ansi_regression

March 30, 2018

1 ANSI Application analysis

```
In [1]: import numpy
        import pandas
        import matplotlib.pyplot as plotter
        from scipy.stats import pearsonr
        from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
In [2]: def view_boxplot(df):
            %matplotlib
            df.boxplot()
            plotter.show()
```

1.1 CPU data

```
In [3]: cpu_df = pandas.read_csv('data/ansi_fake_data/ansi_fake_data_cpu.csv', index_col='Time
```

```
In [4]: #cpu_df.columns
```

```
In [5]: #view_boxplot(cpu_df)
```

1.2 Network TX

```
In [6]: txnet_df = pandas.read_csv('data/ansi_fake_data/ansi_fake_data_network_tx.csv', index_
```

```
In [7]: #txnet_df.columns
```

```
In [8]: #view_boxplot(txnet_df)
```

1.3 Network RX

```
In [9]: rxnet_df = pandas.read_csv('data/ansi_fake_data/ansi_fake_data_network_rx.csv', index_
```

```
In [10]: #rxnet_df.columns
```

```
In [11]: rxnet_df = rxnet_df.clip(lower=0, upper=15000)
         #view_boxplot(rxnet_df)
```

1.4 Disk IO data

```
In [12]: disk_df = pandas.read_csv('data/ansi_fake_data/ansi_fake_data_disk_io.csv', index_col=

In [13]: #disk_df.columns

In [14]: disk_df = disk_df.clip(lower=0, upper=4000)
         #view_boxplot(disk_df)
```

1.5 Context switching

```
In [15]: context_df = pandas.read_csv('data/ansi_fake_data/ansi_fake_data_context.csv', index_c

In [16]: #context_df.columns

In [17]: context_df = context_df.clip(lower=0, upper=5000)
         #view_boxplot(context_df)
```

1.6 Seperate into proper dataframes for each node

```
In [18]: dframes = [cpu_df, txnet_df, rxnet_df, context_df, disk_df]
         node = {}

         for i in range(1,5):
             frames = []

             for dframe in dframes:
                 columns = list(filter(lambda x: f'bb{i}l' in x, dframe.columns))
                 frames.append(dframe[columns])

             node[i] = pandas.concat(frames, join='inner', axis=1).fillna(0)[:38200]

In [19]: for i in range(1,5):
             print(node[i].shape)

         print(node[1].columns)
```

```
(38200, 29)
(38200, 29)
(38200, 29)
(38200, 29)
Index(['cpu_value host bb1localdomain type_instance idle',
       'cpu_value host bb1localdomain type_instance interrupt',
       'cpu_value host bb1localdomain type_instance nice',
       'cpu_value host bb1localdomain type_instance softirq',
       'cpu_value host bb1localdomain type_instance steal',
       'cpu_value host bb1localdomain type_instance system',
       'cpu_value host bb1localdomain type_instance user',
       'cpu_value host bb1localdomain type_instance wait',
```

```
       'interface_tx host bb1localdomain instance lo type if_dropped',
       'interface_tx host bb1localdomain instance lo type if_errors',
       'interface_tx host bb1localdomain instance lo type if_octets',
       'interface_tx host bb1localdomain instance lo type if_packets',
       'interface_tx host bb1localdomain instance wlan0 type if_dropped',
       'interface_tx host bb1localdomain instance wlan0 type if_errors',
       'interface_tx host bb1localdomain instance wlan0 type if_octets',
       'interface_tx host bb1localdomain instance wlan0 type if_packets',
       'interface_rx host bb1localdomain instance lo type if_dropped',
       'interface_rx host bb1localdomain instance lo type if_errors',
       'interface_rx host bb1localdomain instance lo type if_octets',
       'interface_rx host bb1localdomain instance lo type if_packets',
       'interface_rx host bb1localdomain instance wlan0 type if_dropped',
       'interface_rx host bb1localdomain instance wlan0 type if_errors',
       'interface_rx host bb1localdomain instance wlan0 type if_octets',
       'interface_rx host bb1localdomain instance wlan0 type if_packets',
       'contextswitch_value host bb1localdomain type contextswitch',
       'disk_io_time host bb1localdomain instance mmcblk1 type disk_io_time',
       'disk_io_time host bb1localdomain instance mmcblk1boot0 type disk_io_time',
       'disk_io_time host bb1localdomain instance mmcblk1boot1 type disk_io_time',
       'disk_io_time host bb1localdomain instance mmcblk1p1 type disk_io_time'],
      dtype='object')
```

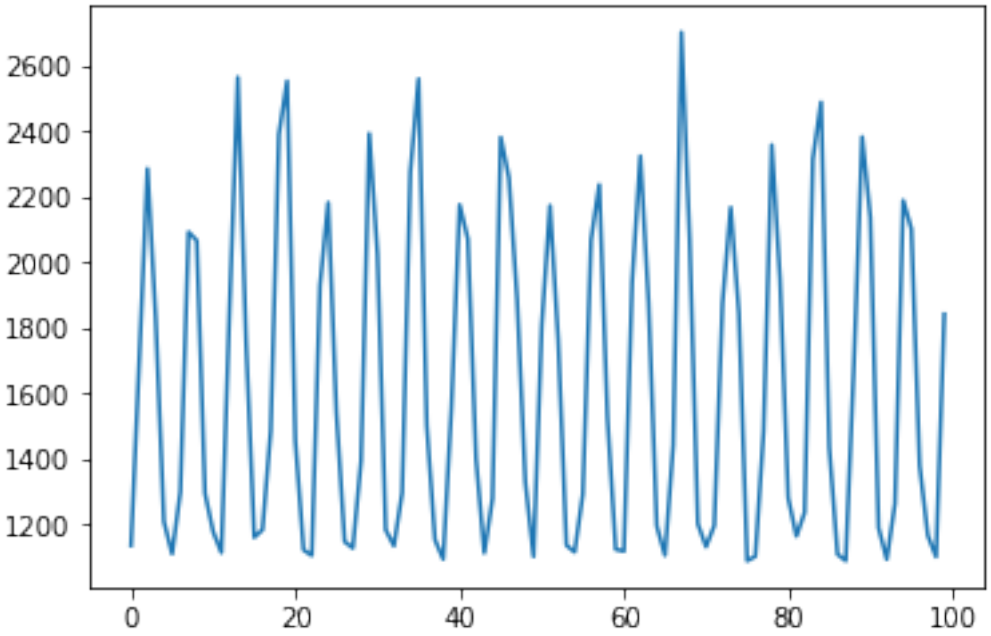
1.7 Get data

```
In [20]: data_matrices = []

         for i in range(1,5):
             data_matrices.append(node[i].as_matrix())

         data = numpy.vstack(data_matrices)
```

```
In [21]: data.shape
```

```
Out[21]: (152800, 29)
```

```
In [22]: tdata = data[:,24]
         plotter.plot(tdata.T[:100])
         plotter.show()
         print(data.shape)
```




```
(152800, 29)
```

```
In [23]: #data = data[:,24]
```

1.8 Prepare scaler

```
In [24]: from sklearn.preprocessing import MinMaxScaler
         from sklearn.preprocessing import StandardScaler
         from sklearn.preprocessing import RobustScaler
         scaler = MinMaxScaler()
```

```
In [25]: scaler.fit(data)
         del data
```

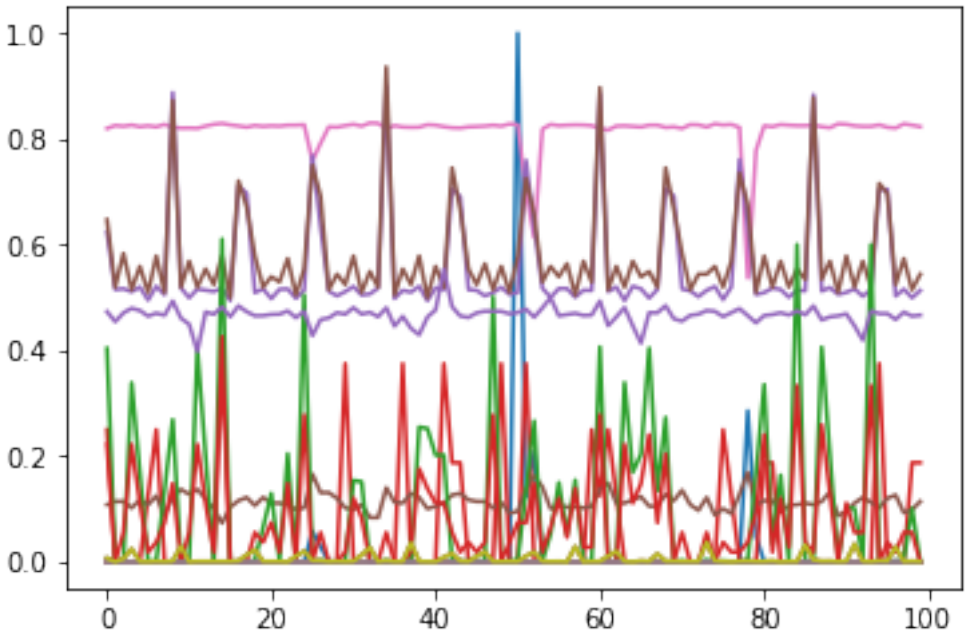
1.9 Correrlation measurement

---

2 Prediction

In [26]:
```
for i in range(len(data_matrices)):

    transformed = scaler.transform(data_matrices[i])
    data_matrices[i] = transformed

X = numpy.stack(data_matrices[:-1], axis=1)
test_X = numpy.array([data_matrices[3]])
test_X[0,50,0] = 1.0
#x_val = numpy.array([data_matrices[2]])
```

In [27]:
```
plotter.plot(test_X.squeeze()[:100])
plotter.show()
```



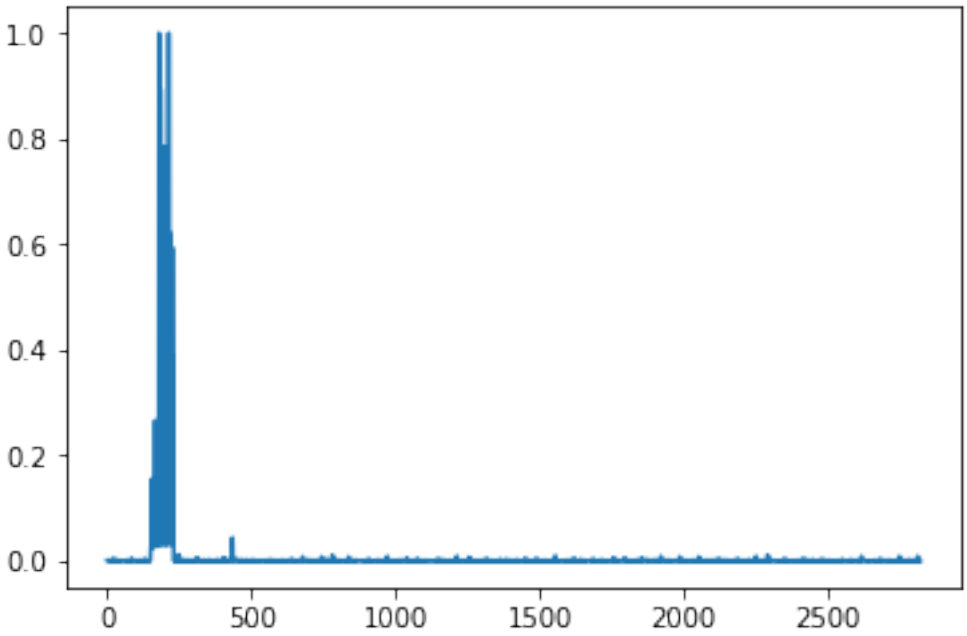
In [28]:
```
print(X.shape)
LEN = X.shape[0]
SPLIT = int(0.9*LEN)

train_X = X[:SPLIT,:,:]
val_X = X[SPLIT:SPLIT+1000,:,:]
test_X = X[SPLIT+1000:,:,:]
```

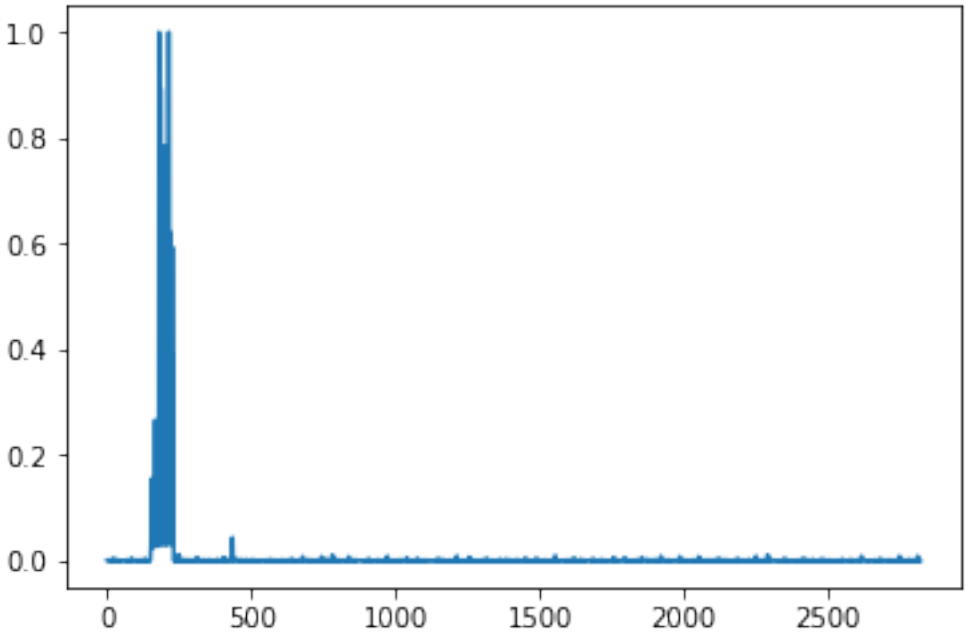
```
(38200, 3, 29)
```

In [29]:
```
X = train_X
X = numpy.transpose(X, (1, 0, 2))
#X = X.reshape((-1,382,29))
val_X = numpy.transpose(val_X, (1, 0, 2))
test_X = numpy.transpose(test_X, (1, 0, 2))
#val_X = val_X.reshape((-1,382,29))
print(X.shape)
print(val_X.shape)
```

```
(3, 34380, 29)
(3, 1000, 29)
```

In [30]:
```
plotter.plot(test_X[0][:,25])
plotter.show()
#test_X[0][:,25] = 0.0
#test_X[0][:,28] = 0.0
plotter.plot(test_X[0][:,25])
plotter.show()
```






```
In [31]: def flat_generator(X, tsteps = 5, ravel=1):
             i = 0

             while True:
                 batch_X = X[:,i:i+tsteps,:]
                 batch_y = X[:,i+tsteps,:]

                 if ravel:
                     batch_X = batch_X.reshape((batch_X.shape[0], -1))
                 #print(batch_X.shape)
                 #print(batch_y.shape)

                 yield batch_X, batch_y

                 i += 1
                 if i > (X.shape[1] - tsteps - 1):
                     i = 0
                     continue
```

2.1 Flat models

```
In [32]: from keras.models import Model
         from keras.layers import Dense, Input, Dropout, GRU
         from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

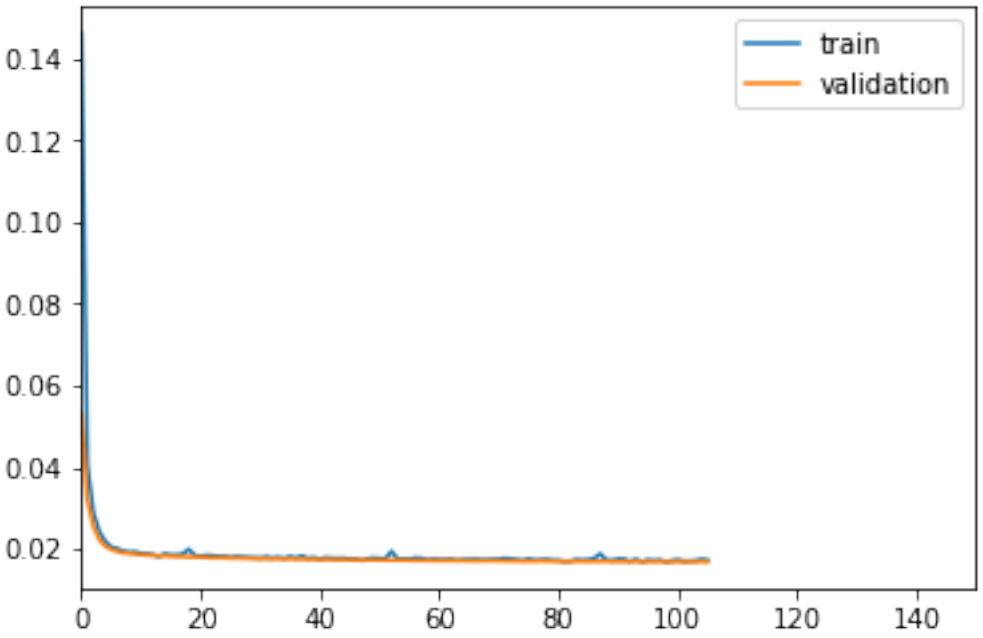
```
In [33]: def train(model, tgen, vgen, name="none"):
             estopper = EarlyStopping(patience=15, min_delta=0.0001)
             history = model.fit_generator(tgen, steps_per_epoch=1000, epochs=10000, callbacks=
             plotter.plot(history.history['loss'],label='train')
             plotter.plot(history.history['val_loss'],label='validation')
             plotter.legend()
             plotter.xlim(0,150)
             plotter.show()
             plotter.savefig(f"{name}_train.png")
             print(history.history['loss'][-1])

In [34]: def test(model, dataset=test_X[0], ravel=1, write=0, name="none"):
             test_gen = flat_generator(numpy.array([dataset]), TIMESTEPS,0)
             error = []
             targets = []
             preds = []
             for i in range(2000):
                 _input,target = next(test_gen)

                 if i != 0:
                     #print(_input.shape)
                     _input = _input.squeeze()[1:,:]
                     #print(_input.shape)
                     _input = numpy.append(pred,_input, axis=0)[numpy.newaxis,:,:]
                     #print(_input.shape)

                 targets.append(target.squeeze())
                 if ravel:
                     _input = _input.ravel()[:,numpy.newaxis].T

                 pred = model.predict(_input)
                 #print(target.shape)
                 #print(pred.shape)
                 preds.append(pred.squeeze())
                 error.append(mean_absolute_error(y_pred=pred, y_true=target))

             targets = numpy.vstack(targets)
             preds = numpy.vstack(preds)

             plotter.plot(error, 'g-', alpha=0.5)
             plotter.ylim(0,0.2)
             plotter.xlabel("time")
             plotter.ylabel("Error")
             plotter.show()
             plotter.savefig(f"{name}_testloss.png")
             error = numpy.array(error)
             print(numpy.mean(error))
             plotter.boxplot(error)
```

```
plotter.ylim(0,0.2)
plotter.xlabel("time")
plotter.ylabel("Error")
plotter.show()
plotter.savefig(f"{name}_boxplot.png")
if write:
    numpy.savetxt('loss.txt', numpy.array(error))
#print(error)
```

2.1.1 Linear Regression

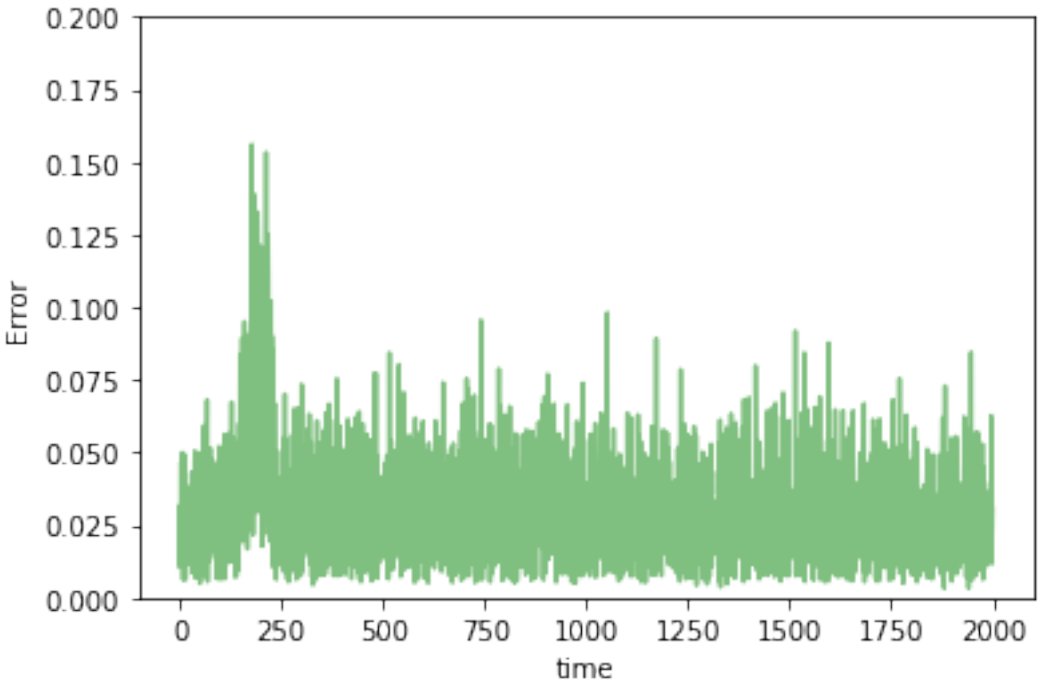
2 steps

```
In [35]: TIMESTEPS = 2
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [36]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         output = Dense(DIM, activation='sigmoid')(input_layer)

In [37]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [38]: train(model, tgen, vgen, name="lin2")
```

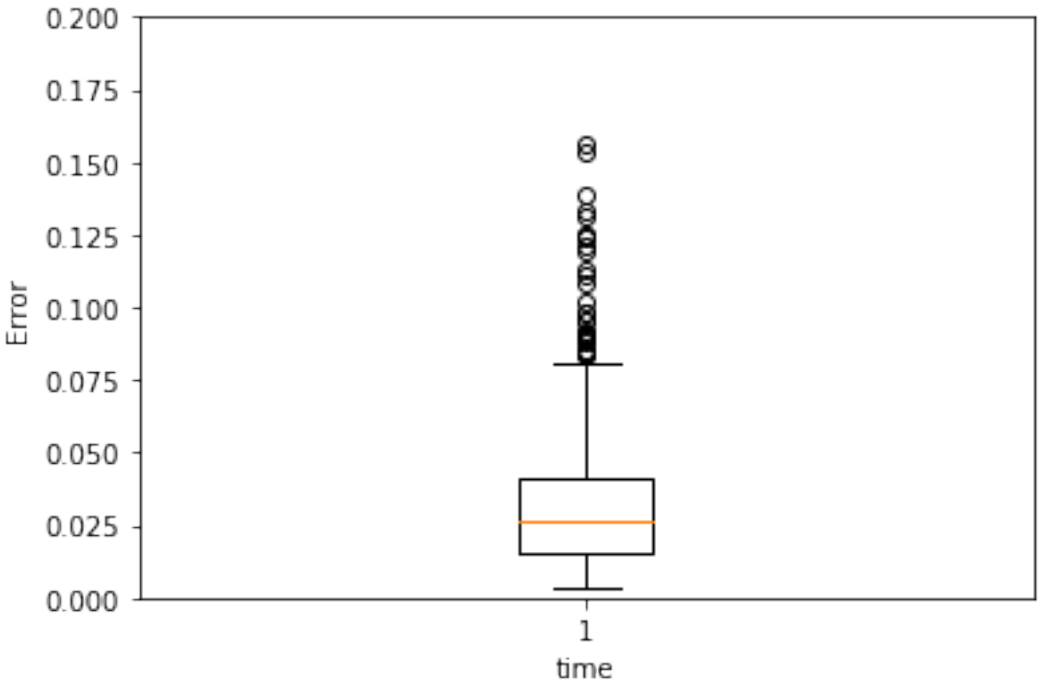


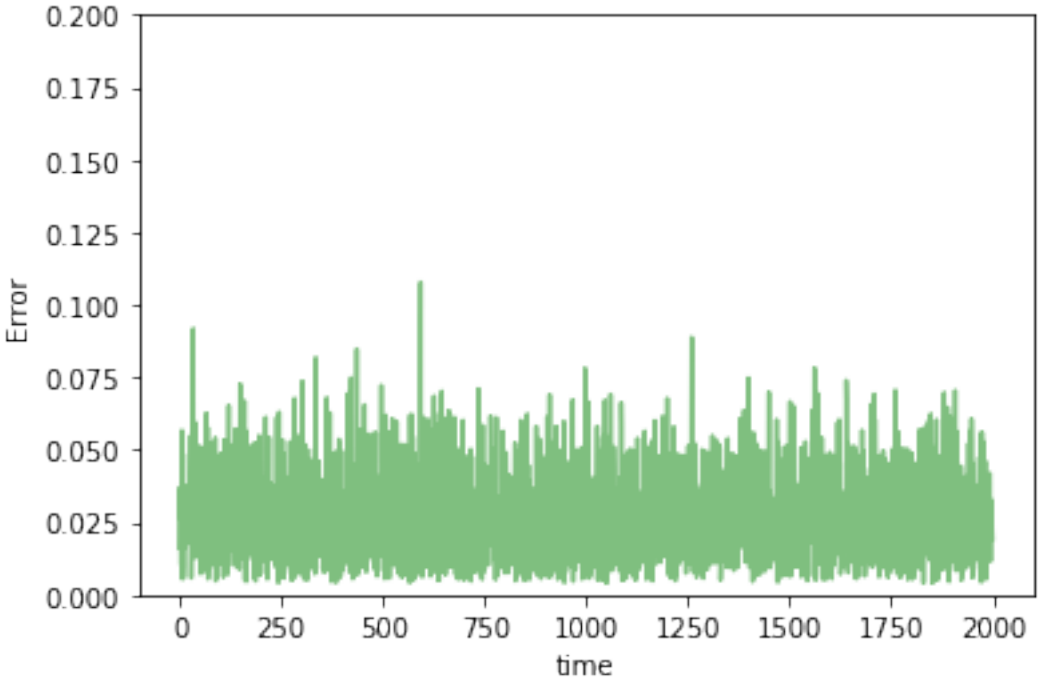
```
0.0172995862553
```

```
In [39]: test(model, test_X[0], name="lin2ano")
         test(model, test_X[2], name="lin2norm")
```

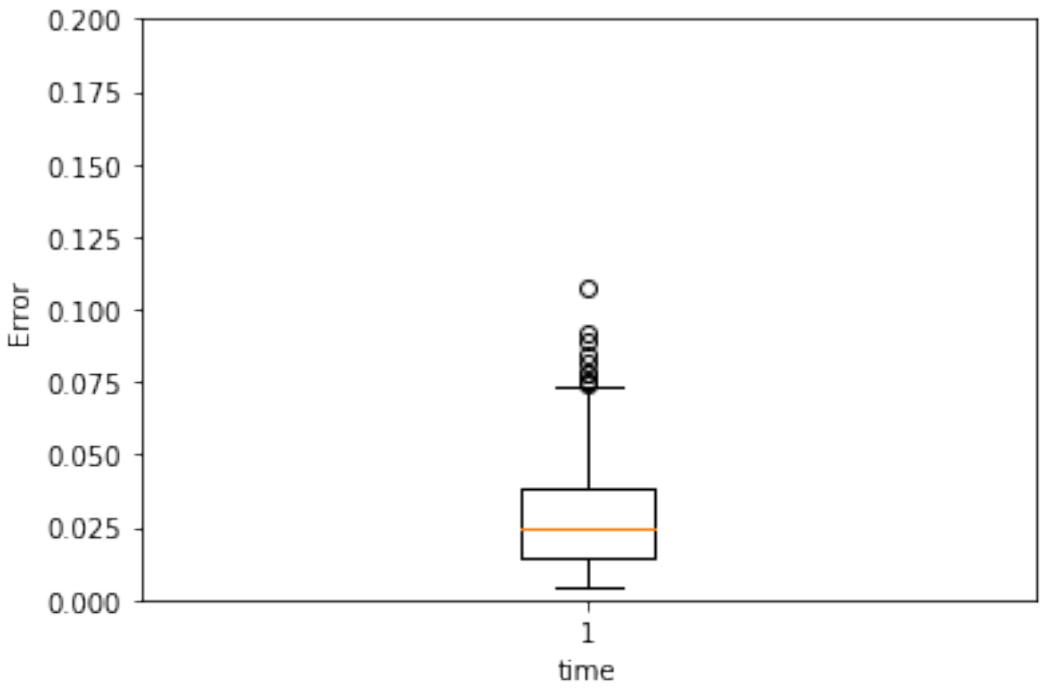



```
0.0300305566464
```



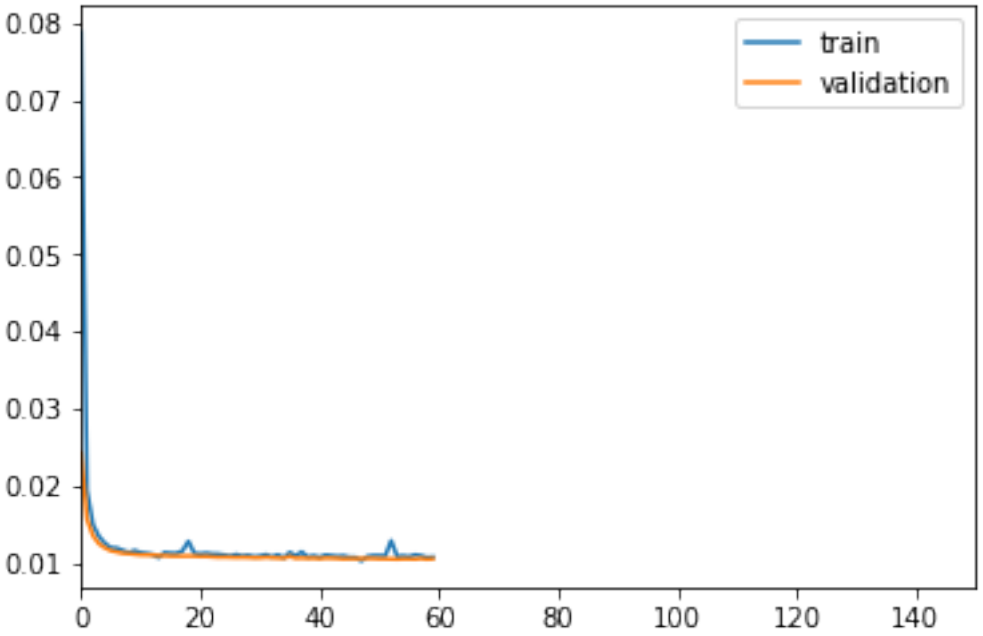



```
0.0273094215342
```



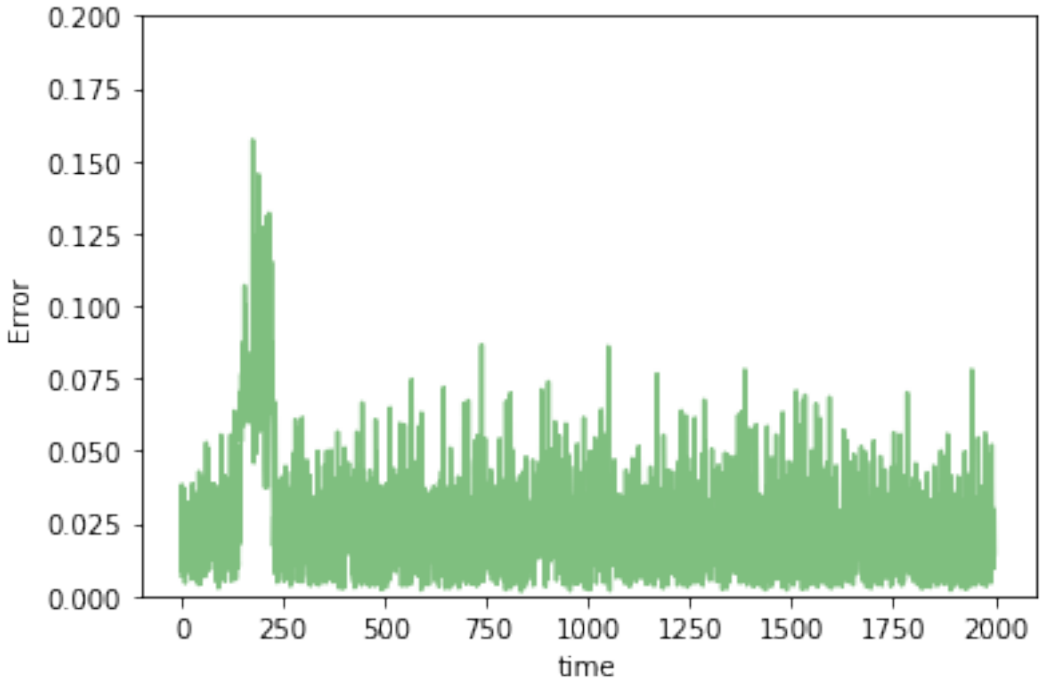
5 steps

```
In [40]: TIMESTEPS = 5
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [41]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         output = Dense(DIM, activation='sigmoid')(input_layer)

In [42]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [43]: train(model, tgen, vgen, name="lin5")
```

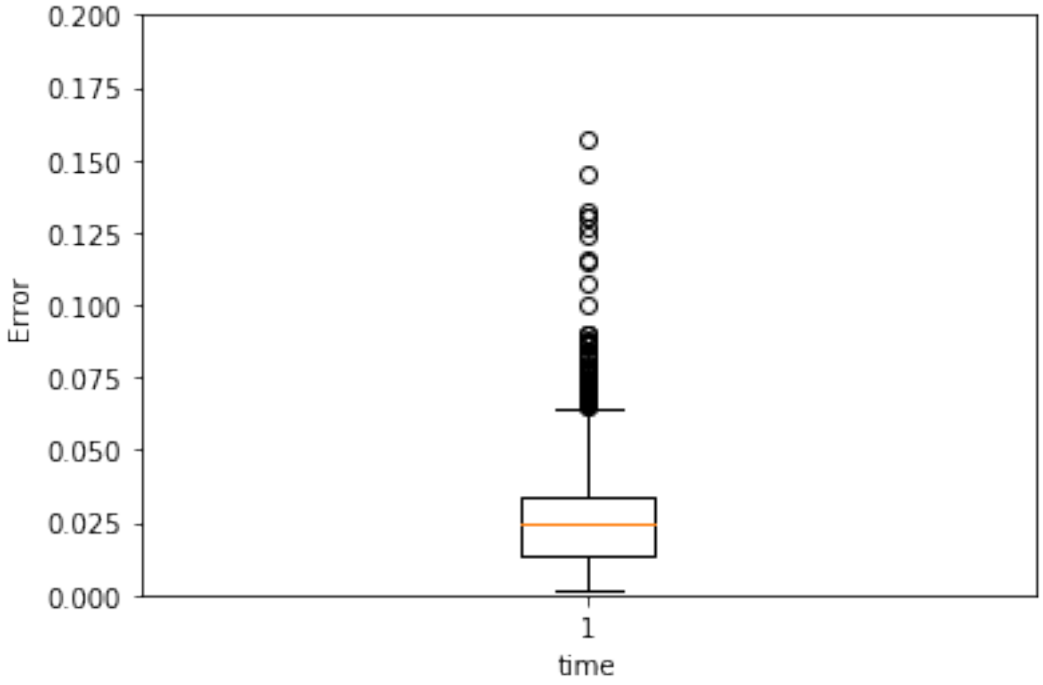


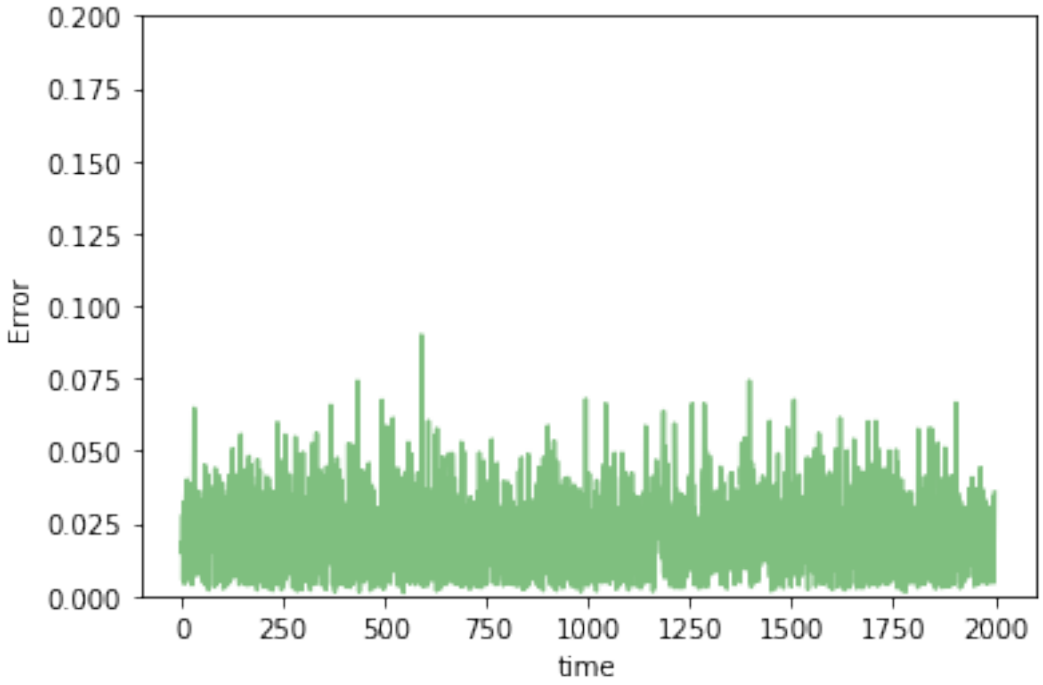

```
0.0107861282913
```

```
In [44]: test(model, test_X[0], name="lin5ano")
         test(model, test_X[2], name="lin5norm")
```

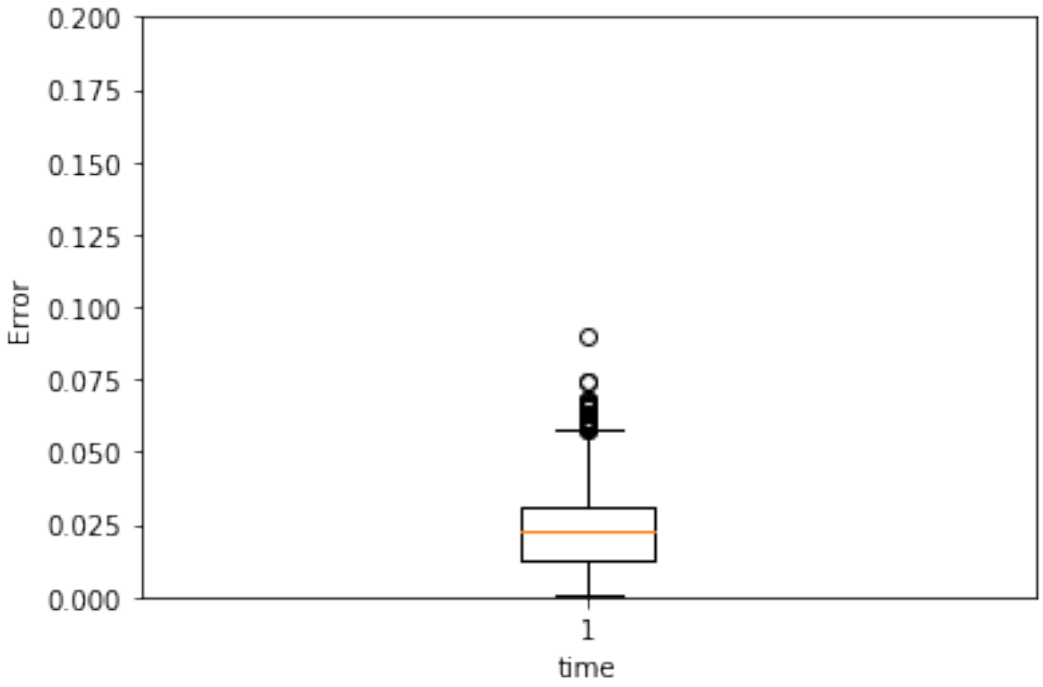


```
0.0262182959487
```



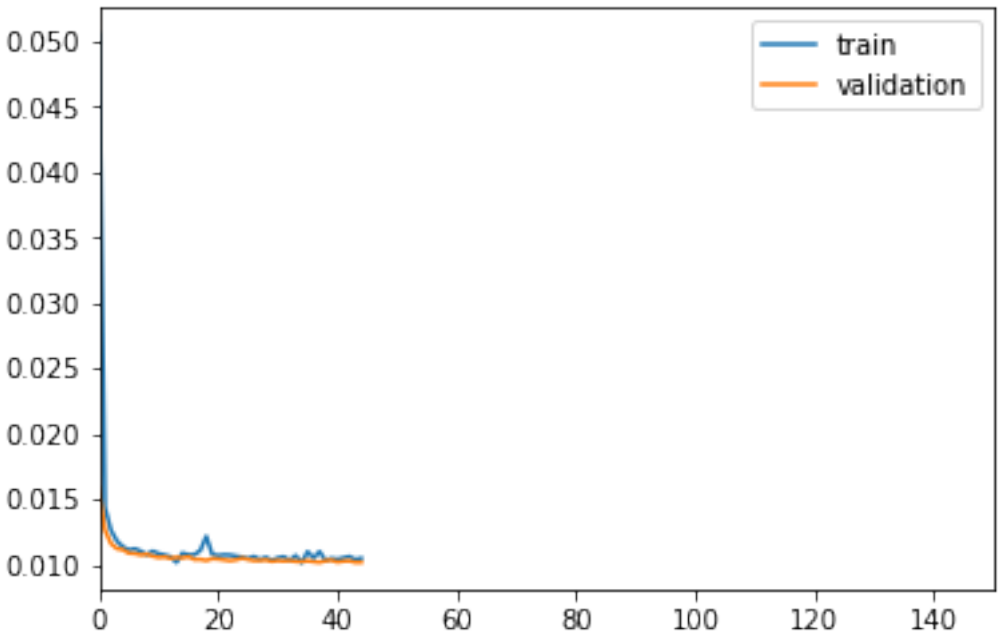


```
0.0226698785813
```



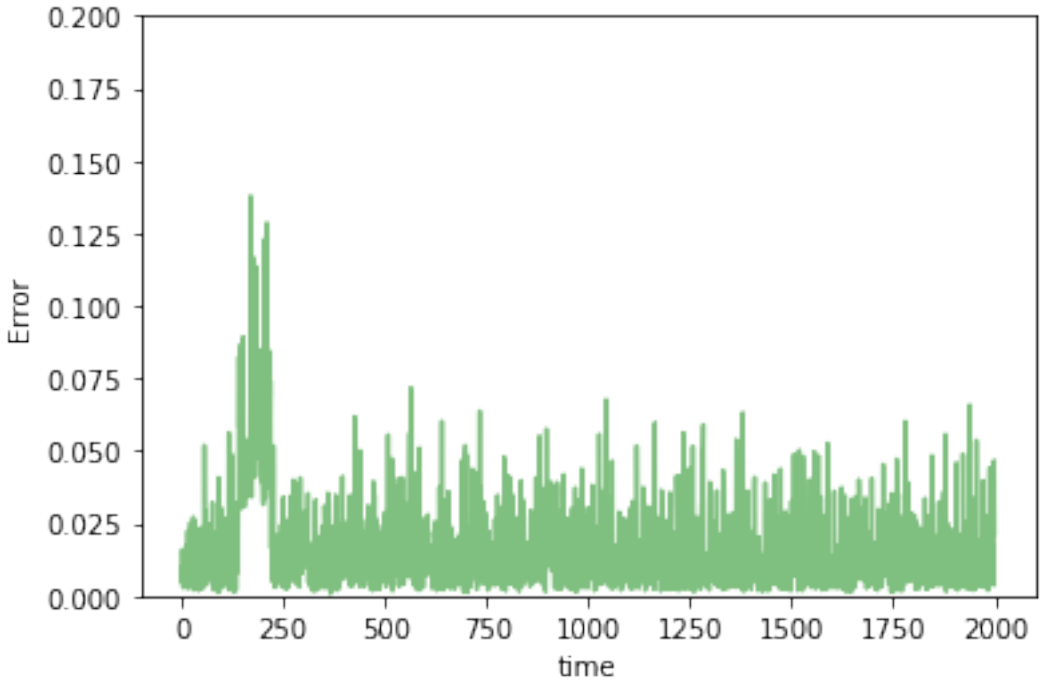
10 steps

```
In [45]: TIMESTEPS = 10
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [46]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         output = Dense(DIM, activation='sigmoid')(input_layer)

In [47]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [48]: train(model, tgen, vgen, name="lin10")
```

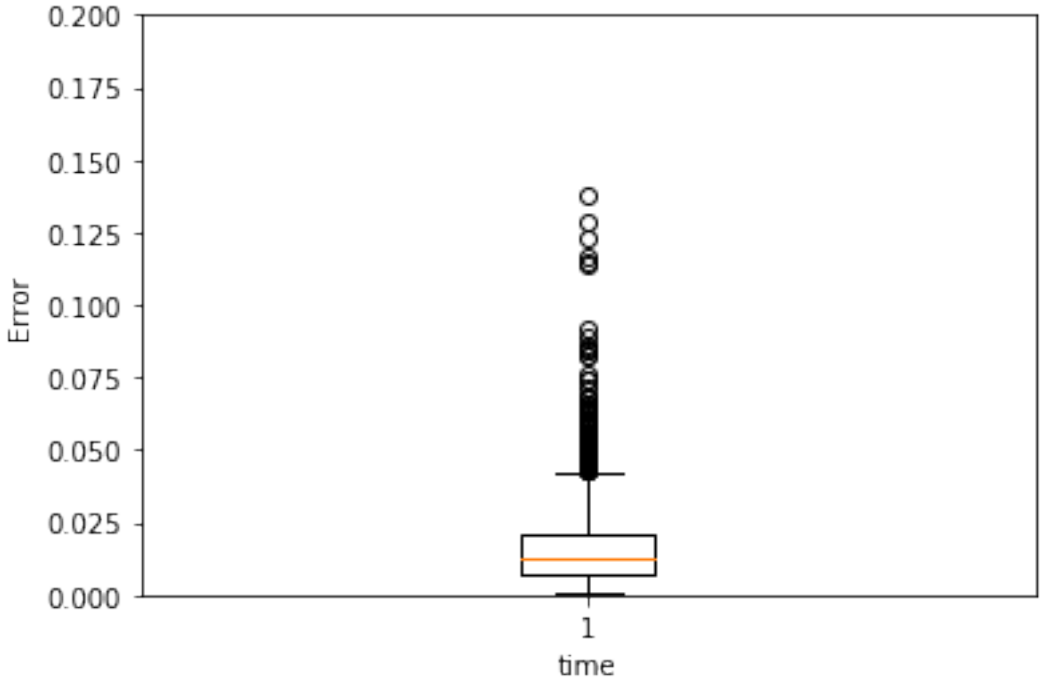


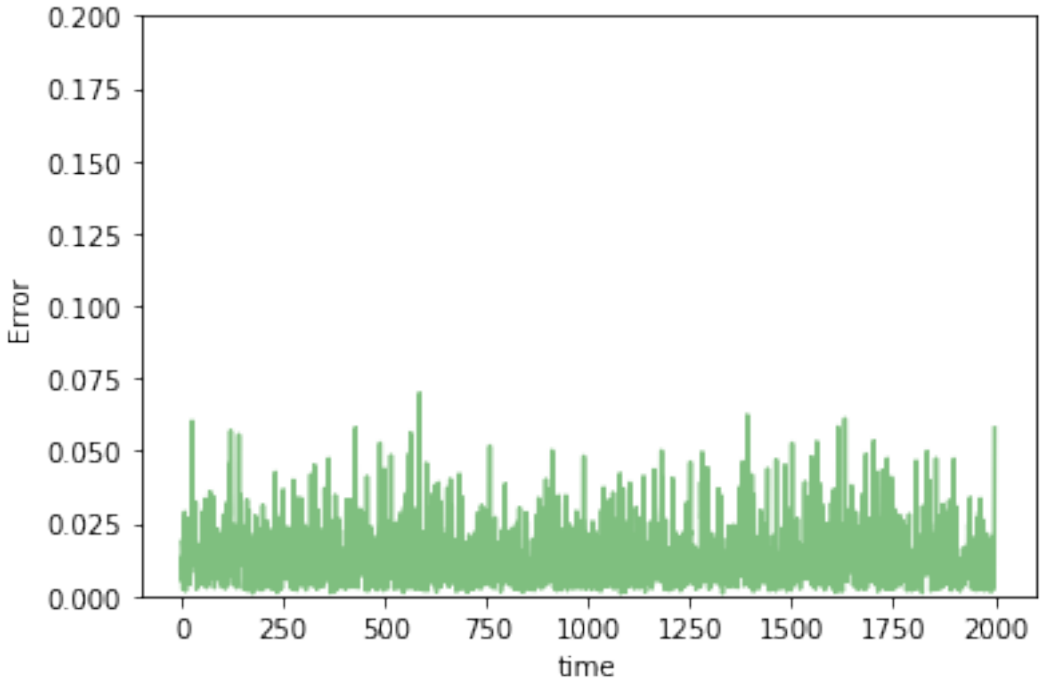

```
0.0104746631576
```

```
In [49]: test(model, test_X[0], name="lin10ano")
         test(model, test_X[2], name="lin10norm")
```

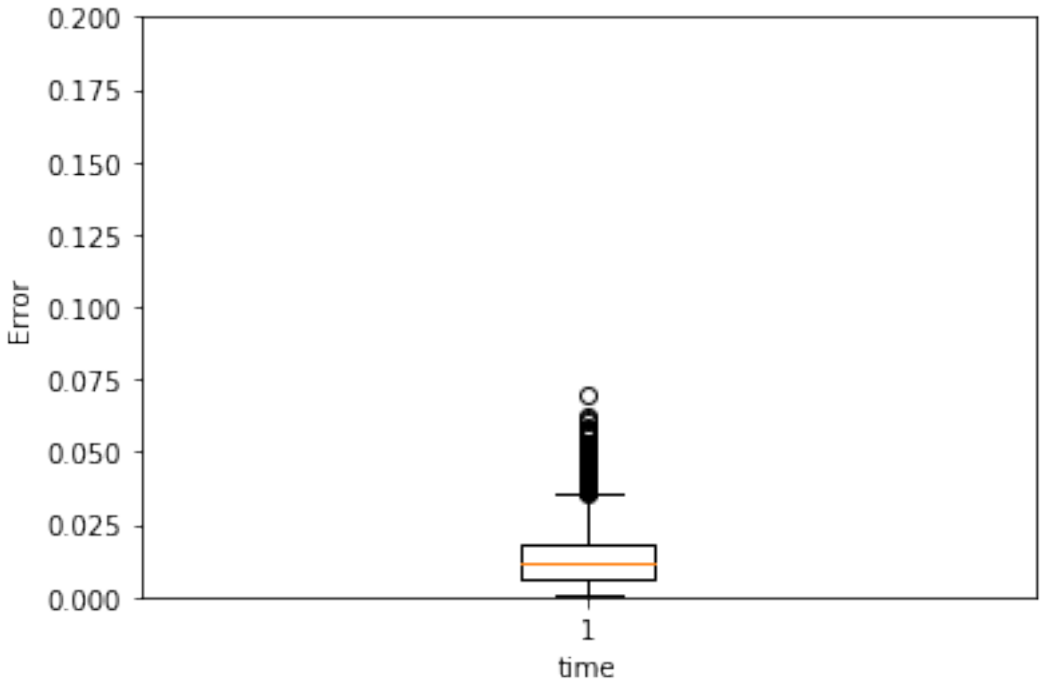


```
0.016672326146
```



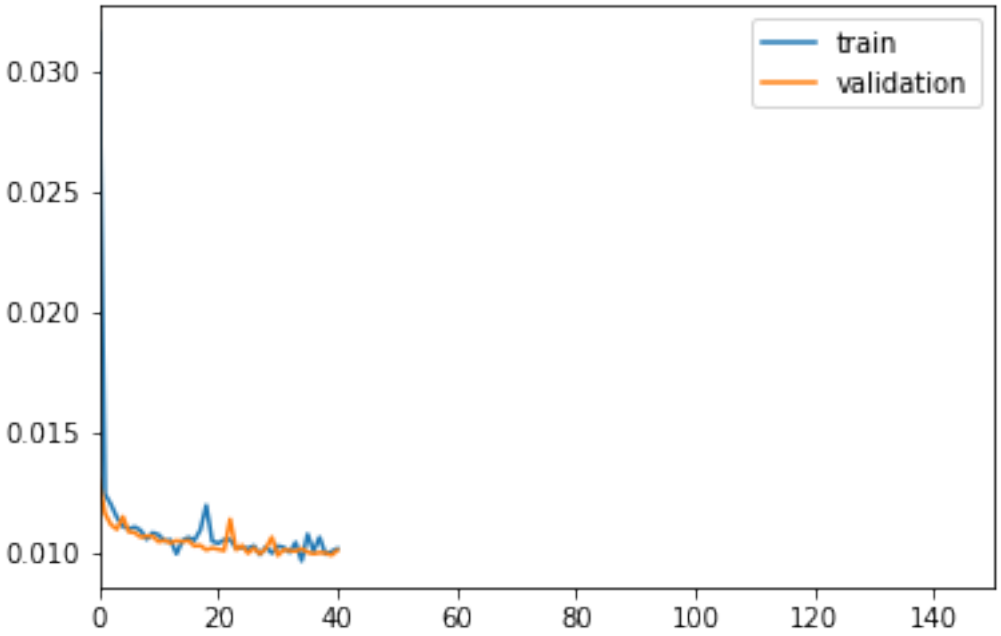


```
0.0139884758787
```



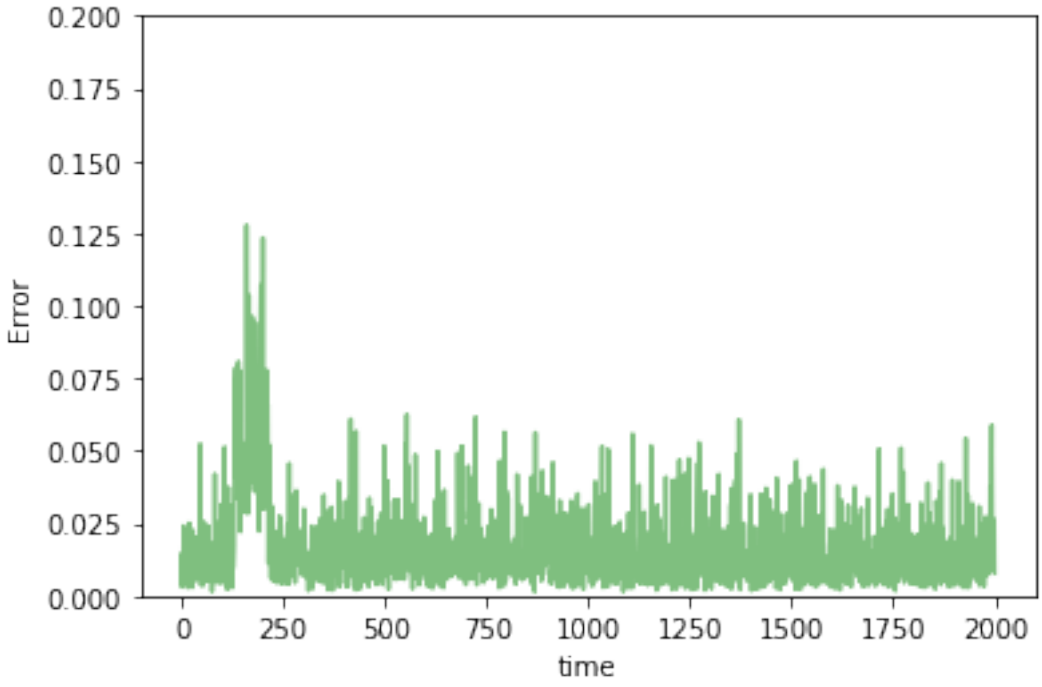
20 steps

```
In [50]: TIMESTEPS = 20
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [51]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         output = Dense(DIM, activation='sigmoid')(input_layer)

In [52]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [53]: train(model, tgen, vgen, name="lin20")
```

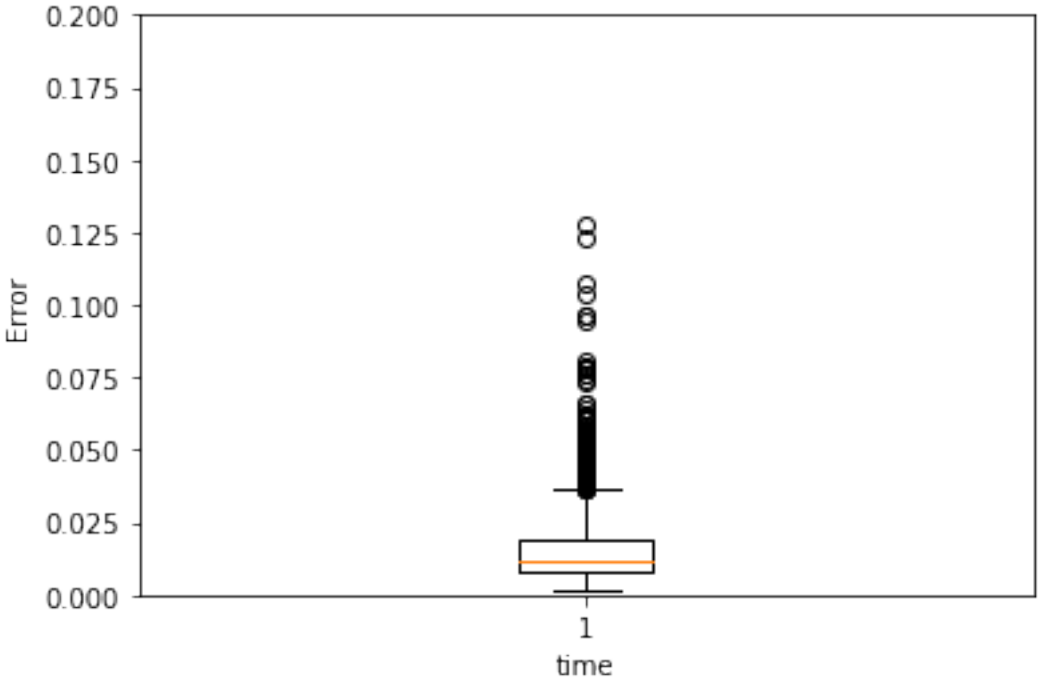


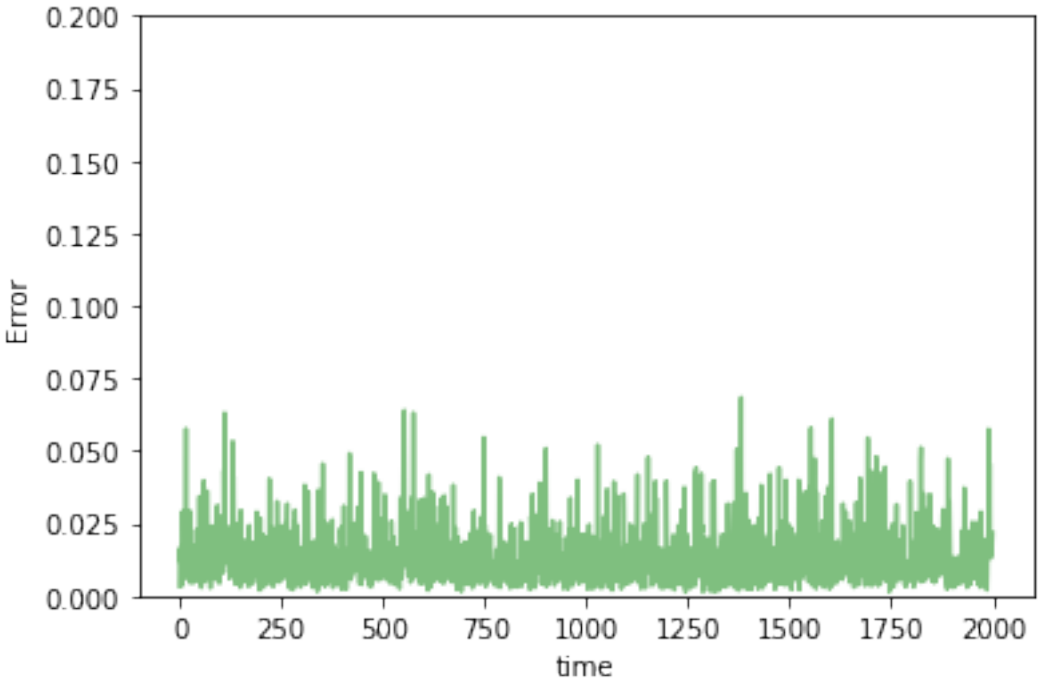

```
0.0101977151516
```

```
In [54]: test(model, test_X[0], name="lin20ano")
         test(model, test_X[2], name="lin20norm")
```

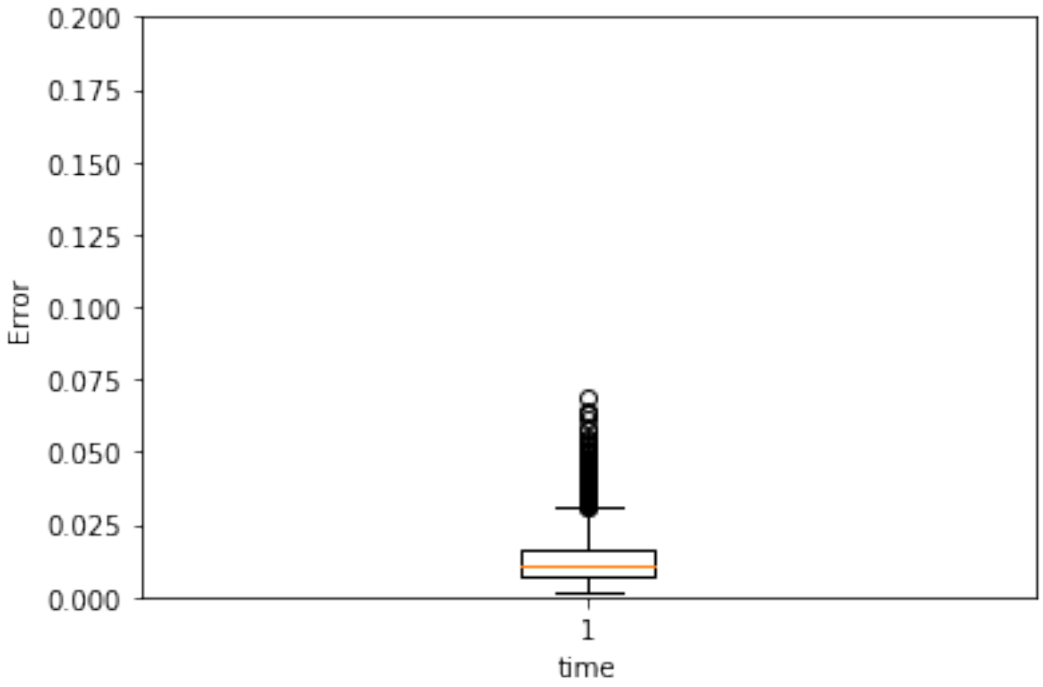


```
0.0157760789735
```



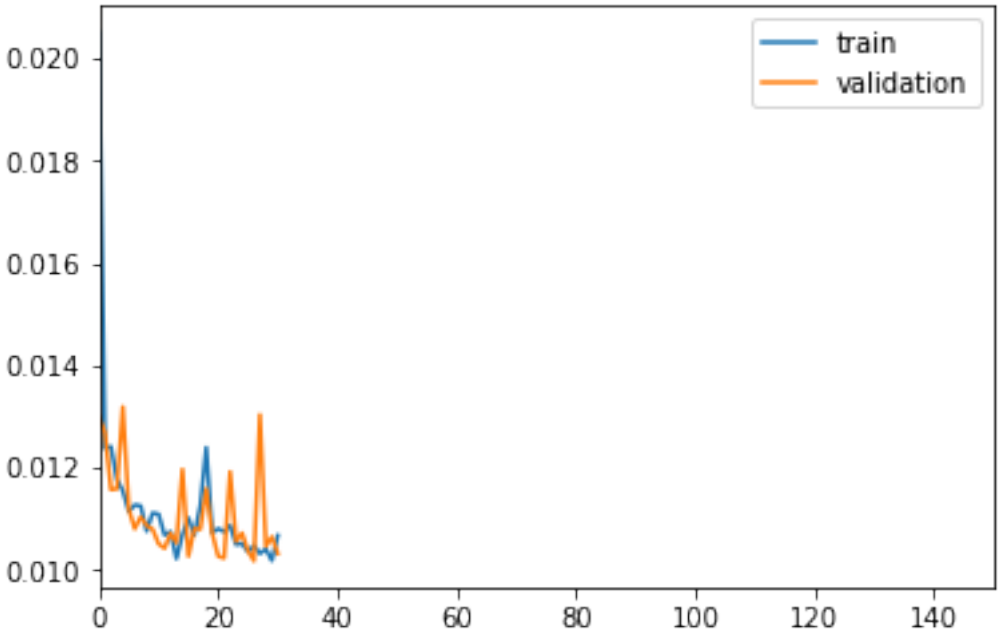


```
0.0132359593233
```



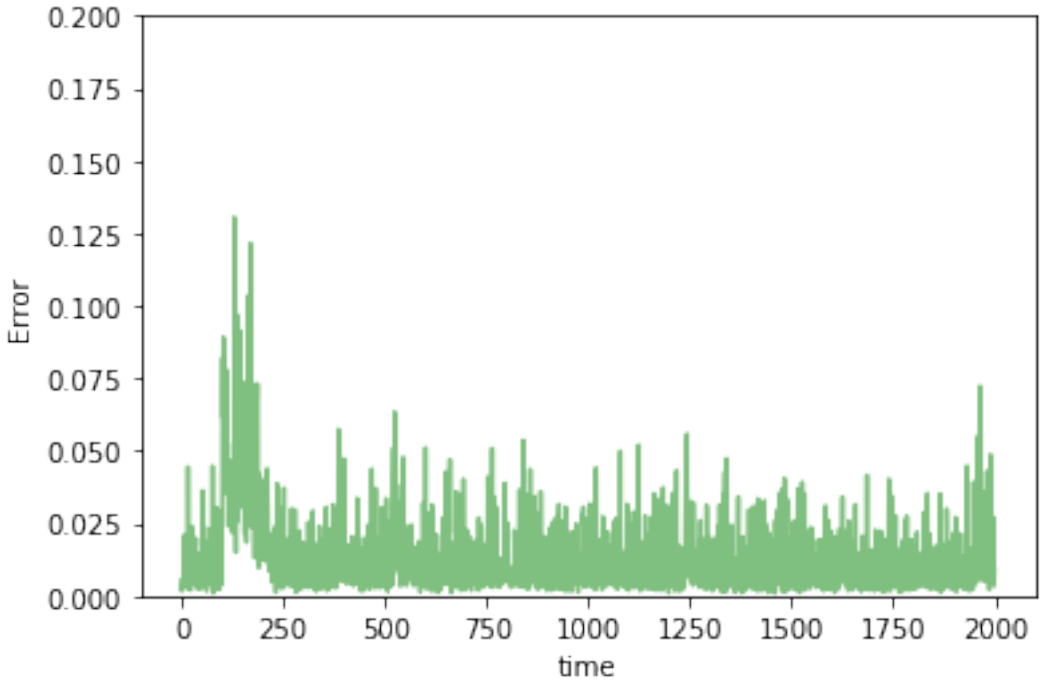
50 steps

```
In [55]: TIMESTEPS = 50
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [56]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         output = Dense(DIM, activation='sigmoid')(input_layer)

In [57]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [58]: train(model, tgen, vgen, name="lin50")
```

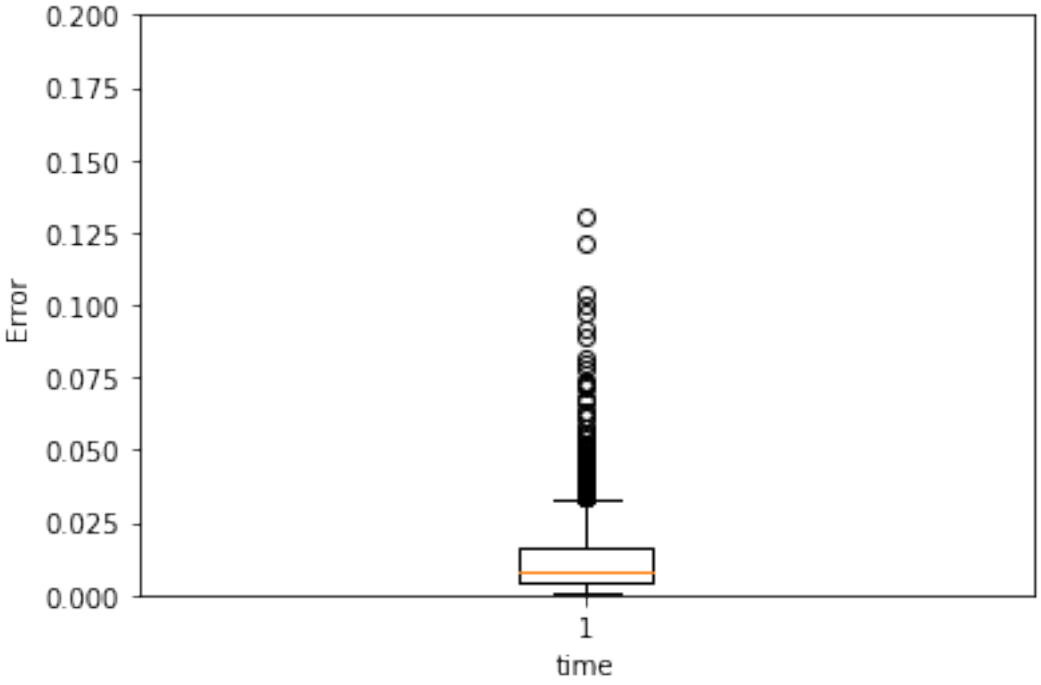


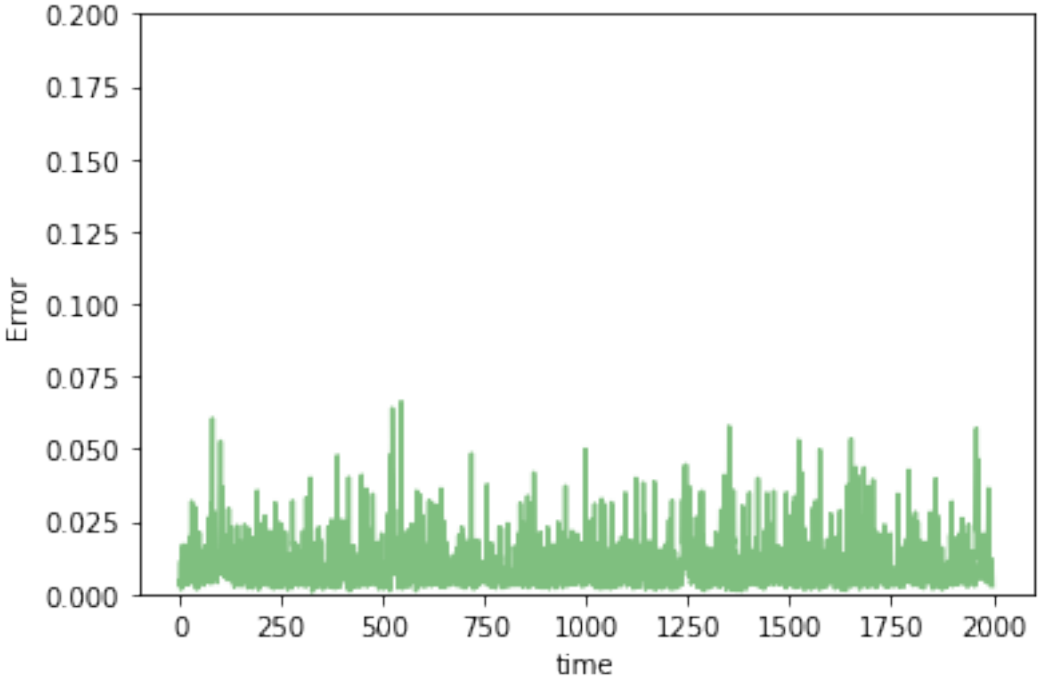

```
0.0106756427106
```

```
In [59]: test(model, test_X[0], name="lin50ano")
         test(model, test_X[2], name="lin50norm")
```

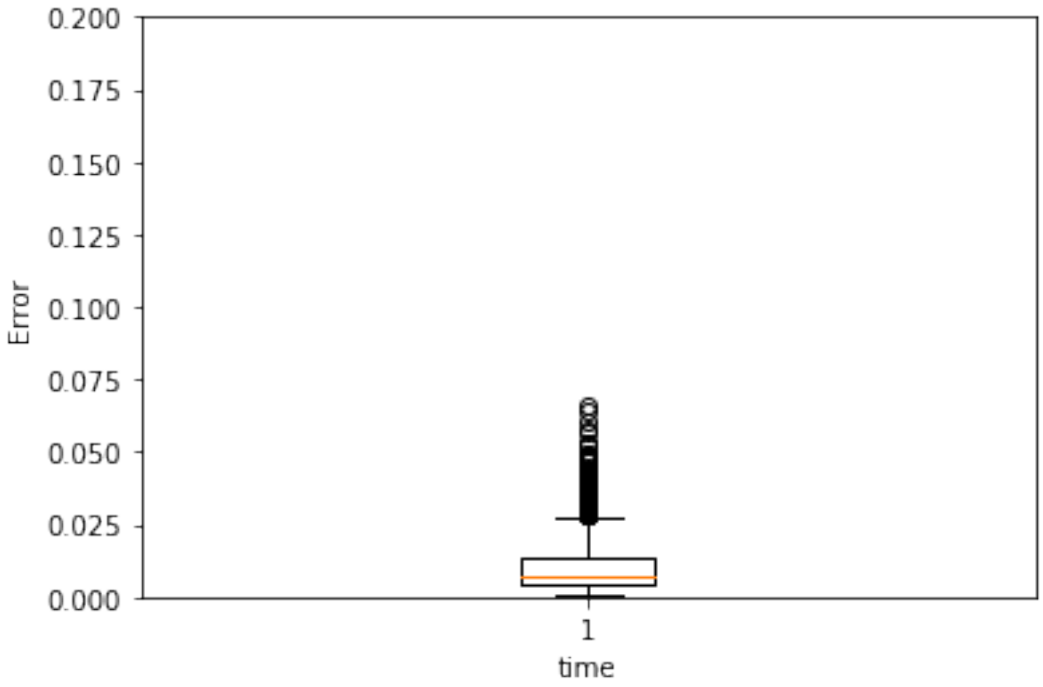


```
0.0125625428044
```





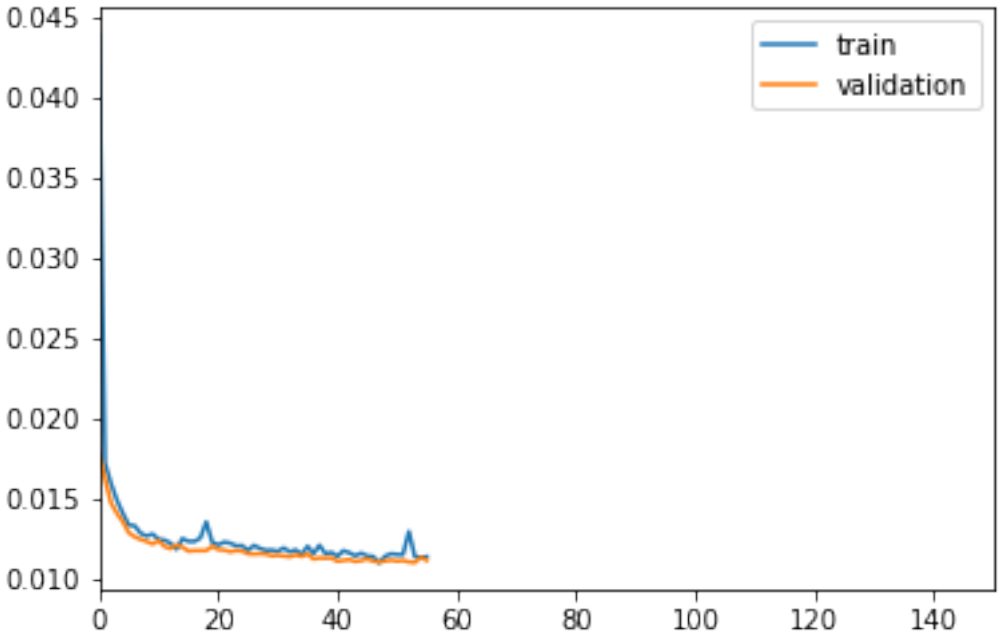
```
0.00995252898372
```



2.1.2 NN with 1 hidden layer

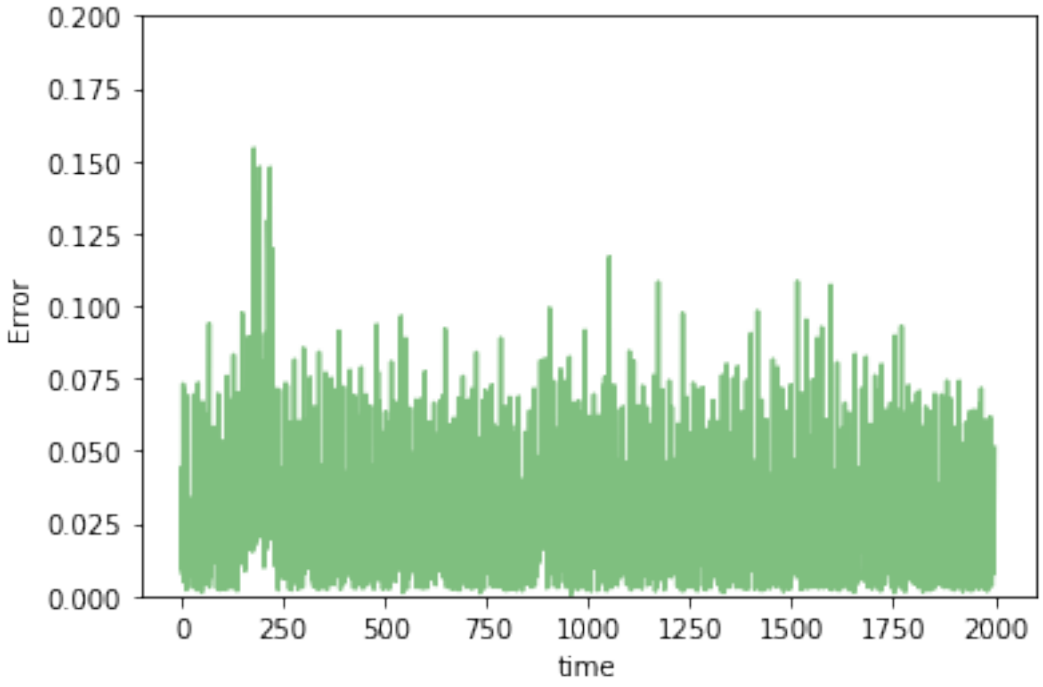
2 steps

```
In [60]: TIMESTEPS = 2
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [61]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(100, activation='relu')(input_layer)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [62]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [63]: train(model, tgen, vgen, name="nn1_2")
```

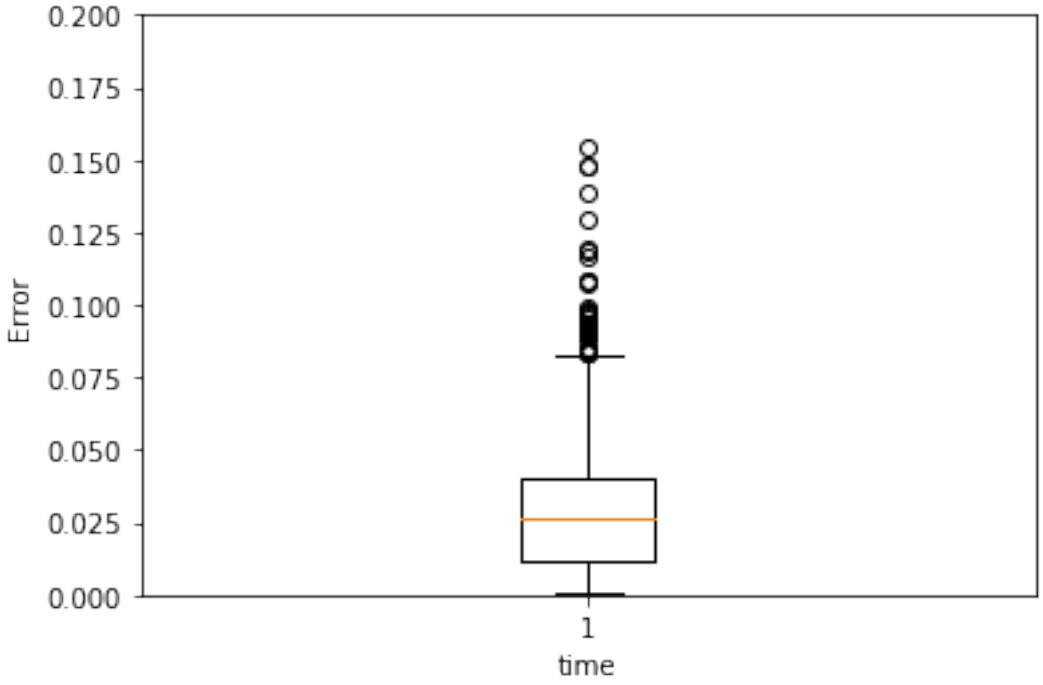


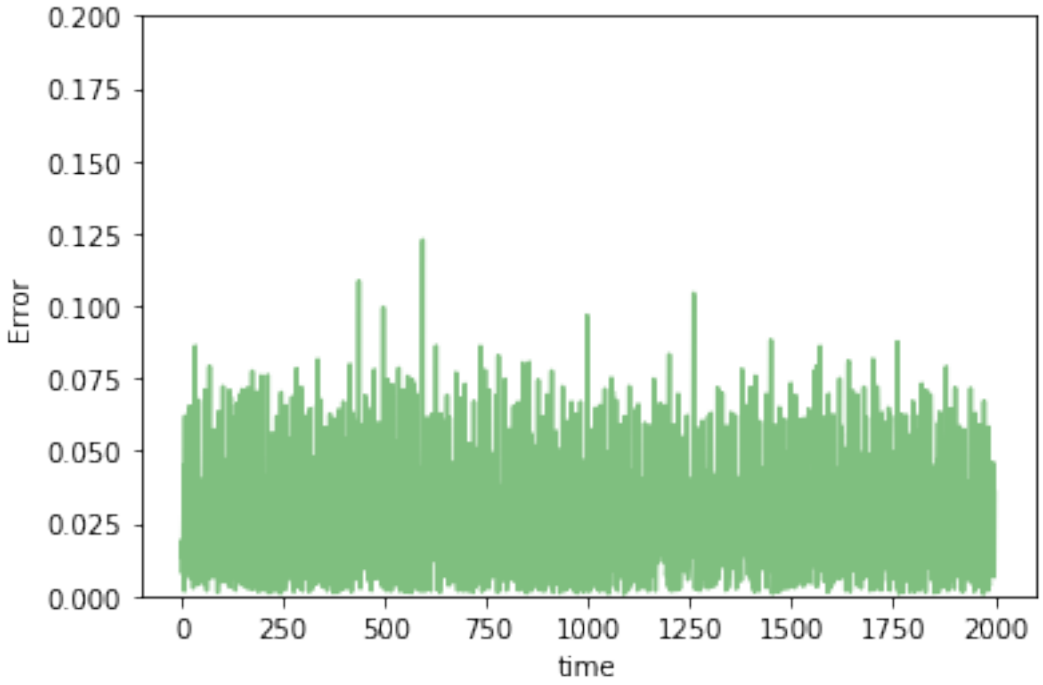

```
0.011322047986
```

```
In [64]: test(model, test_X[0], name="nn1_2ano")
         test(model, test_X[2], name="nn1_2norm")
```

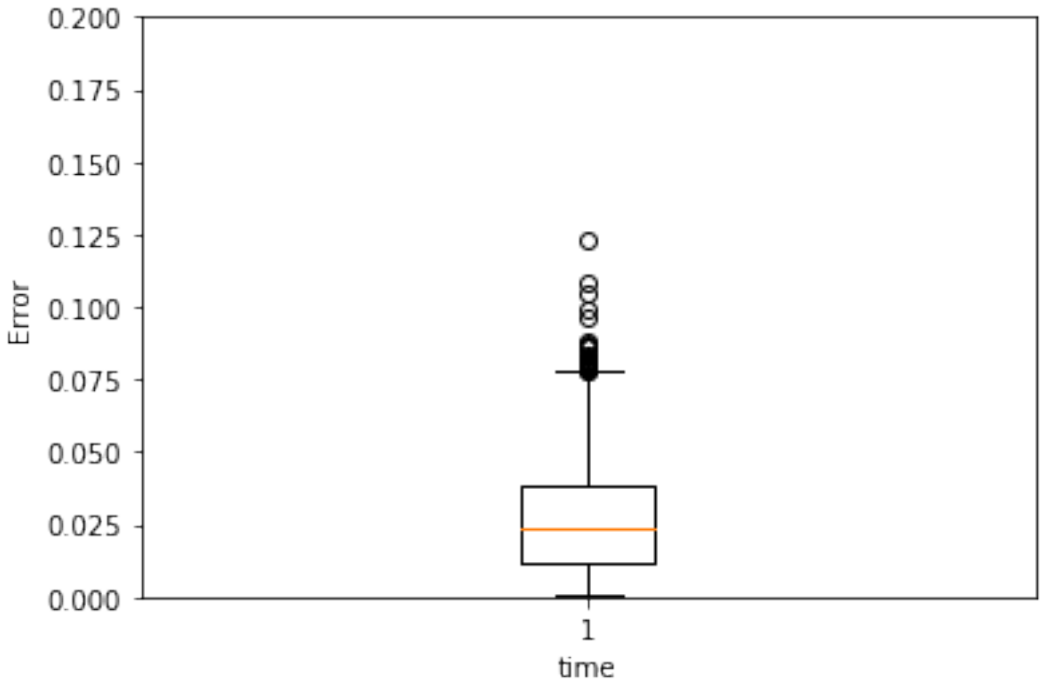


```
0.0292486486395
```



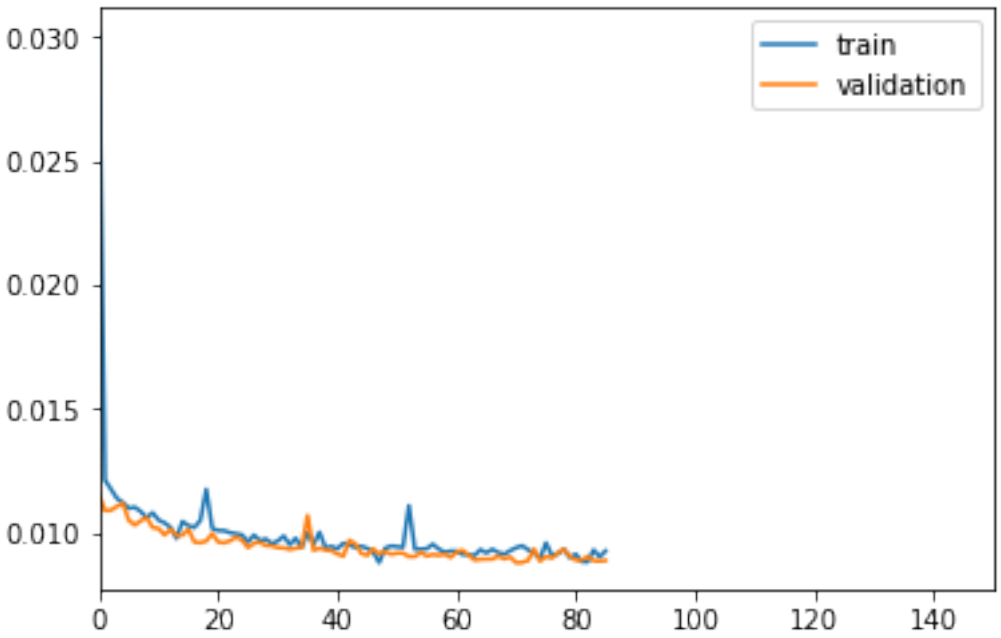


```
0.0271232949804
```



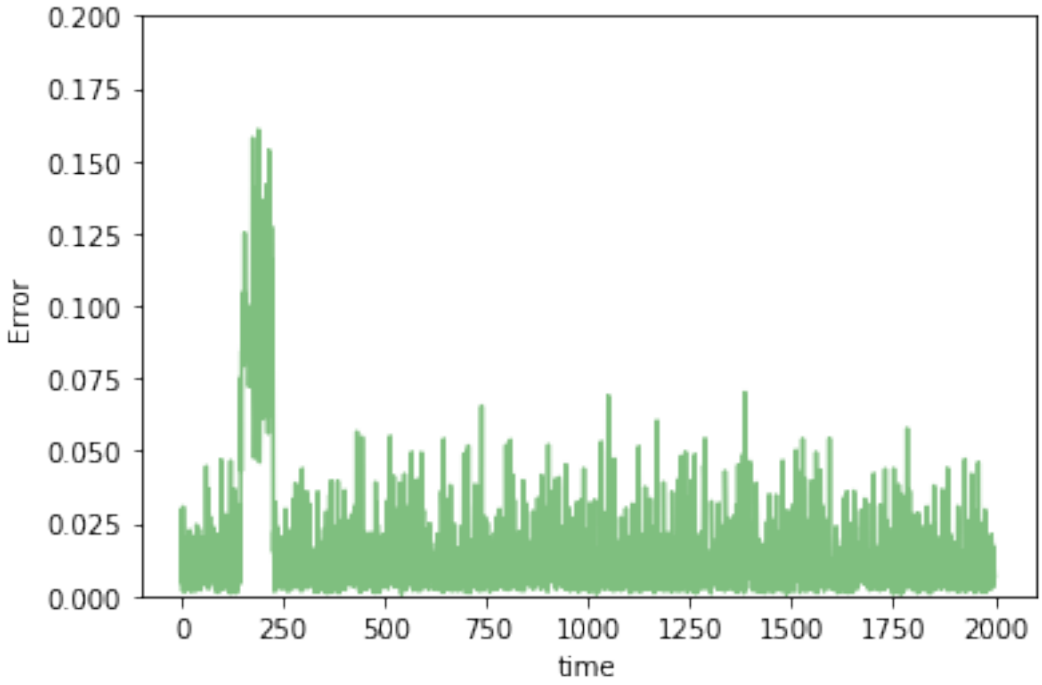
5 steps

```
In [65]: TIMESTEPS = 5
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [66]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(100, activation='relu')(input_layer)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [67]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [68]: train(model, tgen, vgen, name="nn1_5")
```

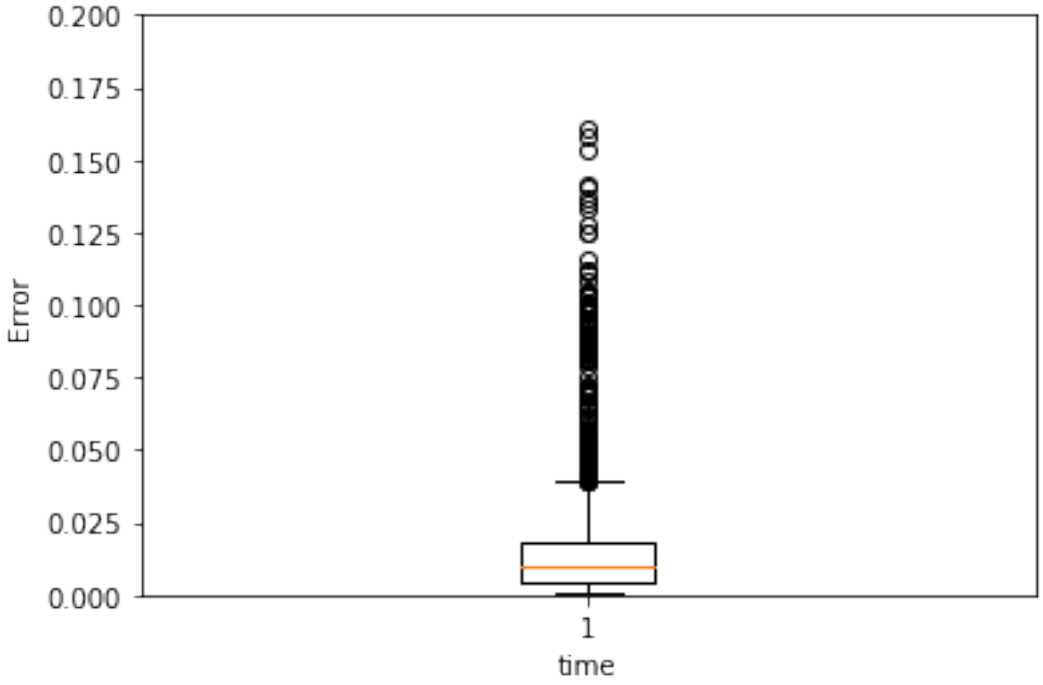


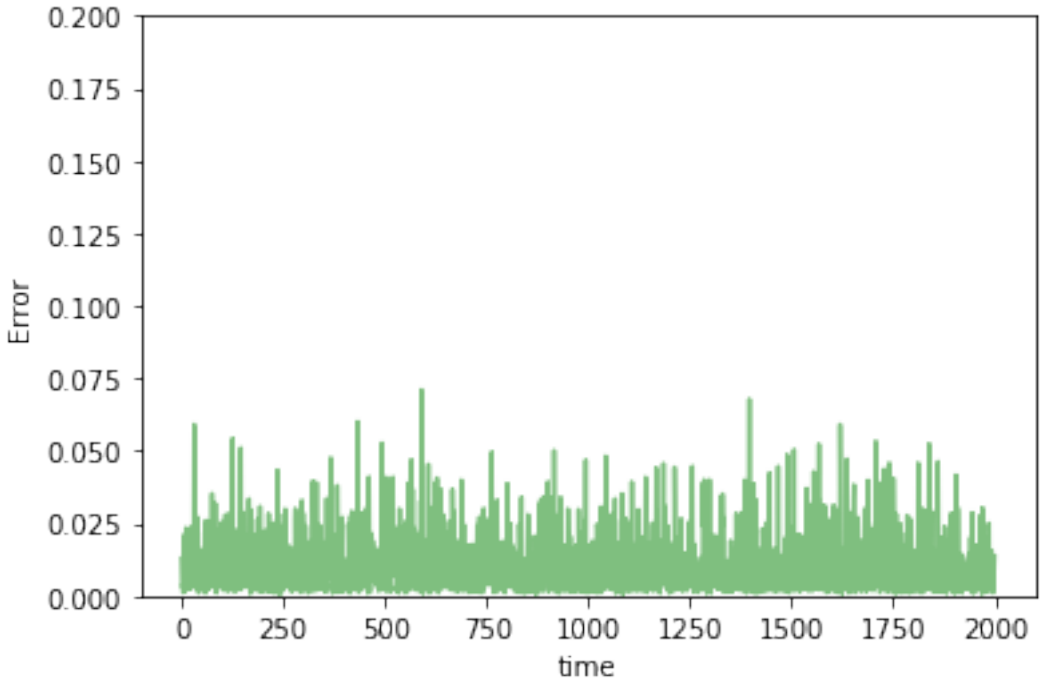

```
0.00925659720204
```

```
In [69]: test(model, test_X[0], name="nn1_5ano")
         test(model, test_X[2], name="nn1_5norm")
```

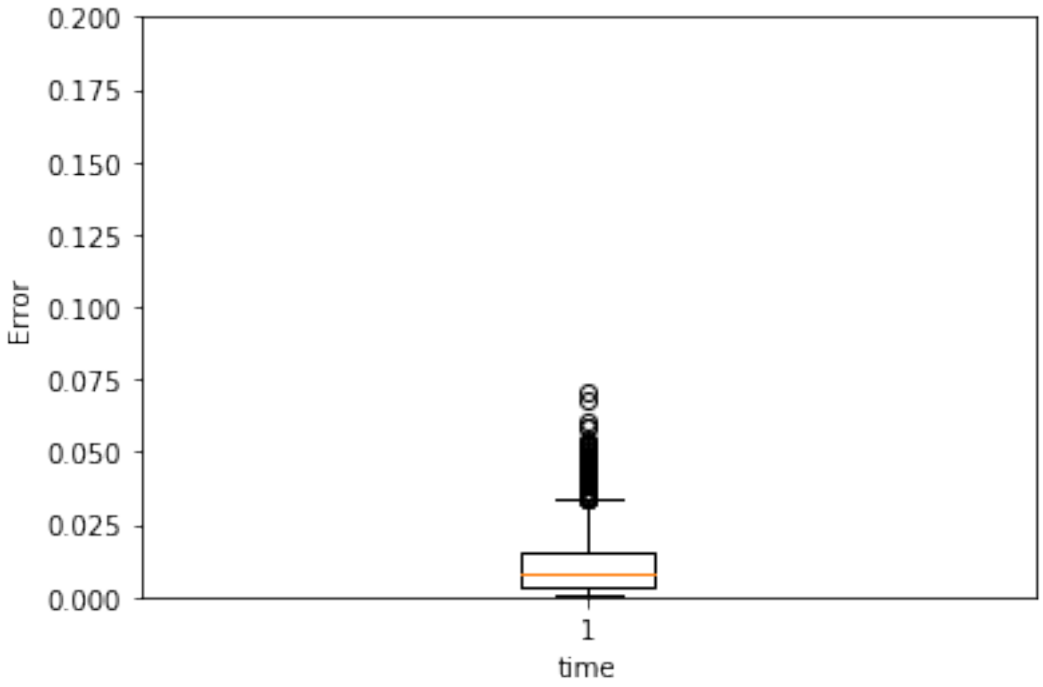


```
0.0155304239169
```



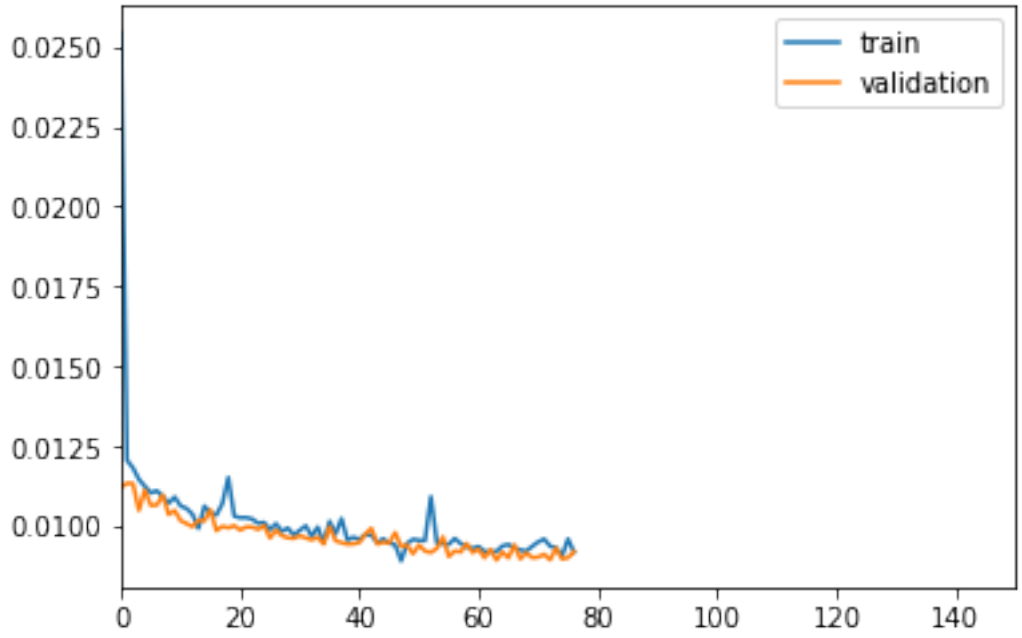


```
0.0111480305704
```



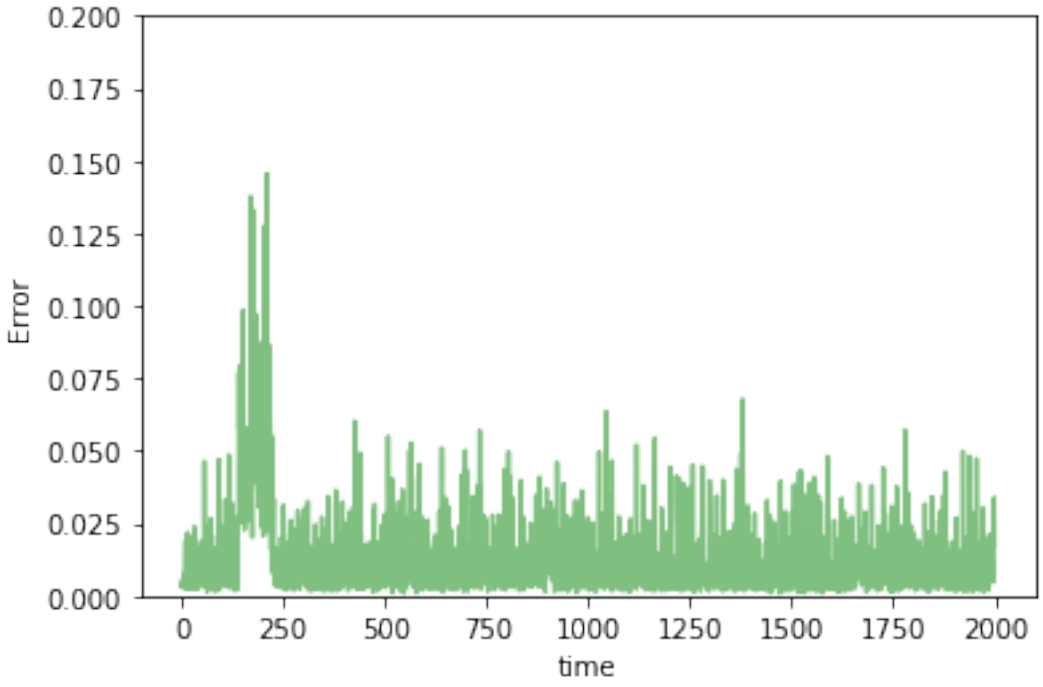
10 steps

```
In [70]: TIMESTEPS = 10
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [71]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(100, activation='relu')(input_layer)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [72]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [73]: train(model, tgen, vgen, name="nn1_10")
```

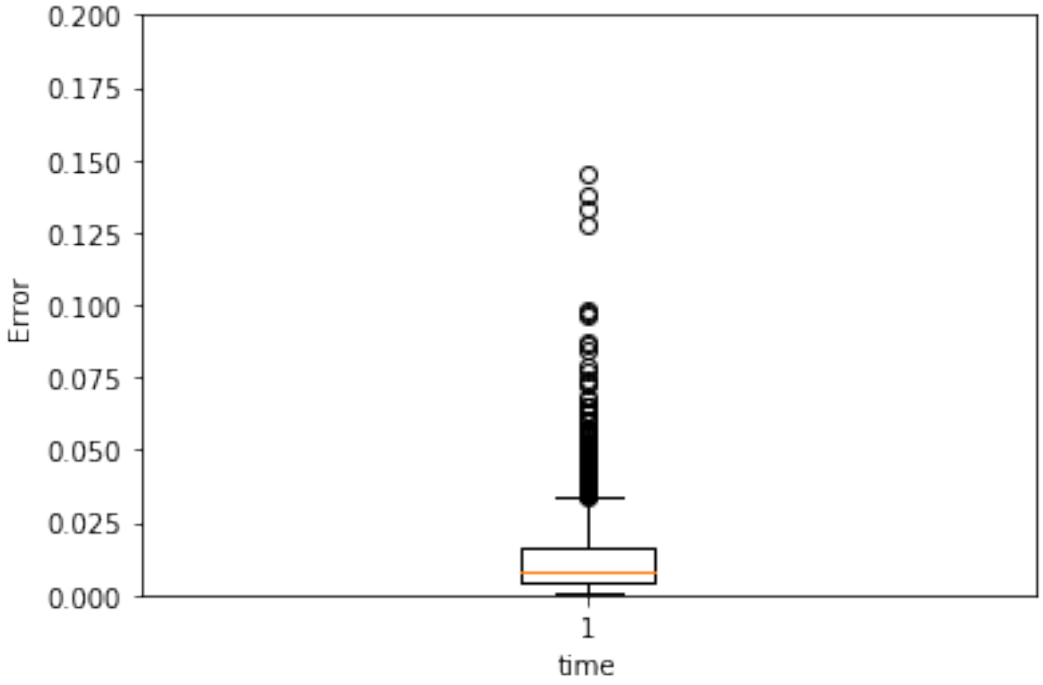


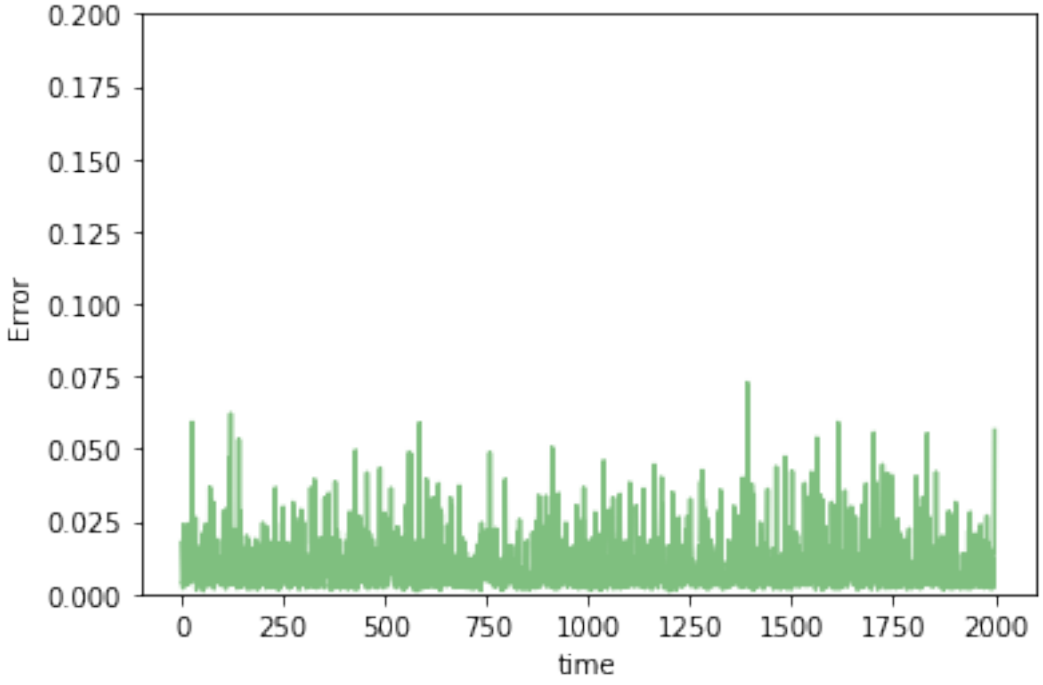

```
0.00920411706157
```

```
In [74]: test(model, test_X[0], name="nn1_10ano")
         test(model, test_X[2], name="nn1_10norm")
```

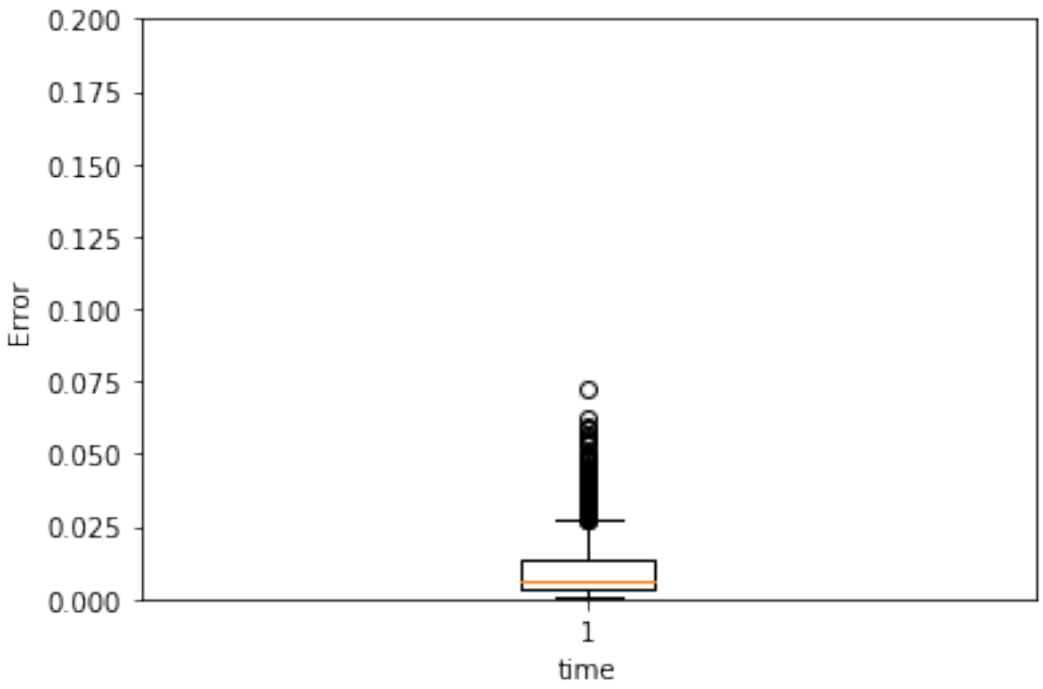


```
0.0129678423925
```



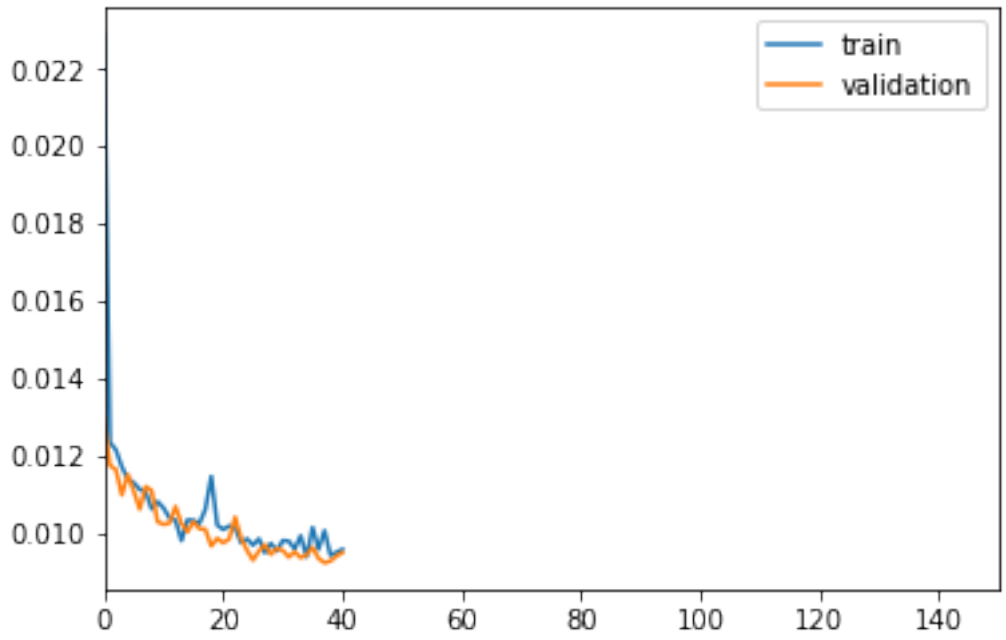


```
0.010001134186
```



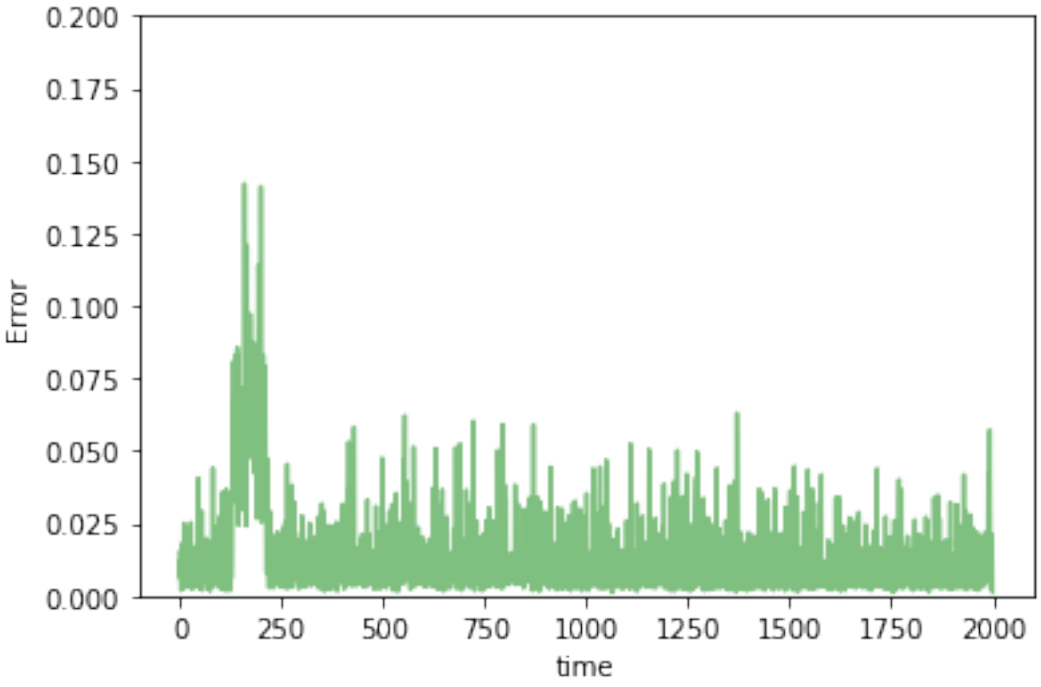
20 steps

```
In [75]: TIMESTEPS = 20
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [76]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(100,activation='relu')(input_layer)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [77]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [78]: train(model, tgen, vgen, name="nn1_20")
```

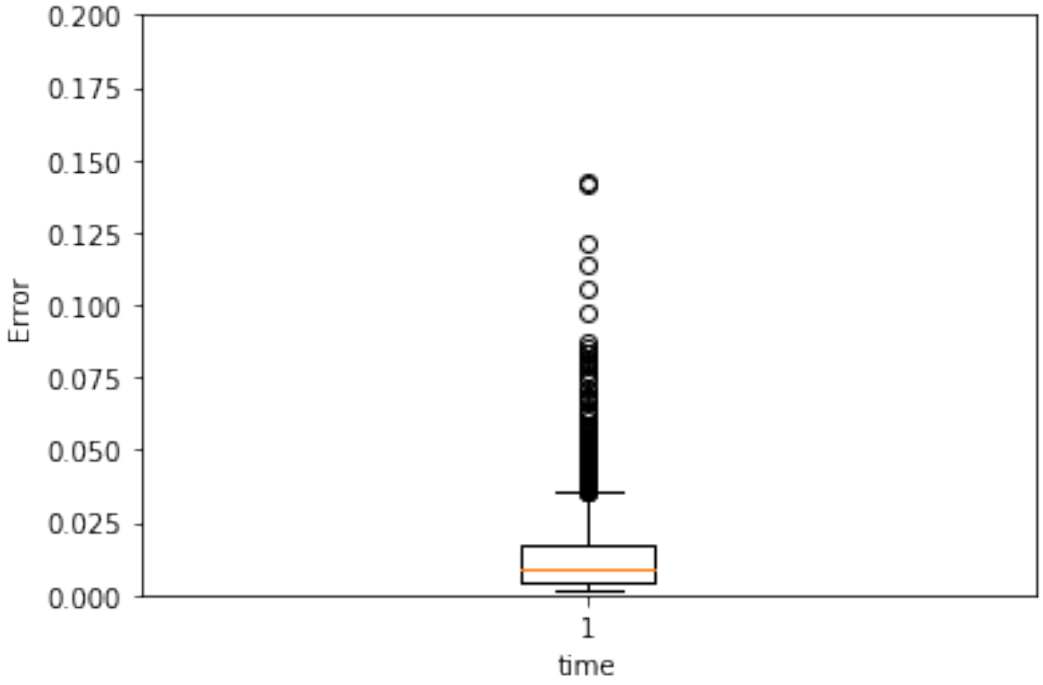


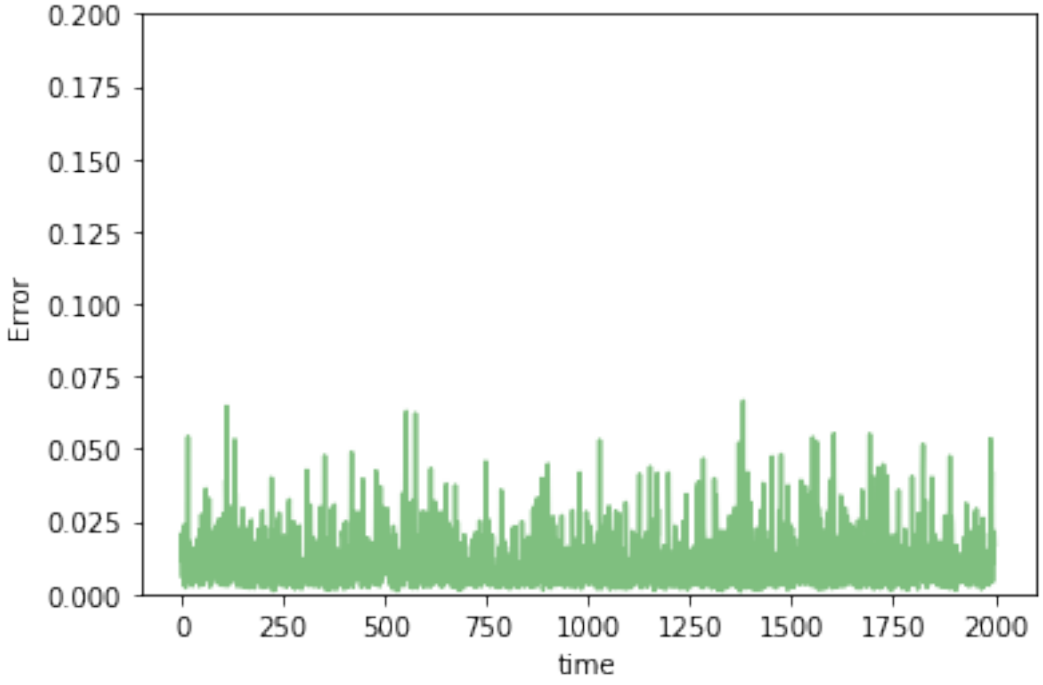

```
0.00955839820695
```

```
In [79]: test(model, test_X[0], name="nn1_20ano")
         test(model, test_X[2], name="nn1_20norm")
```

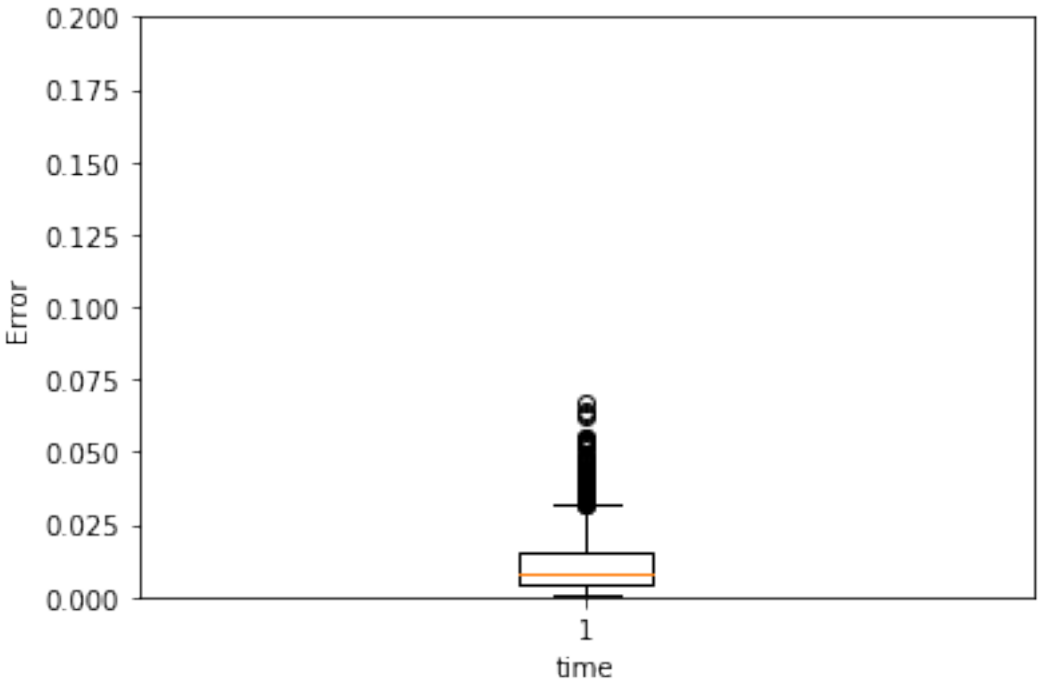


```
0.0135633373553
```



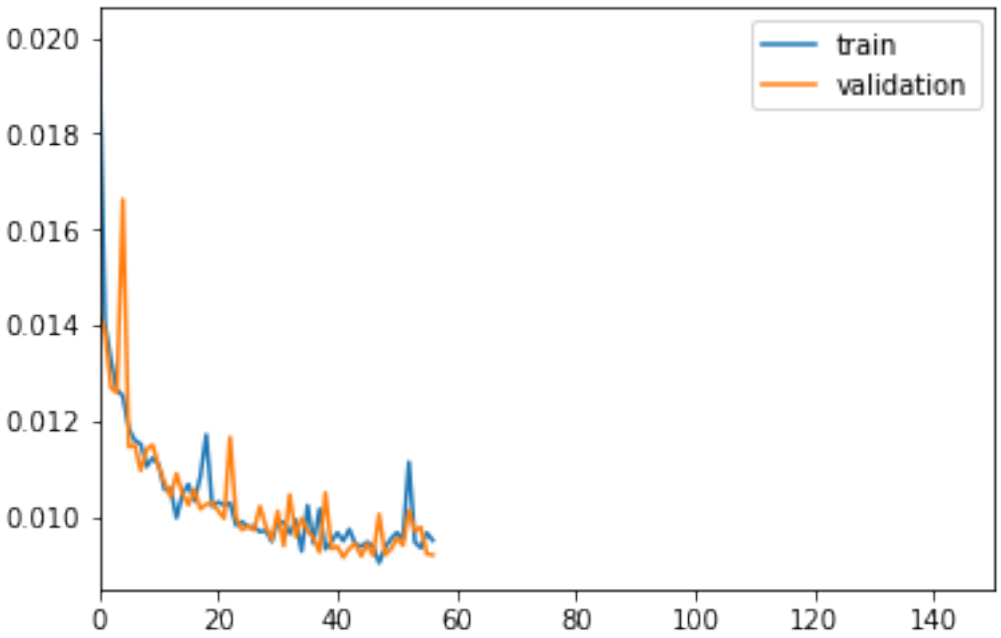


```
0.0110610616489
```



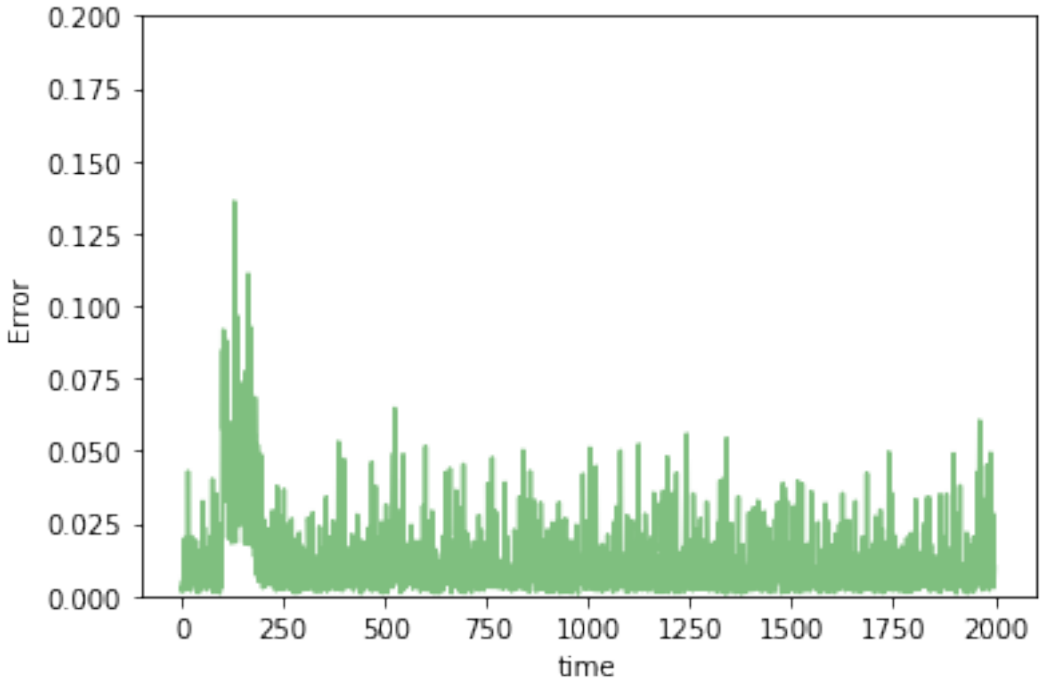
50 steps

```
In [80]: TIMESTEPS = 50
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [81]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(100,activation='relu')(input_layer)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [82]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [83]: train(model, tgen, vgen, name="nn1_50")
```

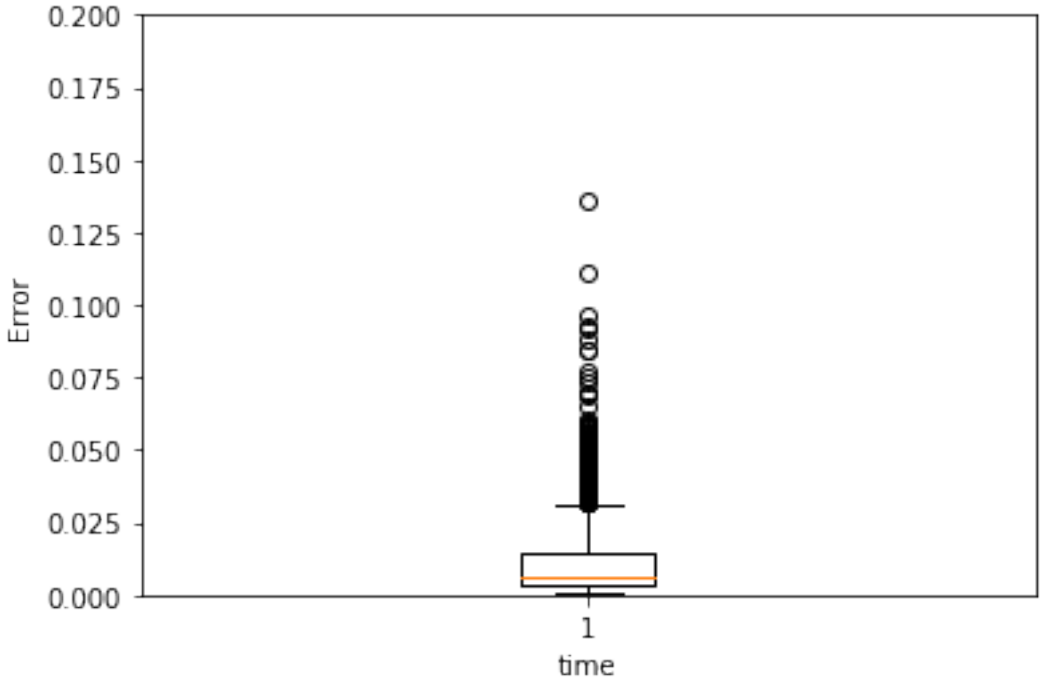


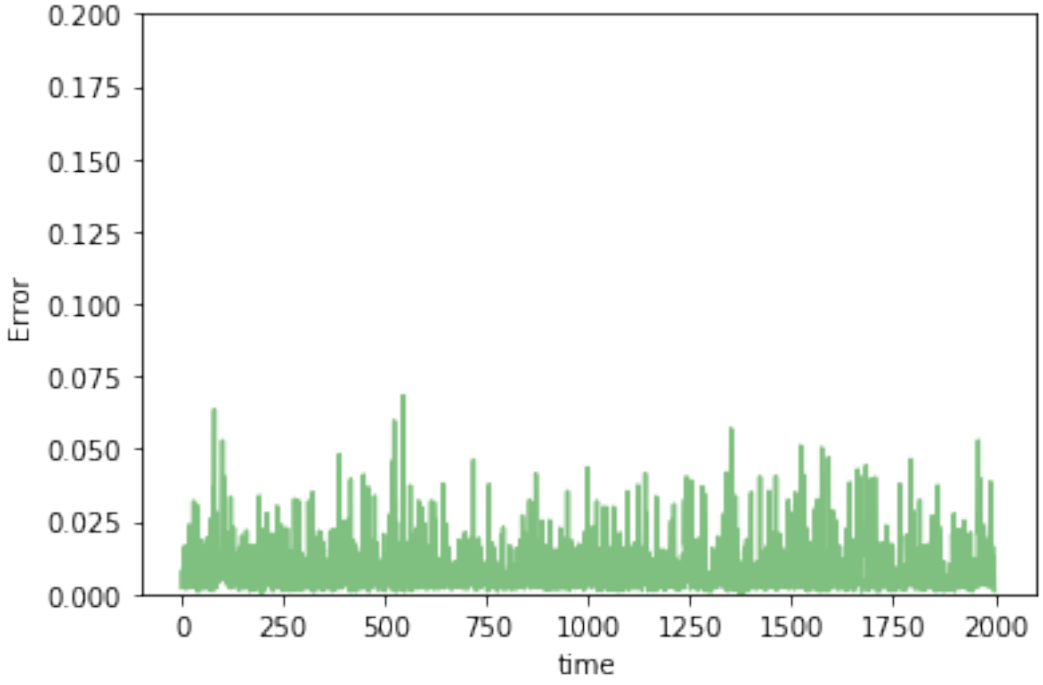

```
0.00950964211253
```

```
In [84]: test(model, test_X[0], name="nn1_50ano")
         test(model, test_X[2], name="nn1_50norm")
```

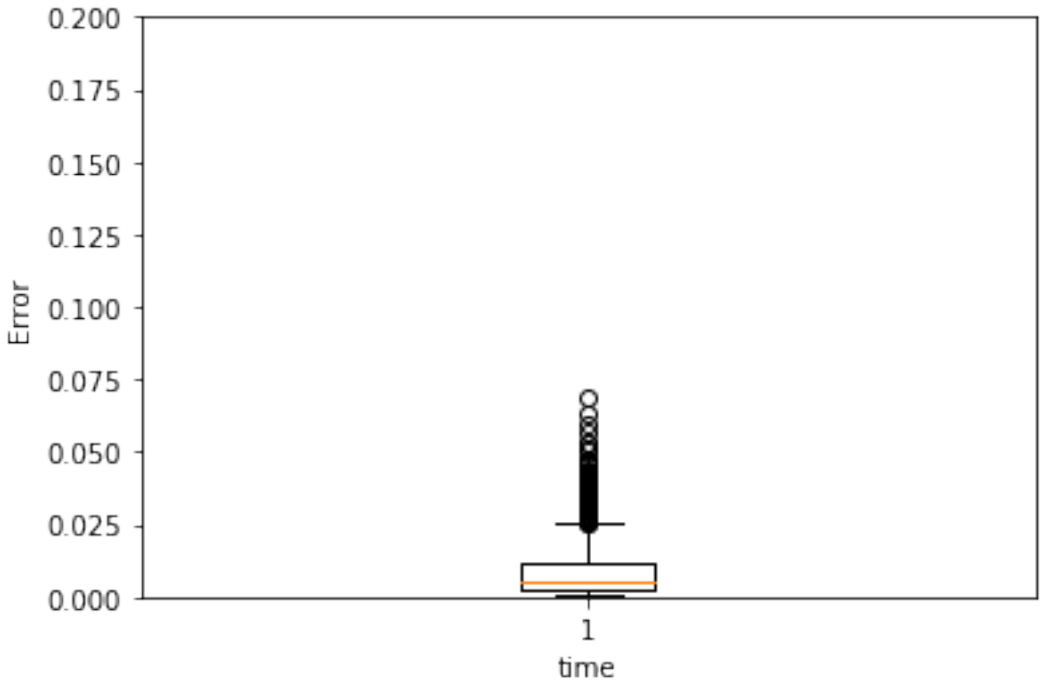


```
0.0114504249401
```





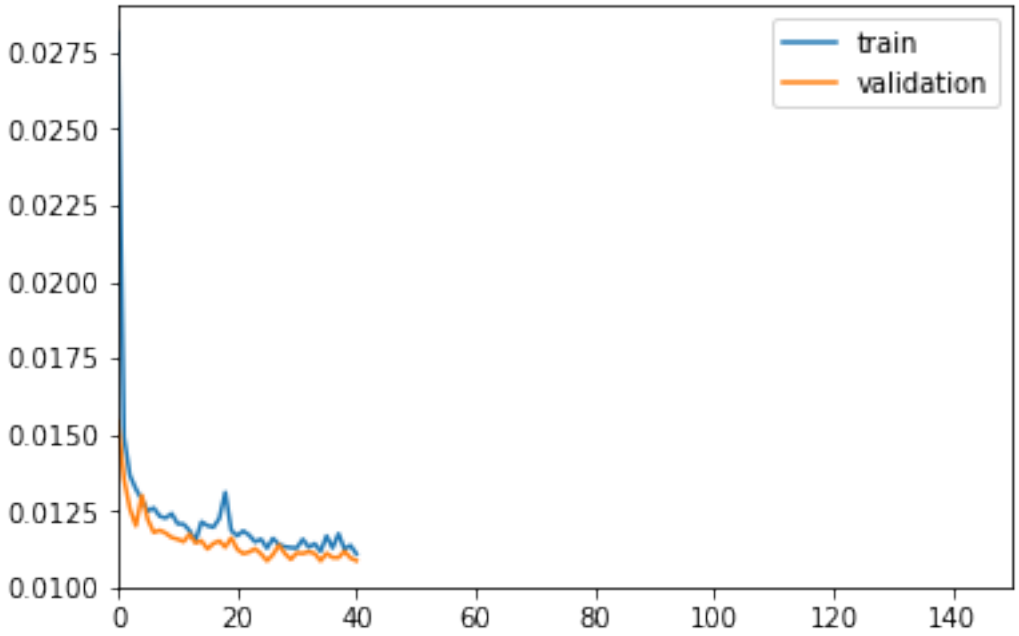
```
0.00878205366855
```



2.1.3 NN with 2 hidden layers

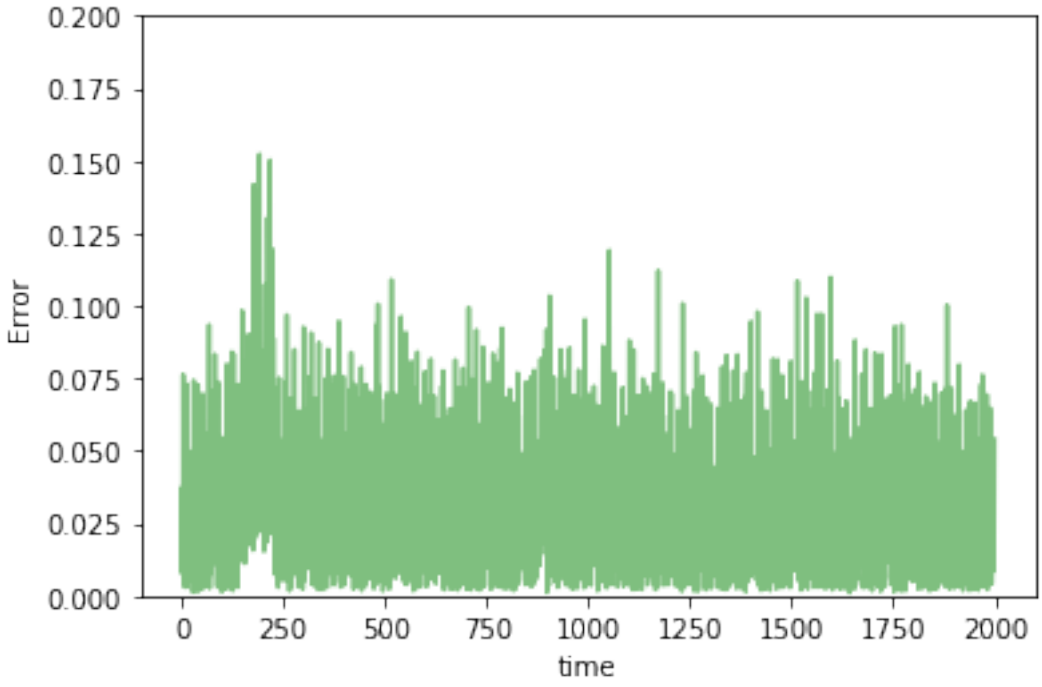
2 steps

```
In [85]: TIMESTEPS = 2
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [86]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(500, activation='relu')(input_layer)
         hidden = Dense(100, activation='relu')(hidden)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [87]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [88]: train(model, tgen, vgen, name="nn2_2")
```

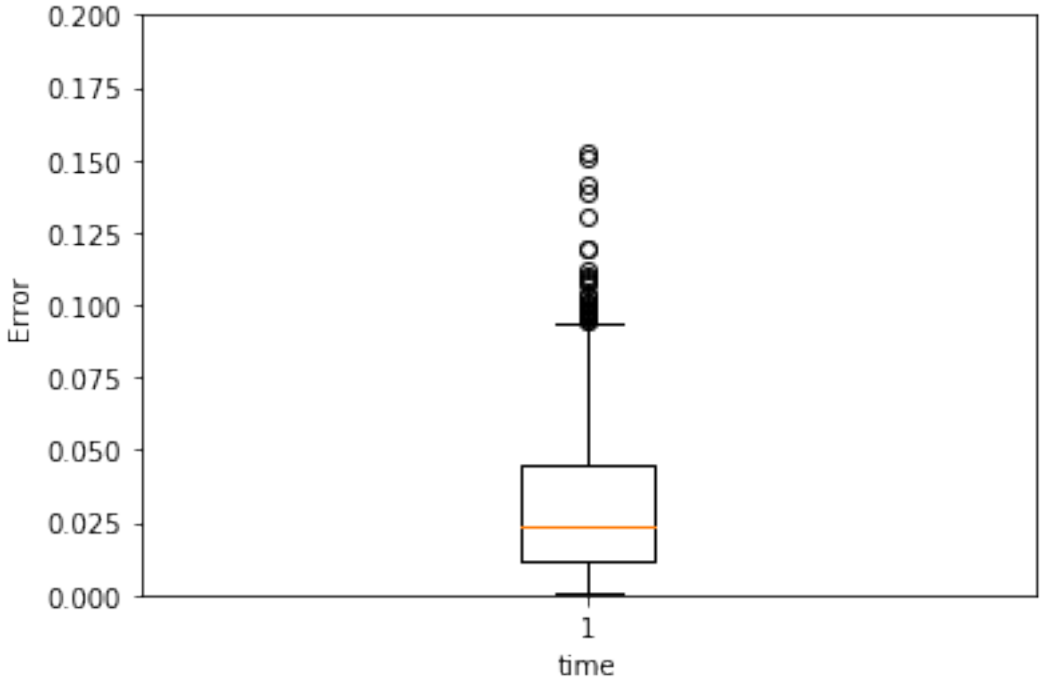


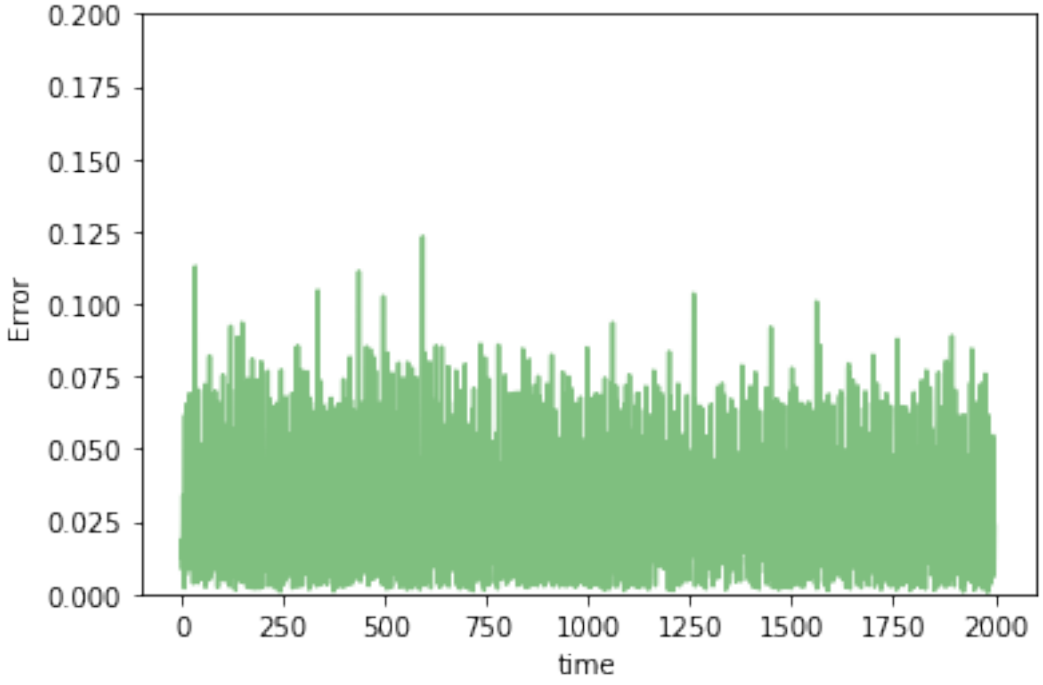

```
0.0111092675955
```

```
In [89]: test(model, test_X[0], name="nn2_2ano")
         test(model, test_X[2], name="nn2_2norm")
```

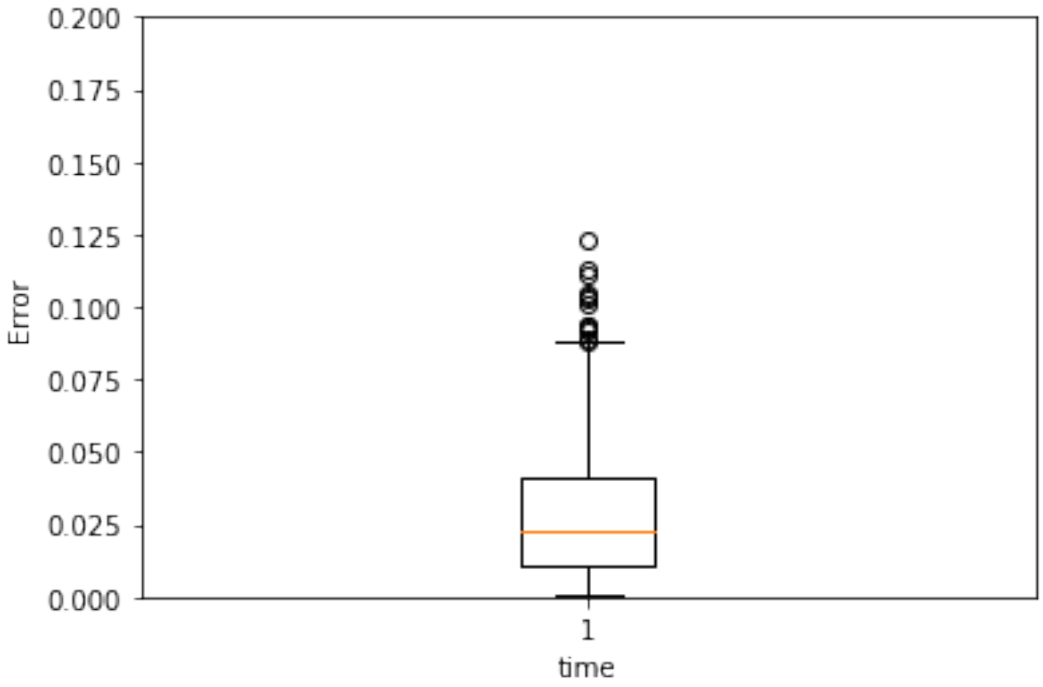


```
0.0307701168935
```



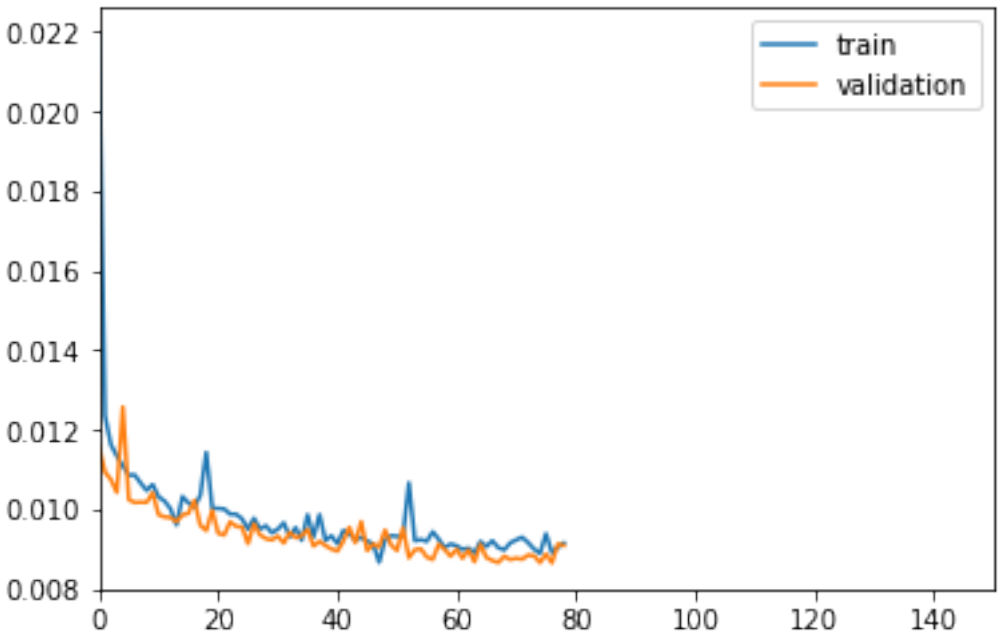


```
0.0285915274607
```



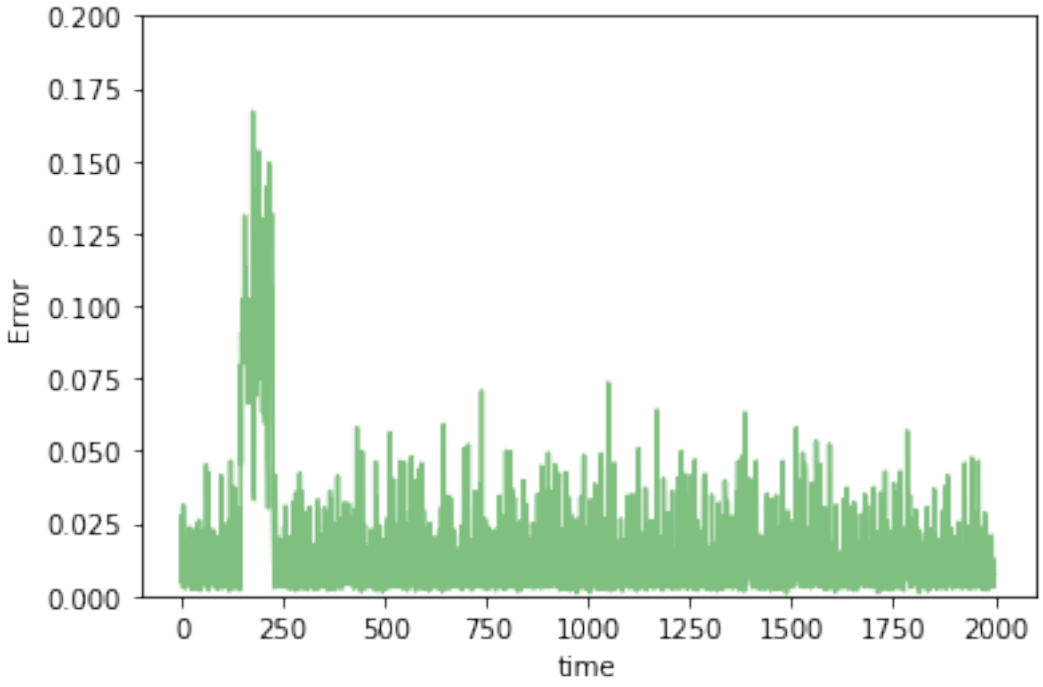
5 steps

```
In [90]: TIMESTEPS = 5
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [91]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(500, activation='relu')(input_layer)
         hidden = Dense(100, activation='relu')(hidden)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [92]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [93]: train(model, tgen, vgen, name="nn2_5")
```

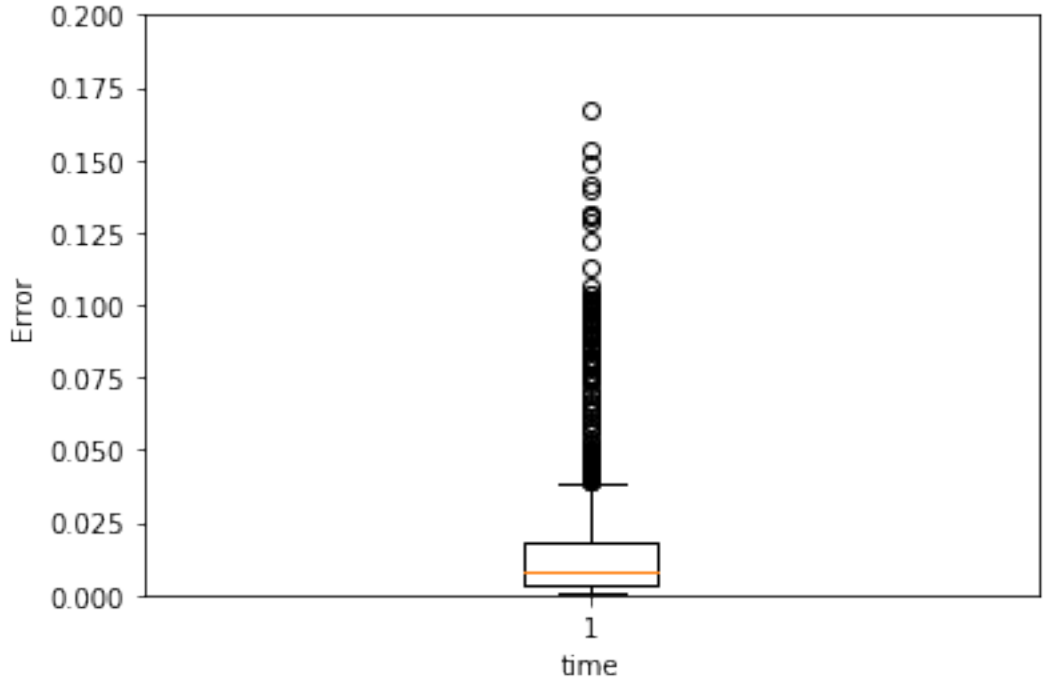


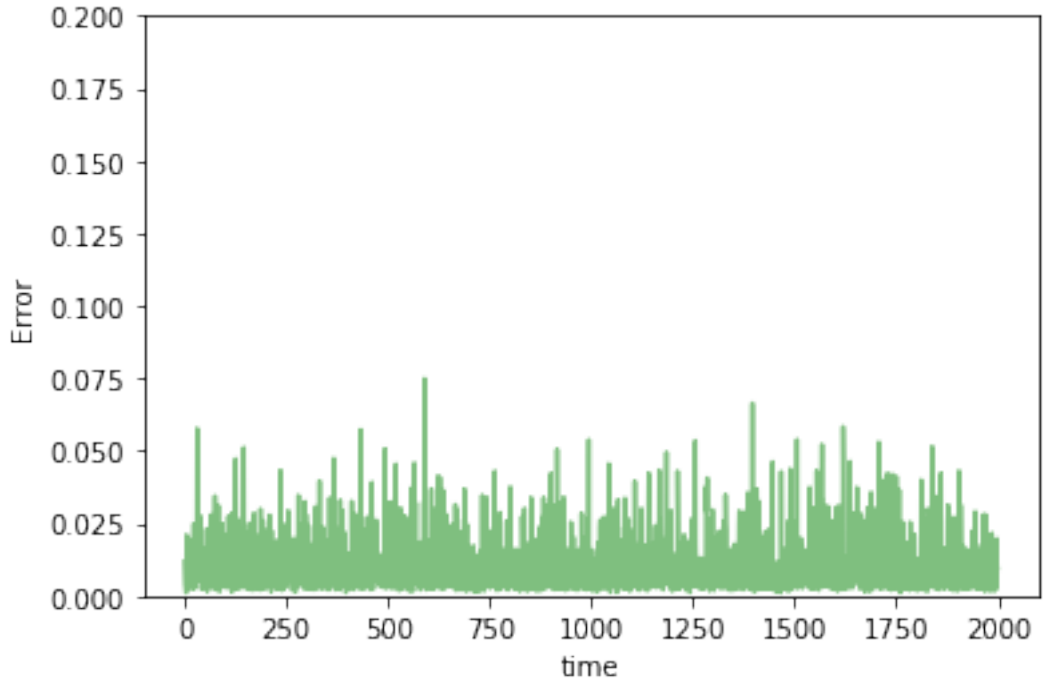

```
0.00915318725118
```

```
In [94]: test(model, test_X[0], name="nn2_5ano")
         test(model, test_X[2], name="nn2_5norm")
```

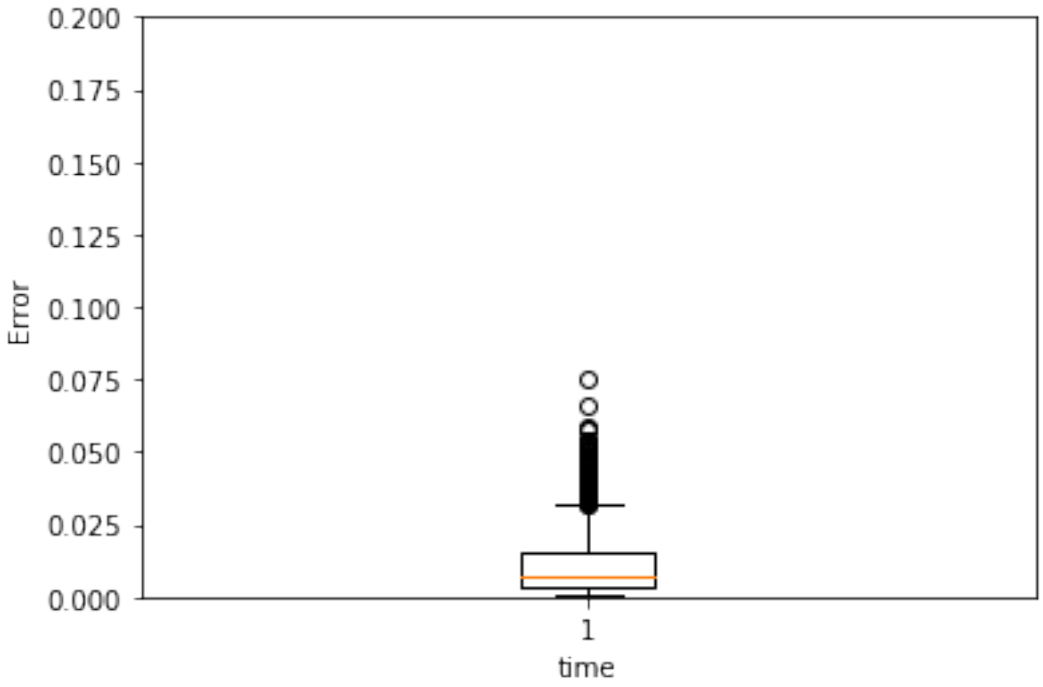


```
0.014962636683
```



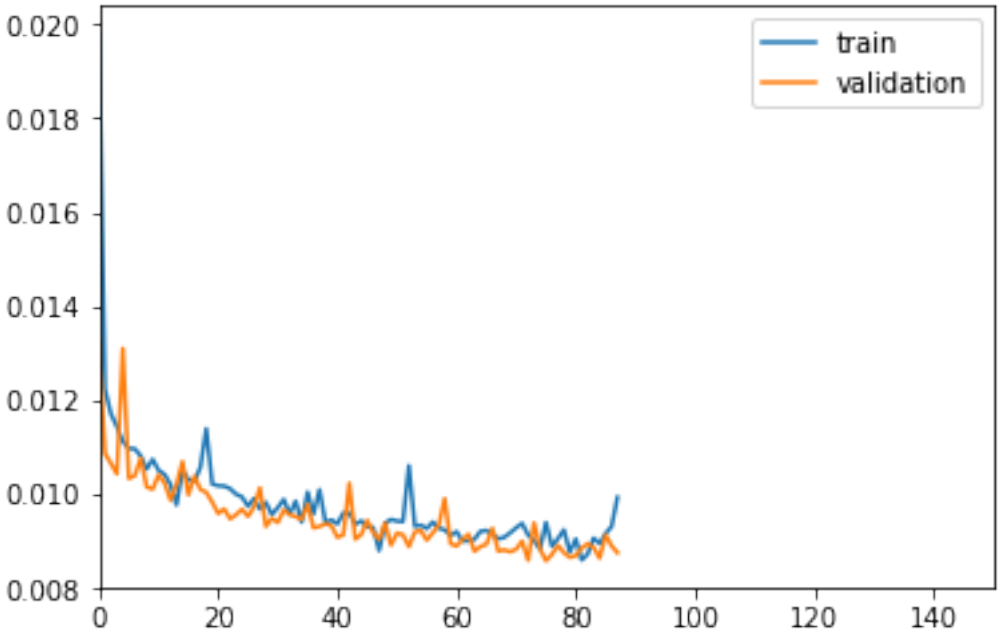


```
0.0107029876701
```



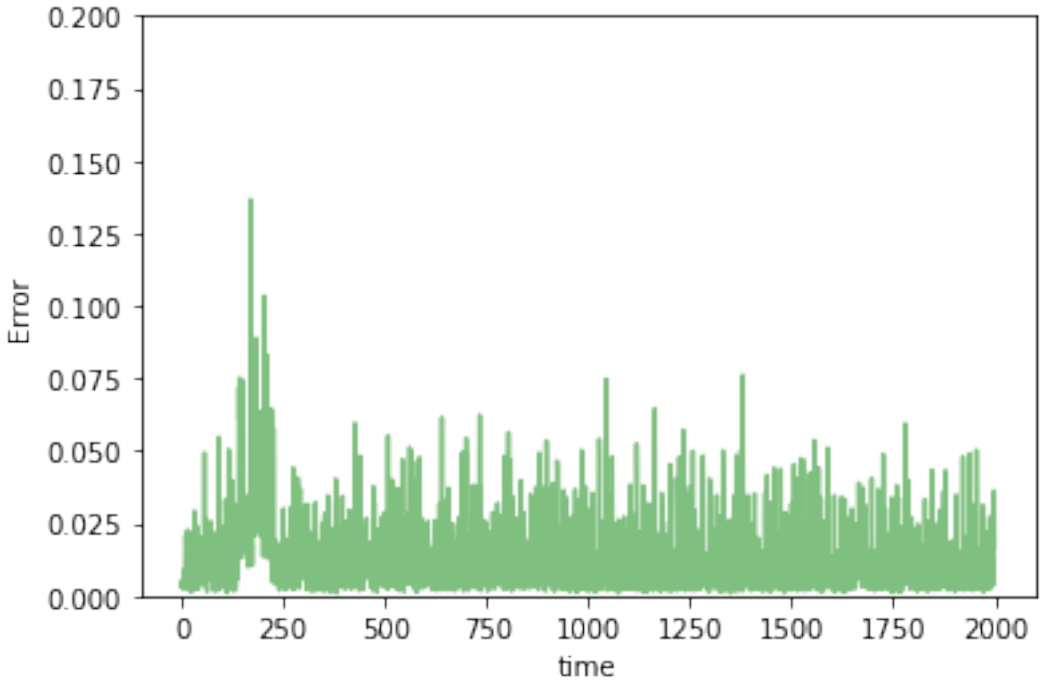
10 steps

```
In [95]: TIMESTEPS = 10
         DIM = 29
         tgen = flat_generator(X, TIMESTEPS)
         vgen = flat_generator(val_X, TIMESTEPS)

In [96]: input_layer = Input(shape=(TIMESTEPS*DIM,))
         hidden = Dense(500, activation='relu')(input_layer)
         hidden = Dense(100, activation='relu')(hidden)
         output = Dense(DIM, activation='sigmoid')(hidden)

In [97]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [98]: train(model, tgen, vgen, name="nn2_10")
```

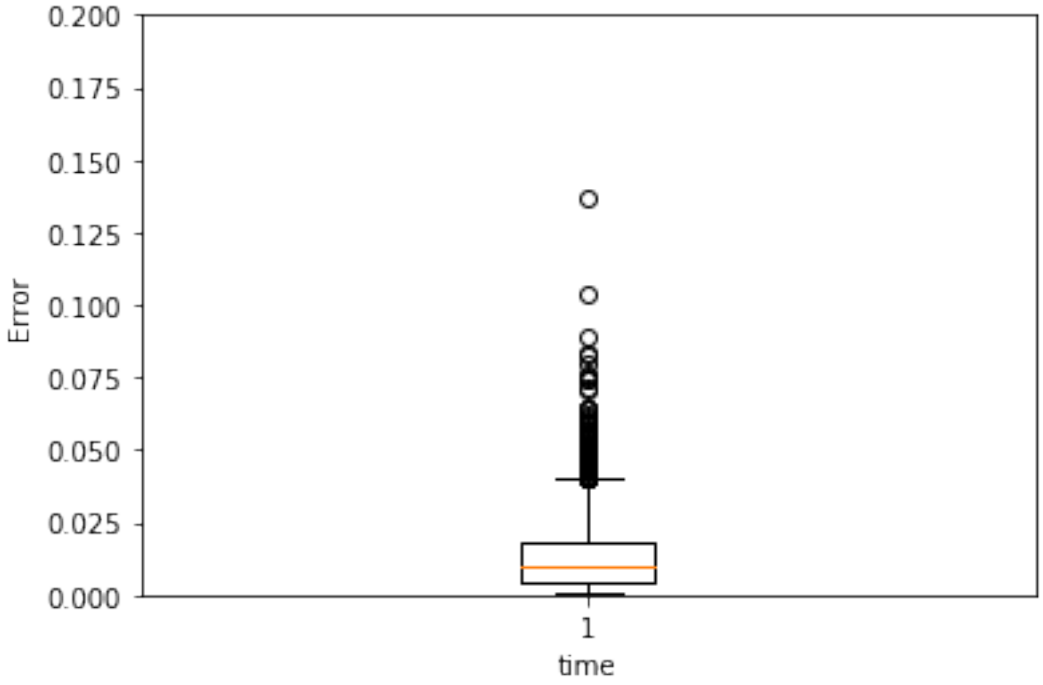


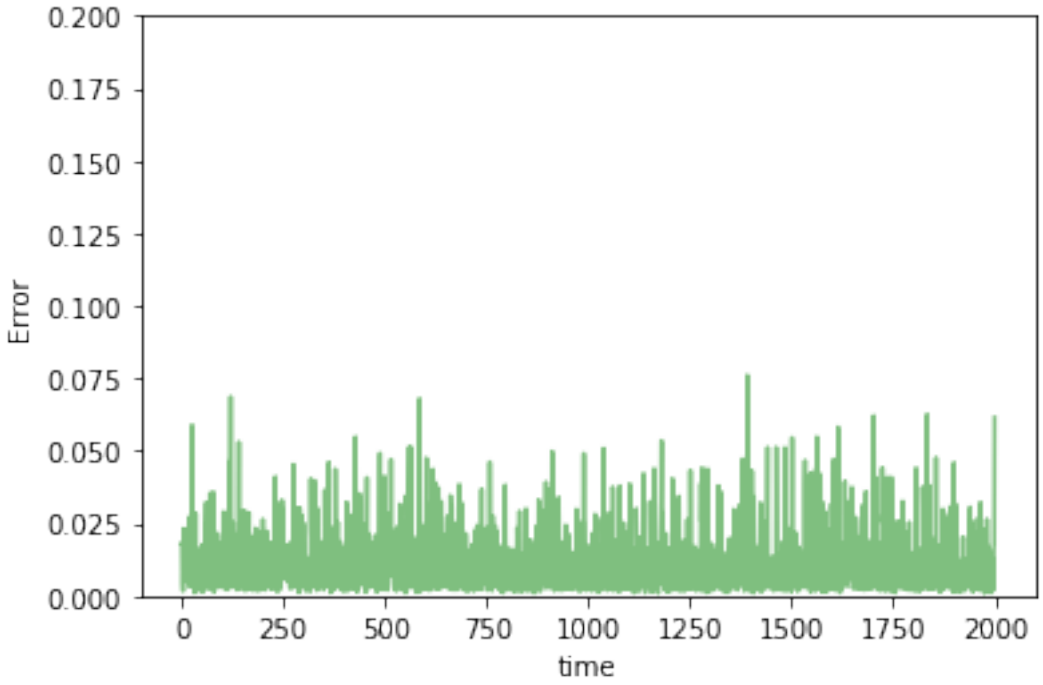

```
0.00993760640232
```

```
In [99]: test(model, test_X[0], name="nn2_10ano")
         test(model, test_X[2], name="nn2_10norm")
```

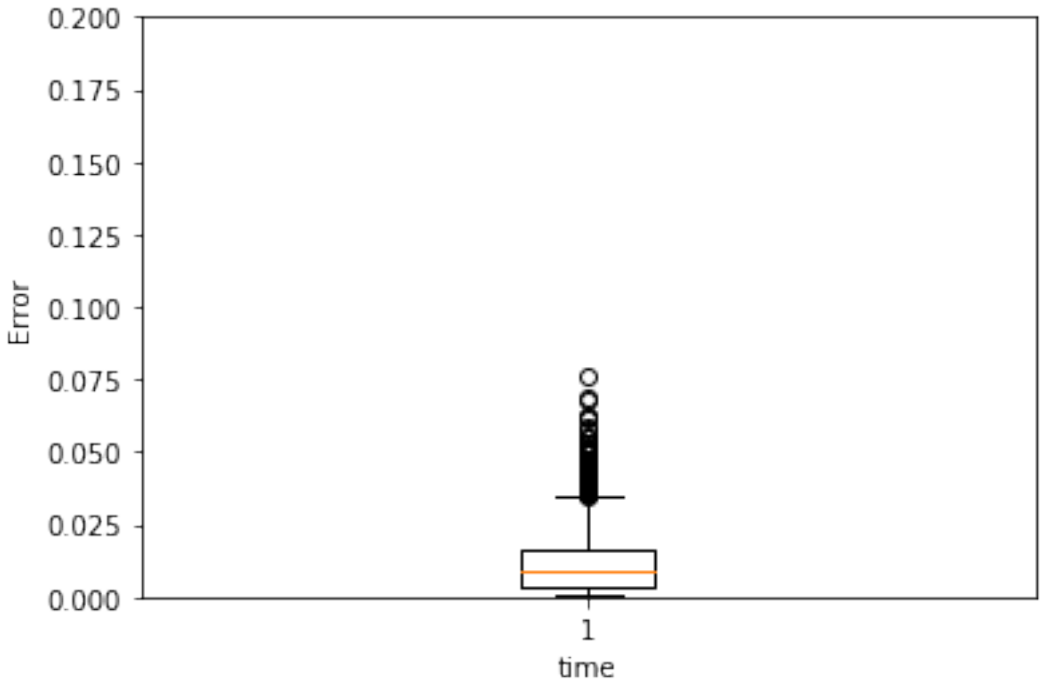


```
0.0136270285173
```



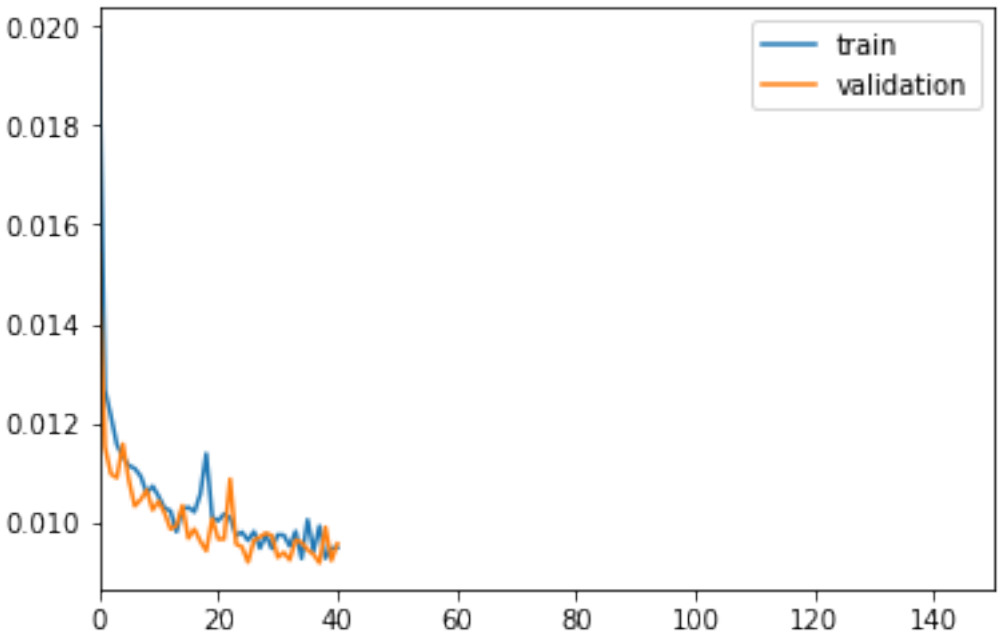


```
0.011800808805
```



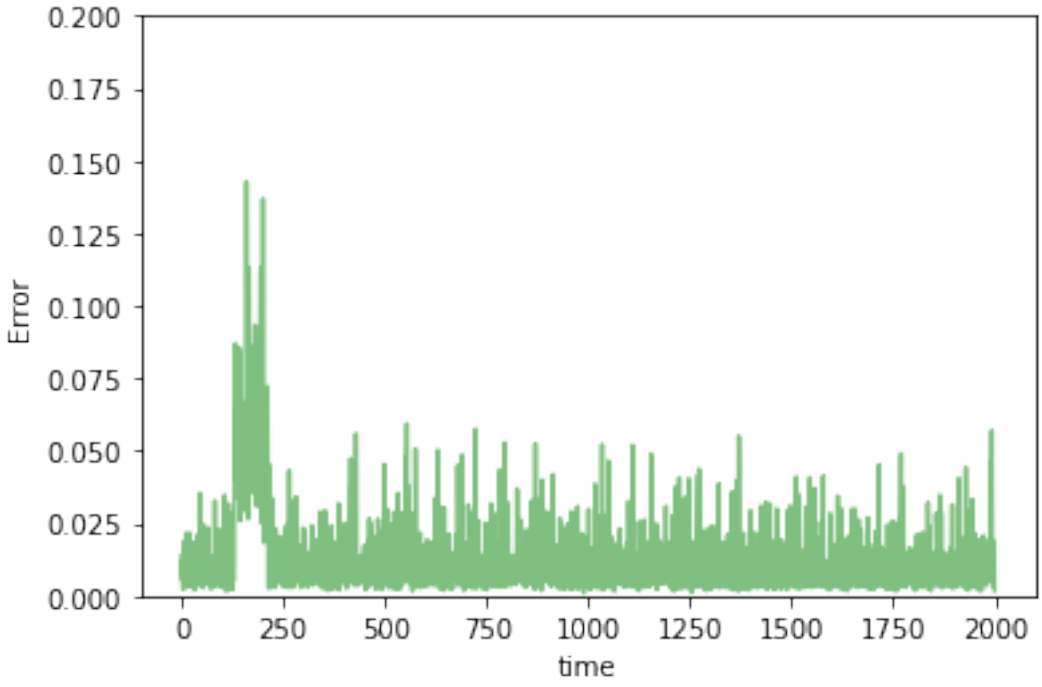
20 steps

```
In [100]: TIMESTEPS = 20
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [101]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(500, activation='relu')(input_layer)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [102]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [103]: train(model, tgen, vgen, name="nn2_20")
```

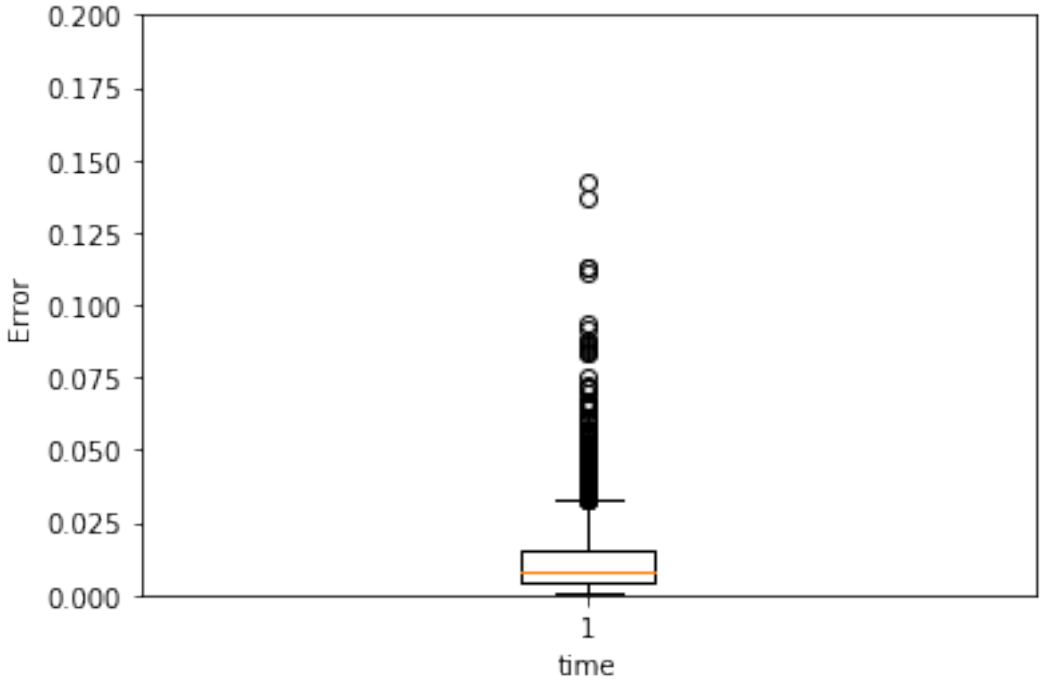


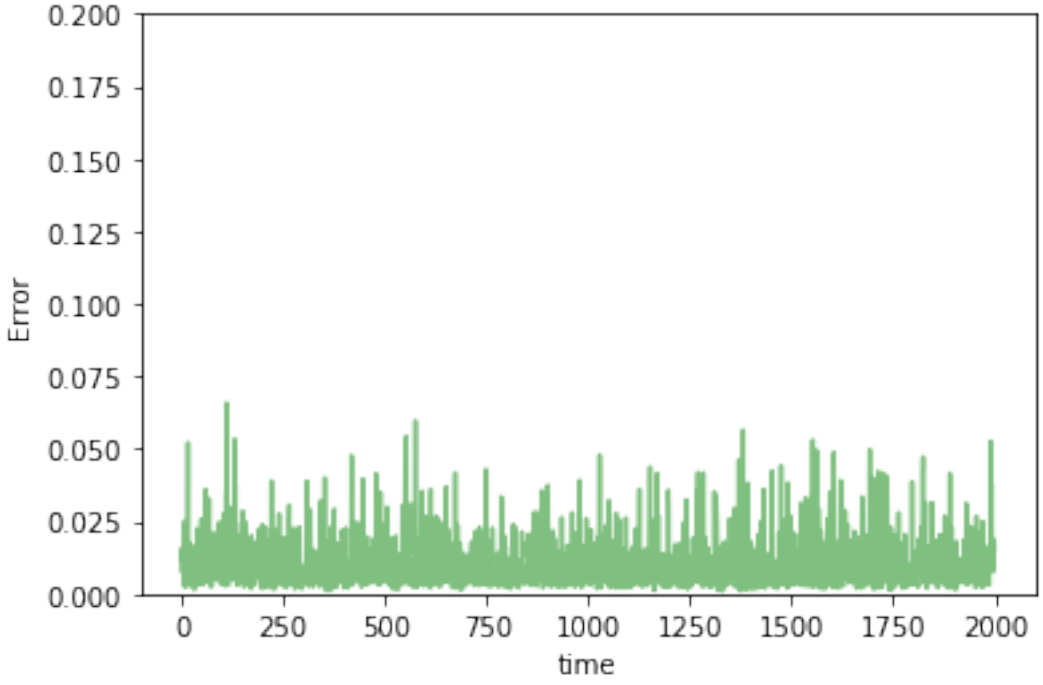

```
0.00949717467639
```

```
In [104]: test(model, test_X[0], name="nn2_20ano")
          test(model, test_X[2], name="nn2_20norm")
```

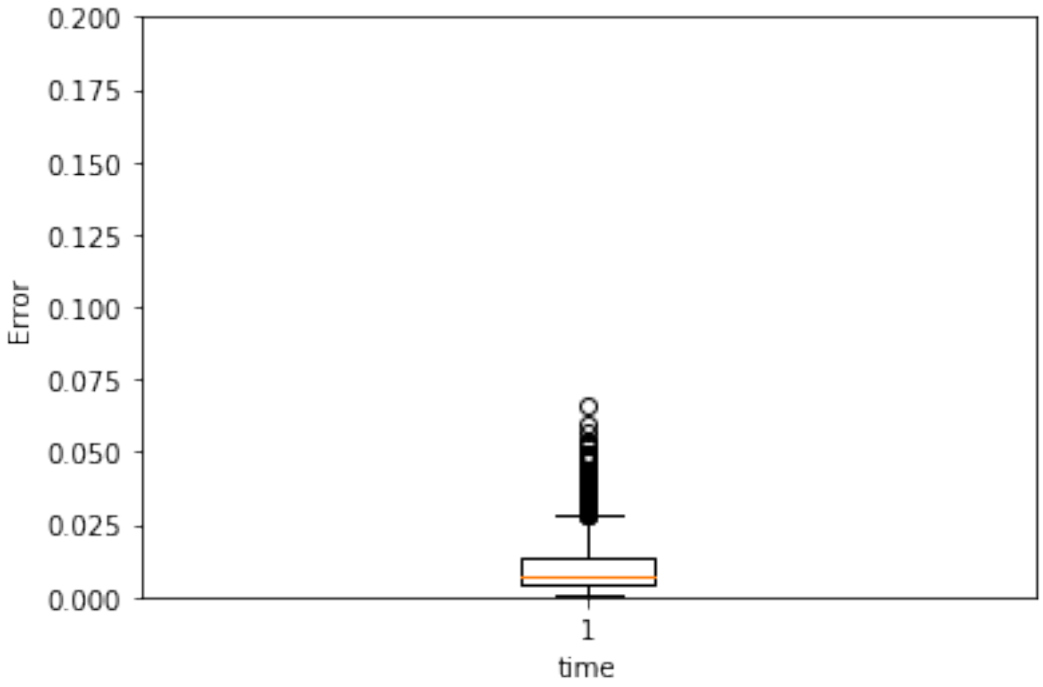


```
0.012834098495
```



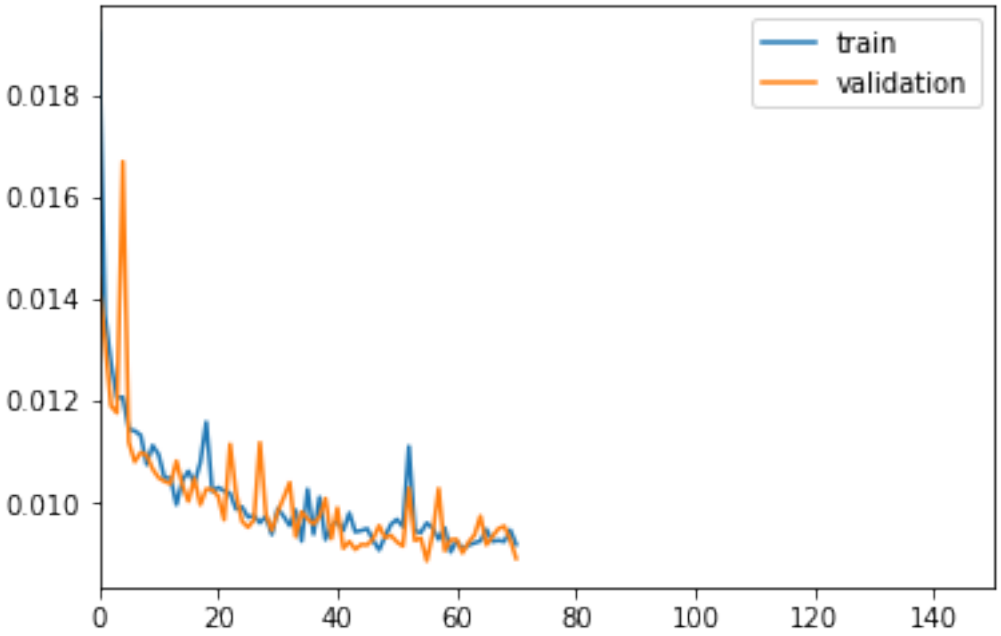


```
0.0104359103315
```



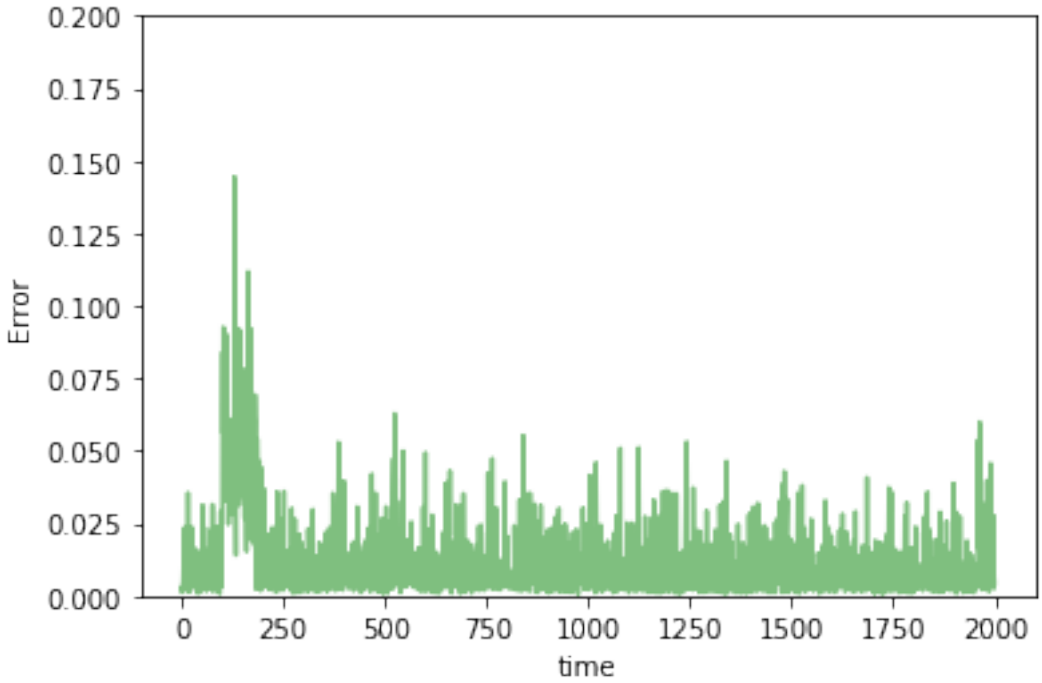
50 steps

```
In [105]: TIMESTEPS = 50
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [106]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(500, activation='relu')(input_layer)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [107]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [108]: train(model, tgen, vgen, name="nn2_50")
```

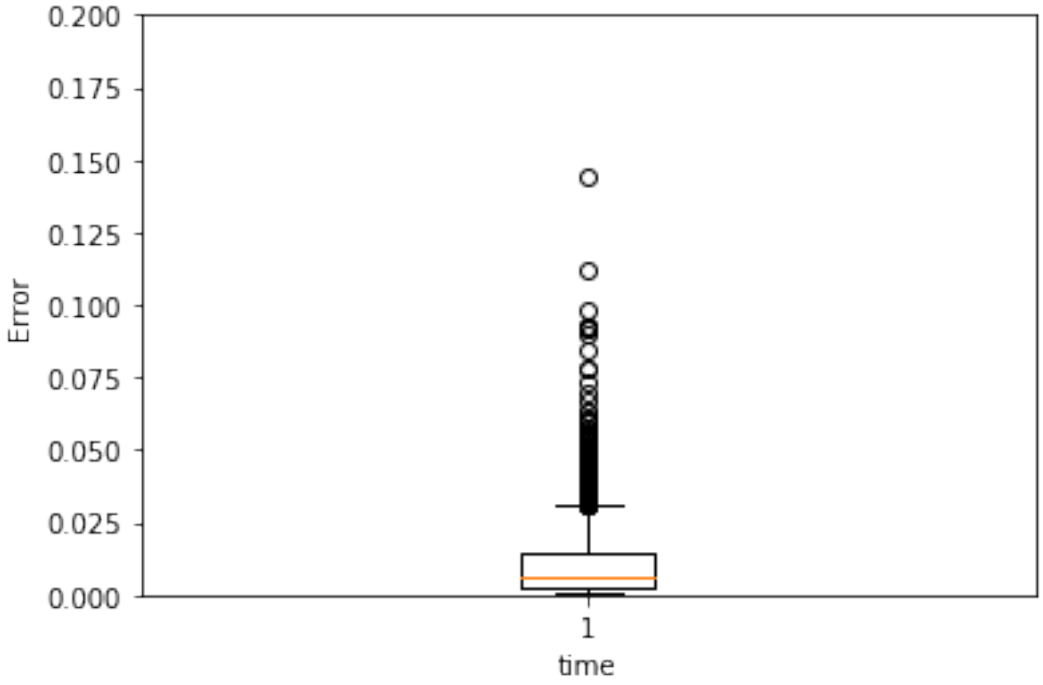


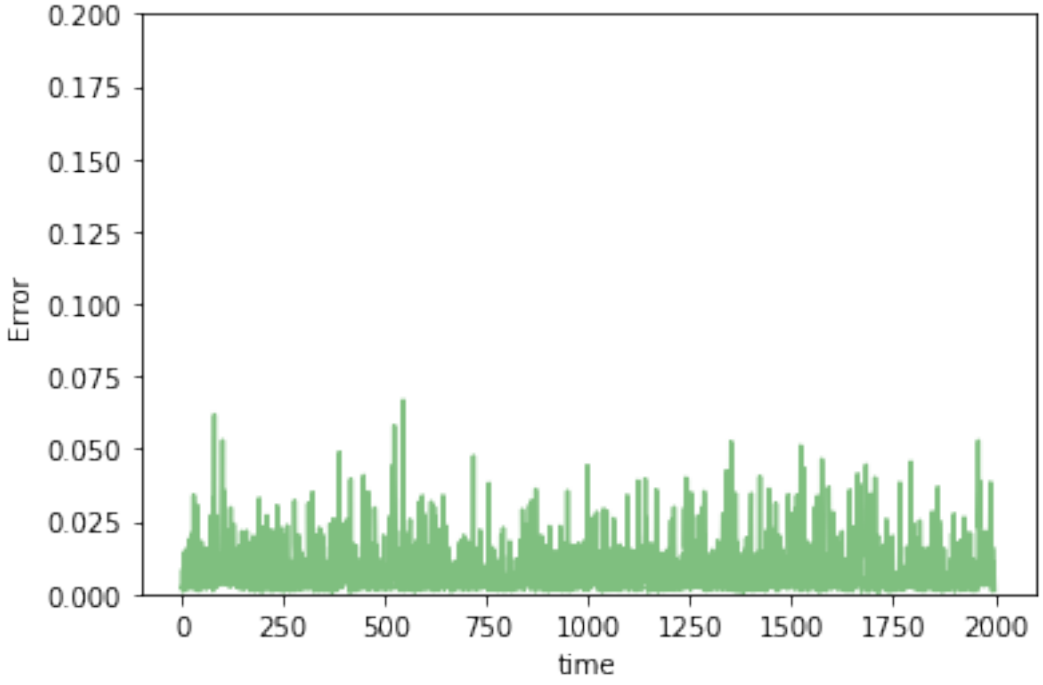

```
0.00918484578095
```

```
In [109]: test(model, test_X[0], name="nn2_50ano")
          test(model, test_X[2], name="nn2_50norm")
```

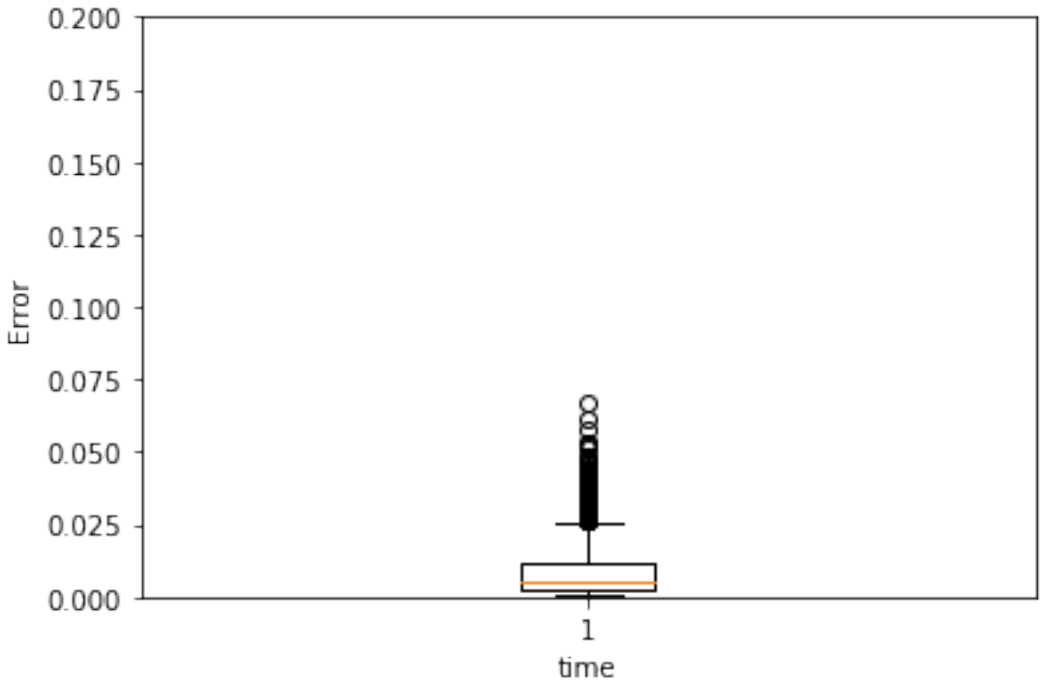


```
0.0110541197056
```





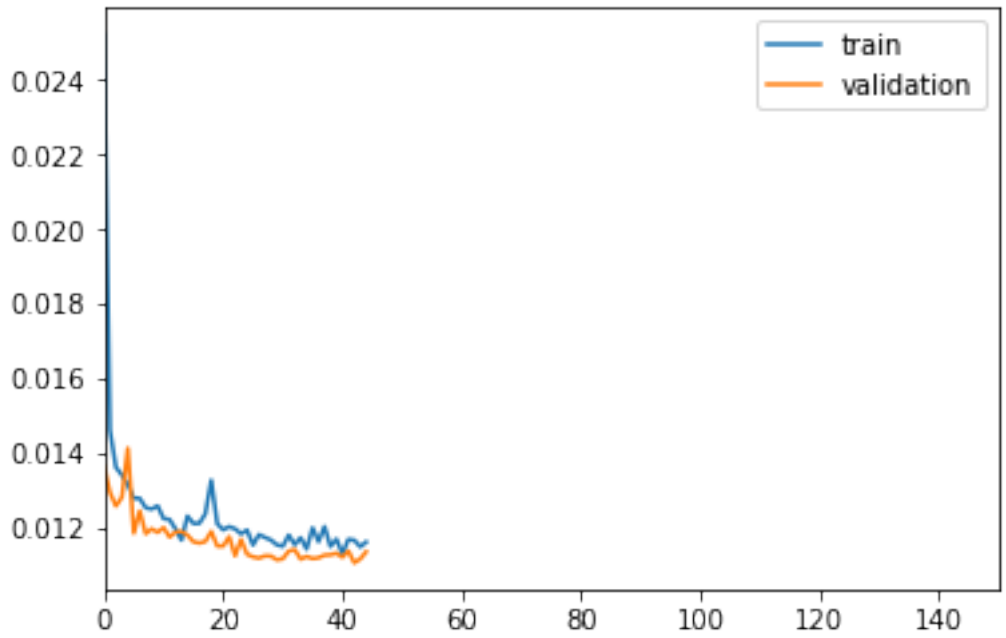
```
0.00847905990141
```



2.1.4 NN with 3 hidden layers

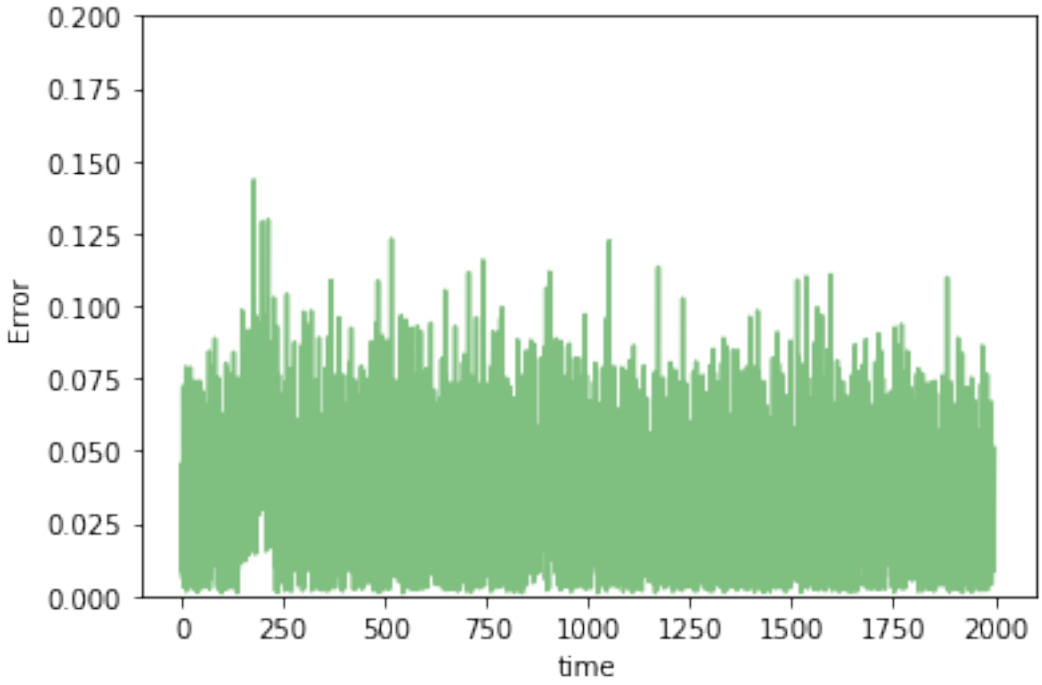
2 steps

```
In [110]: TIMESTEPS = 2
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [111]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(1000, activation='relu')(input_layer)
          hidden = Dense(500, activation='relu')(hidden)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [112]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [113]: train(model, tgen, vgen, name="nn3_2")
```

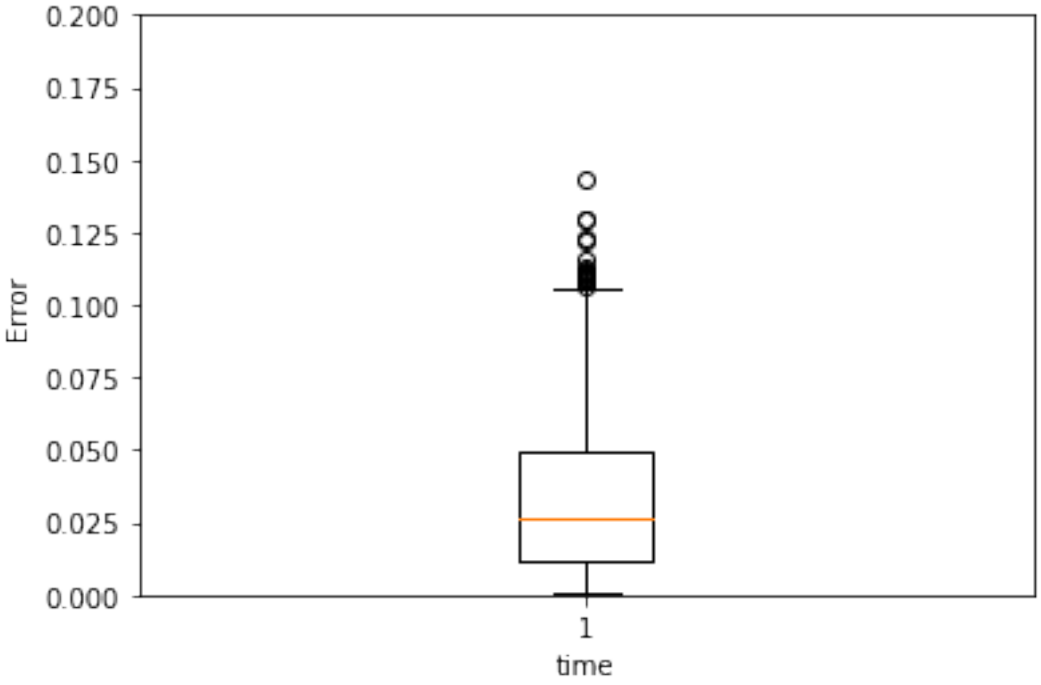


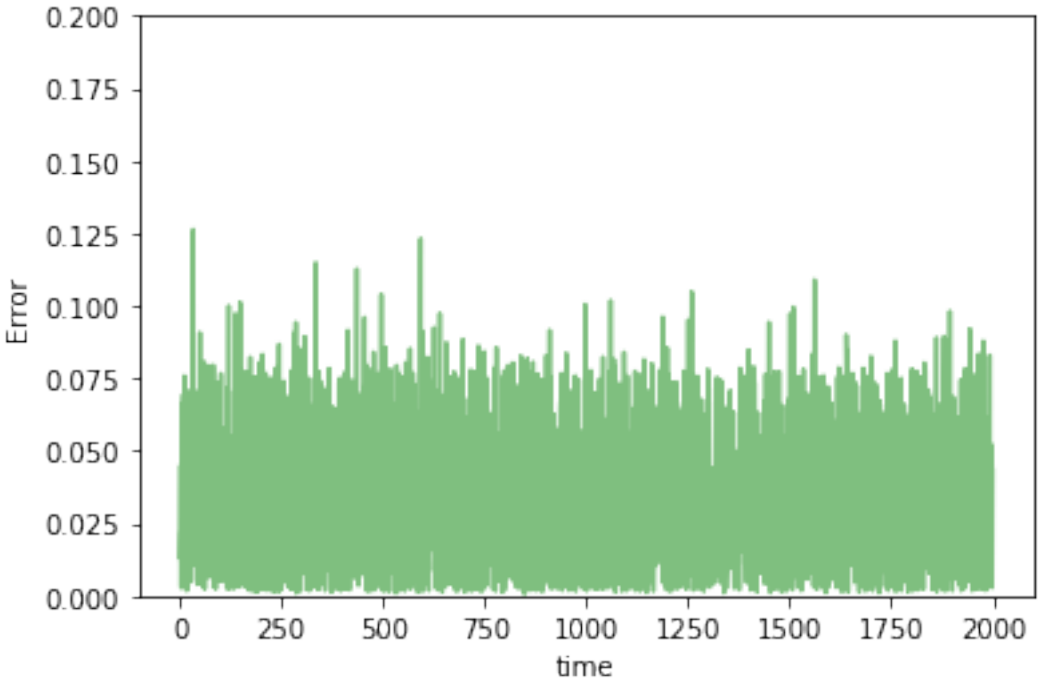

```
0.0116173778933
```

```
In [114]: test(model, test_X[0], name="nn3_2ano")
          test(model, test_X[2], name="nn3_2norm")
```

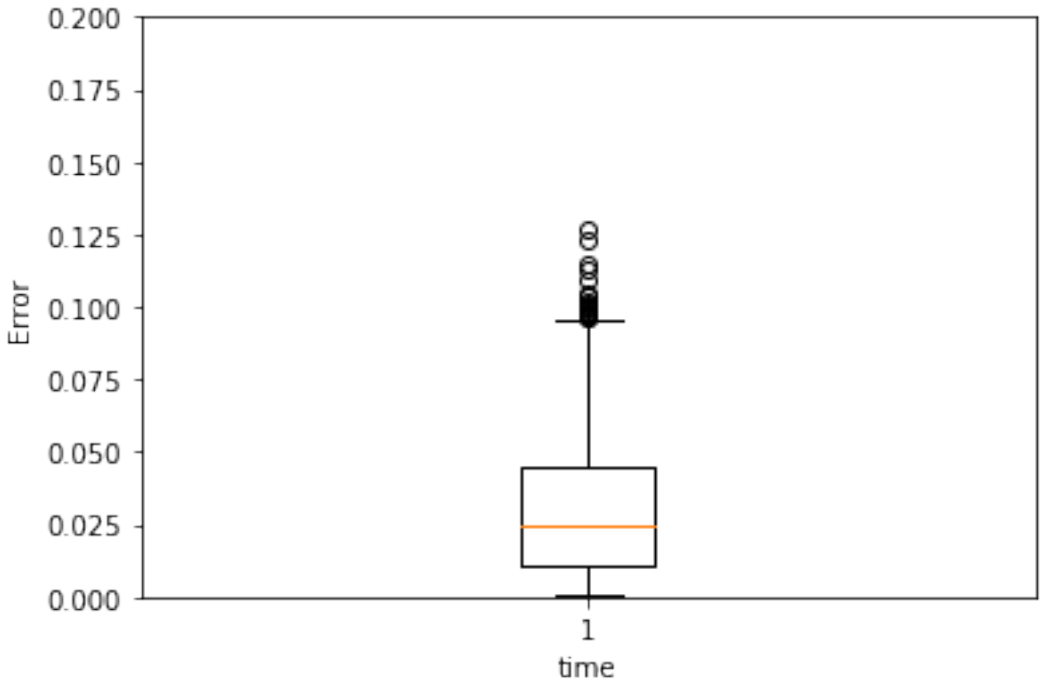


```
0.0332485561944
```



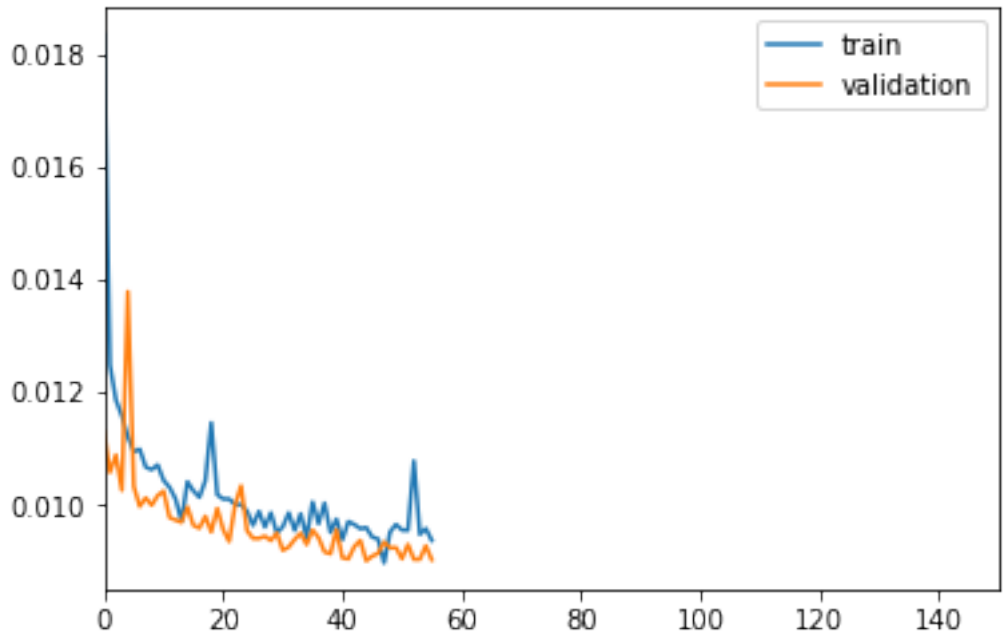


```
0.0309278875577
```



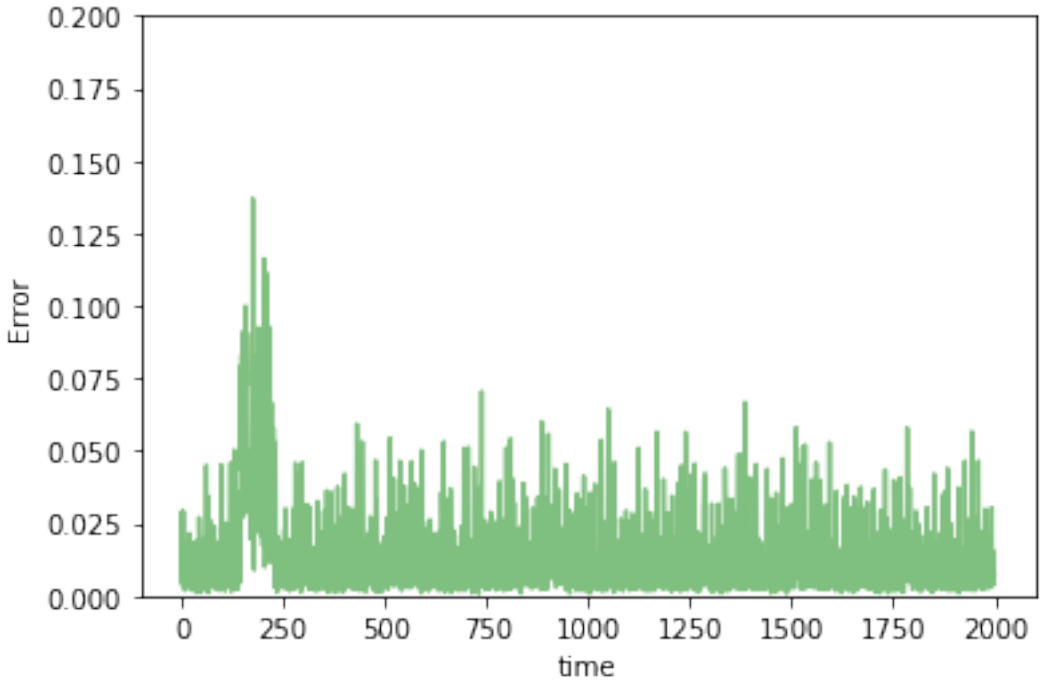
5 steps

```
In [115]: TIMESTEPS = 5
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [116]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(1000, activation='relu')(input_layer)
          hidden = Dense(500, activation='relu')(hidden)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [117]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [118]: train(model, tgen, vgen, name="nn3_5")
```

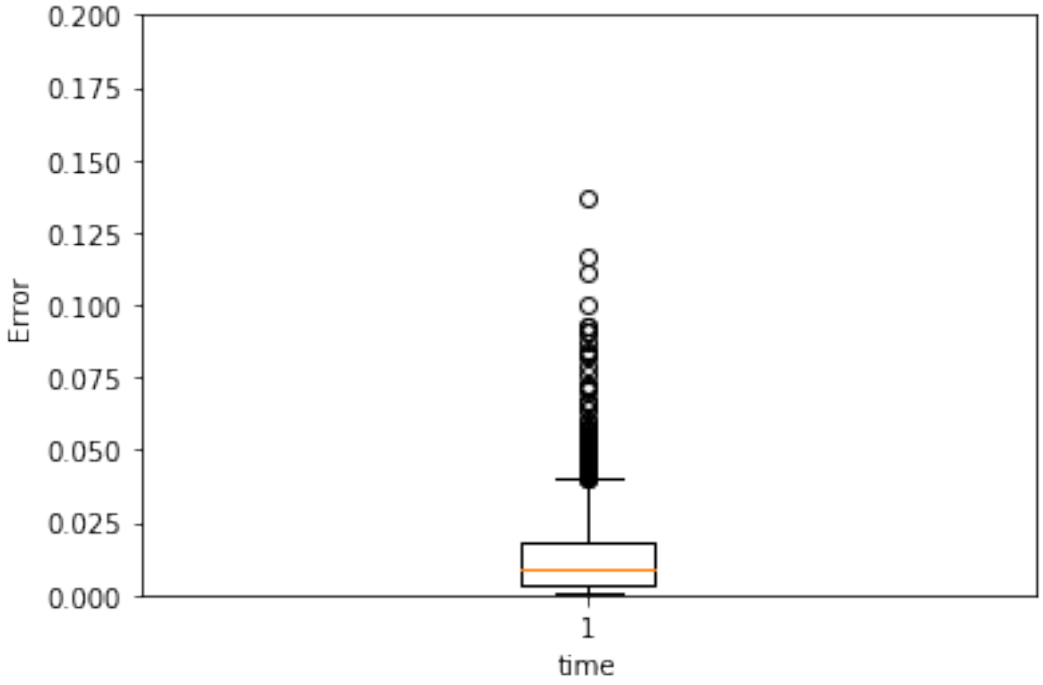


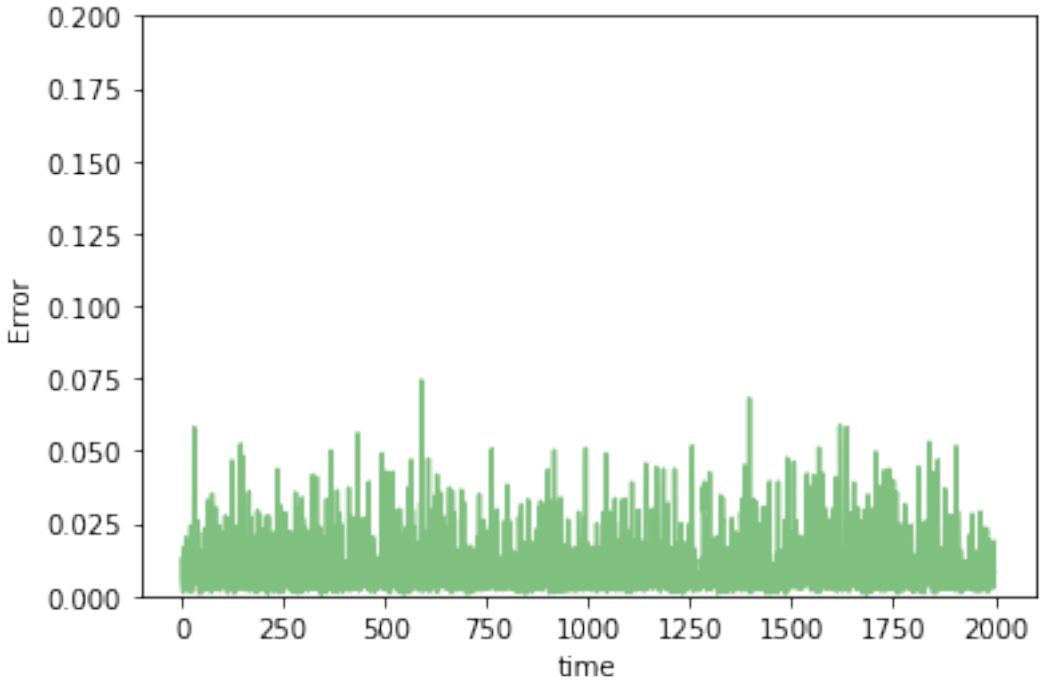

```
0.00936750763759
```

```
In [119]: test(model, test_X[0], name="nn3_5ano")
          test(model, test_X[2], name="nn3_5norm")
```

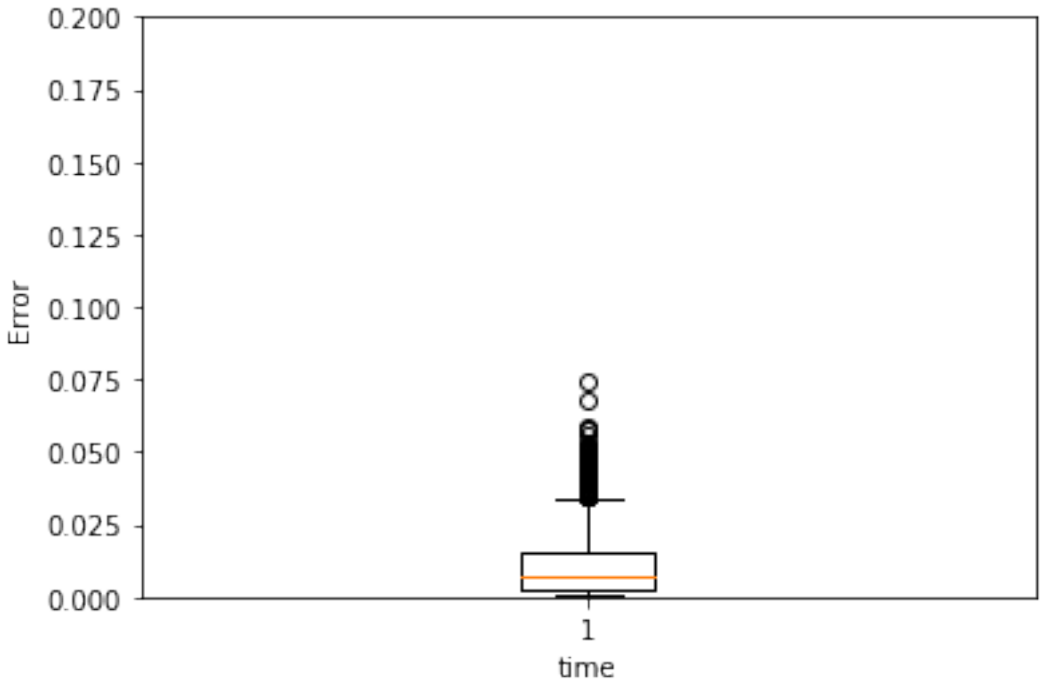


```
0.0134928065881
```



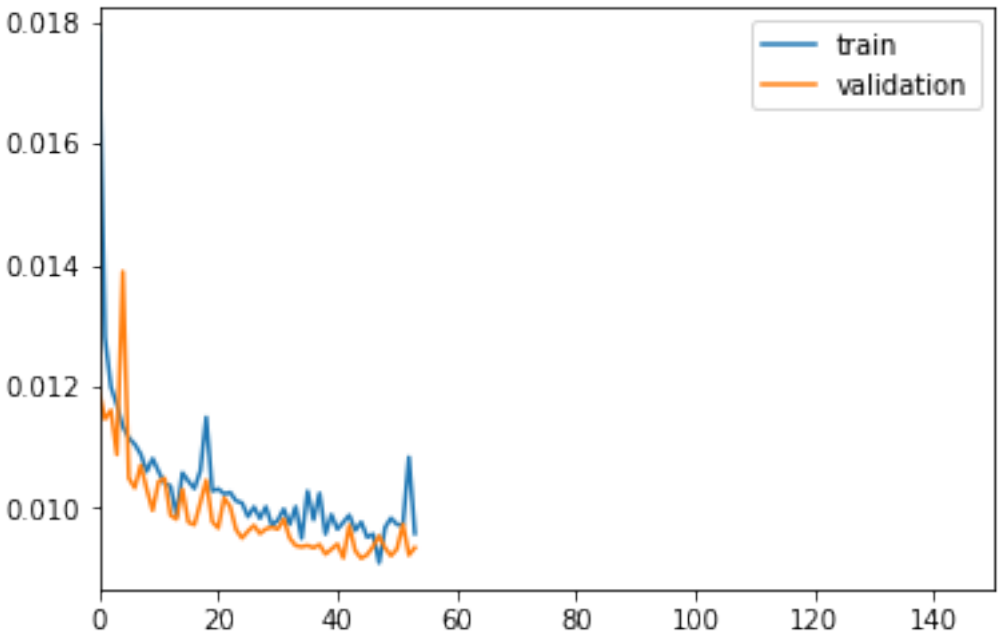


```
0.0106781493133
```



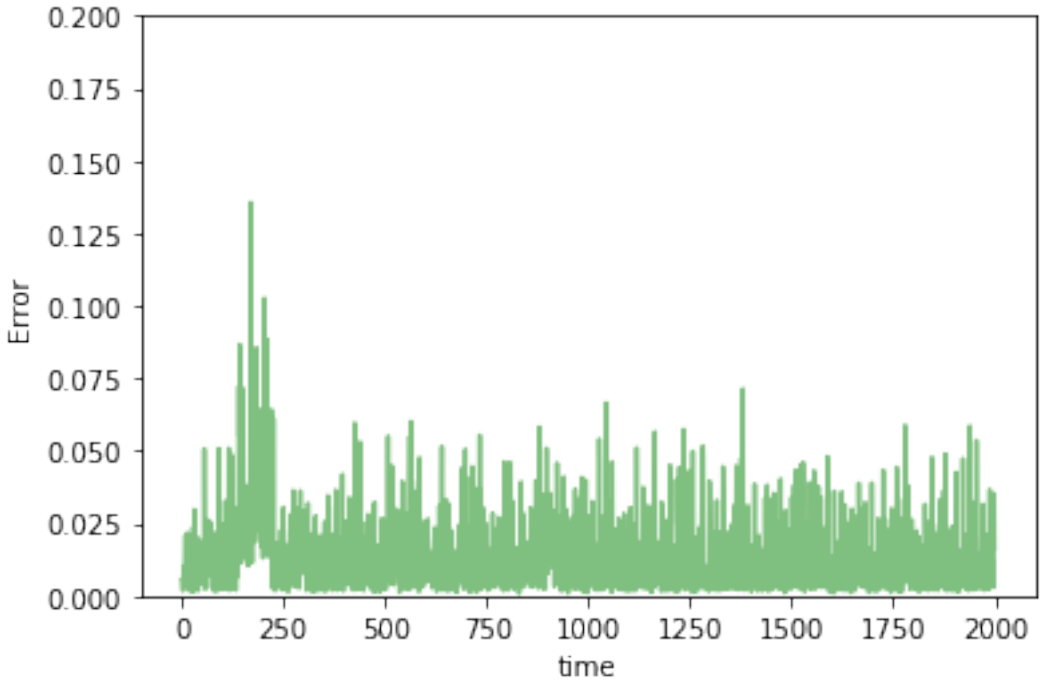
10 steps

```
In [120]: TIMESTEPS = 10
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [121]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(1000, activation='relu')(input_layer)
          hidden = Dense(500, activation='relu')(hidden)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [122]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [123]: train(model, tgen, vgen, name="nn3_10")
```

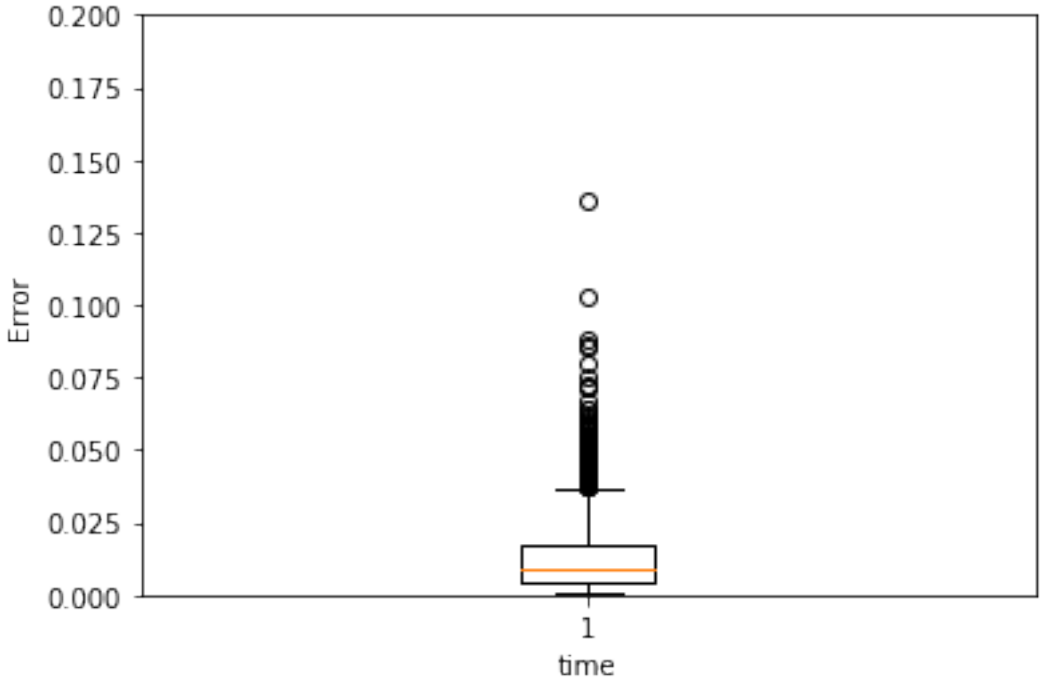


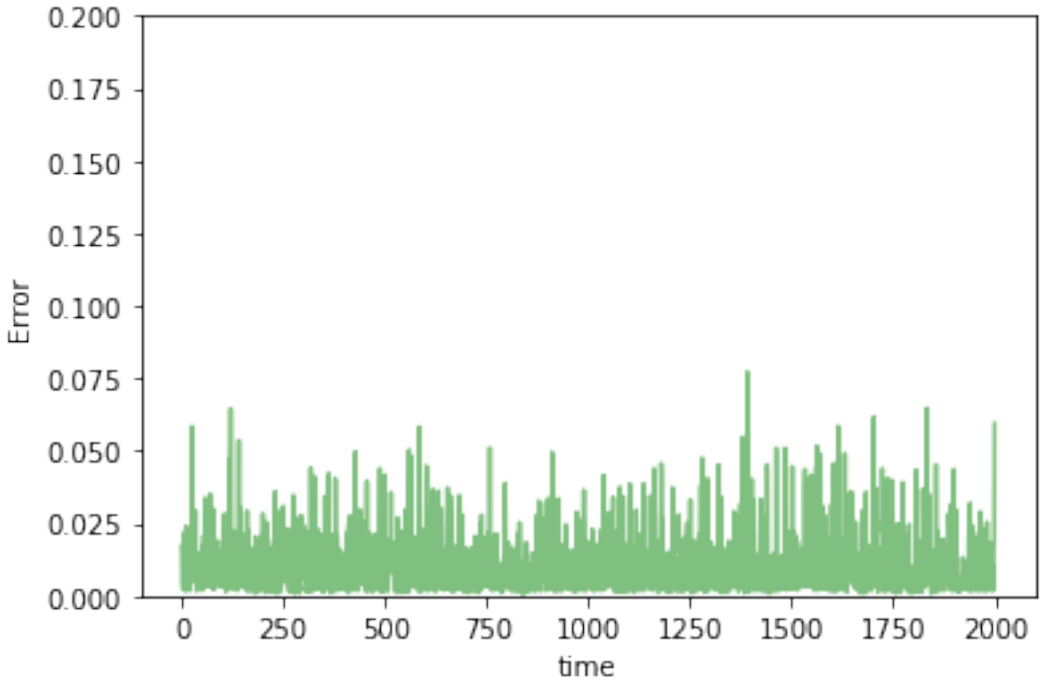

```
0.00957909464161
```

```
In [124]: test(model, test_X[0], name="nn3_10ano")
          test(model, test_X[2], name="nn3_10norm")
```

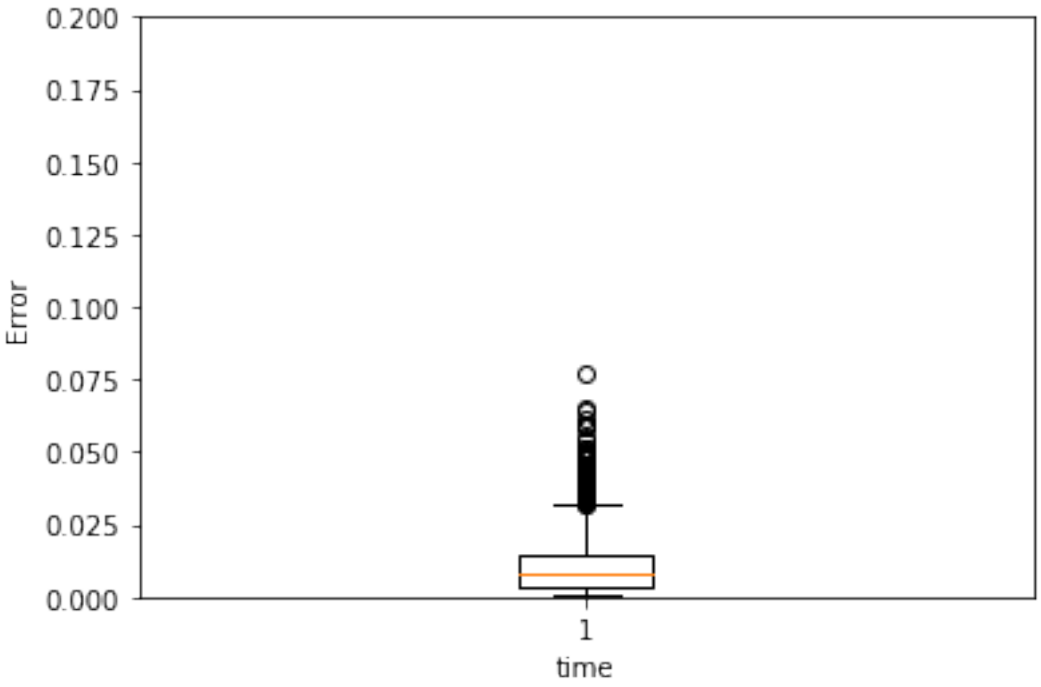


```
0.01330958826
```



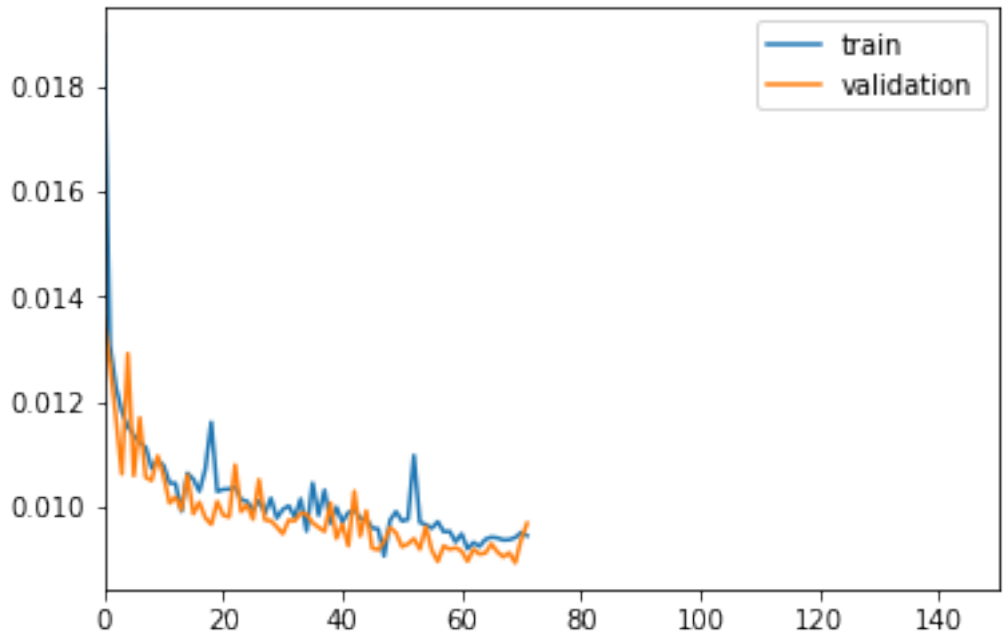


```
0.0108460719314
```



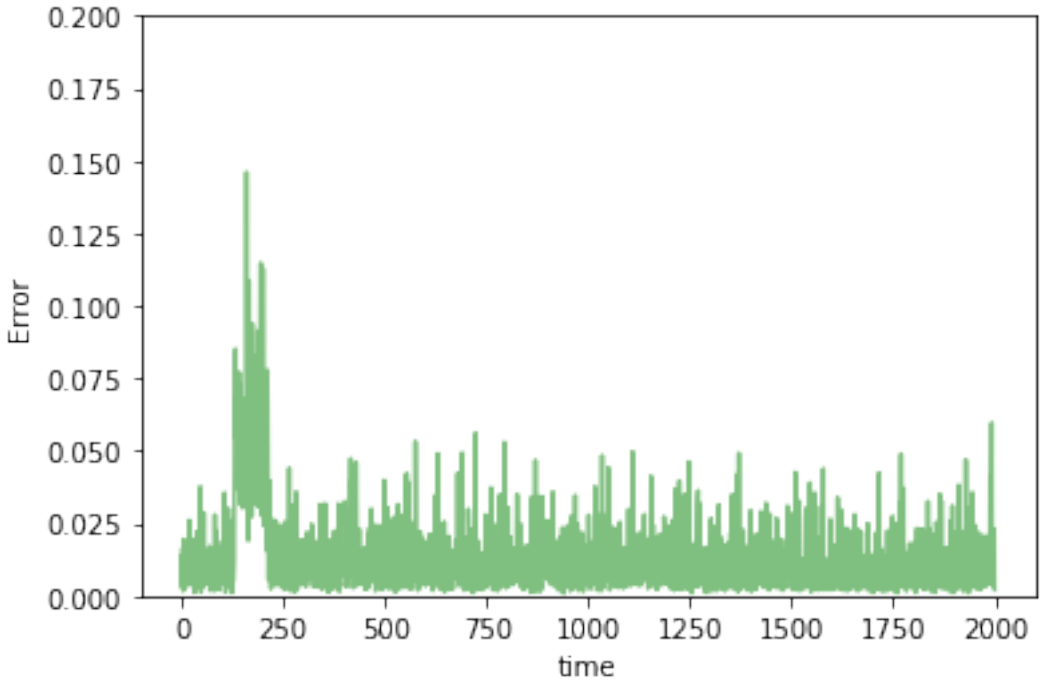
20 steps

```
In [125]: TIMESTEPS = 20
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [126]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(1000, activation='relu')(input_layer)
          hidden = Dense(500, activation='relu')(hidden)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [127]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [128]: train(model, tgen, vgen, name="nn3_20")
```

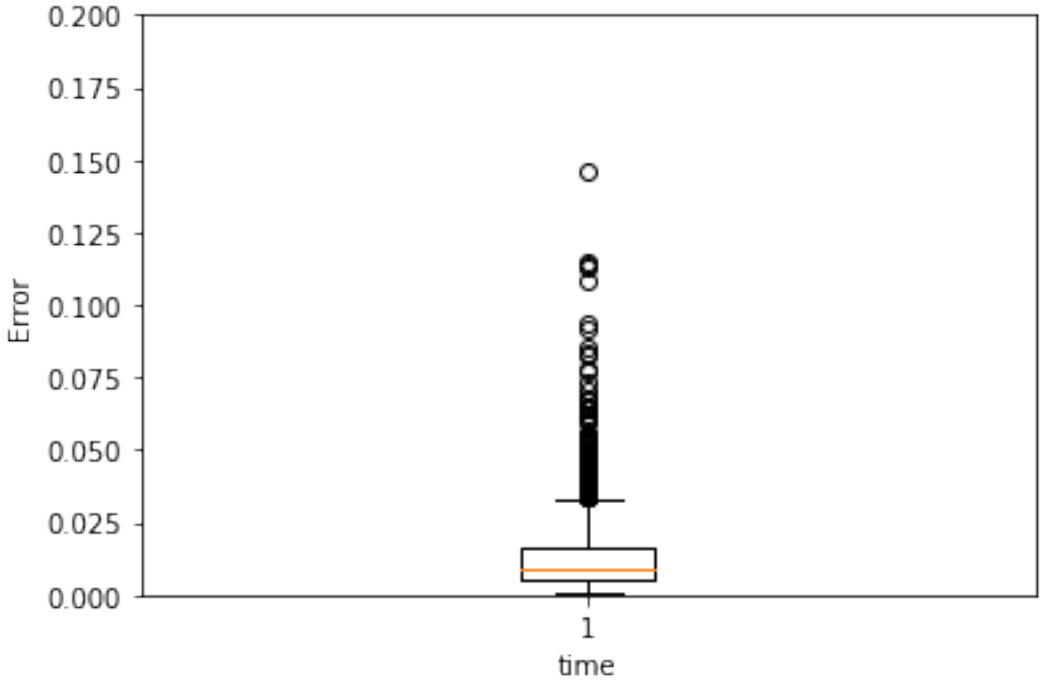


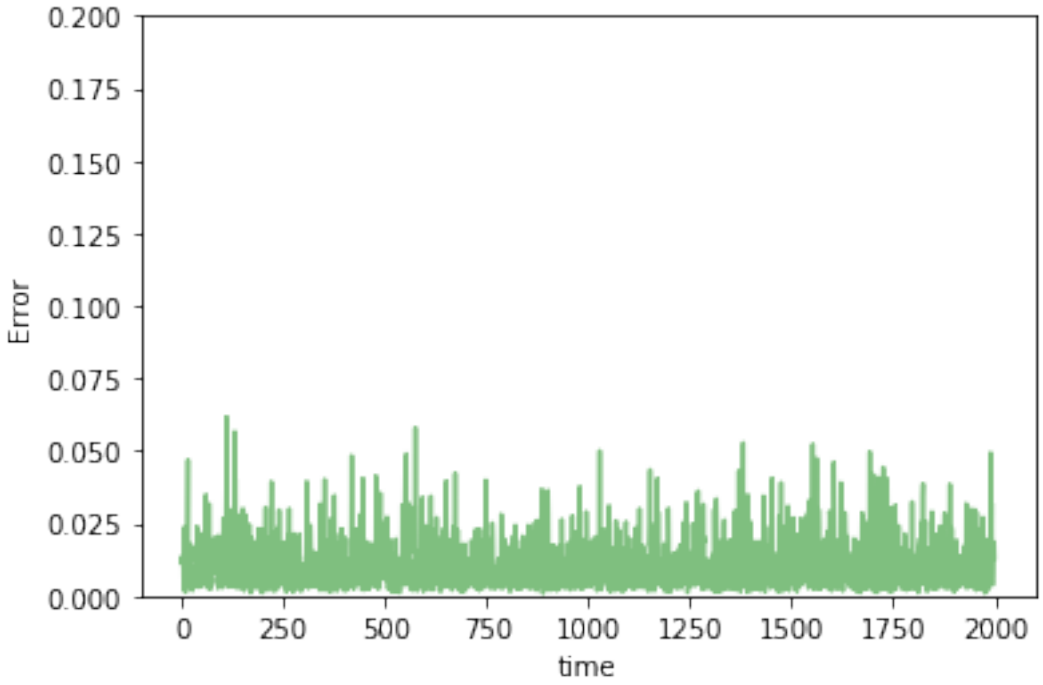

```
0.00945406853361
```

```
In [129]: test(model, test_X[0], name="nn3_20ano")
          test(model, test_X[2], name="nn3_20norm")
```

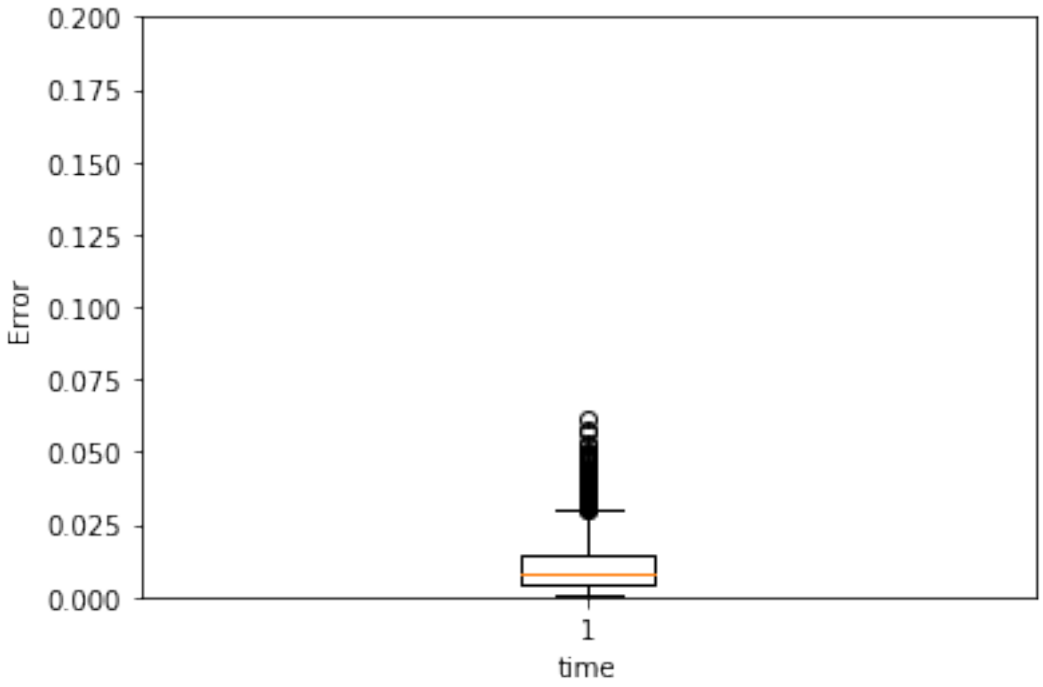


```
0.0131684896884
```



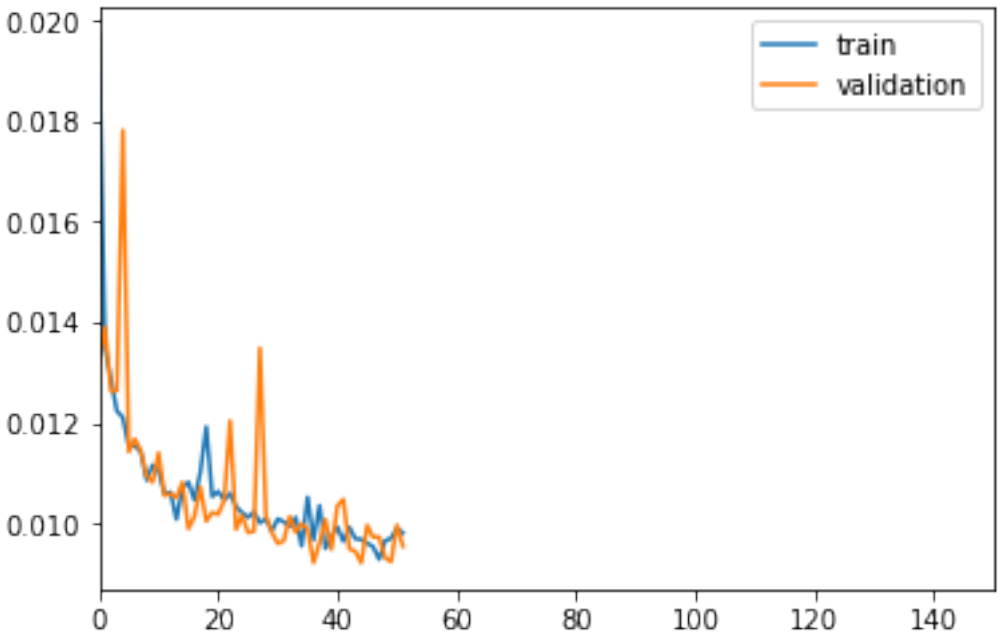


```
0.0108368867922
```



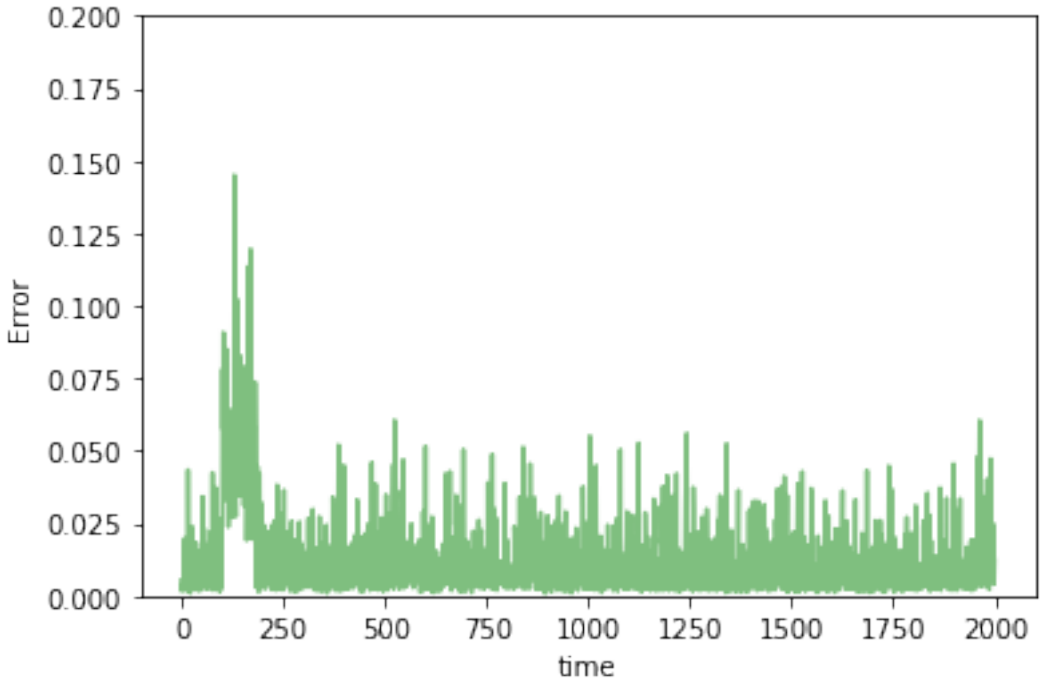
50 steps

```
In [130]: TIMESTEPS = 50
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS)
          vgen = flat_generator(val_X, TIMESTEPS)

In [131]: input_layer = Input(shape=(TIMESTEPS*DIM,))
          hidden = Dense(1000, activation='relu')(input_layer)
          hidden = Dense(500, activation='relu')(hidden)
          hidden = Dense(100, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [132]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [133]: train(model, tgen, vgen, name="nn3_50")
```

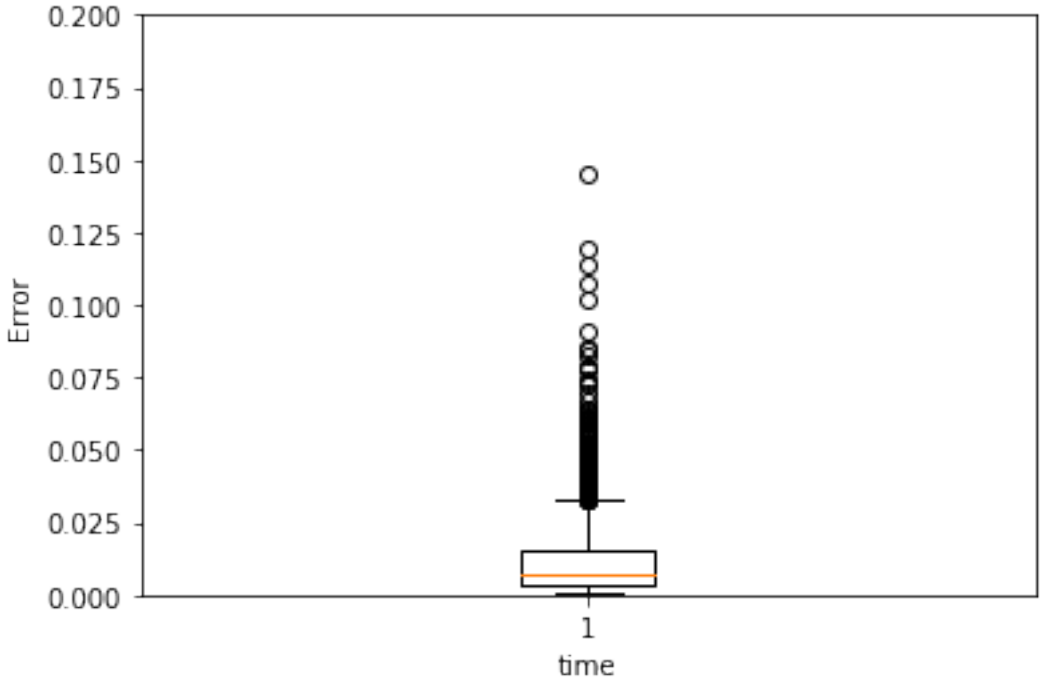


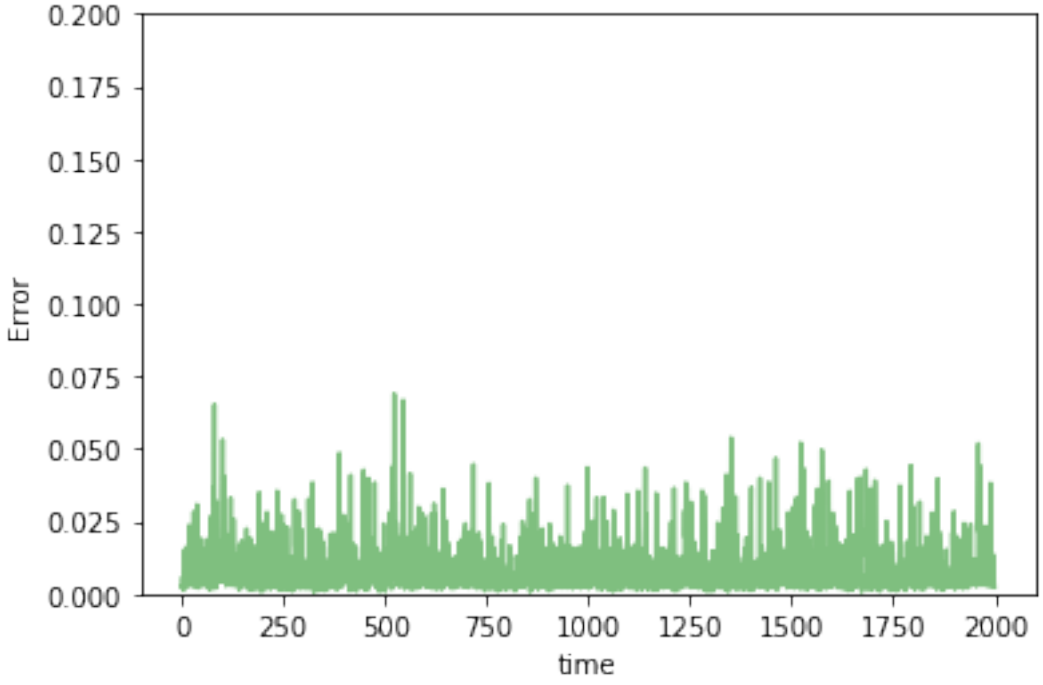

```
0.00981520255597
```

```
In [134]: test(model, test_X[0], name="nn3_50ano")
          test(model, test_X[2], name="nn3_50norm")
```

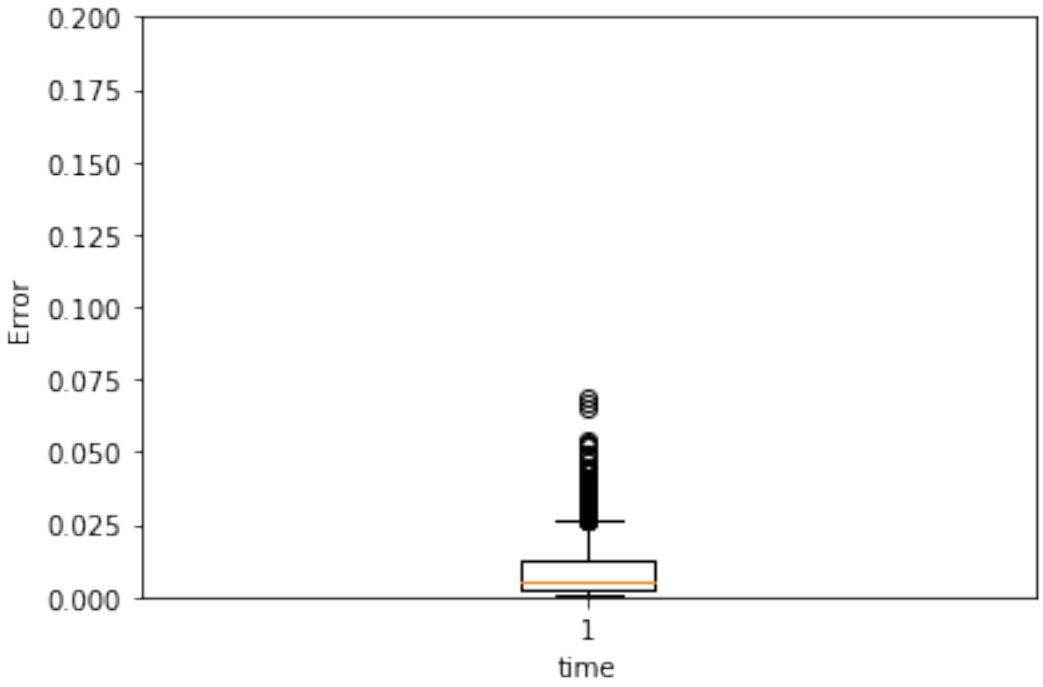


```
0.0118748299071
```





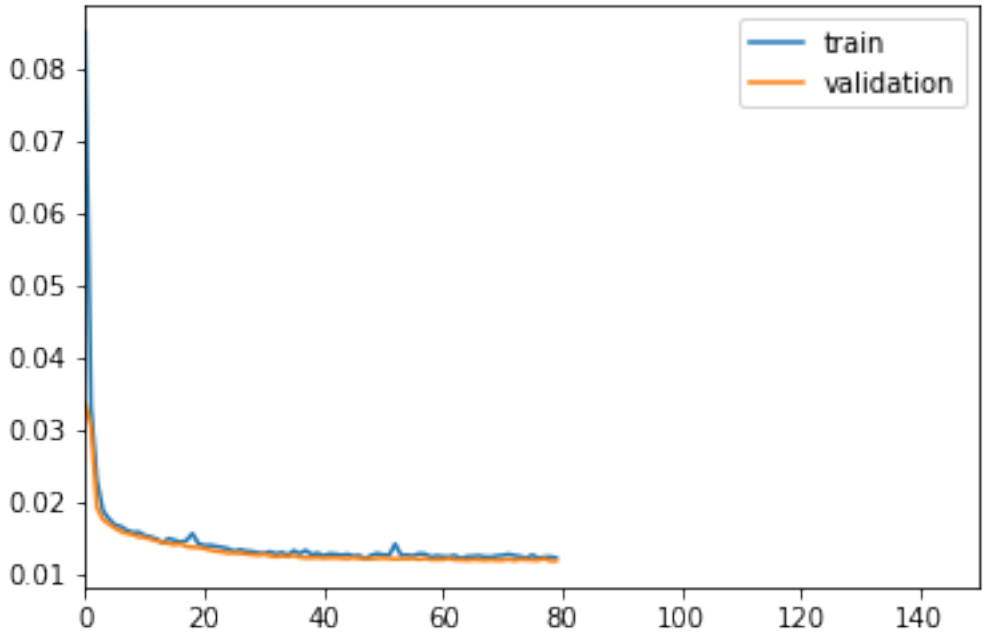
```
0.00902413064448
```



2.1.5 RNN with 1 GRU layers

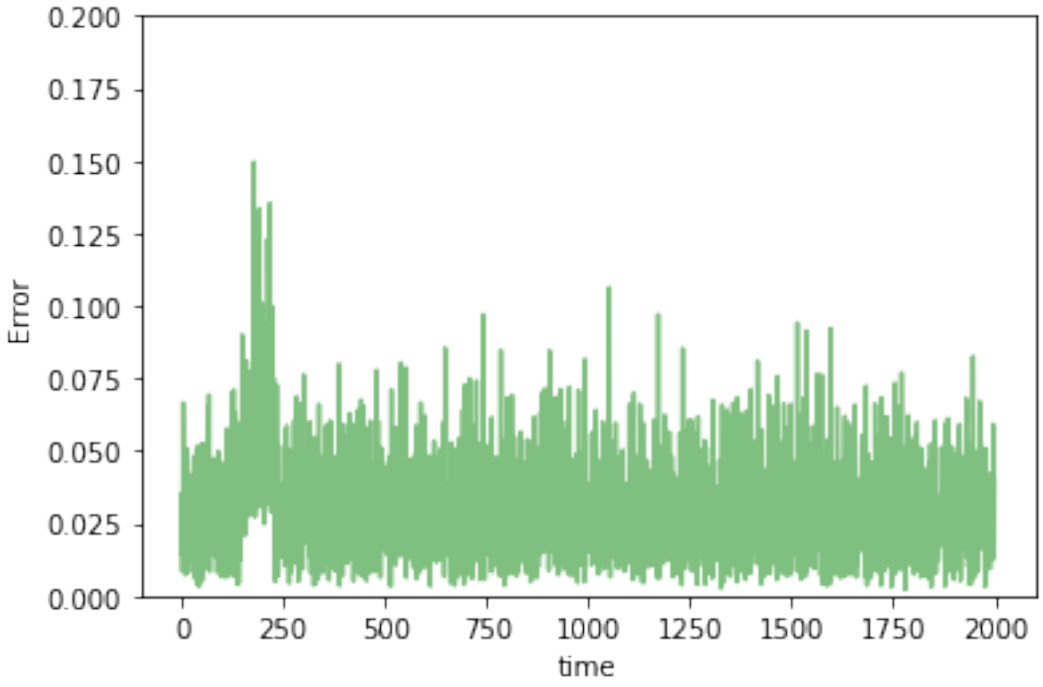
2 steps

```
In [135]: TIMESTEPS = 2
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [136]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu')(input_layer)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [137]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [138]: train(model, tgen, vgen, name="gru1_2")
```

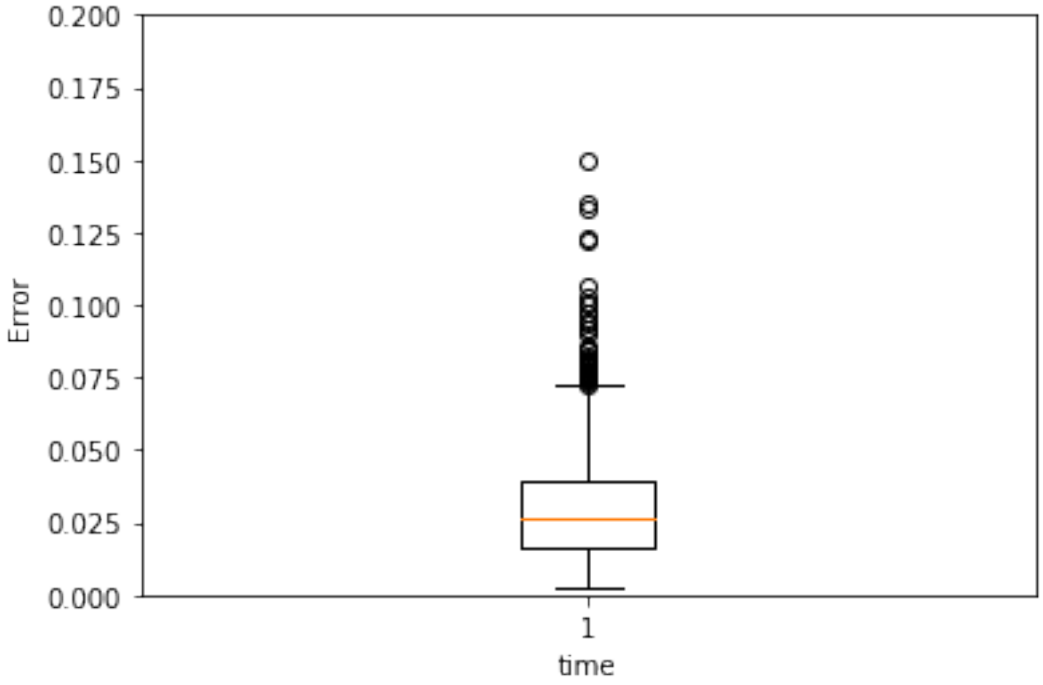


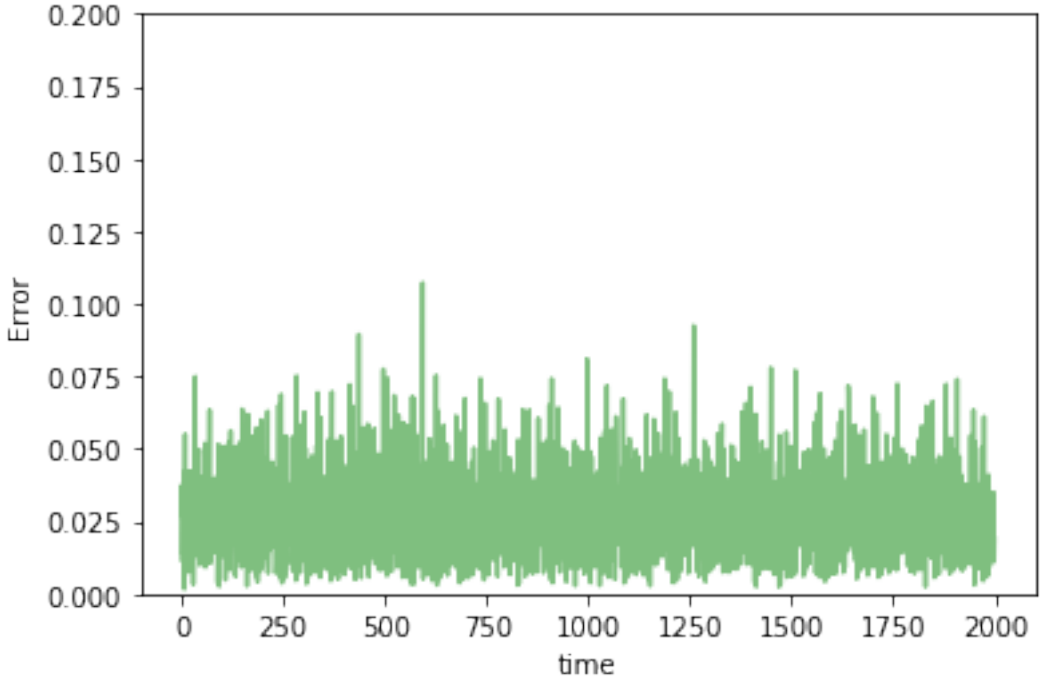

```
0.0122319026904
```

```
In [139]: test(model, test_X[0],0, name="gru1_2ano")
          test(model, test_X[2],0, name="gru1_2norm")
```

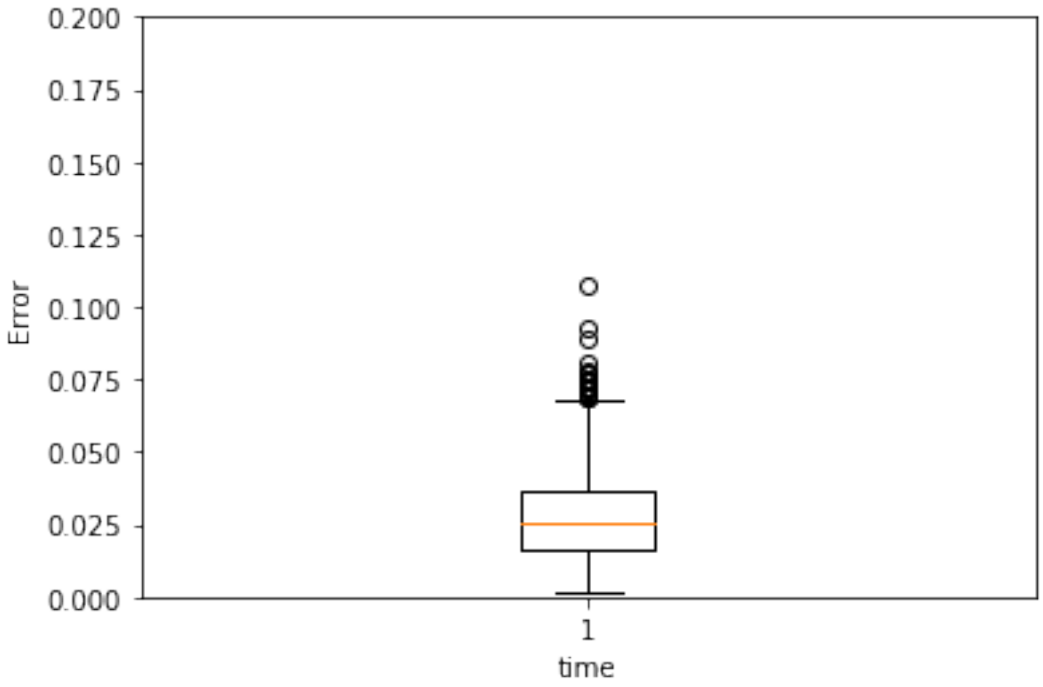


```
0.0297641205284
```



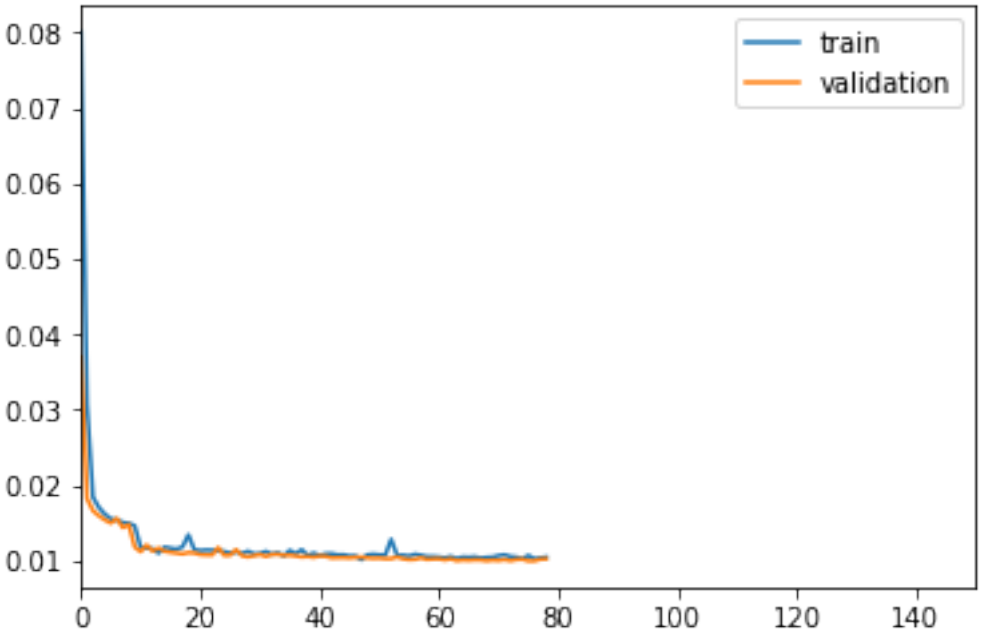


```
0.0273371679114
```



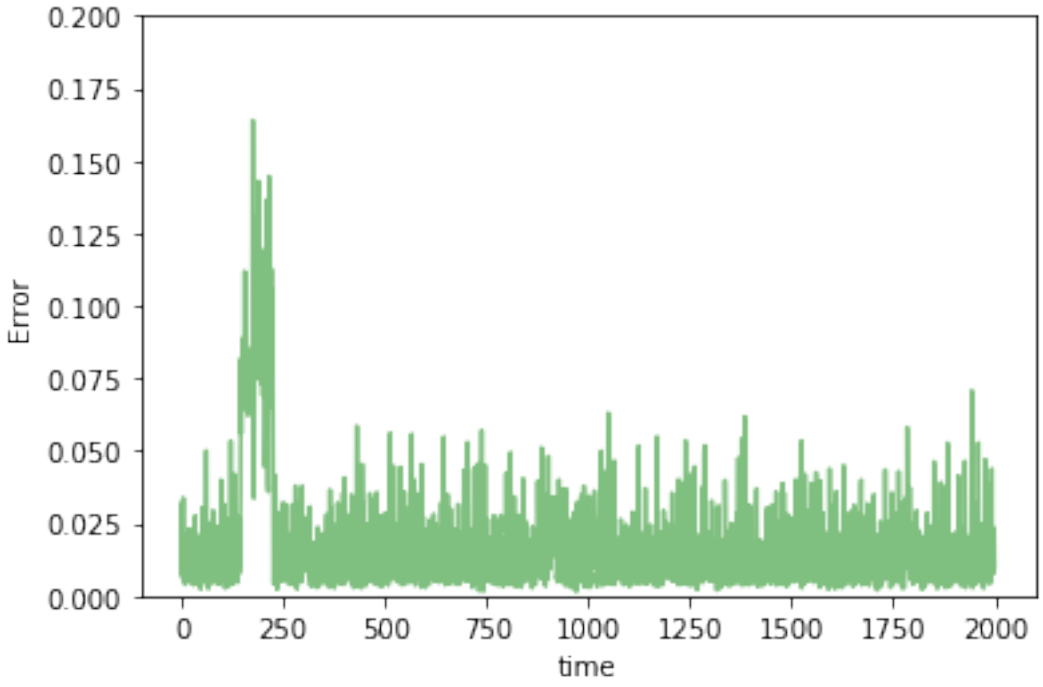
5 steps

```
In [140]: TIMESTEPS = 5
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [141]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu')(input_layer)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [142]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [143]: train(model, tgen, vgen, name="gru1_5")
```

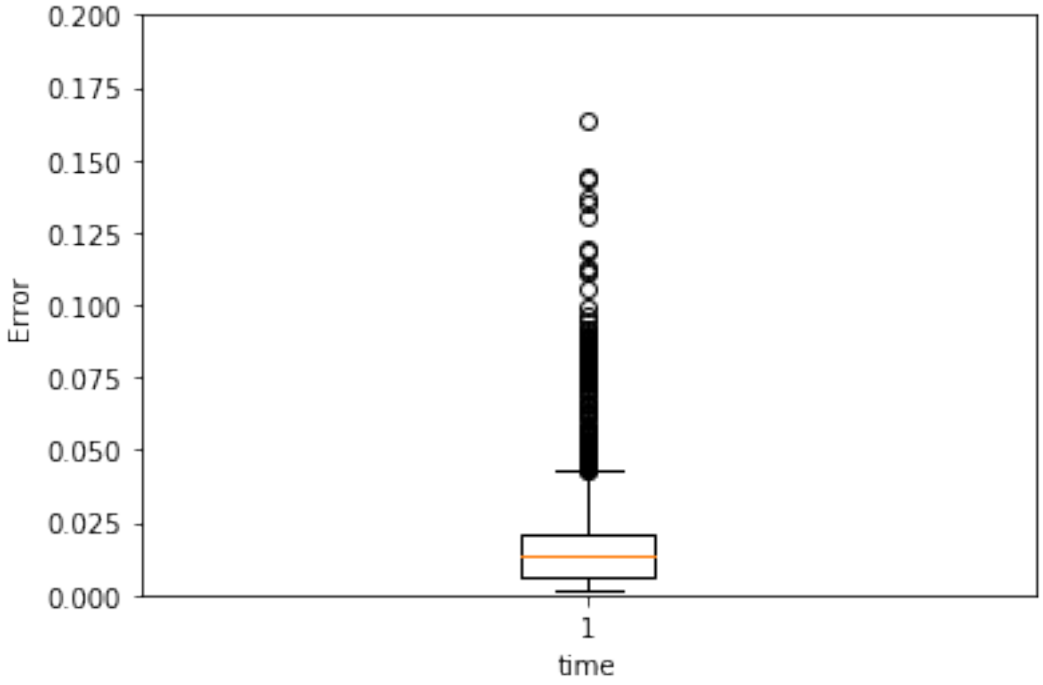


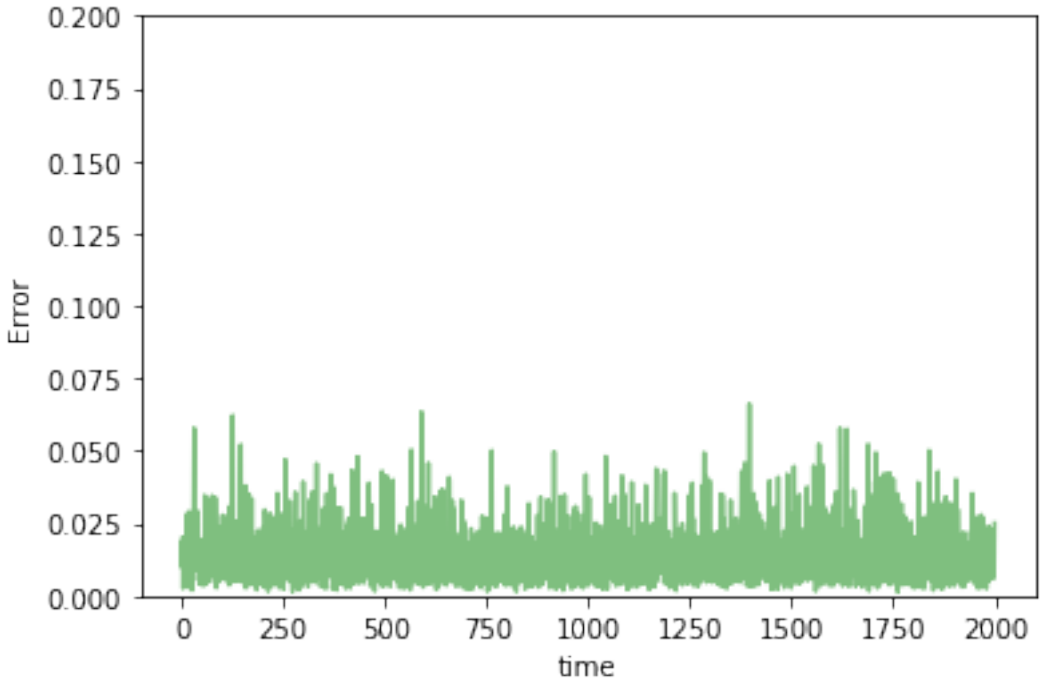

```
0.0104931976465
```

```
In [144]: test(model, test_X[0],0, name="gru1_5ano")
          test(model, test_X[2],0, name="gru1_5norm")
```

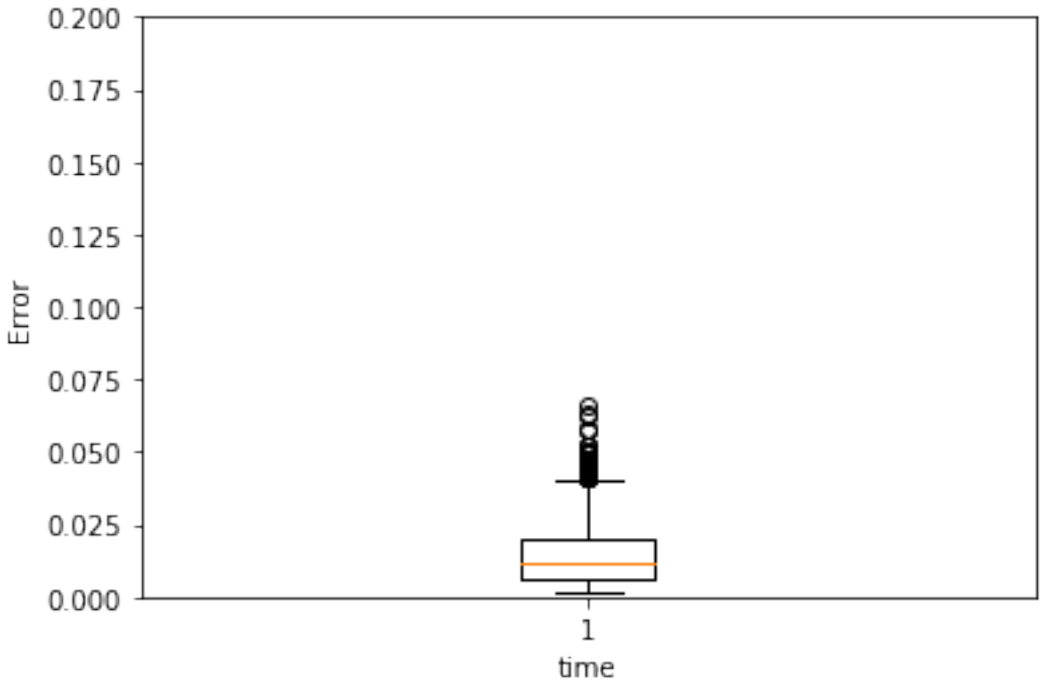


```
0.0177169879183
```



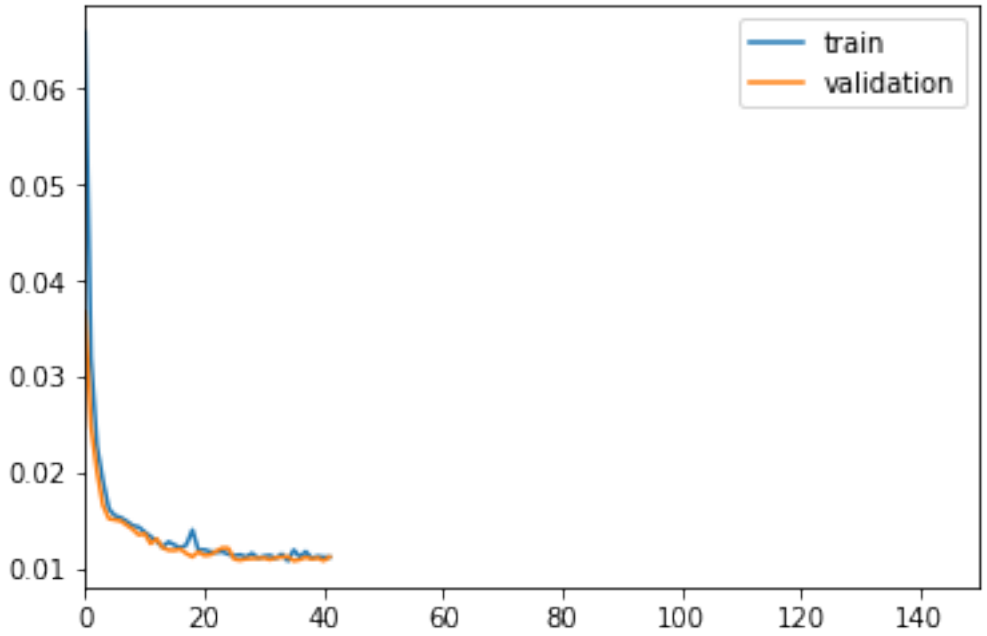


```
0.0142960346843
```



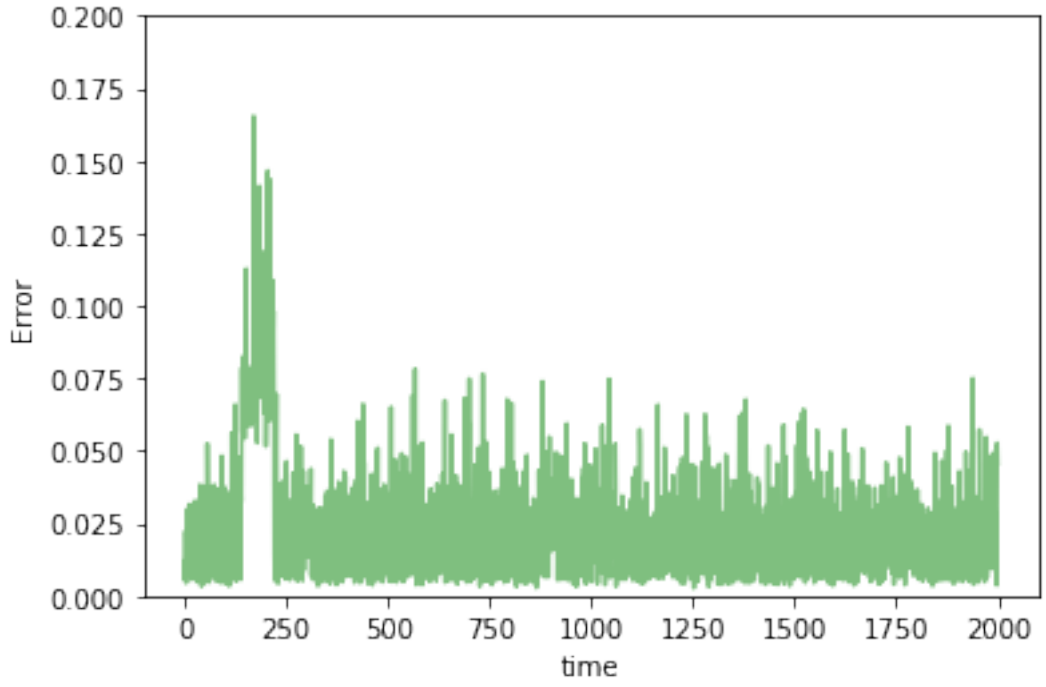
10 steps

```
In [145]: TIMESTEPS = 10
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS, 0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [146]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu')(input_layer)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [147]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [148]: train(model, tgen, vgen, name="gru1_10")
```

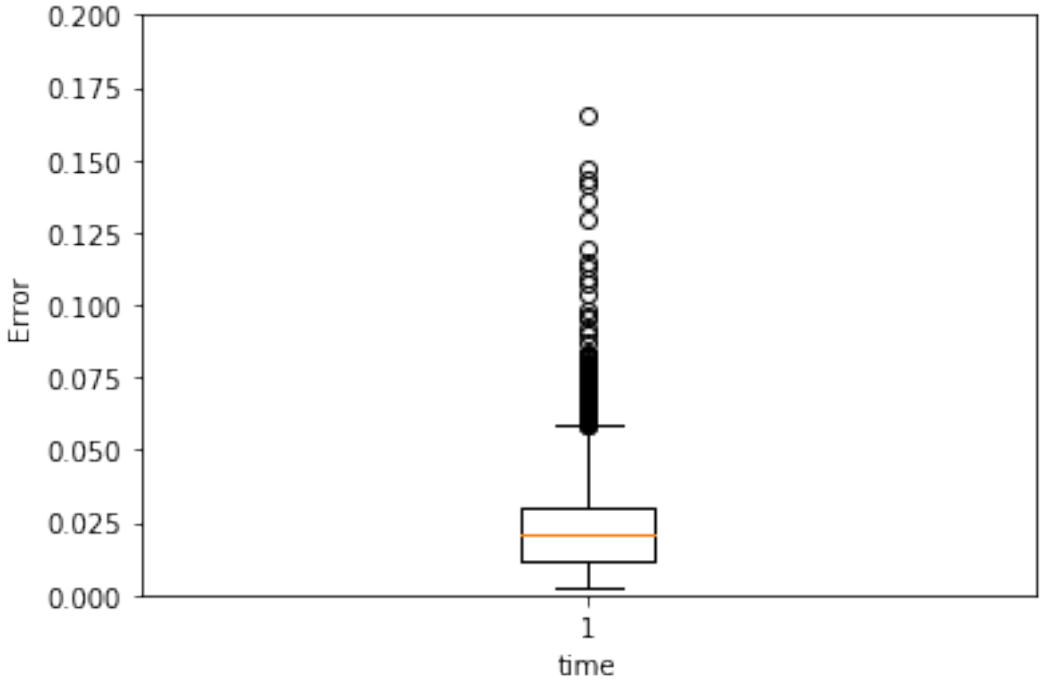


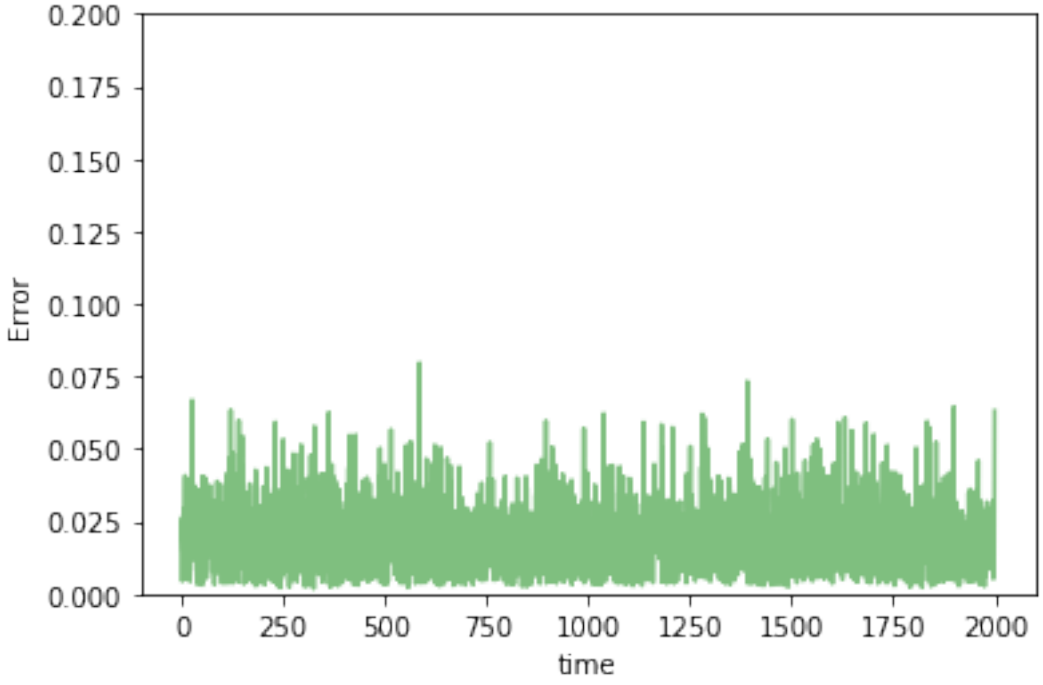

```
0.0112368315409
```

```
In [149]: test(model, test_X[0],0, name="gru1_10ano")
          test(model, test_X[2],0, name="gru1_10norm")
```

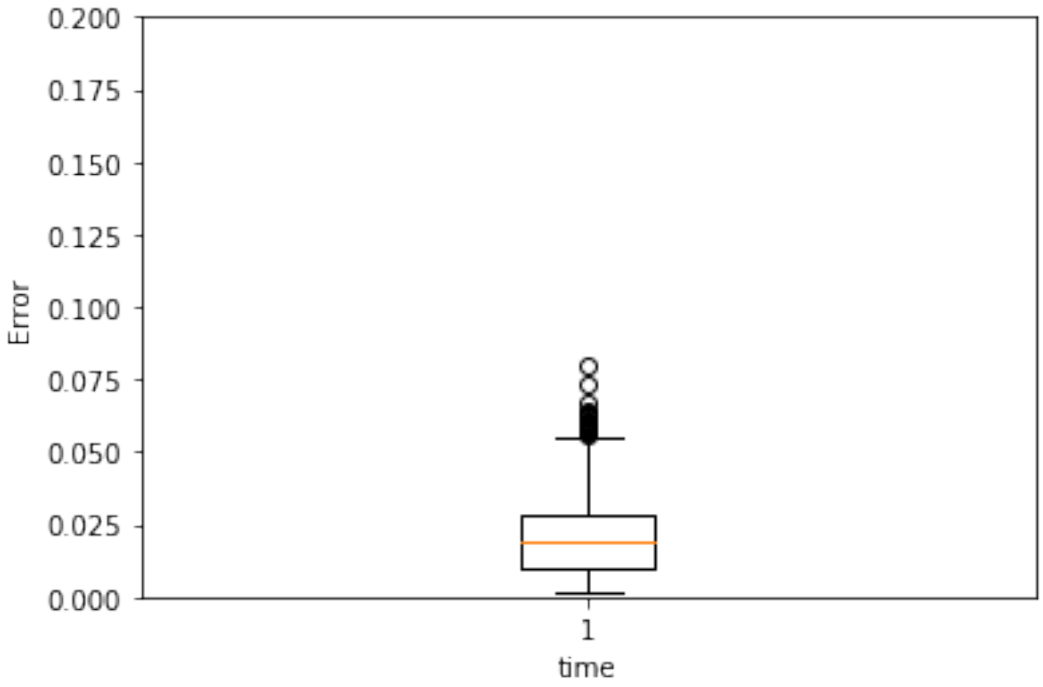


```
0.0240515997829
```



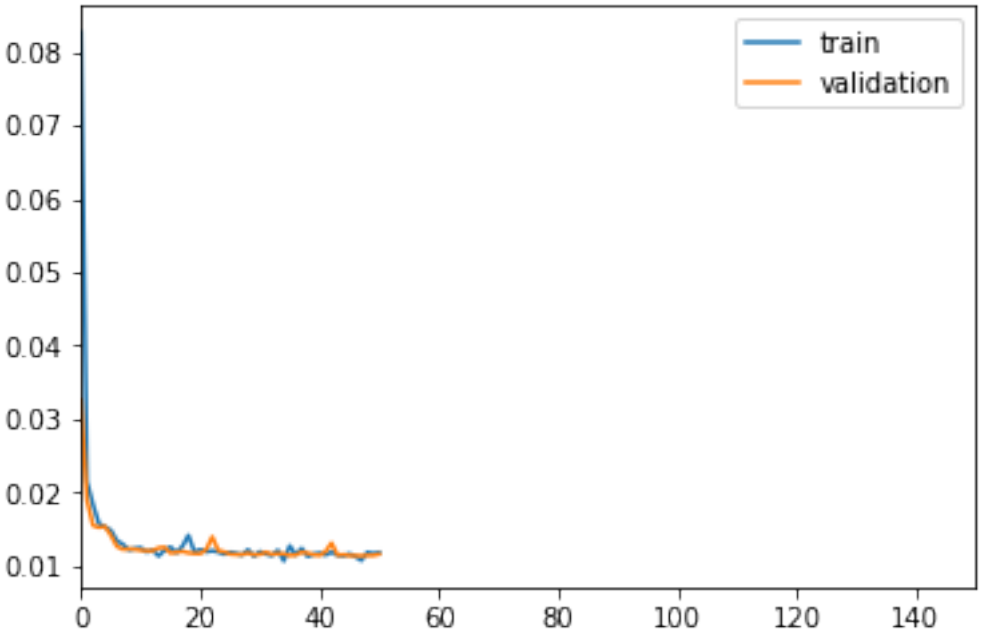


```
0.0203914175081
```



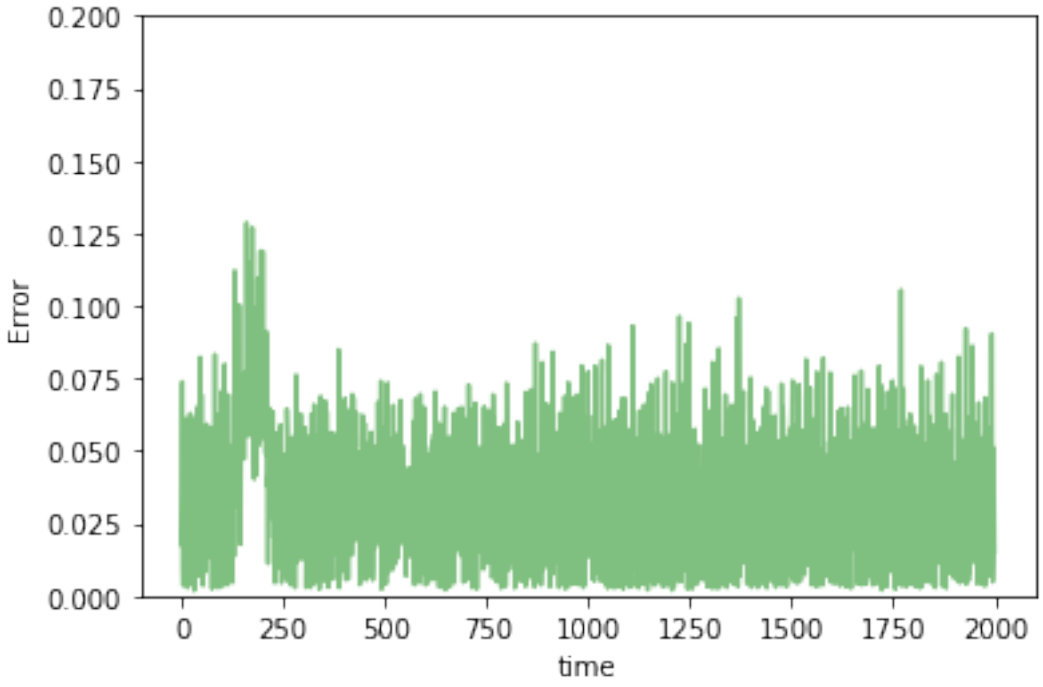
20 steps

```
In [150]: TIMESTEPS = 20
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [151]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu')(input_layer)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [152]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [153]: train(model, tgen, vgen, name="gru1_20")
```

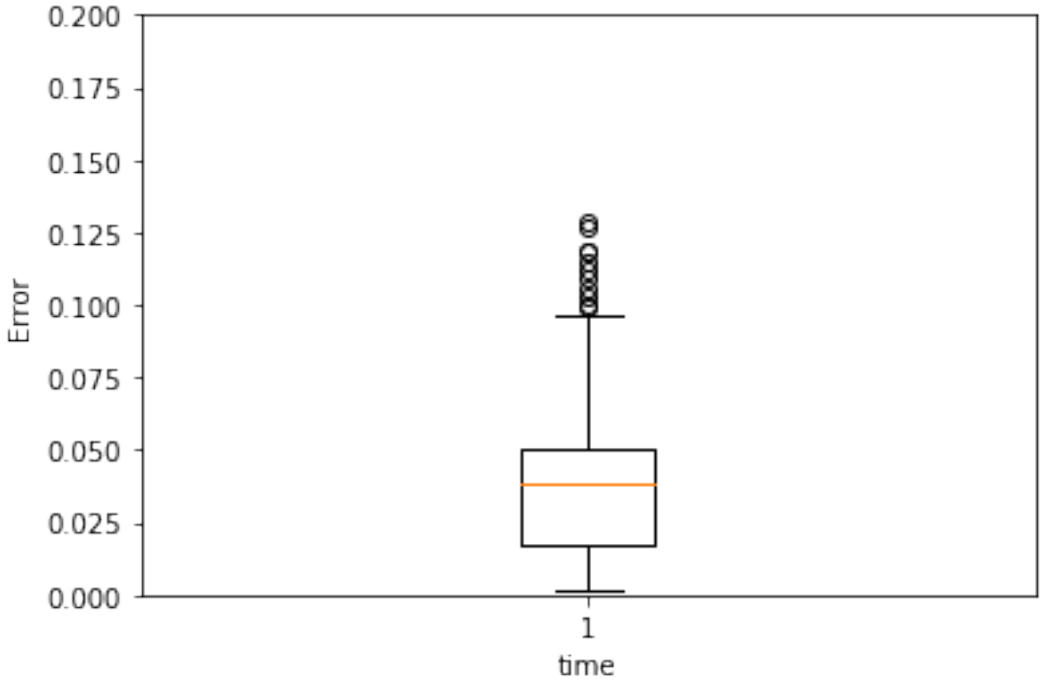


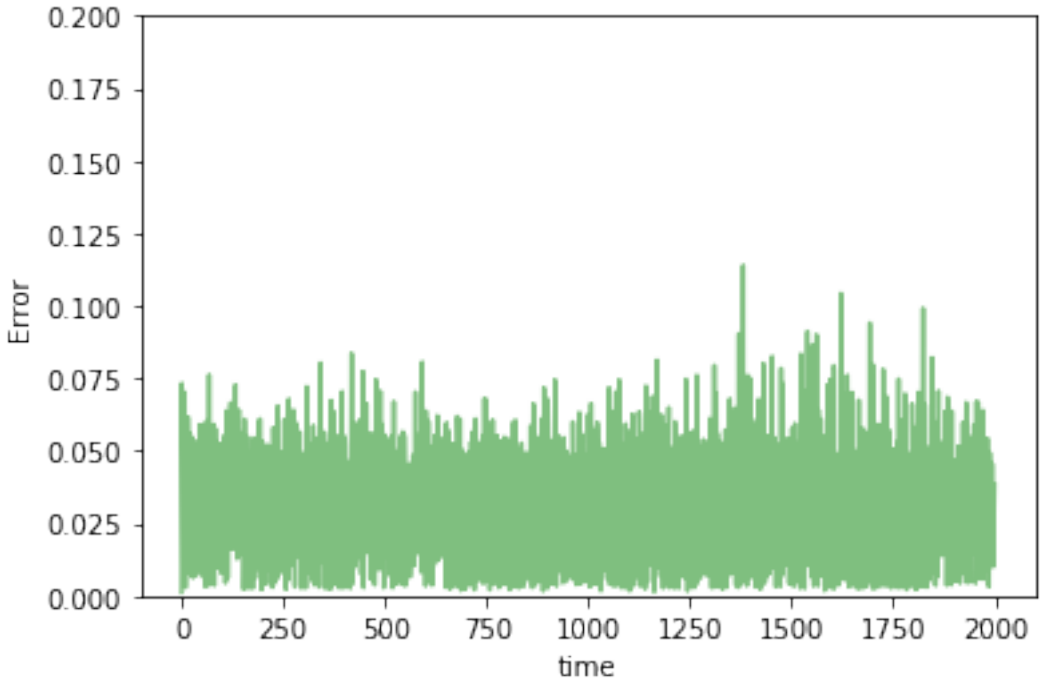

```
0.0117418383087
```

```
In [154]: test(model, test_X[0],0, name="gru1_20ano")
          test(model, test_X[2],0, name="gru1_20norm")
```

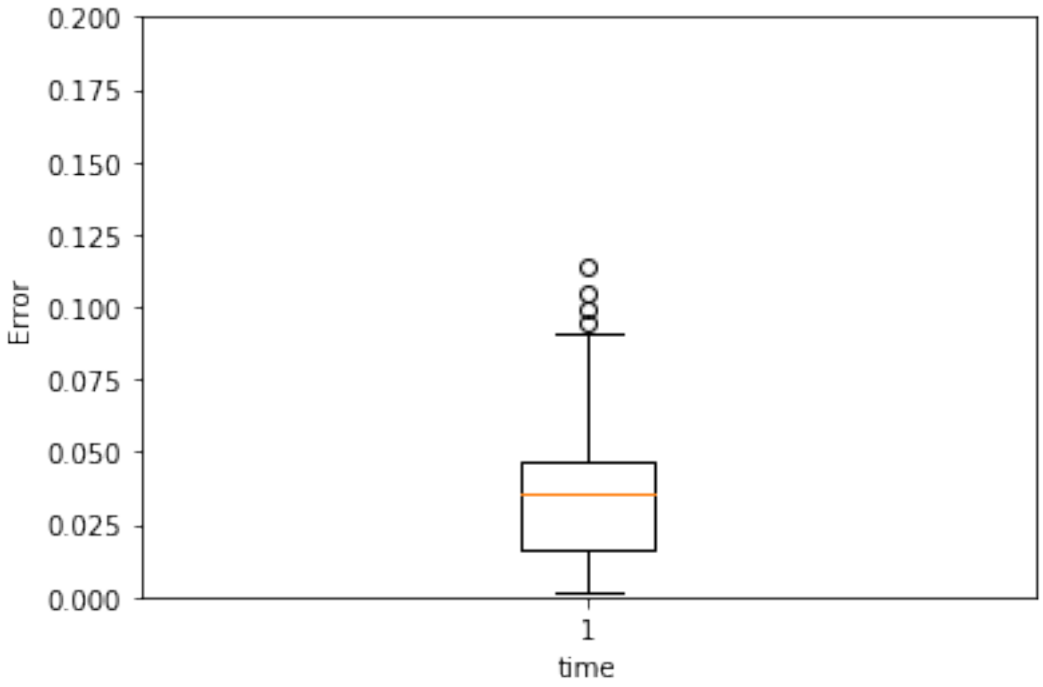


```
0.0360399631518
```



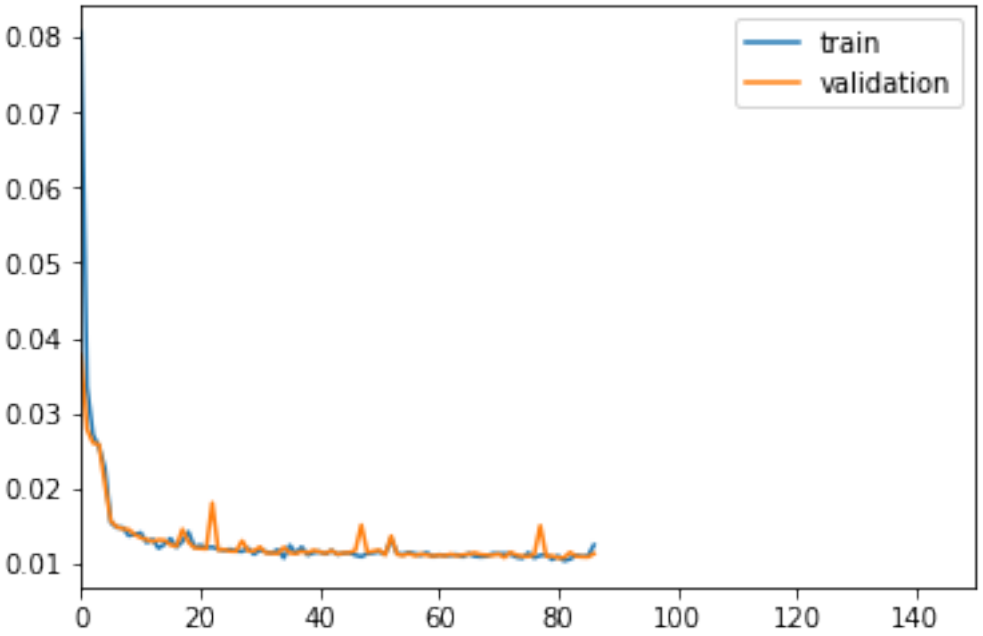


```
0.0330399261152
```



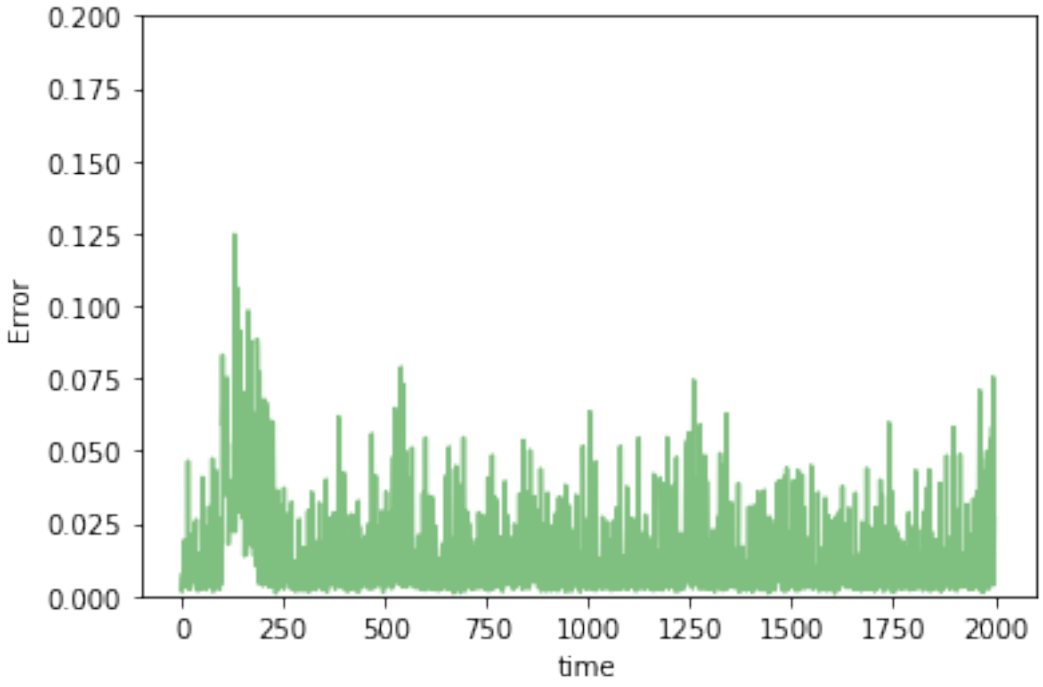
50 steps

```
In [155]: TIMESTEPS = 50
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [156]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu')(input_layer)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [157]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [158]: train(model, tgen, vgen, name="gru1_50")
```

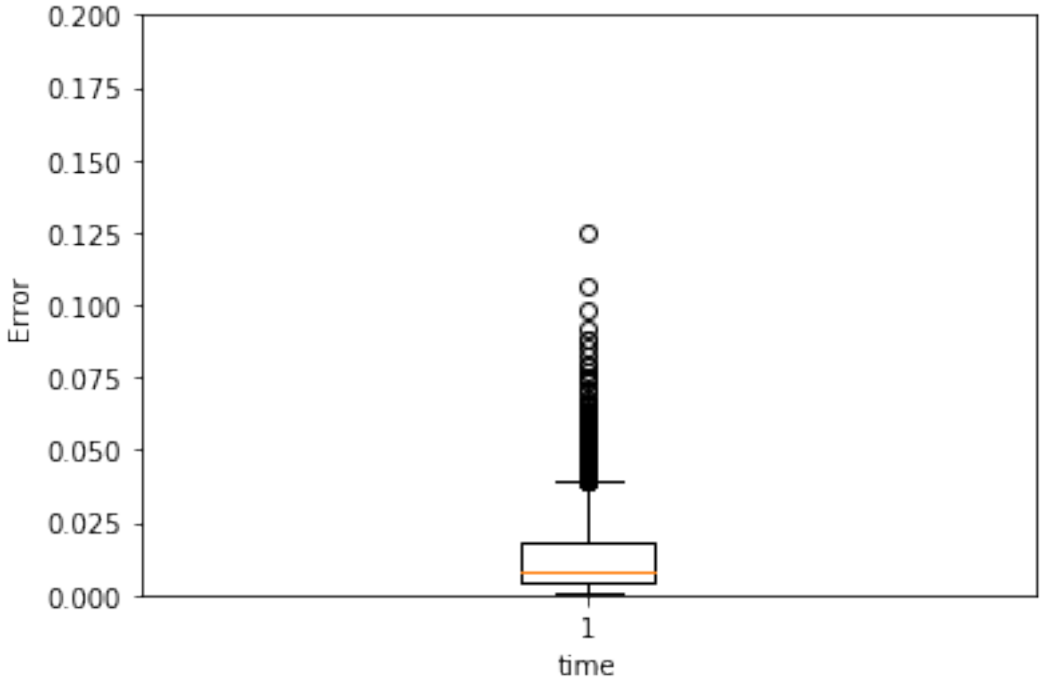


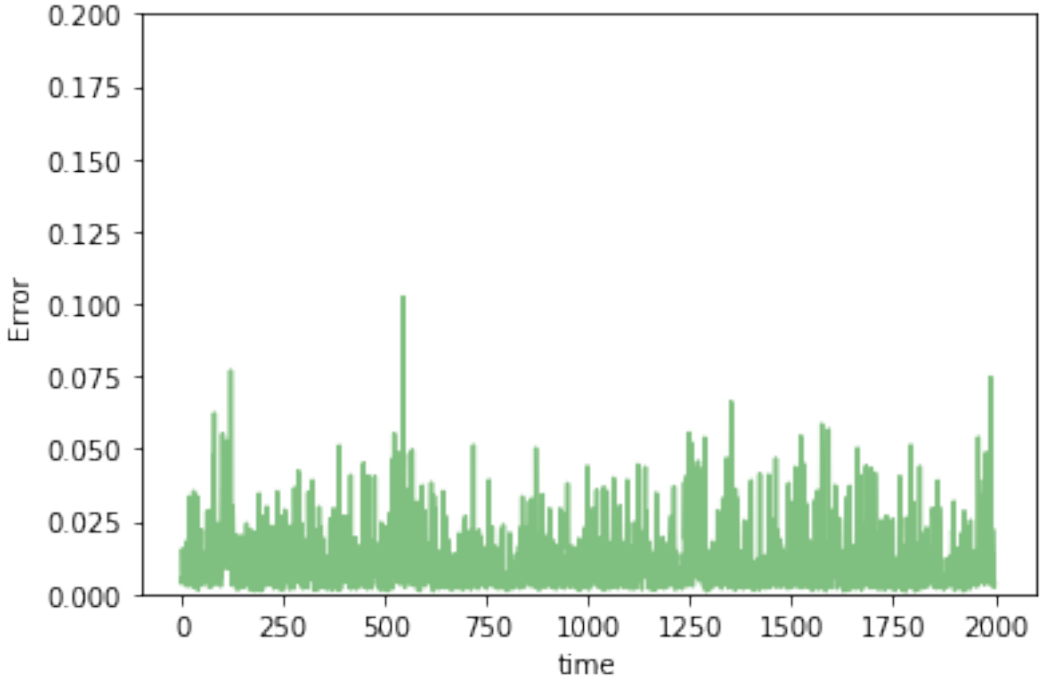

```
0.0124431411962
```

```
In [159]: test(model, test_X[0],0, name="gru1_50ano")
          test(model, test_X[2],0, name="gru1_50norm")
```

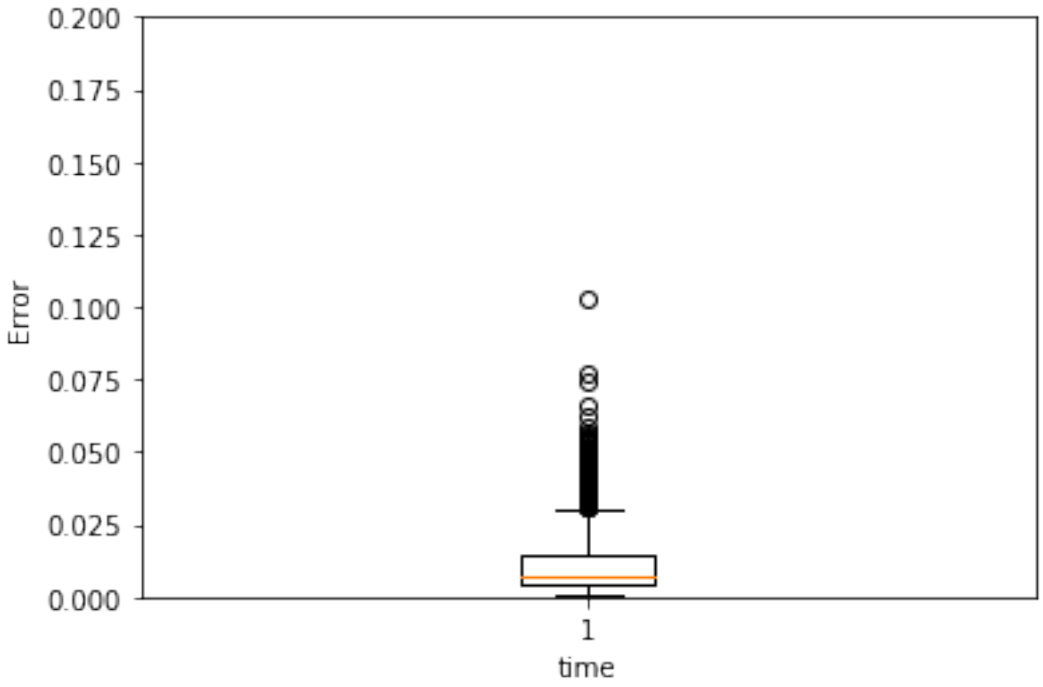


```
0.0137536352342
```





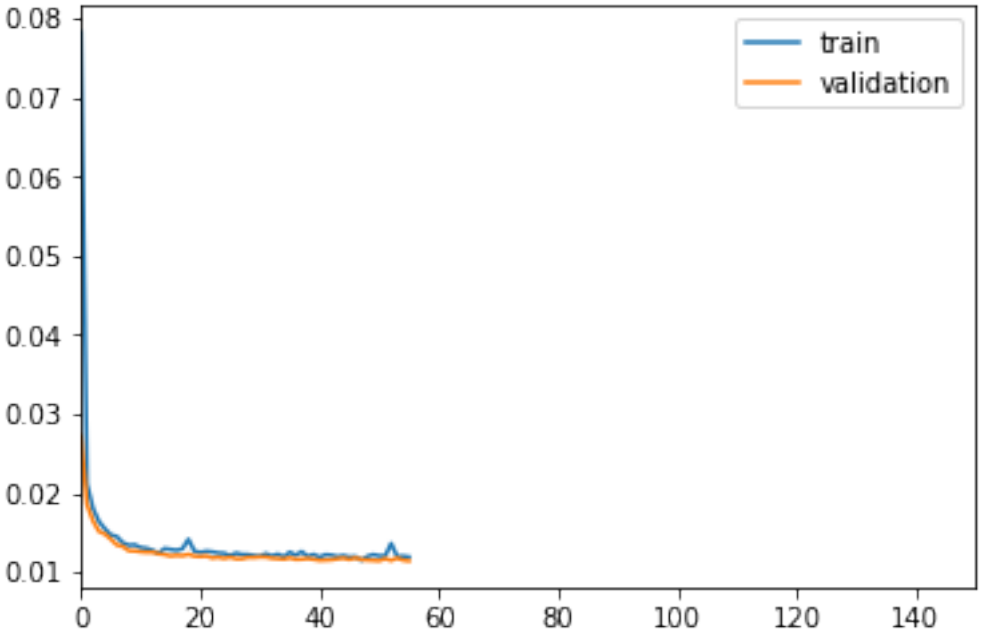
```
0.0110065358948
```



2.1.6 RNN with 2 GRU layers

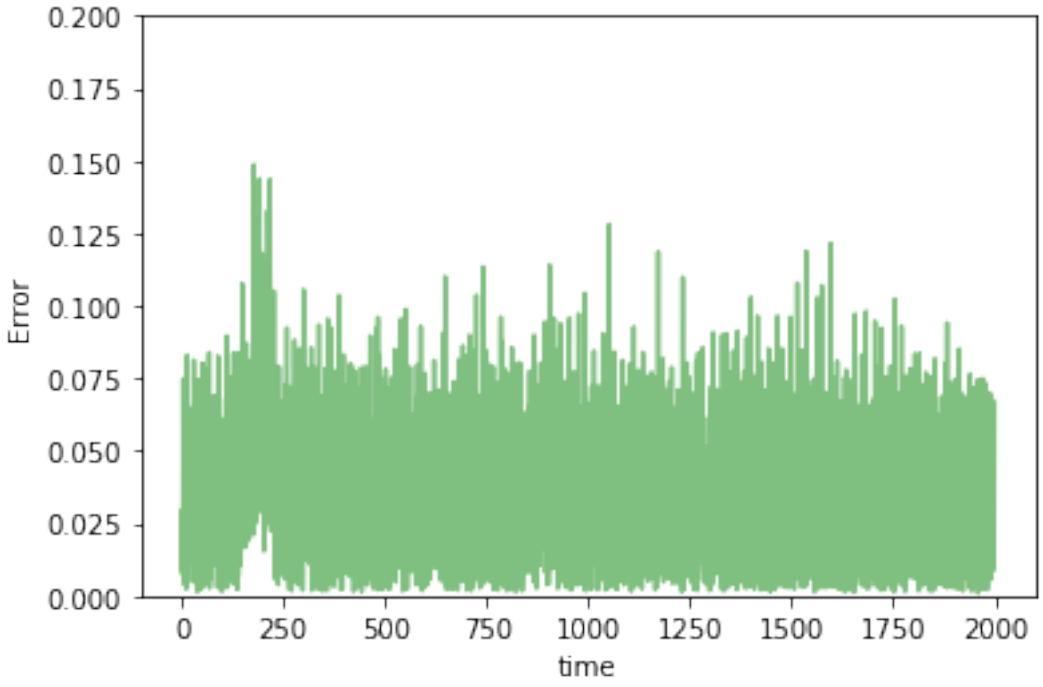
2 steps

```
In [160]: TIMESTEPS = 2
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [161]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [162]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [163]: train(model, tgen, vgen, name="gru2_2")
```

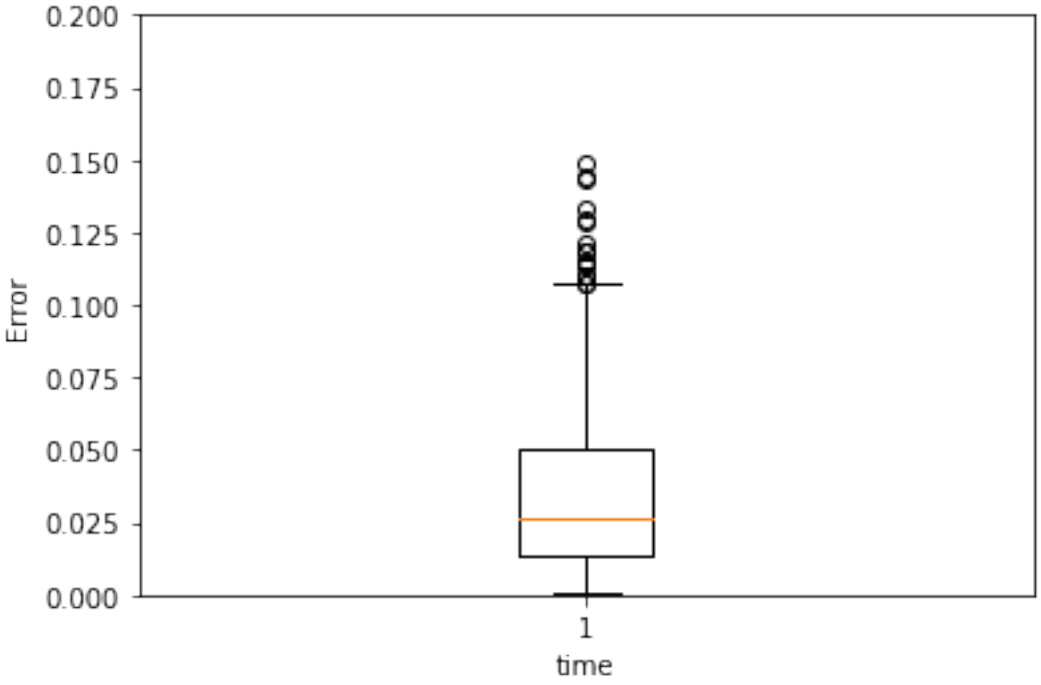


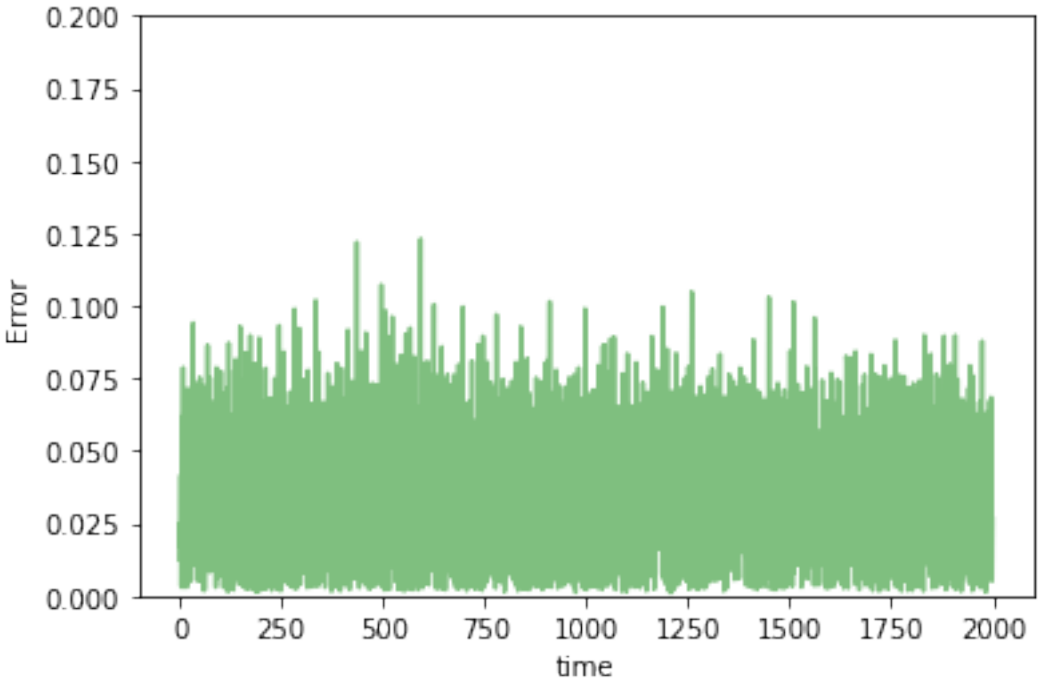

```
0.0119159275005
```

```
In [164]: test(model, test_X[0],0, name="gru2_2ano")
          test(model, test_X[2],0, name="gru_2norm")
```

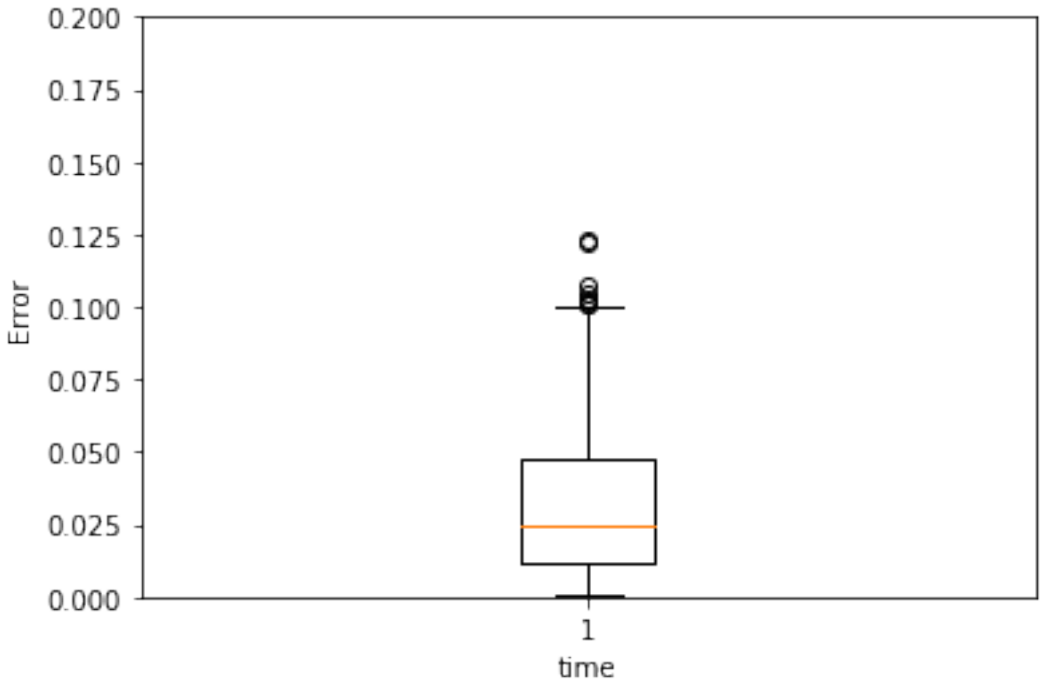


```
0.0336125948232
```



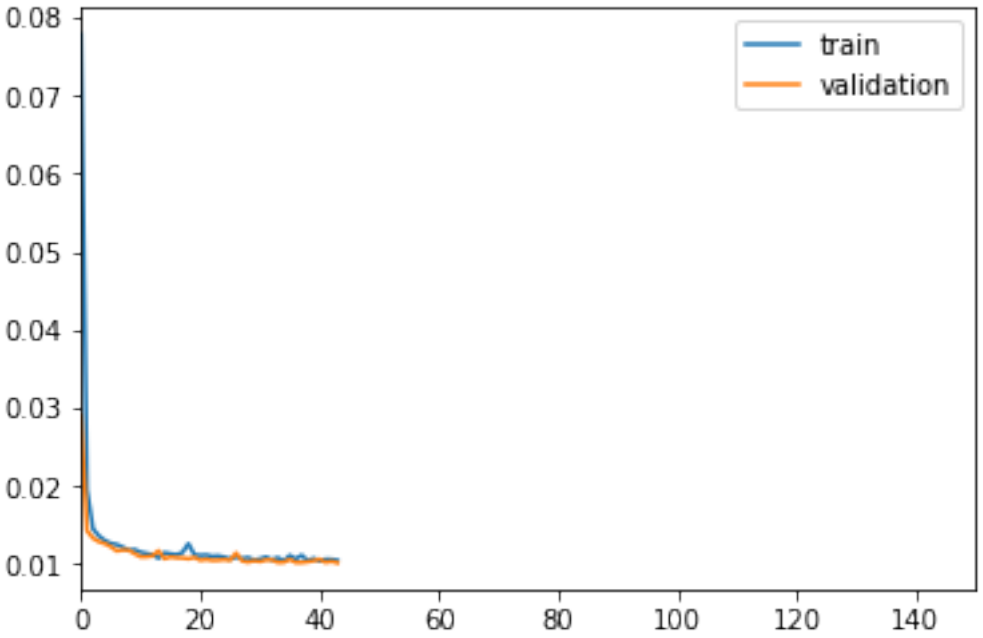


```
0.0314472533705
```



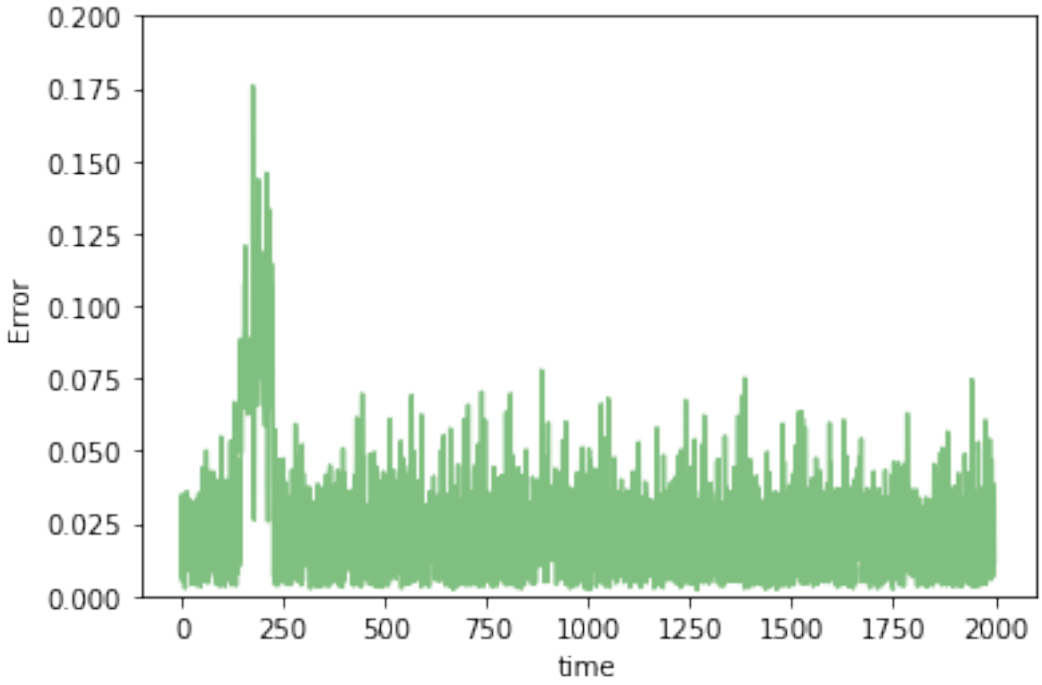
5 steps

```
In [165]: TIMESTEPS = 5
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [166]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [167]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [168]: train(model, tgen, vgen, name="gru2_5")
```

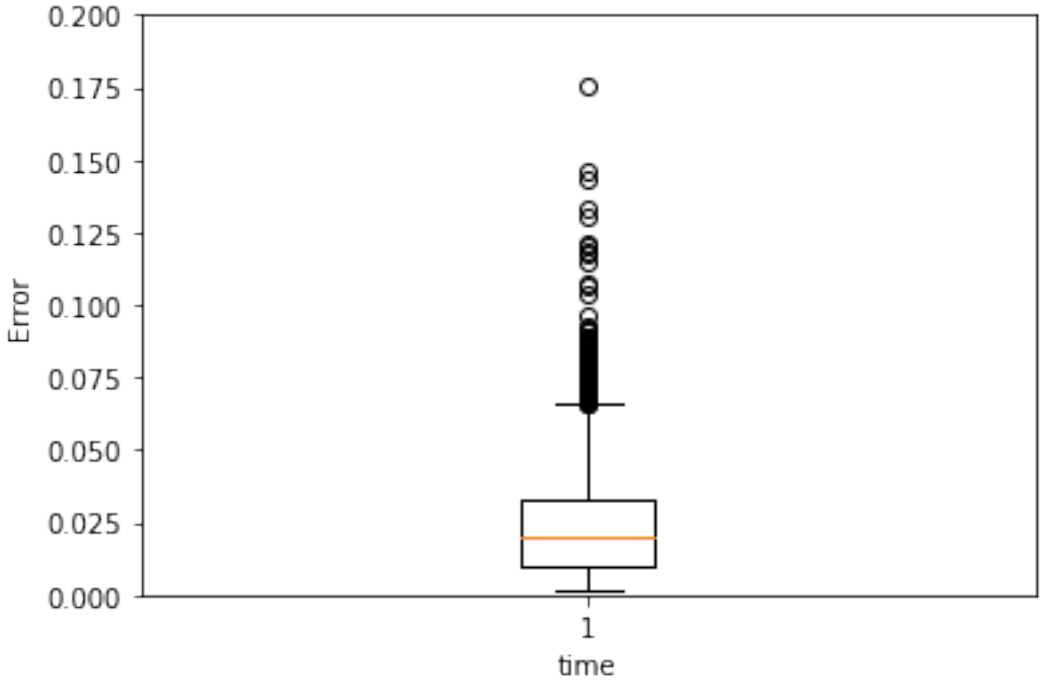


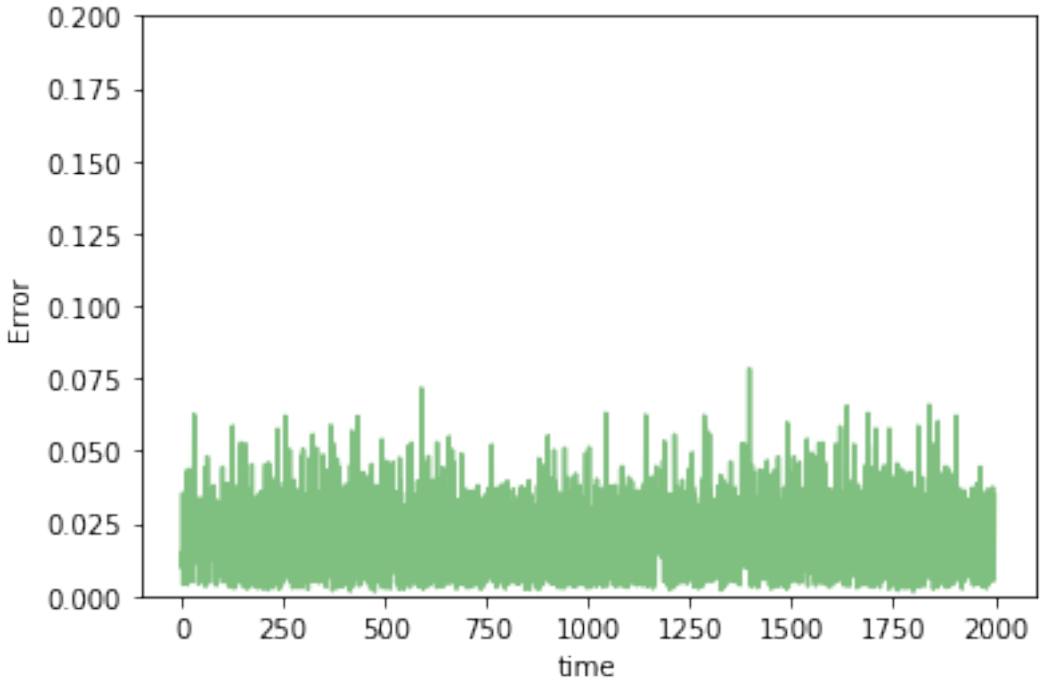

```
0.010465567232
```

```
In [169]: test(model, test_X[0],0, name="gru2_5ano")
          test(model, test_X[2],0, name="gru2_5norm")
```

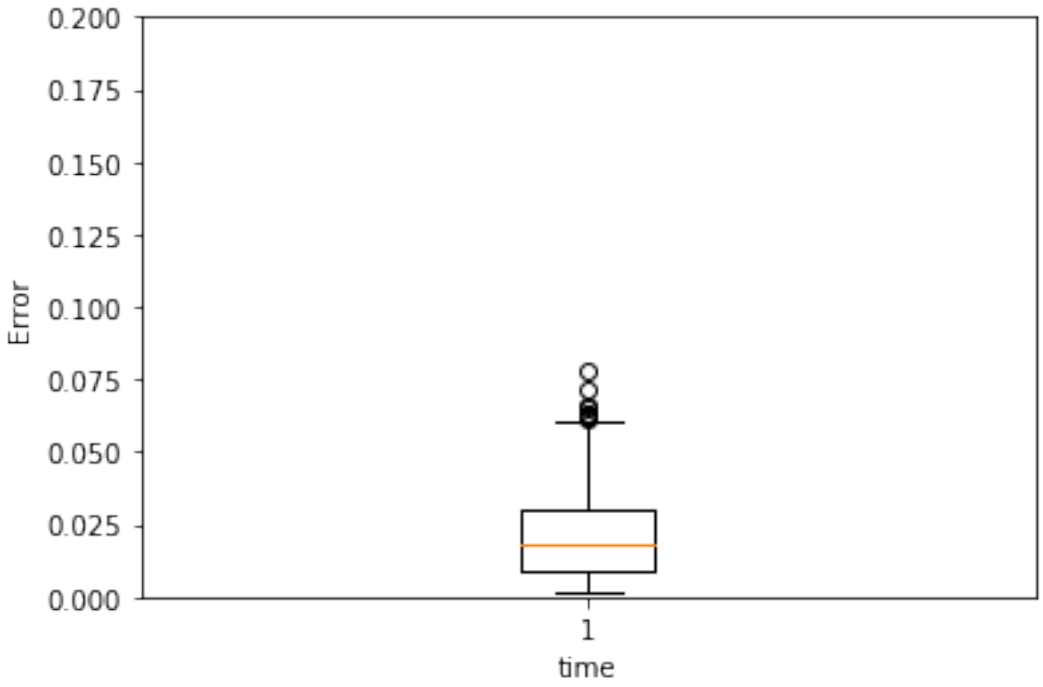


```
0.0239122815306
```



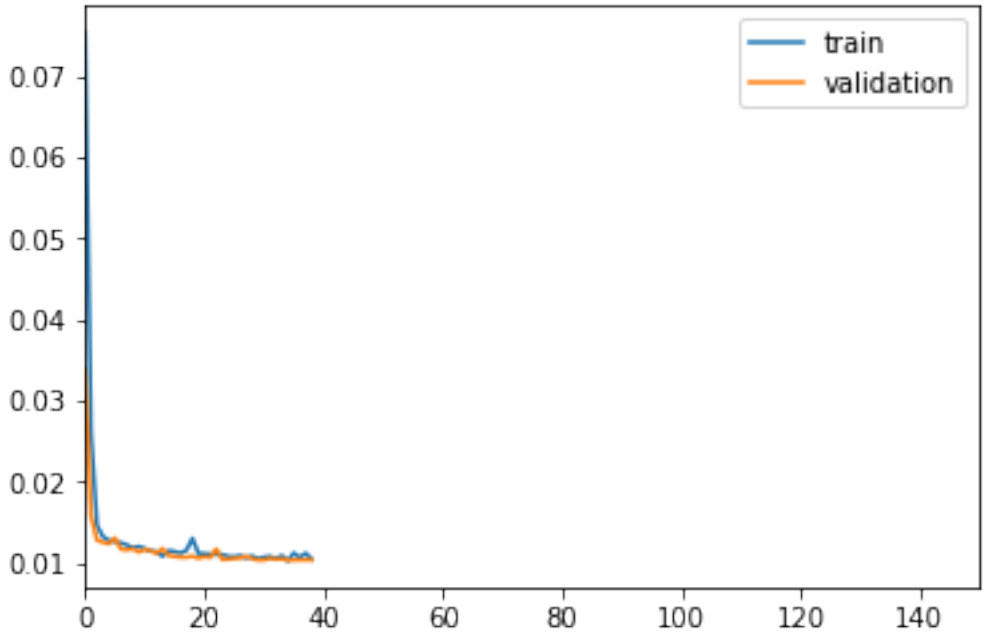


```
0.0204081843196
```



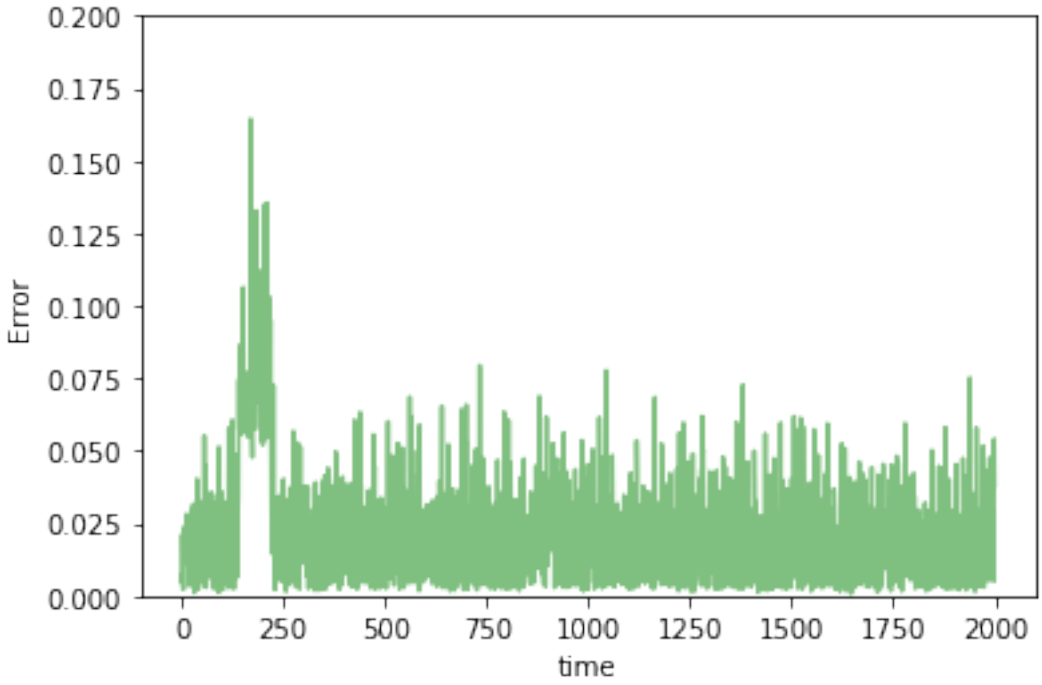
10 steps

```
In [170]: TIMESTEPS = 10
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS, 0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [171]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [172]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [173]: train(model, tgen, vgen, name="gru2_10")
```

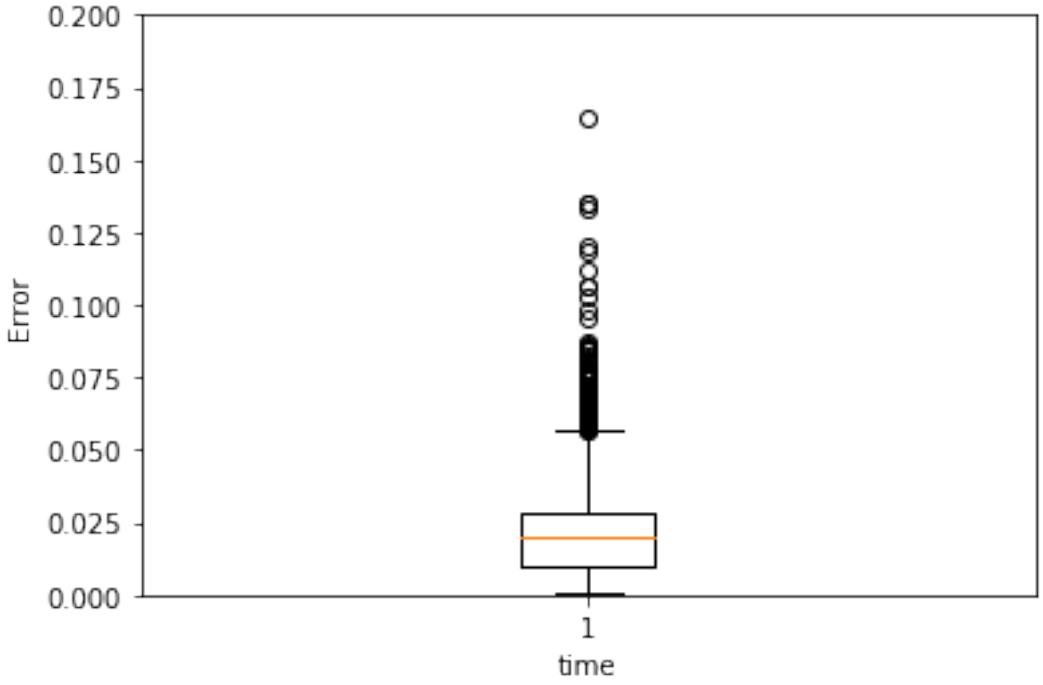


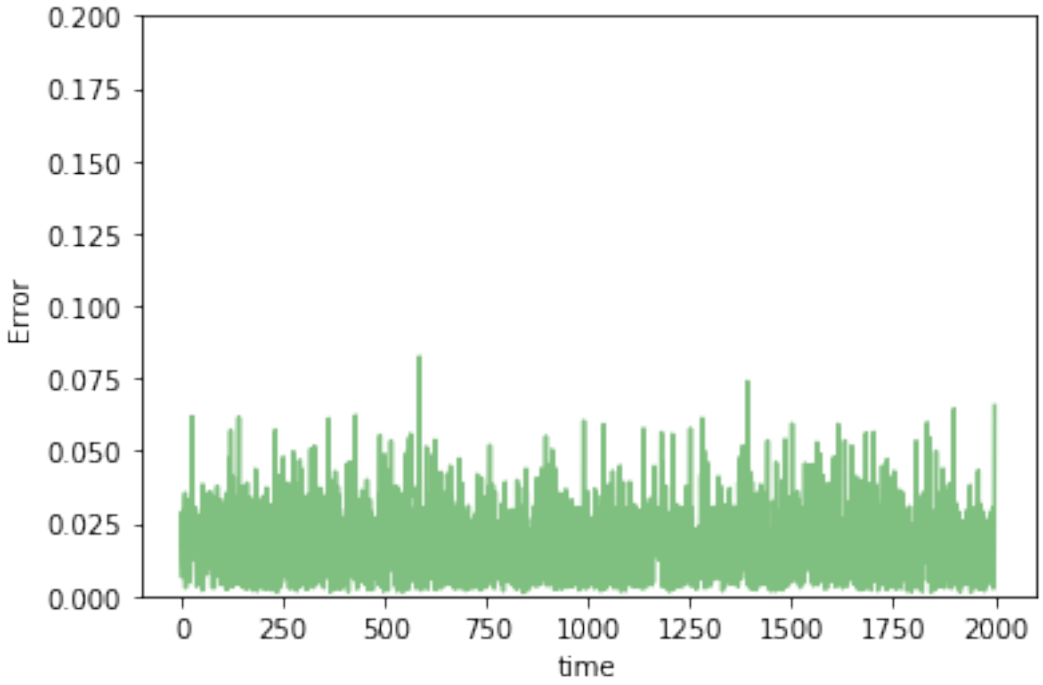

```
0.0104587077856
```

```
In [174]: test(model, test_X[0],0, name="gru2_10ano")
          test(model, test_X[2],0, name="gru2_10norm")
```

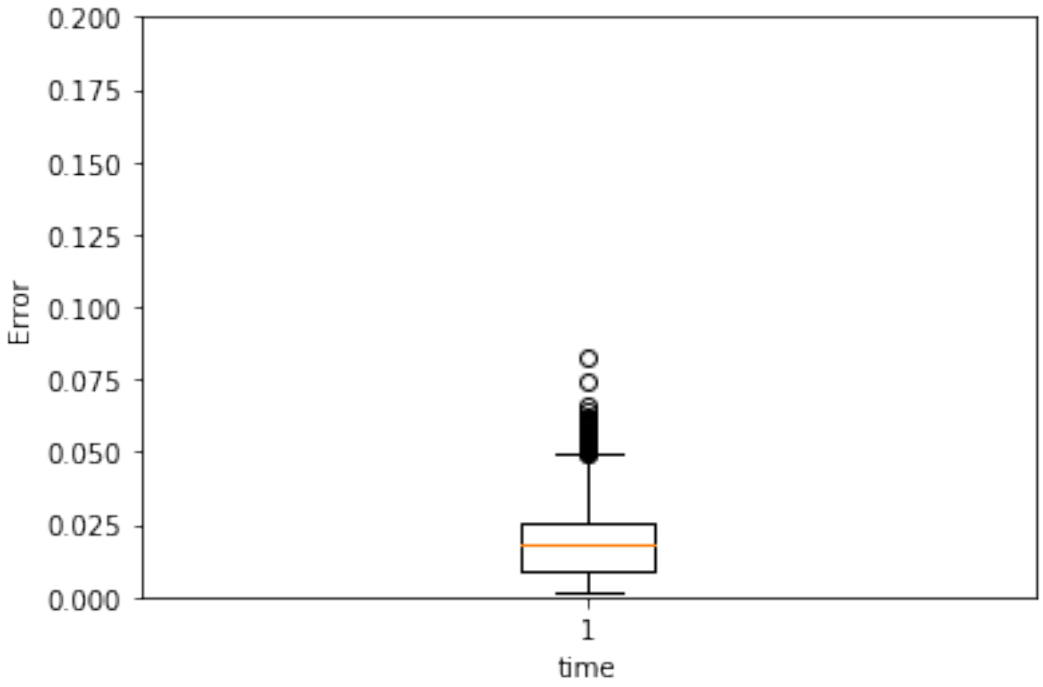


```
0.0223163464548
```



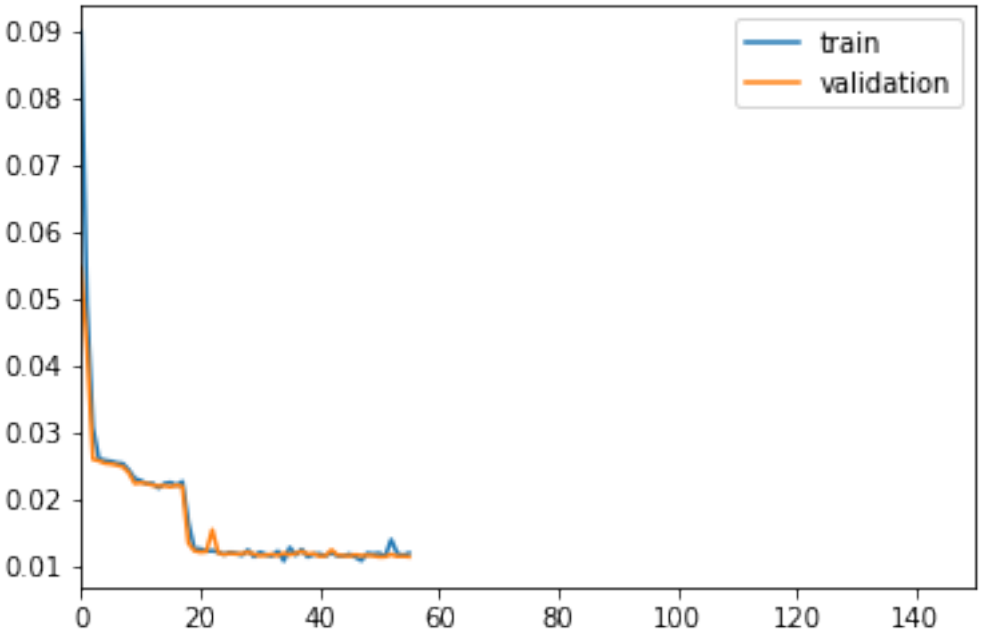


```
0.0189544071191
```



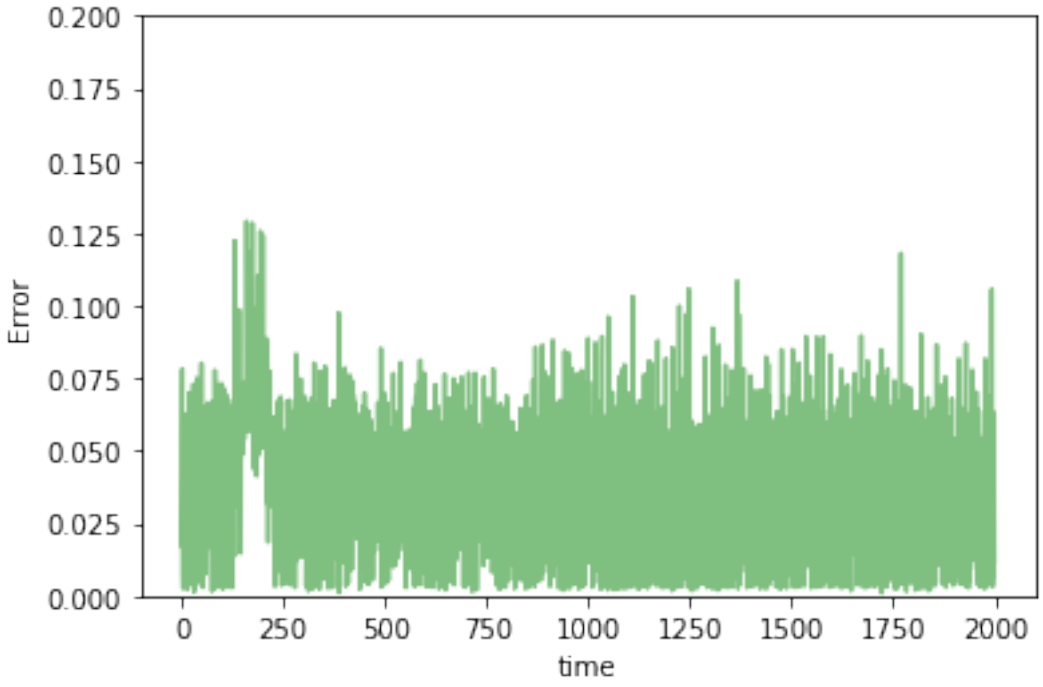
20 steps

```
In [175]: TIMESTEPS = 20
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [176]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [177]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [178]: train(model, tgen, vgen, name="gru2_20")
```

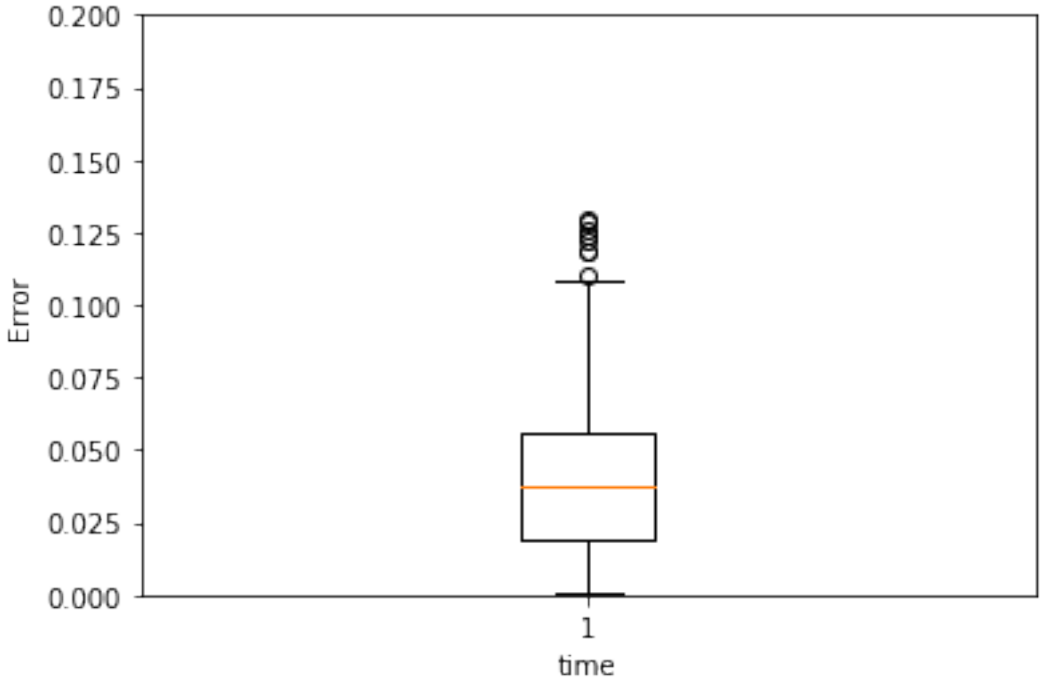


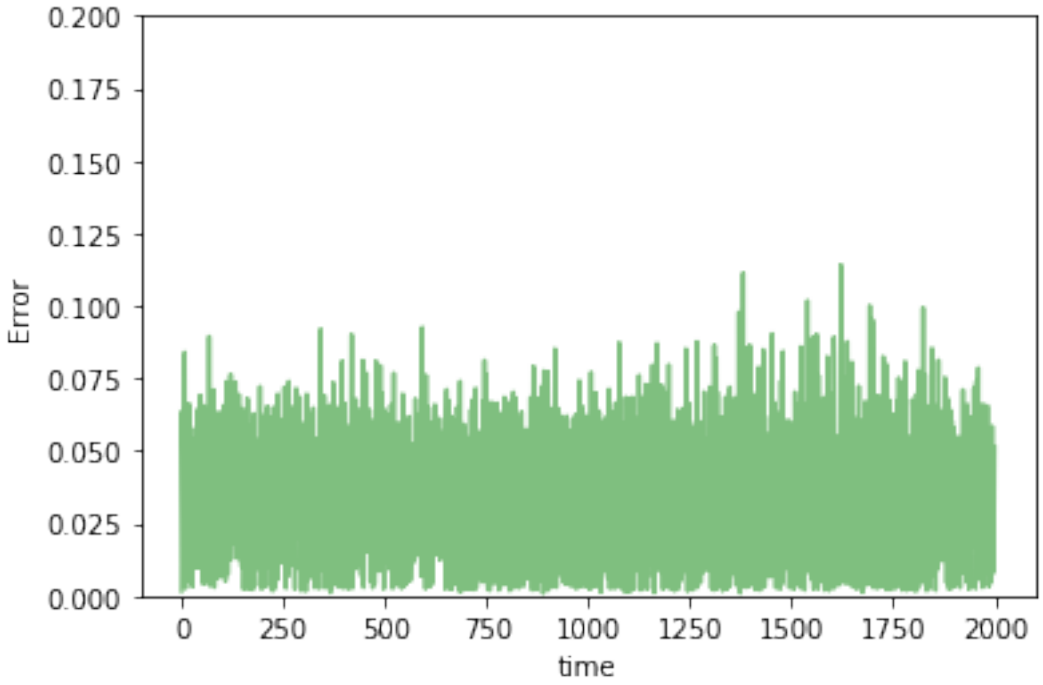
0.0118403707009

In [179]:
```
test(model, test_X[0],0, name="gru2_20ano")
test(model, test_X[2],0, name="gru2_20norm")
```

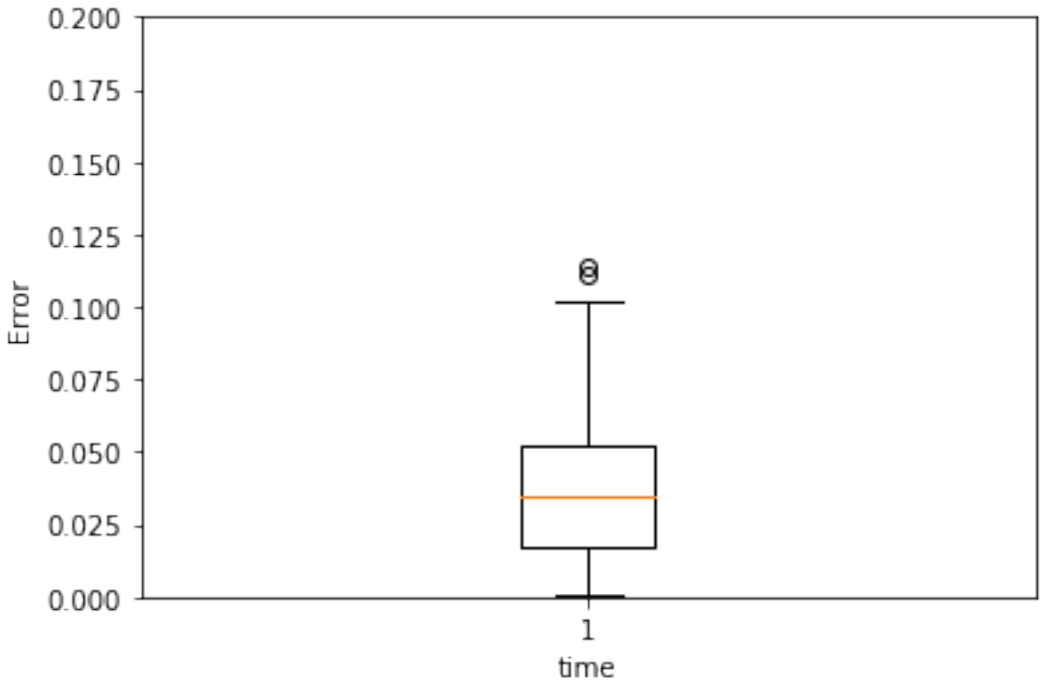


```
0.0385139418665
```



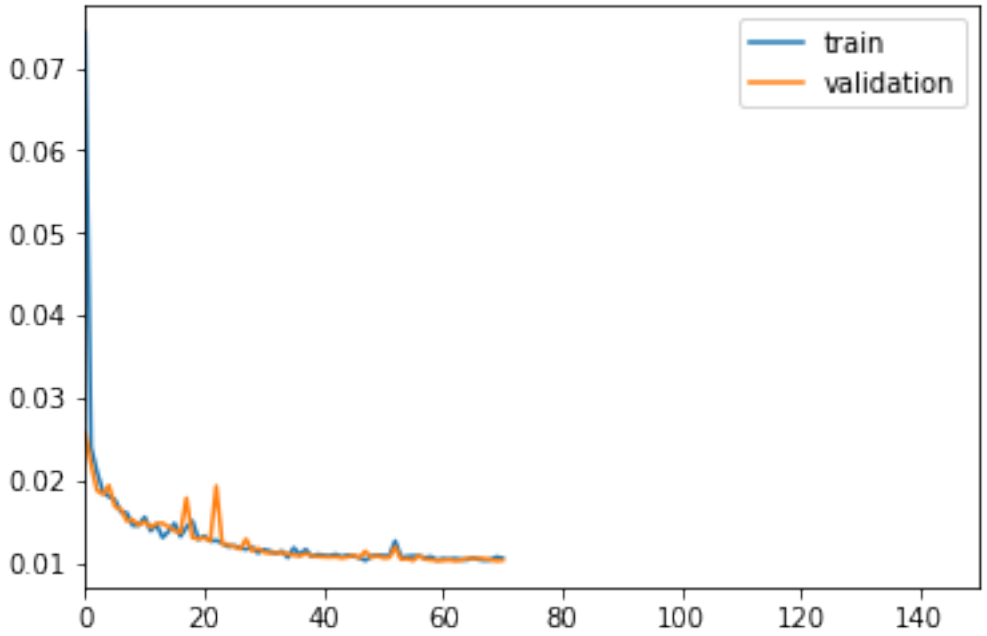


```
0.0355755808529
```



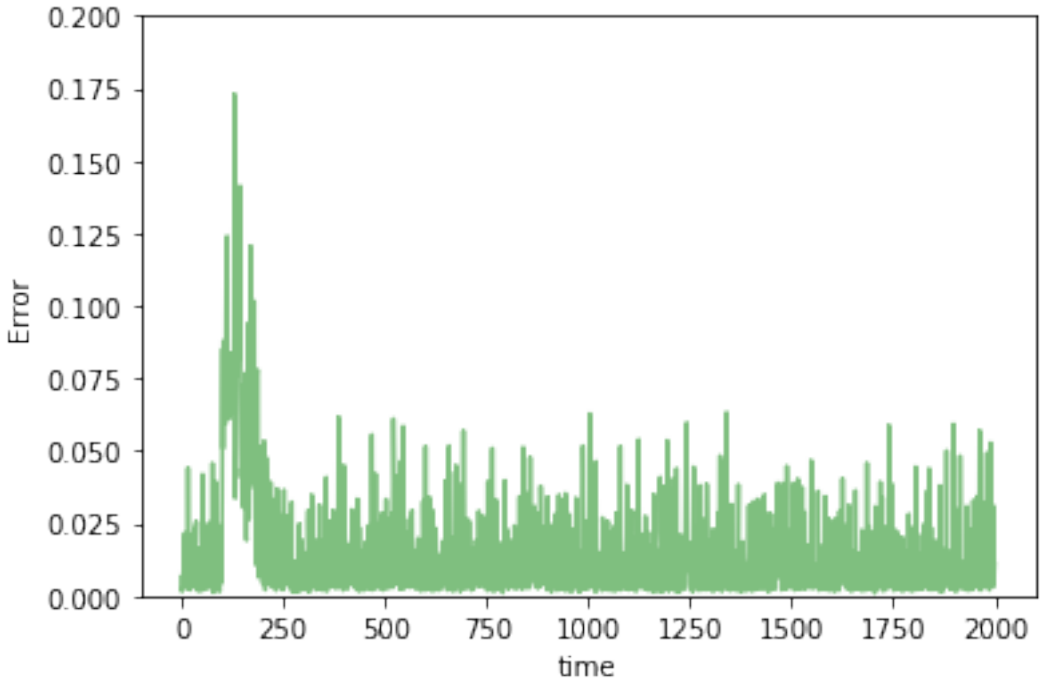
50 steps

```
In [180]: TIMESTEPS = 50
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [181]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [182]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [183]: train(model, tgen, vgen, name="gru2_50")
```

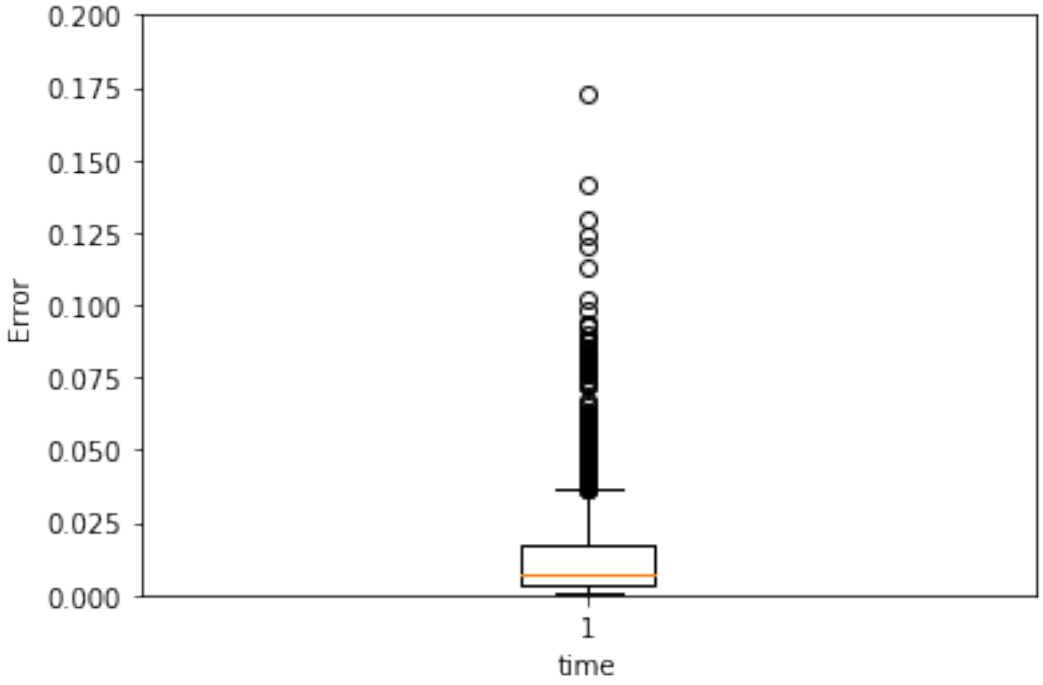


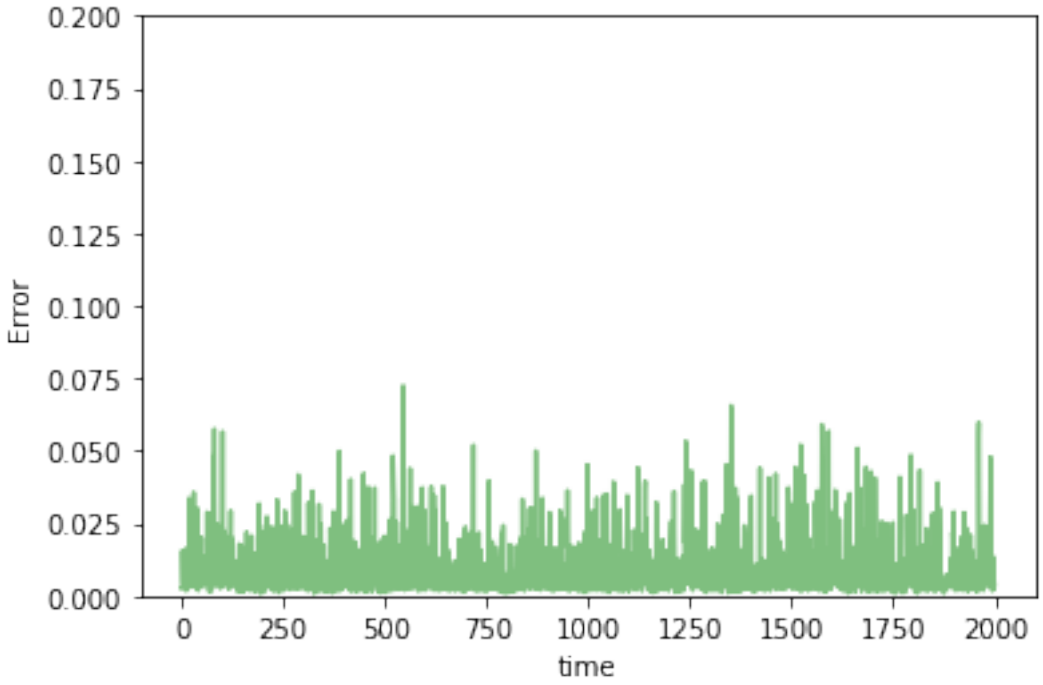

```
0.0105134046751
```

```
In [184]: test(model, test_X[0],0, name="gru2_50ano")
          test(model, test_X[2],0, name="gru2_50norm")
```

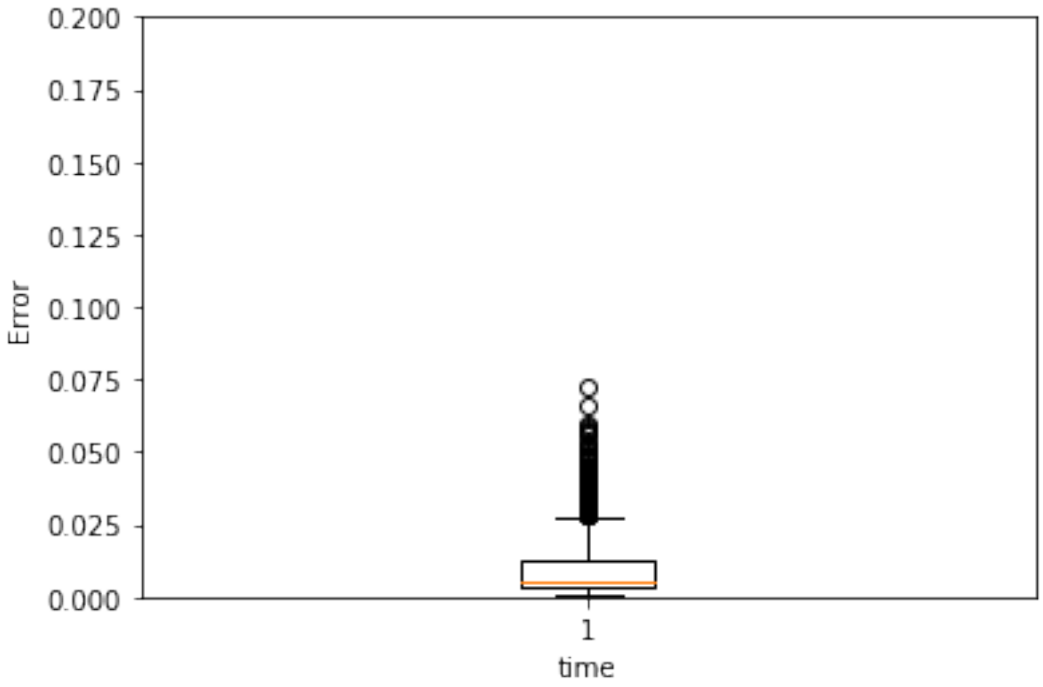


```
0.0138064416006
```





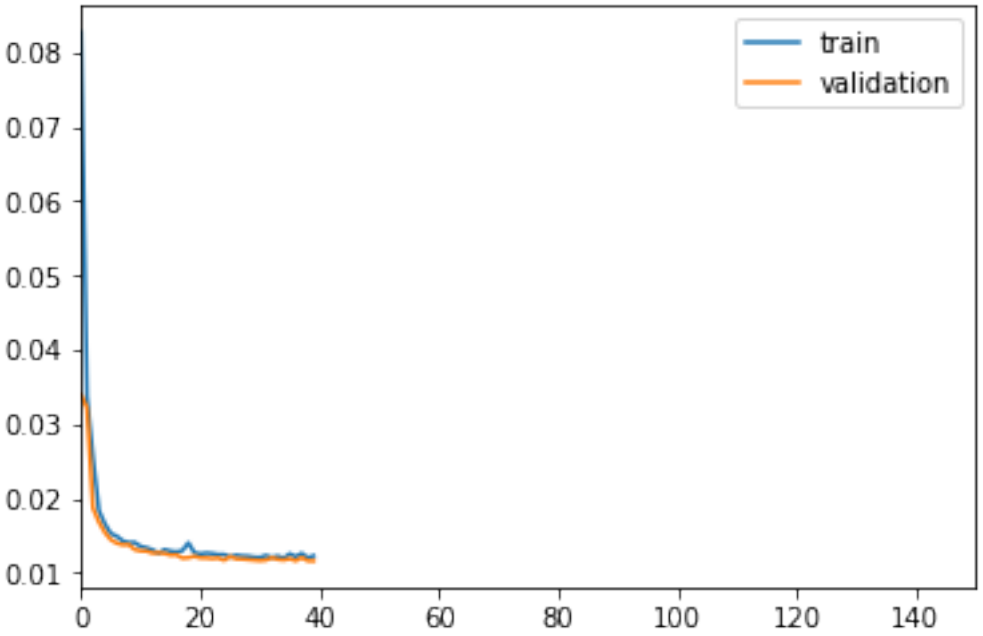
```
0.00958662816616
```



2.1.7 RNN with 3 GRU layers

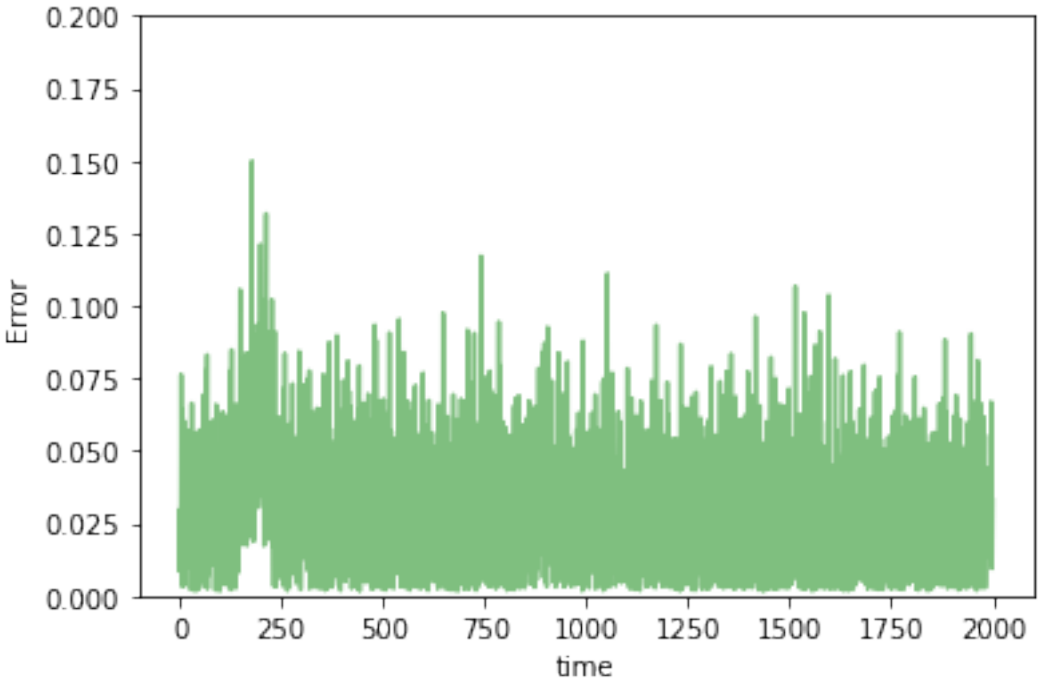
2 steps

```
In [185]: TIMESTEPS = 2
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [186]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [187]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [188]: train(model, tgen, vgen, name="gru3_2")
```

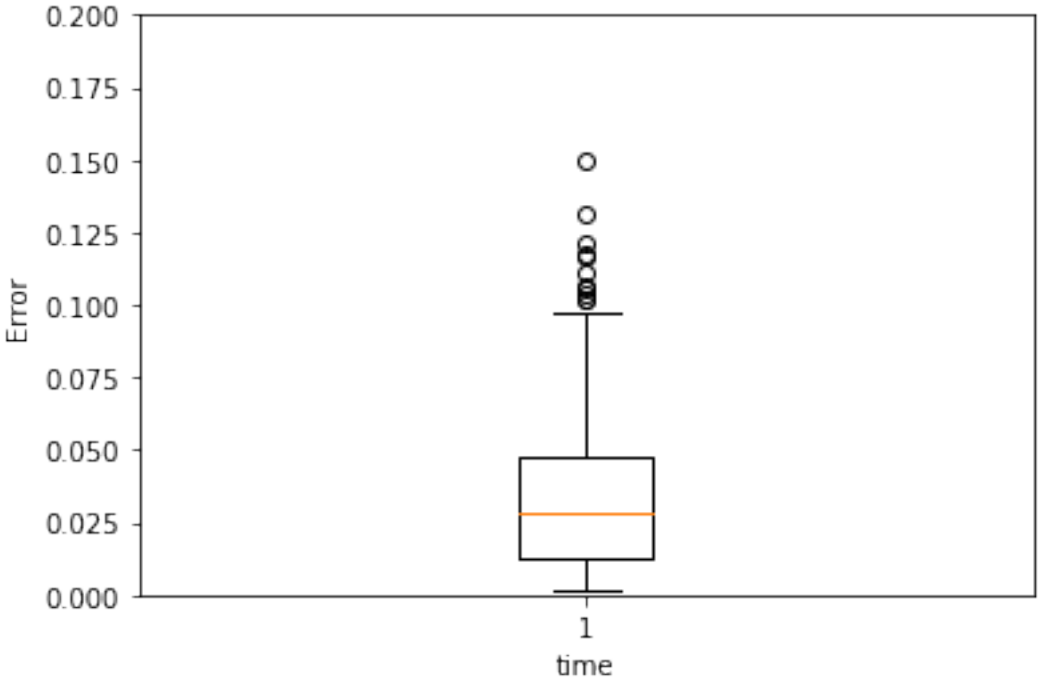


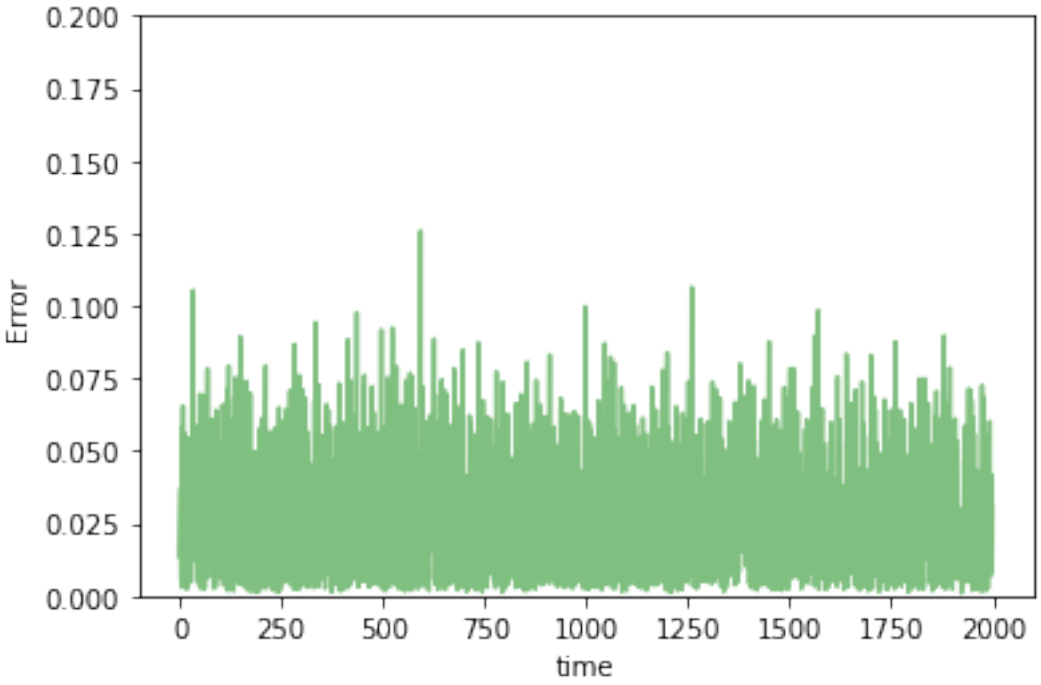

```
0.0122796919814
```

```
In [189]: test(model, test_X[0],0, name="gru3_2ano")
          test(model, test_X[2],0, name="gru3_2norm")
```

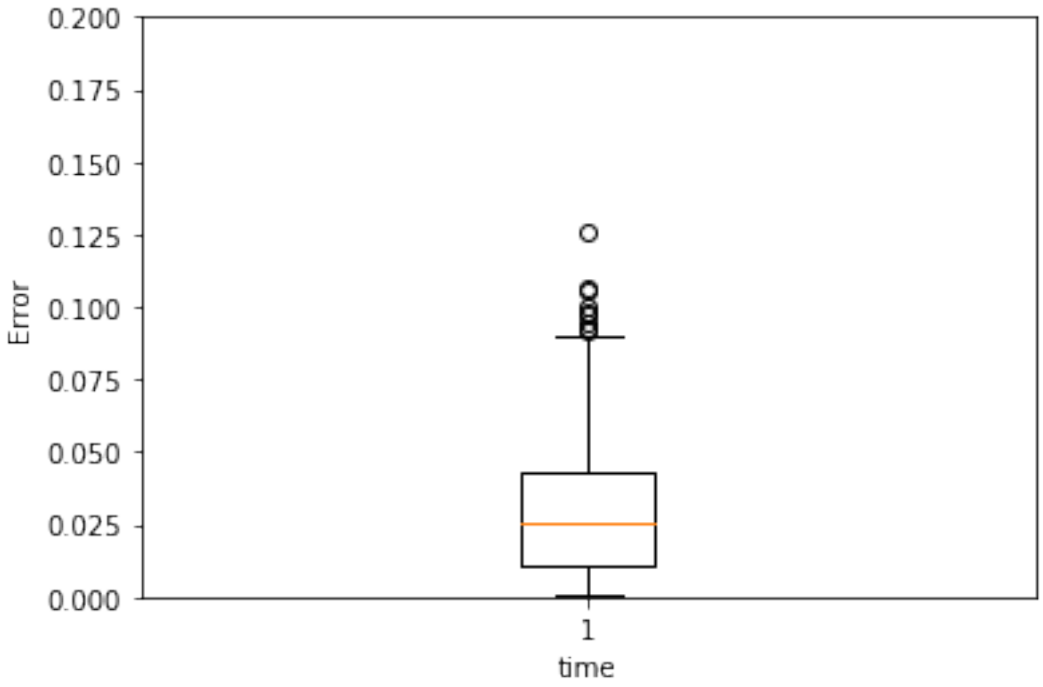


```
0.0314869493709
```



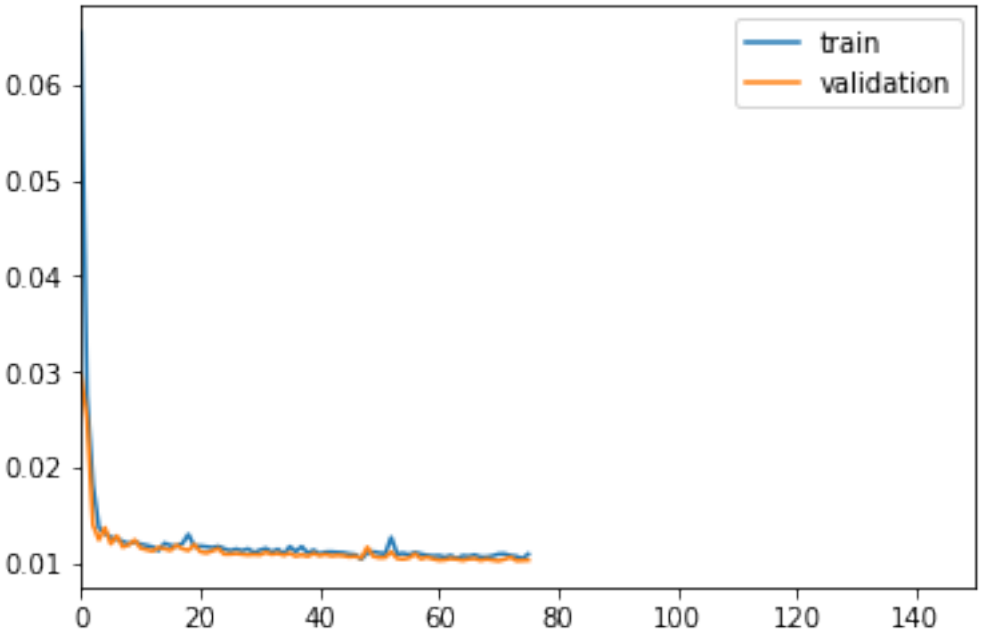


```
0.0288159244458
```



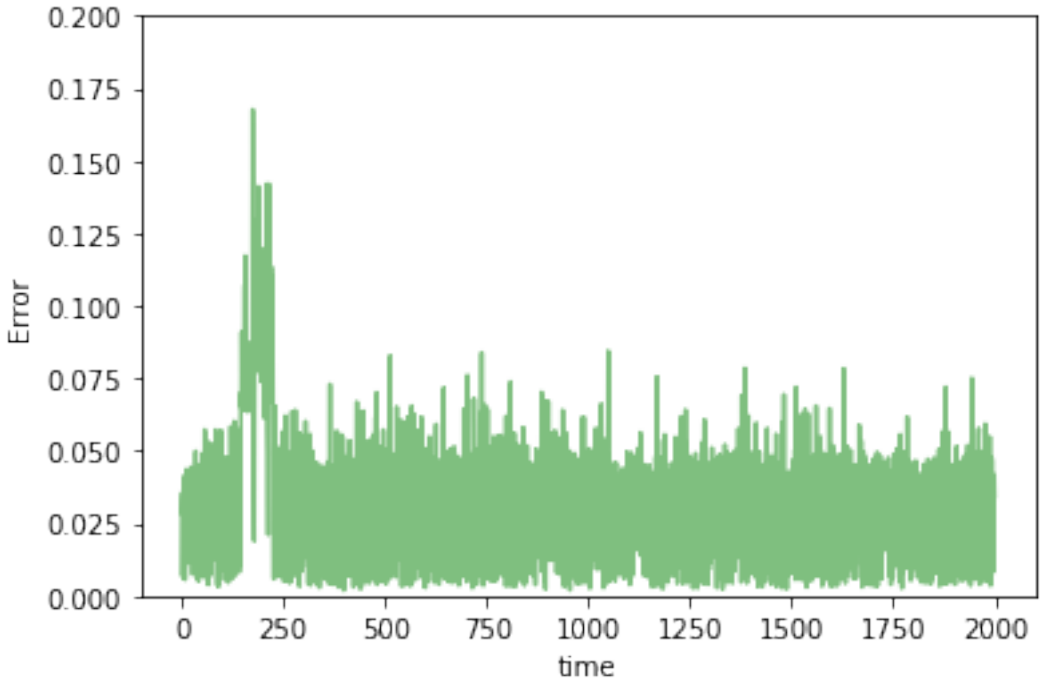
5 steps

```
In [190]: TIMESTEPS = 5
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [191]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [192]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [193]: train(model, tgen, vgen, name="gru3_5")
```

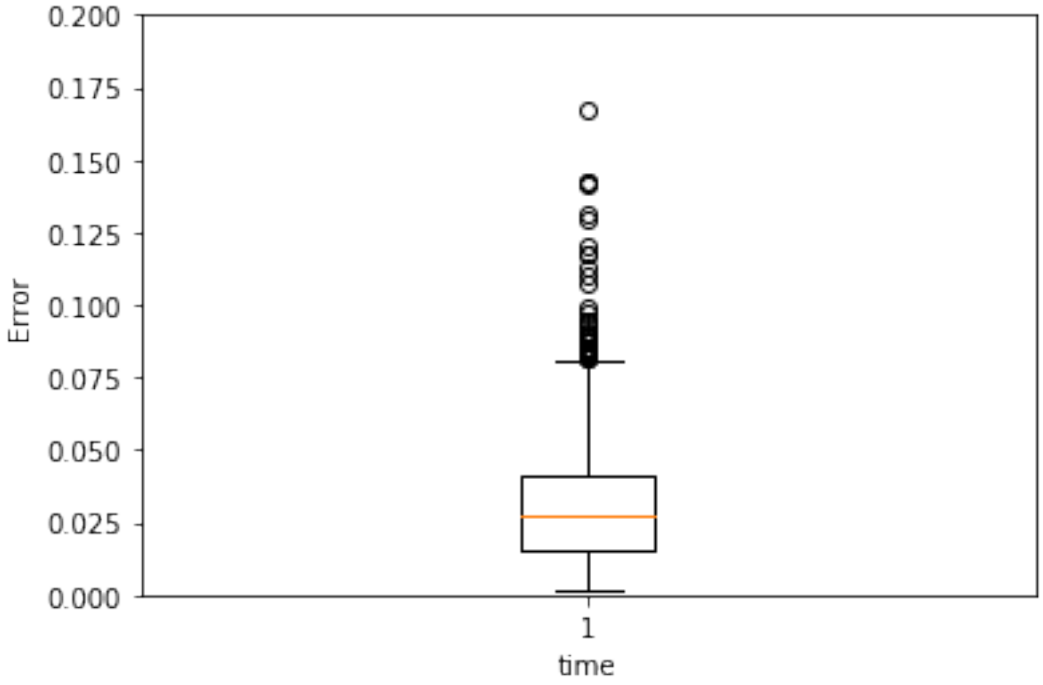


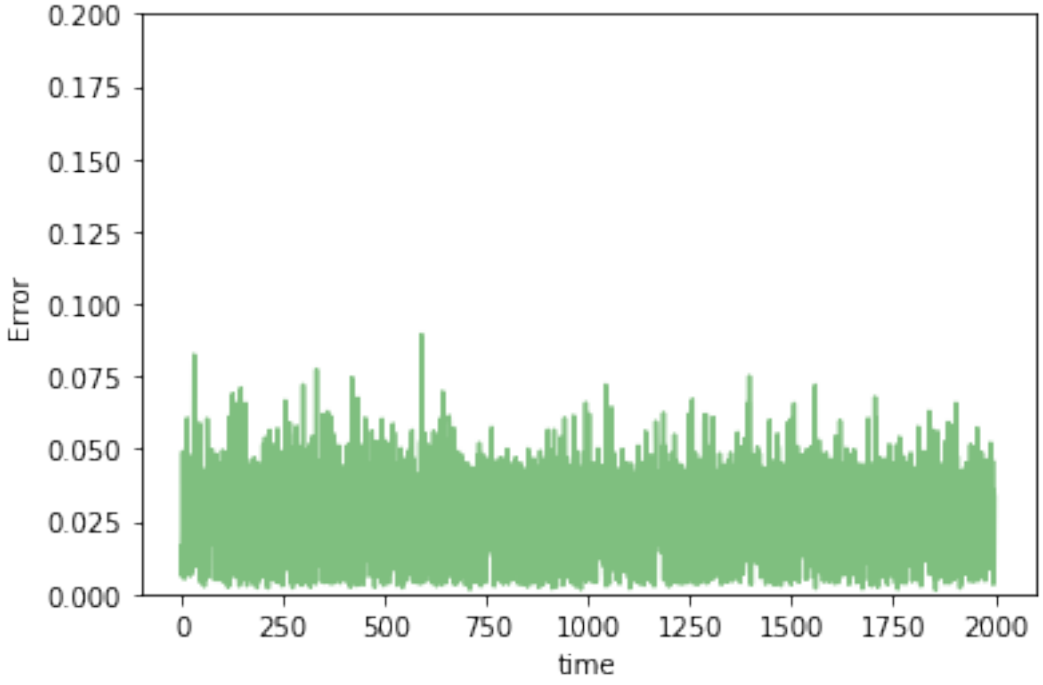

```
0.0108838670679
```

```
In [194]: test(model, test_X[0],0, name="gru3_5ano")
          test(model, test_X[2],0, name="gru3_5norm")
```

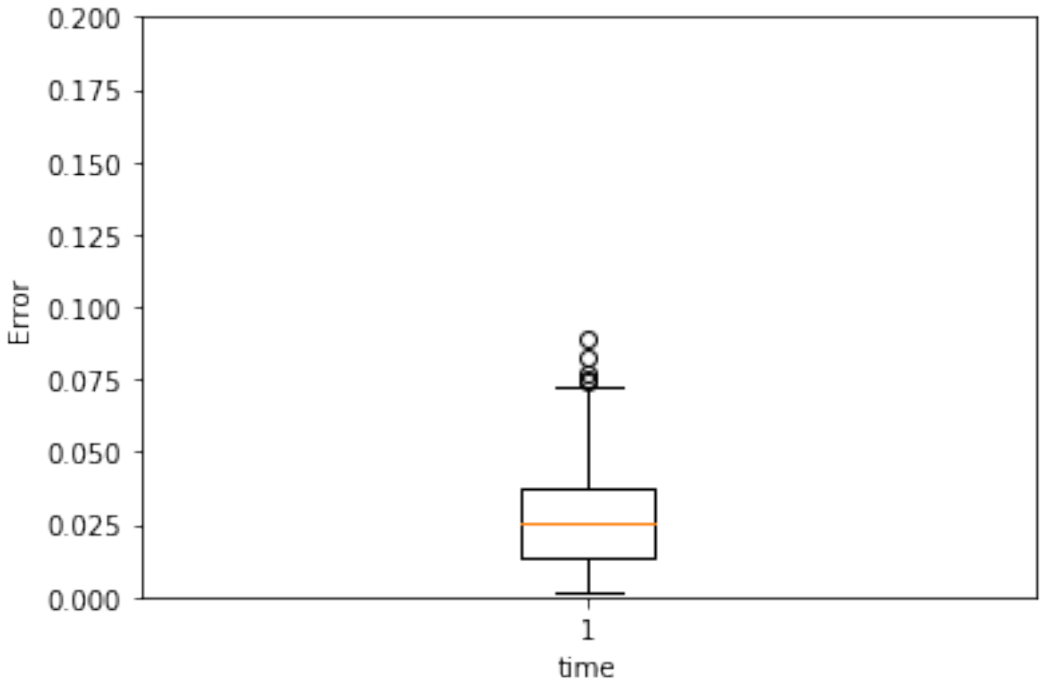


```
0.0299080517858
```



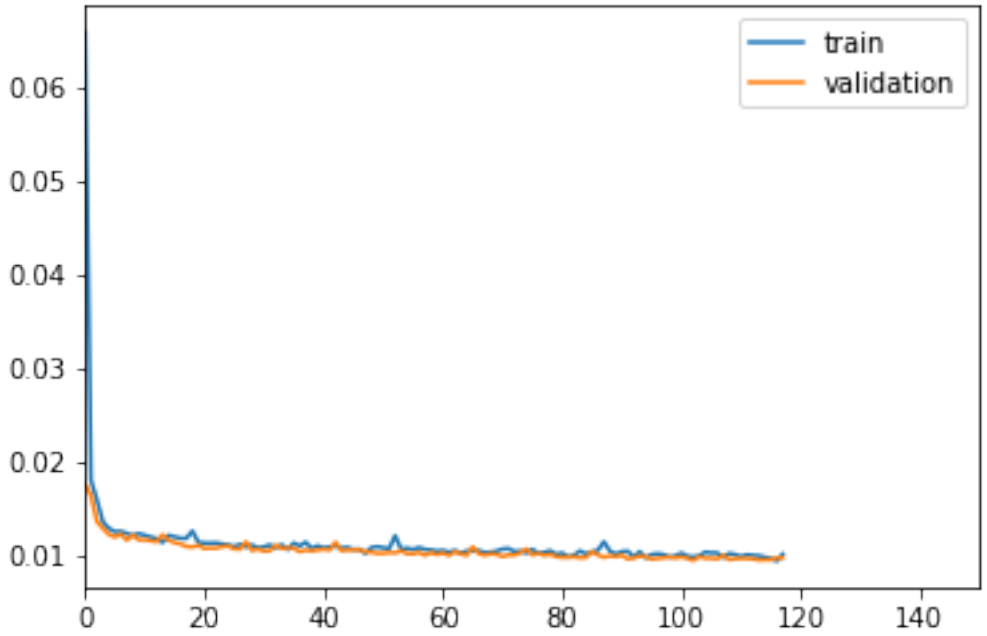


```
0.0264847985192
```



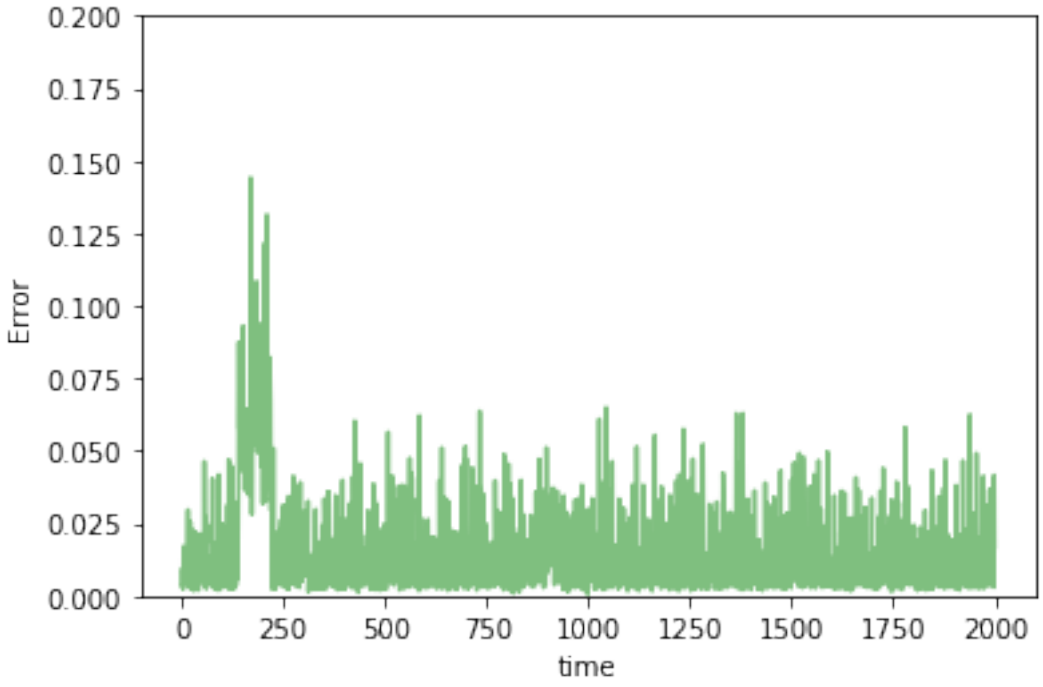
10 steps

```
In [195]: TIMESTEPS = 10
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS, 0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [196]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [197]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [198]: train(model, tgen, vgen, name="gru3_10")
```

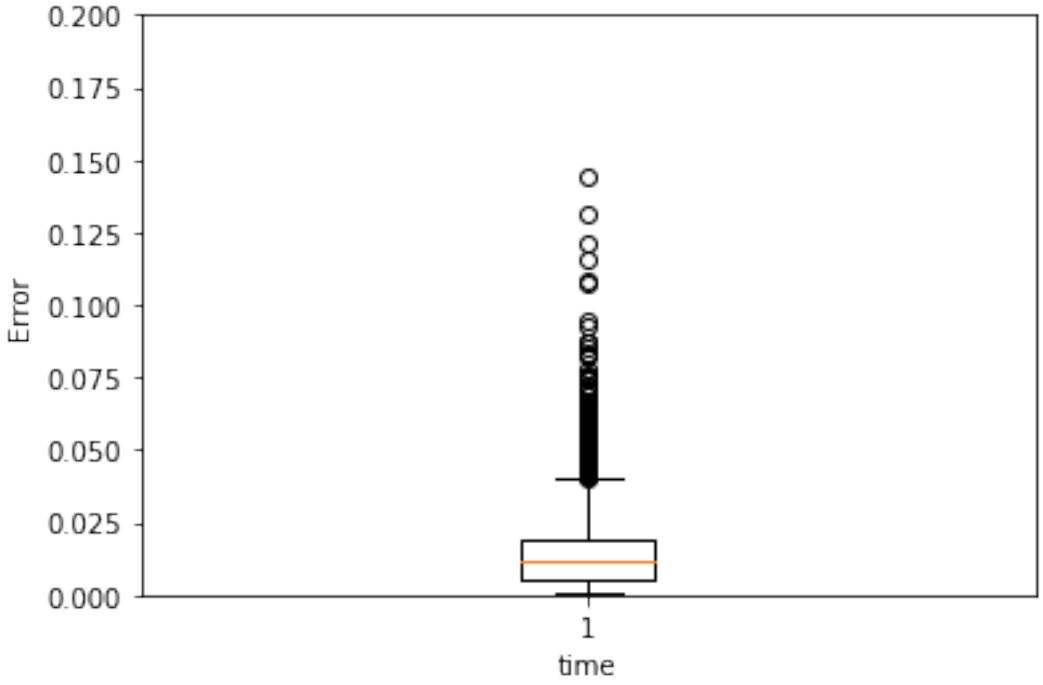


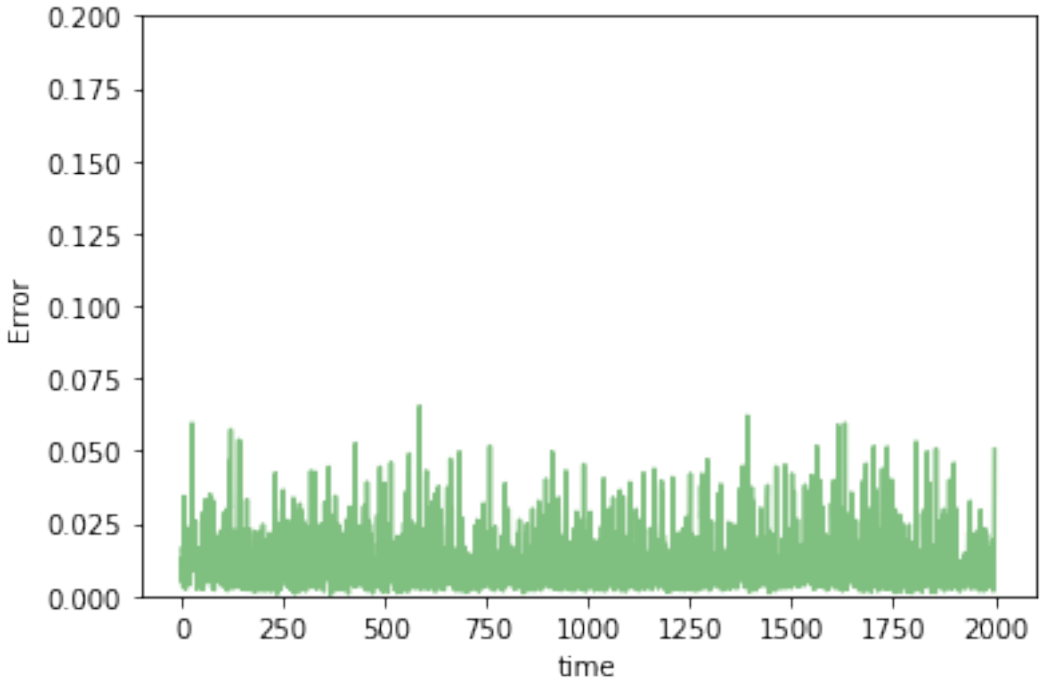

```
0.0100391358652
```

```
In [199]: test(model, test_X[0],0, name="gru3_10ano")
          test(model, test_X[2],0, name="gru3_10norm")
```

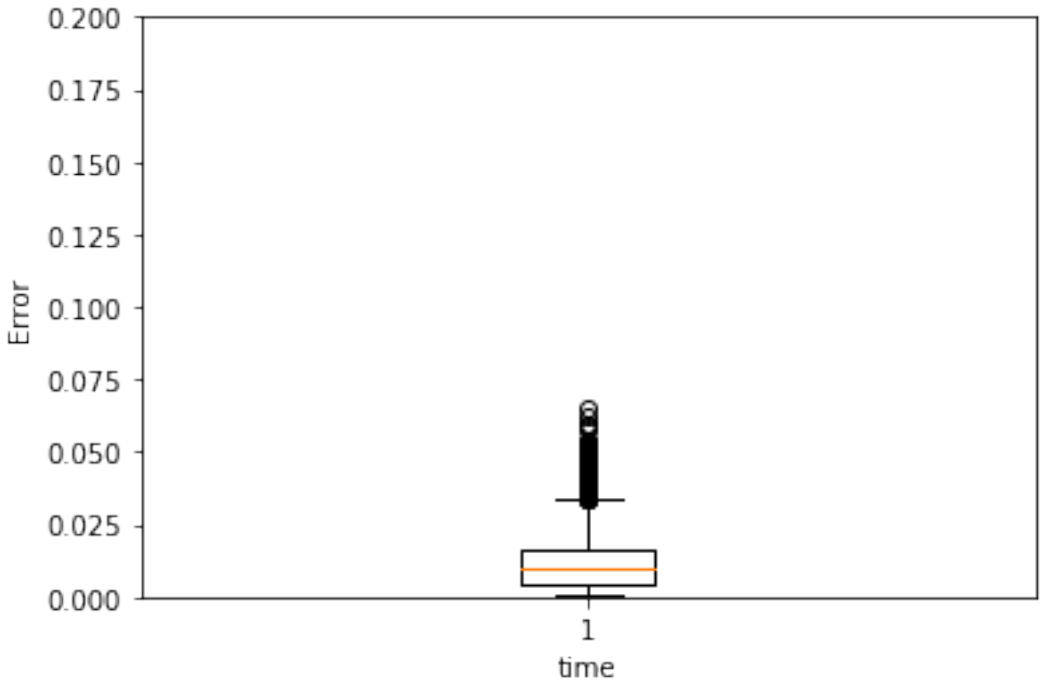


```
0.0156242101756
```



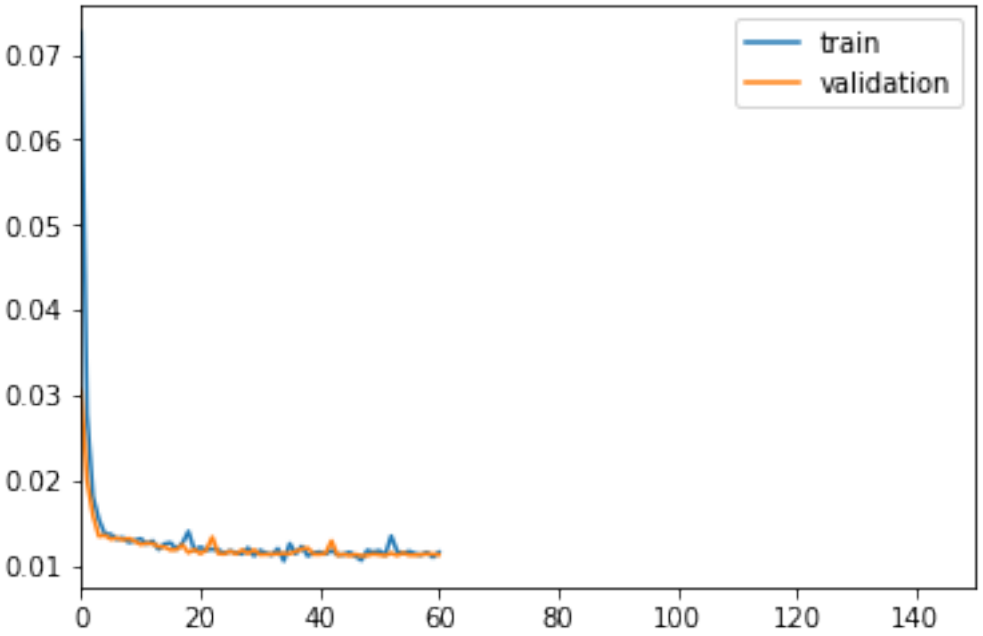


```
0.0121511181857
```



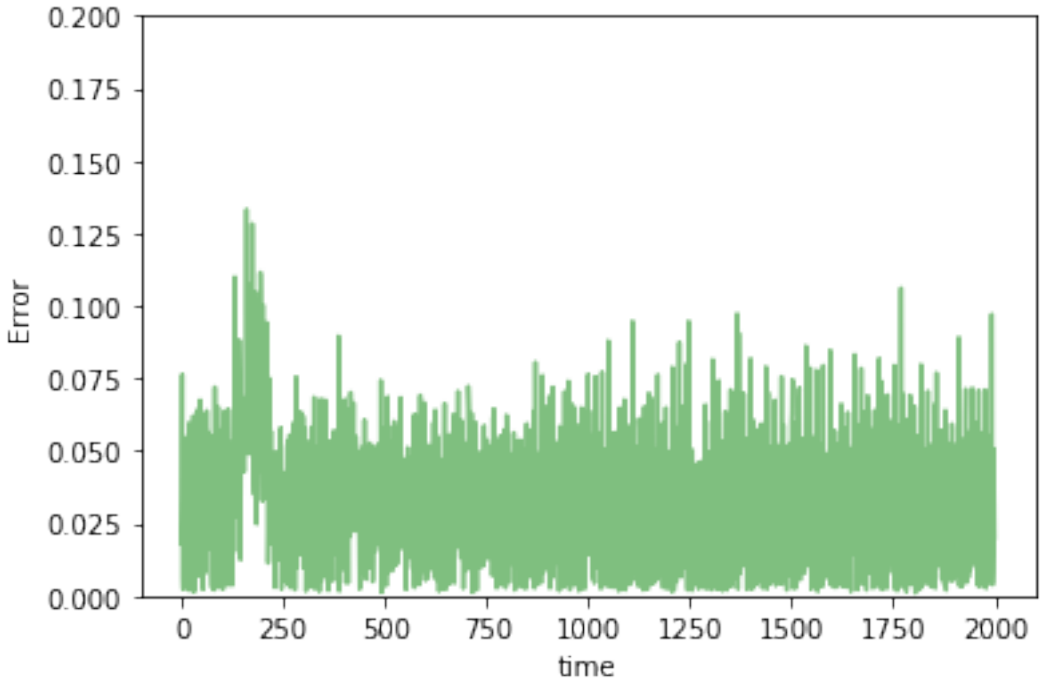
20 steps

```
In [200]: TIMESTEPS = 20
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [201]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [202]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [203]: train(model, tgen, vgen, name="gru3_20")
```

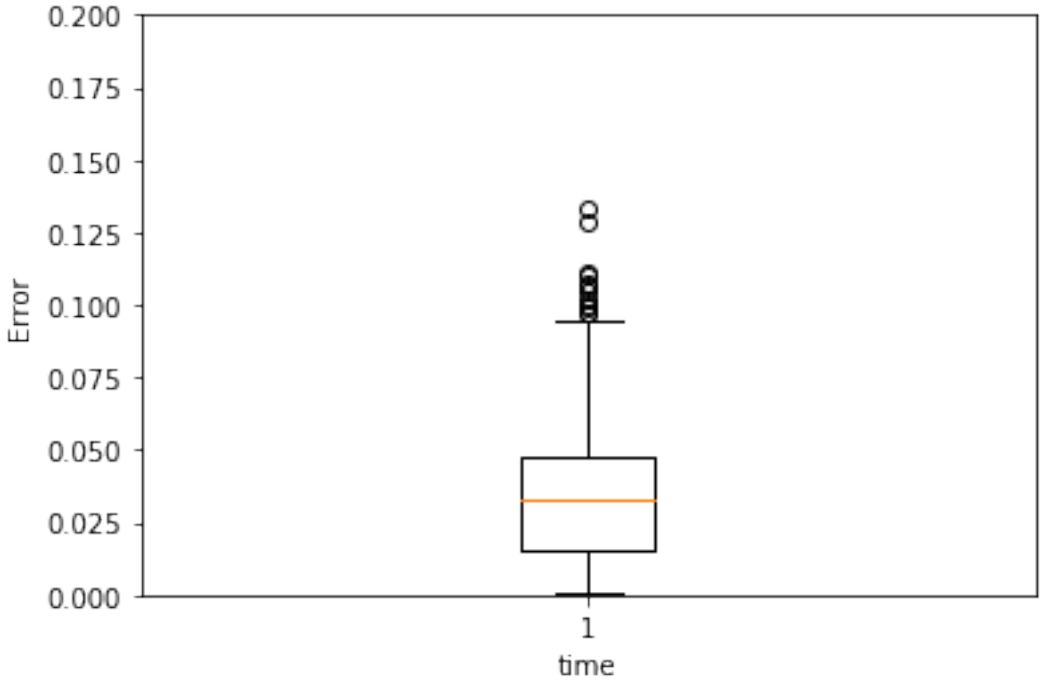


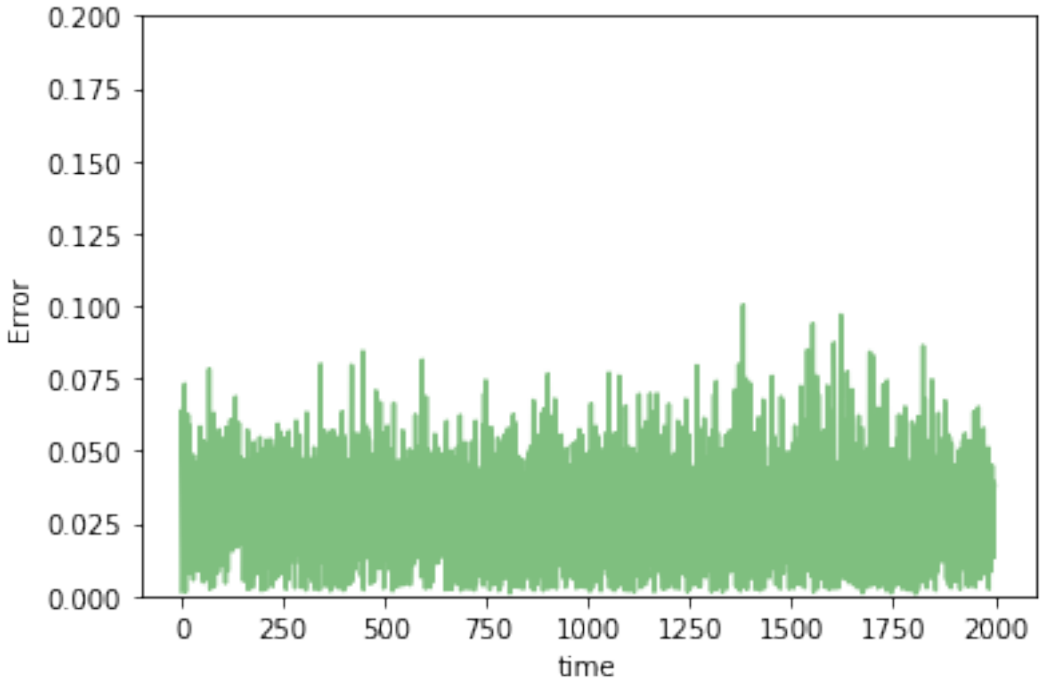

```
0.0115741605596
```

```
In [204]: test(model, test_X[0],0, name="gru3_20ano")
          test(model, test_X[2],0, name="gru3_20norm")
```

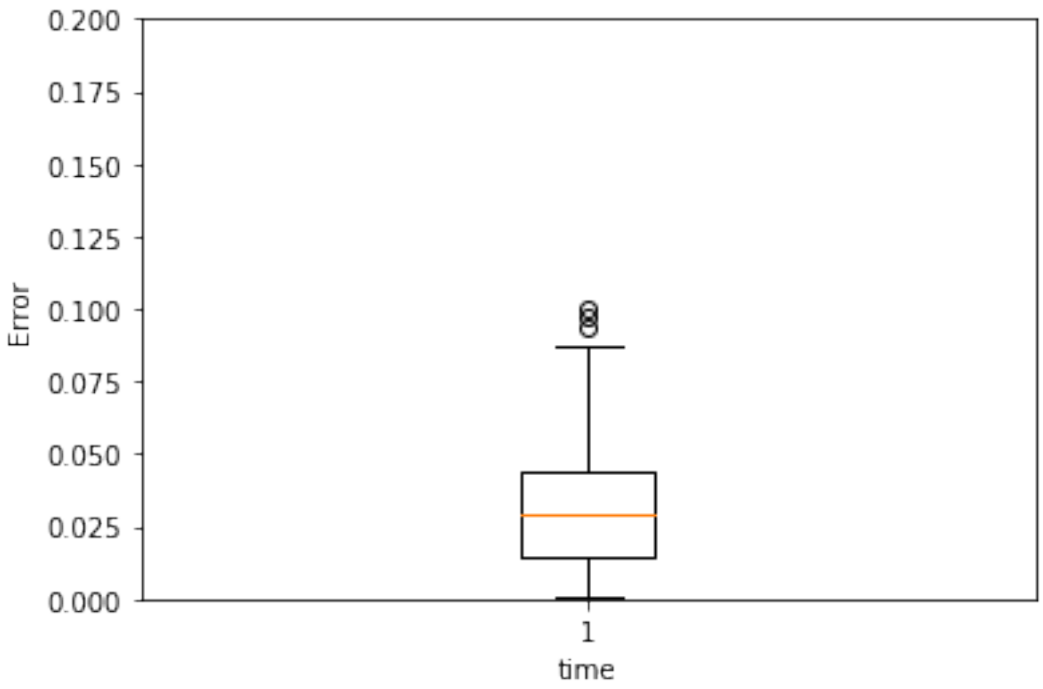


```
0.0329928523387
```



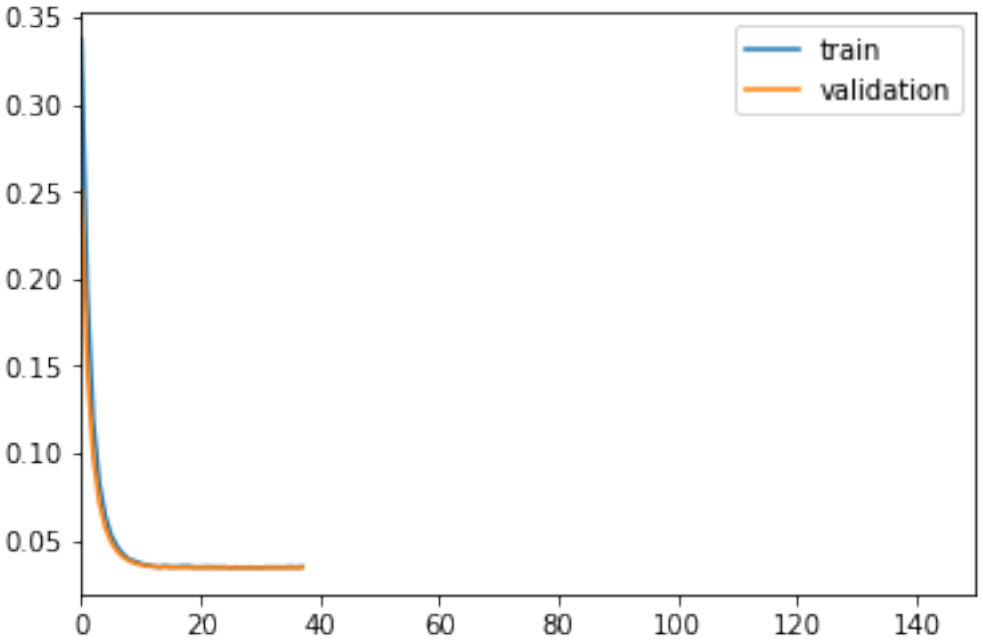


```
0.0300274486896
```



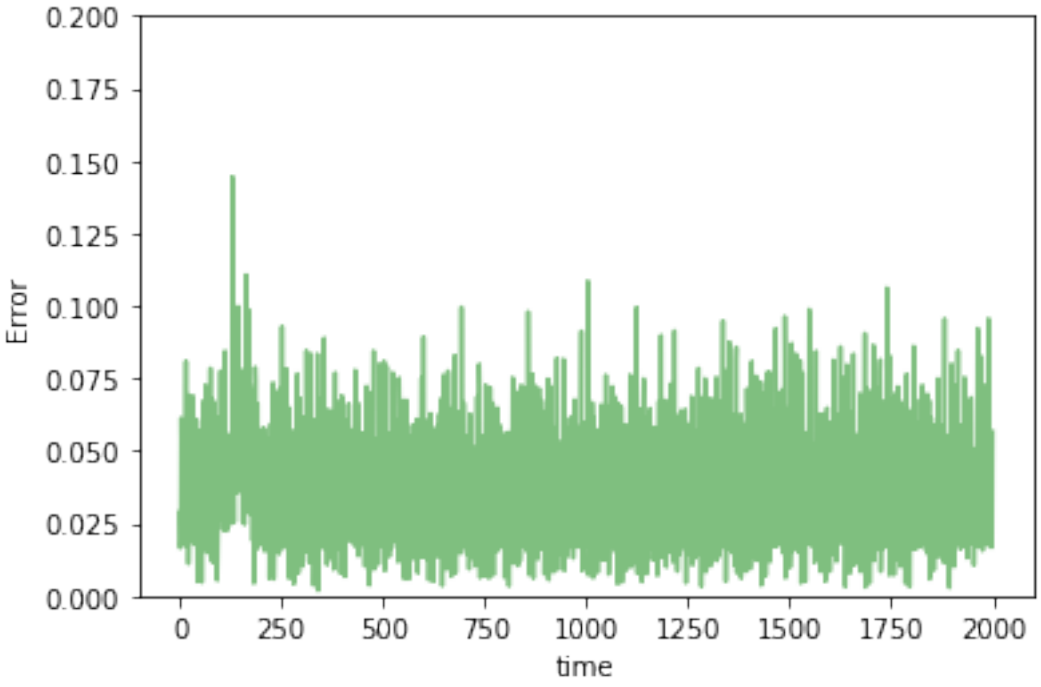
50 steps

```
In [205]: TIMESTEPS = 50
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [206]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(10, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(10, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [207]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [208]: train(model, tgen, vgen, name="gru3_50")
```

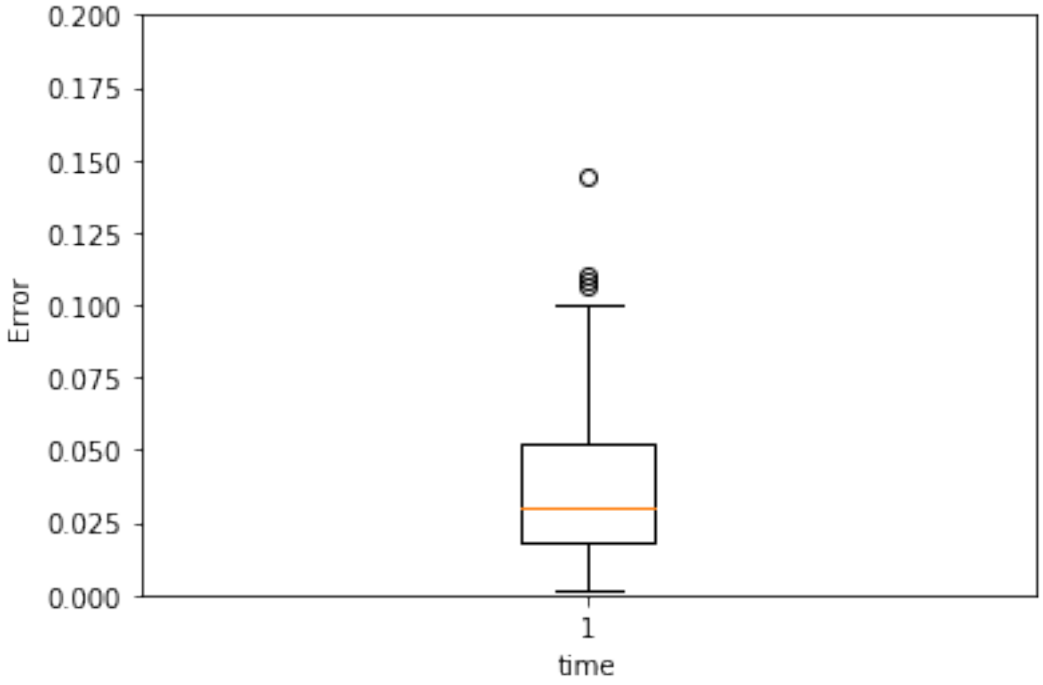


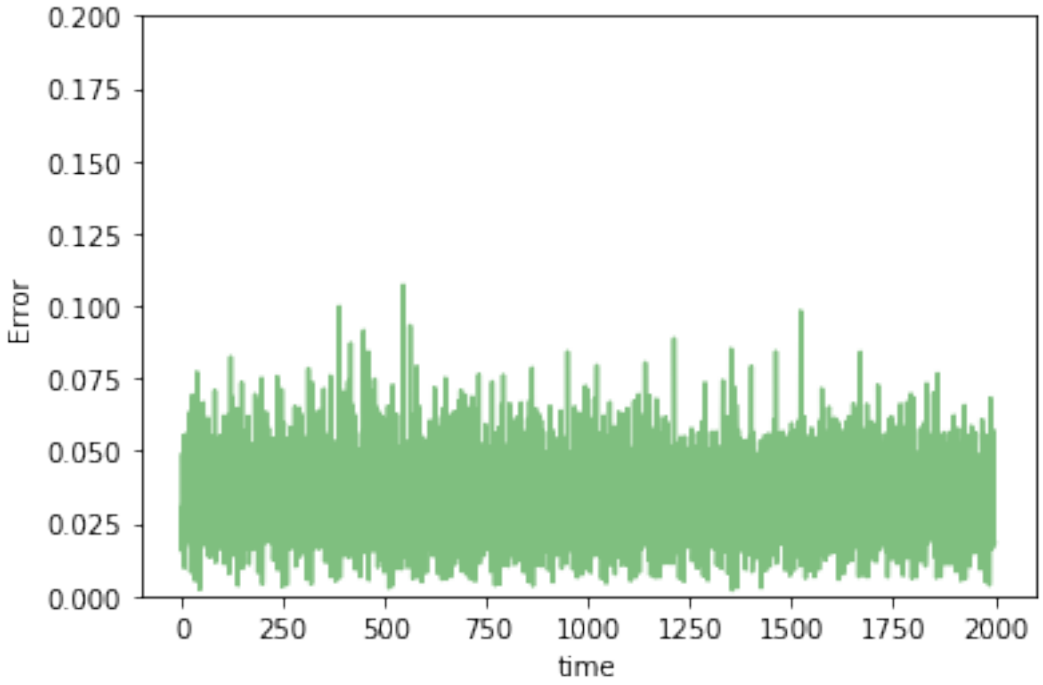

```
0.0349619682319
```

```
In [209]: test(model, test_X[0],0, name="gru3_50ano")
          test(model, test_X[2],0, name="gru3_50norm")
```

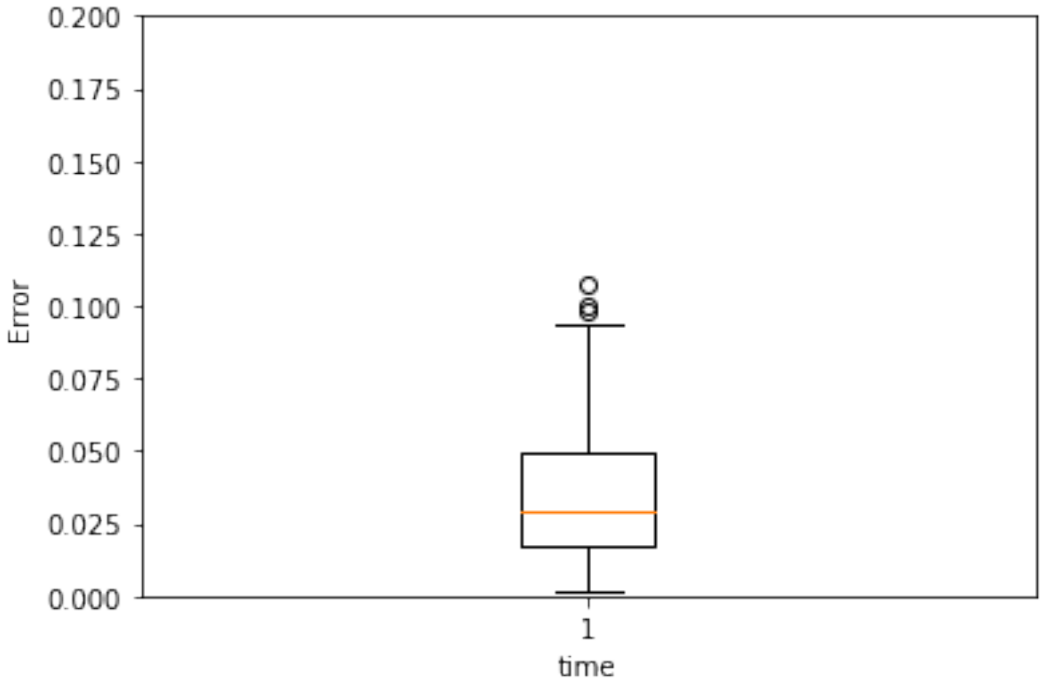


```
0.0359100282605
```





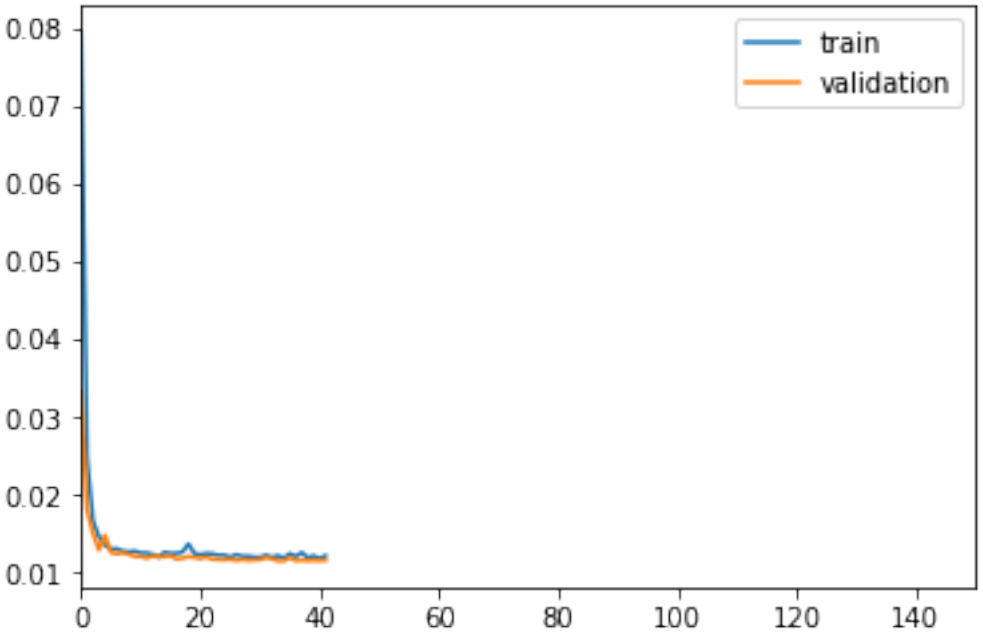
```
0.0335816419064
```



2.1.8 RNN with 4 GRU layers dim compression.

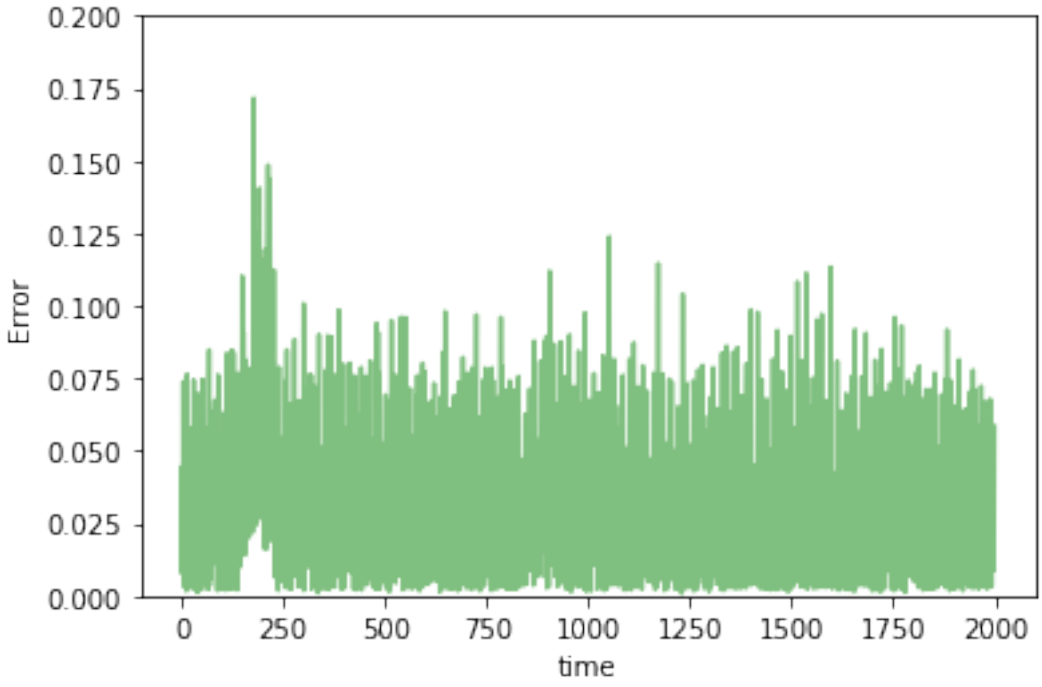
2 steps

```
In [210]: TIMESTEPS = 2
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [211]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(7, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(5, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(DIM, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [212]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [213]: train(model, tgen, vgen, name="gru4_2")
```

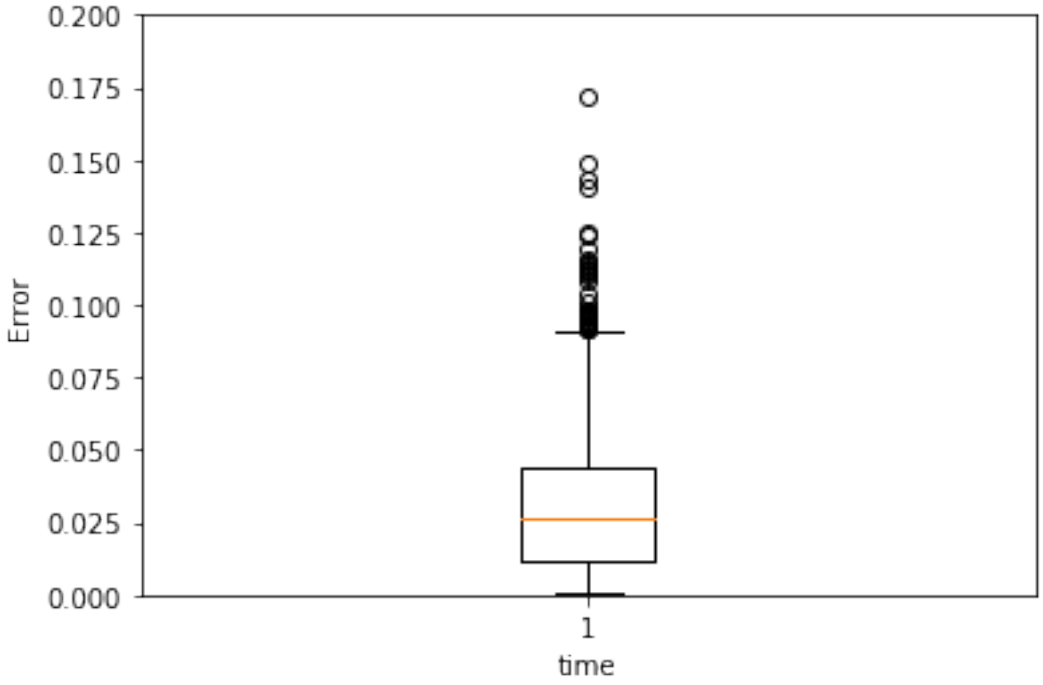


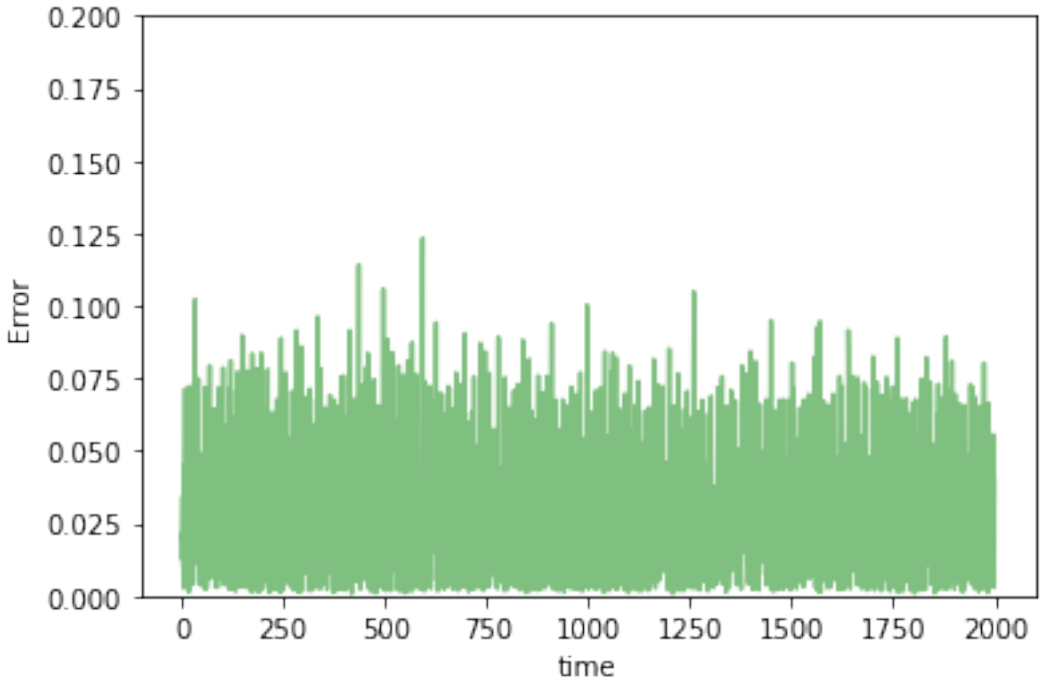

```
0.0121246495842
```

```
In [214]: test(model, test_X[0],0, name="gru4_2ano")
          test(model, test_X[2],0, name="gru4_2norm")
```

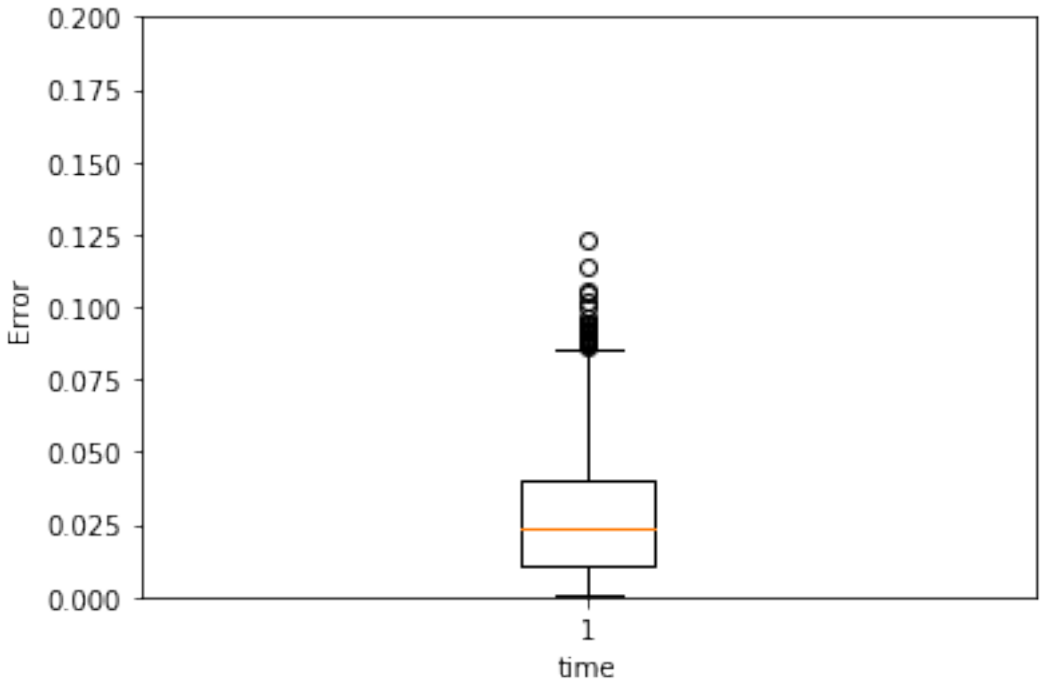


```
0.0312103579622
```



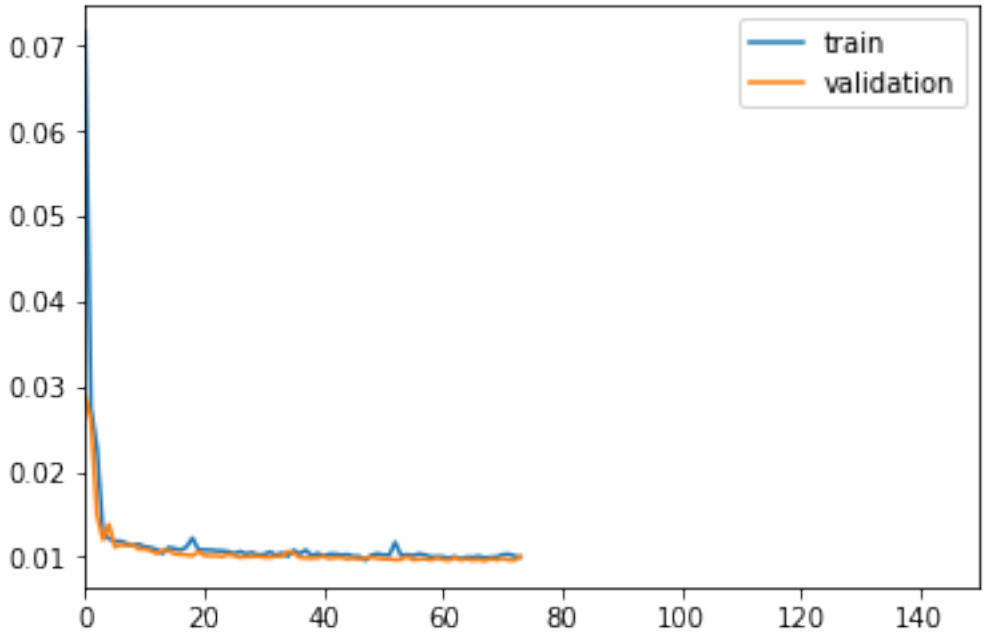


```
0.0284968367952
```



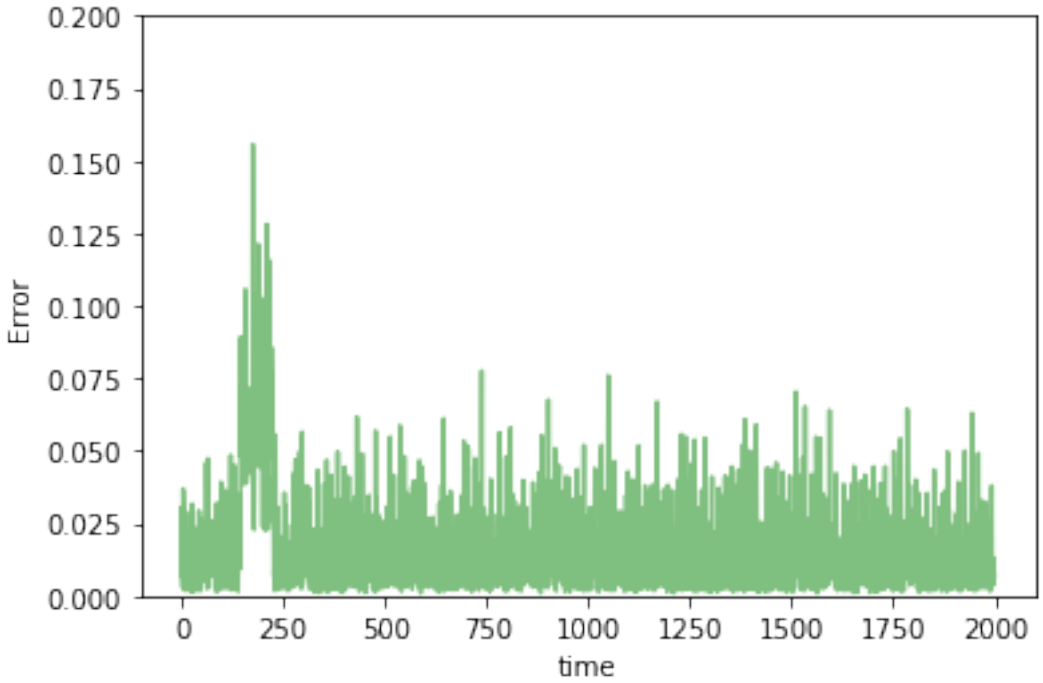
5 steps

```
In [215]: TIMESTEPS = 5
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [216]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(7, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(5, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(DIM, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [217]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [218]: train(model, tgen, vgen, name="gru4_5")
```

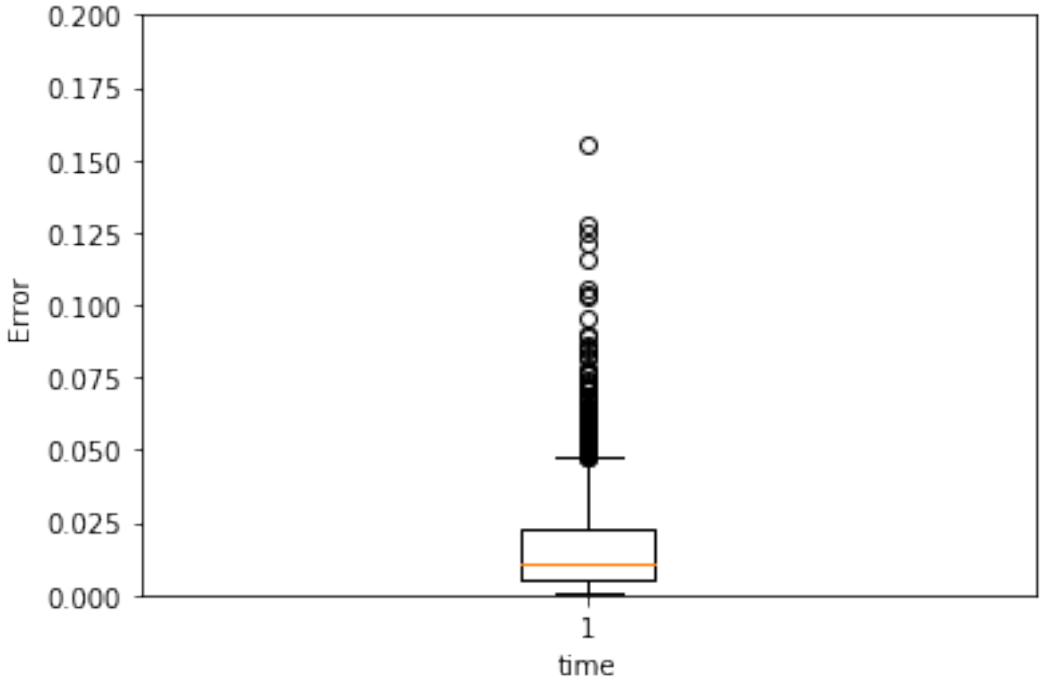


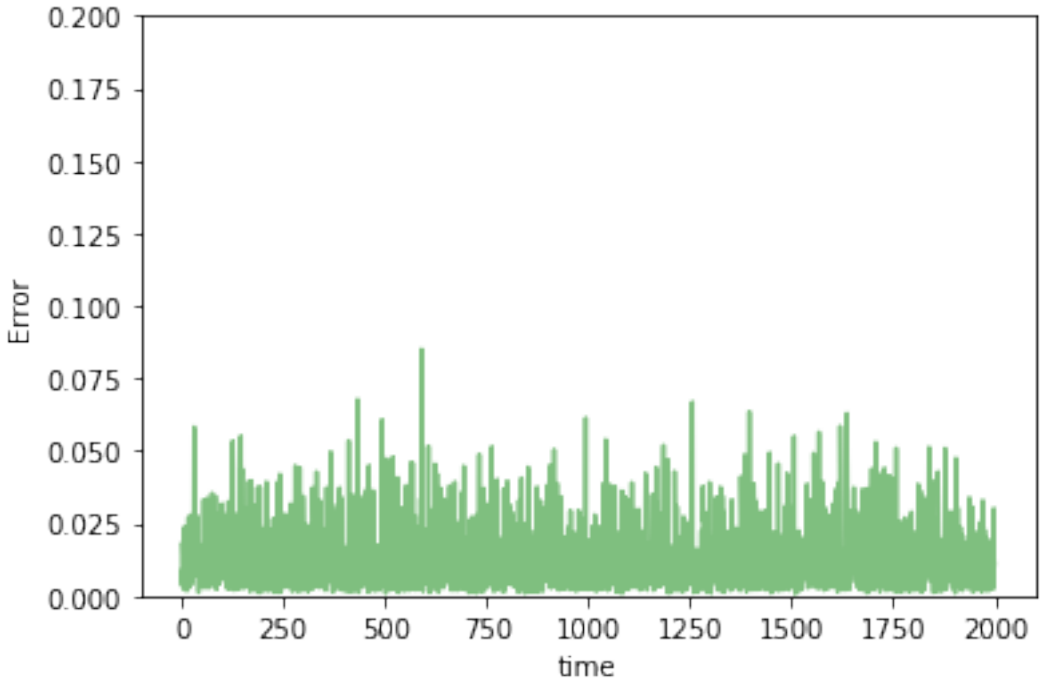

```
0.00997376100905
```

```
In [219]: test(model, test_X[0],0, name="gru4_5ano")
          test(model, test_X[2],0, name="gru4_5norm")
```

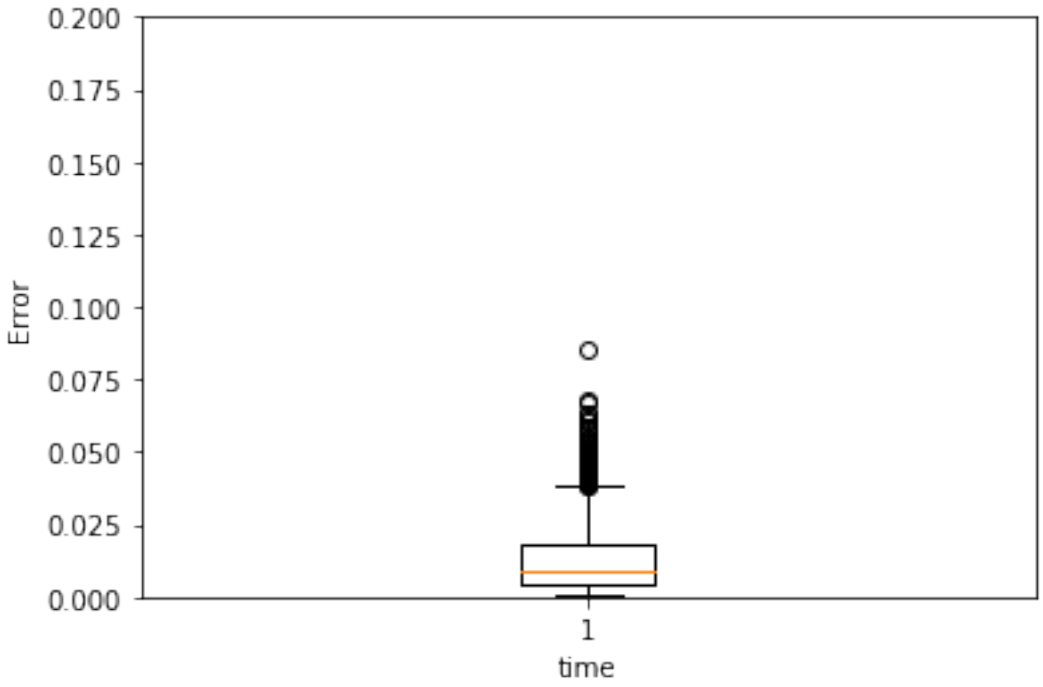


```
0.0164295316071
```



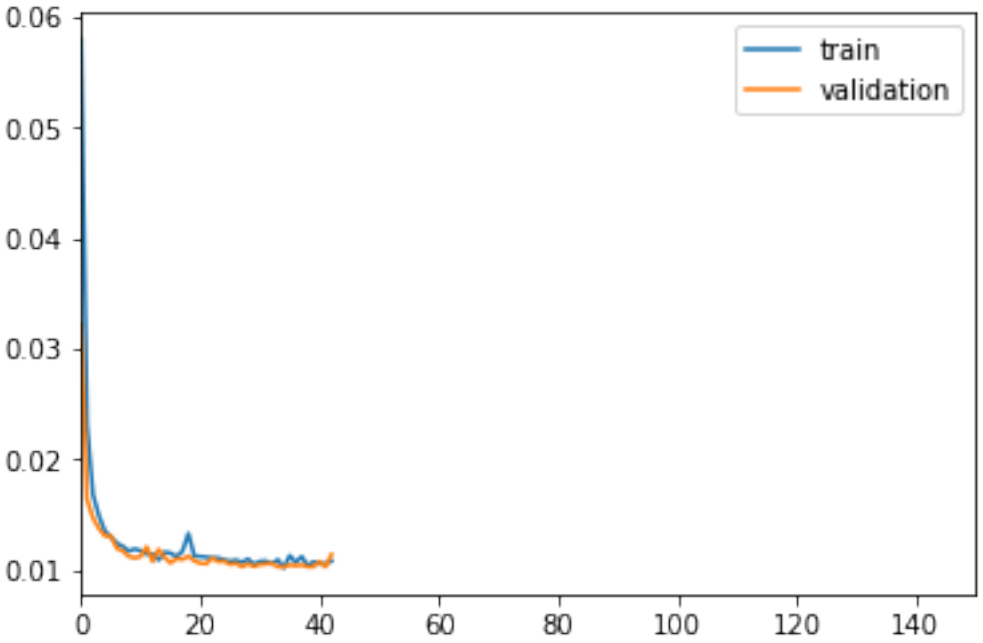


```
0.012975248915
```



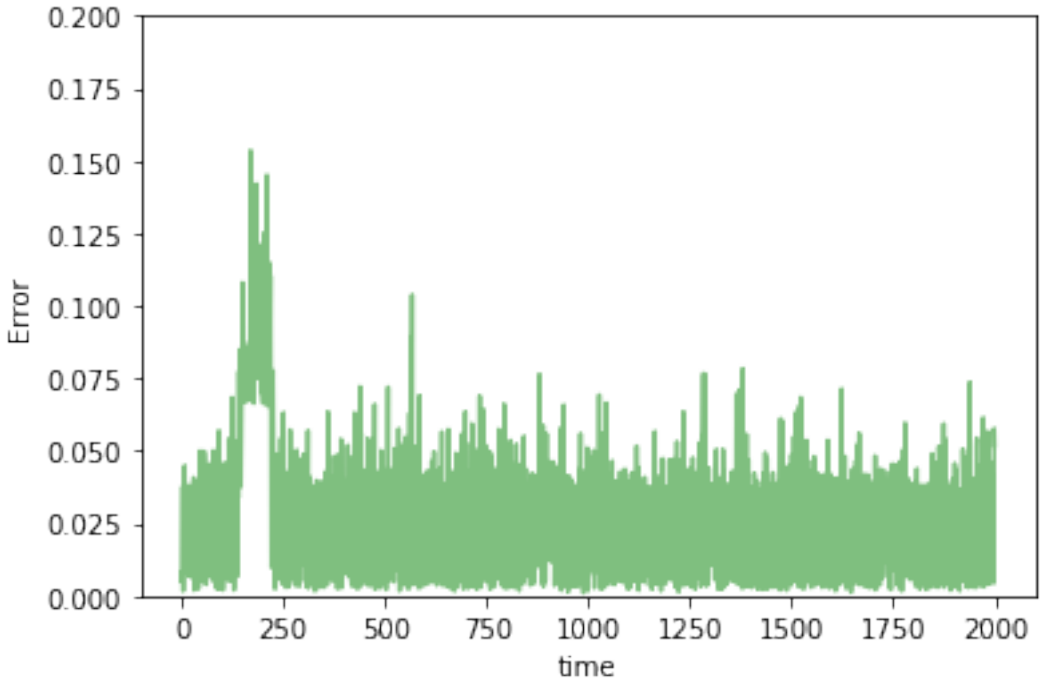
10 steps

```
In [220]: TIMESTEPS = 10
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS, 0)
          vgen = flat_generator(val_X, TIMESTEPS, 0)

In [221]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(7, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(5, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(DIM, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [222]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [223]: train(model, tgen, vgen, name="gru4_10")
```

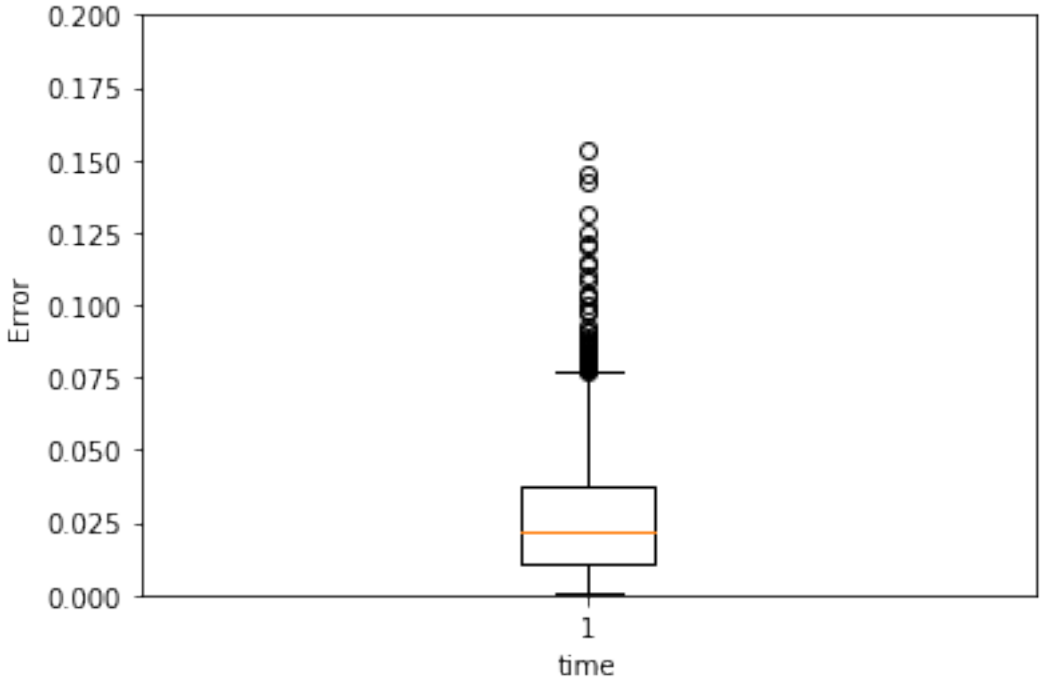


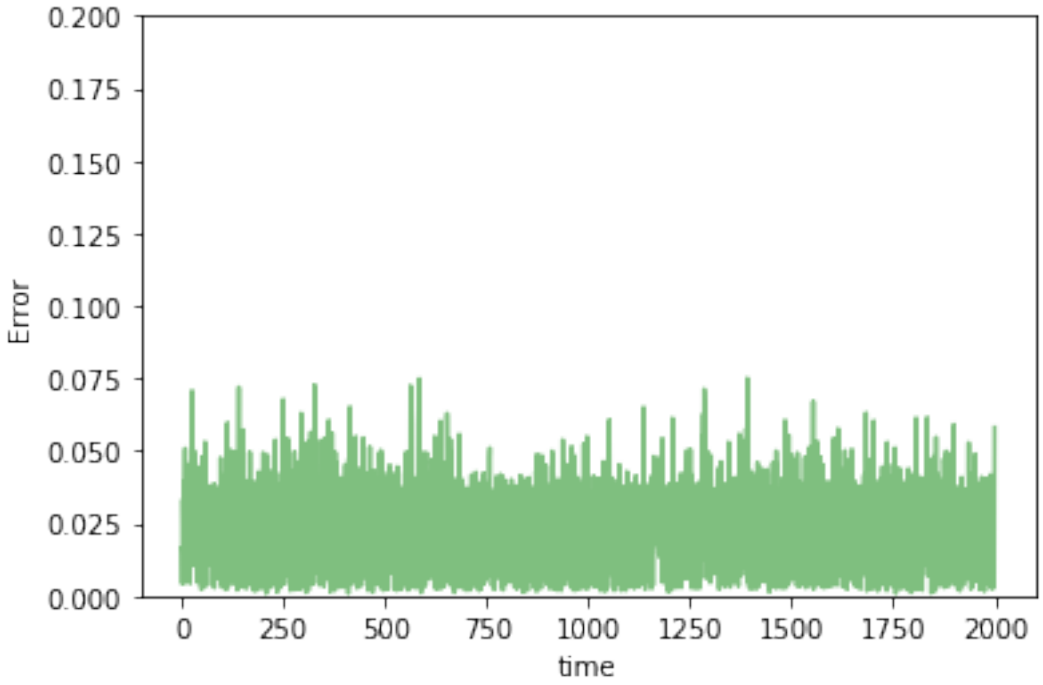

```
0.0107966393562
```

```
In [224]: test(model, test_X[0],0, name="gru4_10ano")
          test(model, test_X[2],0, name="gru4_10norm")
```

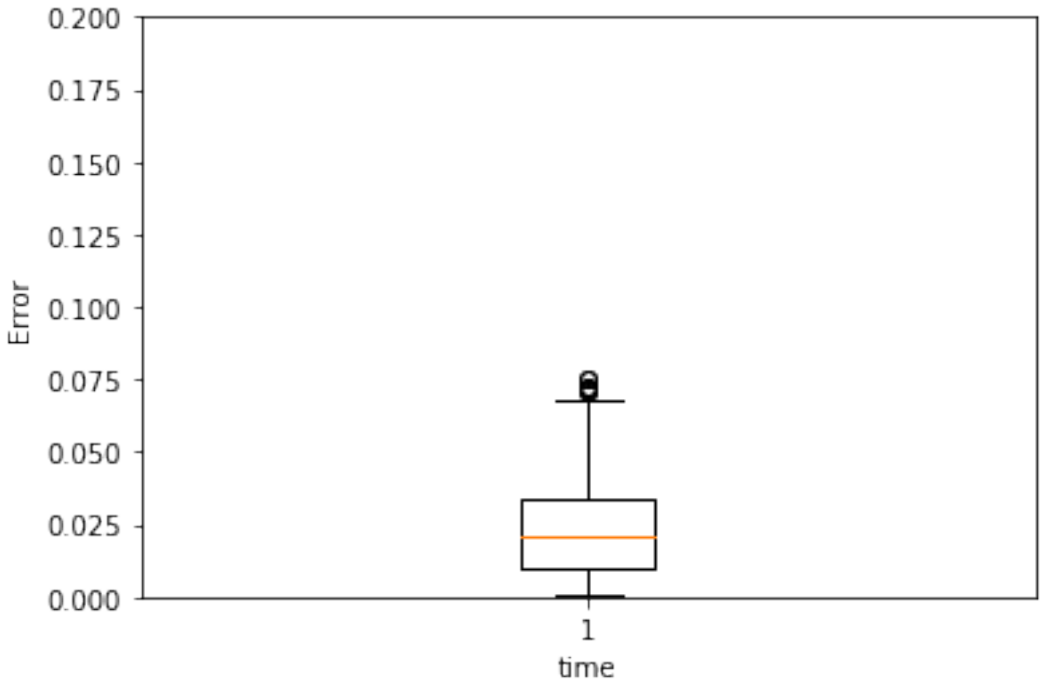


```
0.0261452149673
```



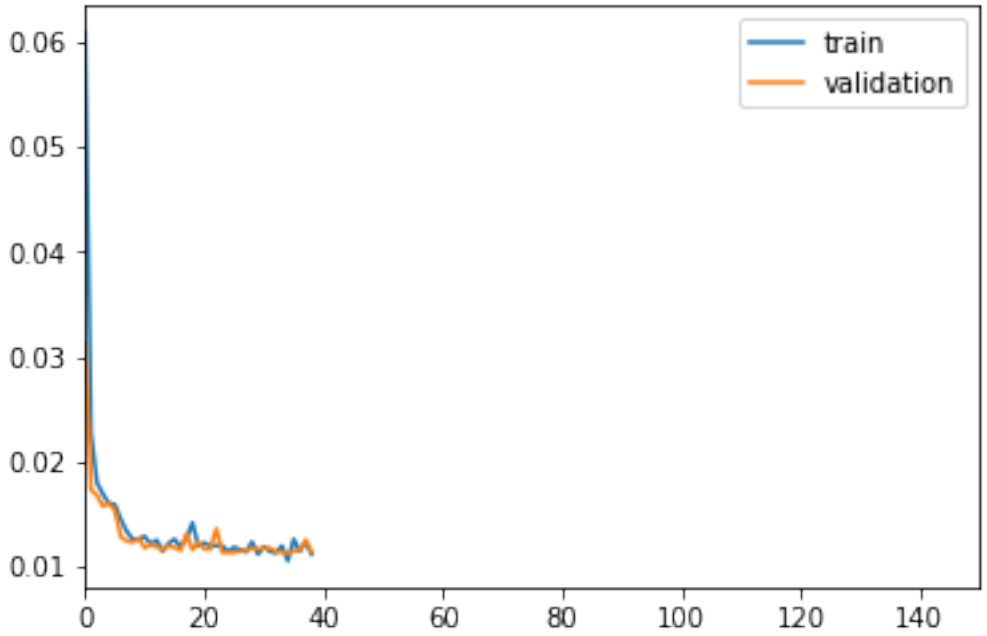


```
0.0222445057809
```



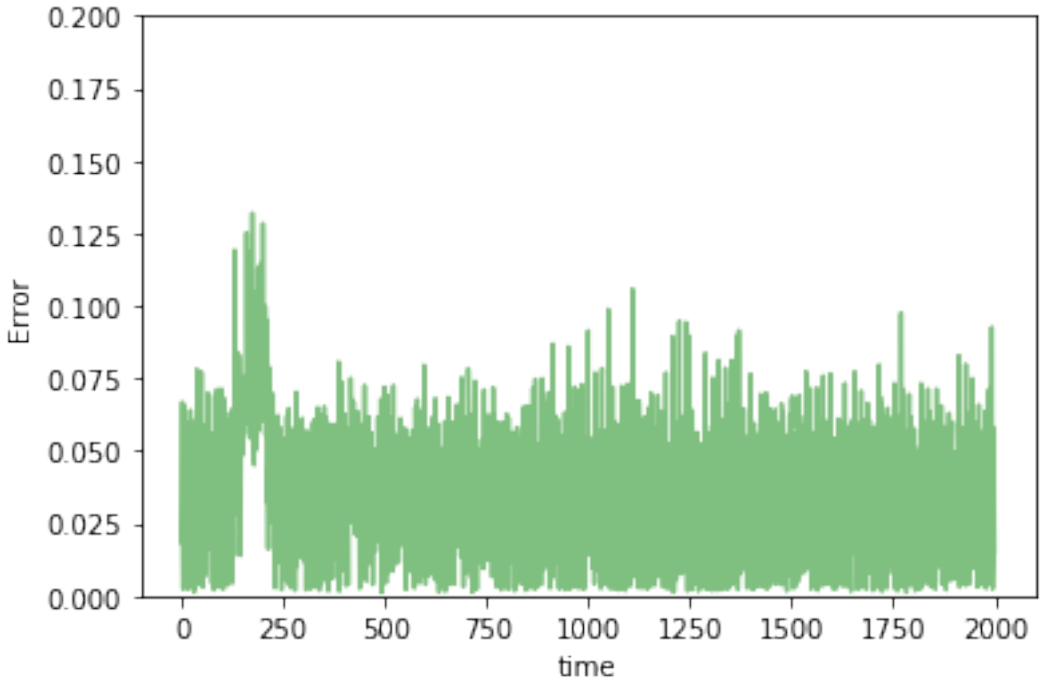
20 steps

```
In [225]: TIMESTEPS = 20
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [226]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(7, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(5, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(DIM, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [227]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [228]: train(model, tgen, vgen, name="gru4_20")
```

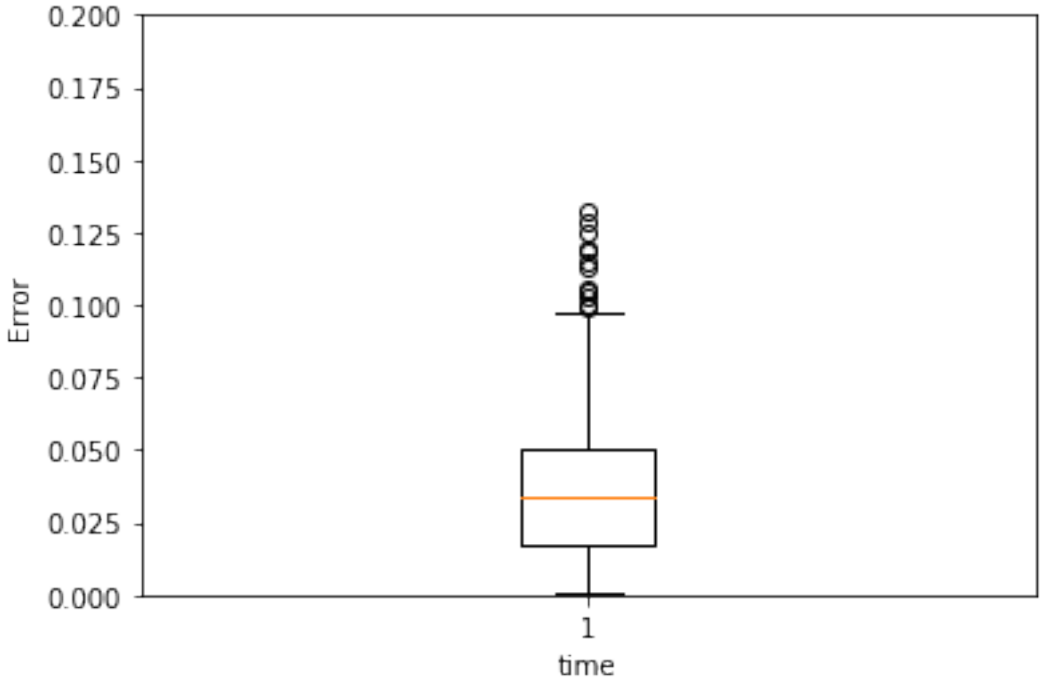


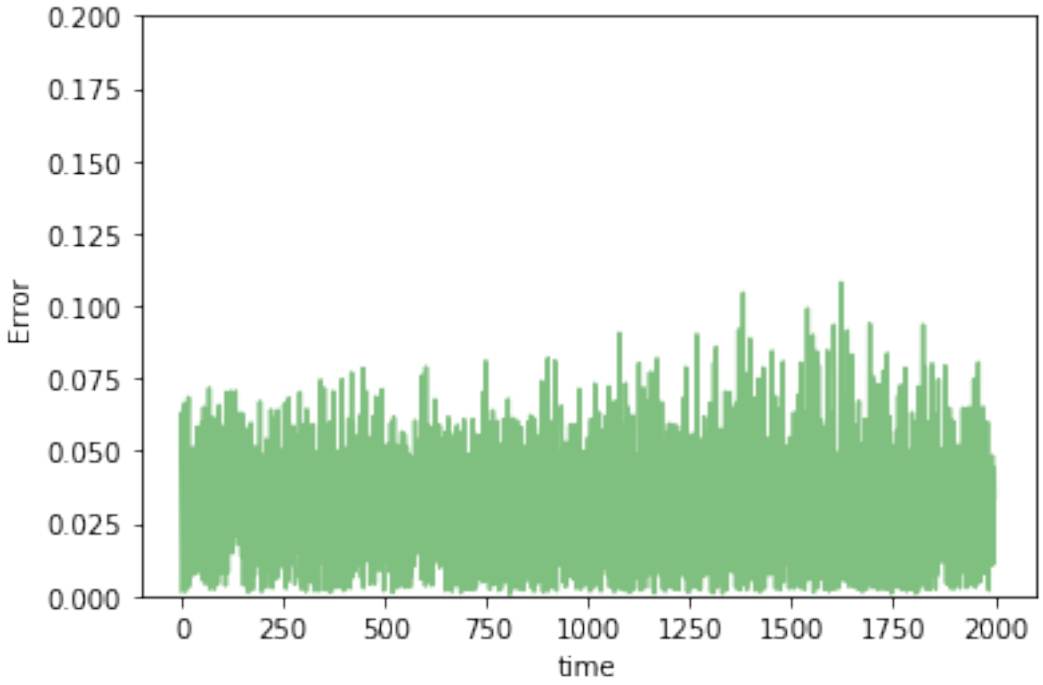

```
0.0111674214085
```

```
In [229]: test(model, test_X[0],0, name="gru4_20ano")
          test(model, test_X[2],0, name="gru4_20norm")
```

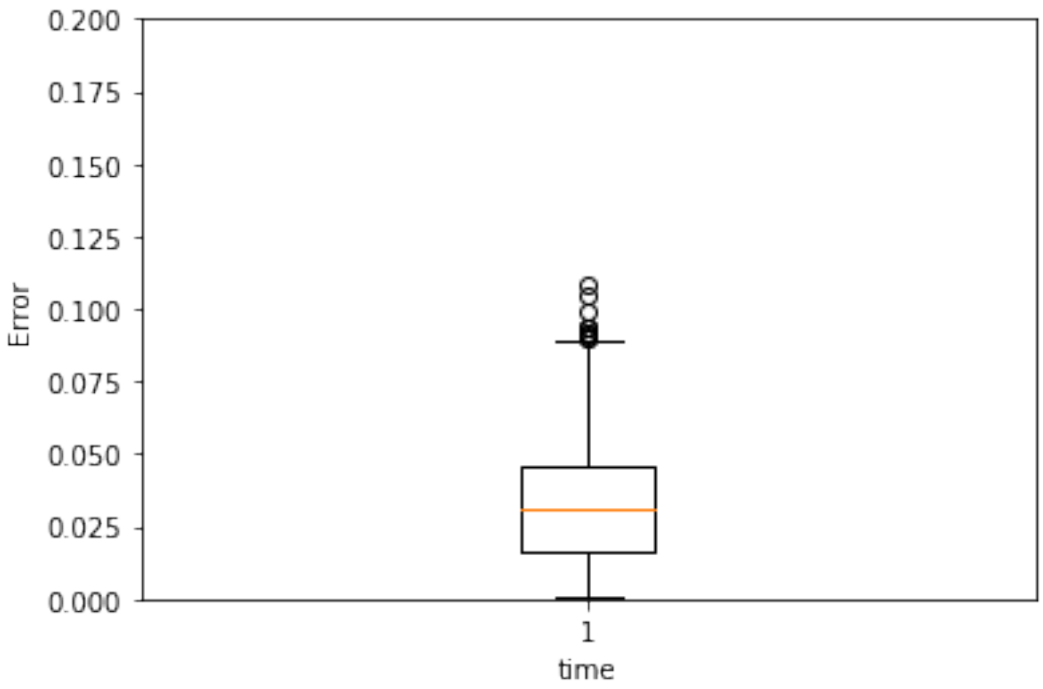


```
0.0352068870398
```



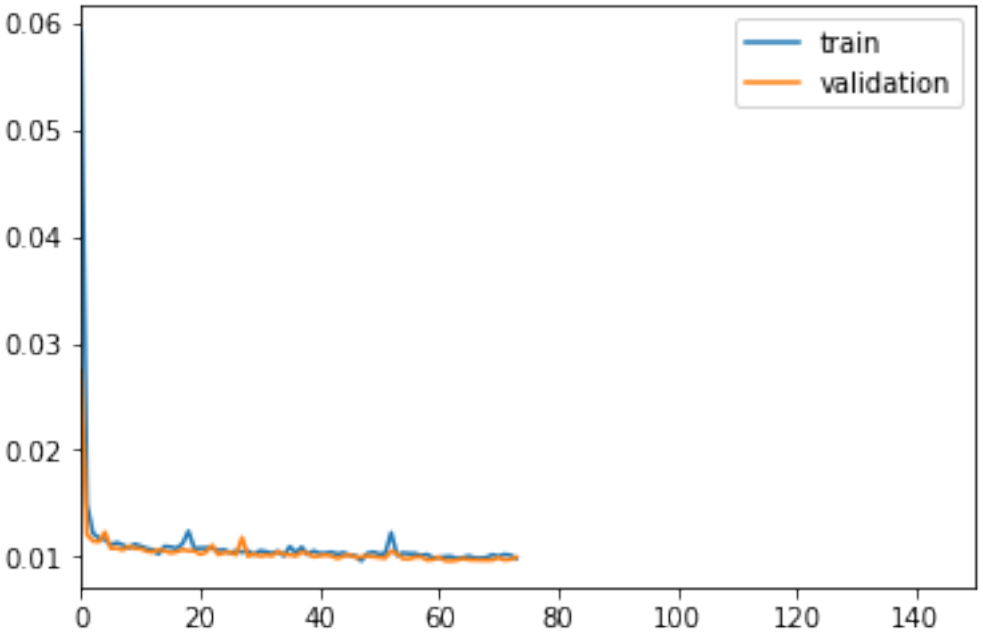


```
0.0319378132602
```



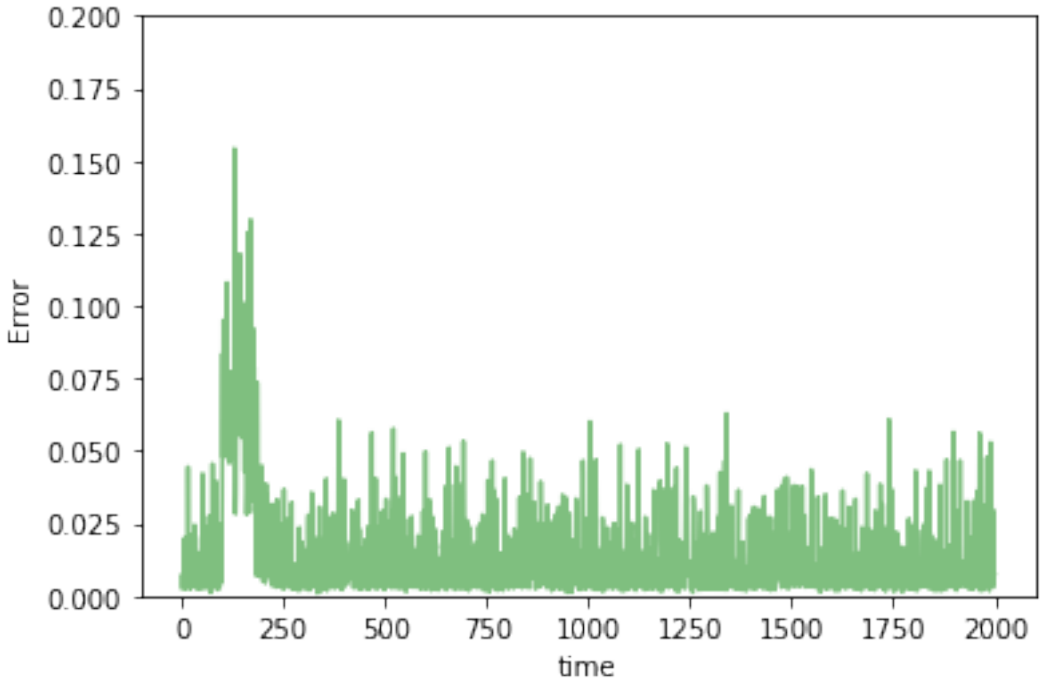
50 steps

```
In [230]: TIMESTEPS = 50
          DIM = 29
          tgen = flat_generator(X, TIMESTEPS,0)
          vgen = flat_generator(val_X, TIMESTEPS,0)

In [231]: input_layer = Input(shape=(TIMESTEPS,DIM))
          hidden = GRU(10, activation='relu', return_sequences=True)(input_layer)
          hidden = GRU(7, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(5, activation='relu', return_sequences=True)(hidden)
          hidden = GRU(DIM, activation='relu')(hidden)
          output = Dense(DIM, activation='sigmoid')(hidden)

In [232]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])

In [233]: train(model, tgen, vgen, name="gru4_50")
```

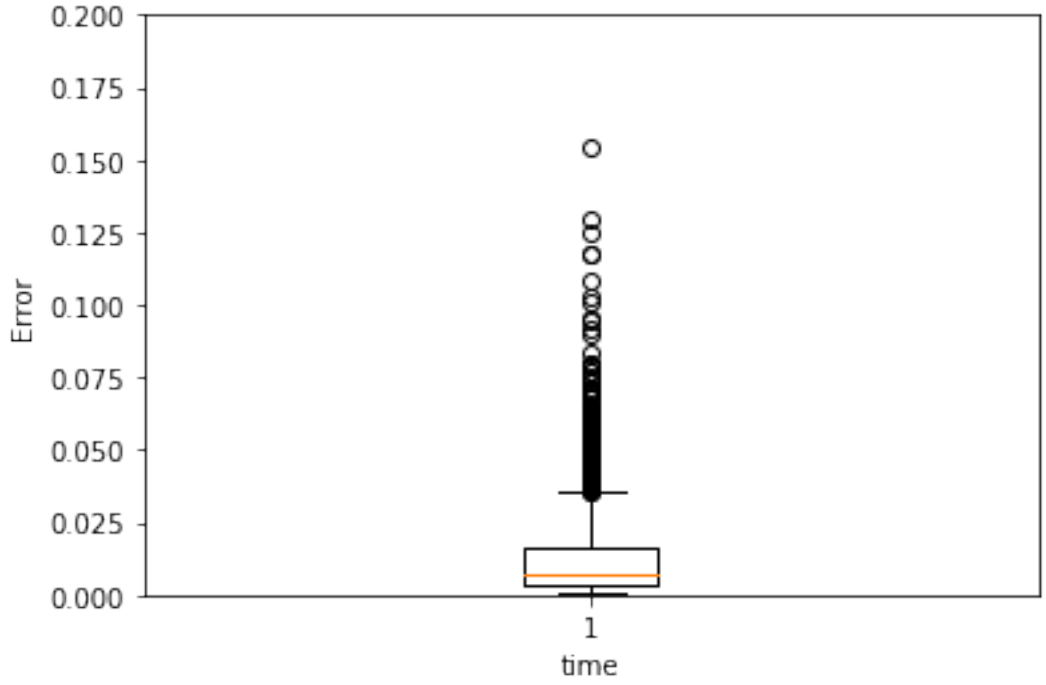


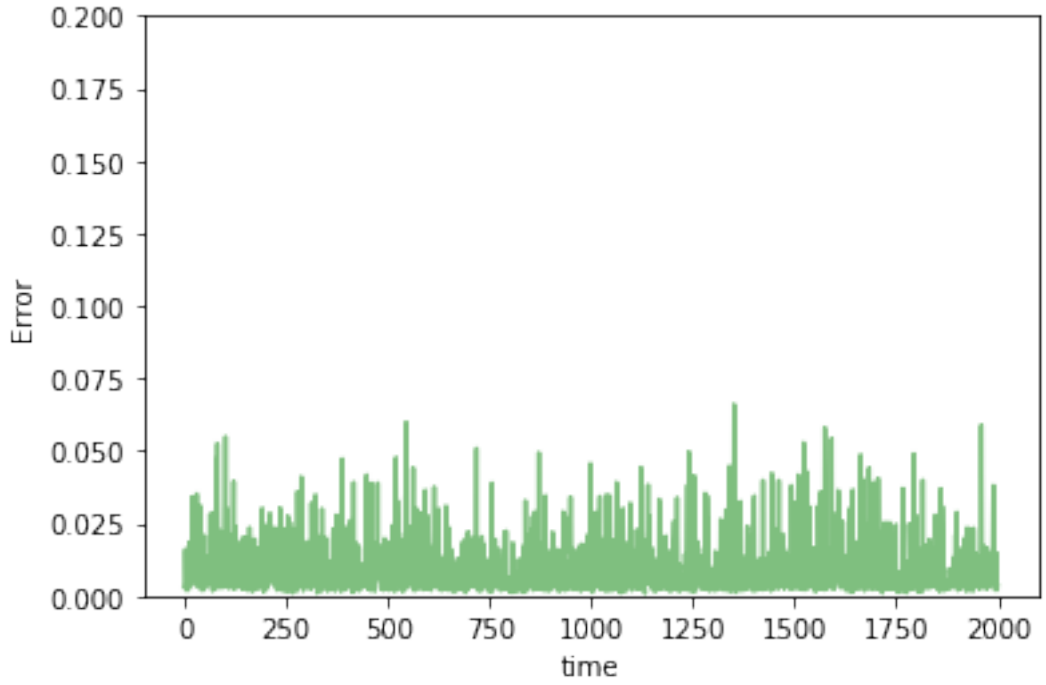

```
0.00982600809634
```

```
In [234]: test(model, test_X[0],0, name="gru4_50ano")
          test(model, test_X[2],0, name="gru4_50norm")
```



```
0.0132422923708
```





```
0.00957613199334
```