UAV State Estimation Using IMU Vector with DNN.

Group 7

ITCS 5156

Saiphani Jasthi, Christian Micklisch, Rohit Alavala,
Yuehan Lan

Problem & Challenges

Problem:

- State Estimation is difficult and we would like to re-create a Extended Kalman Filter. An Extended Kalman Filter, simply takes the gyroscope (angular data) and accelerometer (linear data) data and determines the linear position (x, y, and z) from it. A kalman Filter determines the angular positions (phi, theta, and psi).

Challenges:

- The sensor that we train the model on determines the sensor that we need to utilize when feeding data to the model
- There is no data available to recreate Al-Sharman et al.'s model, let alone train our own model.

While there exist state estimators for Unmanned Aerial Vehicles, UAVs, in simulated environments few rely on strictly the raw data produced by inertial measurement units, IMUs. Most estimators that utilize machine learning models rely on, on/off-board optical sensors, and digital filters to train the models on state estimation. Our implementation aims to estimate the state by providing raw IMU data to a DNN with simulated data, using Micklisch et al.'s simulator, of 501 random quadrotor flights.

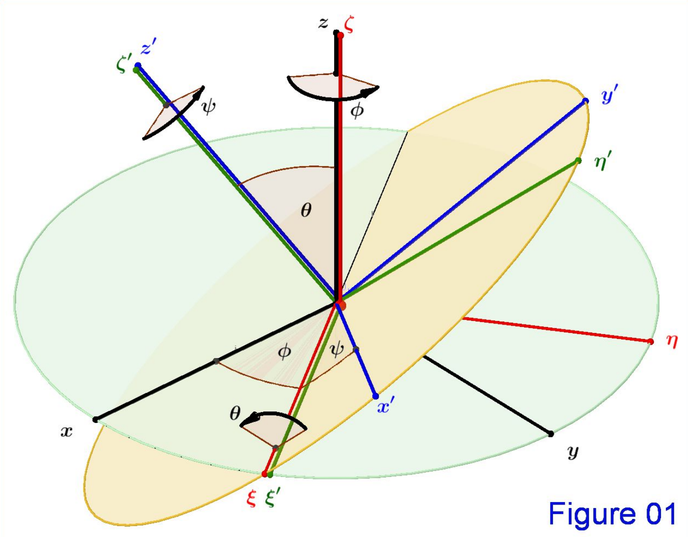


Figure 01

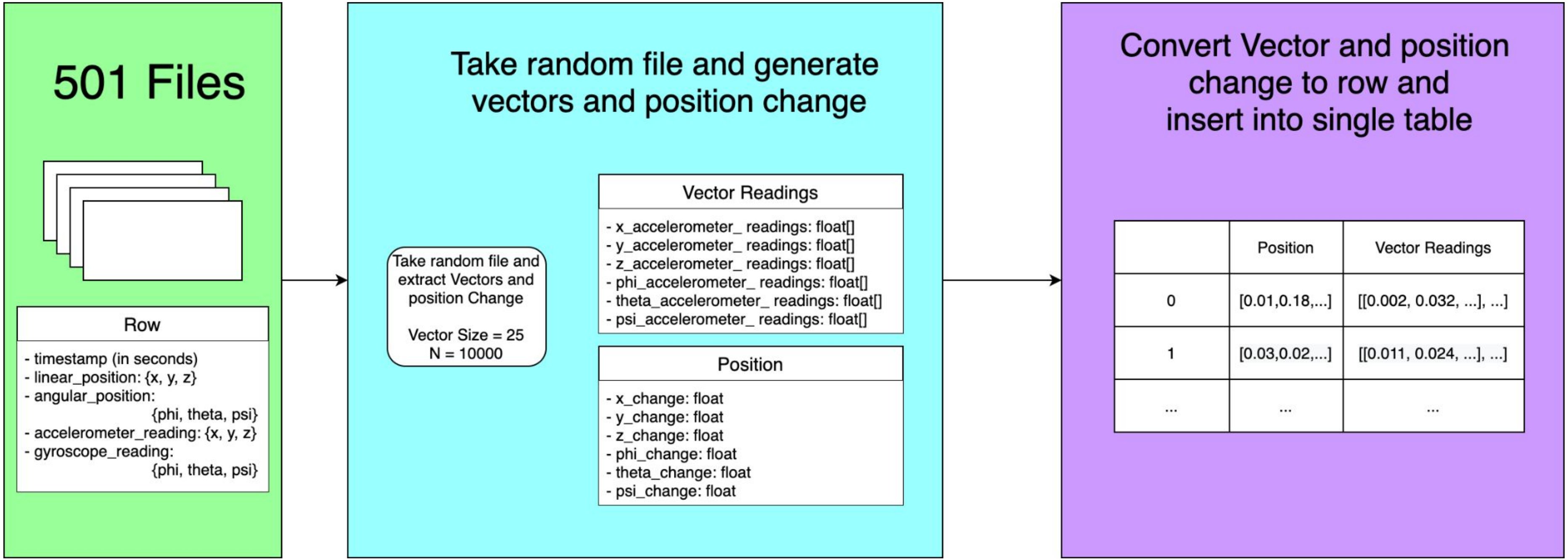
Motivation

One of the most difficult areas of designing a UAV is state estimation, especially with common off-the-shelf components. Current commercial IMU's are cheaply available but contain sensor error dynamics such as noise and drift.

Existing Related Approaches

These error dynamics are usually resolved by the use of an extended Kalman filter (EKF), reviewed by Yang et al., but are complex to implement and require background knowledge of the Inertial body of the UAV. Al-Sharman et al. replaced the Kalman filter and fed the raw IMU data directly to the ML model. This resulted in an RMSEE score for the angular positions phi, theta, and psi of 0.00016, 0.00023, and 0.00200 respectively. We want to see if we can improve the accuracy of this model by sending the raw IMU data as a vector of recorded readings over a time period and predict not only the attitude, angular, position but the linear position as well.

Our Data Processing



	Position	Vector Readings
0	[0.01,0.18,...]	[[0.002, 0.032, ...], ...]
1	[0.03,0.02,...]	[[0.011, 0.024, ...], ...]
...	...	...

Our targets

- x_change
- y_change
- z_change
- phi_change
- theta_change
- psi_change

Our Method

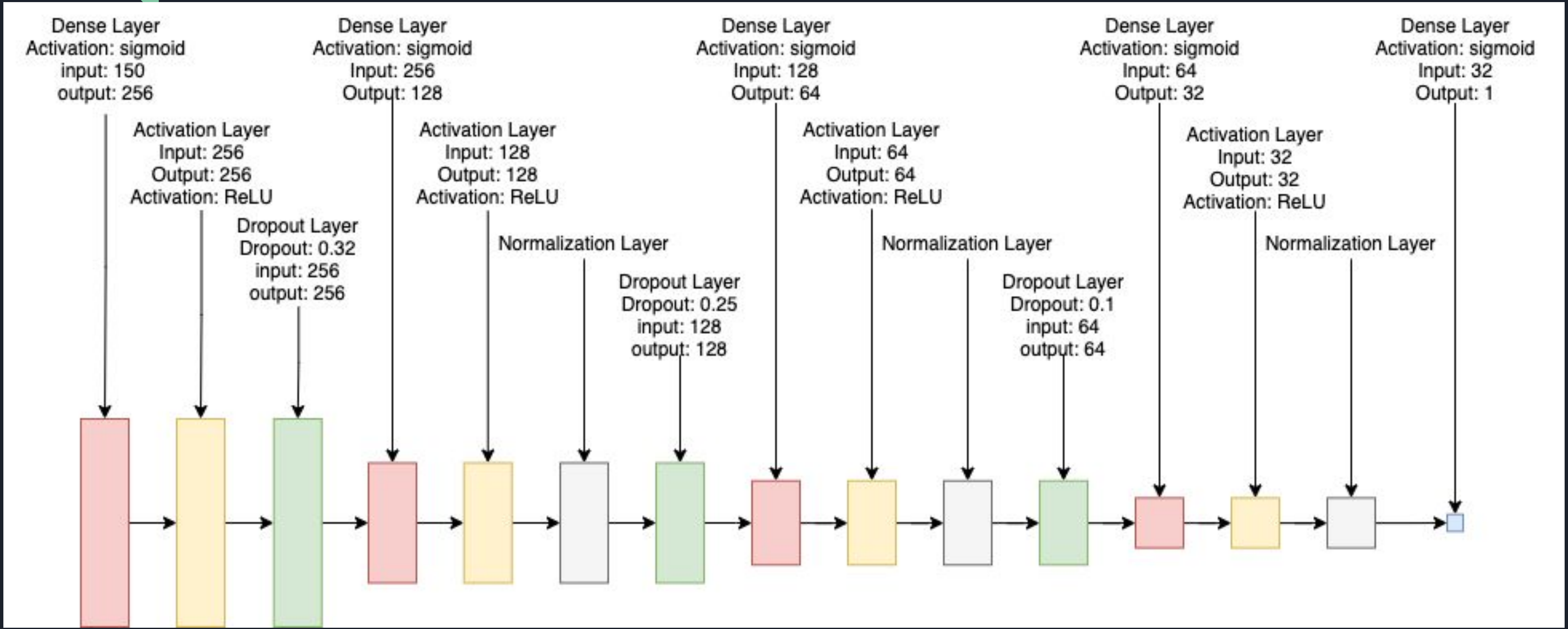
- For our approach we trained separate models to predict each of our targets individually.
    - This gave significant performance boost.
- We tried hundreds of different DNN sequential layer structures.
    - Used Keras libraries
- We tried many different activations such as:
    - Sigmoid, Softmax, Softplus, Softsign, Tanh, Selu, Elu
- We added different types of normalization layers such as:
    - LayerNormalization, and BatchNormalization
- Dropout layers were also experimented with
- Different epochs, and batch_size were also experimented with

Best structures we found in our experimentation.

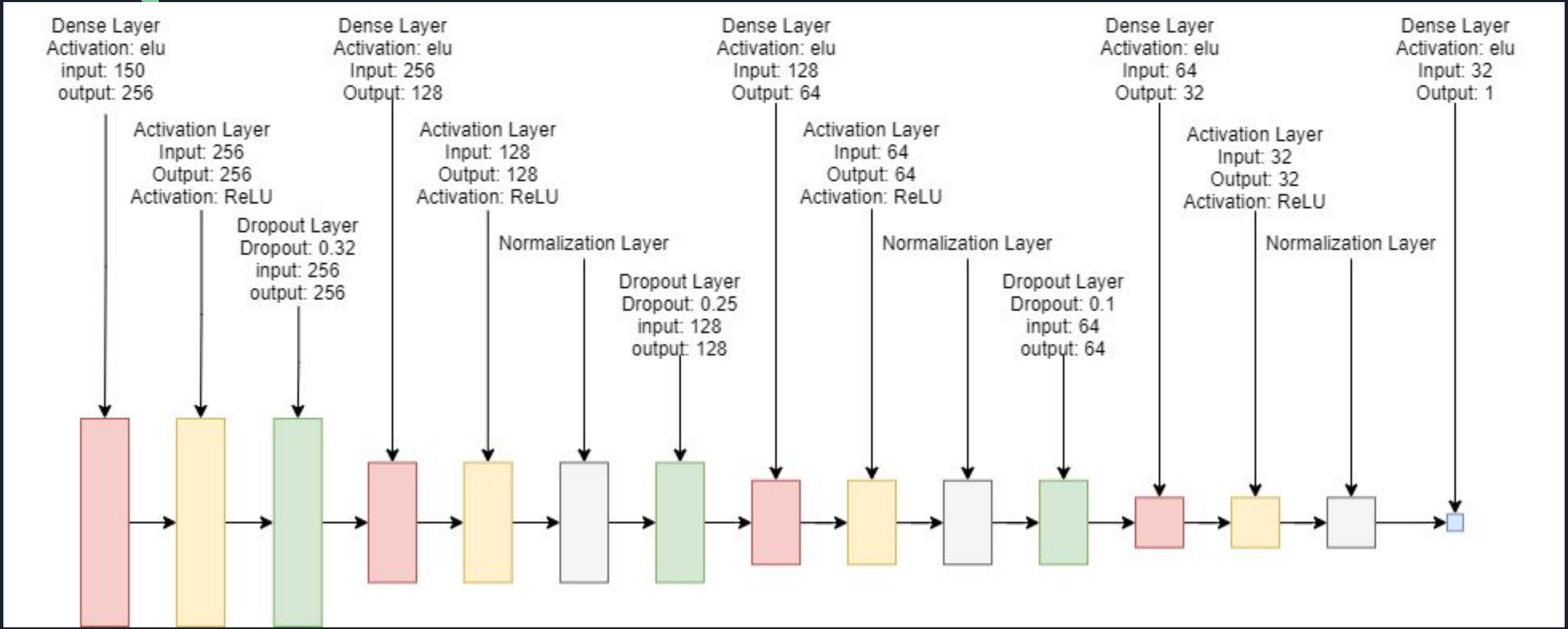
Everything we tried

Targets	Optimizer	Learning rate	Activation	Epochs	Batch Size	Dropout
x_change y_change z_change phi_change theta_change psi_change	sgd	0.01,0.001, 0.0001	selu,elu, relu, sigmoid, softmax, softplus, Softsign, tanh	30-350	6,12	0.1-0.9
Same as above	adam	0.001	Same as above	30-350	6,12	Same as above
Same as above	RMSprop	0.001	Same as above	30-350	6,12	Same as above
Same as above	Adagrad	0.001	Same as above	30-350	6,12	Same as above
Same as above	Adadelta	0.001	Same as above	30-350	6,12	Same as above

Phi_change, Theta_change, Psi_change Models



X_change, Y_change, Z_change Models



What worked best

Target	Optimizer	Learning rate	Activation	Epochs	Batch Size	Dropout
x_change y_change	adam	0.001	elu	150	6	0.32,0.25, 0.1
z_change	adam	0.001	elu	250	6	0.32,0.25, 0.1
phi_change theta_change psi_change	sgd	0.01	sigmoid	100	12	0.32,0.25, 0.1

Observations and Results

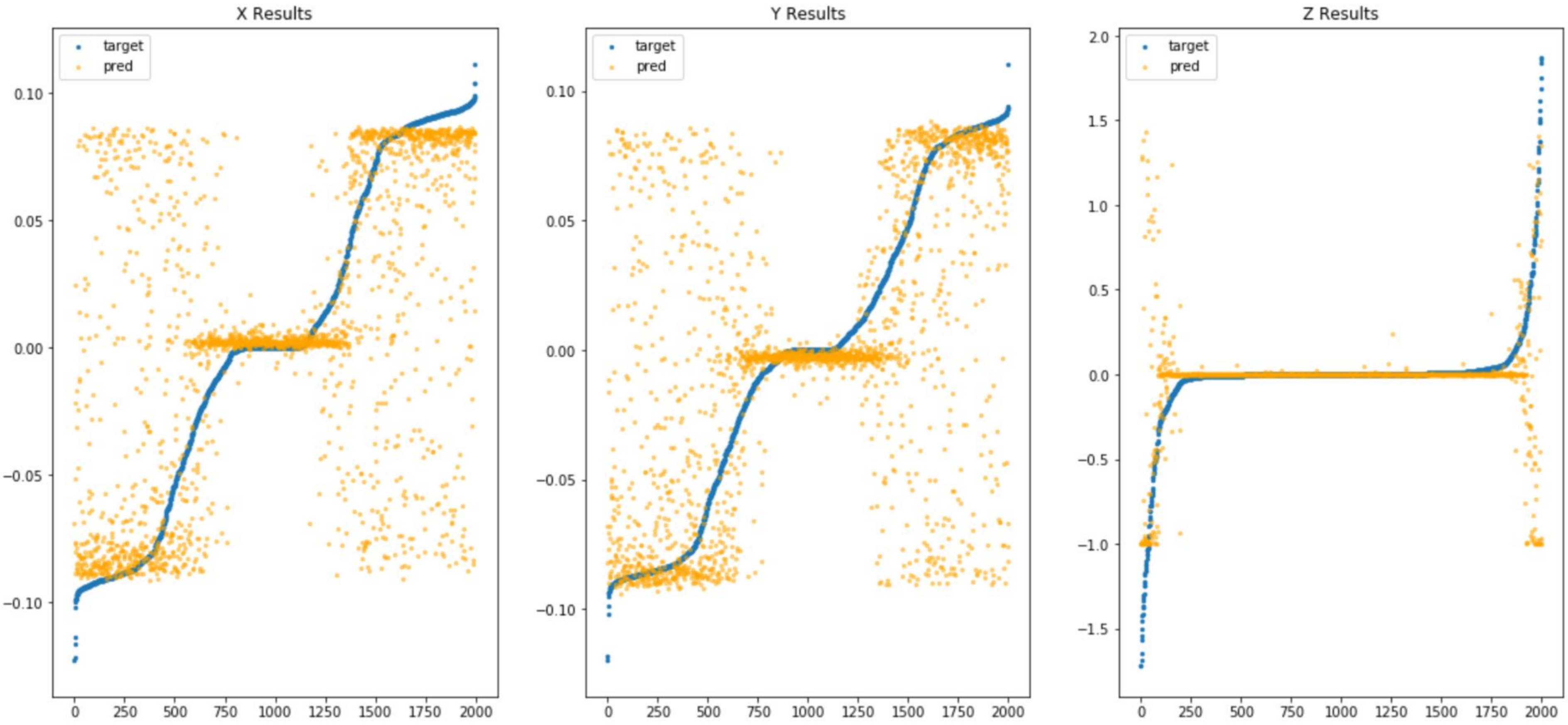
Our Best Metrics

Target	R2	max_err	MAE	MAPE	MSE	RMSE
x	1.65e-01	1.83e-01	3.44e-02	5.93e+03	3.50e-03	1.22e-05
y	1.29e-01	1.83e-01	3.29e-02	1.79e+03	3.17e-03	1.01e-05
z	-4.07e-01	2.96e+00	9.61e-02	6.63e+02	1.14e-01	1.30e-02
phi	-1.68e+00	1.59e-02	8.75e-04	9.45e+05	1.13e-06	1.29e-12
theta	-9.32e-01	1.09e-02	6.36e-04	2.26e+05	7.18e-07	5.15e-13
psi	-1.41e-02	1.05e-01	7.74e-04	3.20e+05	9.62e-06	9.25e-11

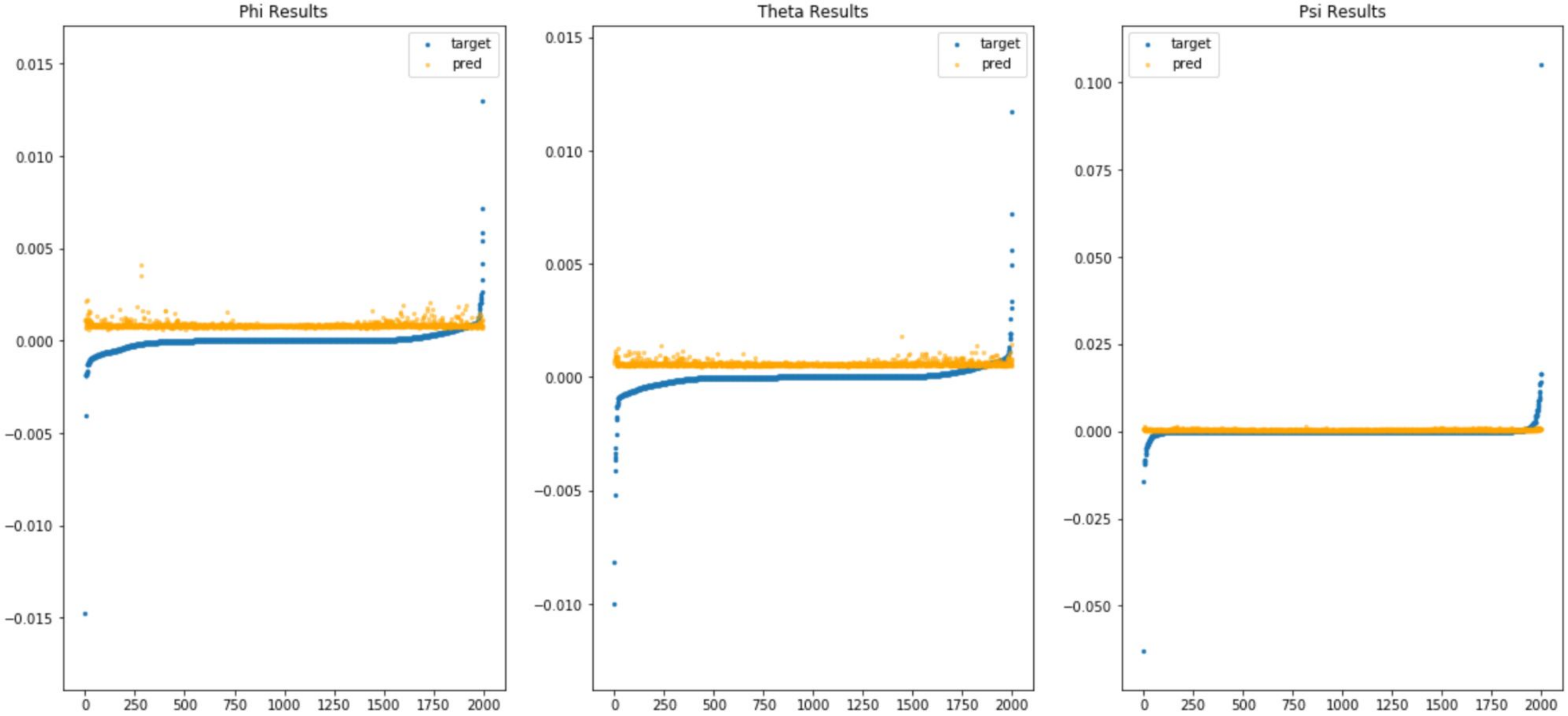
Al Sharman et al. Results

	DLNN RMSEE
phi	0.00016
theta	0.00023
psi	0.00200

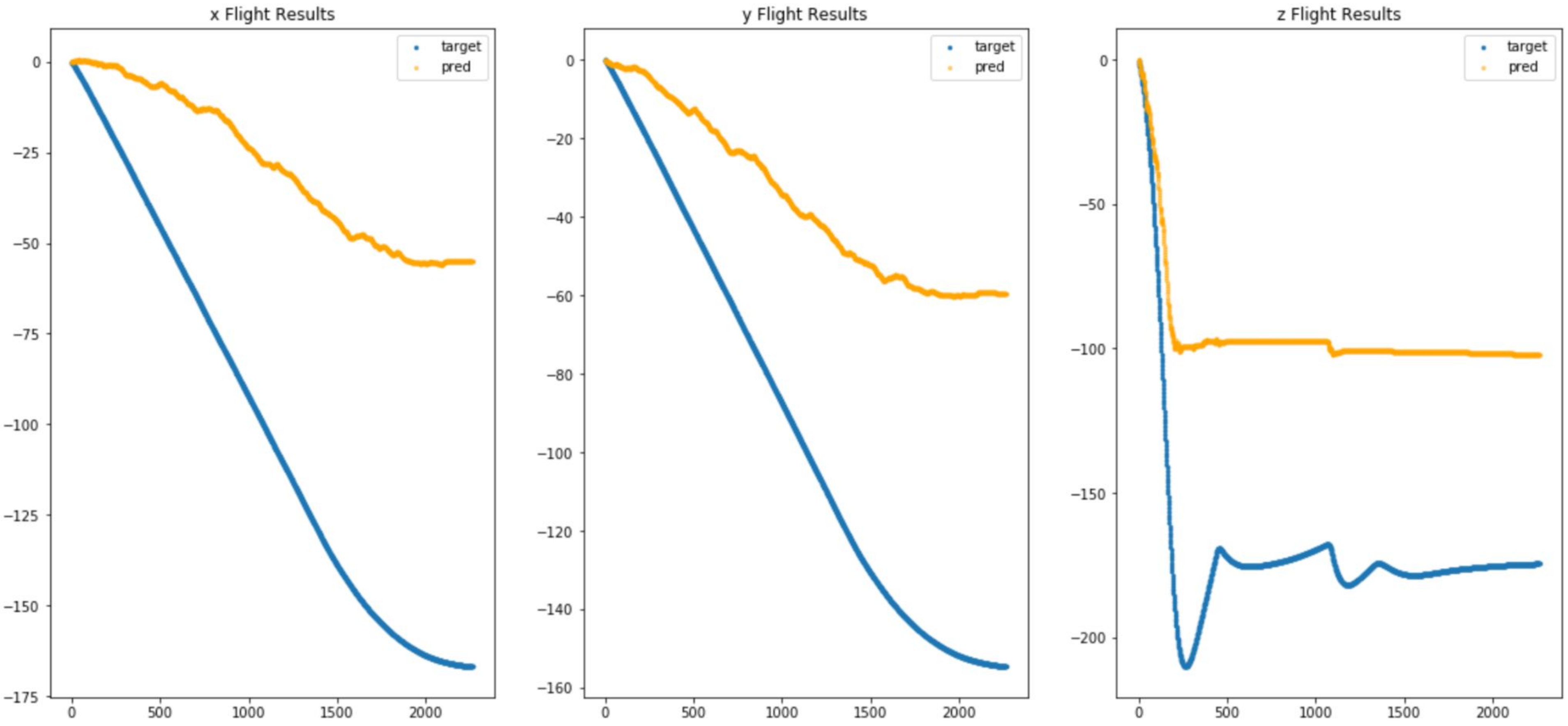
Sorted Targets to predicted points, X/Y/Z



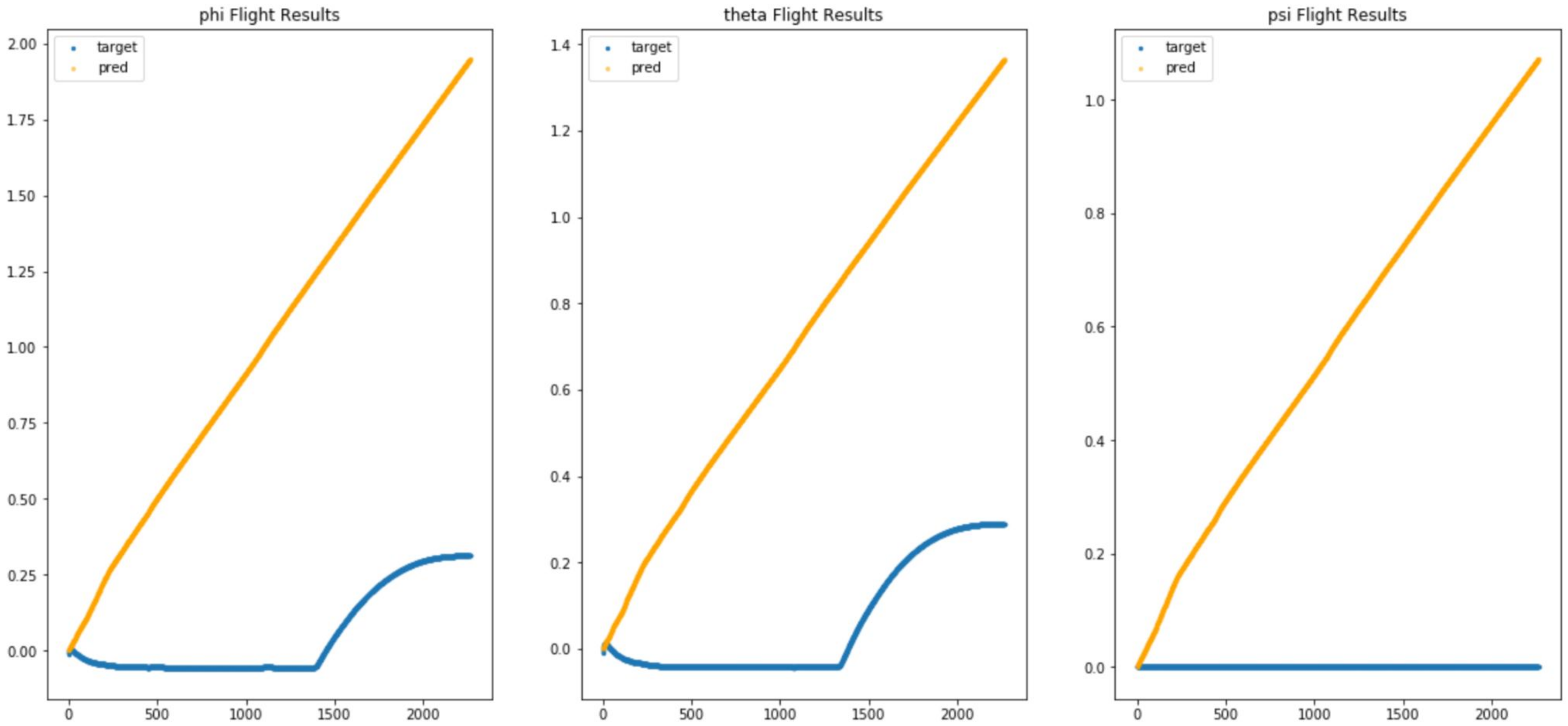
Sorted Targets to predicted points, Phi/Theta/Psi



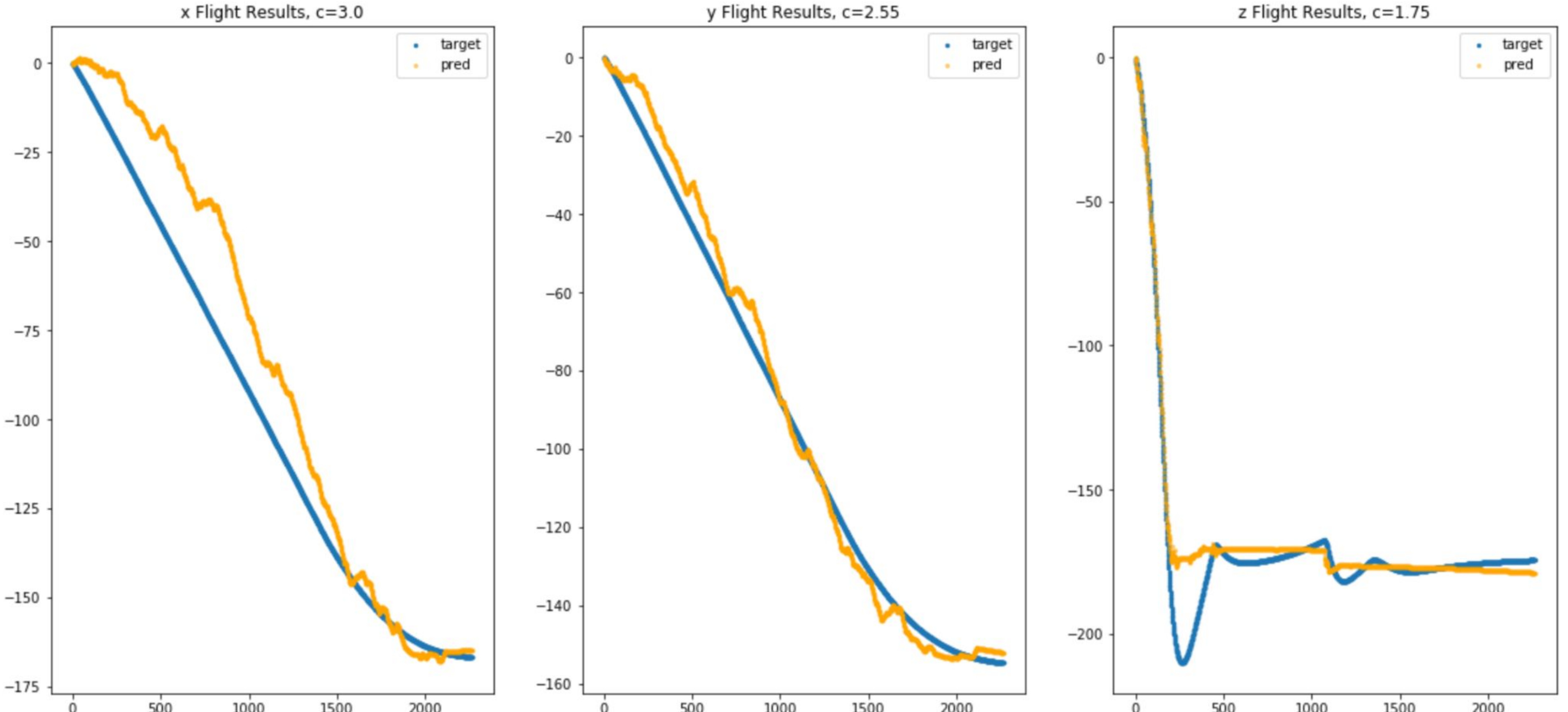
Flight Path Change to predicted Flight Path, X/Y/Z



Flight Path Change to predicted Flight Path, Phi/Theta/Psi



Flight Path Change to predicted Flight Path with Constants, X/Y/Z



Conclusion and Future Work

- In conclusion we were able to get fairly good results for x_change, y_change, and z_change.
- Phi_change, theta_change, and psi_change were a bit more difficult to predict as such the results were not the impressive.
- In the future we would like to experiment with
  - Recurrent neural networks using LSTM(Long short-term memory), and GRU(Gated recurrent units).
  - Transformation models
  - Try more learning rates with our current models