

```
In [1]: 1 import pandas as pd
2 from keras.models import Sequential
3 from keras.layers import Dense, LayerNormalization, Dropout, LSTM, Embedding
4 from keras.layers.advanced_activations import PReLU, ReLU
5 from sklearn.model_selection import train_test_split
6 from tensorflow import keras
7 import matplotlib.pyplot as plt
8 from sklearn.model_selection import train_test_split
9 from sklearn.metrics import mean_squared_error
10 import tensorflow as tf
11 import numpy as np

In [2]: 1 df = pd.read_csv("row_size25_vector_size8000.csv")

In [3]: 1 X = df.iloc[:,1:157]
2 xx = df[['accelerometer_reading_x_0','accelerometer_reading_y_0','accelerometer_reading_z_0','gyroscope_reading_phi_0','gyro
3 pt = df[['linear_position_x','linear_position_y','linear_position_z','angular_position_phi','angular_position_theta','angula
4 pt = df[['ph_change','theta_change','psi_change']]
5
6
7 pt = df[['ph_change']]
8 t = df[['theta_change']]
9 pt = df[['psi_change']]
10
11 pt = df[['ph_change']]
12 pt = df[['ph_change']]
13 pt = df[['ph_change']]
14
15
```

In [4]: 1 t.shape

Out[4]: (8000, 1)

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

In [6]: 1 in\_dim

Out[6]: 150

In [7]: 1 out\_dim

Out[7]: 1

```
In [8]: 1 X_train, X_test, t_train, t_test = train_test_split(X, t, test_size=0.2)
```

In [9]: 1 t\_train.shape

Out[9]: (6400, 1)

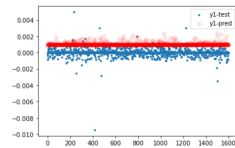
```
In [10]: 1 model = Sequential()
2 model.add(Dense(256, input_dim=in_dim, activation="sigmoid"))
3 model.add(ReLU())
4 model.add(Dropout(.32))
5 model.add(Dense(128, activation="sigmoid"))
6 model.add(ReLU())
7 model.add(LayerNormalization())
8 model.add(Dropout(.25))
9 model.add(Dense(64, activation="sigmoid"))
10 model.add(ReLU())
11 model.add(LayerNormalization())
12 model.add(Dropout(.13))
13 model.add(Dense(32, activation="sigmoid"))
14 model.add(ReLU())
15 model.add(LayerNormalization())
16 model.add(Dense(out_dim, activation="sigmoid"))
17 model.compile(loss="mse", optimizer="sgd")
```

```
In [11]: 1 model.fit(X_train, t_train, epochs=75, batch_size=12, verbose=2)
```

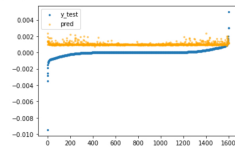
```
Epoch 67/75
534/534 - 1s - loss: 1.2564e-05
Epoch 68/75
534/534 - 1s - loss: 1.3465e-05
Epoch 69/75
534/534 - 1s - loss: 1.2943e-05
Epoch 70/75
534/534 - 1s - loss: 1.2516e-05
Epoch 71/75
534/534 - 1s - loss: 1.4423e-05
Epoch 72/75
534/534 - 1s - loss: 1.1610e-05
Epoch 73/75
534/534 - 1s - loss: 1.4043e-05
Epoch 74/75
534/534 - 1s - loss: 1.2537e-05
Epoch 75/75
534/534 - 1s - loss: 1.3777e-05
Out[11]: <tensorflow.python.keras.callbacks.History at 0x1a75a0610>
```

```
In [12]: 1 ypred = model.predict(X_test)
2 print("y1 MSE: ", mean_squared_error(t_test.iloc[:, 0], ypred[:,0]))
3 #print("\t MSE: ", mean_squared_error(t_test.iloc[:, 1], ypred[:,1]))
4 #print("\t MSE: ", mean_squared_error(t_test.iloc[:, 2], ypred[:,2]))
5
y1 MSE: 1.2629229974267874e-06
```

```
In [13]: 1 x_ax = range(len(X_test))
2
3 plt.scatter(x_ax, t_test.iloc[:, 0], s=6, label="y1-test")
4 plt.scatter(x_ax, ypred[:,0], label="y1-pred", c="red", alpha = 0.1)
5
6 plt.legend()
7 plt.show()
```



```
In [14]: 1 x_ax = range(len(X_test))
2
3 y_test_index = np.argsort(t_test.iloc[:, 0], axis=0).to_numpy()
4
5 f = plt.figure()
6 plt.scatter(x_ax, t_test.iloc[y_test_index], s=6, label="y-test")
7 plt.scatter(x_ax, ypred[y_test_index], s=6, label="pred", c="orange", alpha=0.5)
8 plt.plot(t_test.iloc[y_test_index[0]].to_numpy()[0])
9 plt.legend()
10 plt.show()
11
12 f.savefig("foo.pdf", bbox_inches='tight')
```



In [ ]: 1

In [ ]: 1