```python
In [1]: import pandas as pd
        from keras.models import Sequential
        from keras.layers import Dense, LayerNormalization , Dropout, LSTM, Embedding
        from keras.layers.advanced_activations import PReLU
        from sklearn.model_selection import train_test_split
        from tensorflow import keras
        import matplotlib.pyplot as plt
        #from sklearn.model_selection import train_test_split
        from sklearn.metrics import mean_squared_error
        import tensorflow as tf
        import numpy as np
```

```python
In [2]: df = pd.read_csv("row_size25_vector_size8000.csv")
```

```python
In [3]: X = df.iloc[:,7:157]
        #X = df[['accelerometer_reading_x_0','accelerometer_reading_y_0','accelerometer_reading_z_0','gyroscope_reading_phi_0','gyro
        #t = df[['linear_position_x','linear_position_y','linear_position_z','angular_position_phi','angular_position_theta','angula
        #t = df[['phi_change','theta_change','psi_change']]
        t = df[['phi_change']]
```

```python
In [4]: t.shape
Out[4]: (8000, 1)
```

```python
In [5]: in_dim = X.shape[1]
        out_dim = t.shape[1]
```

```python
In [6]: in_dim
Out[6]: 150
```

```python
In [7]: out_dim
Out[7]: 1
```

```python
In [8]: X_train, X_test, t_train, t_test = train_test_split(X, t, test_size=0.2)
```

```python
In [9]: t_train.shape
Out[9]: (6400, 1)
```

```python
In [10]: model = Sequential()
         model.add(Dense(256,input_dim=in_dim, activation="sigmoid"))
         model.add(PReLU())
         model.add(Dropout(.4))
         model.add(Dense(128, activation="sigmoid"))
         model.add(PReLU())
         model.add(LayerNormalization ())
         model.add(Dropout(.25))
         model.add(Dense(64, activation="sigmoid"))
         model.add(PReLU())
         model.add(LayerNormalization ())
         model.add(Dropout(.1))
         model.add(Dense(32, activation="sigmoid"))
         model.add(PReLU())
         model.add(LayerNormalization ())
         model.add(Dense(out_dim,activation="sigmoid"))
         model.compile(loss="mse", optimizer="sgd")
```

```python
In [11]: model.fit(X_train, t_train, epochs=75, batch_size=12, verbose=2)
```
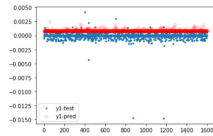
```
Epoch 67/75
534/534 - 1s - loss: 1.2153e-05
Epoch 68/75
534/534 - 1s - loss: 1.2253e-05
Epoch 69/75
534/534 - 1s - loss: 1.2481e-05
Epoch 70/75
534/534 - 1s - loss: 1.3217e-05
Epoch 71/75
534/534 - 1s - loss: 1.0489e-05
Epoch 72/75
534/534 - 1s - loss: 1.4014e-05
Epoch 73/75
534/534 - 1s - loss: 1.1906e-05
Epoch 74/75
534/534 - 1s - loss: 1.1045e-05
Epoch 75/75
534/534 - 1s - loss: 1.0297e-05
```
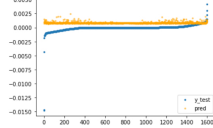
```
Out[11]: <tensorflow.python.keras.callbacks.History at 0x244120a5e20>
```

```python
In [12]: ypred = model.predict(X_test)
         print("y1 MSE: ", mean_squared_error(t_test.iloc[:, 0], ypred[:,0]))
         #print("y2 MSE: ", mean_squared_error(t_test.iloc[:, 1], ypred[:,1]))
         #print("y3 MSE: ", mean_squared_error(t_test.iloc[:, 2], ypred[:,2]))
```

```
y1 MSE:  1.24806557725677733e-06
```

```python
In [13]: x_ax = range(len(X_test))

         plt.scatter(x_ax, t_test.iloc[:, 0],  s=6, label="y1-test")
         plt.scatter(x_ax, ypred[:,0], label="y1-pred",c="red",alpha = 0.1)

         plt.legend()
         plt.show()
```



```python
In [14]: x_ax = range(len(X_test))

         y_test_index = np.argsort(t_test.iloc[:, 0], axis=0).to_numpy()

         f = plt.figure()
         plt.scatter(x_ax, t_test.iloc[y_test_index], s=6, label="y_test")
         plt.scatter(x_ax, ypred[y_test_index], s=6, label="pred",c="orange", alpha=0.5)
         #plt.ylim(t_test.iloc[y_test_index[0]].to_numpy()[0])
         plt.legend()
         plt.show()

         f.savefig("foo.pdf", bbox_inches='tight')
```



```python
In [ ]:
```

```python
In [ ]:
```