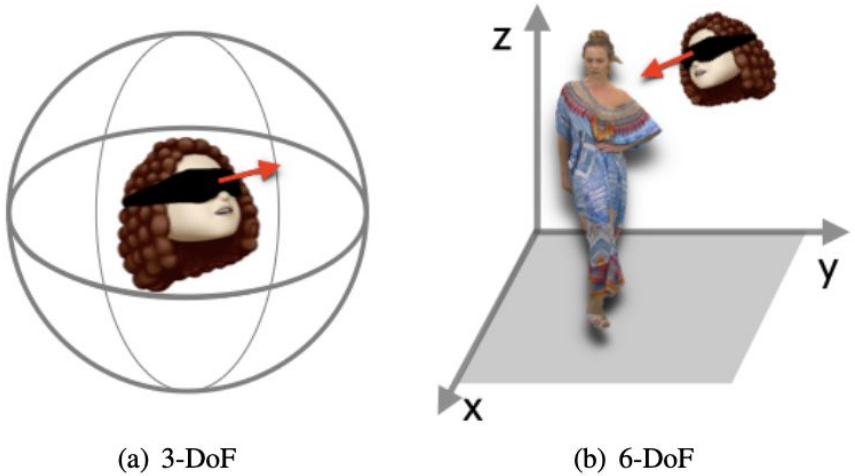


Fig.2. Users viewing point are inward in 3-DoF and outward in 6-DoF

Picture source: Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. From 3-DoF to 6-DoF: New Metrics to Analyse Users Behaviour in Immersive Applications

Problem statement

The user motion and prediction within a 3-DoF environment has been intensely researched for years.

However extending of these approaches from 3-DoF to a 6-DoF environment is not straightforward, due to the change of the users viewing point from inward to outward and additional three degrees of freedom.

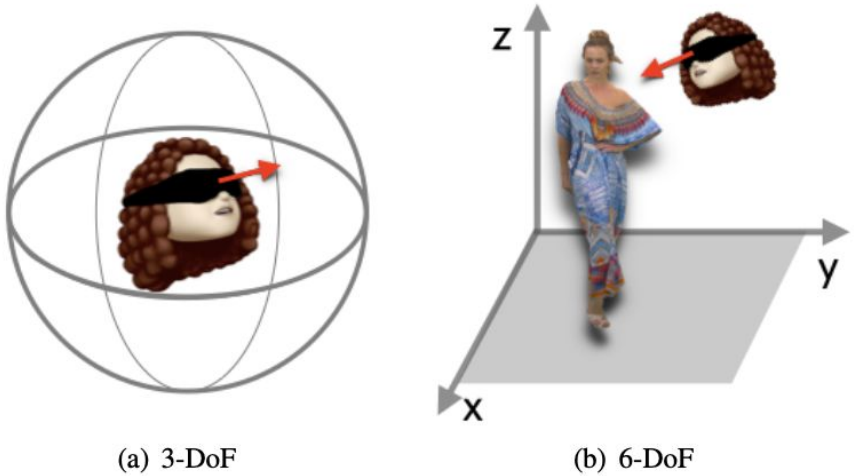


Fig.2. Users viewing point are inward in 3-DoF and outward in 6-DoF

Picture source: Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. From 3-DoF to 6-DoF: New Metrics to Analyse Users Behaviour in Immersive Applications

Motivation

Existing approaches are designed for the client side. Recently the new technique of the **rendering on a cloud server** was presented by the researches.

Thus it makes possible to decrease the computational load on the client device by offloading the task to a server infrastructure.

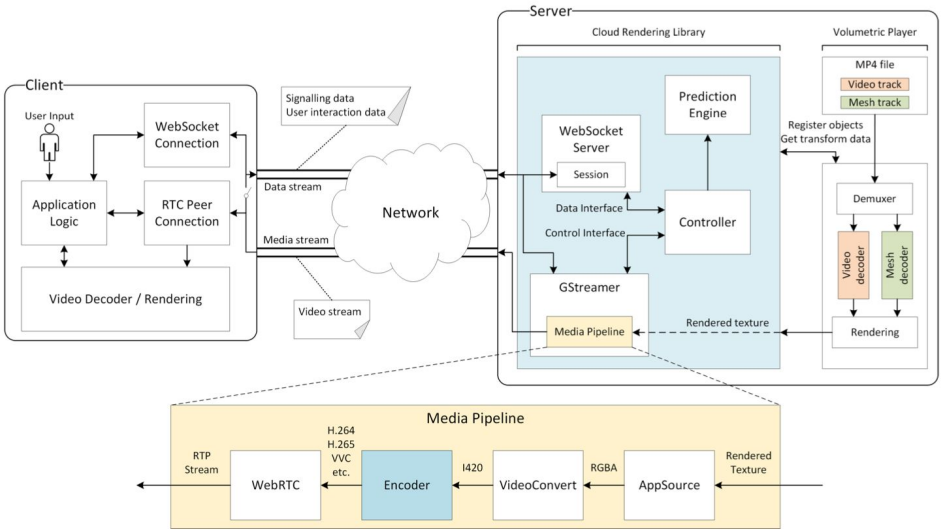


Fig.3. Overview of the system components and interfaces of cloud-based Volumetric Video Streaming

Picture source: Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality.

Motivation

Cloud-based streaming approach **adds network latency** and **processing delays** due to uploading user position to a server, rendering a new 2D picture from the 3D data and sending it back to a device.

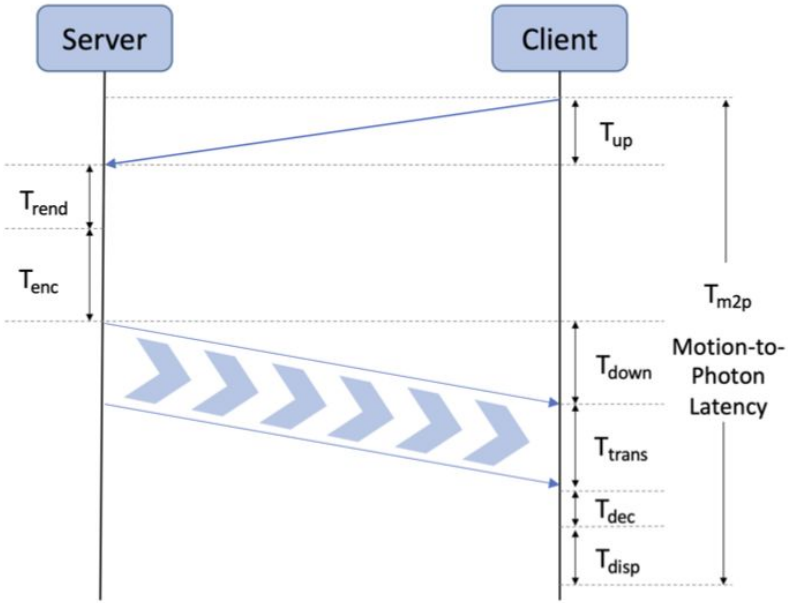


Fig.4. Components of the motion-to-photon latency for a remote rendering system.

Picture source: Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality.

Motivation

A rendered 2D image can appear even later on a display than with the usage of a local rendering system.

This research aims reducing the M2P-latency by **predicting the future user position and orientation** for a look-ahead time (LAT) and sending the corresponding rendered view to a client.

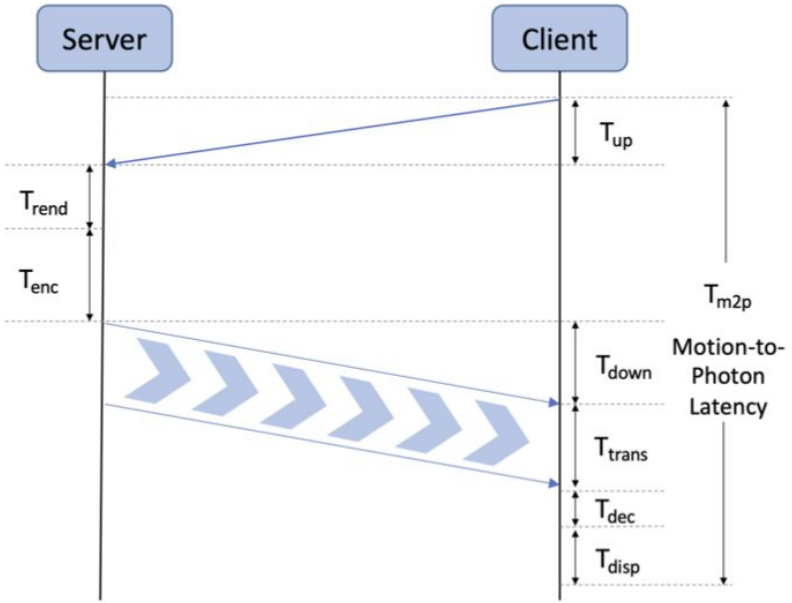


Fig.4. Components of the motion-to-photon latency for a remote rendering system.

Picture source: Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality.

Software

Python libraries allowed to access, process, and transform the data. Scikit-learn, Pandas, Numpy and others libraries are used.

Python native ML Framework. Works with an n-dimensional Tensor, that is similar to NumPy but can run on GPUs.

Bash script is used for launching multiple jobs on GPUs and for automatically post-processing job results and metrics

Unity game engine is used to build a HoloLens 2 application for obtaining the 6DOF-dataset

The underlying code to access the sensors on Hololens is written in C#

Hardware

Microsoft HoloLens

A virtual reality (VR) headset with transparent lenses, must be put on a head and enables VR experience.

It uses hand gestures with easy content manipulation. Any command could be a combination of the user's hand movement combined with voice control.

HoloLens 2 is used for obtaining the 6DOF-Dataset.

Picture source: https://www.microsoft.com/en-us/hololens/

Problem statement and introduction

01

02

Dataset and data exploration

Obtaining, analyze and preprocessing of 6-DoF dataset

Models

03

04

Evaluation

Experiments

05

06

Analyze

Obtaining 6-DoF Dataset

The user position and orientation were obtained with Unity application developed for HoloLens 2.

Main Camera in Unity is automatically configured to track head movements.

A volumetric animated object placed 3 metres ahead of the user in the MR environment.

Using the Main Camera, 6-DoF datasets were logged in a csv-file.

Picture source: Oleksandra Baga

Obtaining 6-DoF Dataset

The obtained 6-DoF dataset has 10 features used in training process:

- position (x, y, z)
- orientation (qx, qy, qz, qw)
- velocity (x, y, z).

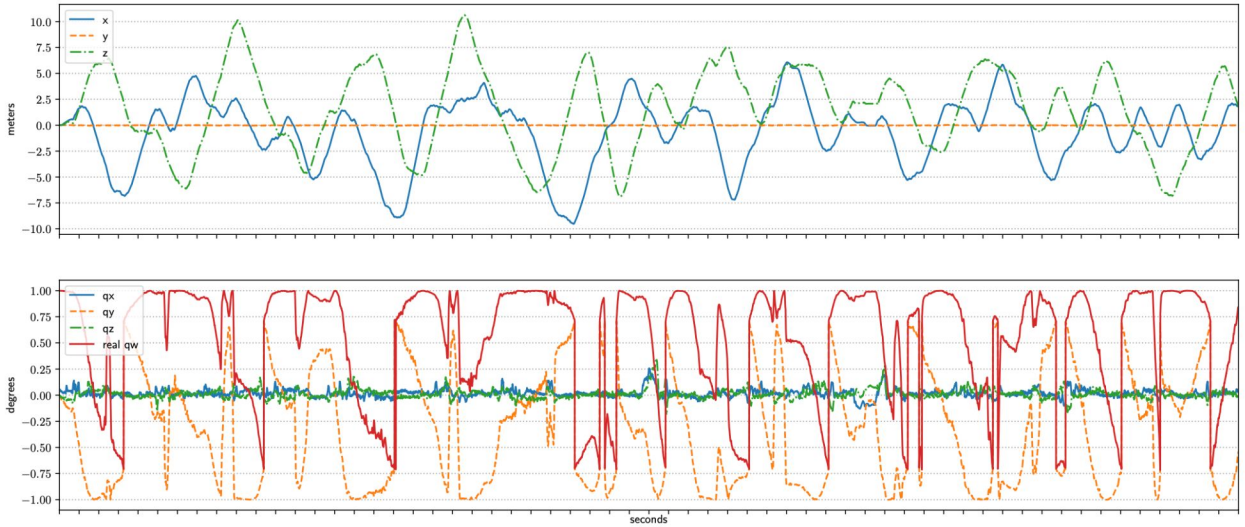
The datasets were recorded in the laboratory space. HMD was presented to users and the basic functions were explained.

During data recording, users freely walked wearing HMD in laboratory space.

<u>No personal data was recorded during these sessions and all traces are obtained anonymously.</u>

Picture source: Oleksandra Baga

Interpolated dataset



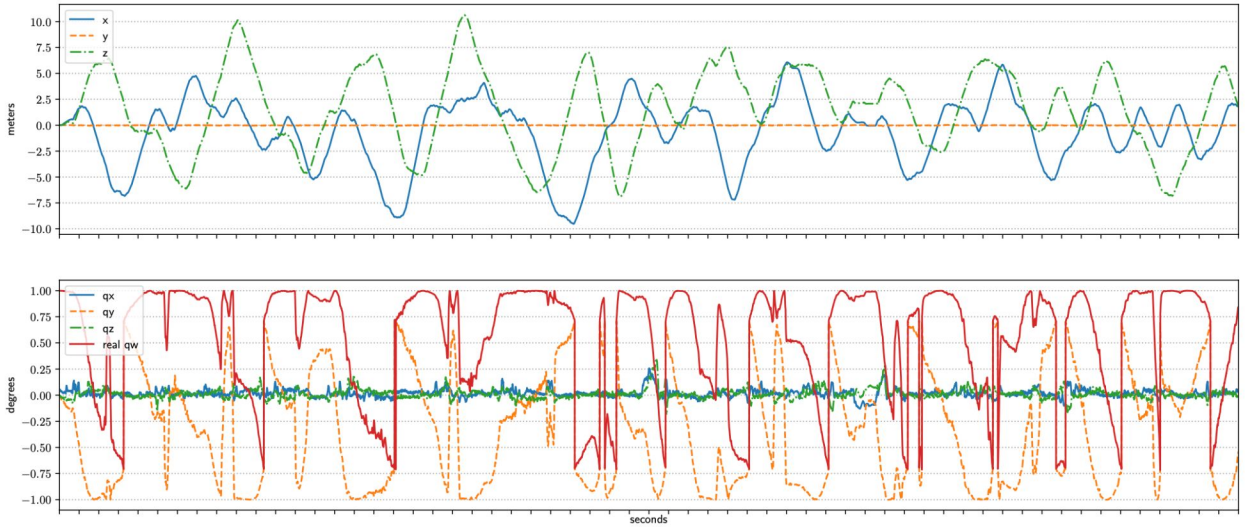
All traces were recorded over 10 minutes long on average 12 minutes.

Then traces were shortened to a precise length of 10 minutes to ensure equal data length for the purpose of visualization and analysis.

Users could freely walk around the volumetric object projected into their HMD.

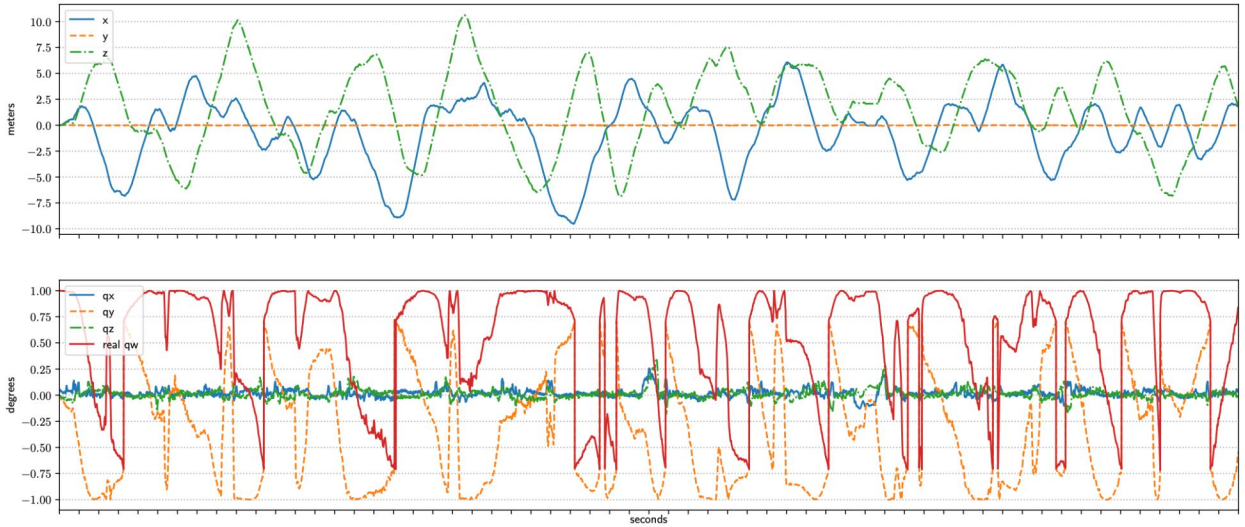
A user rarely moves along the y-axis. The y-axis shows the vertical movement that the users could make if they sit down or stand up what requires more effort.

Interpolated dataset



Due to signal processing and propagation delays, distance in time between two consecutive samples was either increased or decreased.

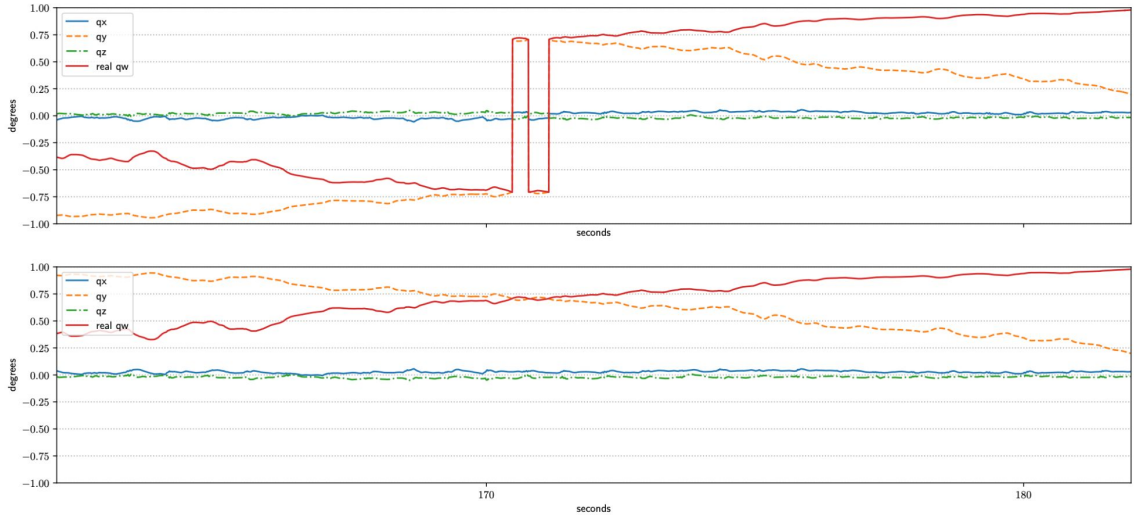
Interpolated dataset



The position and velocity data were upsampled using linear interpolation. SLURP used for quaternions.

The components qy and qw of quaternion have sharp change of sign making it harder for a model to learn.

Flipped negative quaternions



Flipping the sign will not affect the rotation, but it will ensure that there are no large jumps in 4D vector space between the two neighboring quaternions with similar rotation.

Problem statement and introduction 01

02 Dataset and data exploration

Models 03

Analyzed RNN variants

04 Evaluation

Experiments 05

06 Analyze

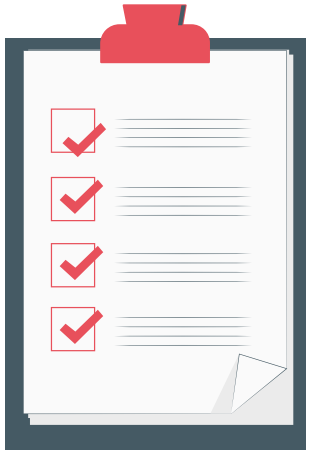
Model Parameters

Dataset

Interpolated dataset with flipped negative quaternions is used. Model i**s trained on 60% and validated on 20% of dataset.** The rest **20% used for test**. No shuffle is applied during dataset slicing.

Epochs

Although the error decreases very slowly after 150-200 epochs, **the model converged to a smallest achievable error after 500 epochs**. The model is overtrained with 1000 epochs



Batch size

Both batch sizes of 512 and 1024

Hidden layers dimension

Both 512 and 1024 hidden layers.

Rule of thumb

Empiric evidence shows that batch size two times smaller than hidden layers yields better results.

Learning rate and weight decay of Adam Optimizer have highest influence!
After grid of parameter were tried, LR is set to 1e-4 and weight decay to 1e-12

Model Inputs

The obtained 6-DoF dataset has 10 features used in training process:

- ➔ position (x, y, z)
- ➔ orientation (qx, qy, qz, qw)
- ➔ velocity (x, y, z).

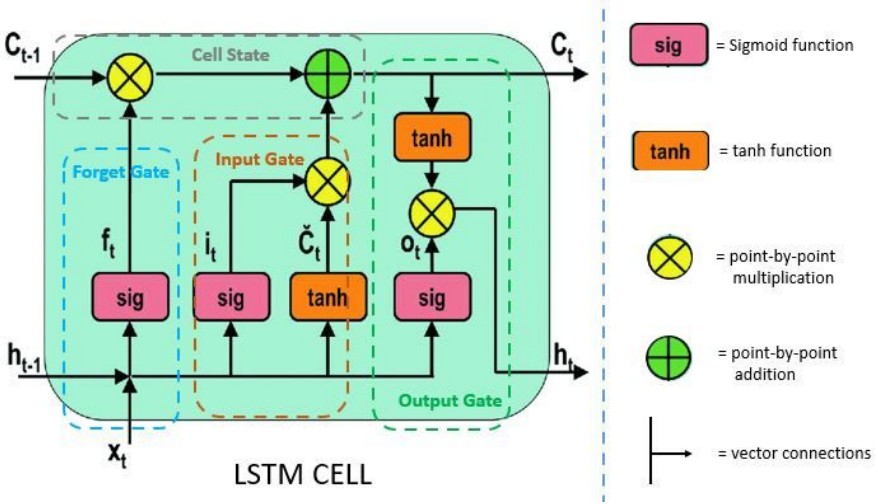
timestamp	x	y	z	qx	qy	qz	qw	velocity_x	velocity_y	velocity_z
0.0	0.004954389	0.003402365	0.01010712	0.0522510460919448	-0.0092471243083722	-0.0147093988998279	0.998482825319659	0.3015088	0.2081023	0.586858
5000000.0	0.0048331026666666	0.0033084996666666	0.0105062066666666	0.0528291300322694	-0.0094015253918042	-0.0147654075580485	0.9984501375031132	0.1943642	0.1336575666666666	0.41589388
10000000.0	0.0047118163333333	0.0032146343333333	0.0109052933333333	0.053407194836499	-0.0095559230697573	-0.0148214108678517	0.9984170880217684	0.0872196	0.0592128333333333	0.24492976
15000000.0	0.00459053	0.003120769	0.01130438	0.0539852402952434	-0.0097103172863047	-0.0148774088089516	0.998383676887596	-0.019925	-0.0152319	0.07396564
20000000.0	0.0044855761666666	0.0030990528333333	0.0115486099999999	0.0545632313576858	-0.0098646897875236	-0.0149333515881849	0.9983499069412224	-0.0215151583333333	-0.0145975916666666	0.0737863916666666
25000000.0	0.0043806223333333	0.0030773366666666	0.0117928399999999	0.0549670346500581	-0.0099280877120111	-0.014740475521052	0.998329993184644	-0.0231053166666666	-0.0139632833333333	0.0736071433333333
30000000.0	0.0042756685	0.0030556205	0.01203707	0.0553708266921017	-0.0099914836044761	-0.0145475964369255	0.9983098763634082	-0.024695475	-0.013328975	0.0734278949999999
35000000.0	0.0041707146666666	0.0030339043333333	0.0122813	0.055774074011709	-0.0100548774519432	-0.0143547143752826	0.9982895545801708	-0.0262856333333333	-0.0126946666666666	0.0732486466666666
40000000.0	0.0040657608333333	0.0030121881666666	0.01252553	0.0561783766946222	-0.0101182692414372	-0.0141618293756015	0.998269028472912	-0.0278757916666666	-0.0120603583333333	0.0730693983333333
45000000.0	0.003960807	0.002990472	0.01276976	0.0565821344898146	-0.0101816589599834	-0.0139689414773605	0.9982482980458324	-0.02946595	-0.01142605	0.07289015
50000000.0	0.00390328375	0.00300229975	0.0128401975	0.0568467445693149	-0.010241445519636	-0.0137800200359769	0.998235278615892	-0.023180431	-0.0082128855	0.0574620875

Model Architecture: LSTM

LSTM stands for Long Short-Term Memory and can predict future values based on previous sequential data and learn long-term dependencies.

The **input gate** decides what information will be stored in long term memory.

The **forget gate** decides which information from long term memory be kept or discarded.



The **output gate** produces new short term memory which will be passed on to the cell in the next time step.

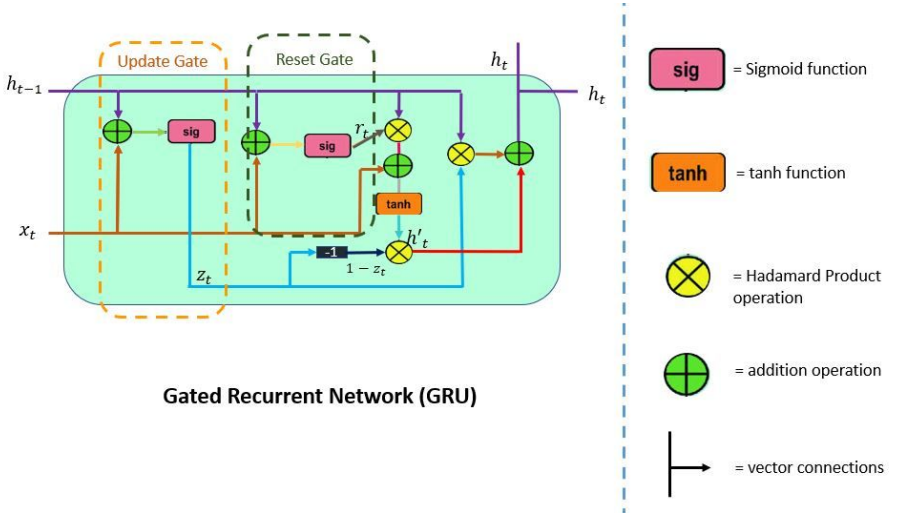
Picture source: hhttps://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

Model Architecture: GRU

GRU stands for Gated recurrent unit and according to researchers is 29.29% faster than LSTM in training speed for processing the same dataset.

The **update gate** sets amount of previous information to pass along the next state.

The **reset gate** decides whether the previous cell state is important or not.



Gated Recurrent Network (GRU)

With powerful update gate the model can decide to copy all the information from the past and eliminate the risk of vanishing gradient.

Picture source: hhttps://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

Problem statement and introduction 01

02 Dataset and data exploration

Models 03

04 Evaluation

Evaluation metrics

Experiments 05

06 Analyze

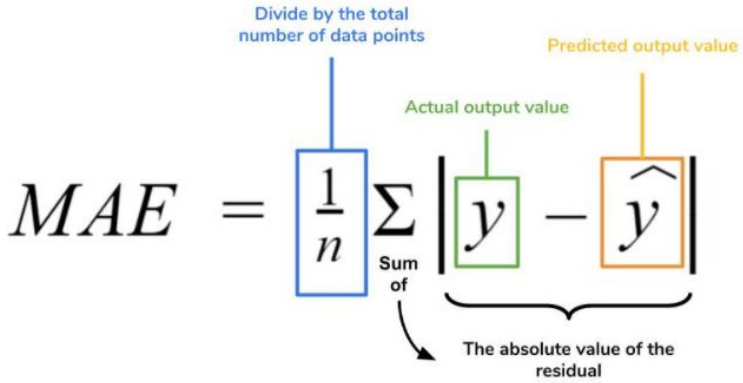
Evaluation metrics

First, model must be trained and validated.

The test dataset that was not seen during training will be used to make new predictions.

Predictions then must be compared to the real values in the test dataset.

Mean Absolute Error (MAE) measures the average magnitude of the errors without considering their direction.



Picture source:https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031

Evaluation metrics

Root mean squared error (RMSE) measures the average magnitude of the error.
Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors.

Euclidean distances is used for position and angular distance in radians between two quaternions is used for rotation.

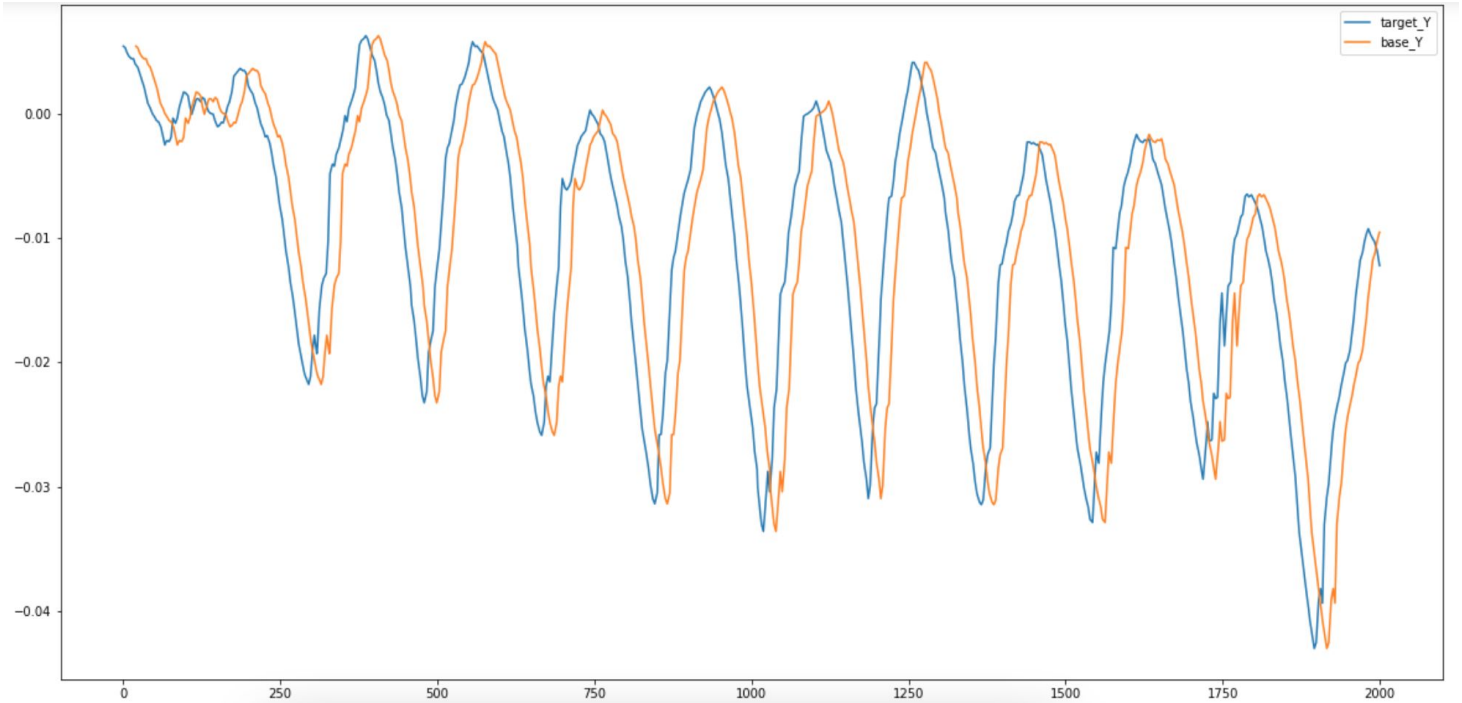
$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Picture source: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

Baseline: y-axis

MAE: position: 0.068m
rotation: 24.8°

RMSE: position: 0.072m
rotation: 29,36°



A baseline in forecast performance provides a point of comparison.

The plot of a baseline model predictions shows that the model is n-step behind reality.

Problem statement and introduction 01

02 Dataset and data exploration

Models 03

04 Evaluation

Experiments

05

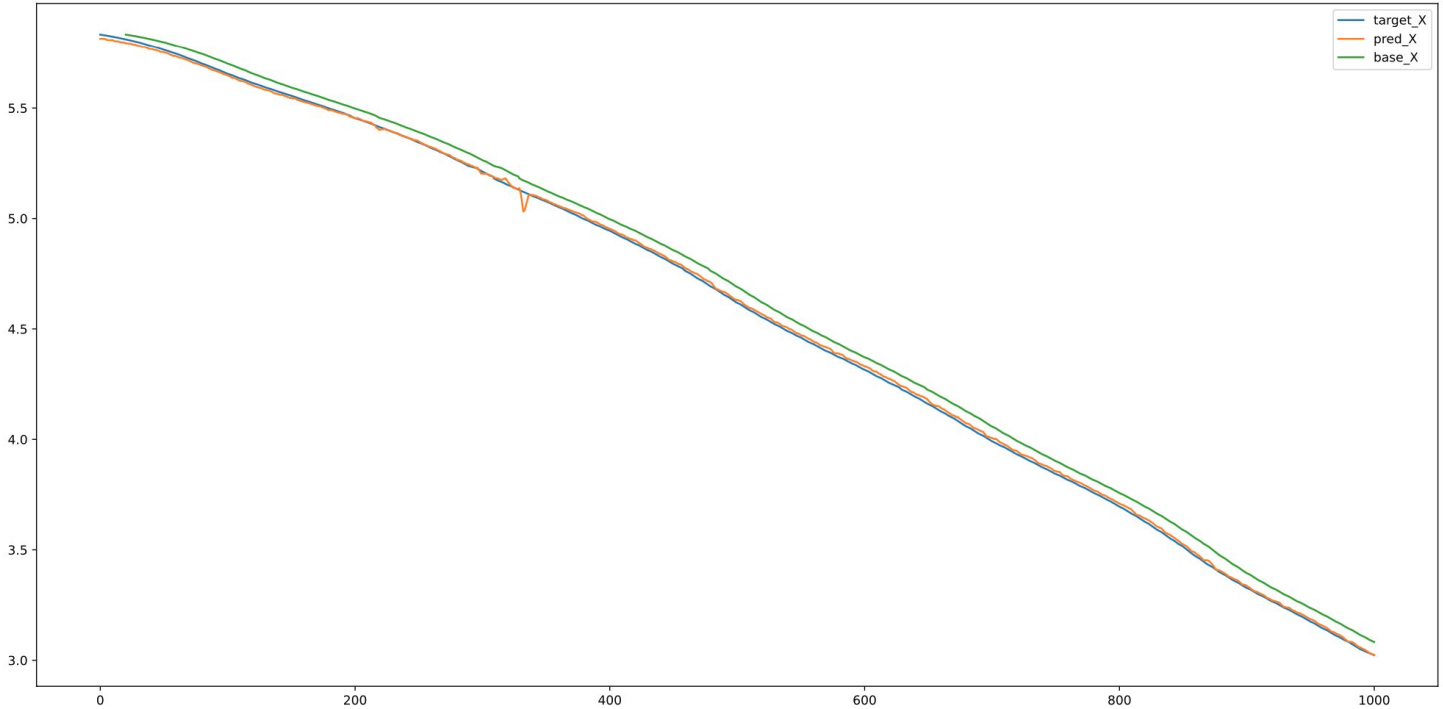
Results of experiments of different proprocessed RNN variants

06 Analyze

LSTM: x-axis

MAE: position: 0.014m↓
rotation: 17.01°↓

RMSE: position: 0.017m↓
rotation: 21,1°↓



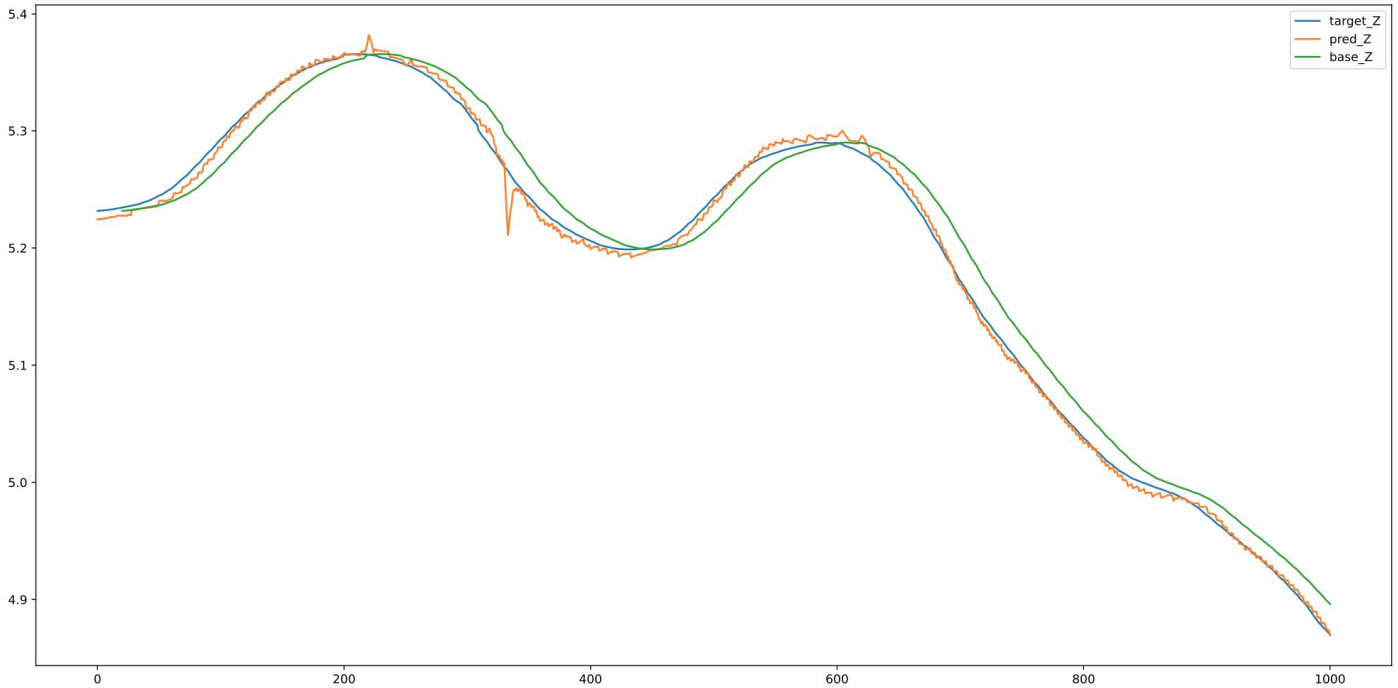
LSTM predicts better then Baseline Model and predictions follow the behaviour of the dataflow.

Both MAE and RMSE are lower for position and rotation predictions.

LSTM: z-axis

MAE: position: 0.014m↓
rotation: 17.01°↓

RMSE: position: 0.017m↓
rotation: 21,1°↓



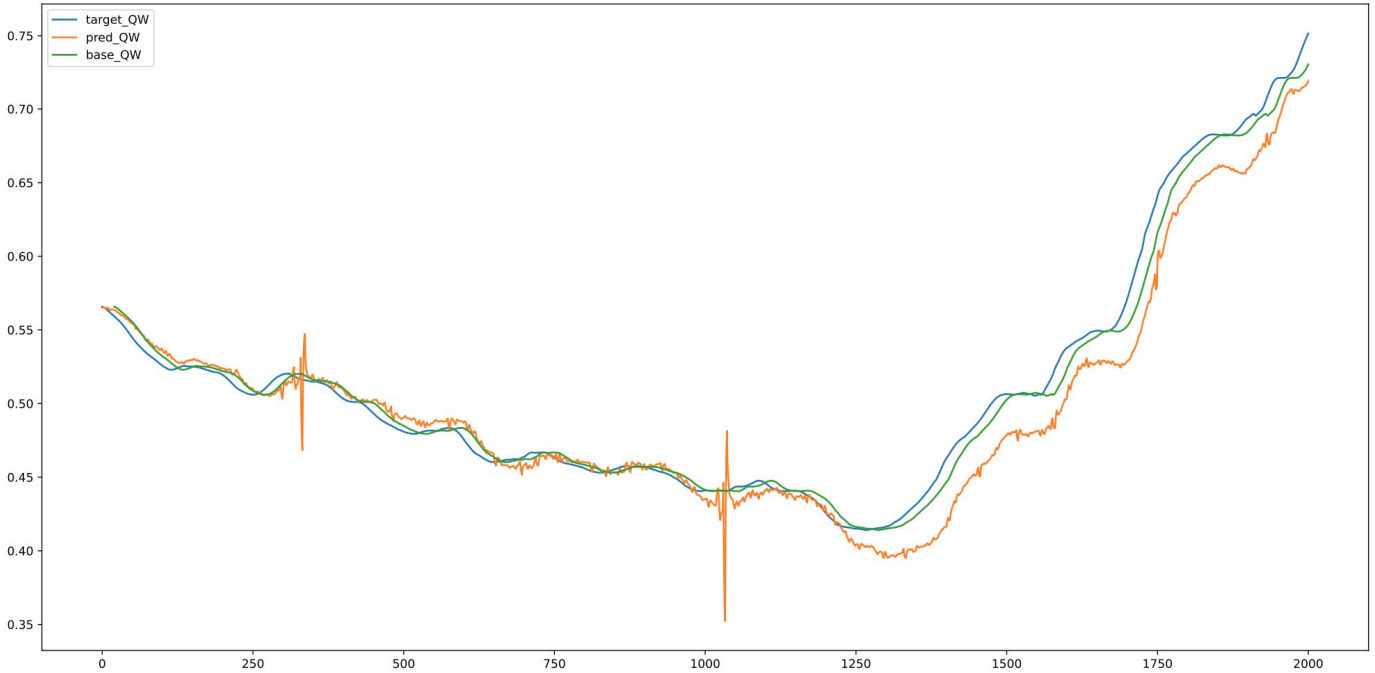
LSTM predicts better then Baseline Model and predictions follow the behaviour of the dataflow.

Both MAE and RMSE are lower for position and rotation predictions.

LSTM: qw-axis

MAE: position: 0.014m↓
rotation: 17.01°↓

RMSE: position: 0.017m↓
rotation: 21,1°↓



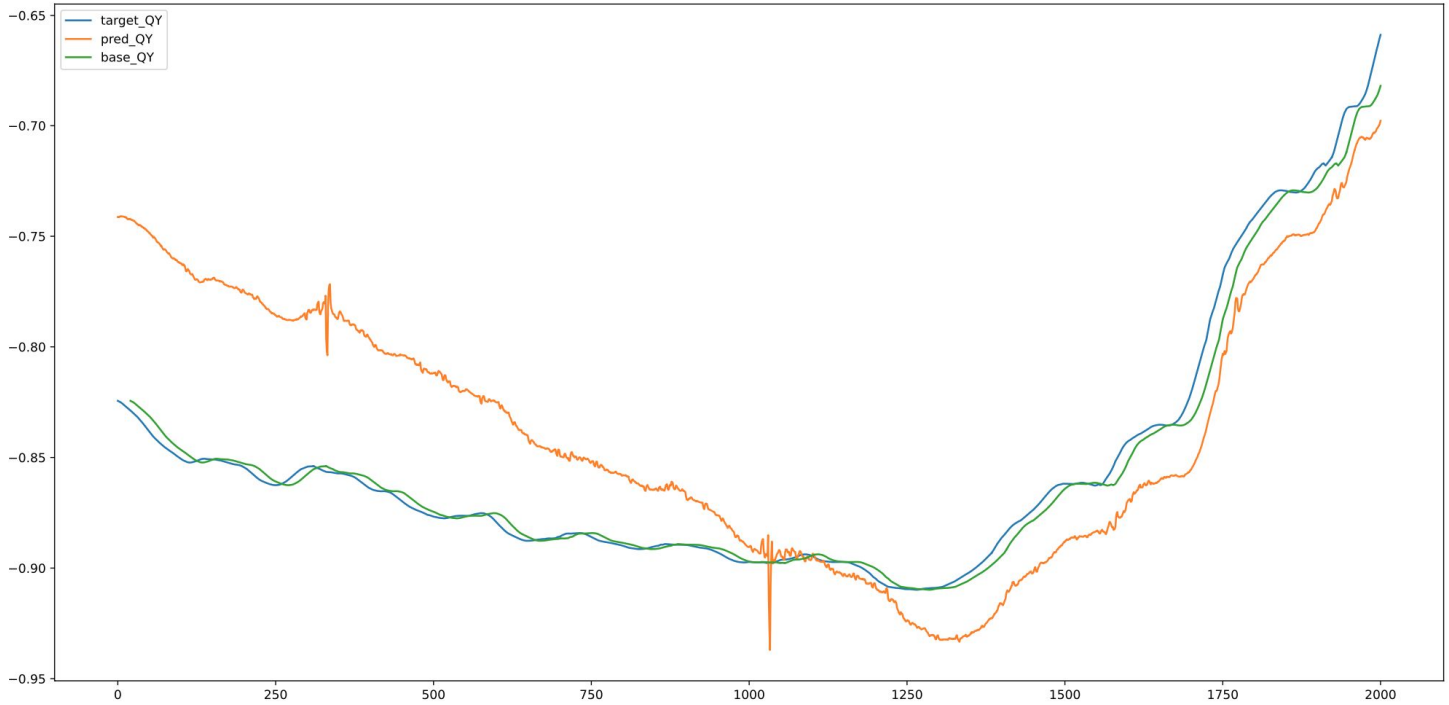
When the data is not rising and is not falling but fluctuates, LSTM is unable to properly predict following motion and tries to go up and down.

Finally LSTM catches the dataflow again.

LSTM: qy-axis

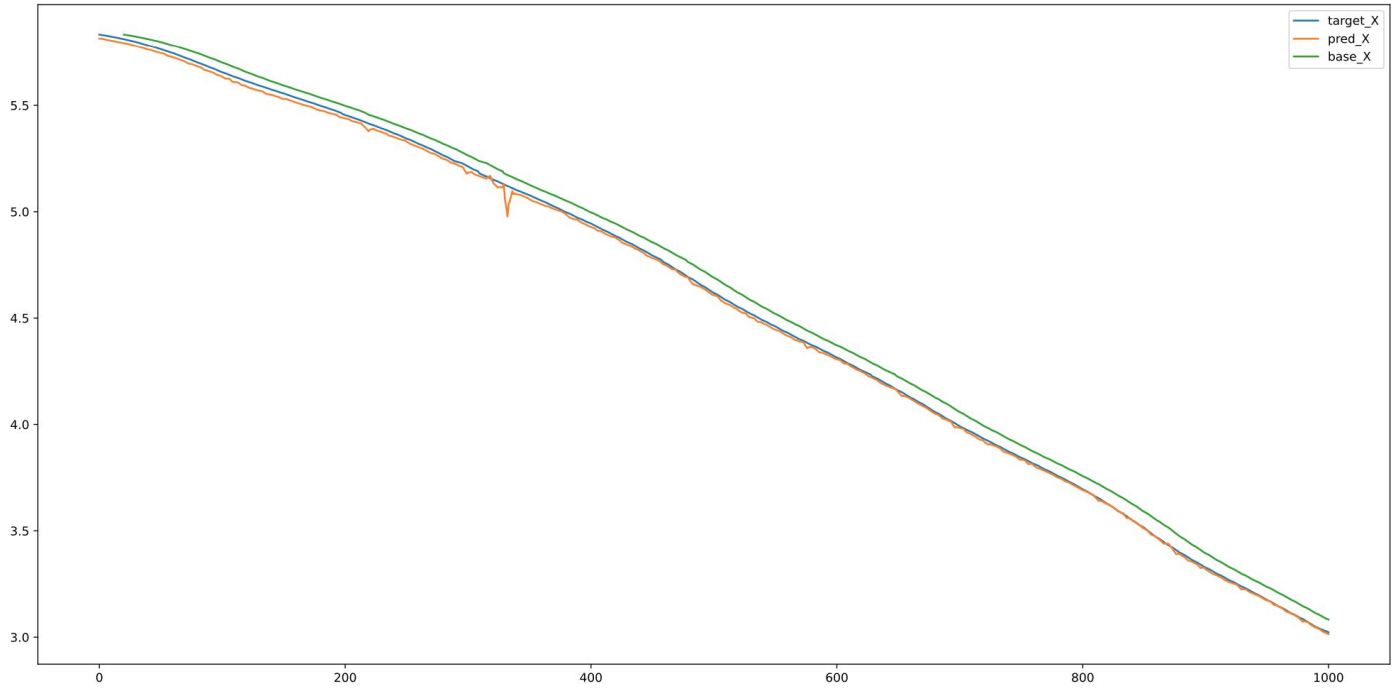
MAE: position: 0.014m↓
rotation: 17.01°↓

RMSE: position: 0.017m↓
rotation: 21,1°↓



GRU: x-axis

MAE: position: 0.0099m↓ RMSE: position: 0.011m↓
rotation: 13,9°↓ rotation: 18,7°↓

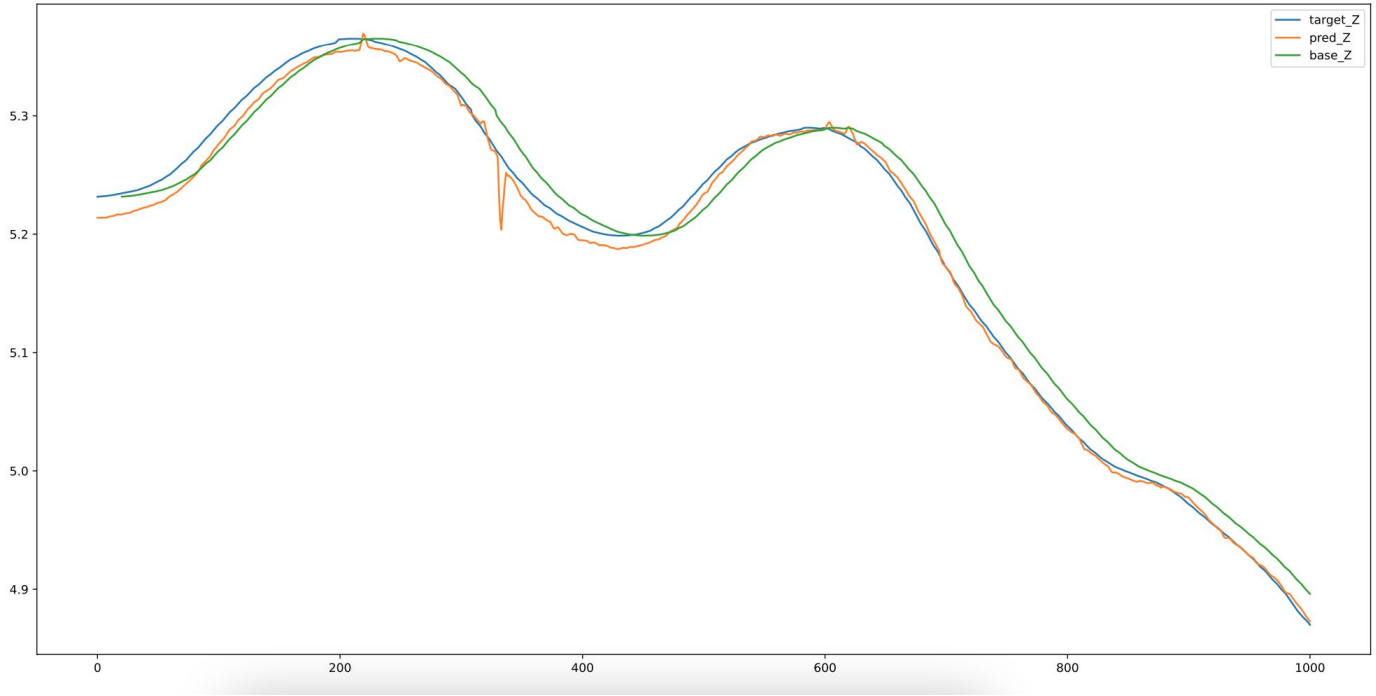


GRU predicts better then LSTM Model and predictions smoothly follow the behaviour of the dataflow.

Both MAE and RMSE are lower for both position and rotation predictions!

GRU: z-axis

MAE: position: 0.0099m↓ RMSE: position: 0.011m↓
rotation: 13,9°↓ rotation: 18,7°↓

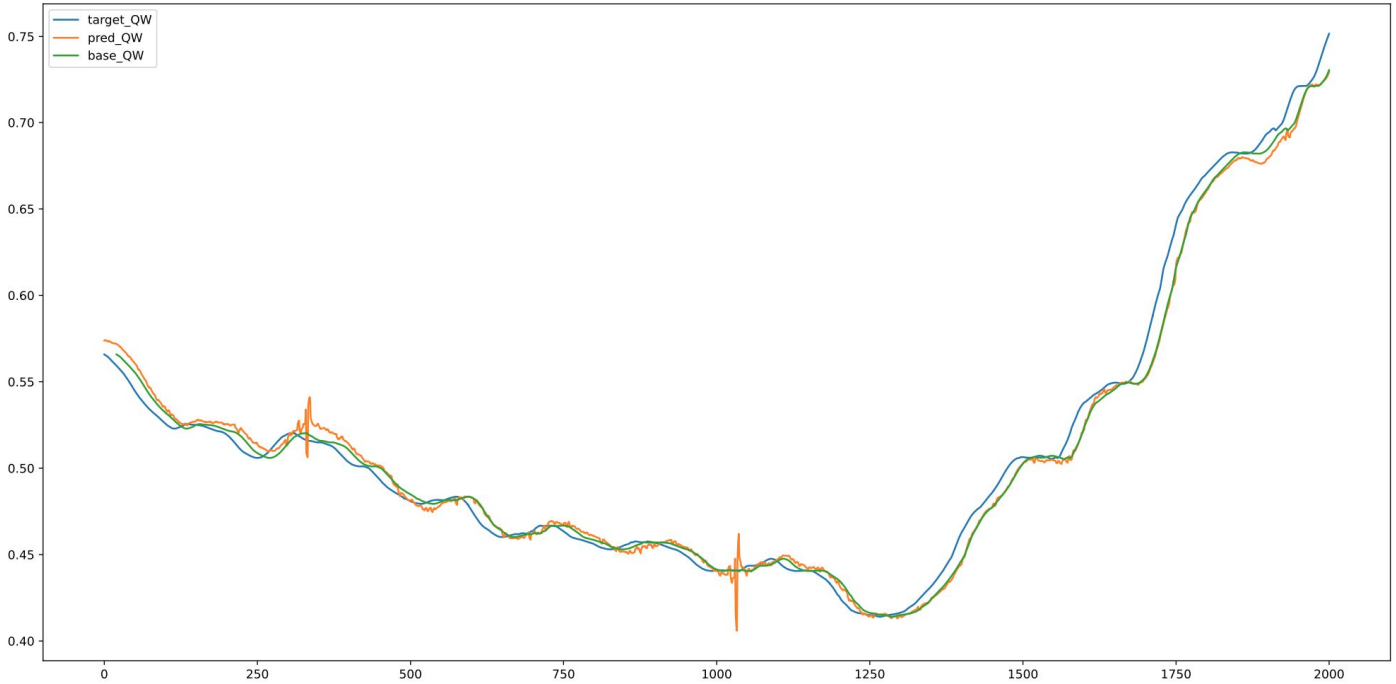


GRU predicts better then LSTM Model and predictions smoothly follow the behaviour of the dataflow.

Both MAE and RMSE are lower for both position and rotation predictions!

GRU: qw-axis

MAE: position: 0.0099m↓ RMSE: position: 0.011m↓
rotation: 13,9°↓ rotation: 18,7°↓

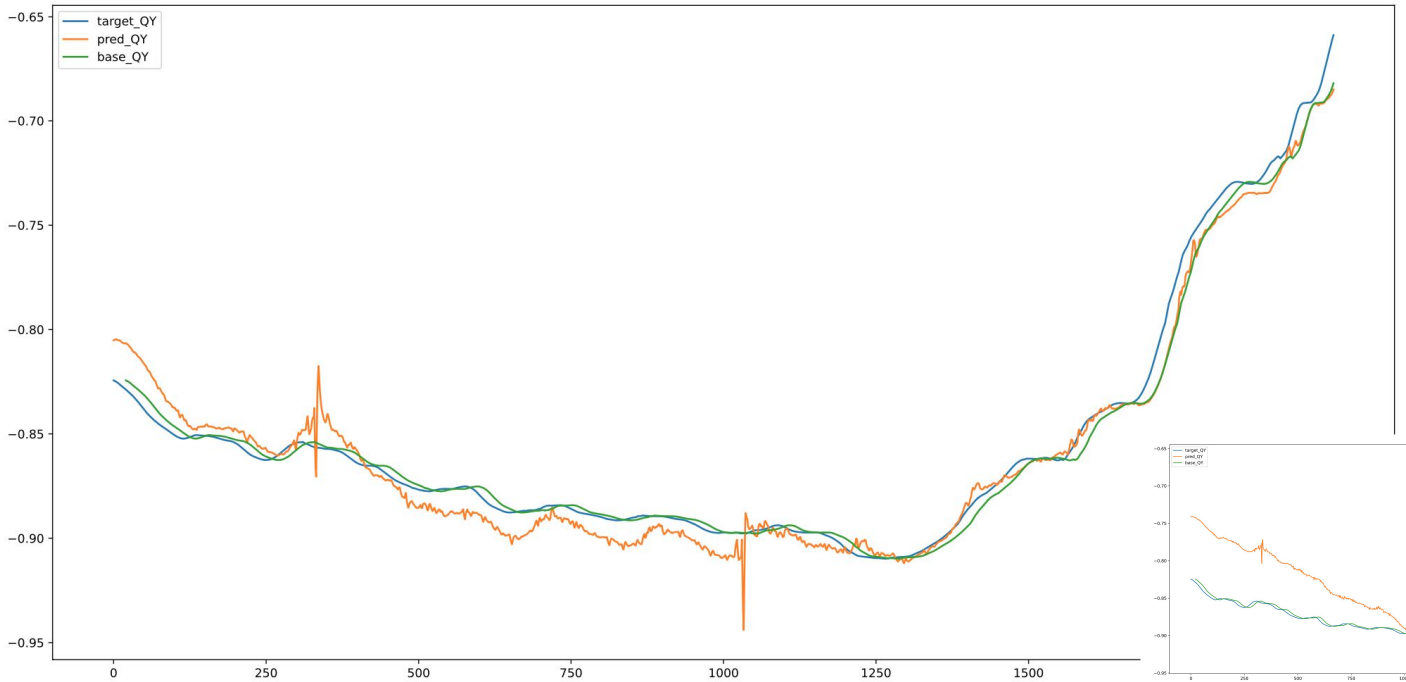


GRU predicts better then LSTM Model and predictions smoothly follow the behaviour of the dataflow.

Both MAE and RMSE are lower for both position and rotation predictions!

GRU: qy-axis

MAE: position: 0.0099m↓ RMSE: position: 0.011m↓
rotation: 13,9°↓ rotation: 18,7°↓



GRU predicts better then LSTM Model and predictions smoothly follow the behaviour of the dataflow.

Both MAE and RMSE are lower for both position and rotation predictions!

Parameters of best models

Criterion: MSELoss Optimizer: Adam

Parameters	LSTM Model	GRU Model
Input size	10	10
Hidden size	1024	512
Output size	7	7
Layers	1	1
Learning rate	1e-4	1e-4
Adaptive learning rate	every 30 epochs reduces by 50%	every 50 epochs reduces by 70%
Weight decay	1e-14	1e-12
Sequence length	20 past values (100 ms)	20 past values (100 ms)

How the hyperparameters were found?

The hyperparameters search is done using GPU cluster of Fraunhofer HHI.

Cluster works with the SLURM resource manager/scheduler.

The Singularity alike docker container system was used to containerize the application with the required environment variables needed for model initialization.

Exhaustive Grid Search run few thousands jobs in order to find the best parameters for LSTM and GRU Models.

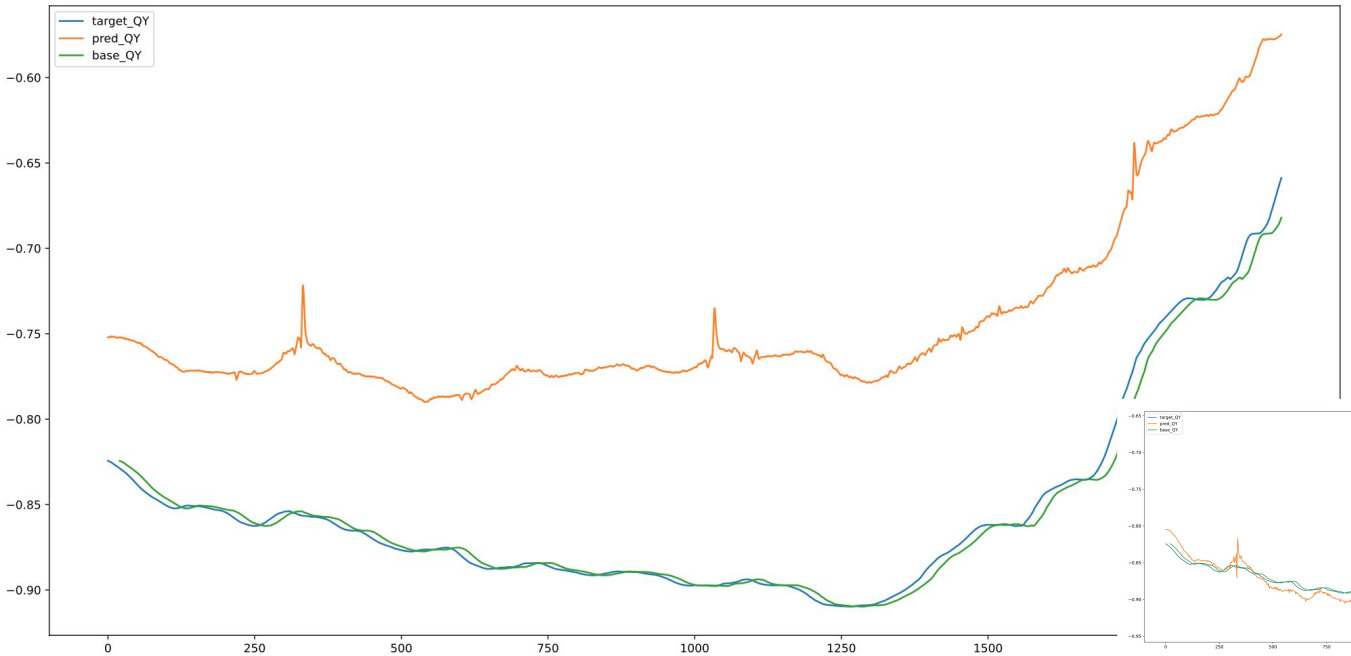
The simplest architecture (1-layered LSTM and GRU) performed the best.

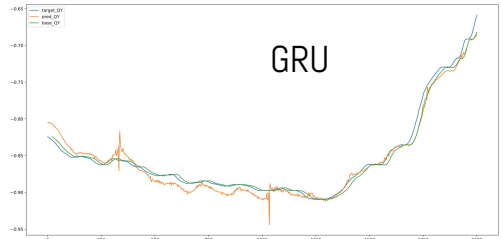
Mish Activation function used on additional layers improved LSTM performance by **16%.** Using of Mish-function with GRU gave worse result by **22%**. ReLU worsened results of both models.

Stacked architectures with/without dropout performed the worst.

The low prediction accuracy example: y-axis

MAE: position: 0.057m↑
rotation: 25.13°↑

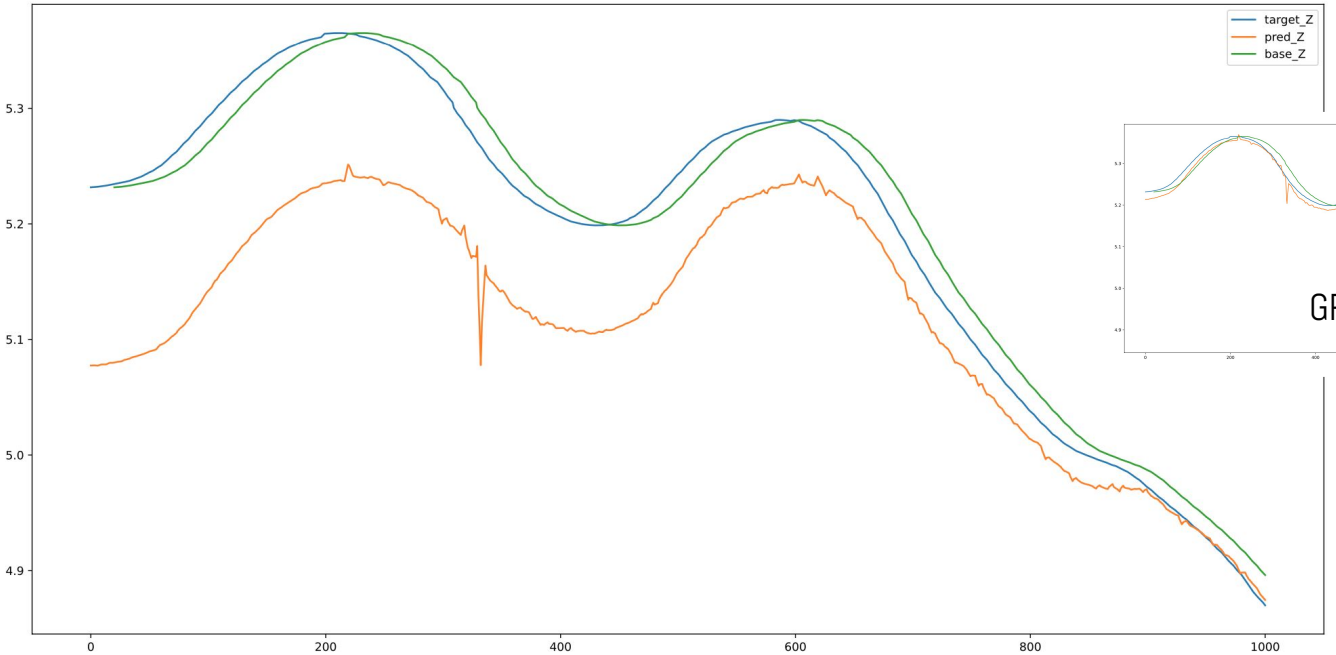




2-layered stacked LSTM with Mish Activation function, Linear layer and Dropout (0.1)

The low prediction accuracy example: z-axis

MAE: position: 0.061m↑
rotation: 27.18°↑



Bidirectional GRU

Problem statement and introduction

01

02

Dataset and data exploration

Models

03

04

Evaluation

Experiments

05

06

Analyze

Discussion of the obtained results and steps for finishing the research

Conclusion

LSTM and GRU models with different variations were applied in this research to obtained from HoloLens 2 6-DoF dataset.

Conducted experiments showed that the **LSTM-based architecture leads to a significant improvement** of the MAE and RMSE metrics.

Although best performance for head motion prediction with LSTM was achieved, the another RNN variant GRU was tried.

GRU is computationally **more efficient, as it has fewer parameters** and states than LSTM units.

GRU-based network is able to improve the MAE and RMSE compared to LSTM.
GRU predicts by 28,8% preciser! -> BEST Result!

Final steps:

1. Try the combination of LSTM/GRU Models with CNN in order to try to improve achieved MAE and RMSE metrics.

2. Deliver a Python Application with trained models for future usage in cloud-based streaming platform developed by Fraunhofer HHI

Thank you for your attention

Oleksandra Baga
Master Computer Science (MSC)
Bachelor of Engineering (B.Eng.)
E-Mail: oleksandra.baga@gmail.com
https://github.com/oleksa-oleksa