ansi_regression

March 25, 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_
```

```
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')
```

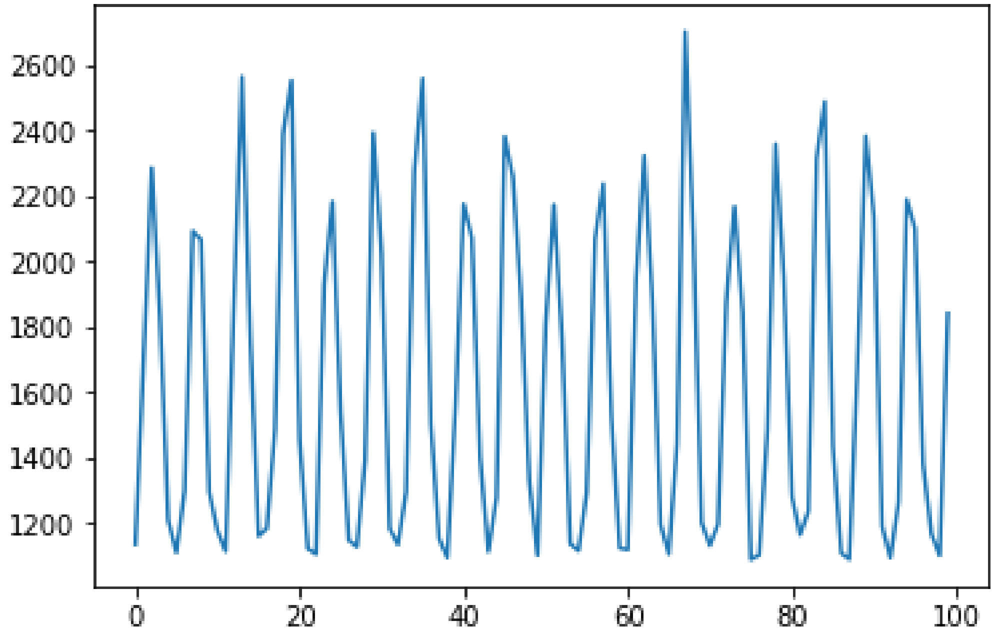
1.7 Get data

```
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
```

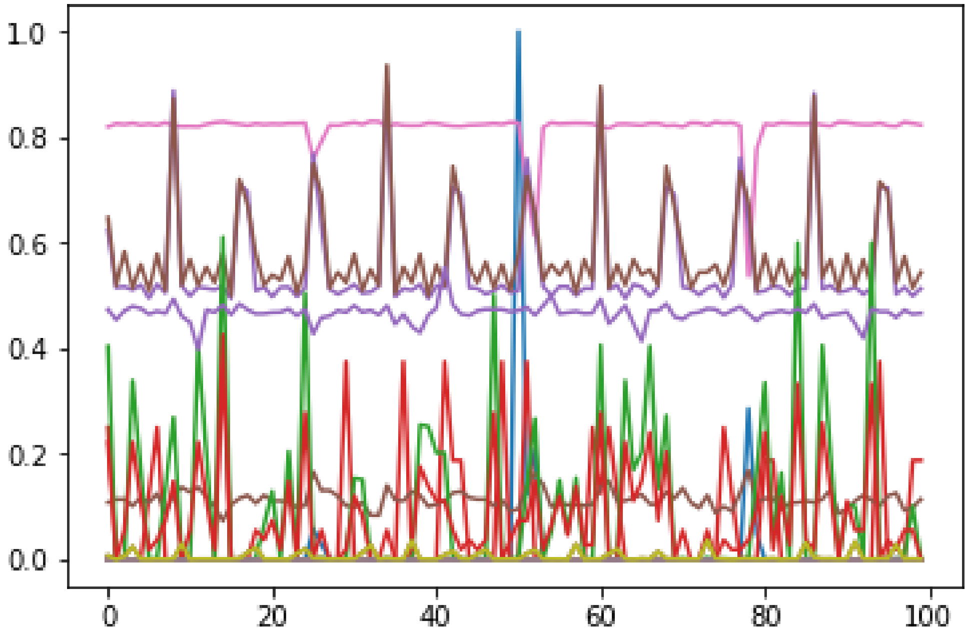
1.9 Correrlation measurement

---

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()
```



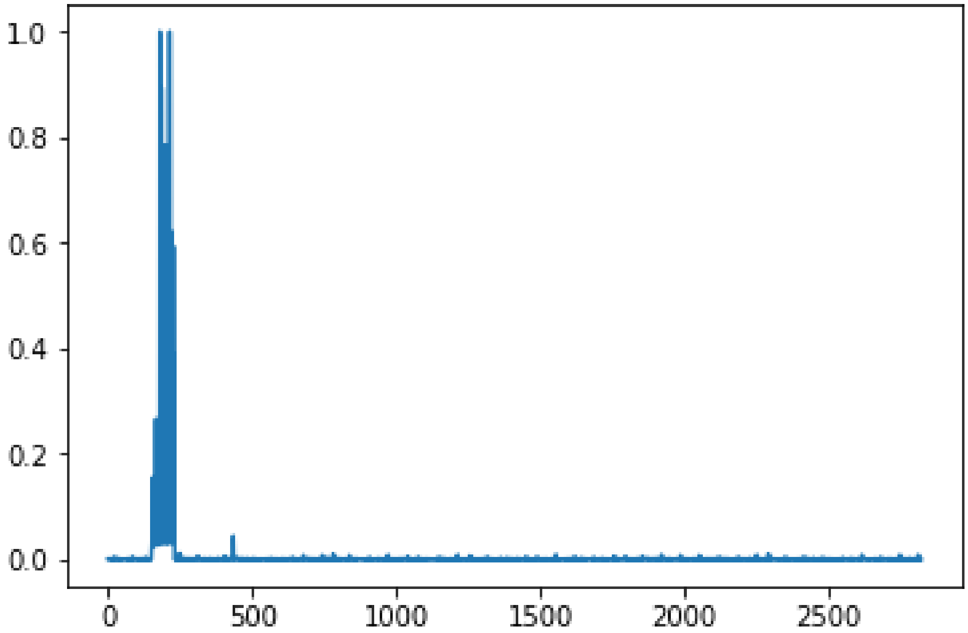
```
In [28]: print(X.shape)
         LEN = X.shape[0]
         SPLIT = int(0.9*LEN)

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

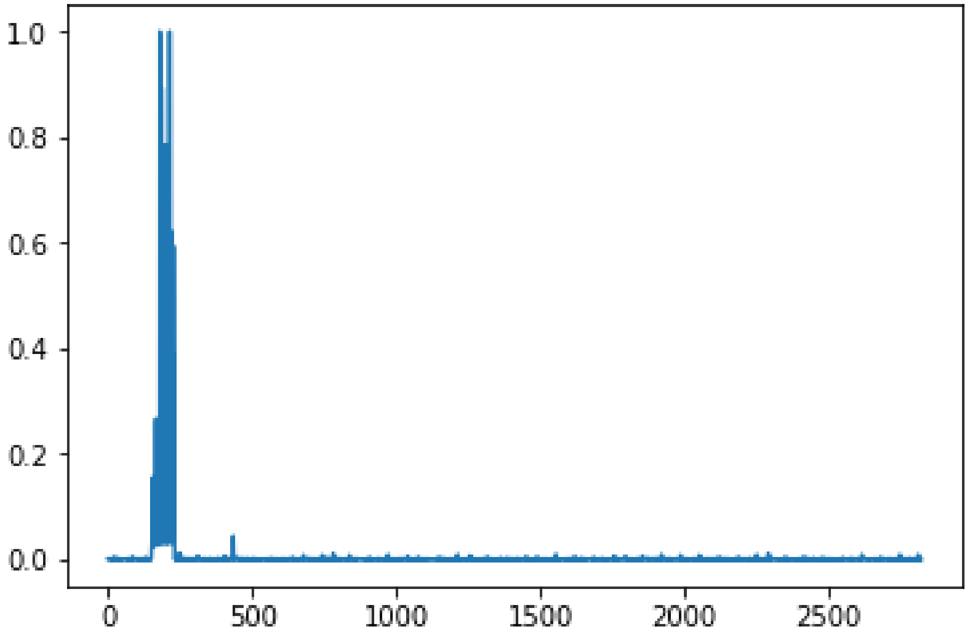
```
(38200, 3, 29)
```

```
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):
             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()
             print(history.history['loss'][-1])
```

```
In [34]: def test(model, dataset=test_X[0], ravel=1):
             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=0.9)
             plotter.ylim(0,0.2)
             plotter.show()
             error = numpy.array(error)
             print(numpy.mean(error))
             plotter.boxplot(error)
             plotter.ylim(0,0.2)
             plotter.show()
             #print(error)
```

2.1.1 Linear Regression

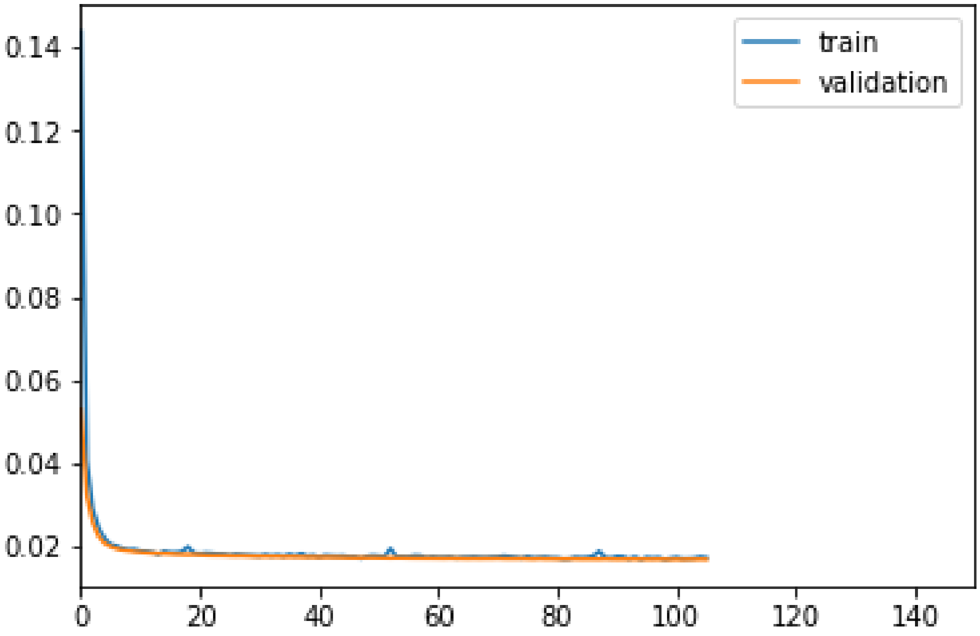
2 steps

```
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)
```

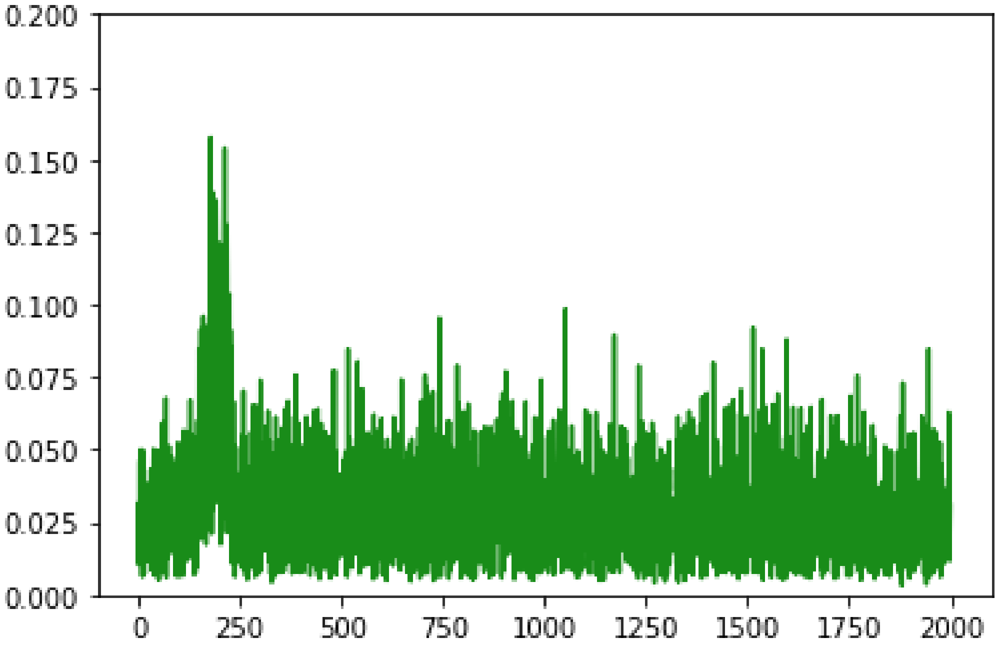
```
In [37]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [38]: train(model, tgen, vgen)
```

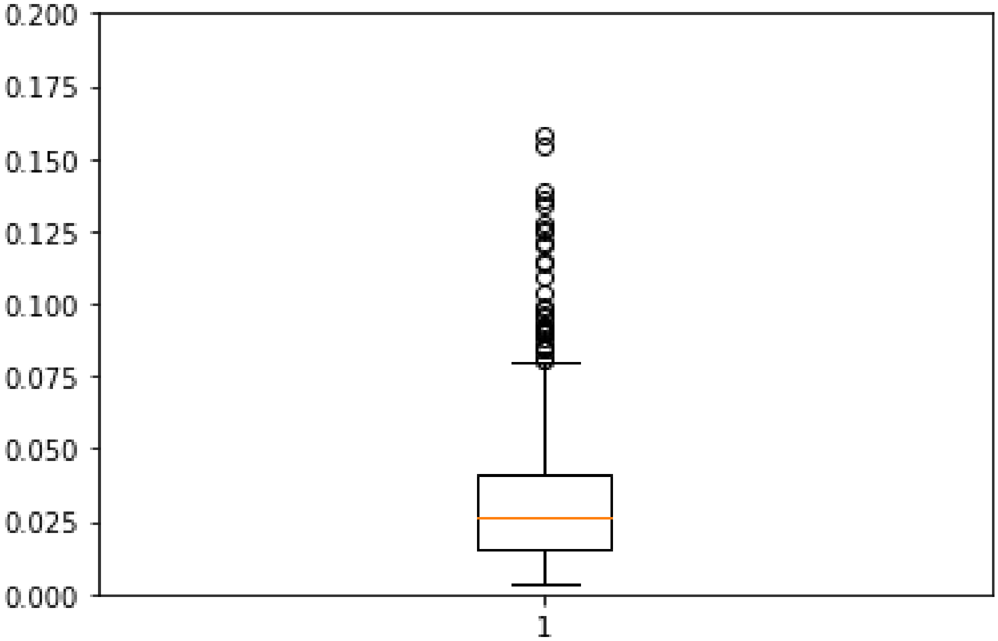


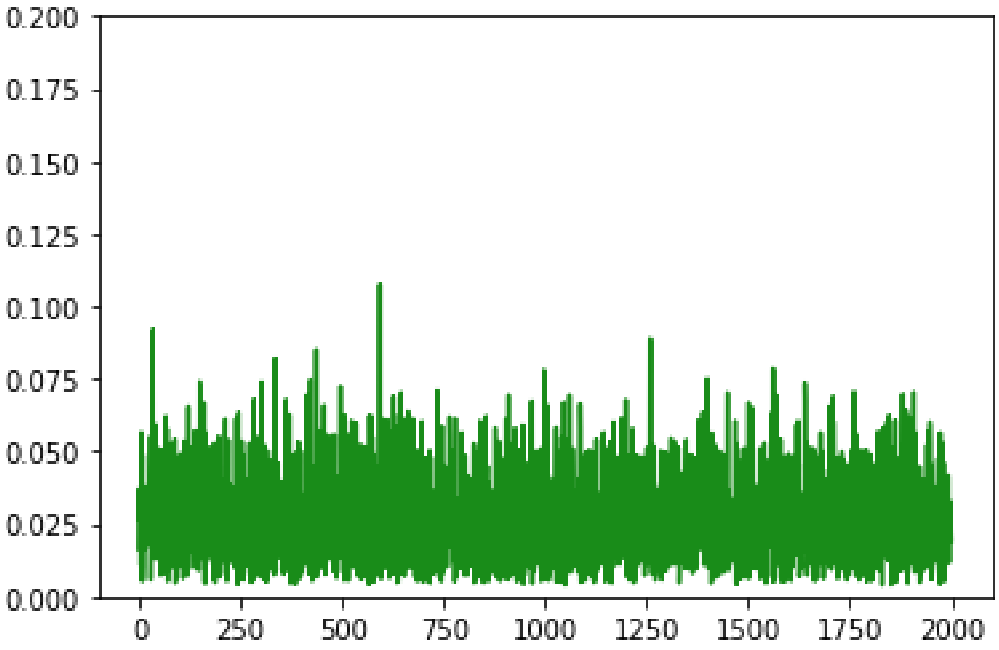
```
0.0172959613965
```

```
In [39]: test(model, test_X[0])
         test(model, test_X[2])
```

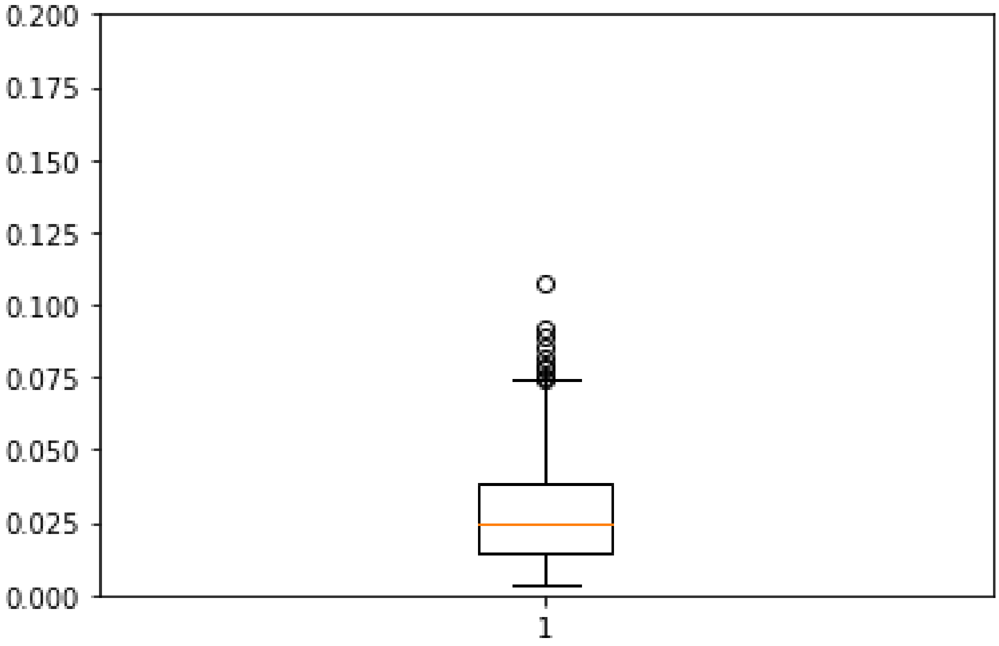


```
0.0300314289002
```





```
0.0273182739825
```



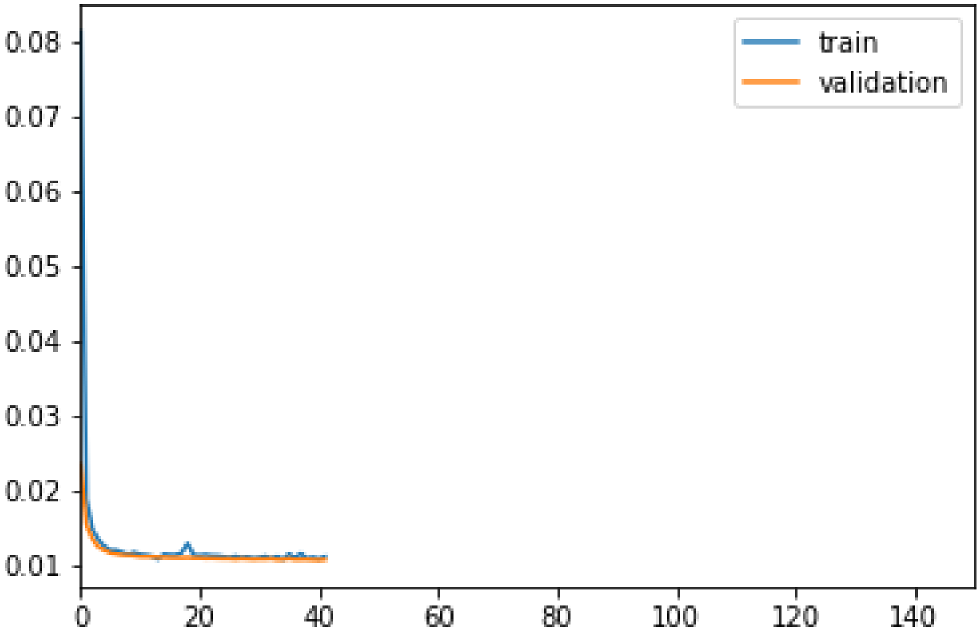
5 steps

```
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)
```

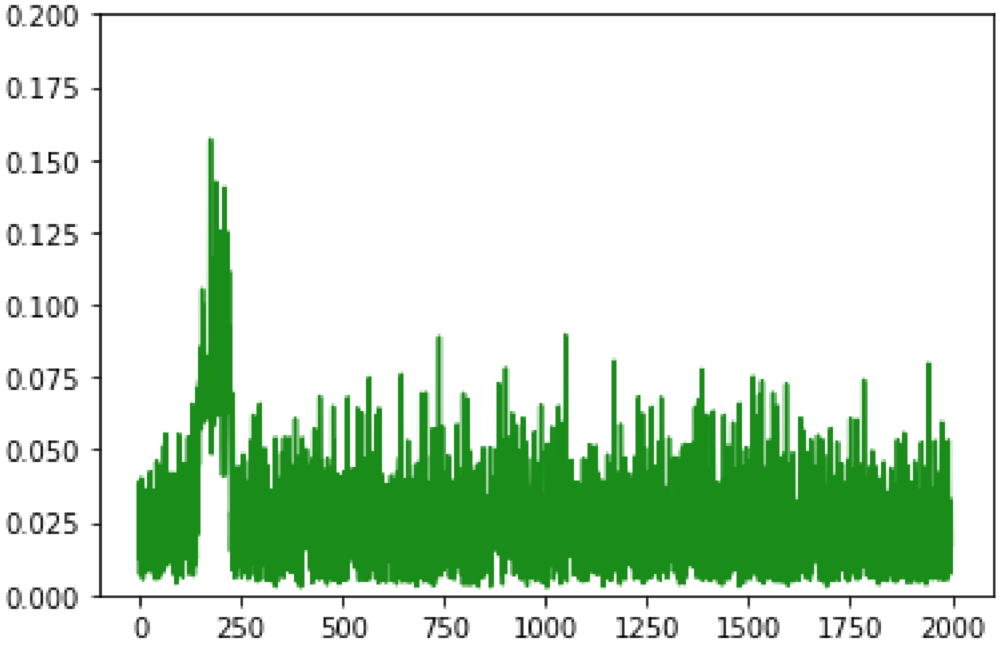
```
In [42]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [43]: train(model, tgen, vgen)
```

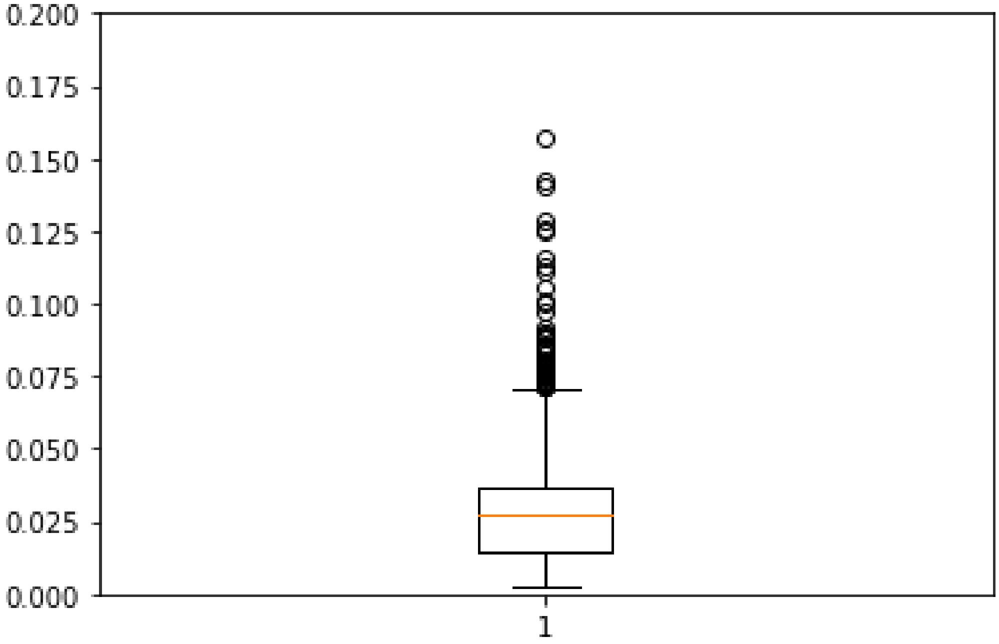


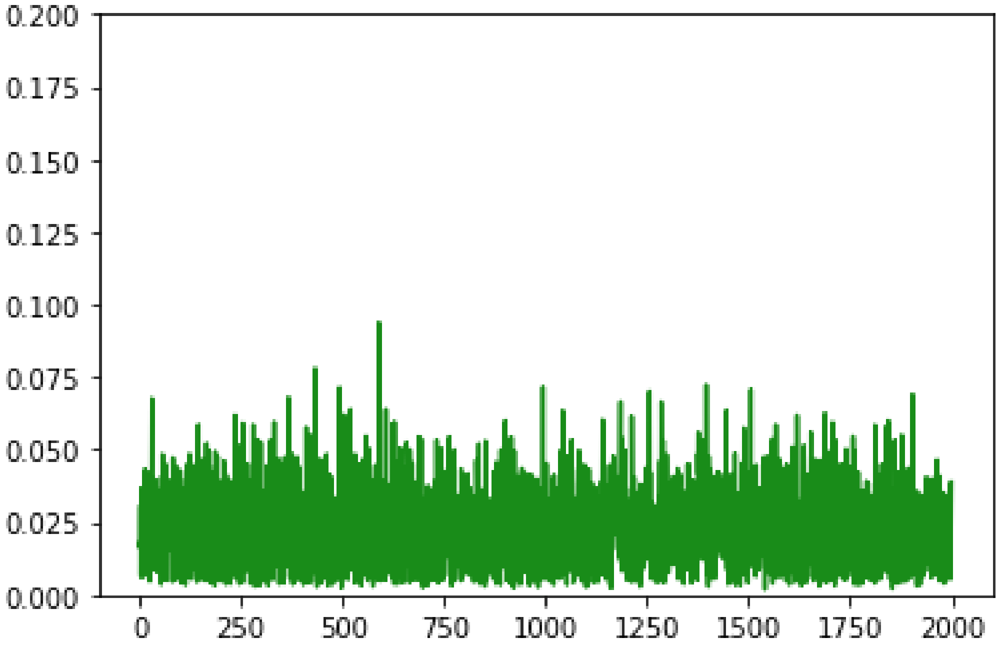
```
0.0110419096758
```

```
In [44]: test(model, test_X[0])
         test(model, test_X[2])
```

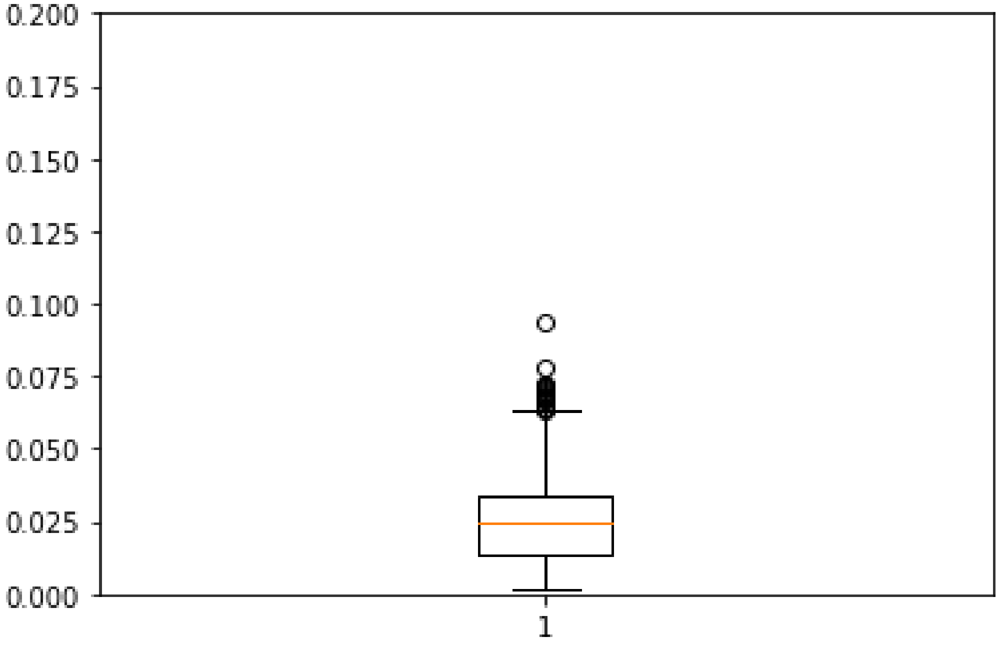


```
0.0282451679227
```





```
0.0246576860422
```



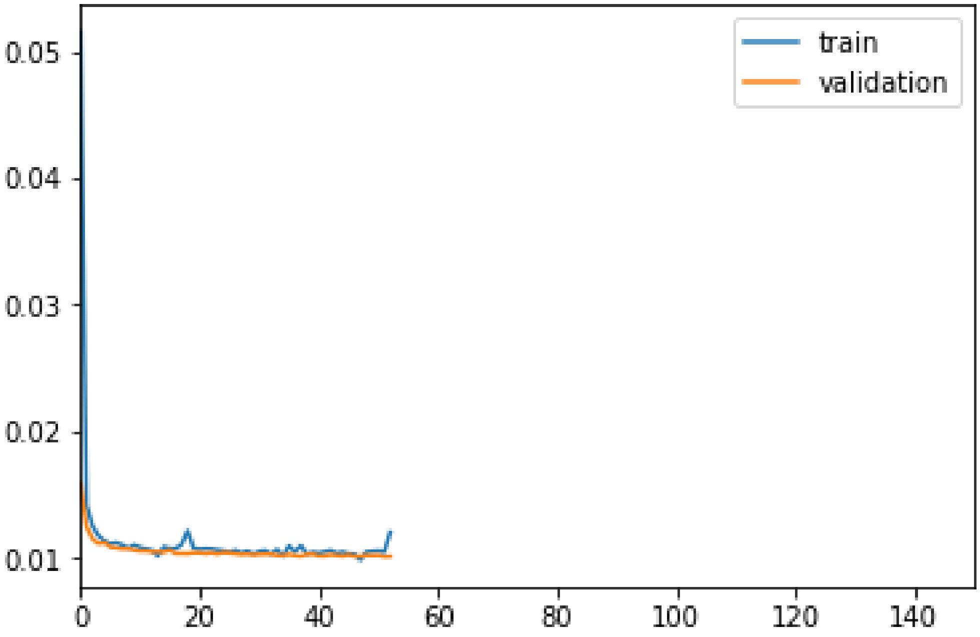
10 steps

```
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)
```

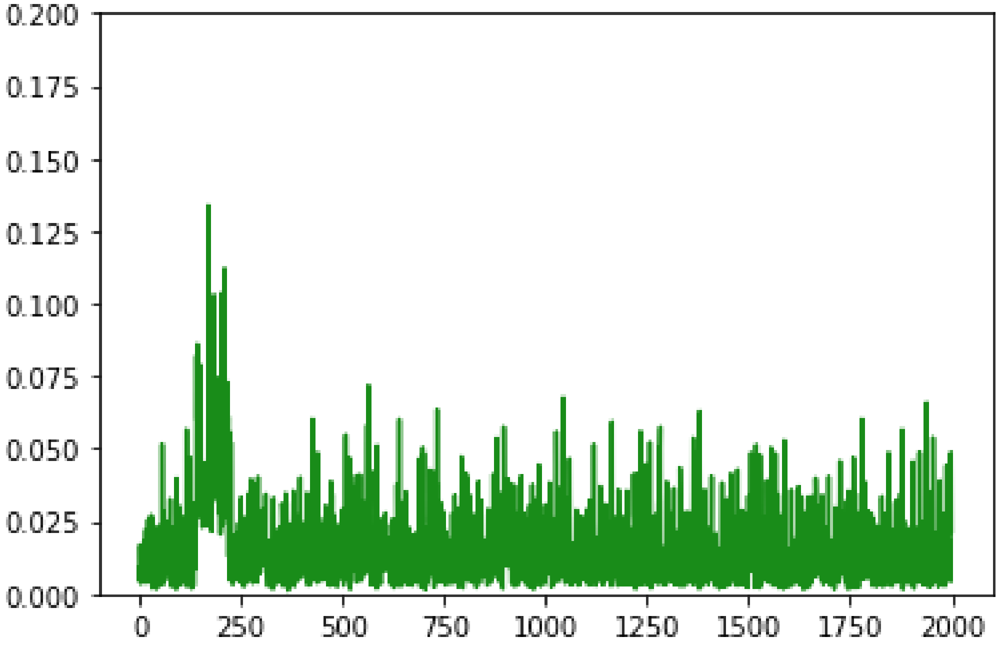
```
In [47]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [48]: train(model, tgen, vgen)
```

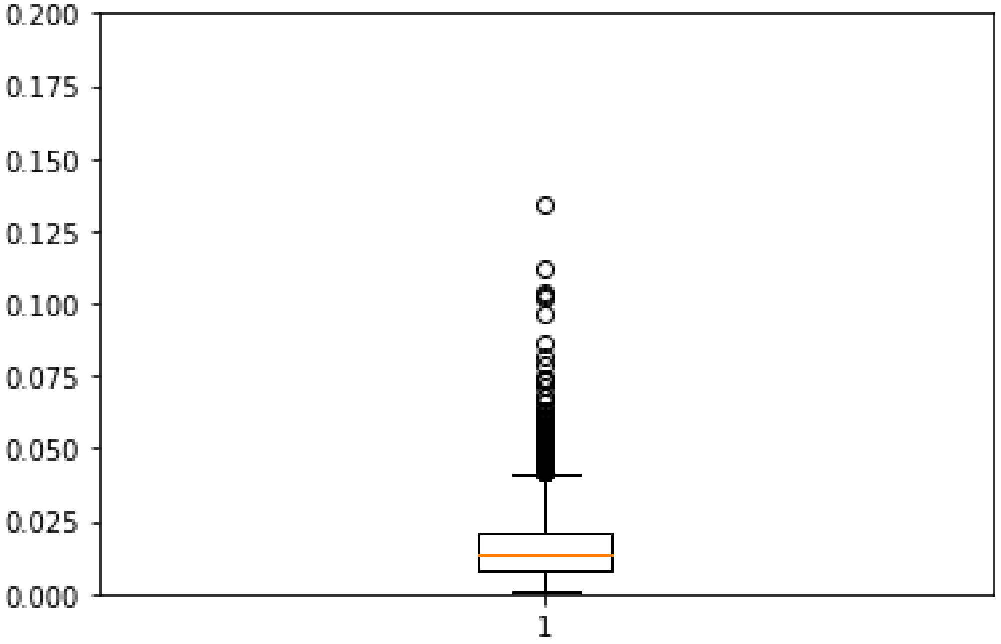


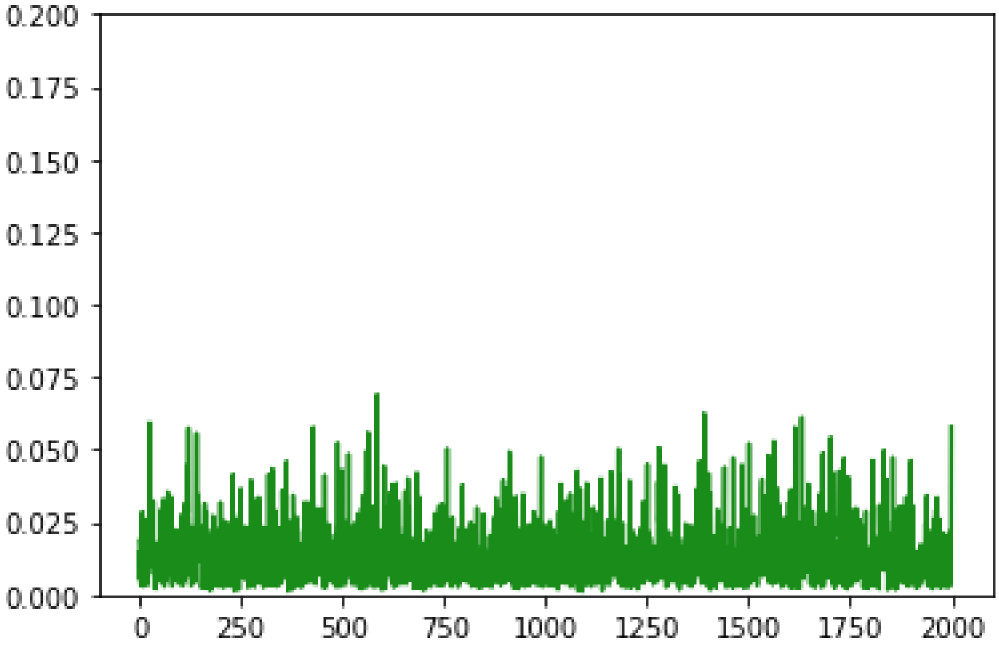
```
0.0120463408006
```

```
In [49]: test(model, test_X[0])
         test(model, test_X[2])
```

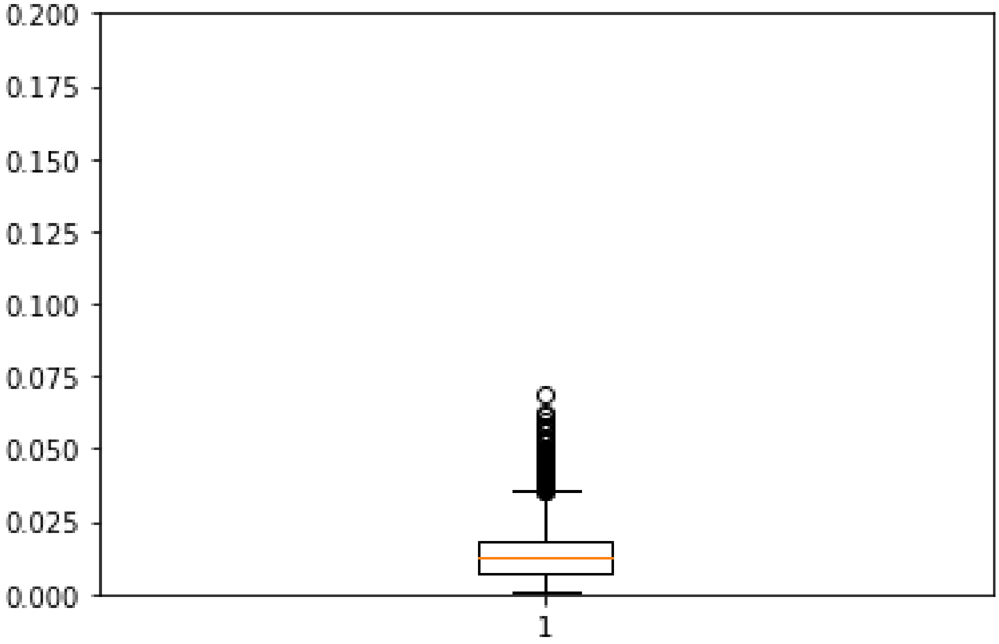


```
0.0166102026323
```





```
0.0142984822601
```



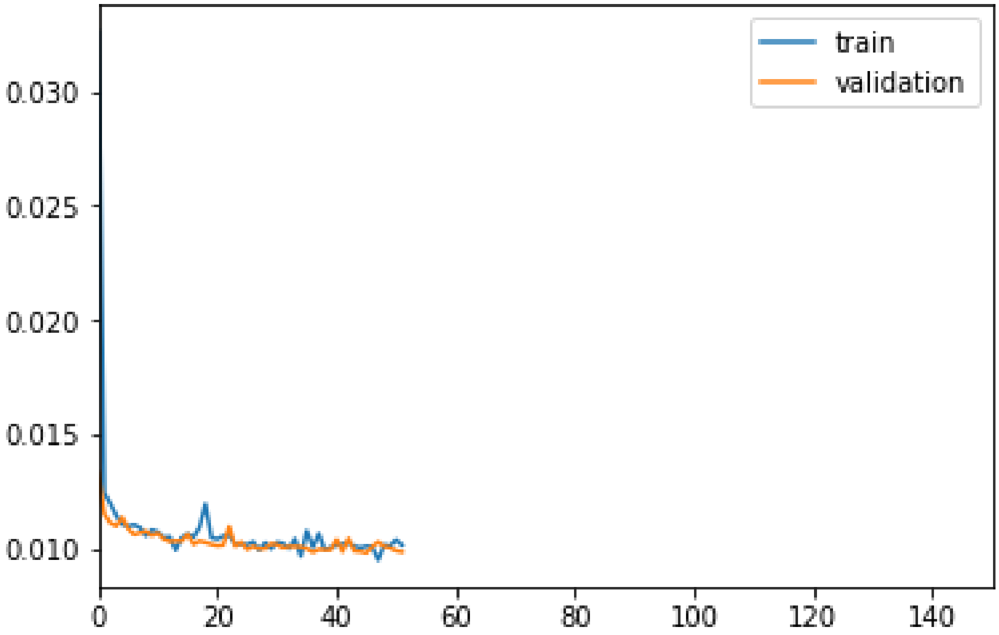
20 steps

```
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)
```

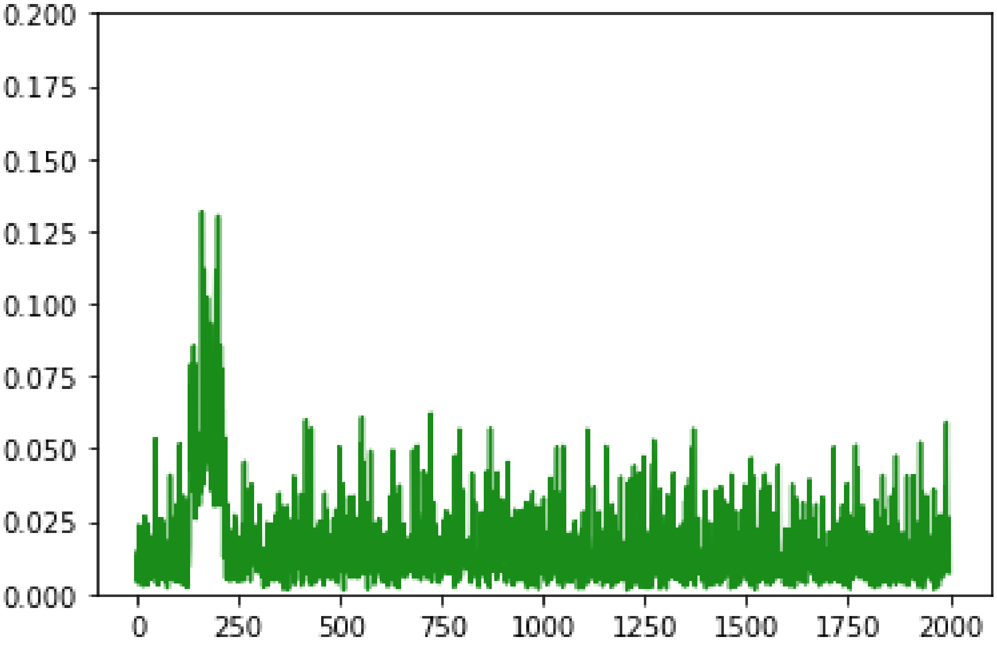
```
In [52]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [53]: train(model, tgen, vgen)
```

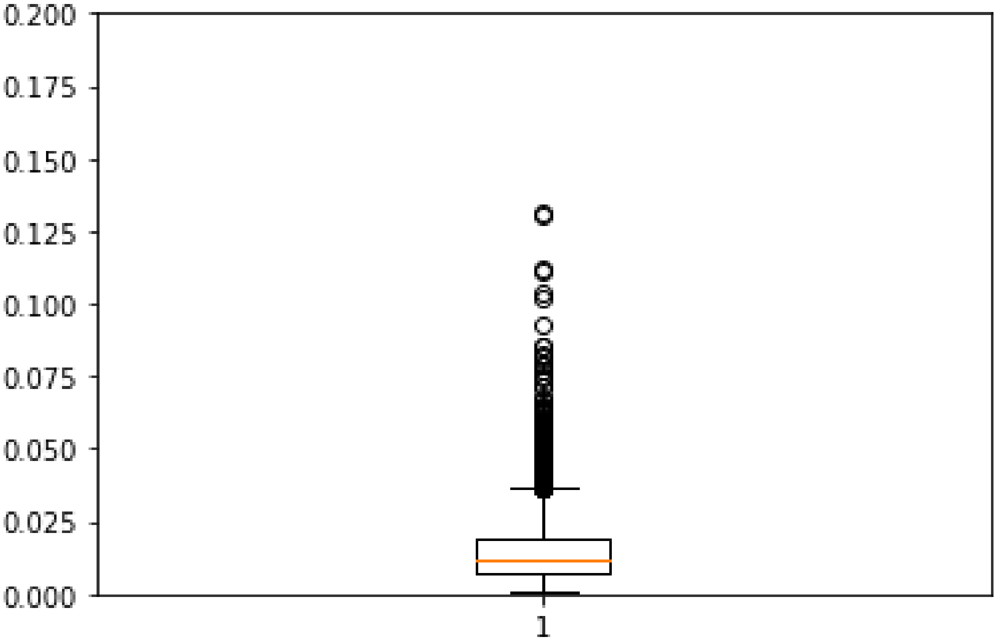


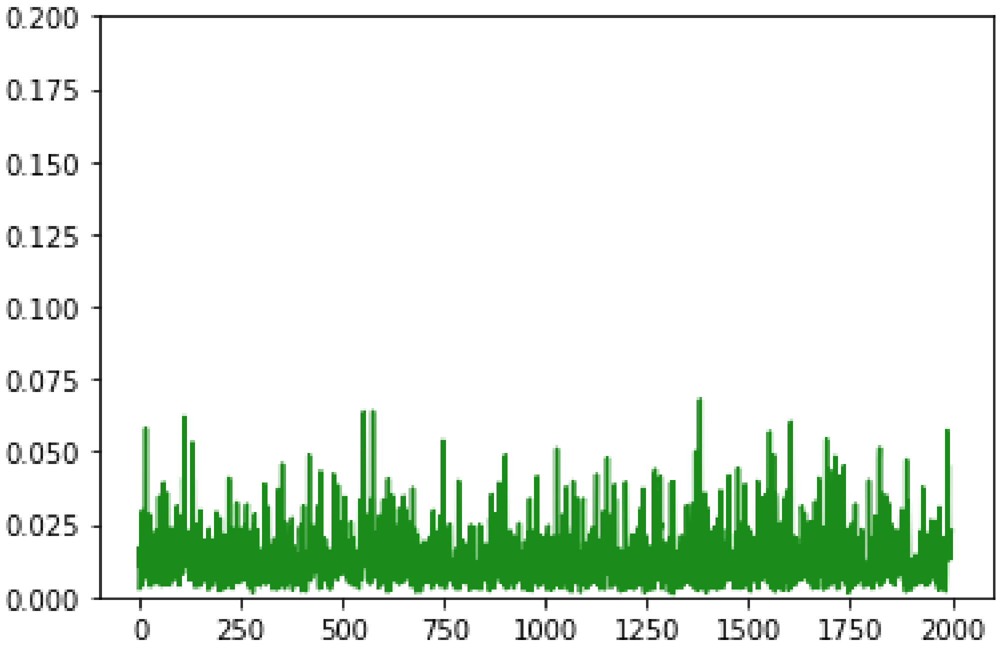
```
0.0101781996673
```

```
In [54]: test(model, test_X[0])
         test(model, test_X[2])
```

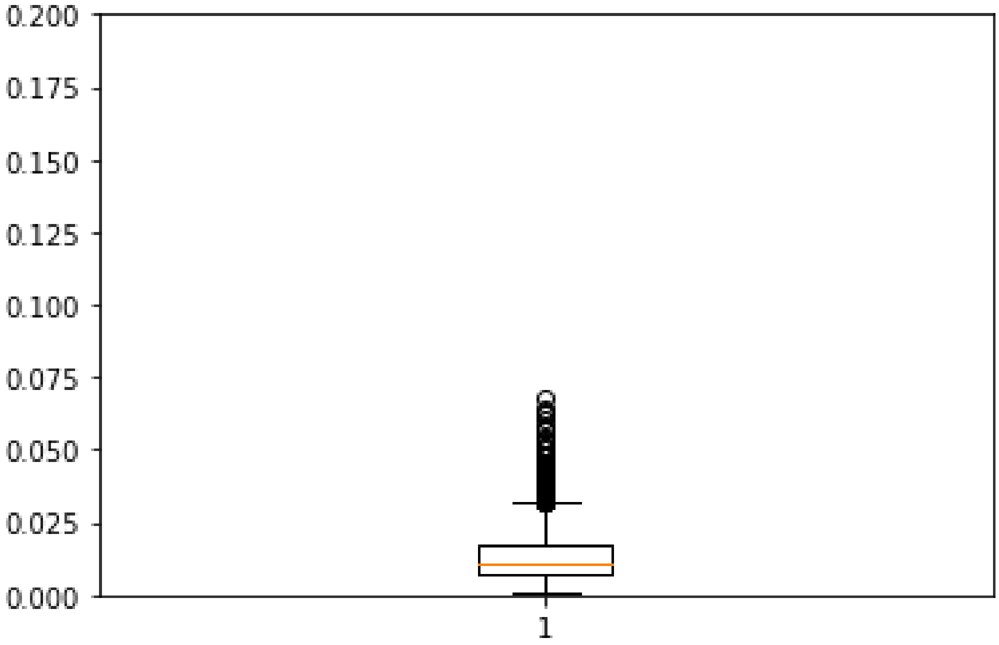


```
0.0158770344712
```





```
0.0131648851873
```



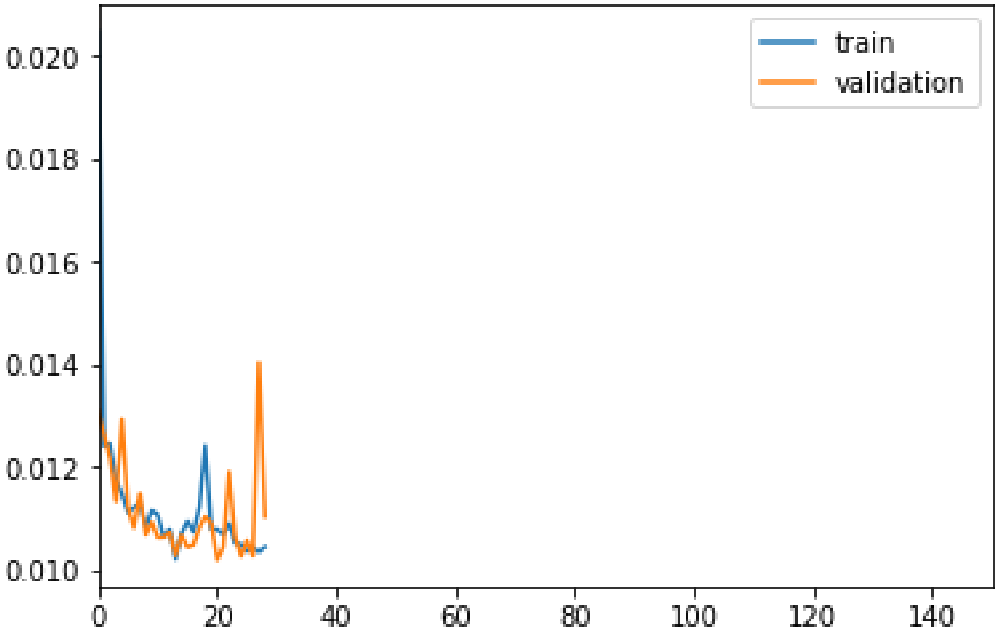
50 steps

```
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)
```

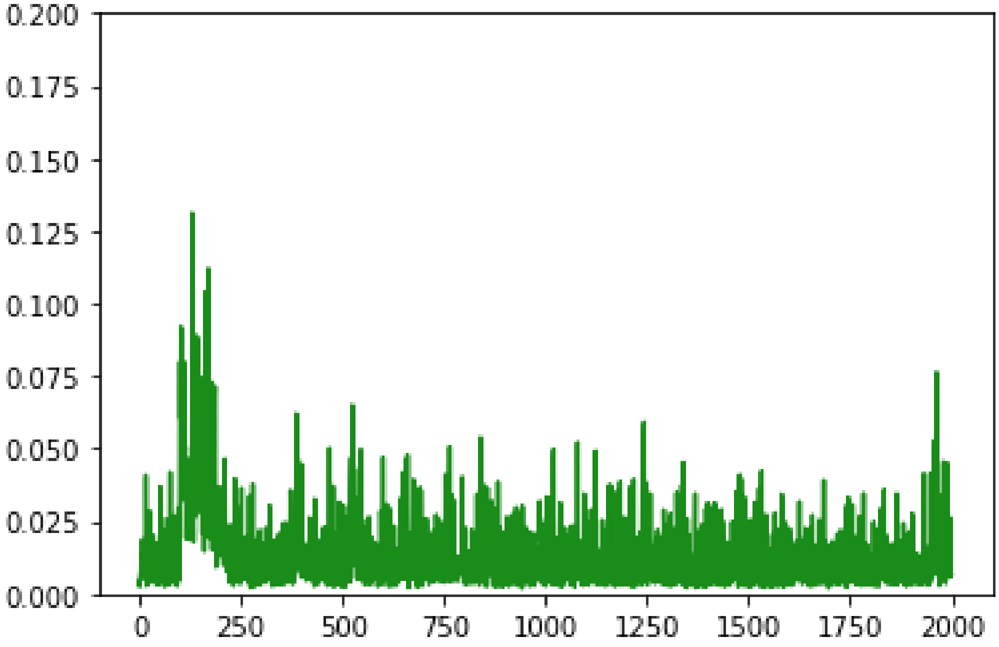
```
In [57]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [58]: train(model, tgen, vgen)
```

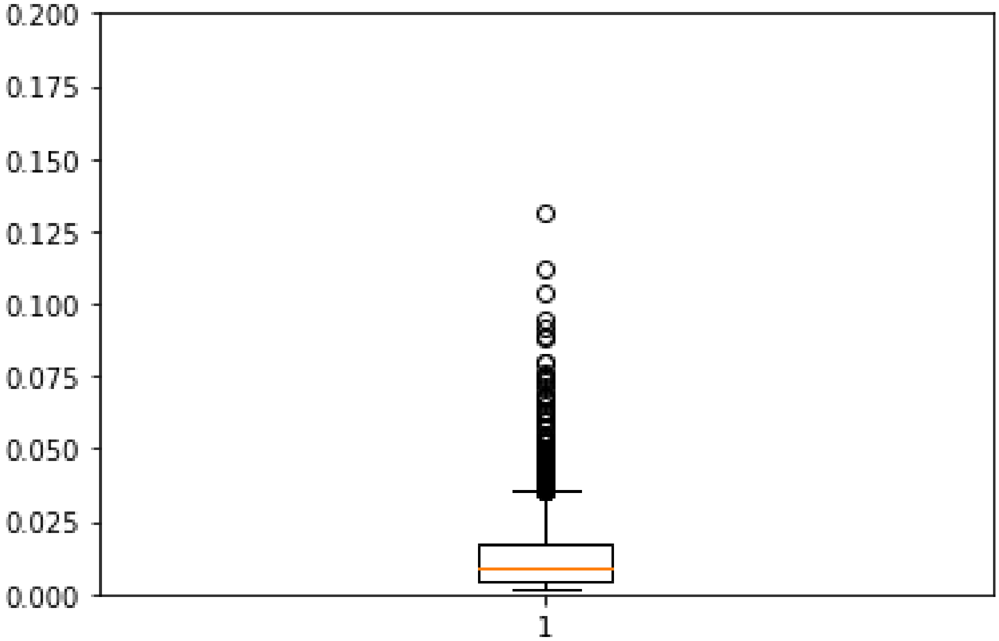


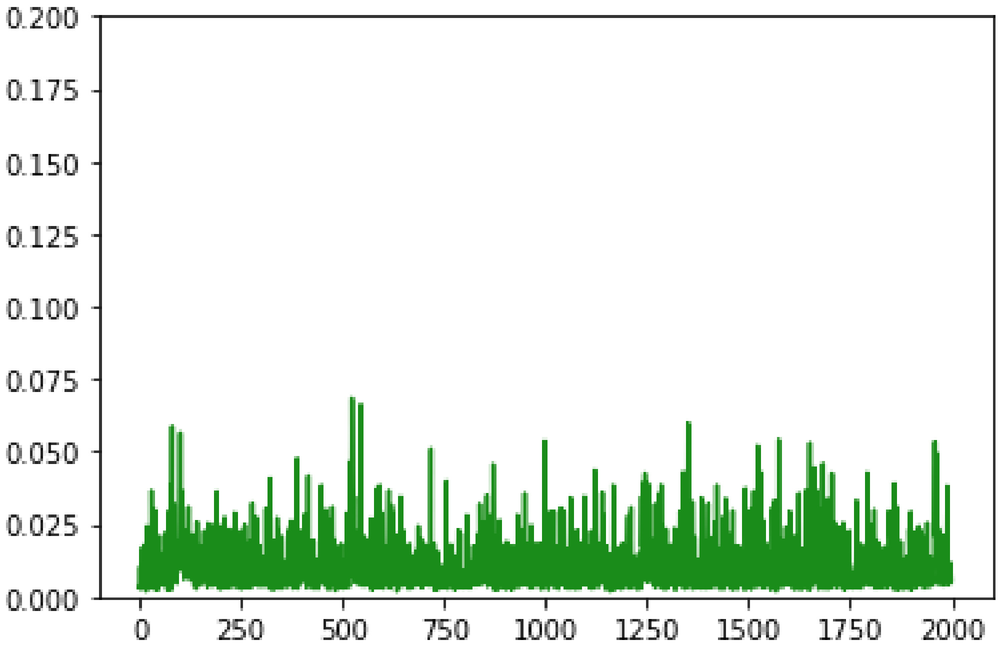
```
0.0104470289015
```

```
In [59]: test(model, test_X[0])
         test(model, test_X[2])
```

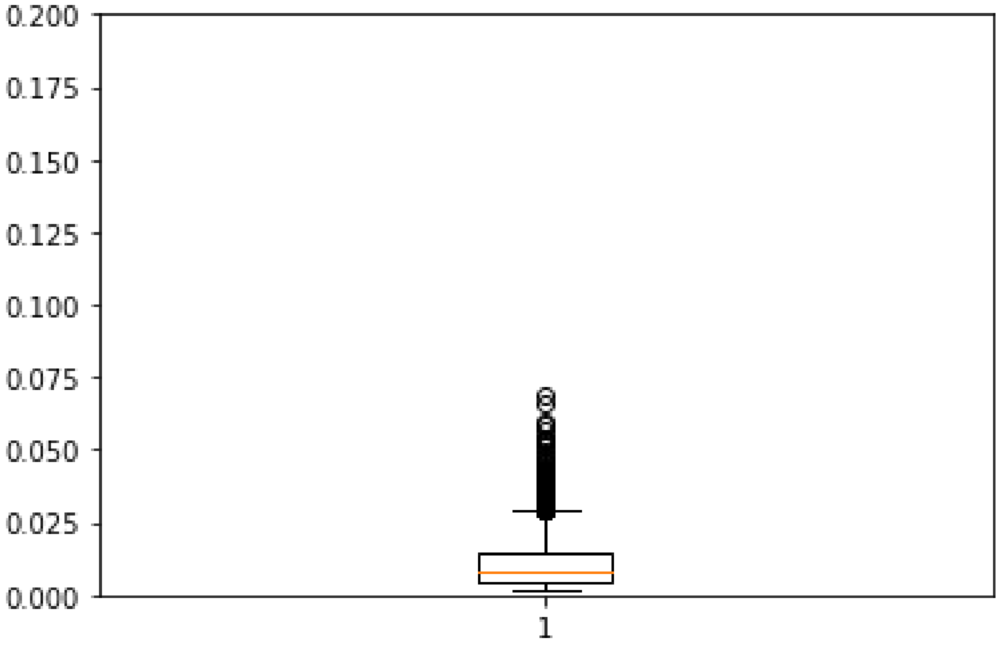


```
0.0131156058655
```





```
0.0107227368134
```



2.1.2 NN with 1 hidden layer

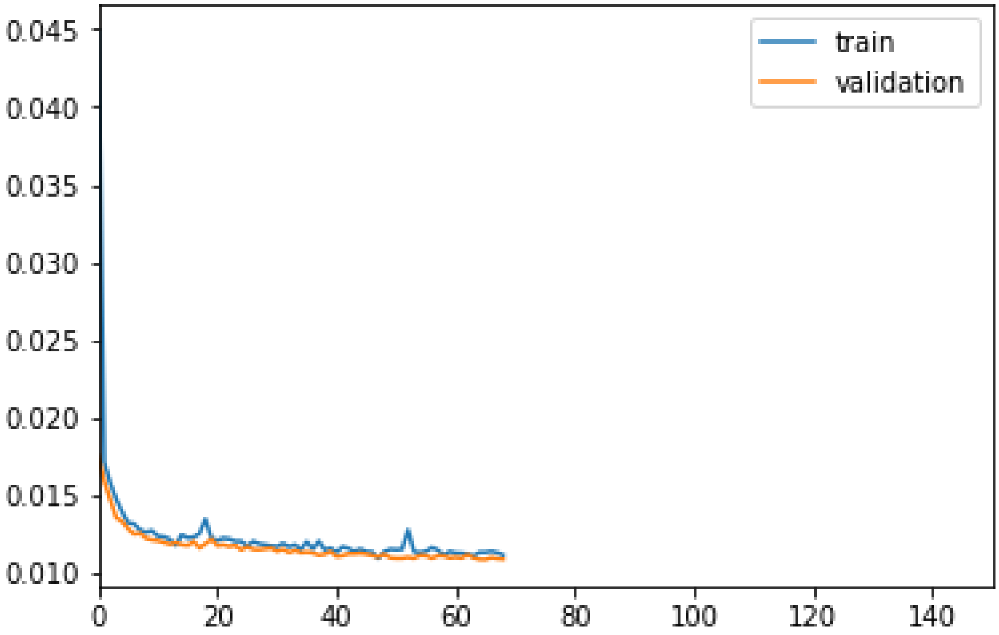
2 steps

```
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)
```

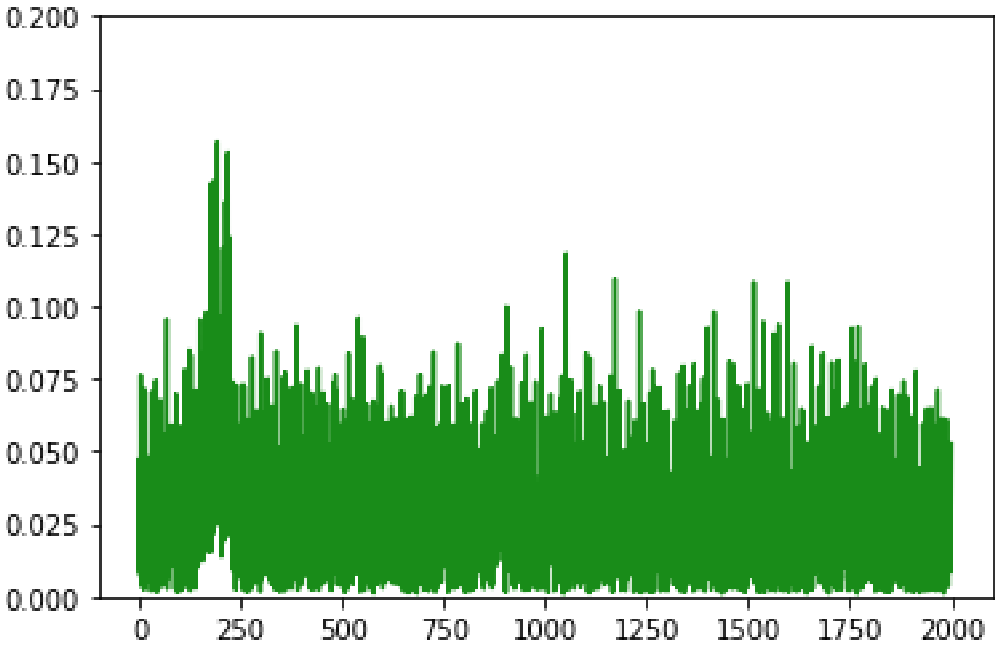
```
In [62]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [63]: train(model, tgen, vgen)
```

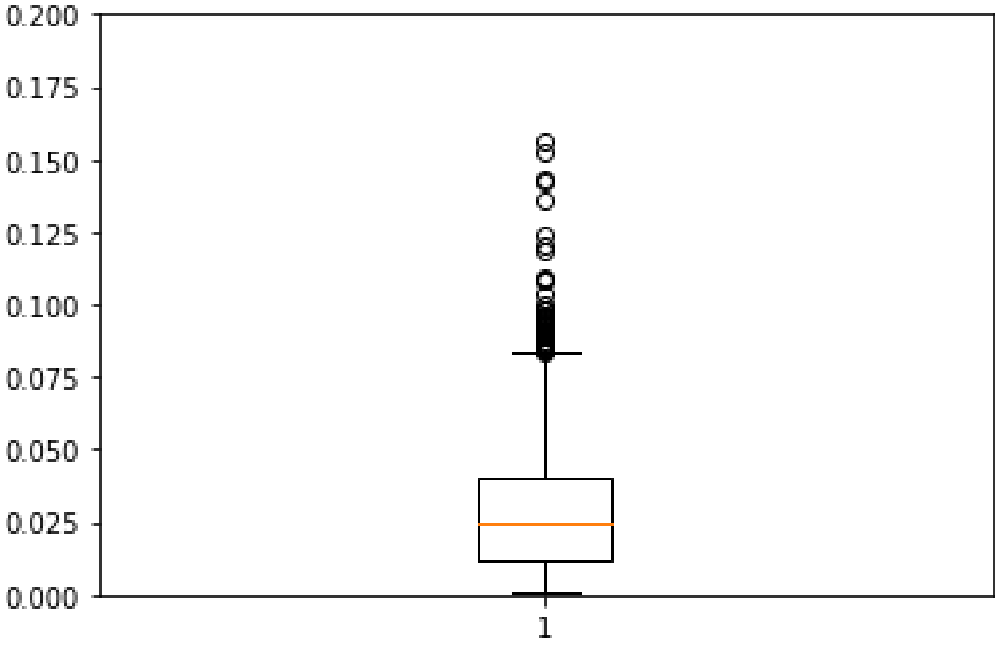


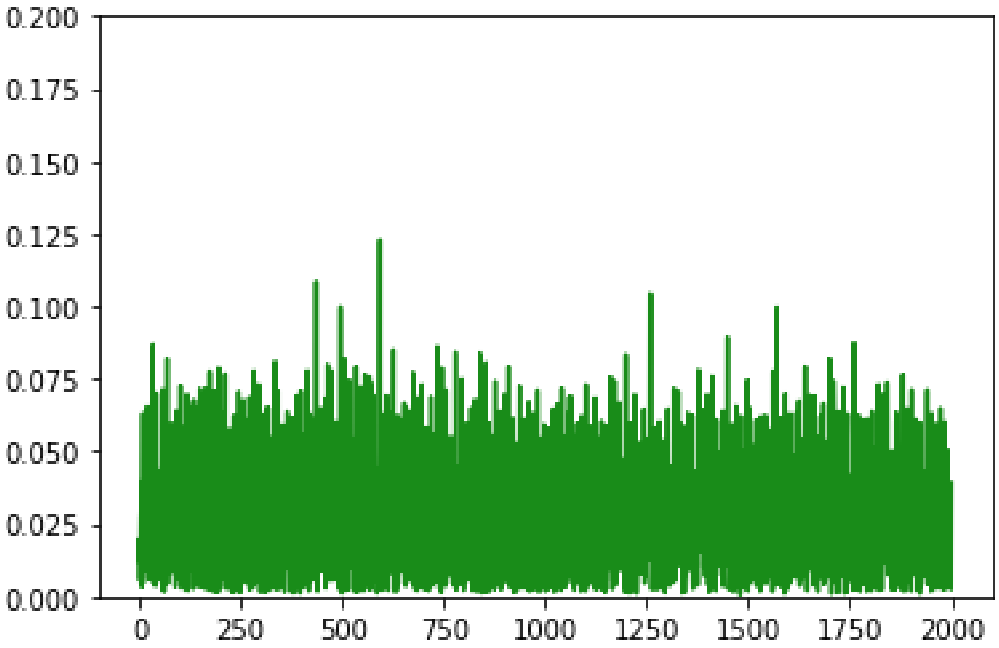
```
0.0111121227426
```

```
In [64]: test(model, test_X[0])
         test(model, test_X[2])
```

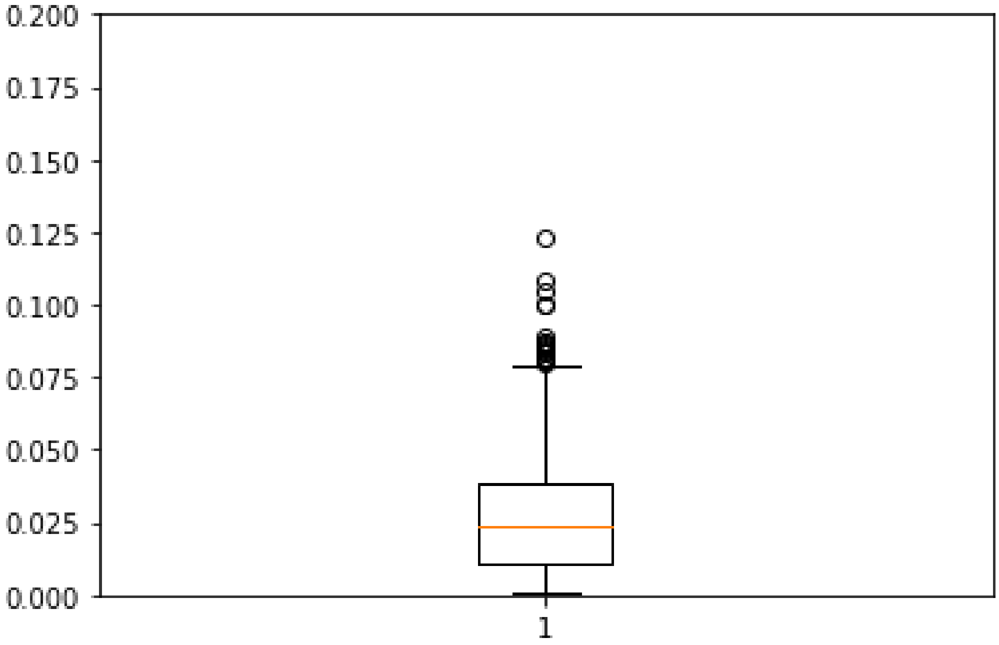


```
0.02924723288
```





```
0.0270351578908
```



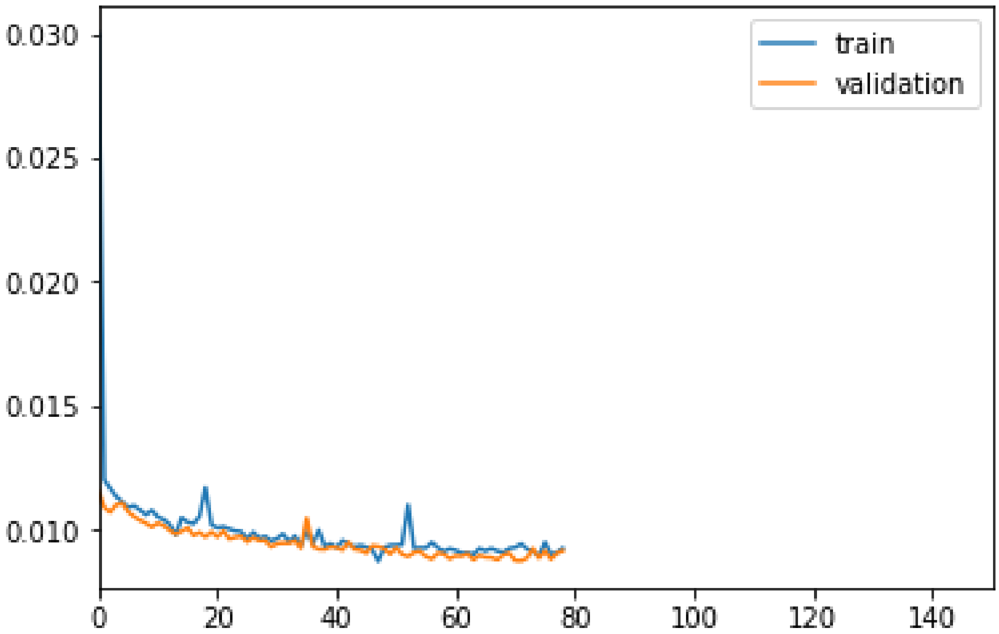
5 steps

```
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)
```

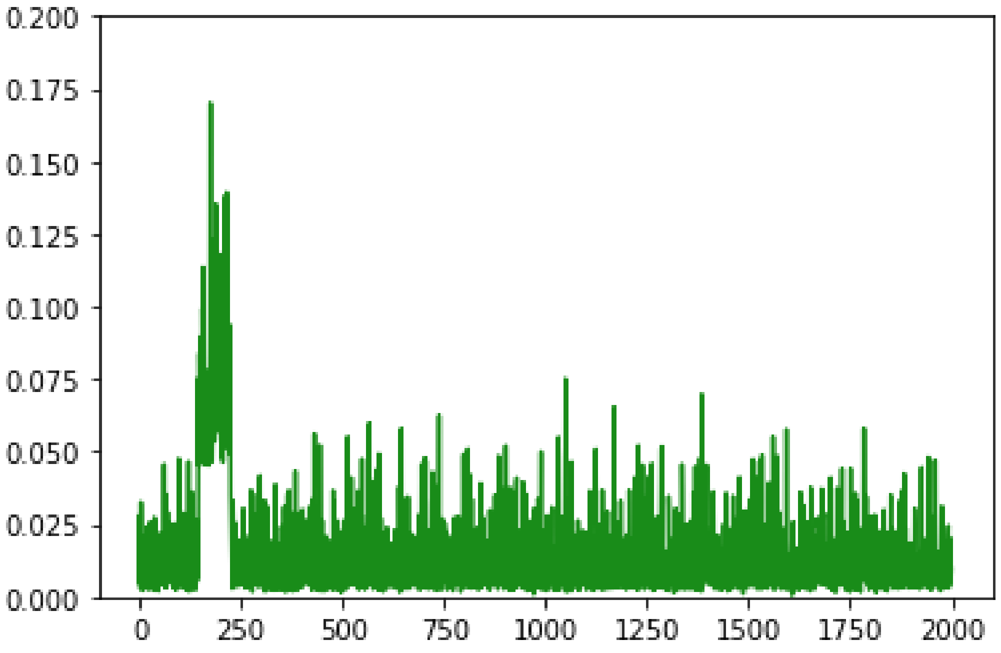
```
In [67]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [68]: train(model, tgen, vgen)
```

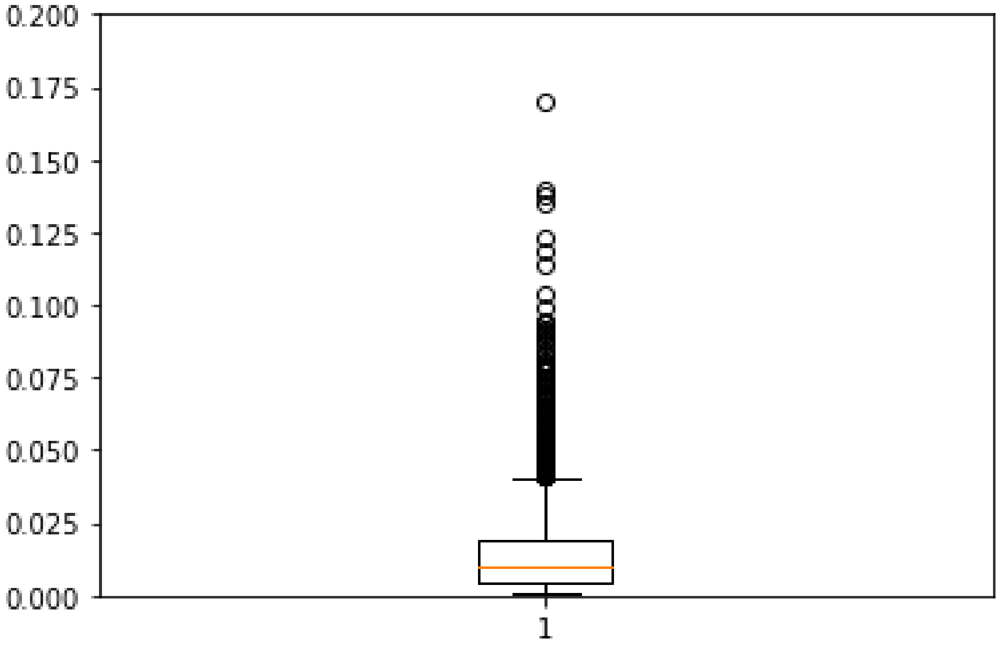


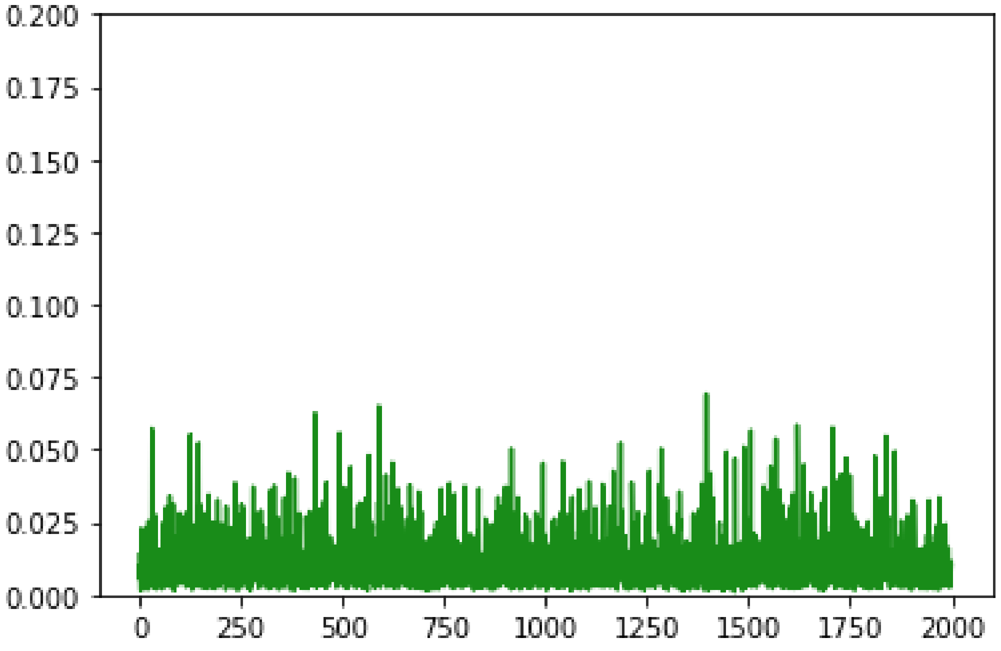
```
0.00928021989902
```

```
In [69]: test(model, test_X[0])
         test(model, test_X[2])
```

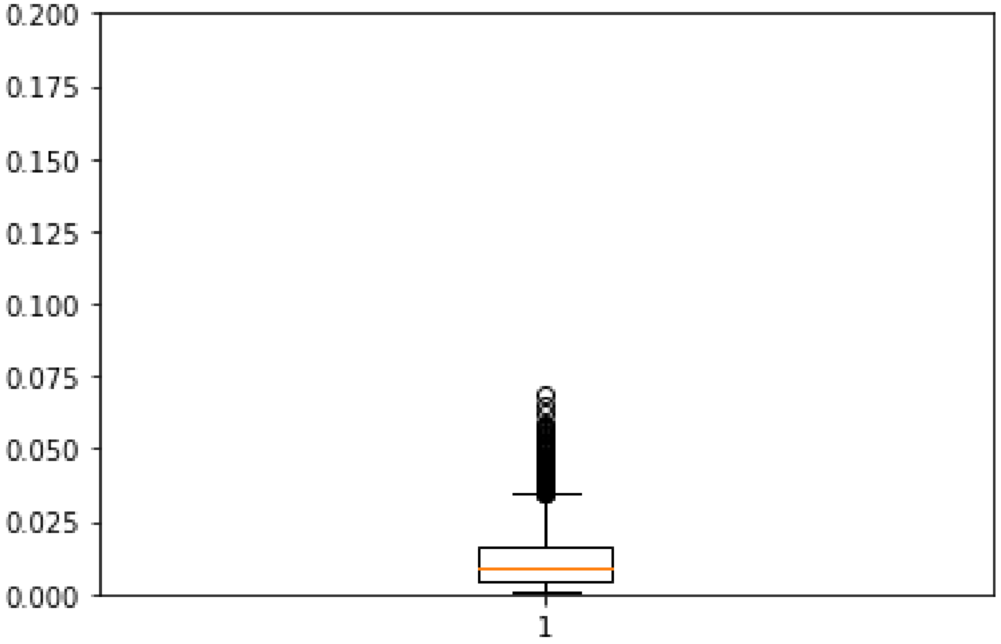


```
0.0151539670686
```





```
0.0117491705594
```



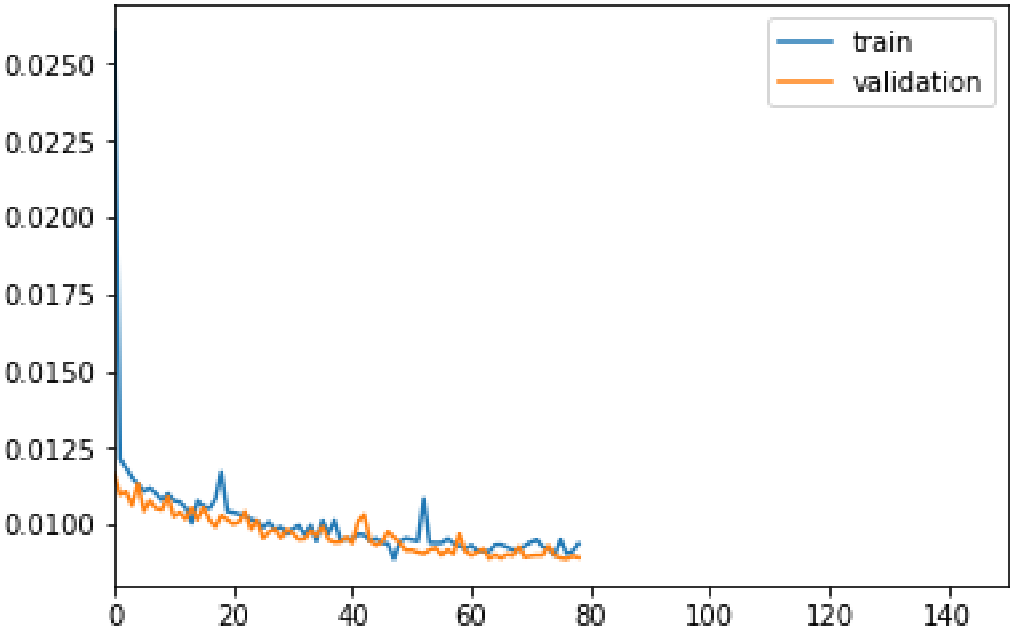
10 steps

```
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)
```

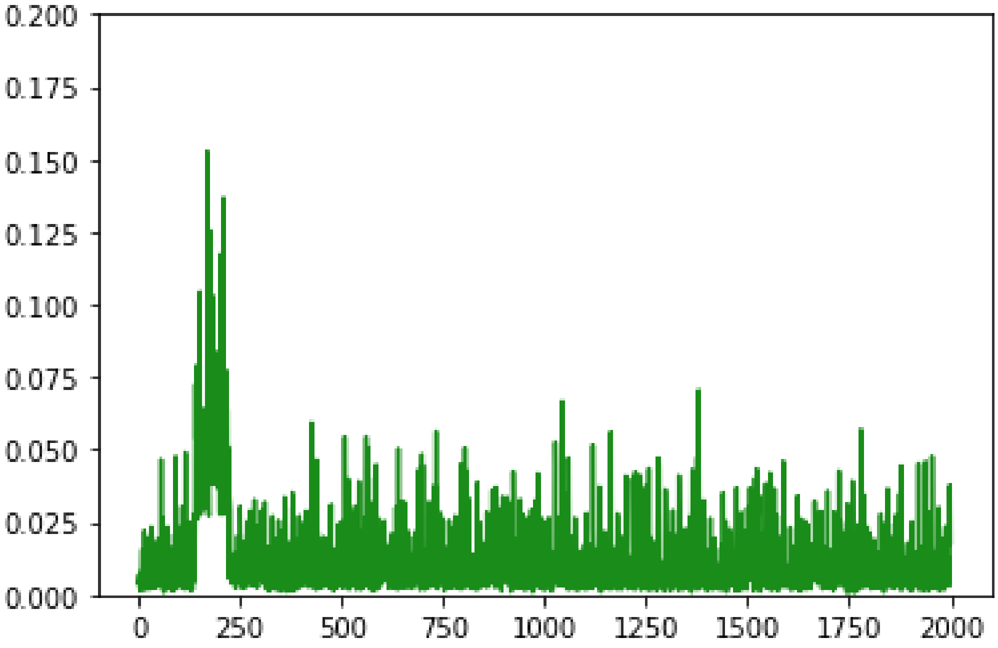
```
In [72]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [73]: train(model, tgen, vgen)
```

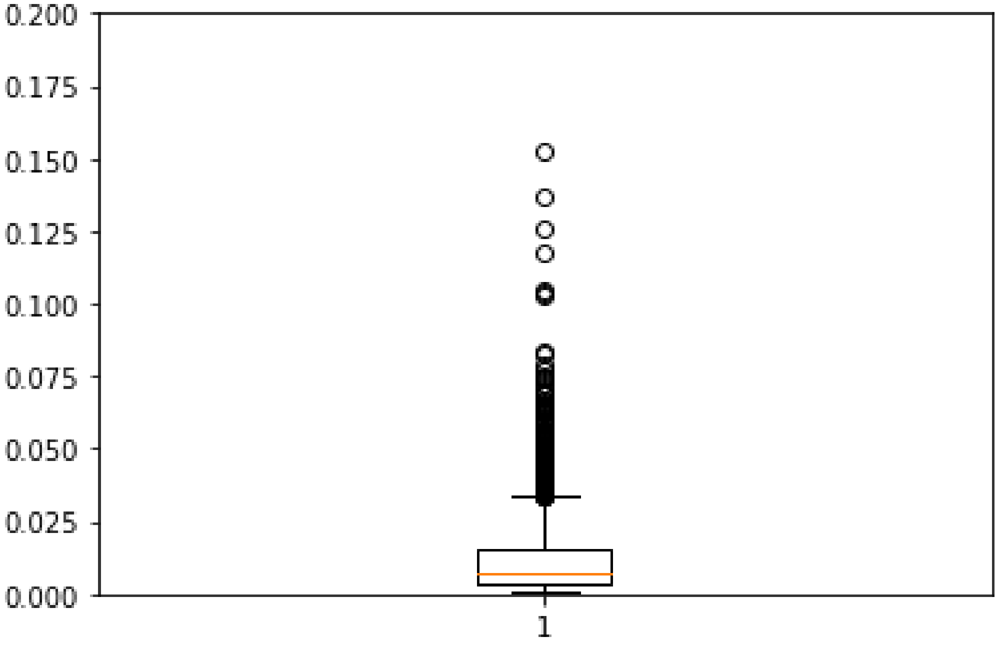


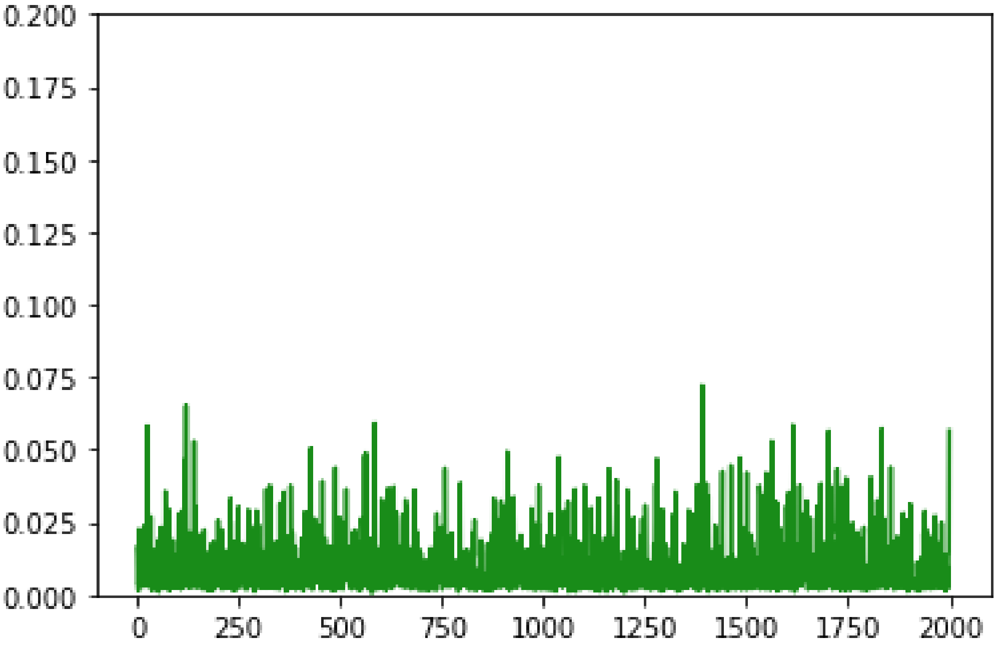
```
0.00937749258394
```

```
In [74]: test(model, test_X[0])
         test(model, test_X[2])
```

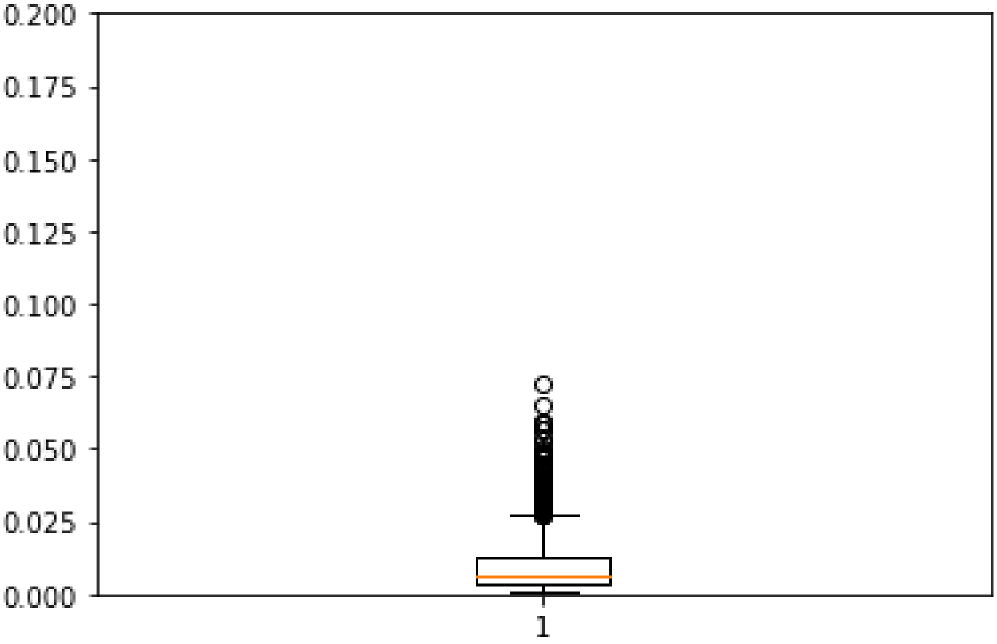


```
0.0125596583654
```





```
0.00964894549785
```



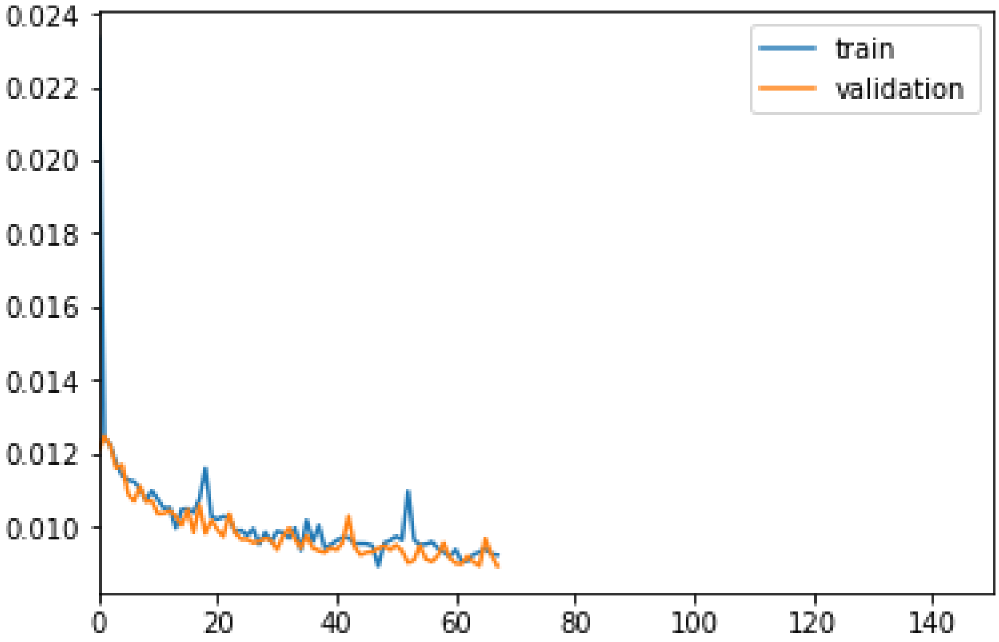
20 steps

```
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)
```

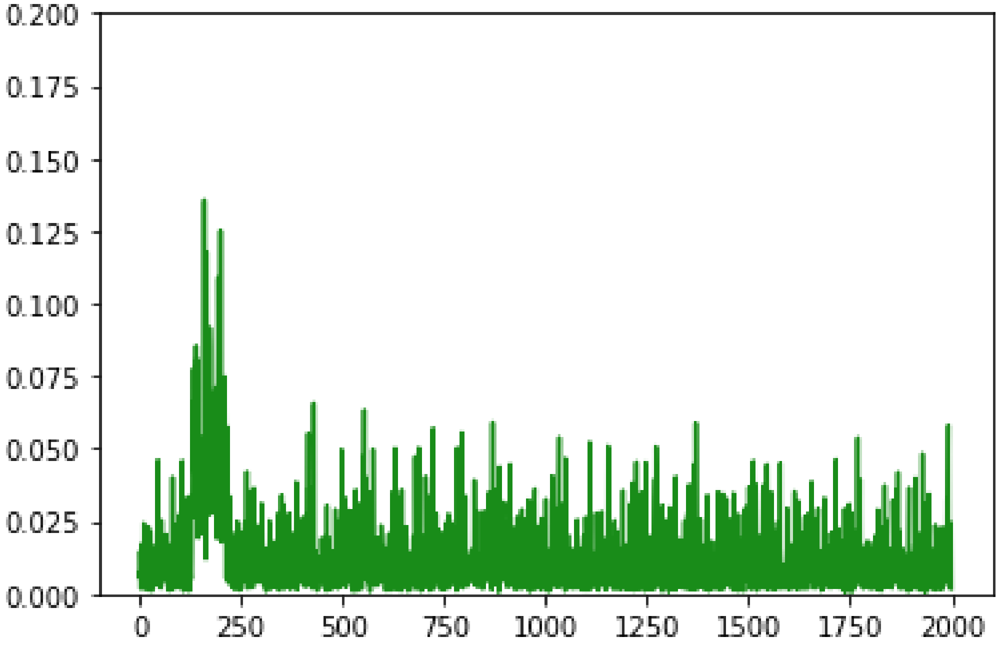
```
In [77]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [78]: train(model, tgen, vgen)
```

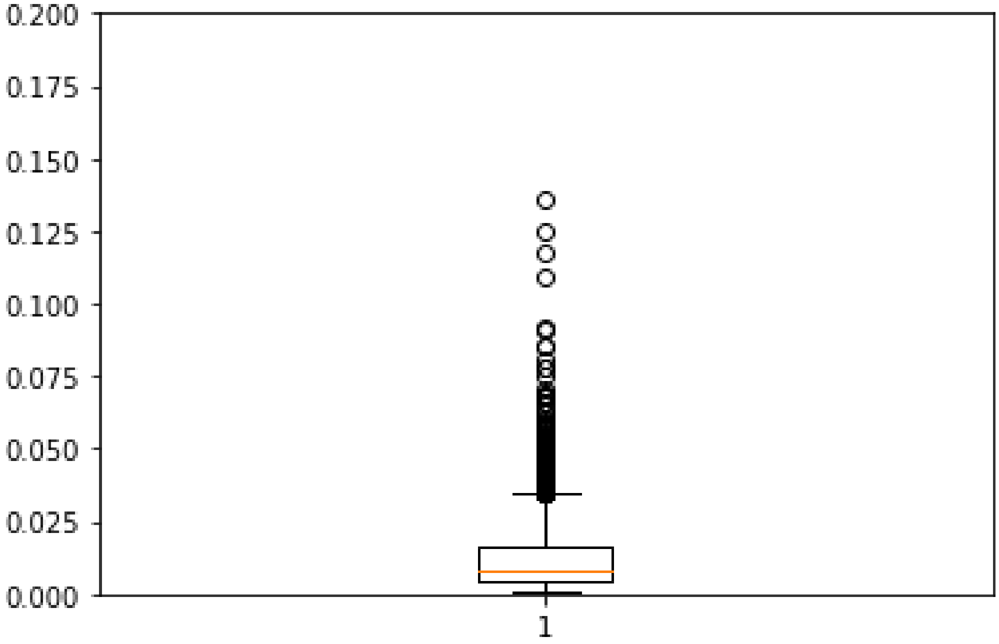


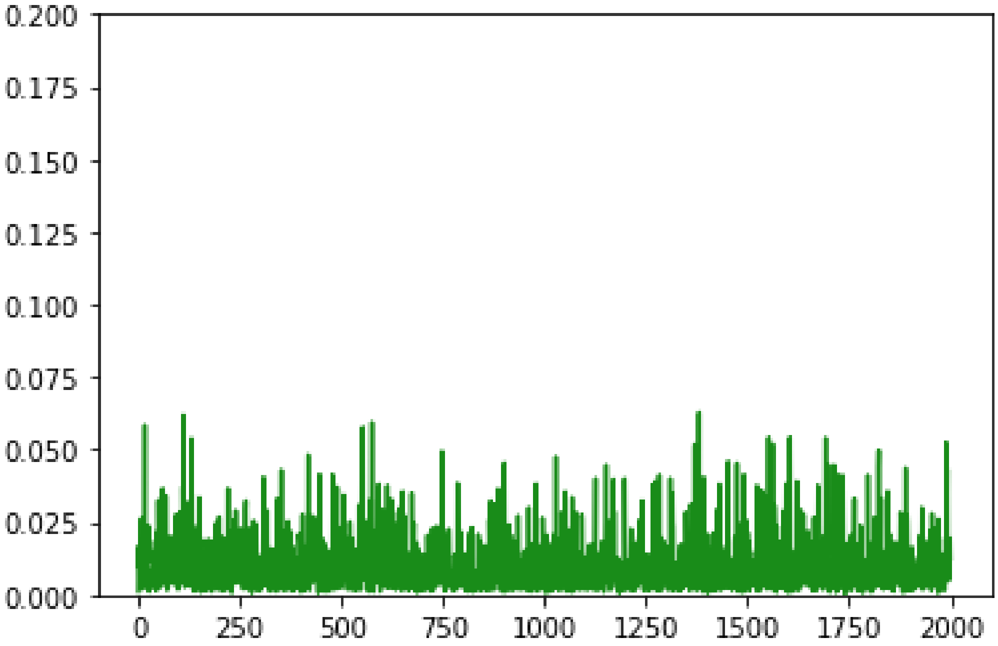
```
0.0092292446075
```

```
In [79]: test(model, test_X[0])
         test(model, test_X[2])
```

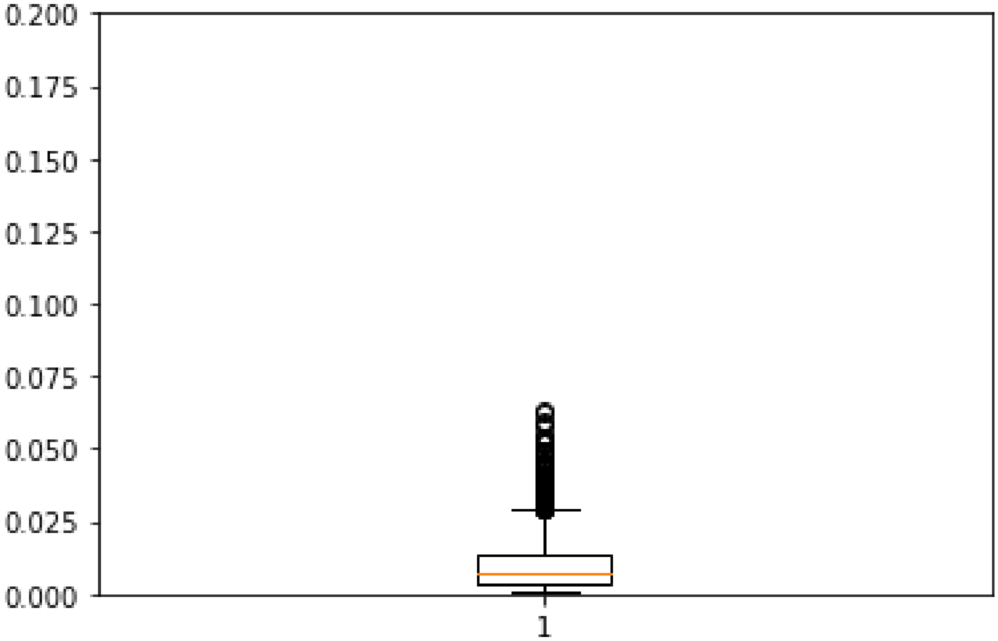


```
0.0125774245364
```





```
0.0102358840262
```



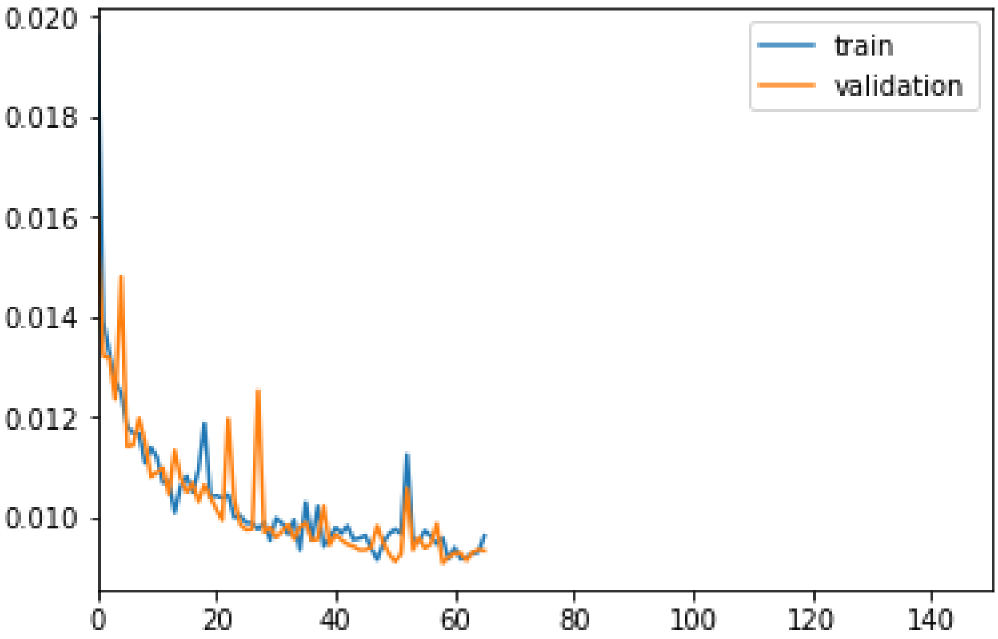
50 steps

```
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)
```

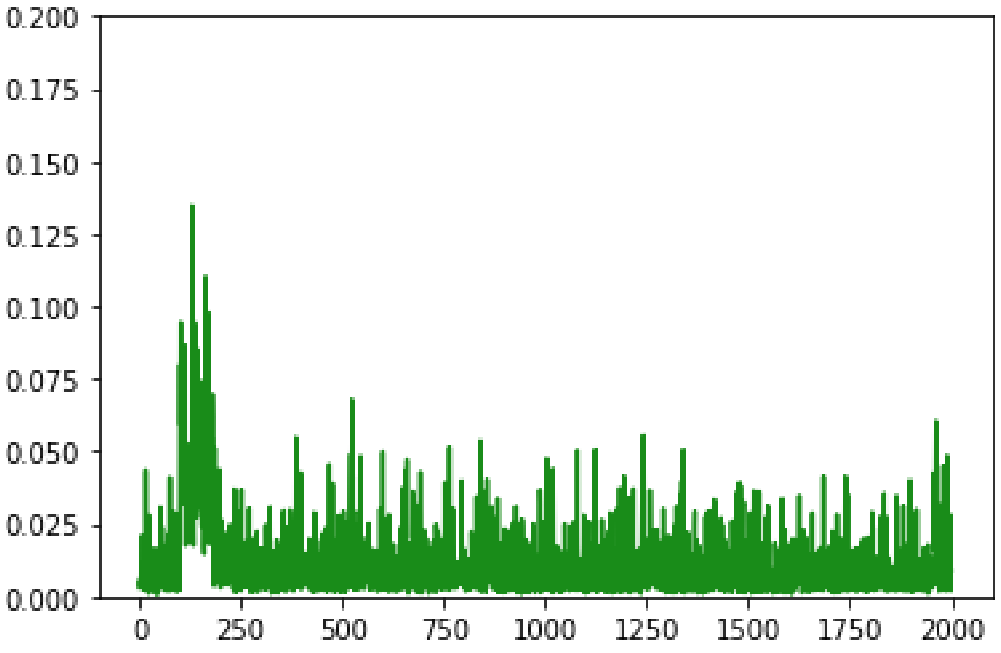
```
In [82]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [83]: train(model, tgen, vgen)
```

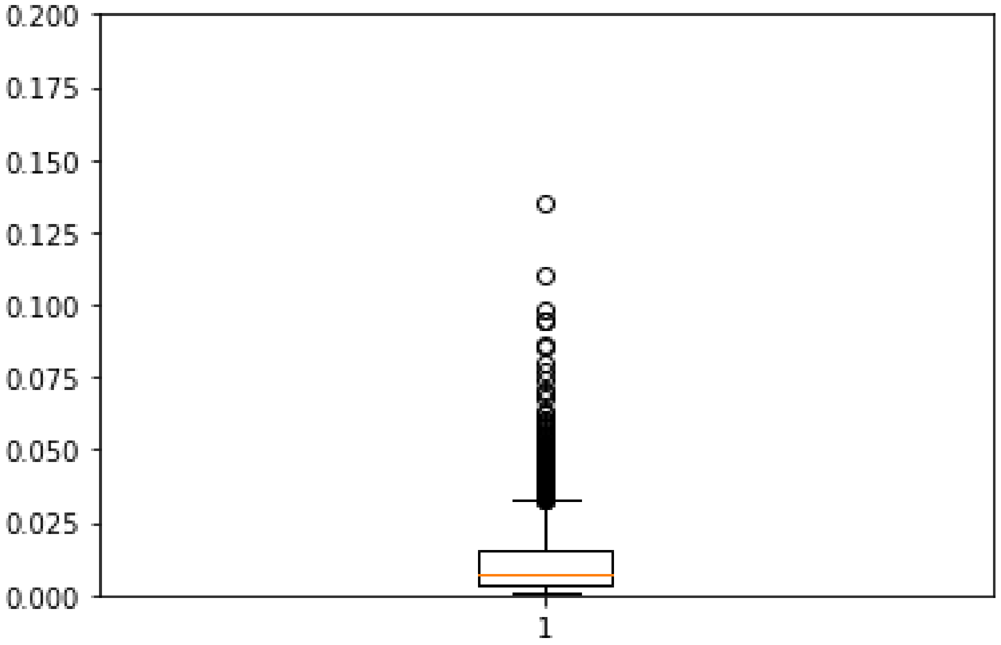


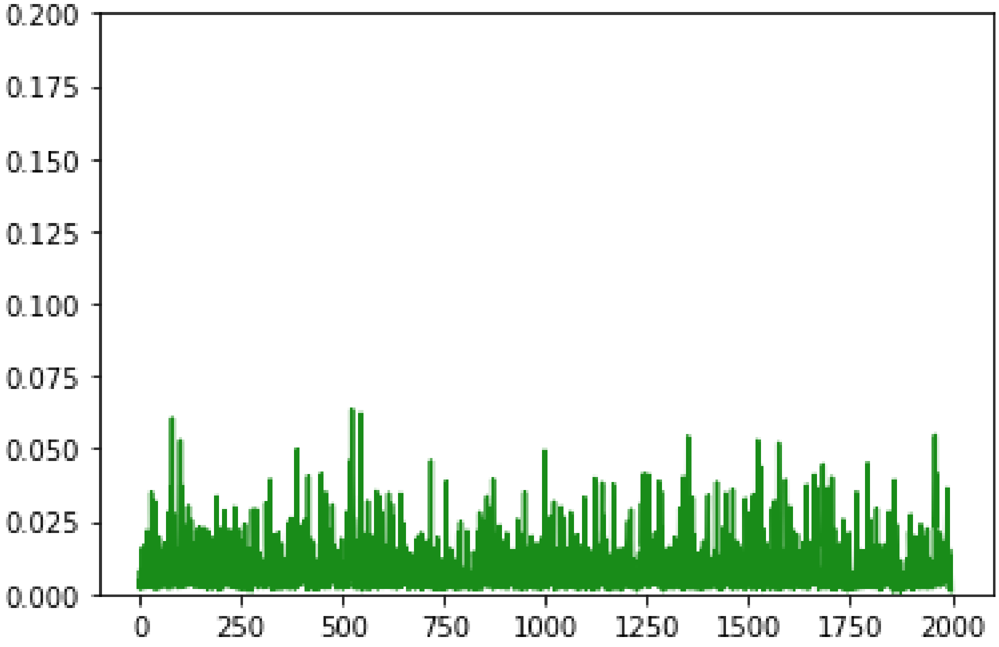
```
0.00962887890008
```

```
In [84]: test(model, test_X[0])
         test(model, test_X[2])
```

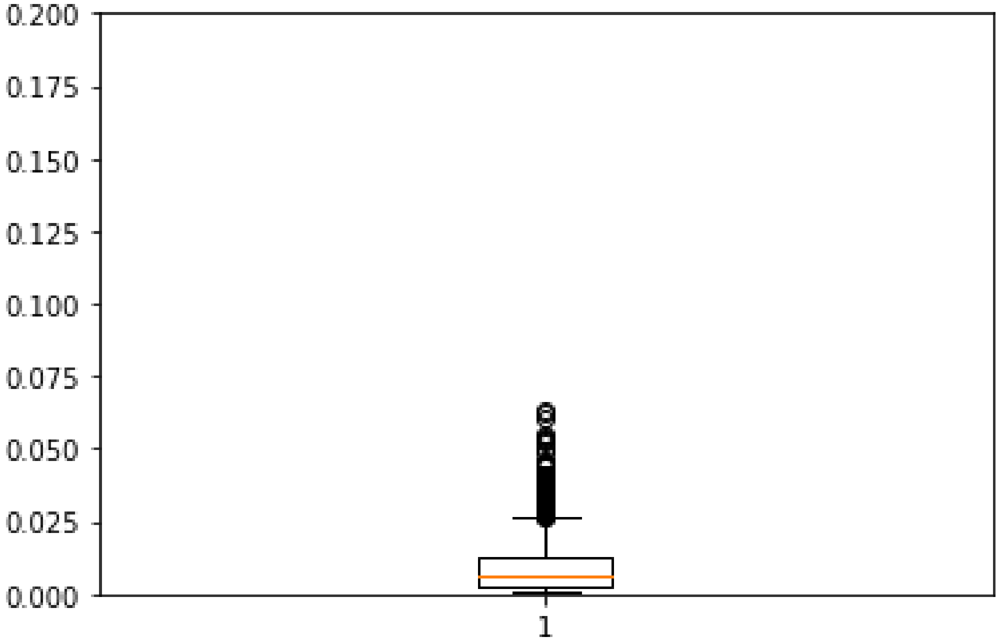


```
0.0113539913944
```





```
0.00905628428645
```



2.1.3 NN with 2 hidden layers

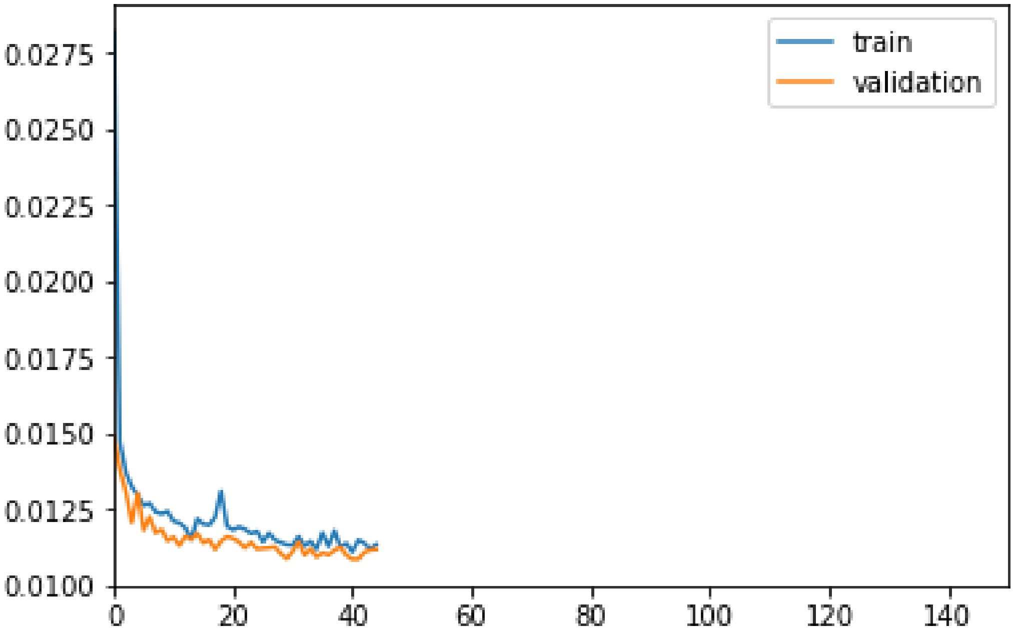
2 steps

```
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)
```

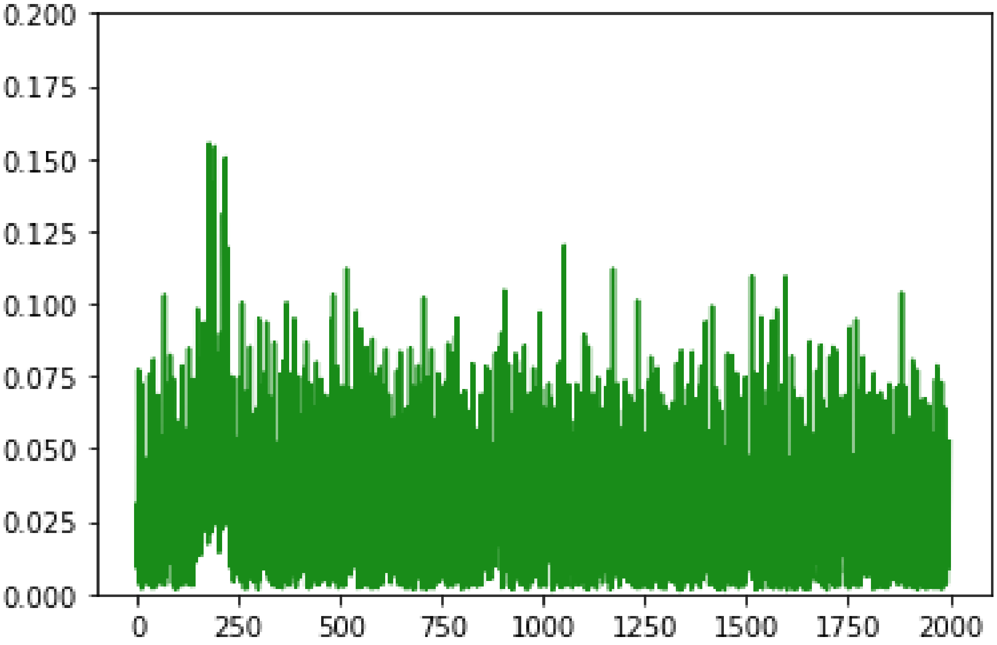
```
In [87]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [88]: train(model, tgen, vgen)
```

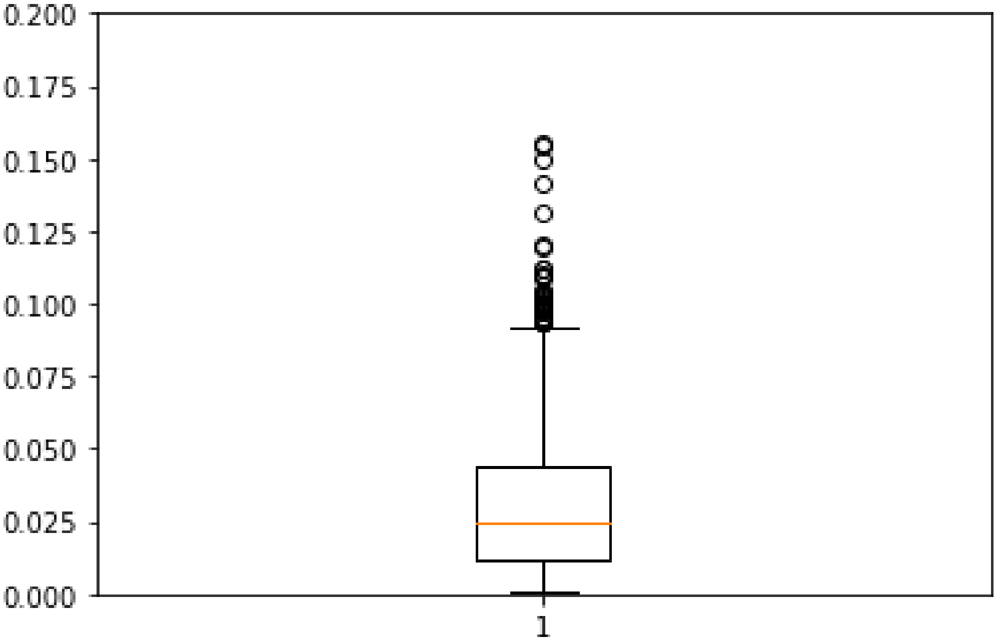


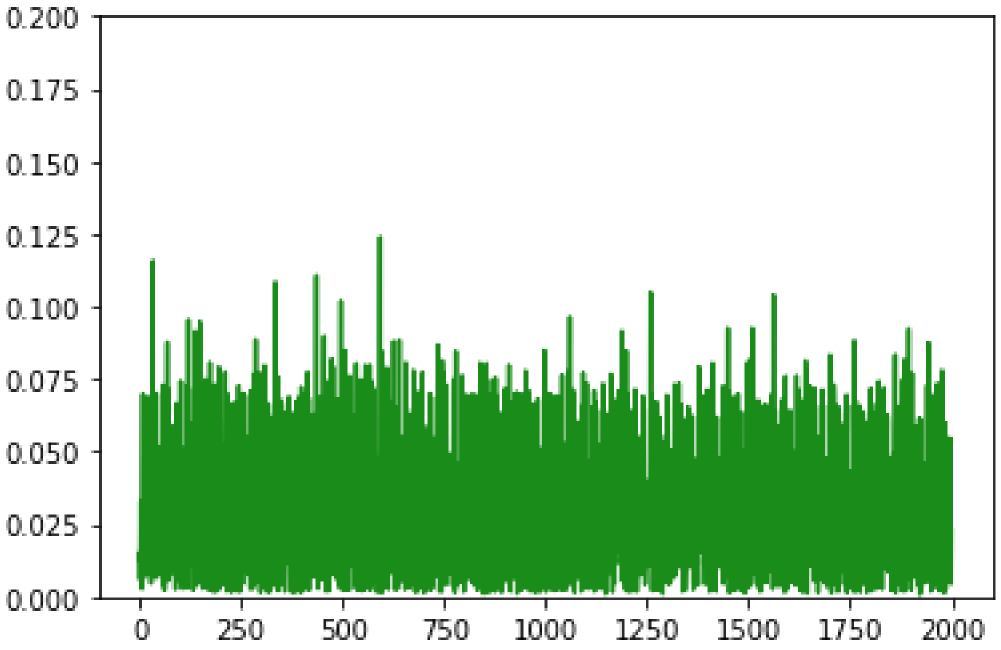
```
0.0113443566352
```

```
In [89]: test(model, test_X[0])
         test(model, test_X[2])
```

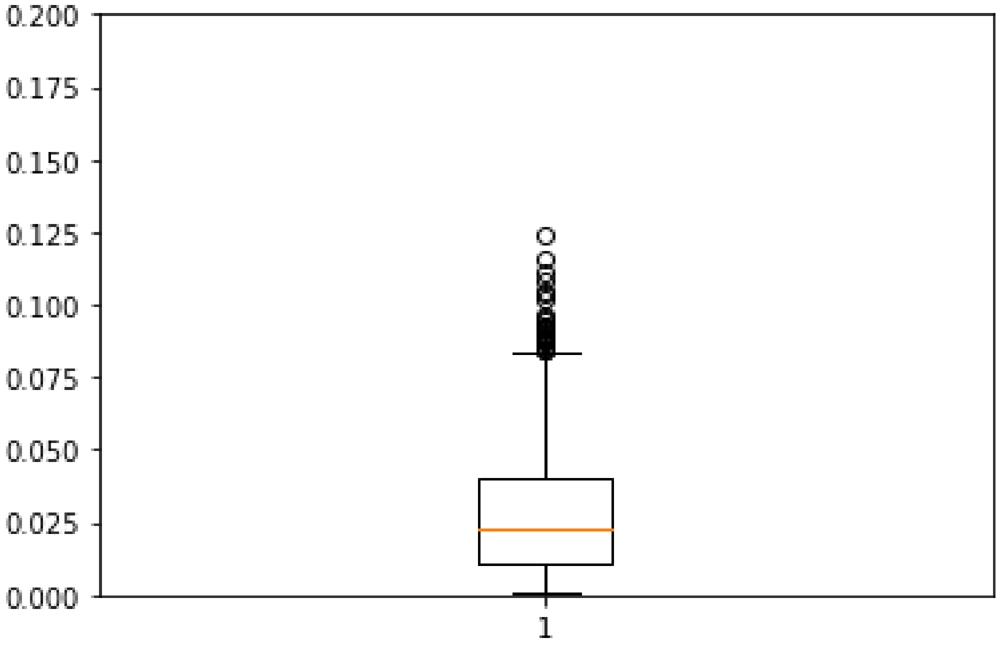


```
0.0307994102172
```





```
0.0287252774297
```



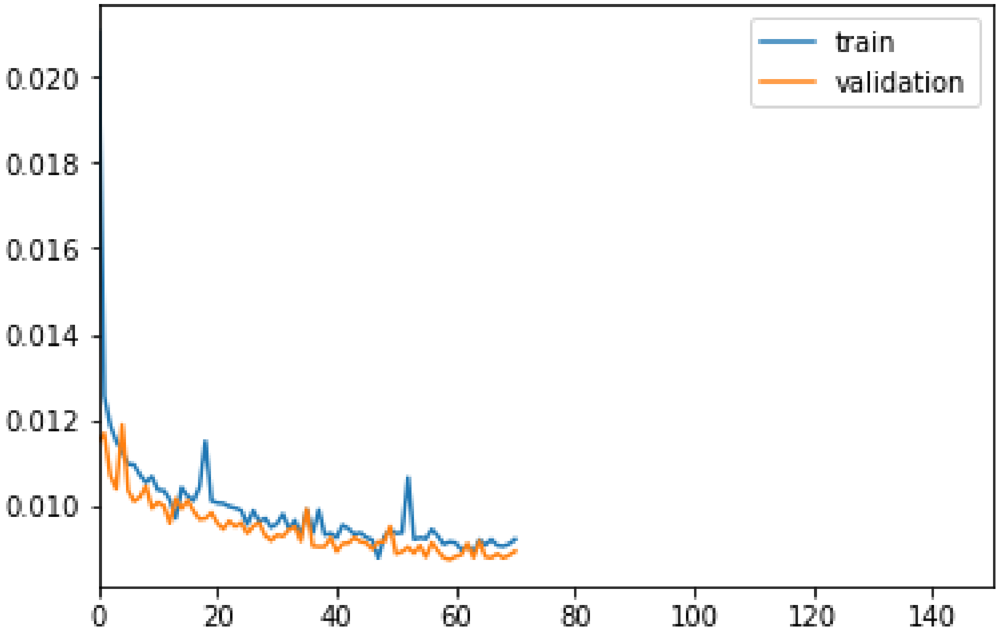
5 steps

```
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)
```

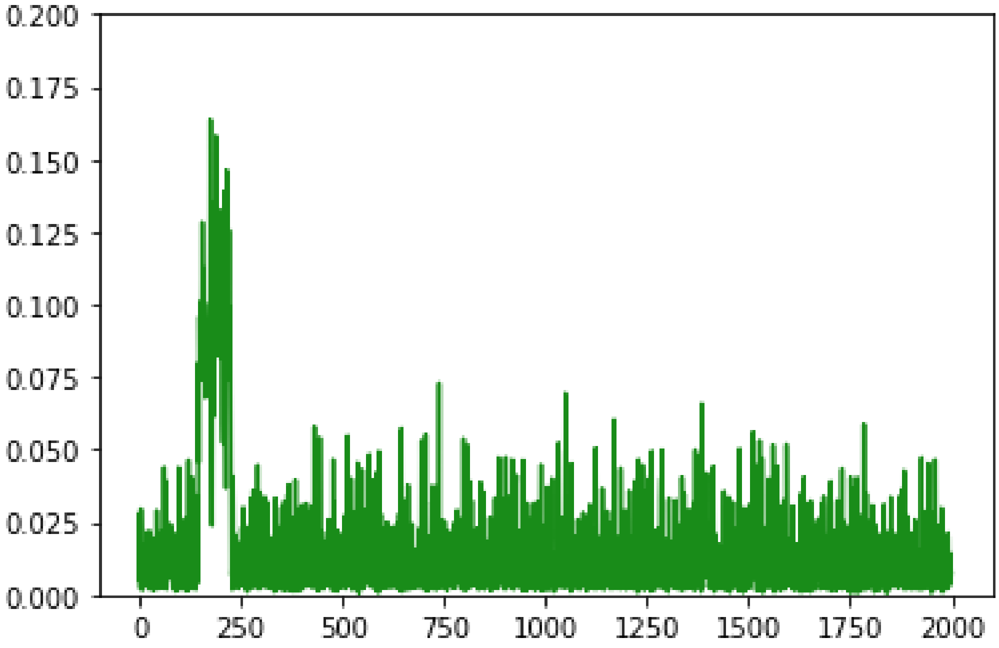
```
In [92]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [93]: train(model, tgen, vgen)
```

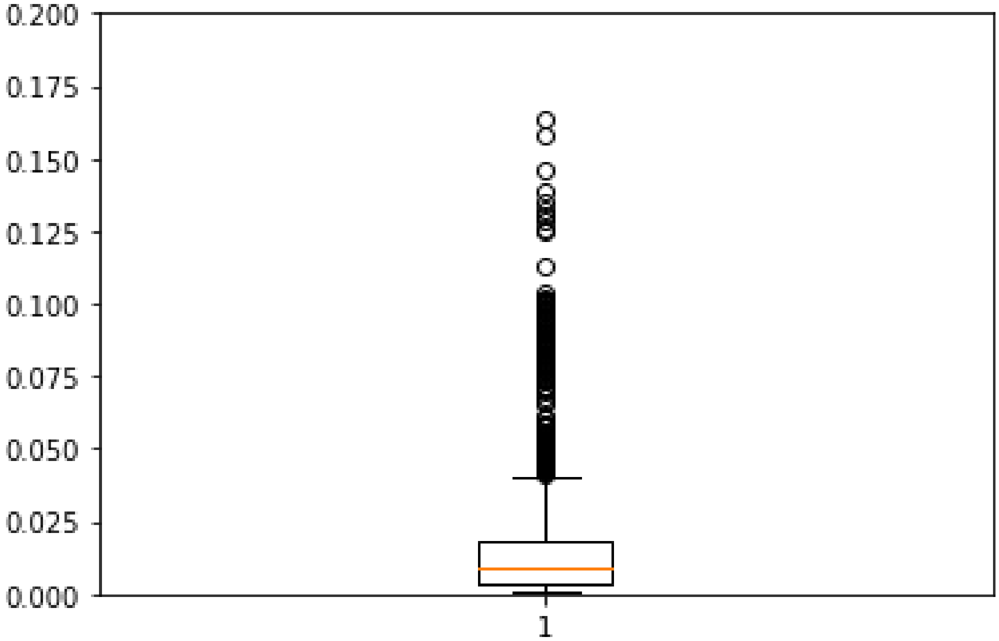


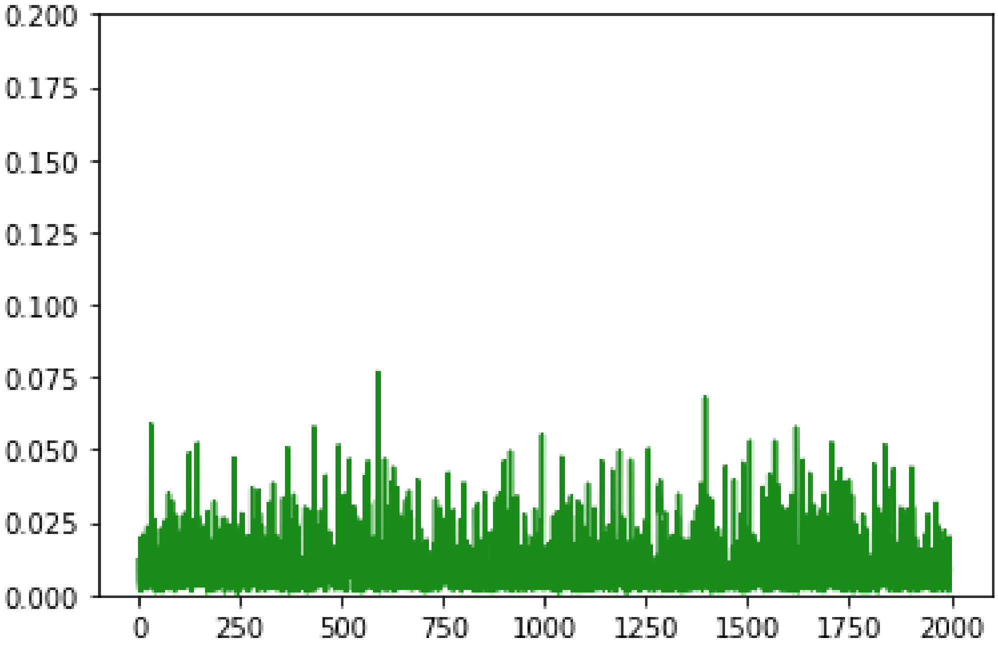
```
0.00920323406172
```

```
In [94]: test(model, test_X[0])
         test(model, test_X[2])
```

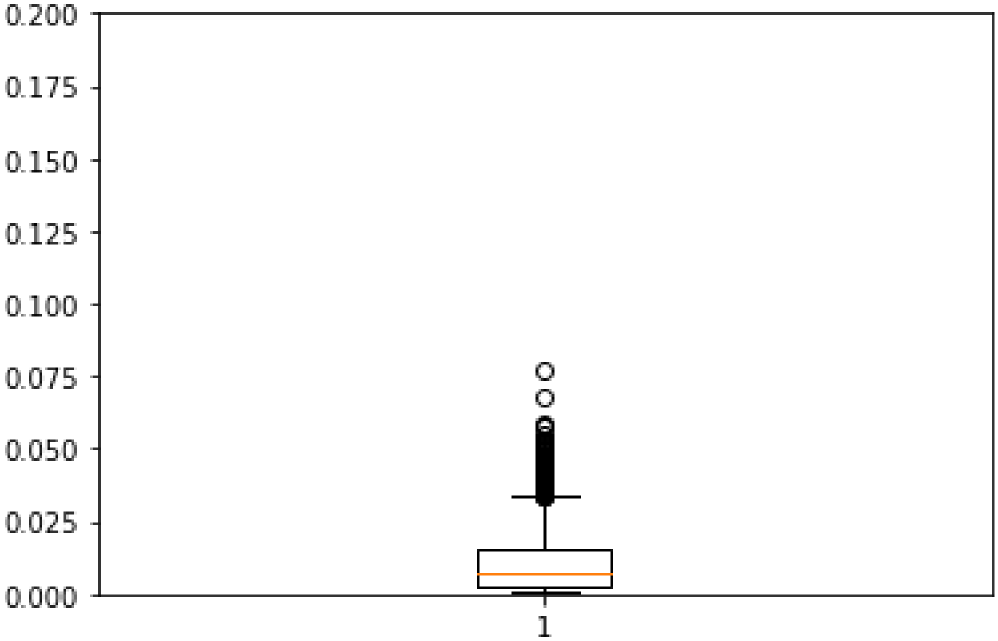


```
0.0150328948564
```





```
0.0105588601395
```



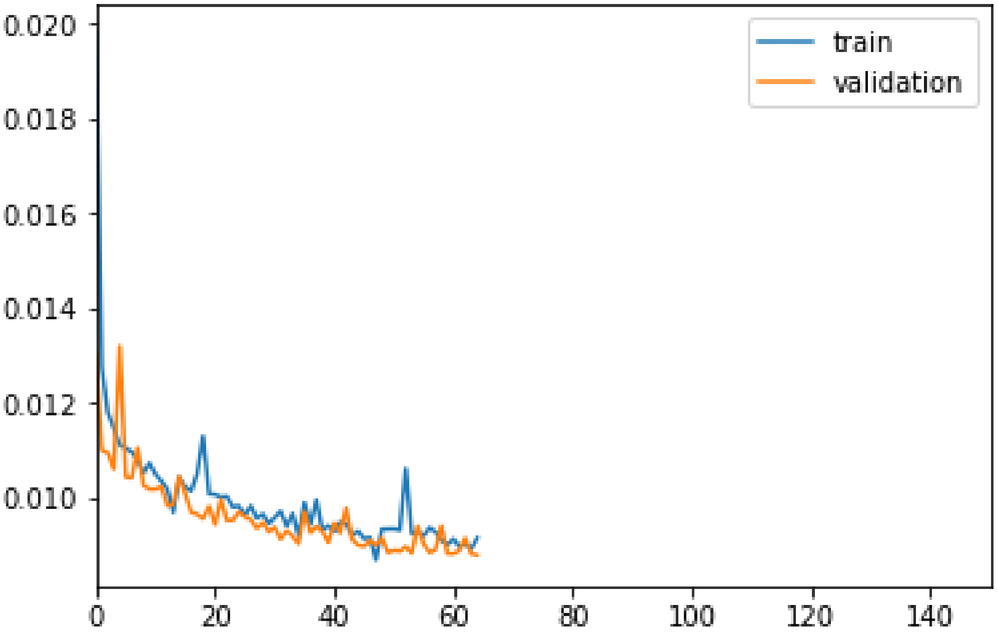
10 steps

```
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)
```

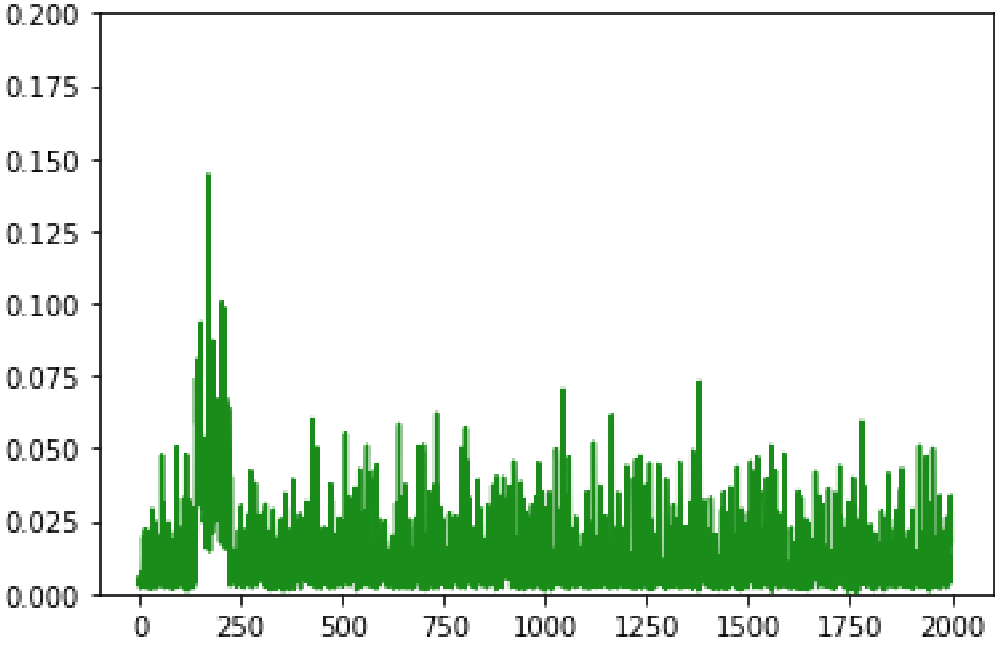
```
In [97]: model = Model(input_layer, output)
         model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [98]: train(model, tgen, vgen)
```

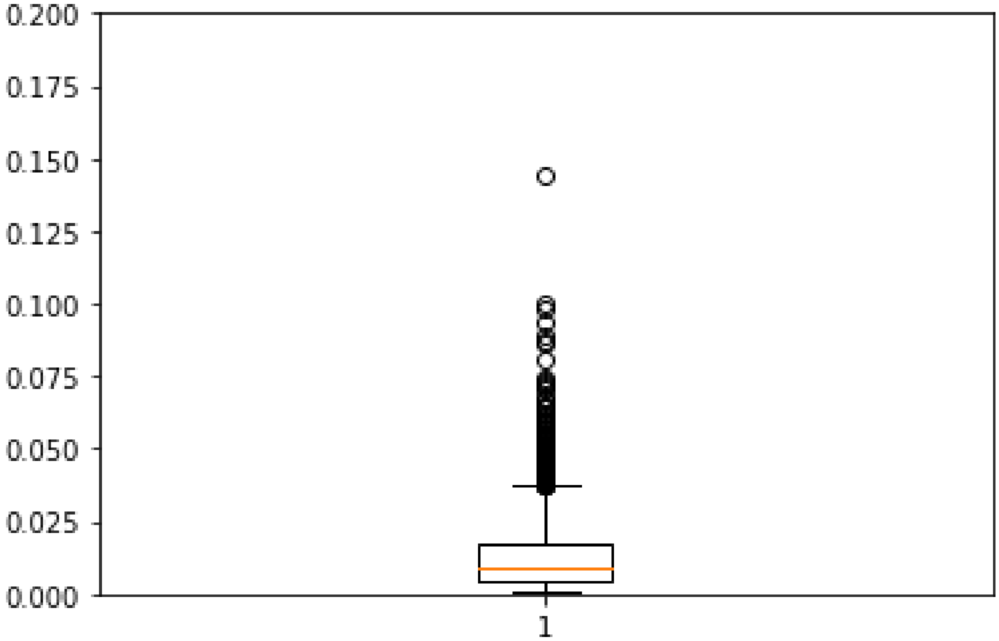


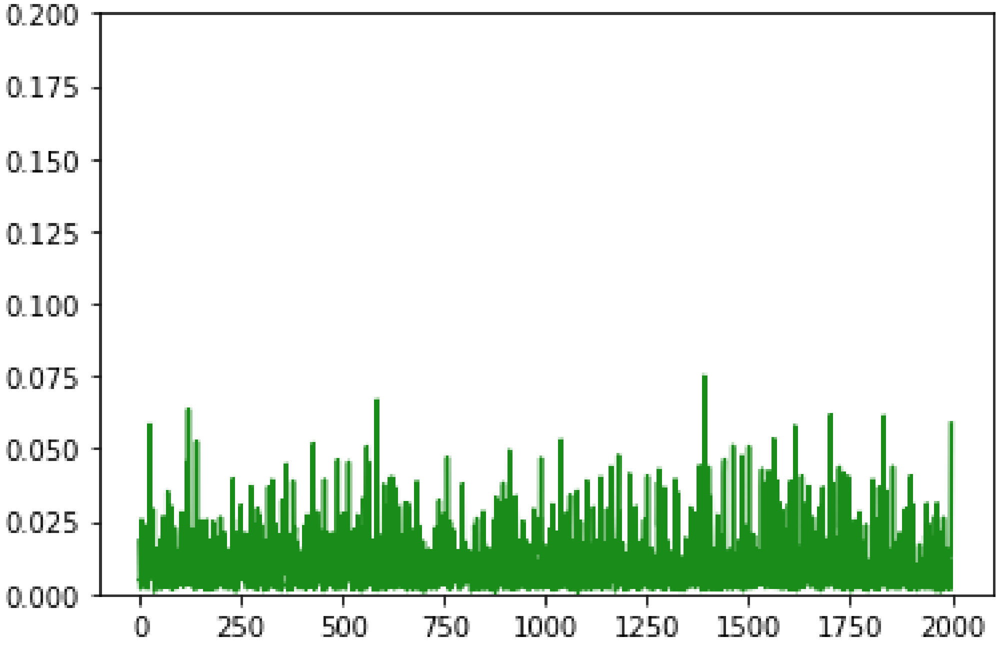
```
0.00915378945298
```

```
In [99]: test(model, test_X[0])
         test(model, test_X[2])
```

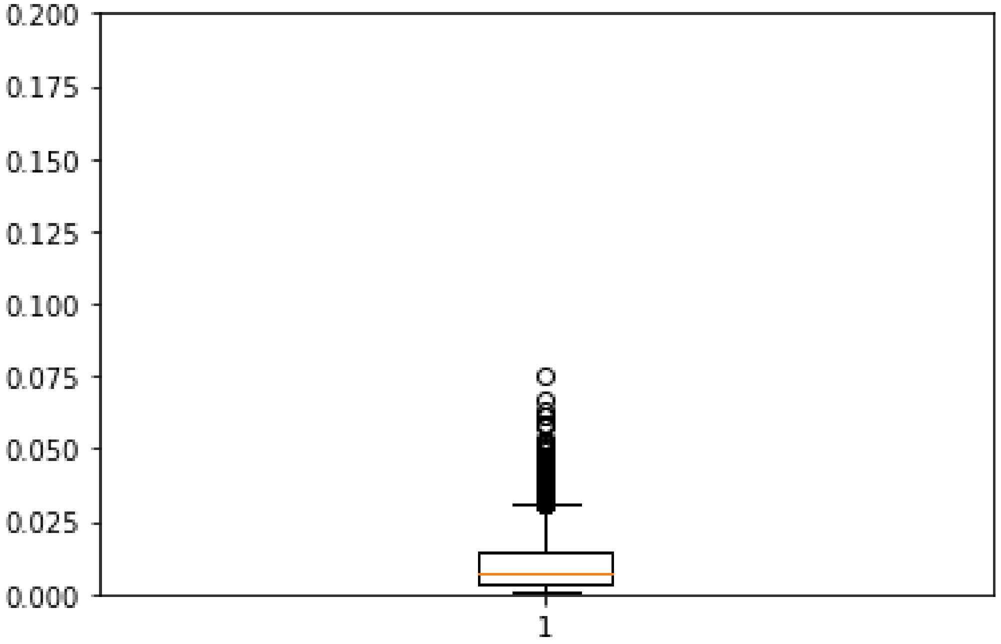


```
0.0132102074704
```





```
0.0107091823321
```



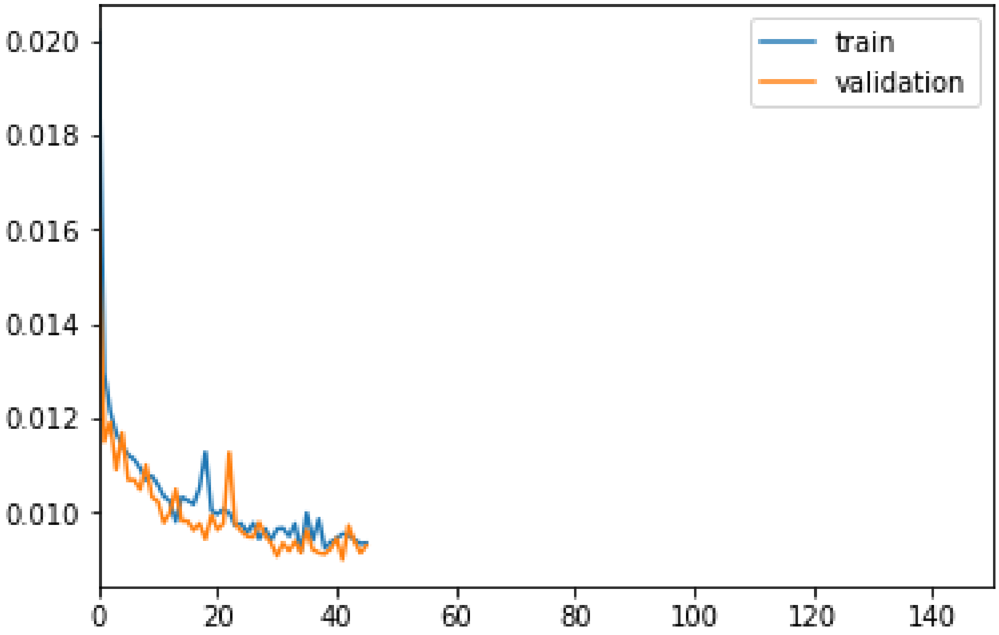
20 steps

```
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)
```

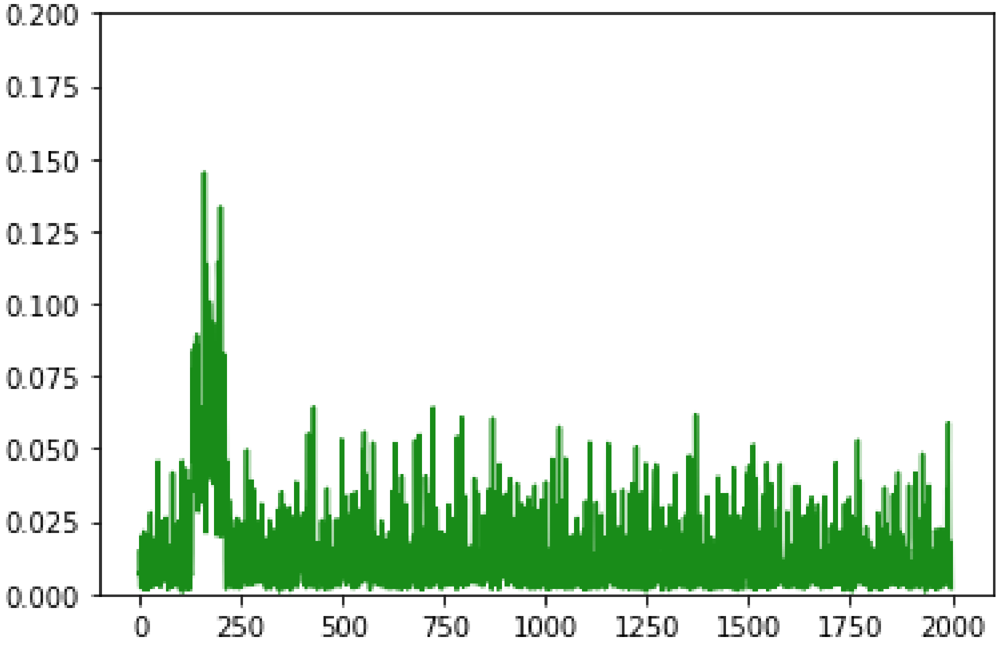
```
In [102]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [103]: train(model, tgen, vgen)
```

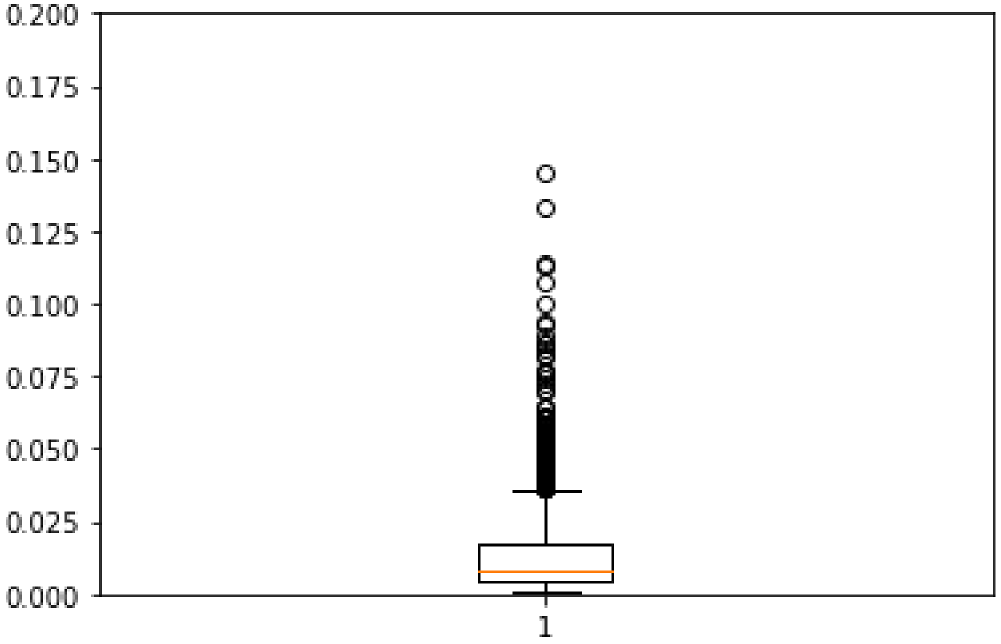


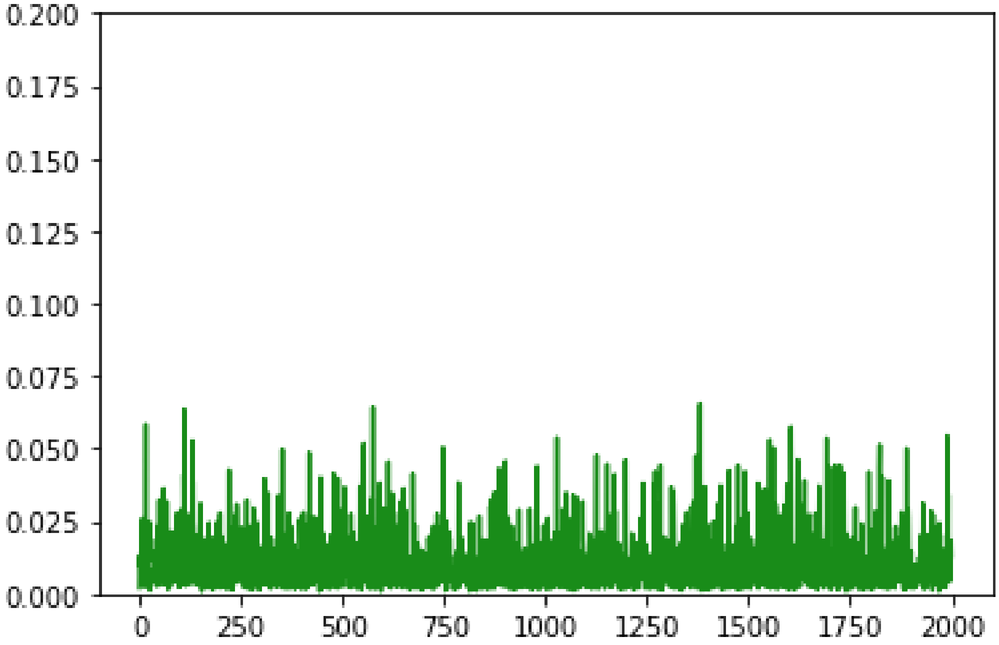
```
0.00936018063803
```

```
In [104]: test(model, test_X[0])
          test(model, test_X[2])
```

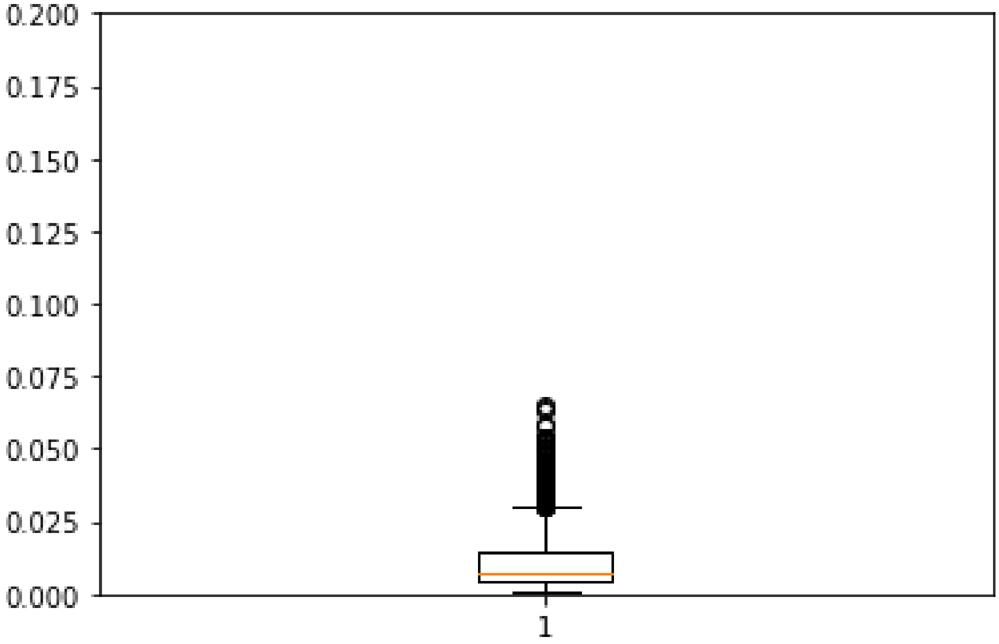


```
0.0136226747842
```



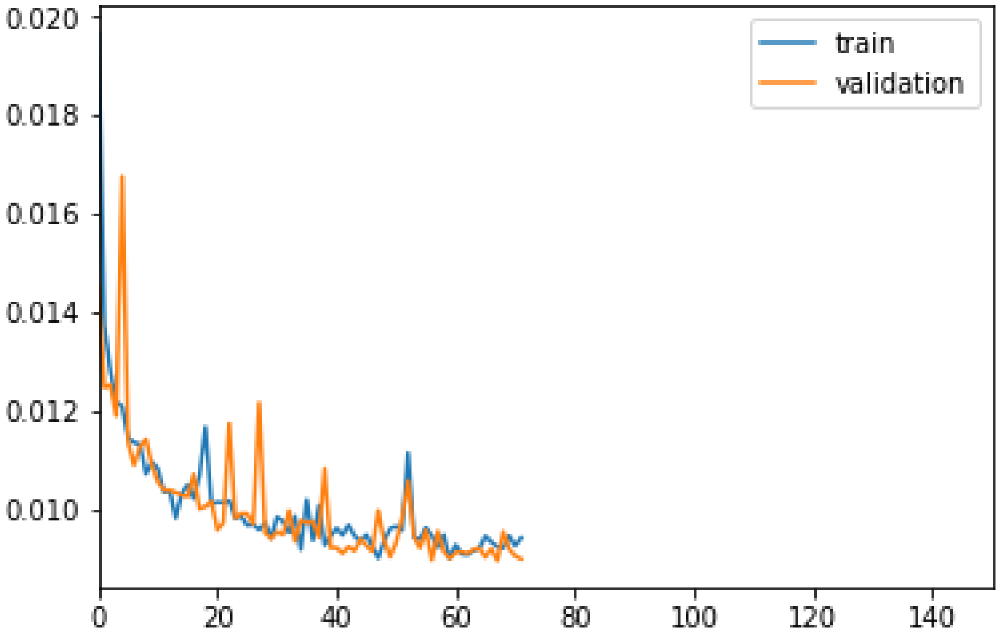


```
0.0107646180914
```



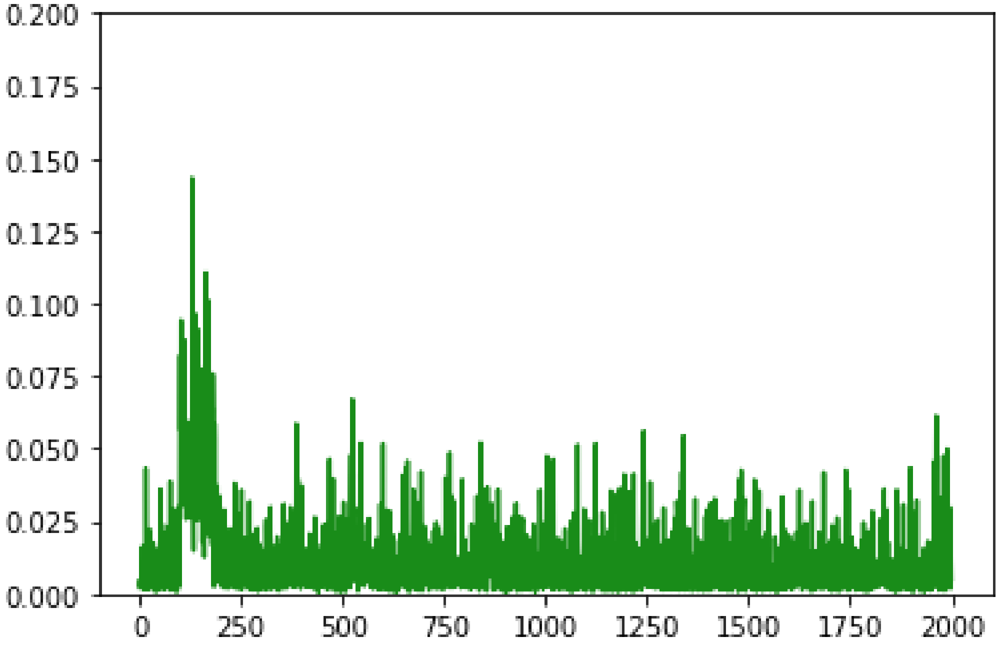
50 steps

```
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)
```

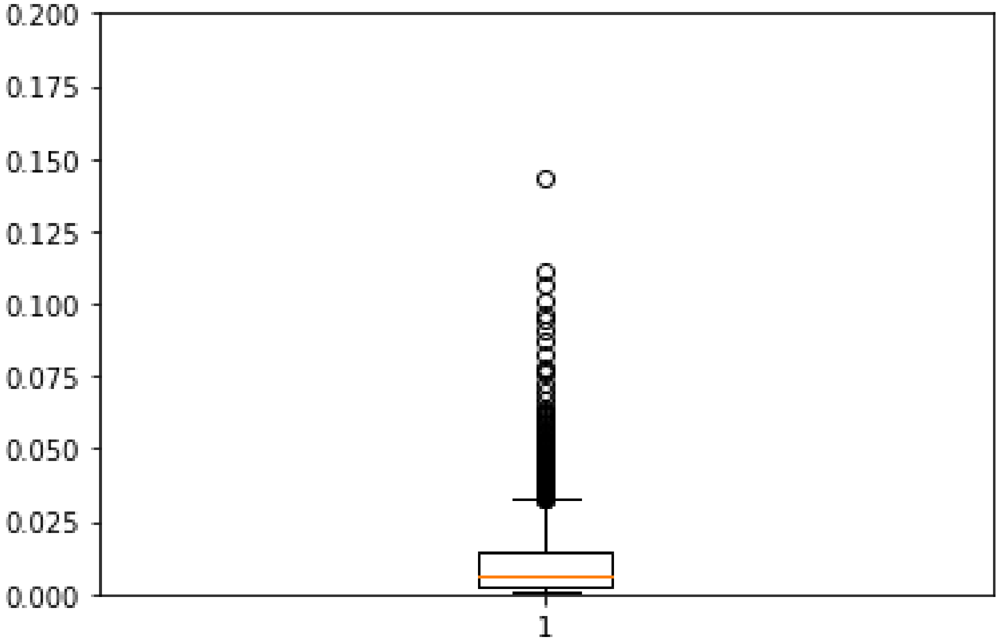


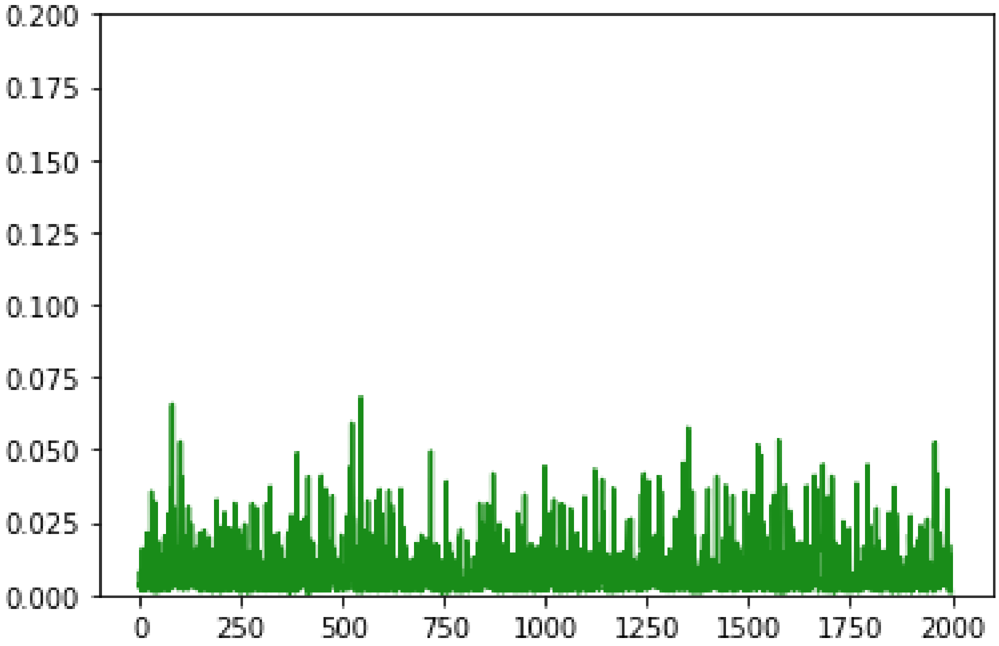
```
0.00940301835164
```

```
In [109]: test(model, test_X[0])
          test(model, test_X[2])
```

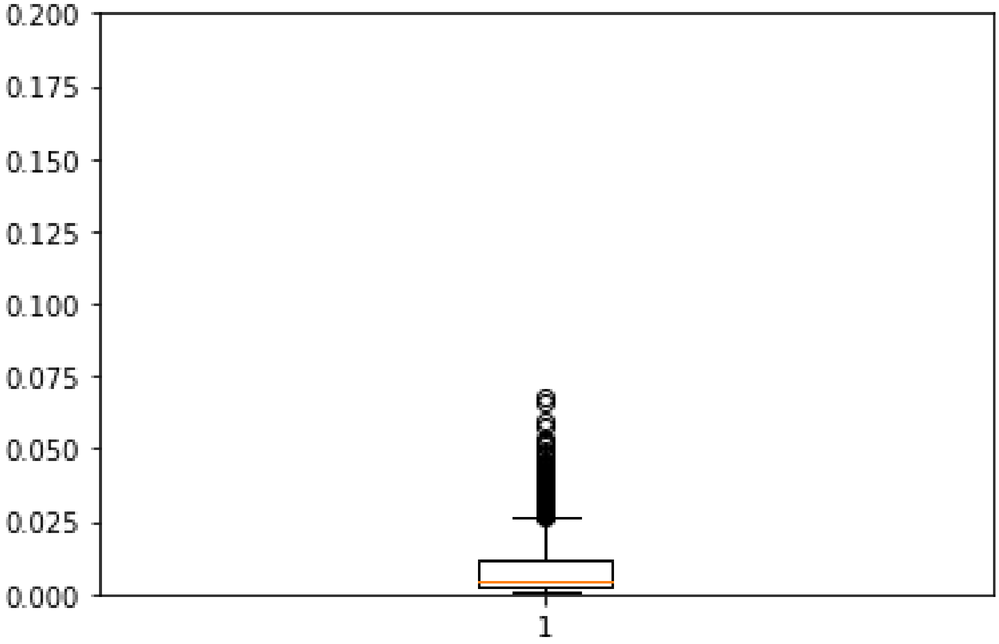


```
0.0111855266247
```





```
0.00851327775644
```



2.1.4 NN with 3 hidden layers

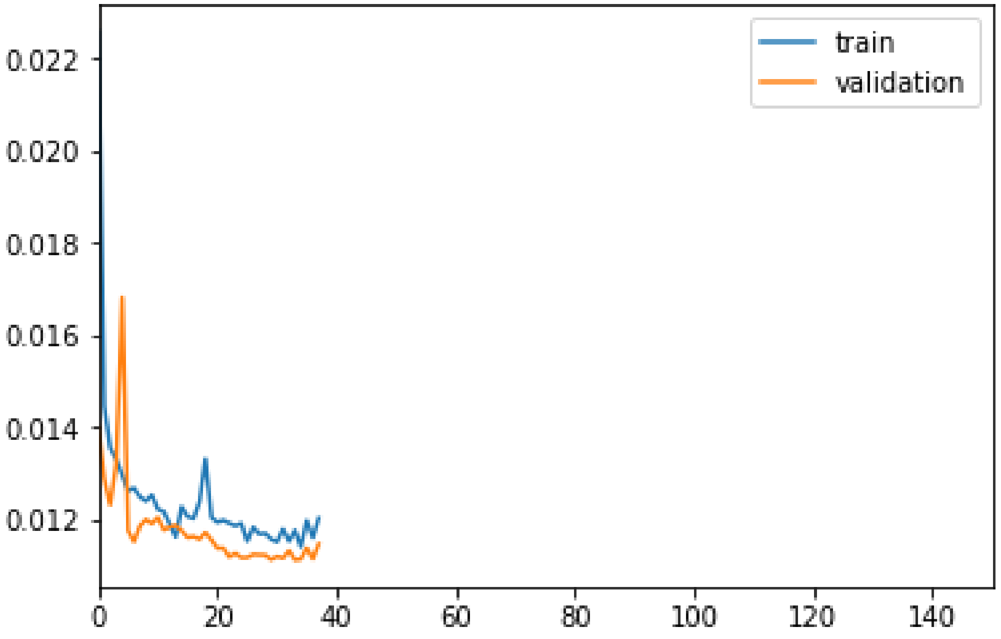
2 steps

```
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)
```

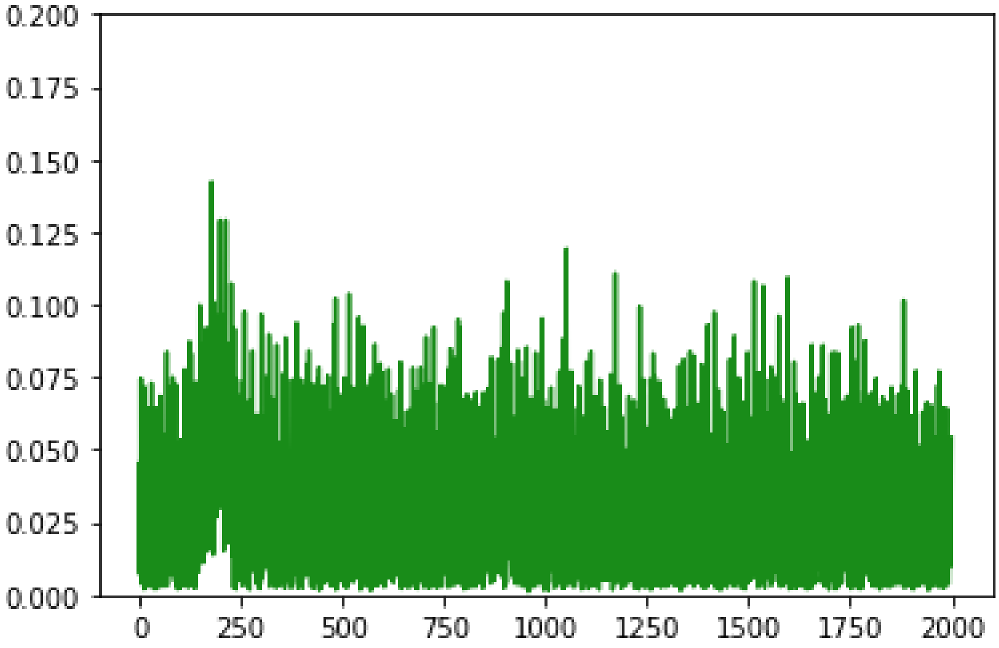
```
In [112]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [113]: train(model, tgen, vgen)
```

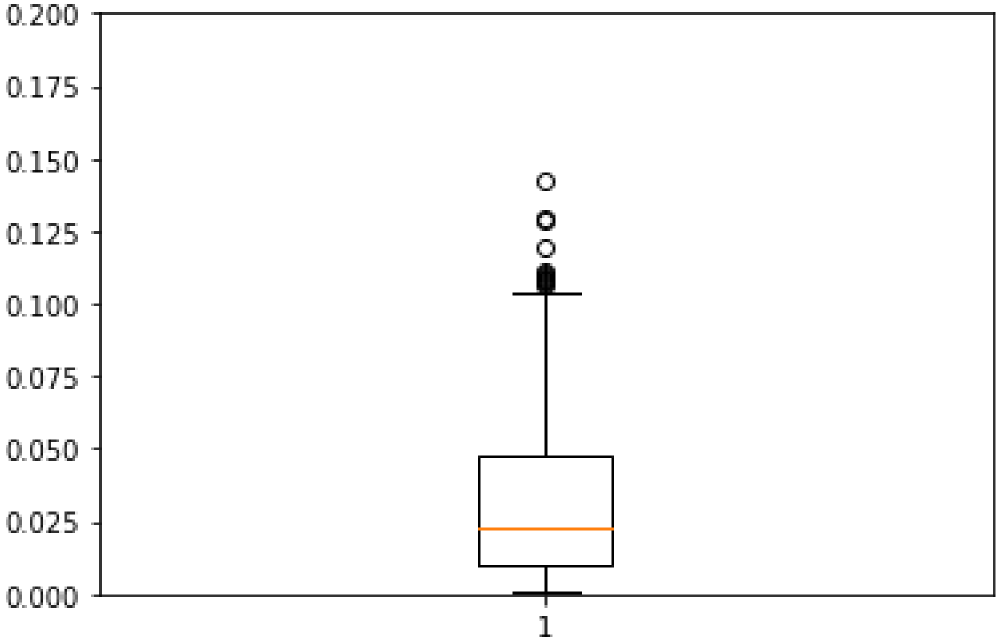


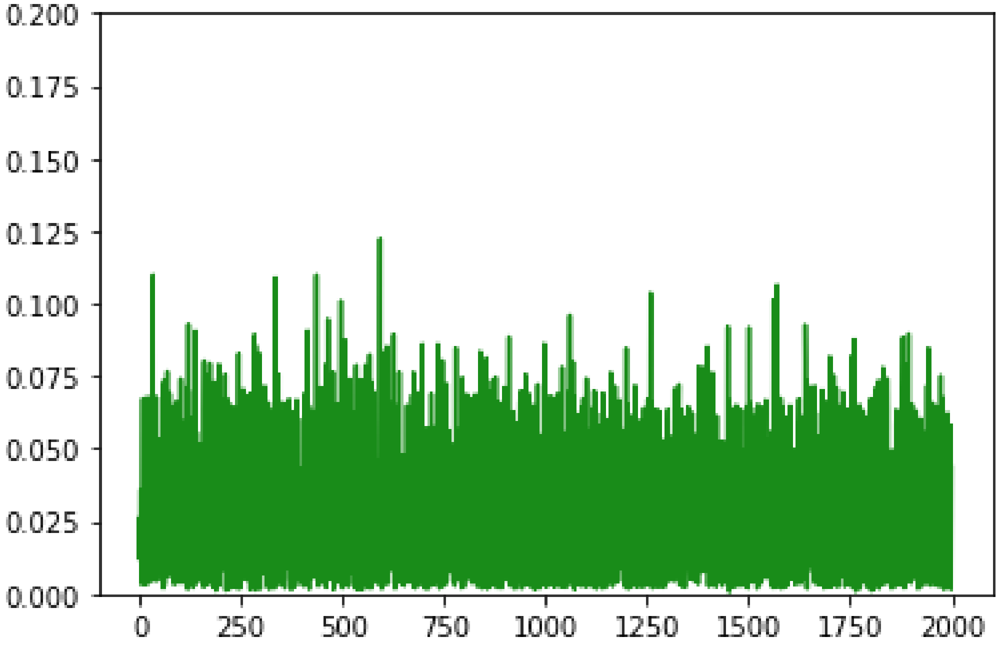
```
0.0120348802254
```

```
In [114]: test(model, test_X[0])
          test(model, test_X[2])
```

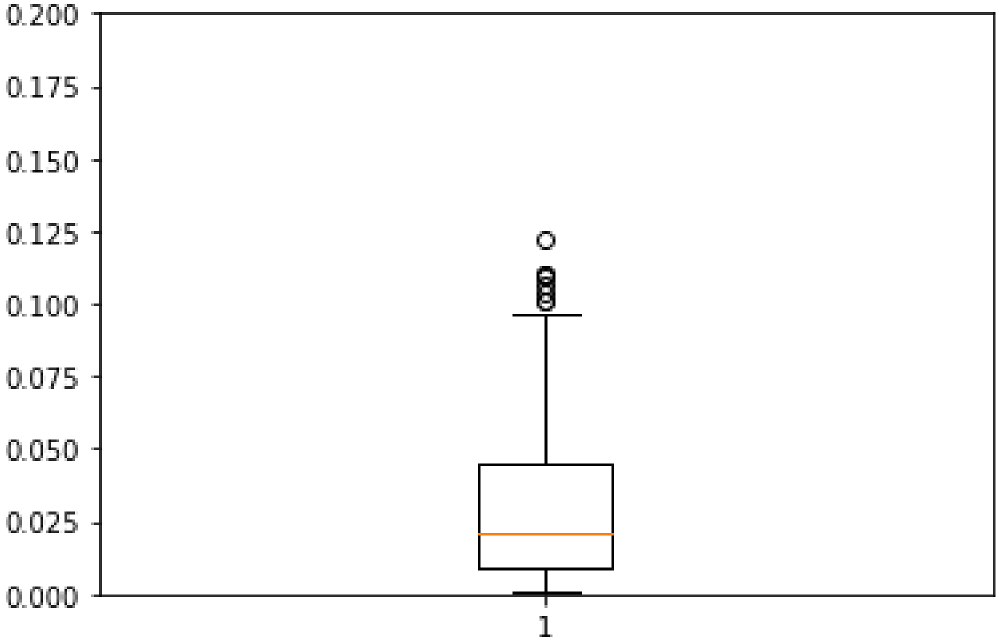


```
0.0308671526685
```





```
0.0285143535861
```



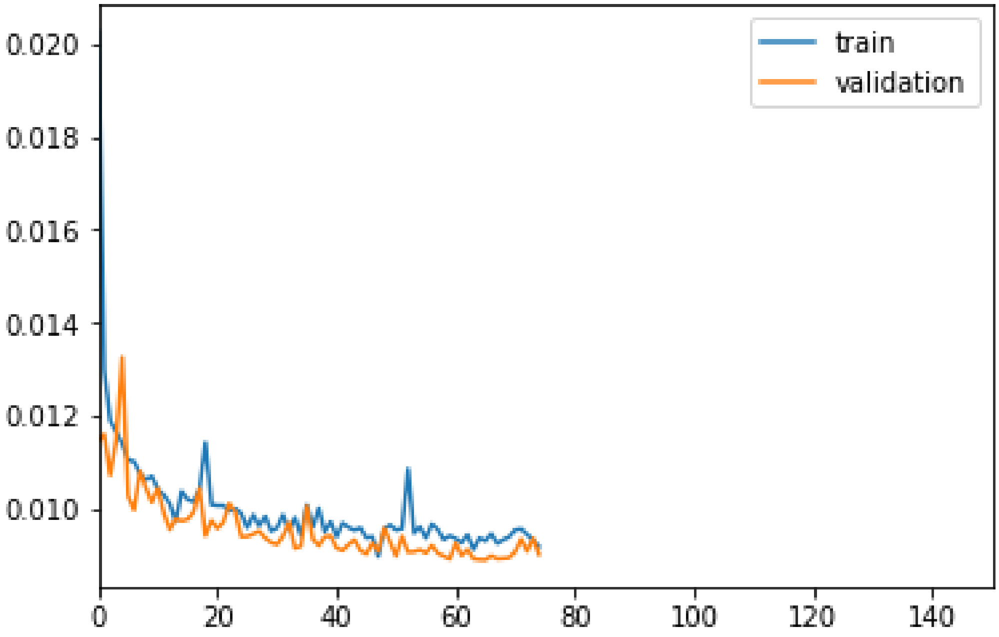
5 steps

```
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)
```

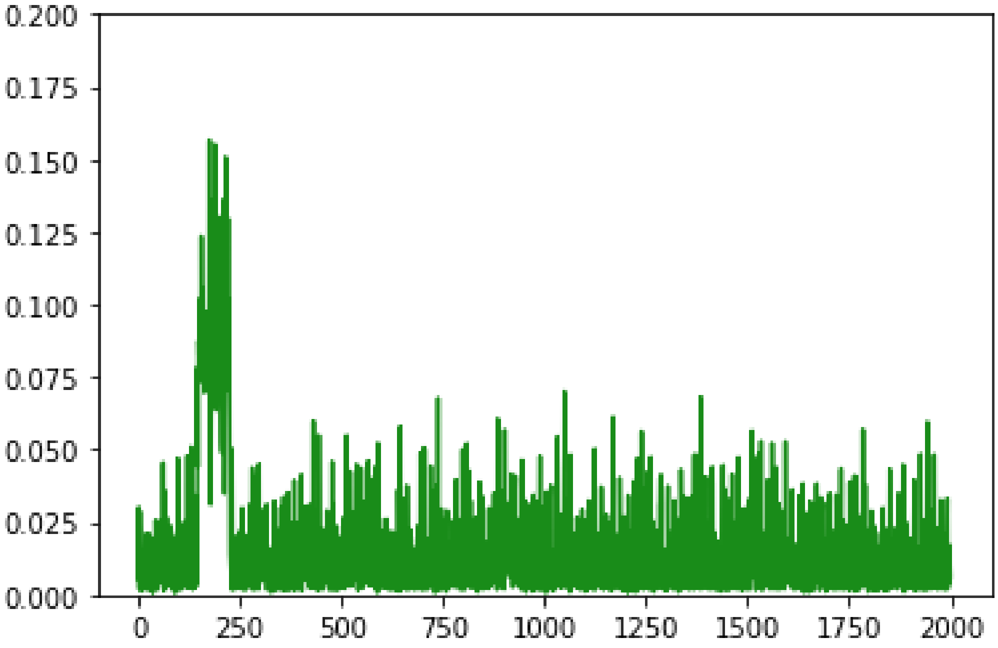
```
In [117]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [118]: train(model, tgen, vgen)
```

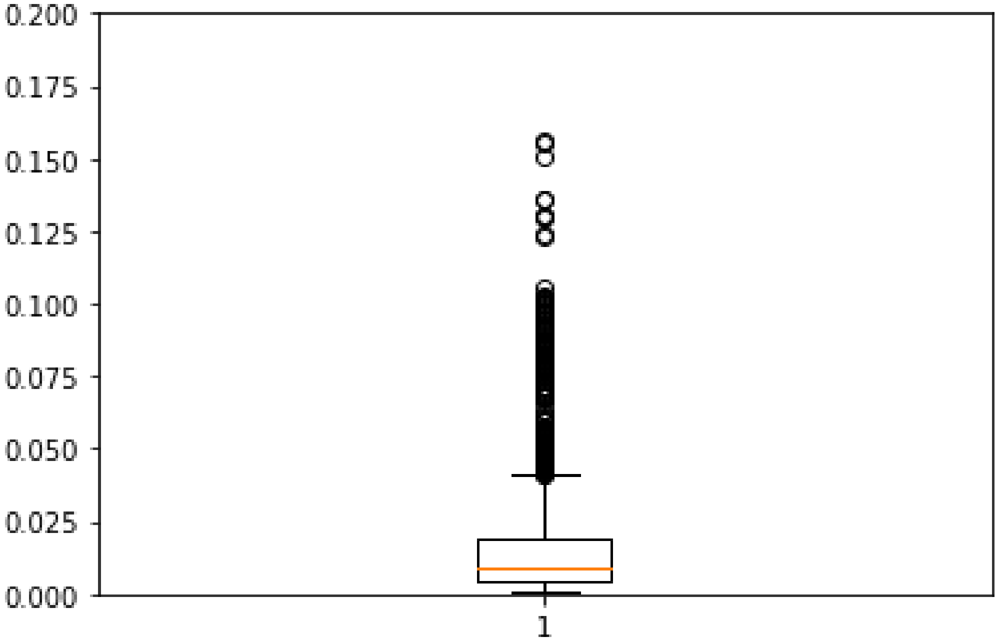


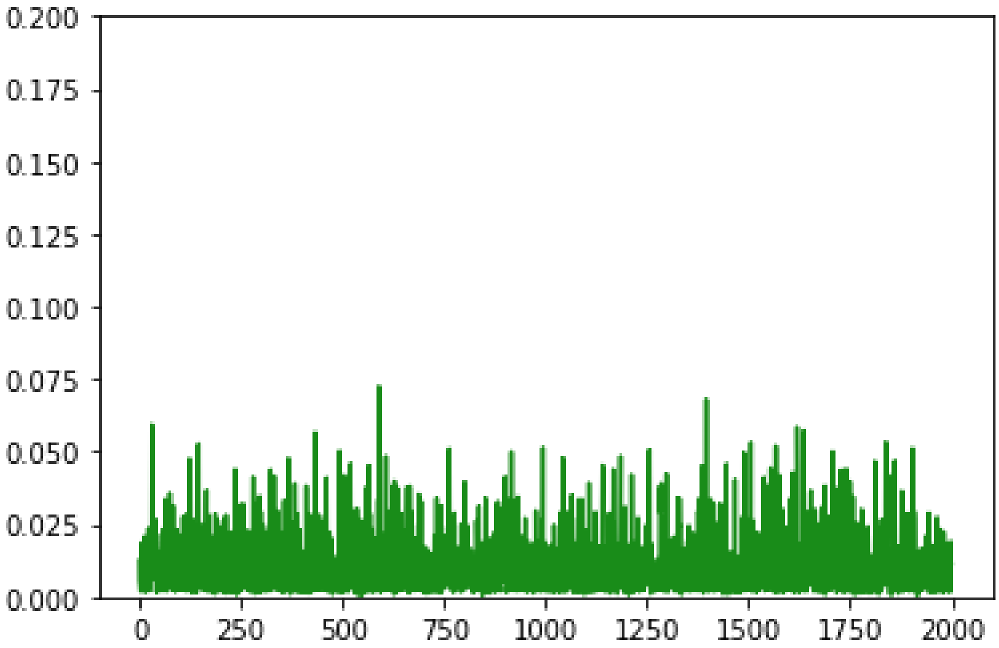
```
0.00918908930931
```

```
In [119]: test(model, test_X[0])
          test(model, test_X[2])
```

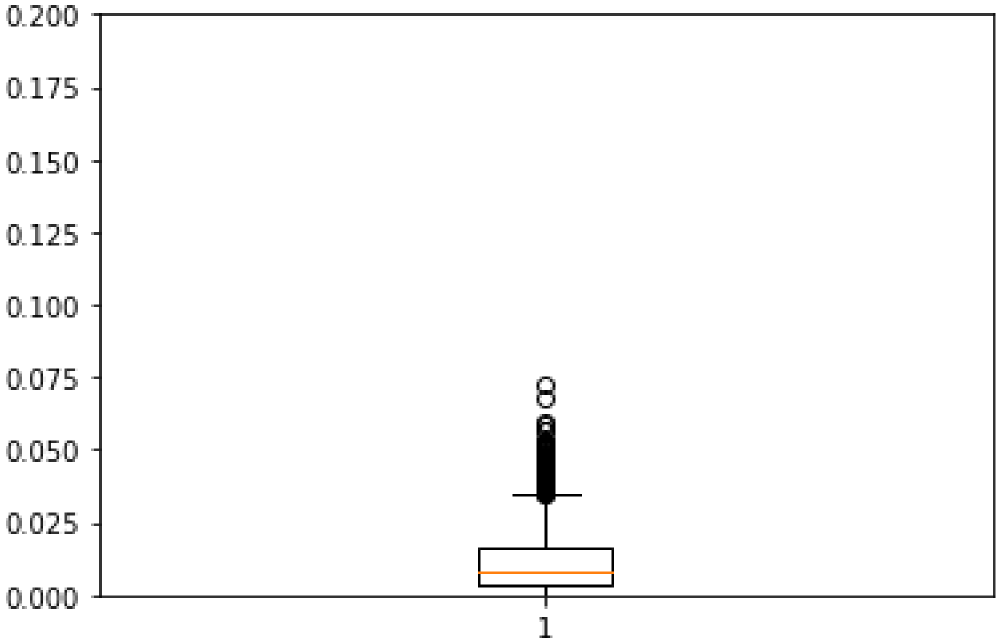


```
0.0155658080012
```





```
0.0113469584103
```



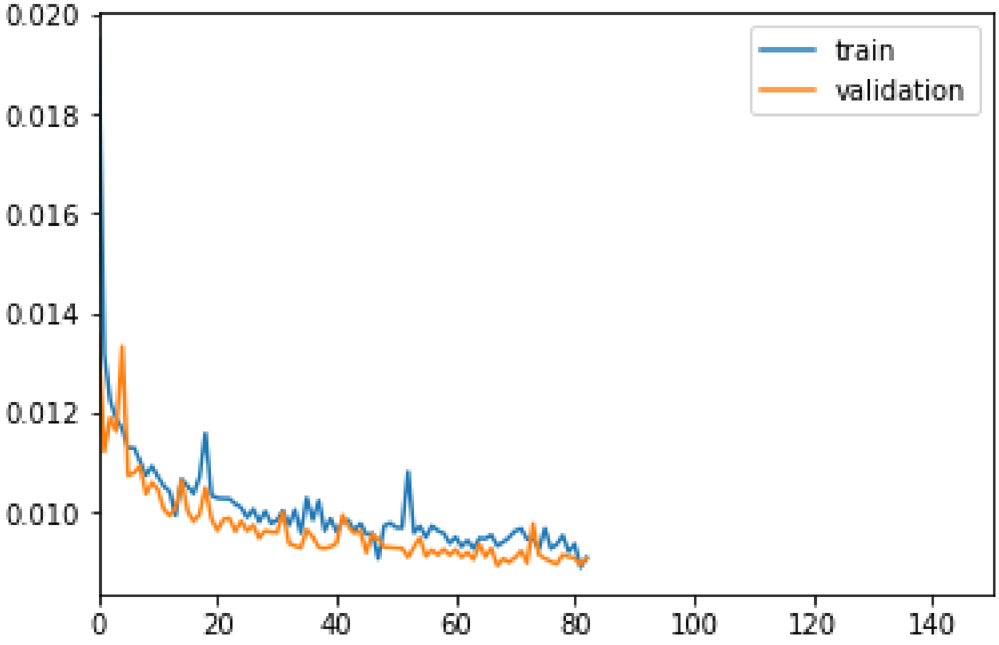
10 steps

```
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)
```

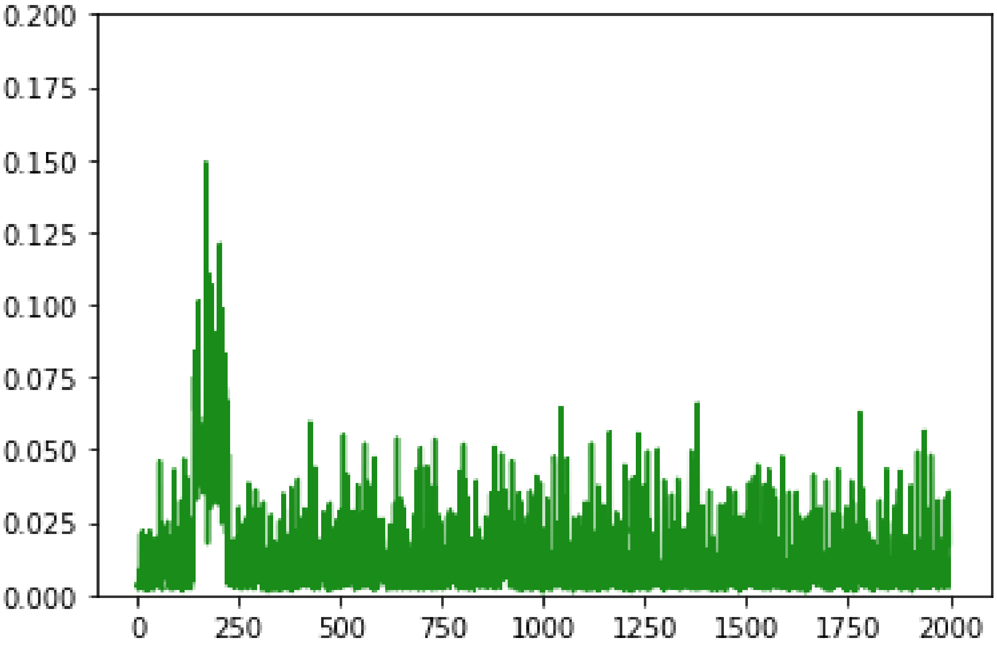
```
In [122]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [123]: train(model, tgen, vgen)
```

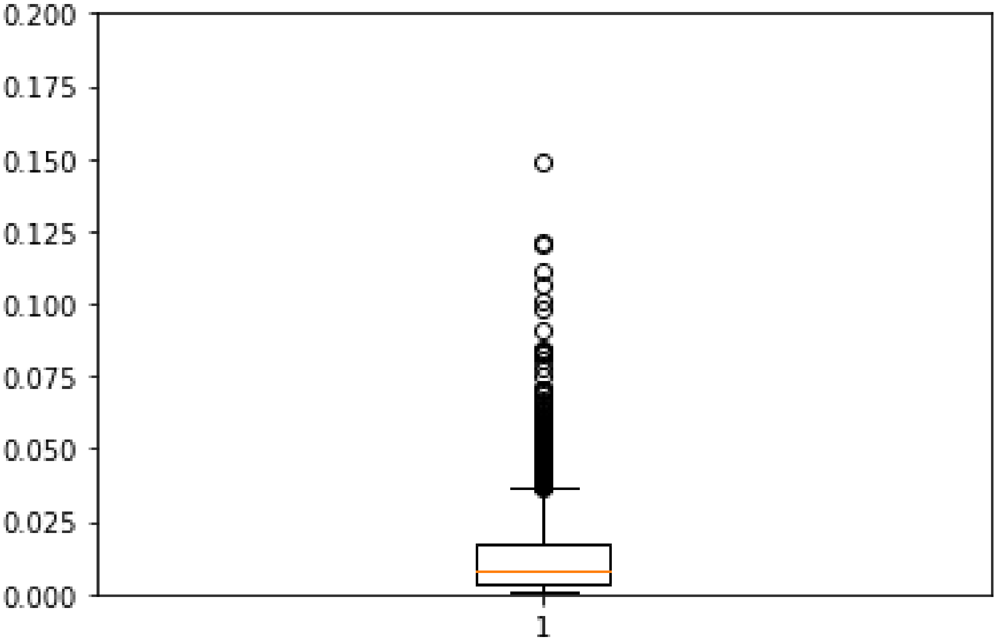


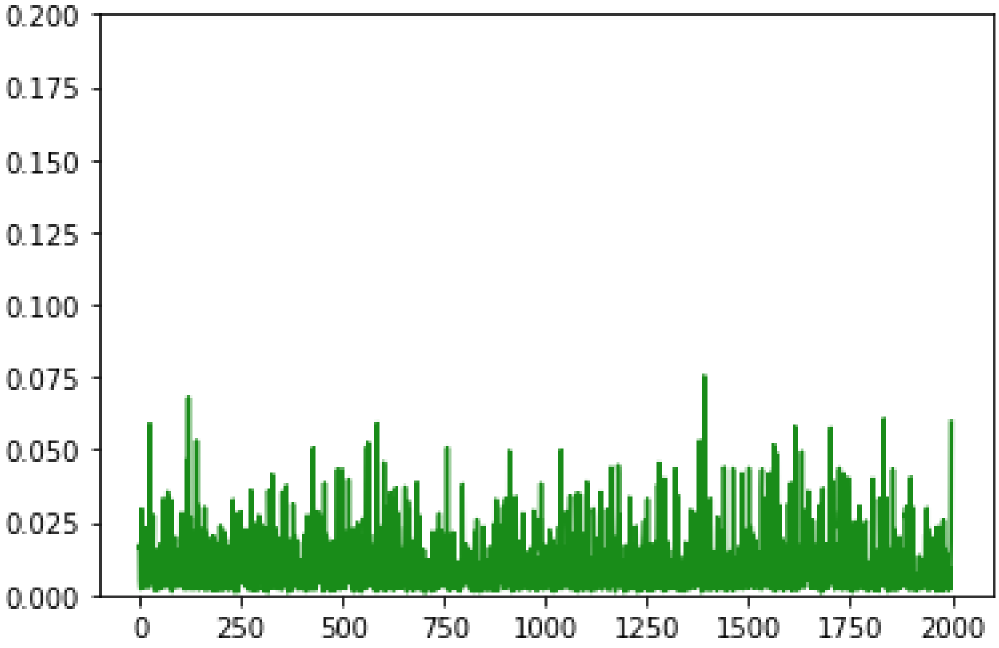
```
0.00909344443787
```

```
In [124]: test(model, test_X[0])
          test(model, test_X[2])
```

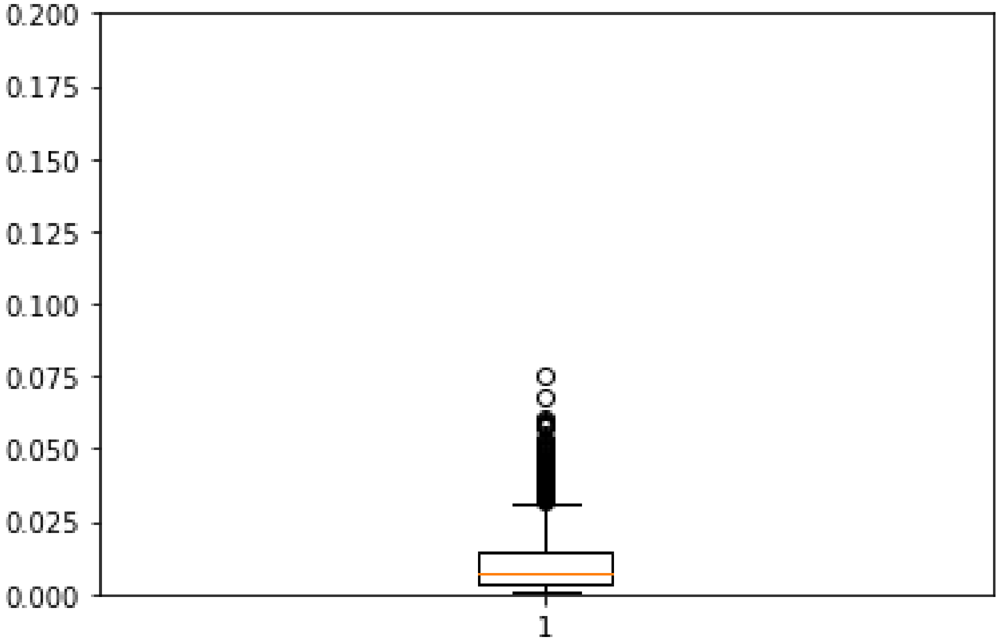


```
0.0134141862204
```





```
0.0104536839922
```



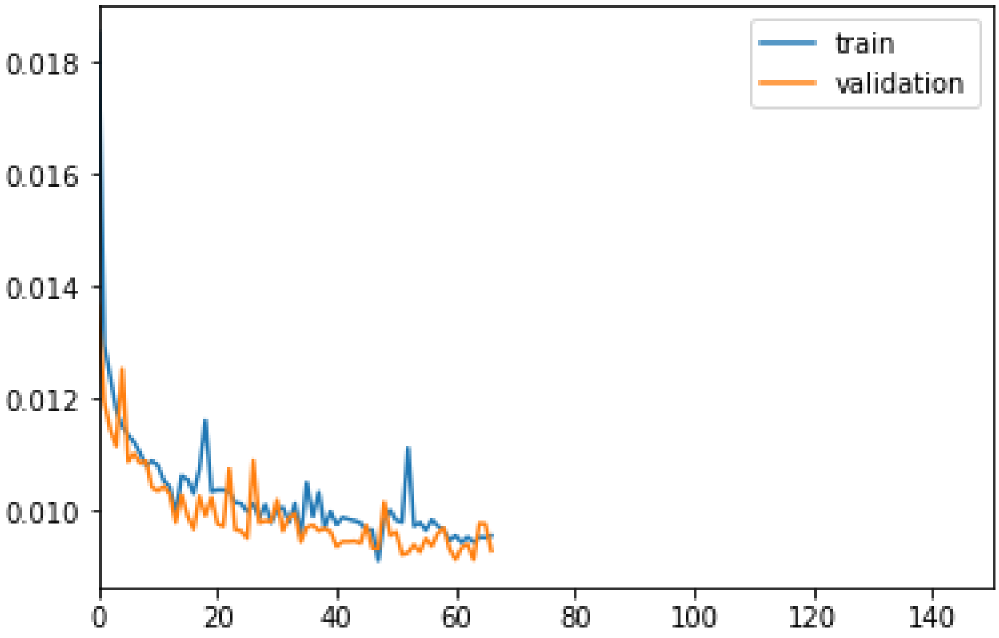
20 steps

```
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)
```

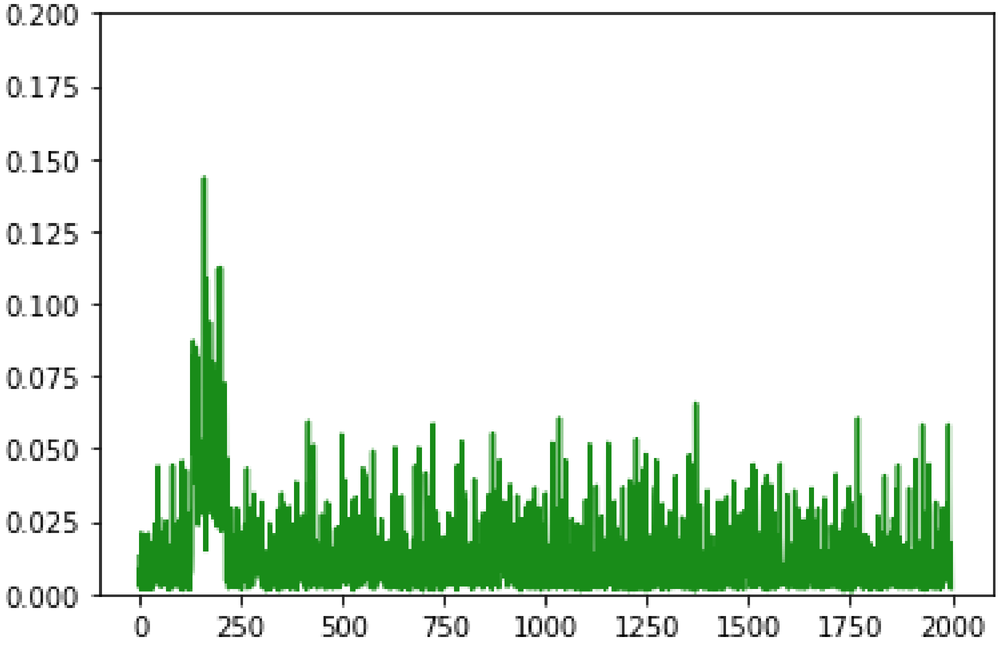
```
In [127]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [128]: train(model, tgen, vgen)
```

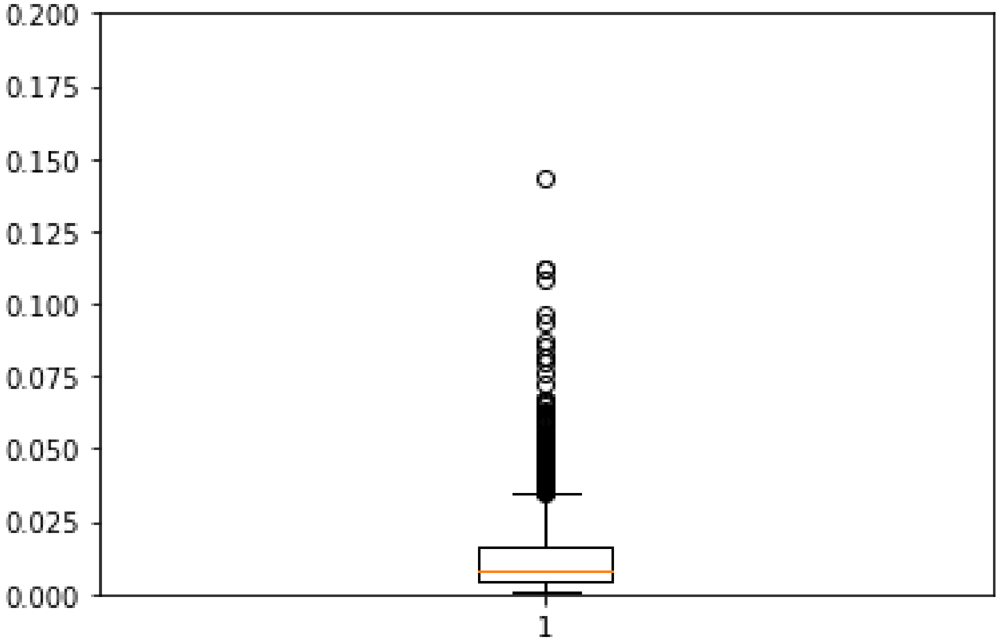


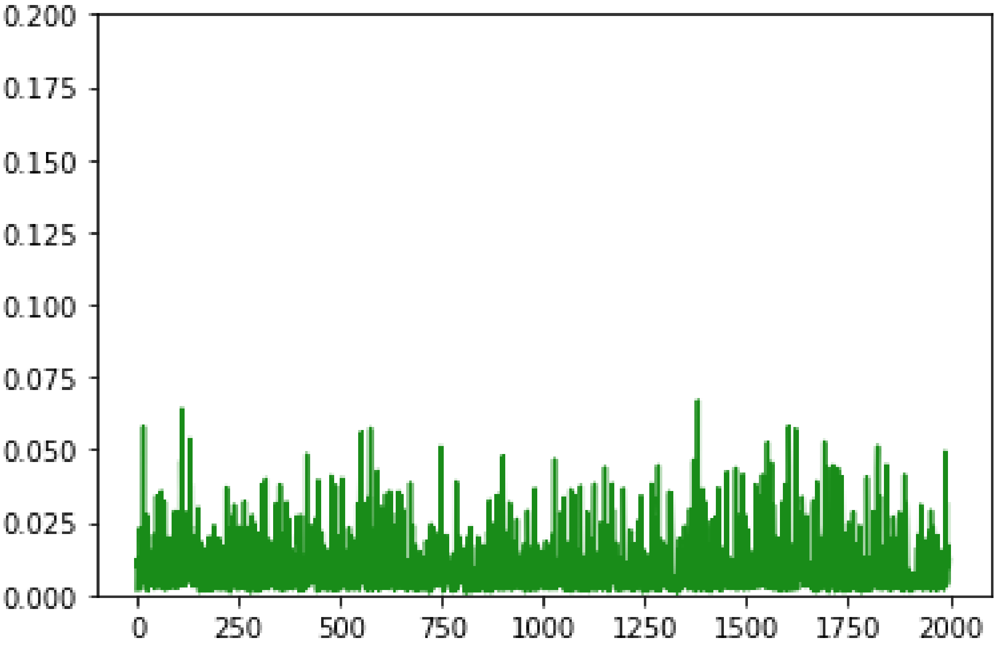
```
0.00956368522055
```

```
In [129]: test(model, test_X[0])
          test(model, test_X[2])
```

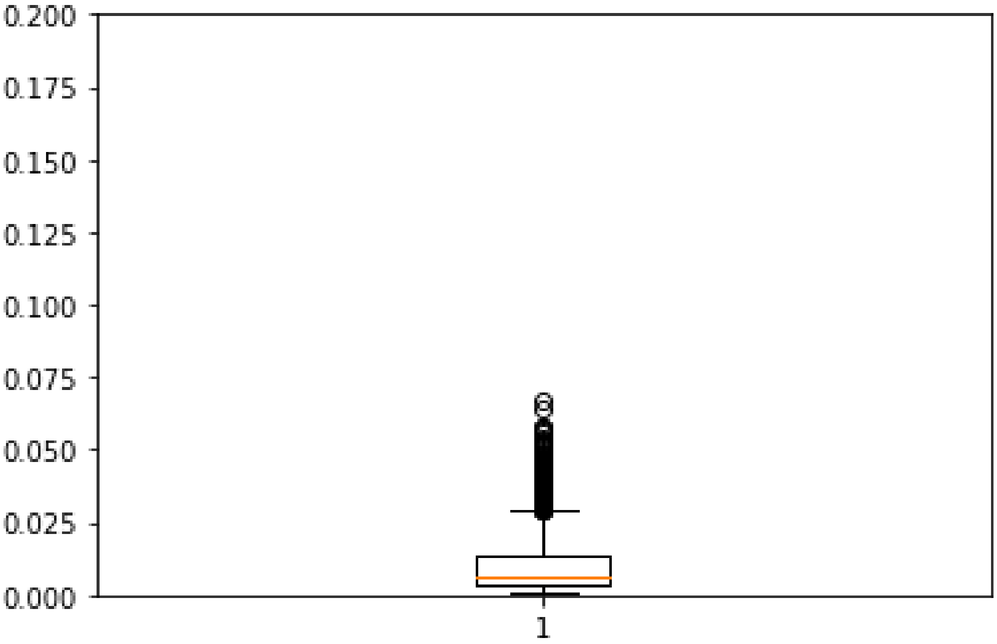


```
0.0129679544081
```





```
0.010077170734
```



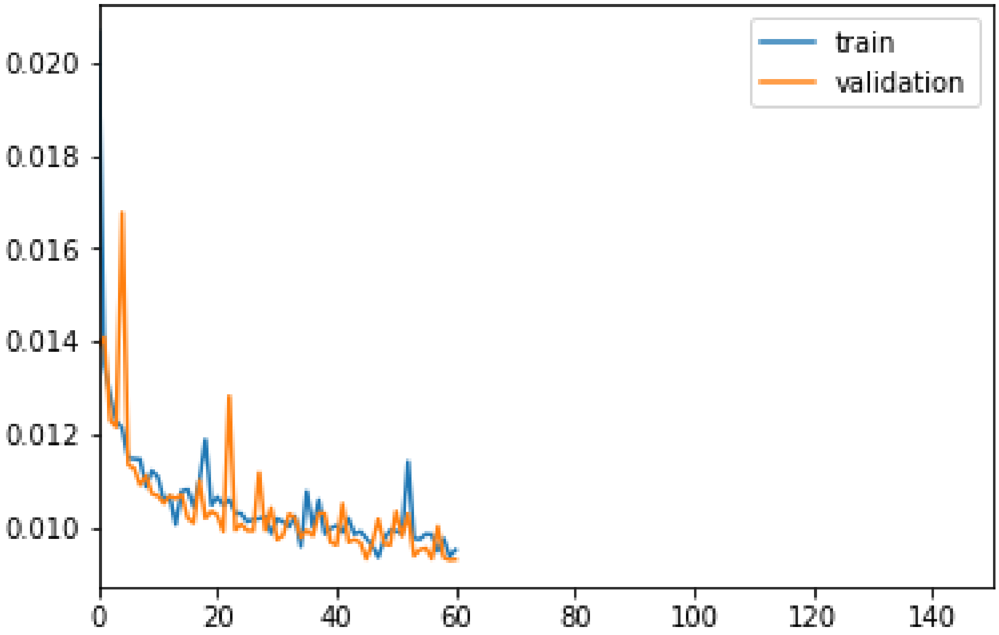
50 steps

```
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)
```

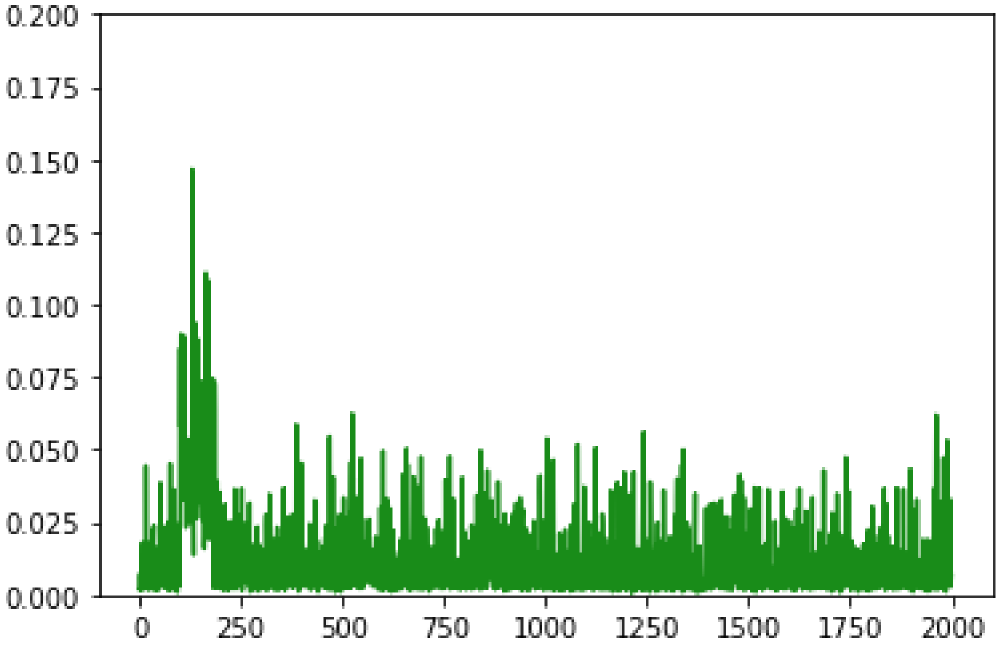
```
In [132]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [133]: train(model, tgen, vgen)
```

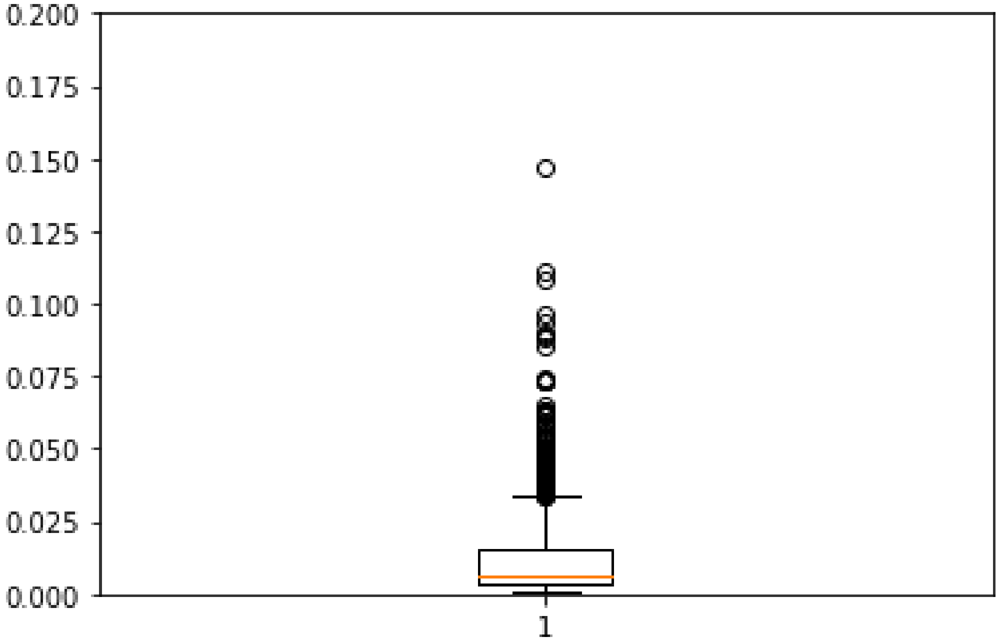


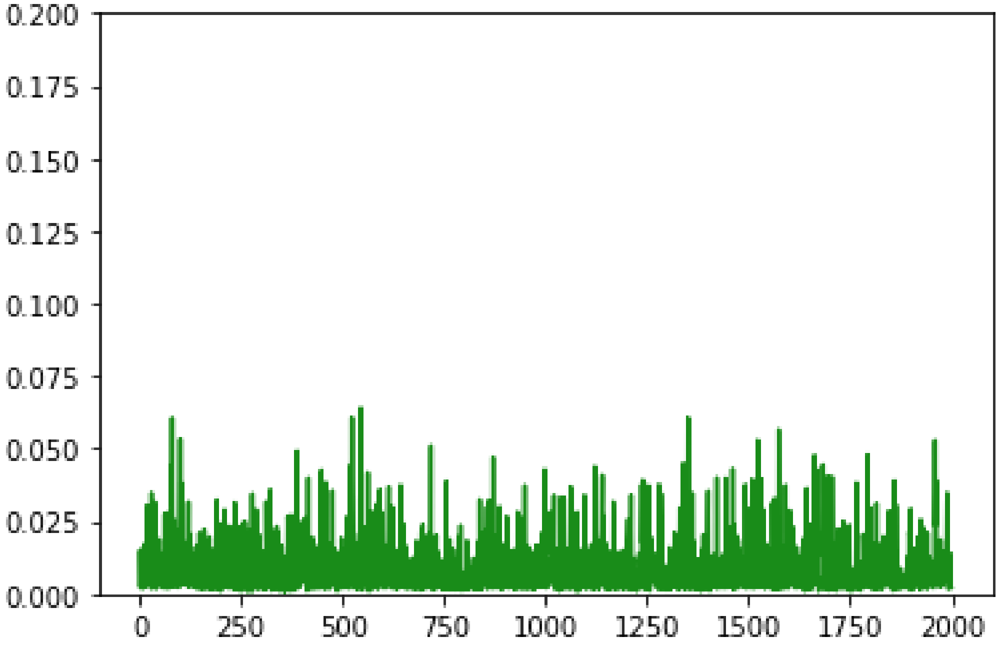
```
0.0095057254103
```

```
In [134]: test(model, test_X[0])
          test(model, test_X[2])
```

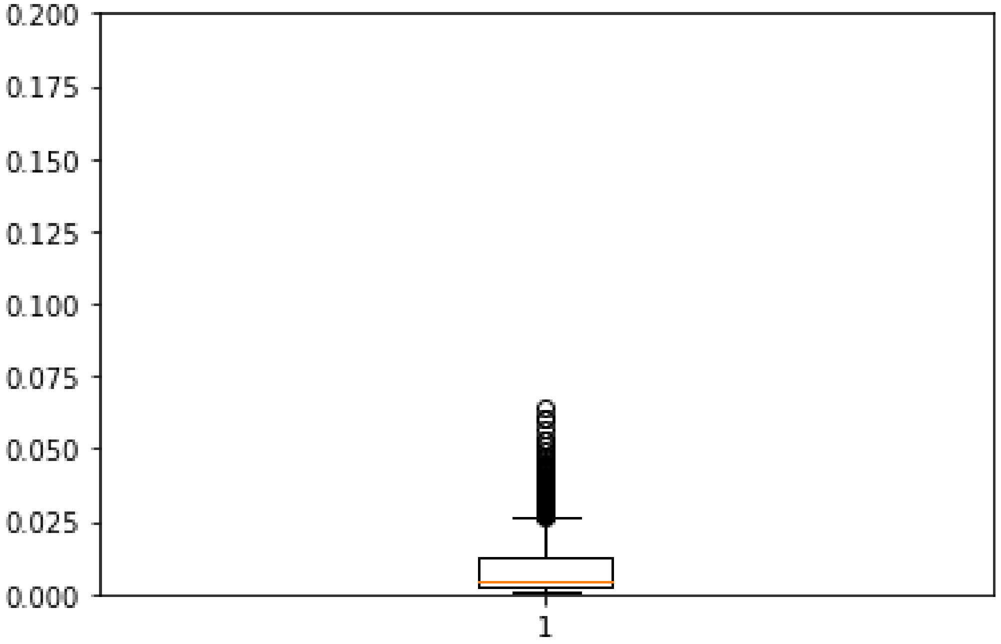


```
0.0116448599708
```





```
0.0088486452308
```



2.1.5 RNN with 1 GRU layers

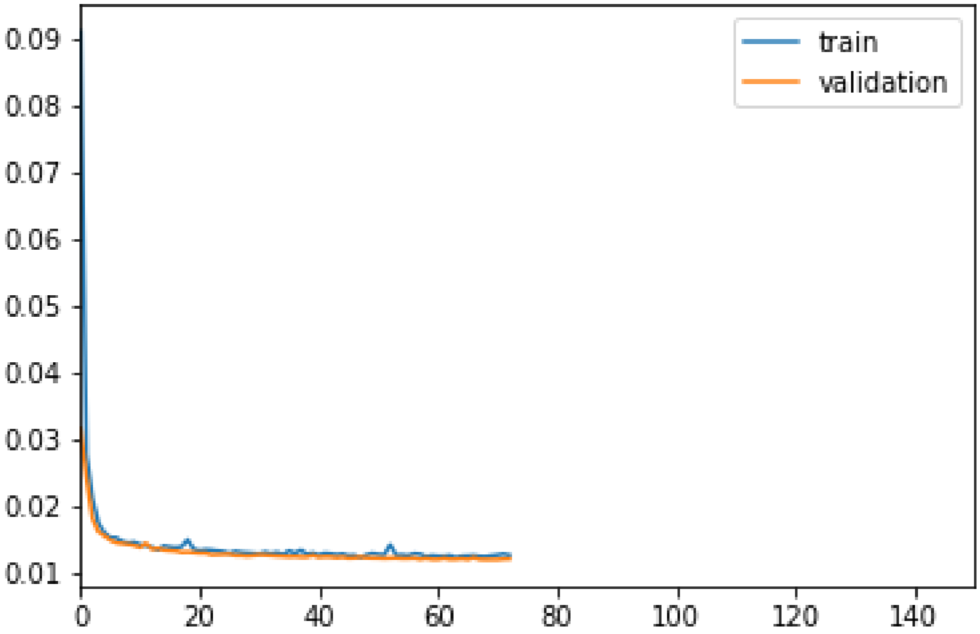
2 steps

```
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)
```

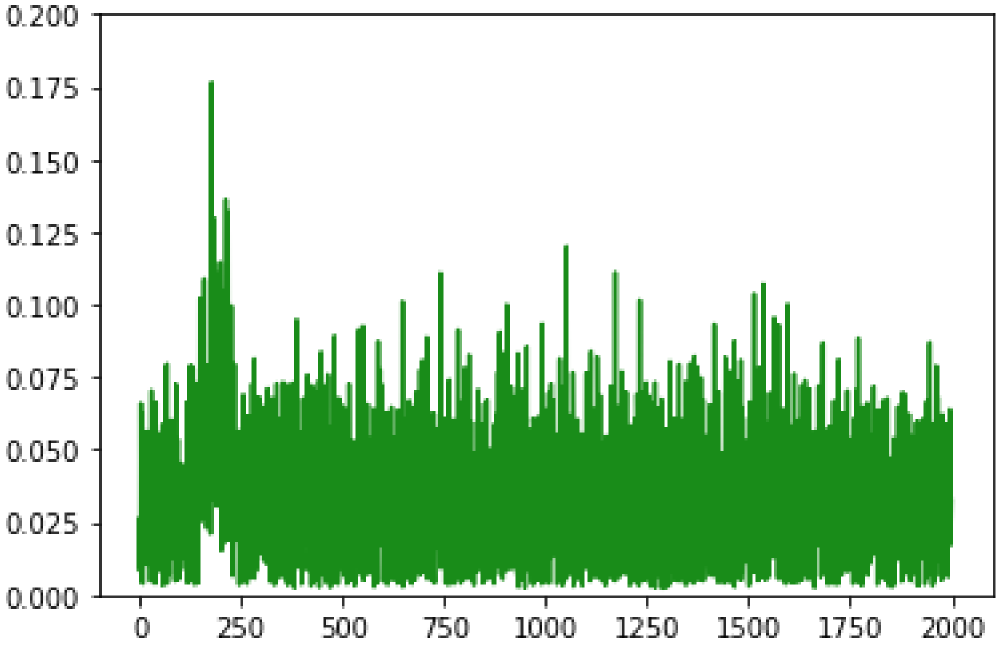
```
In [137]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [138]: train(model, tgen, vgen)
```

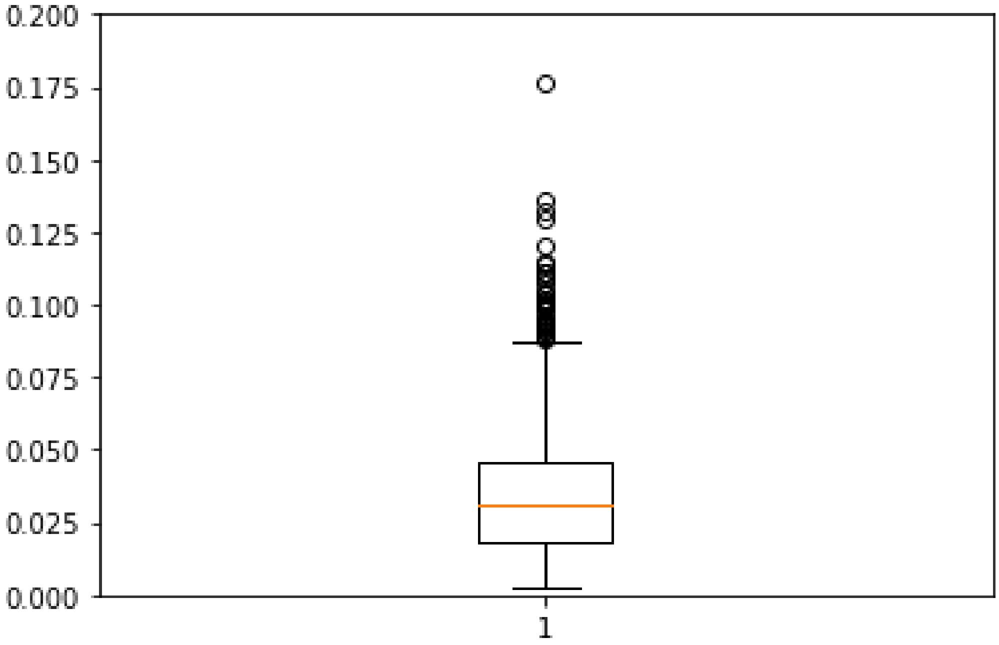


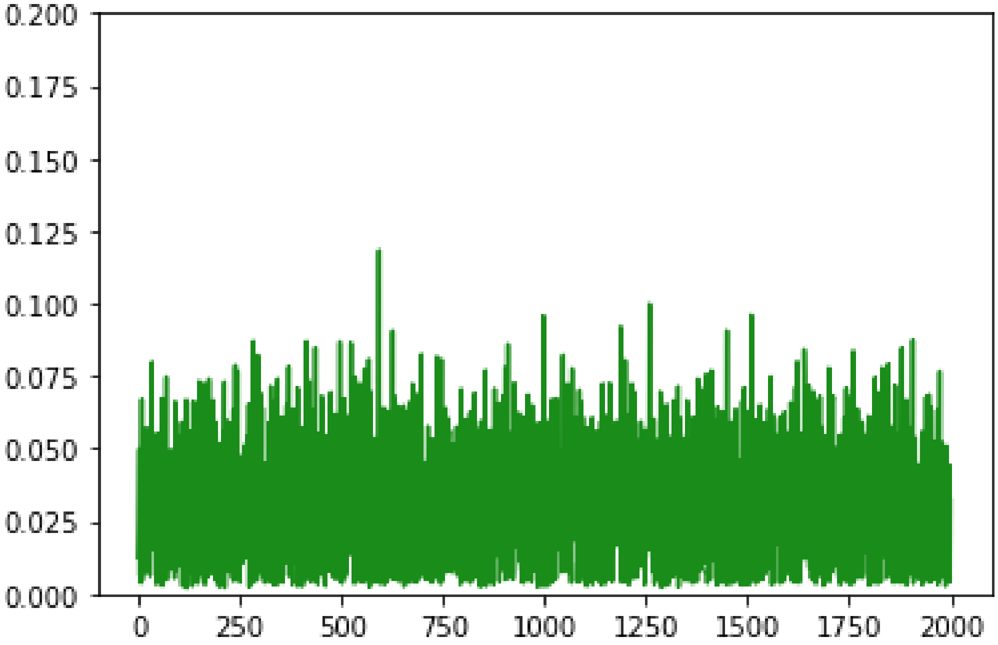
```
0.0125035220452
```

```
In [139]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

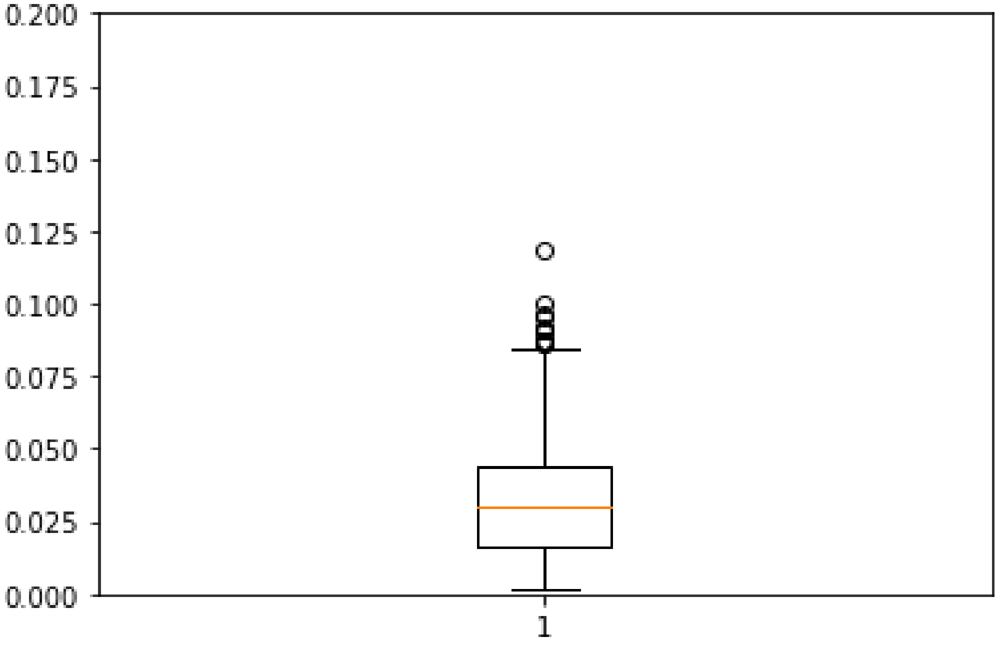


```
0.0336711179175
```





```
0.0316732637591
```



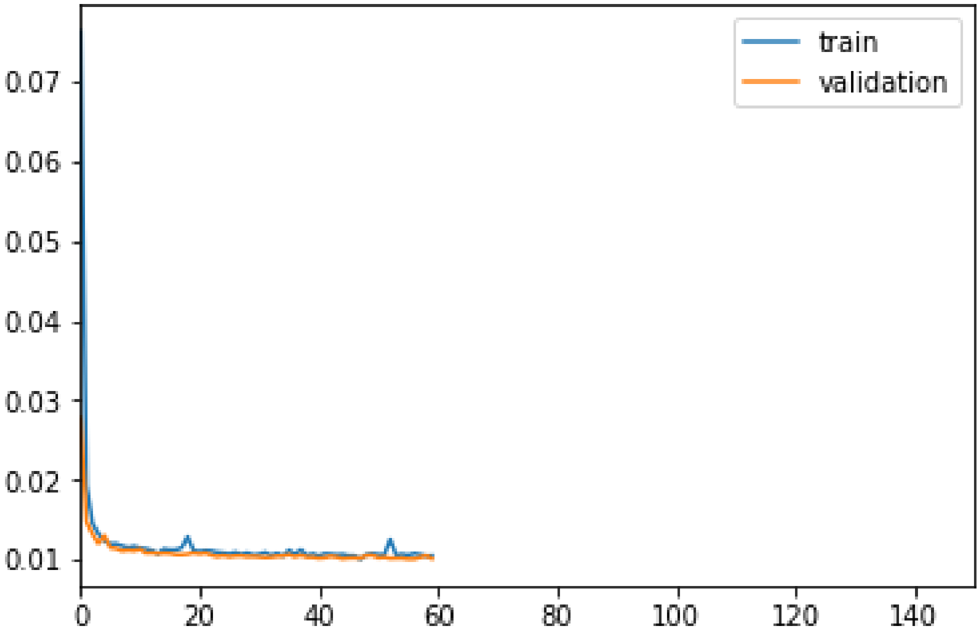
5 steps

```
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)
```

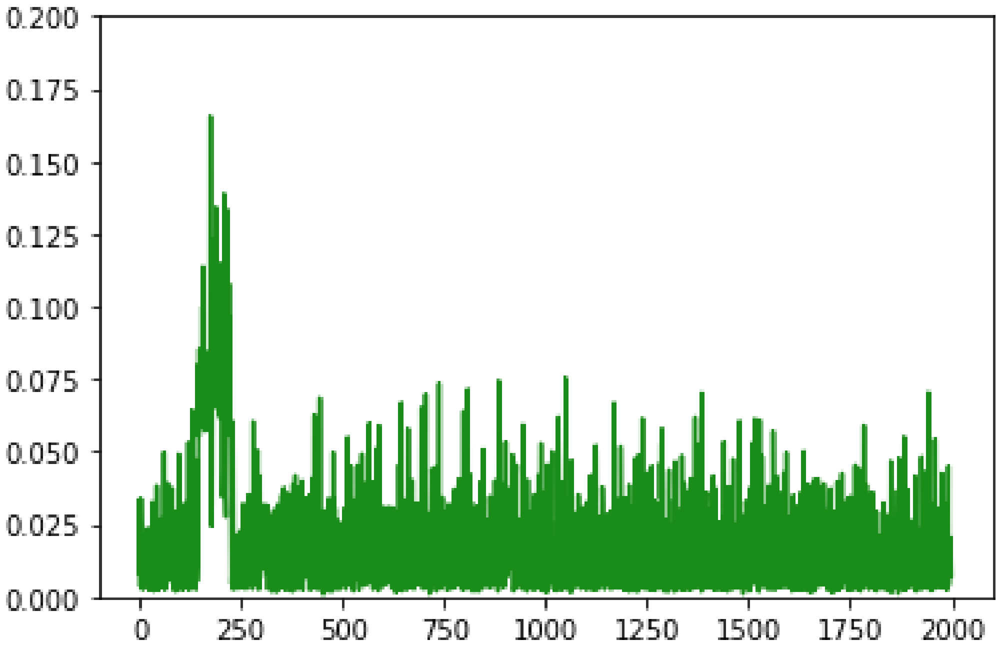
```
In [142]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [143]: train(model, tgen, vgen)
```

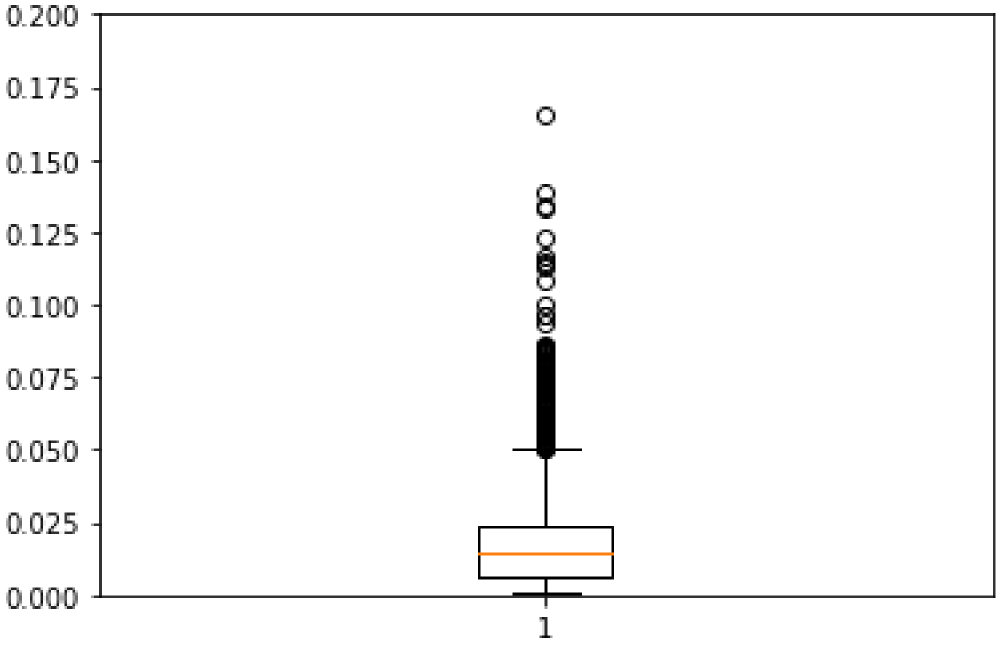


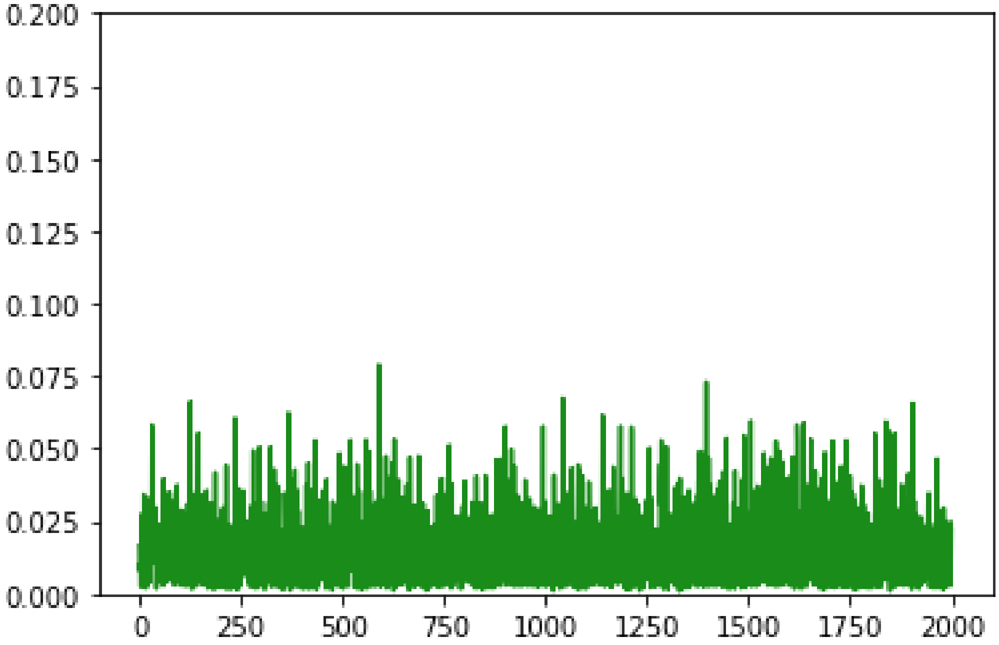
```
0.0105075262996
```

```
In [144]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

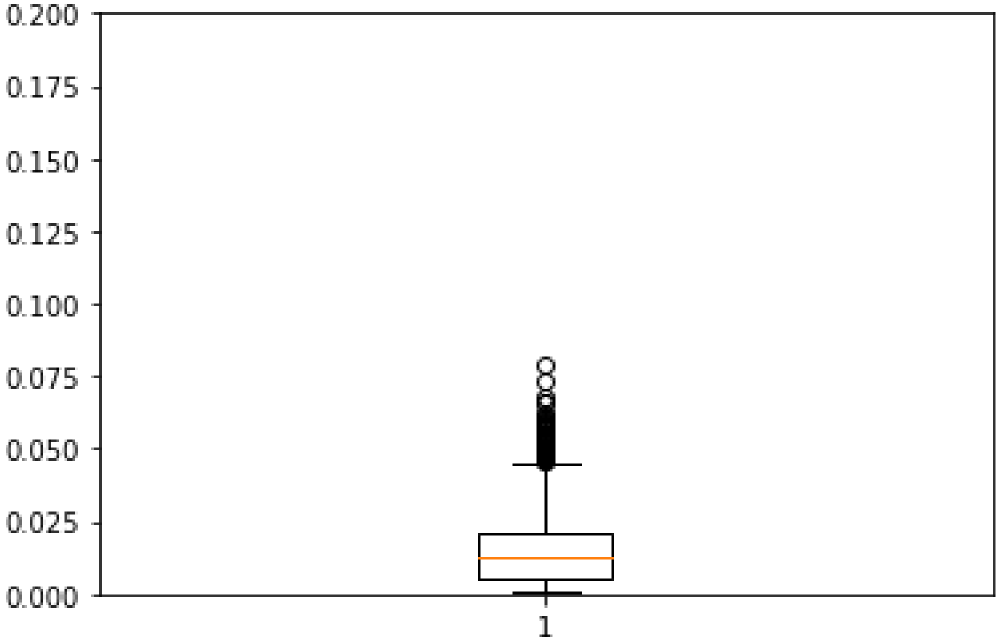


```
0.0185268078819
```





```
0.0152667449199
```



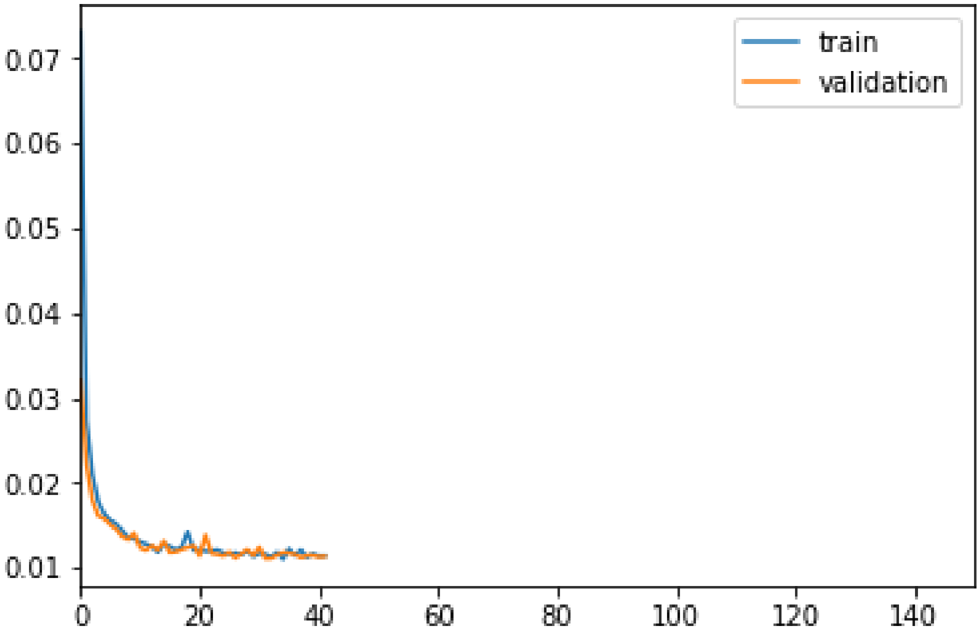
10 steps

```
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)
```

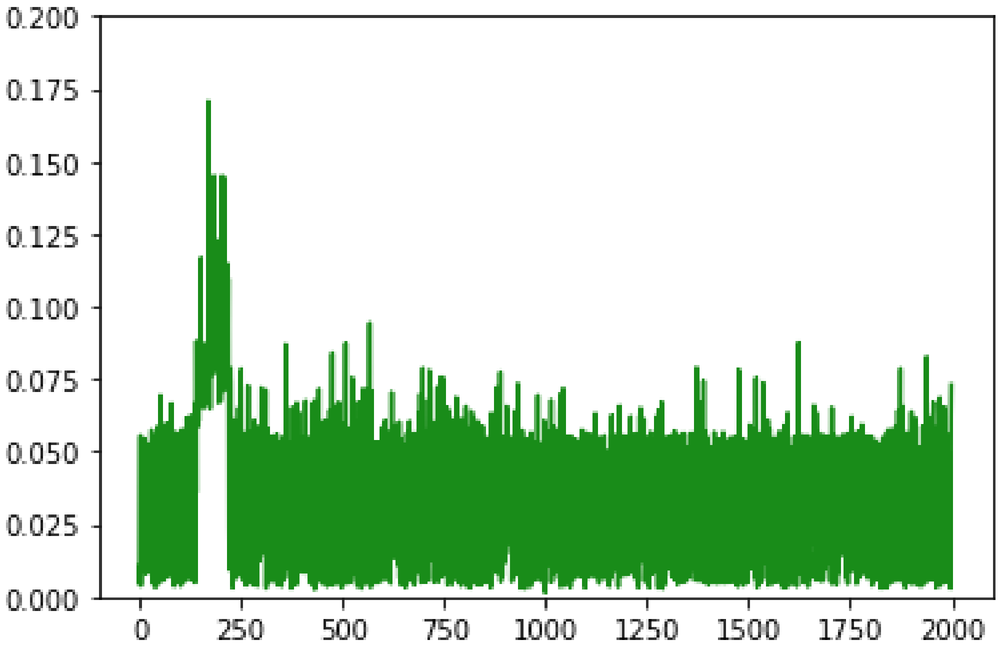
```
In [147]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [148]: train(model, tgen, vgen)
```

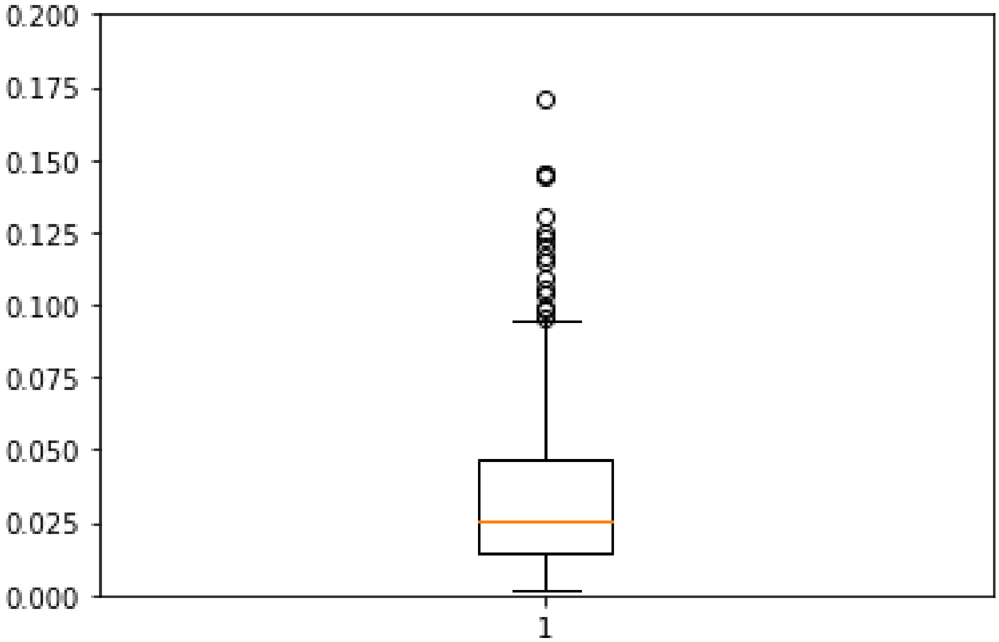


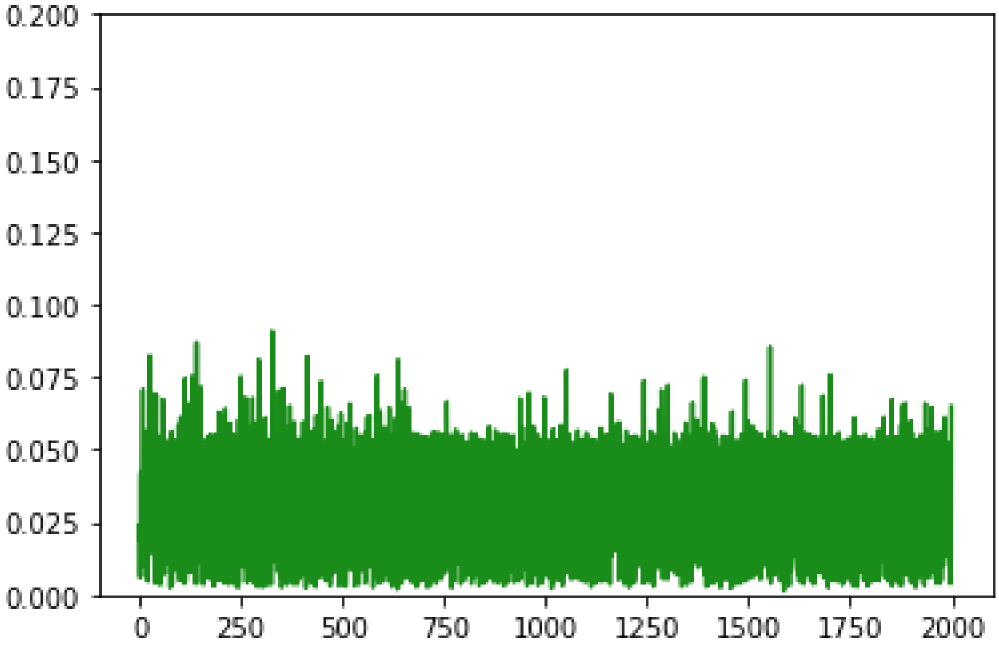
```
0.0114064972028
```

```
In [149]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

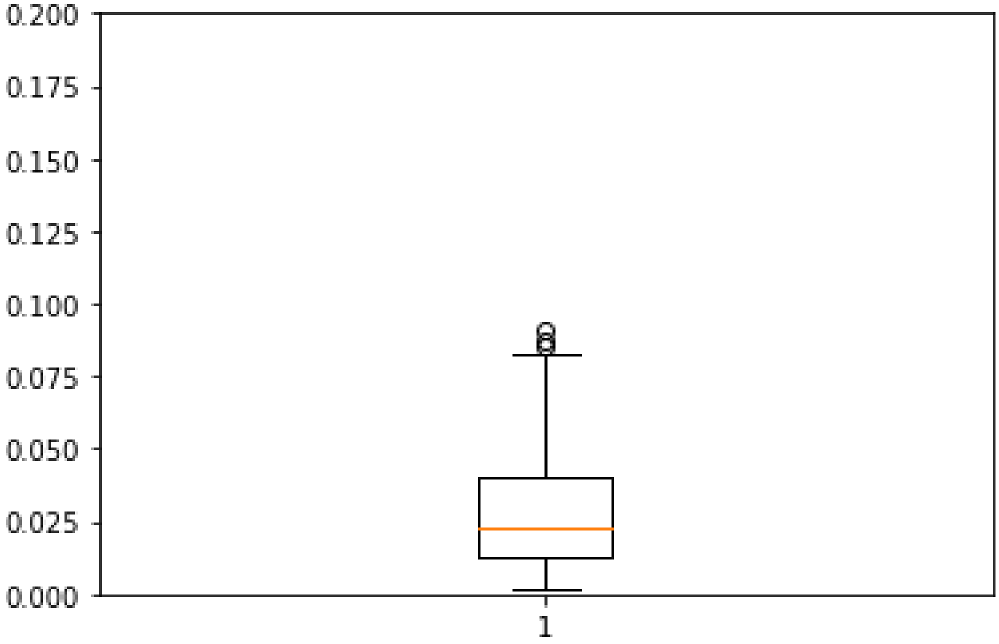


```
0.0310117140412
```





```
0.0272613108639
```



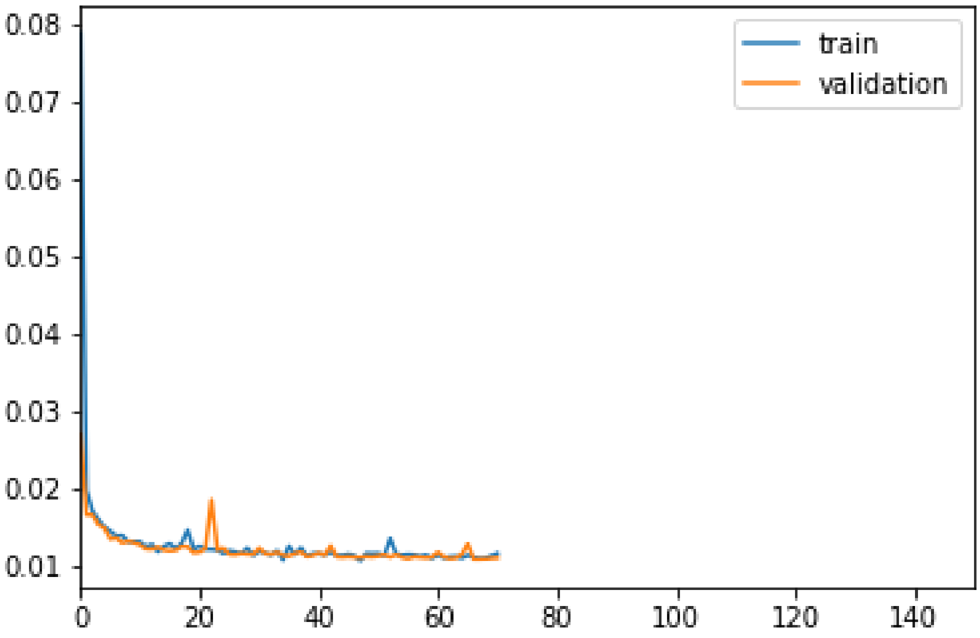
20 steps

```
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)
```

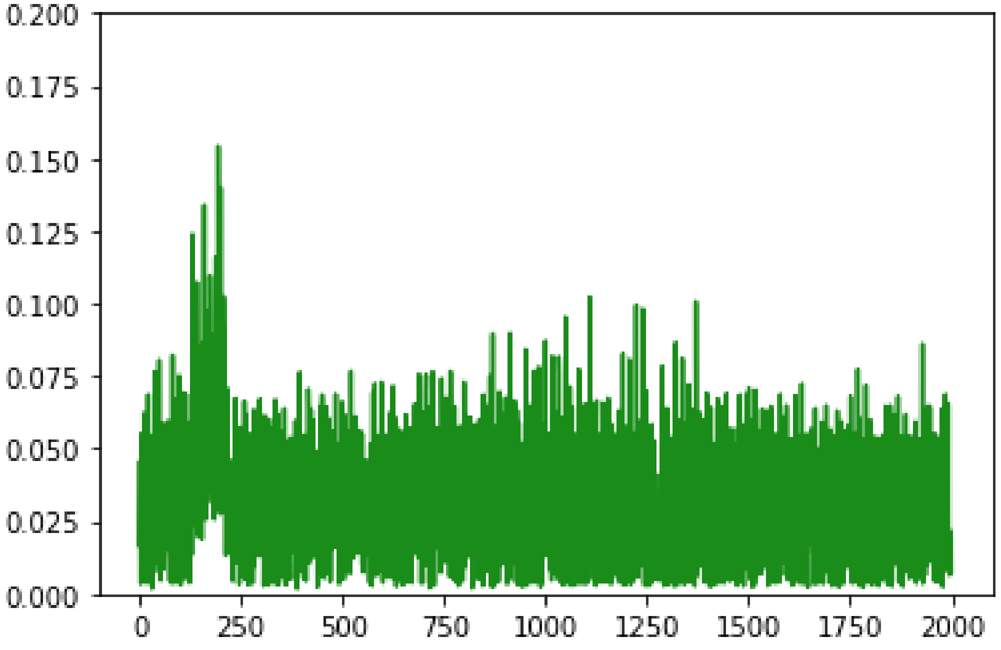
```
In [152]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [153]: train(model, tgen, vgen)
```

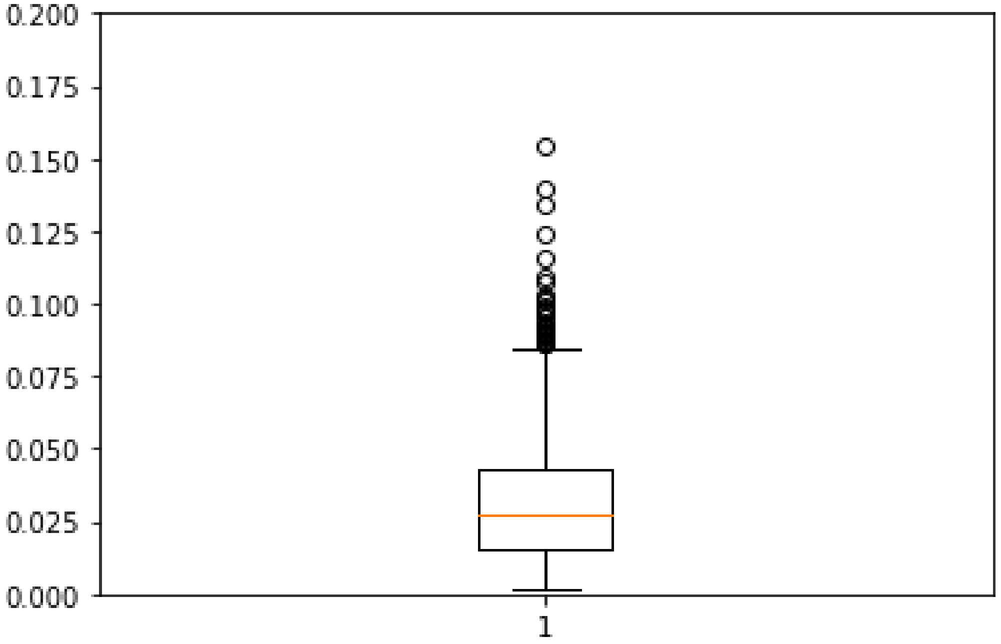


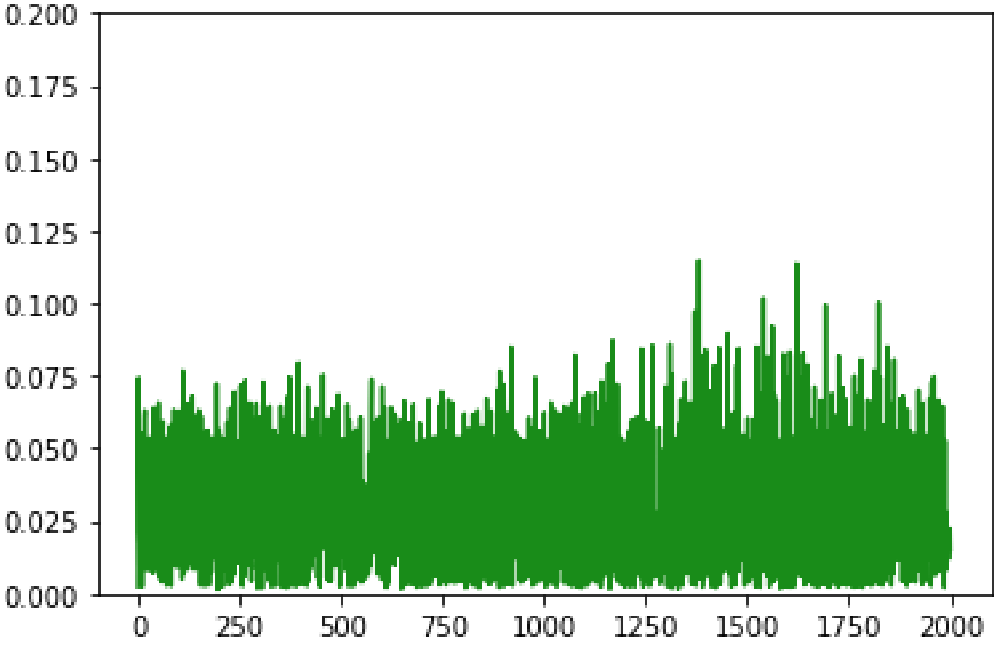
```
0.0115829235336
```

```
In [154]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

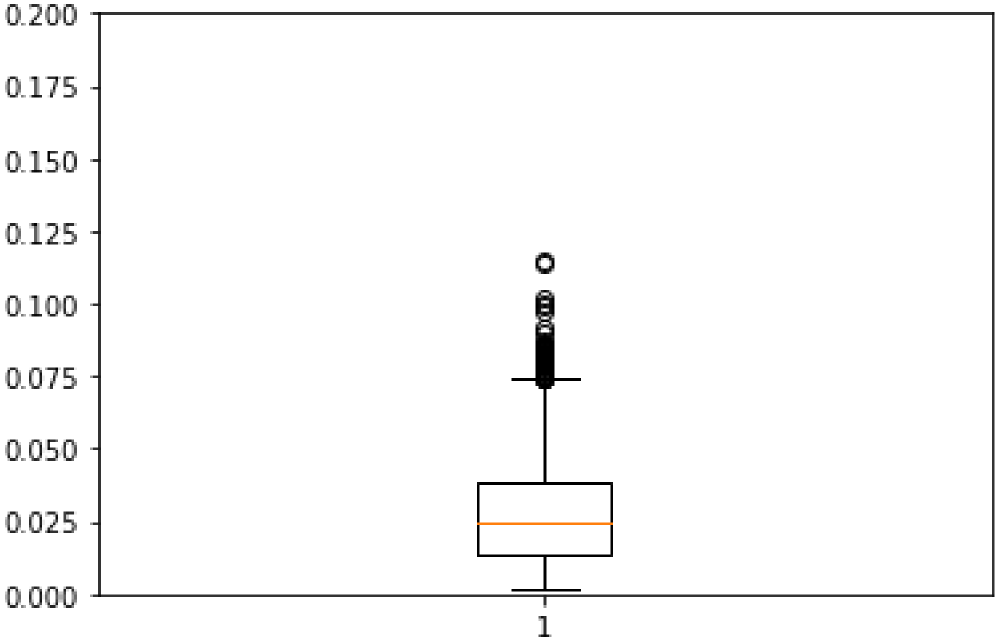


```
0.030898183285
```





```
0.0281960559344
```



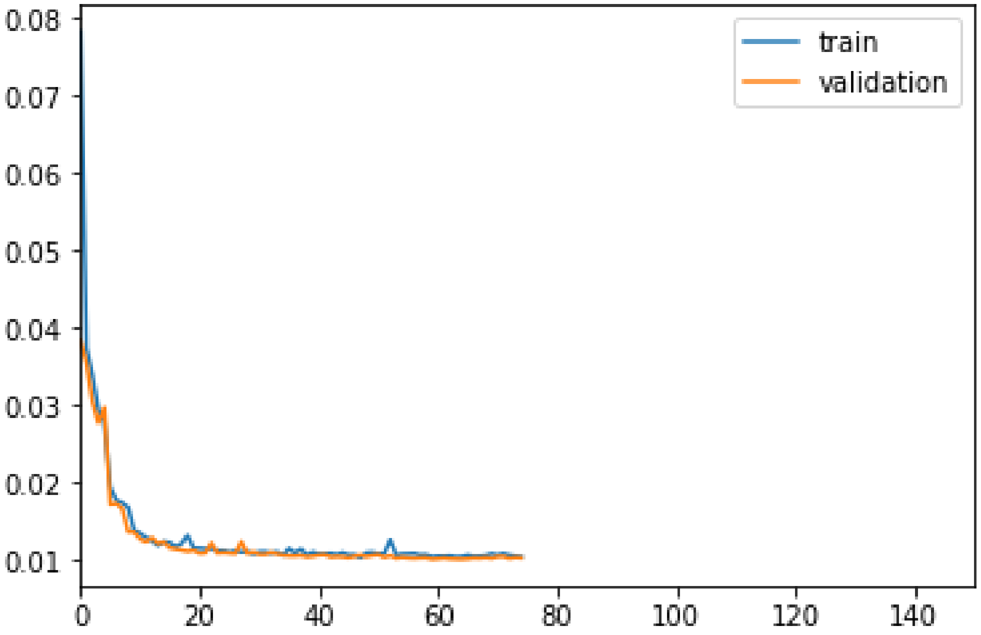
50 steps

```
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)
```

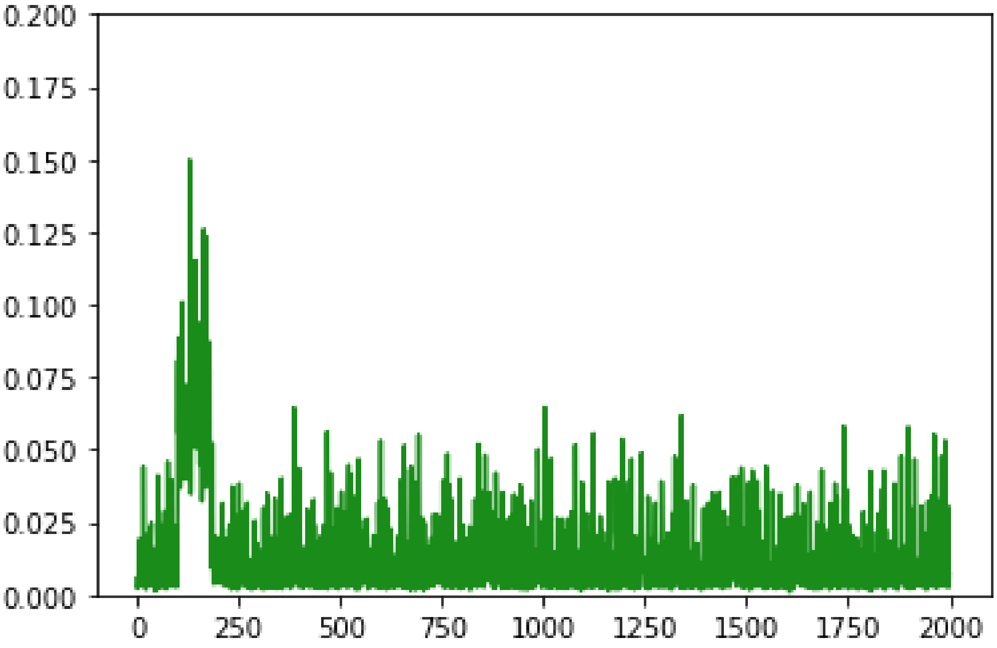
```
In [157]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [158]: train(model, tgen, vgen)
```

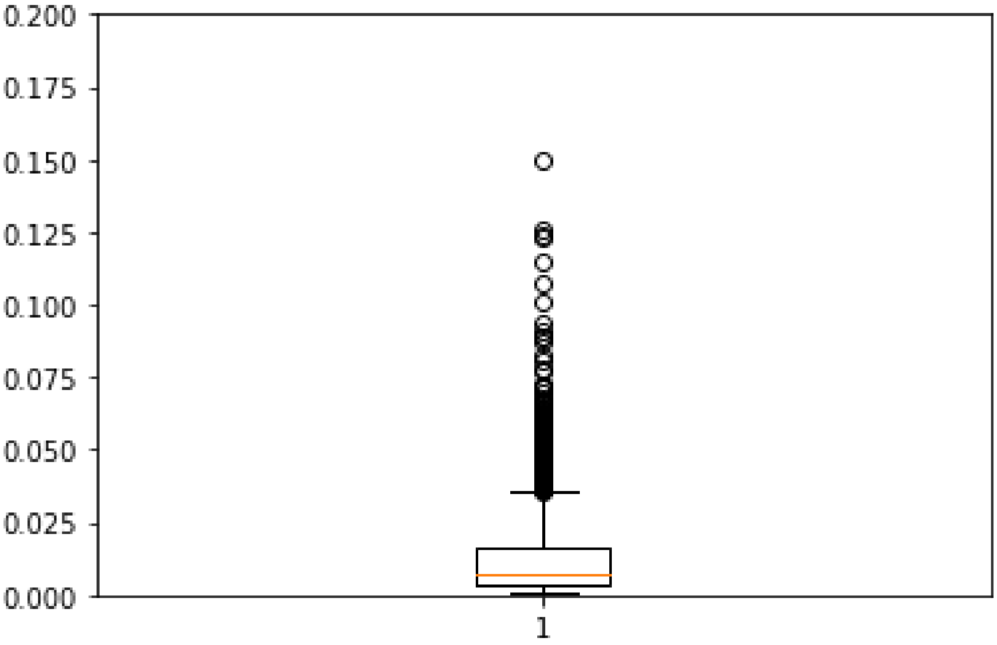


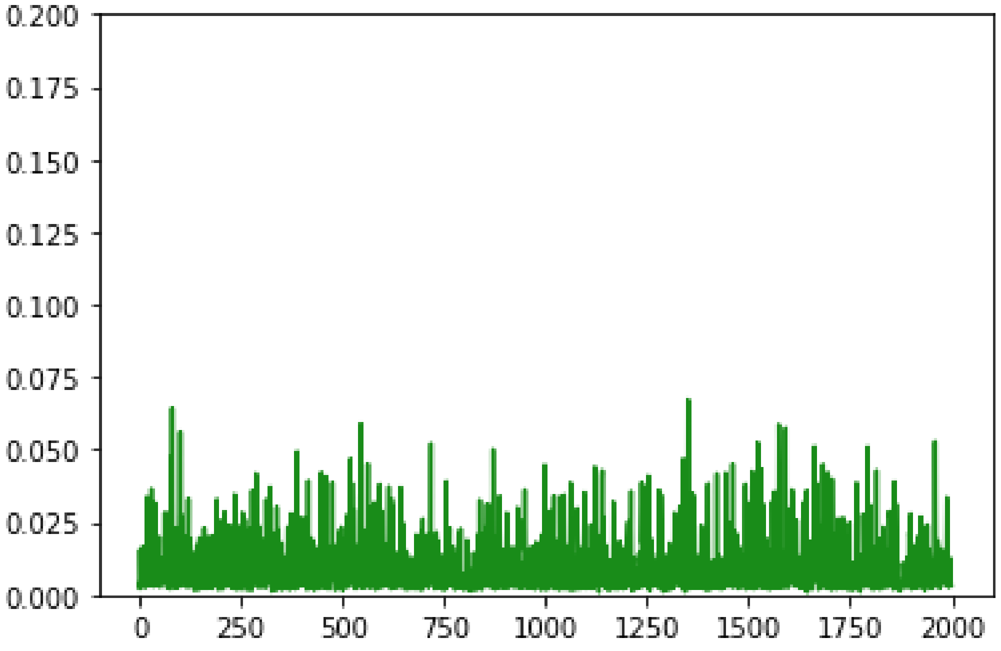
```
0.0103810858442
```

```
In [159]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

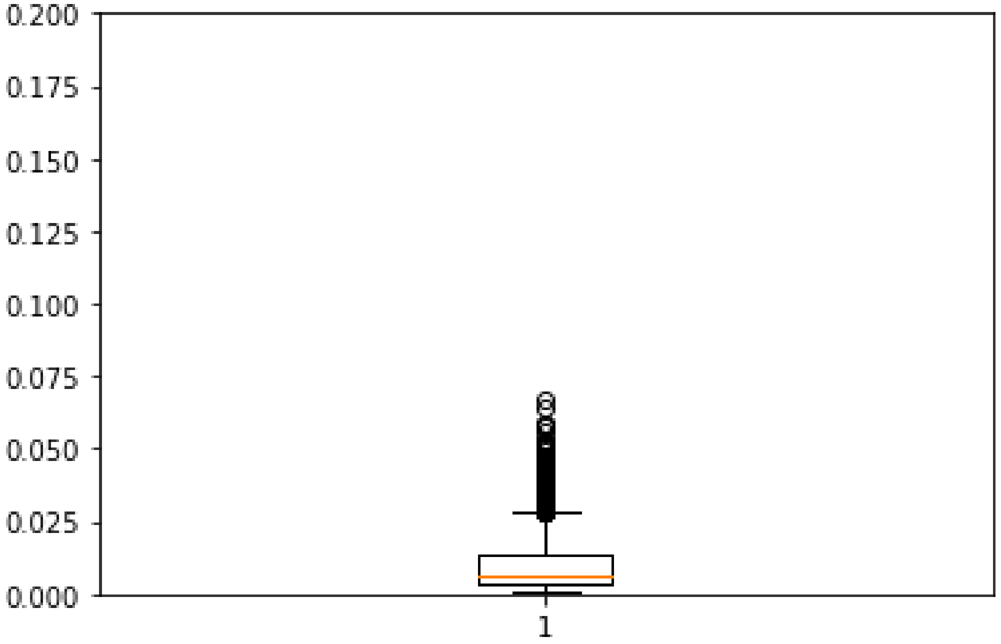


```
0.0131122055379
```





```
0.00980400729242
```



2.1.6 RNN with 2 GRU layers

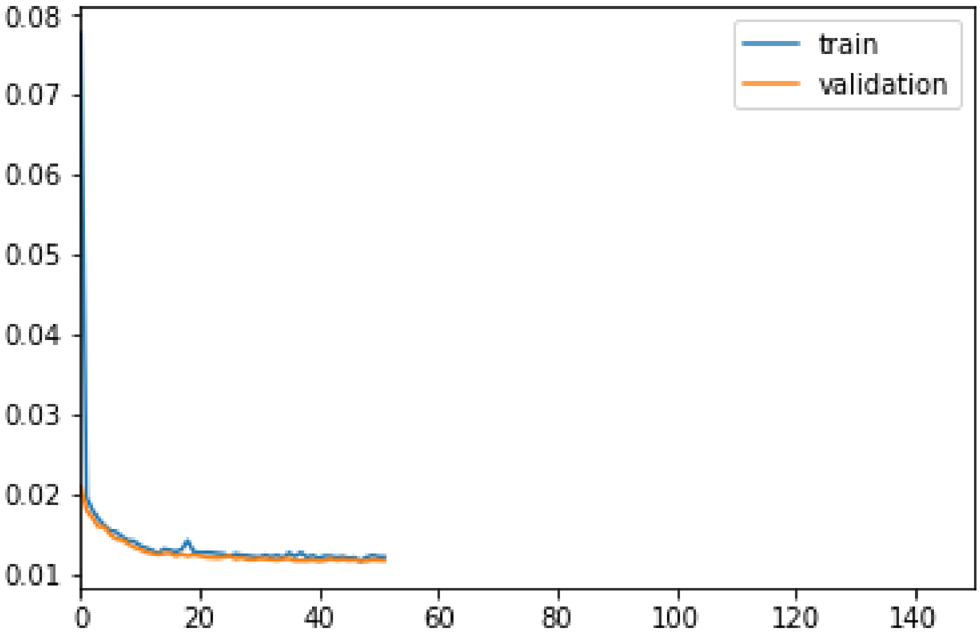
2 steps

```
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)
```

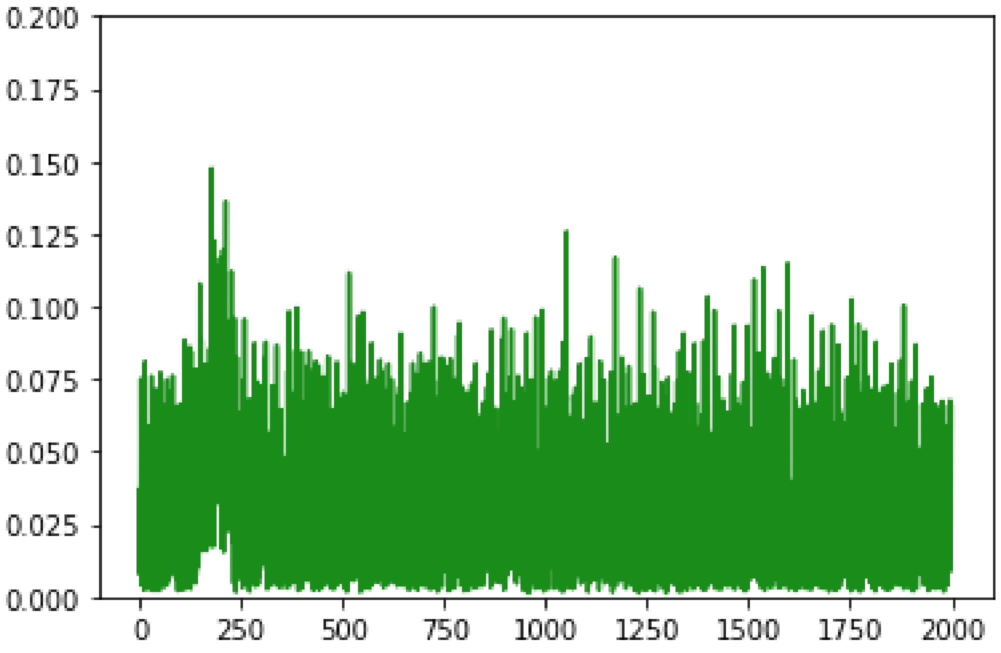
```
In [162]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [163]: train(model, tgen, vgen)
```

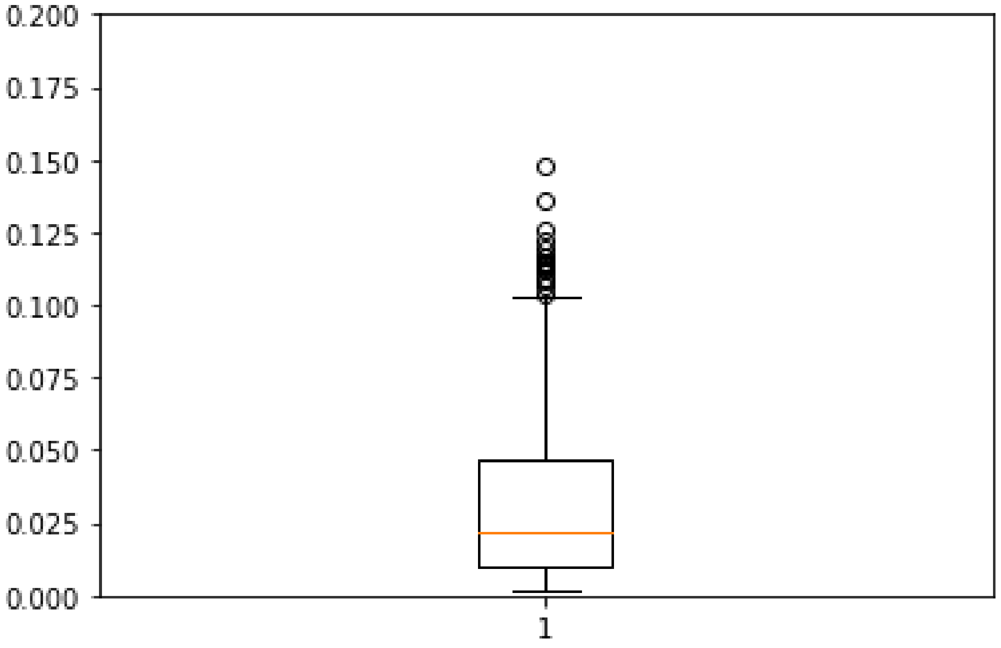


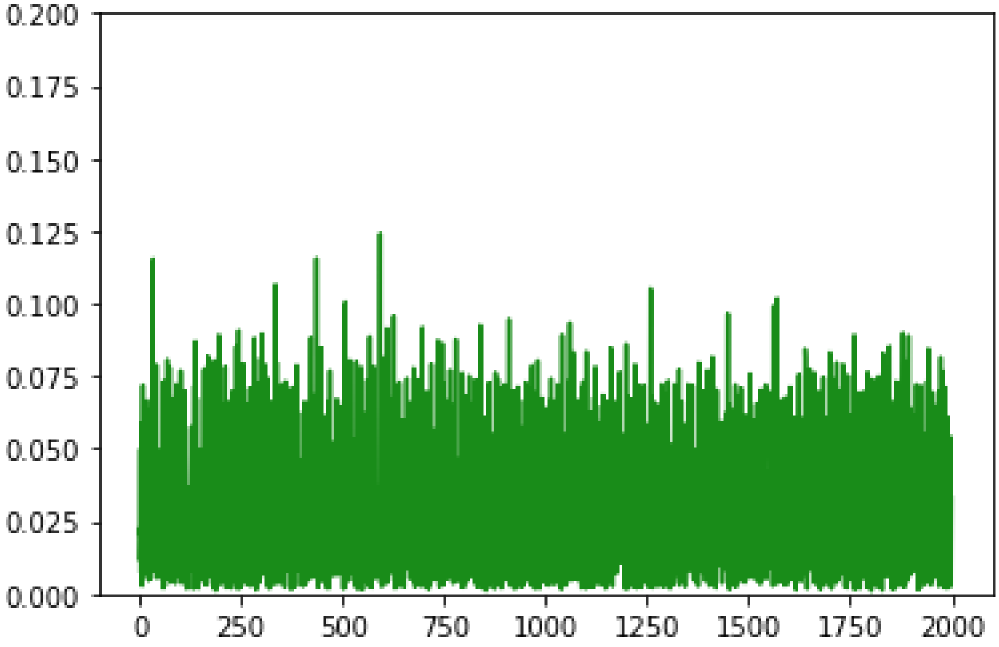
```
0.0120978509646
```

```
In [164]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

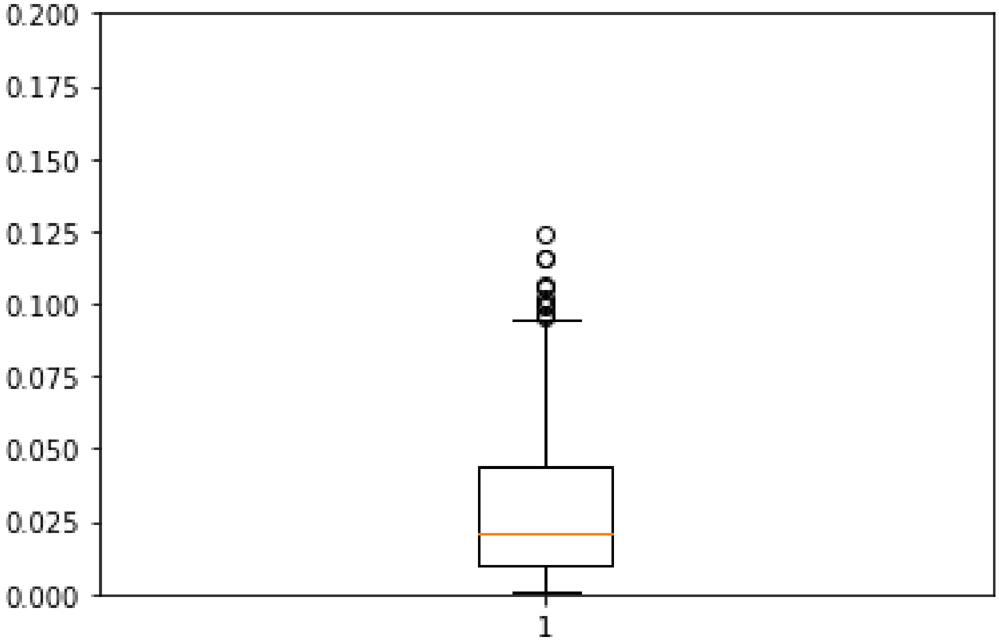


```
0.0306171823363
```





```
0.0285666419295
```



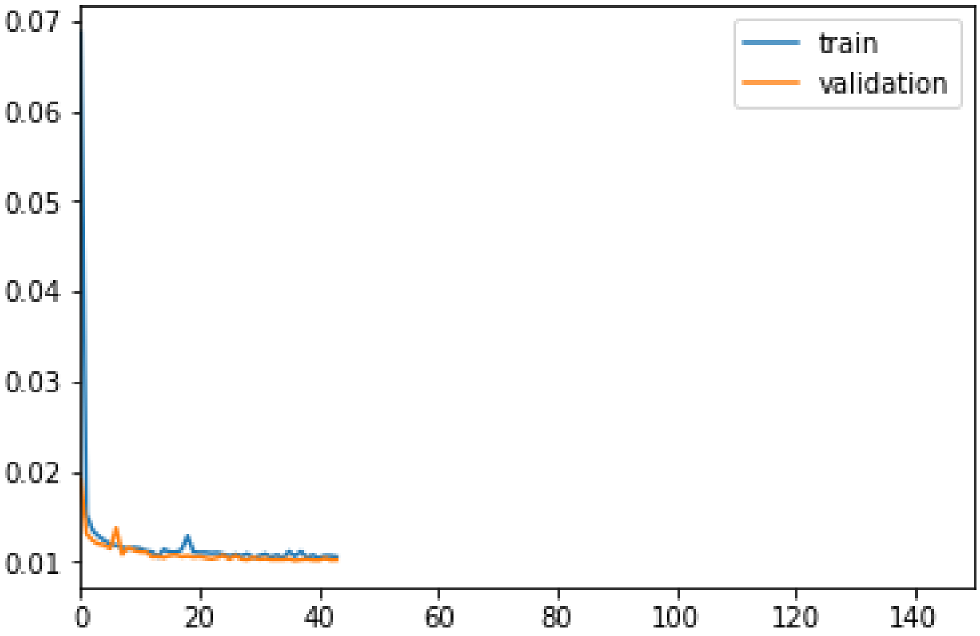
5 steps

```
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)
```

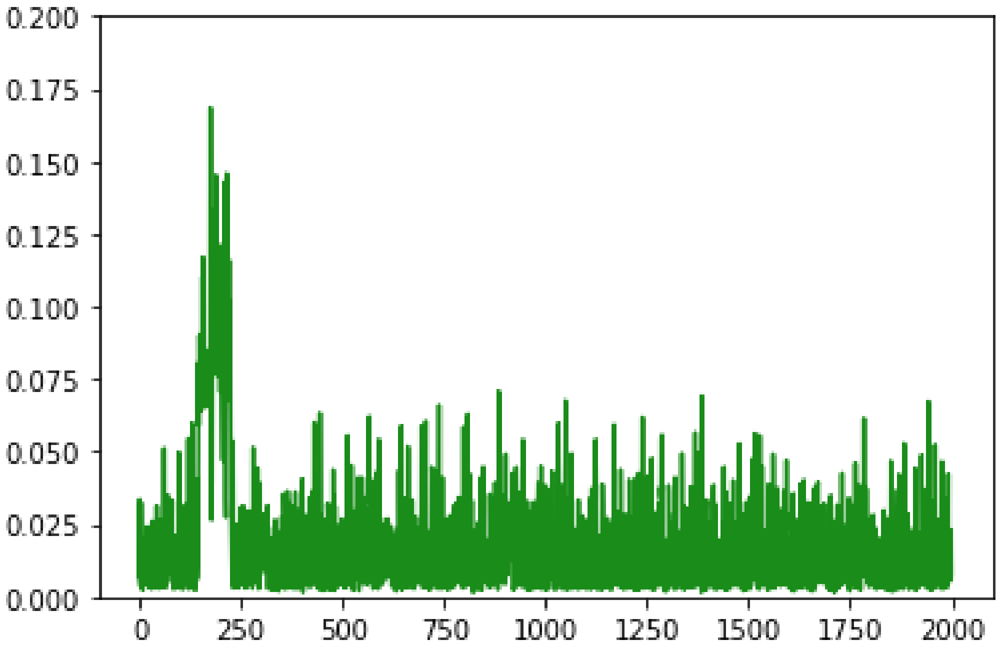
```
In [167]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [168]: train(model, tgen, vgen)
```

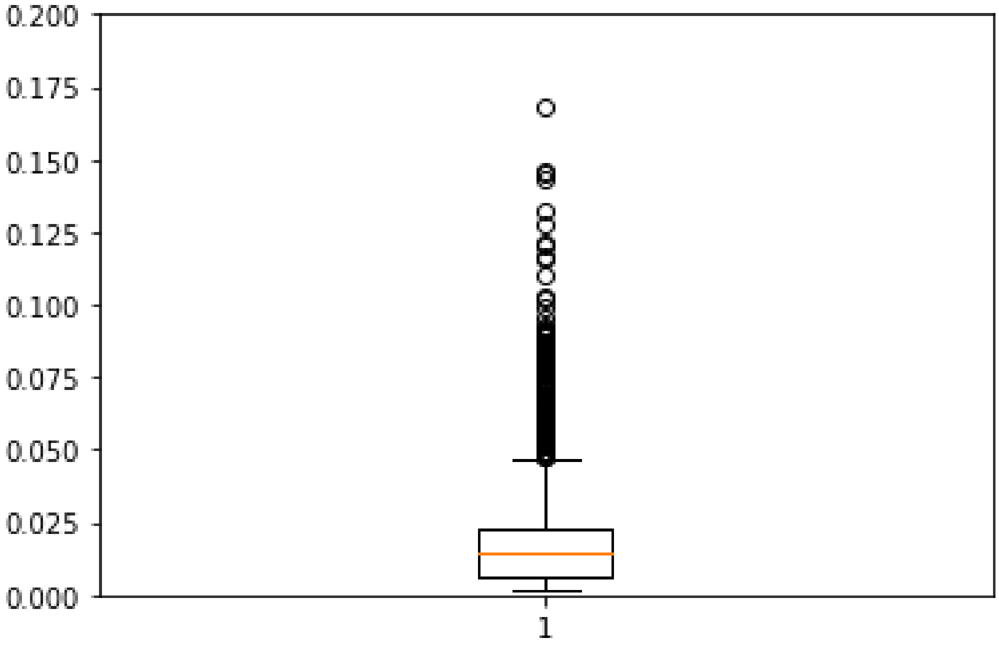


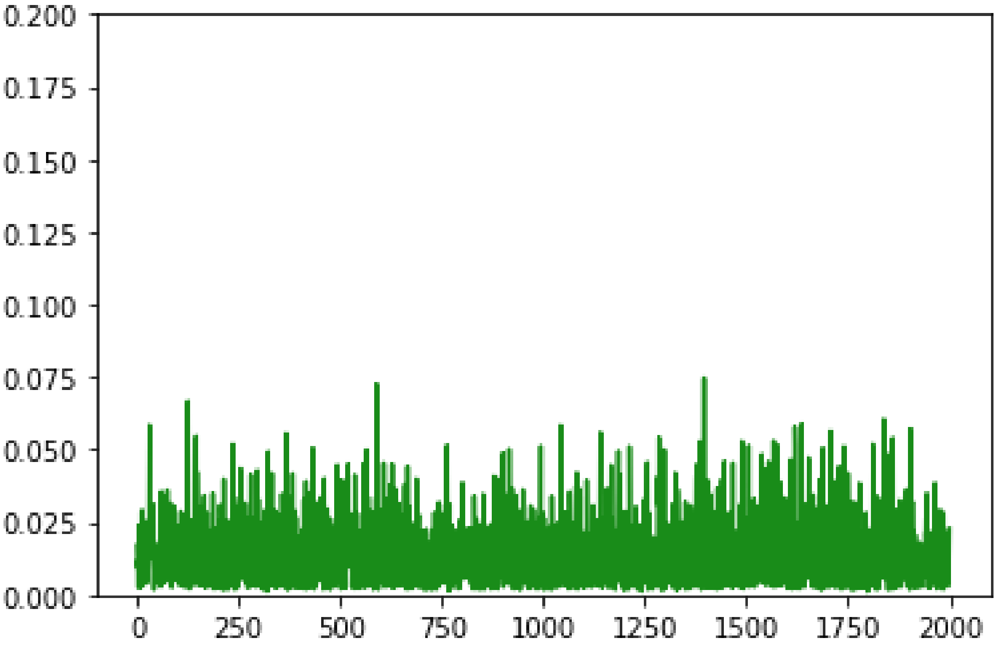
```
0.0105663195567
```

```
In [169]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

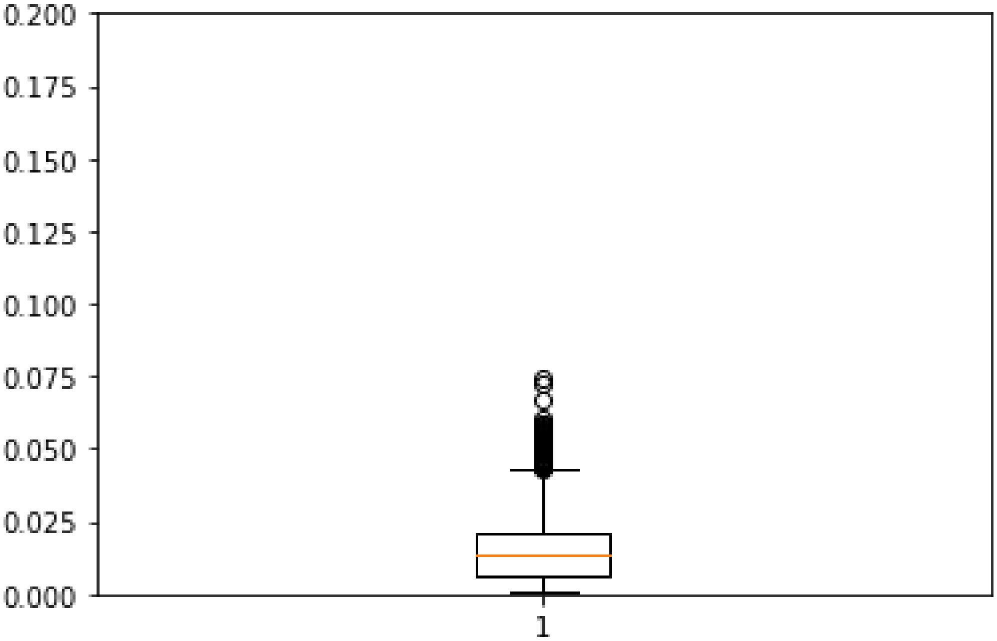


```
0.018543315887
```





```
0.0151939156219
```



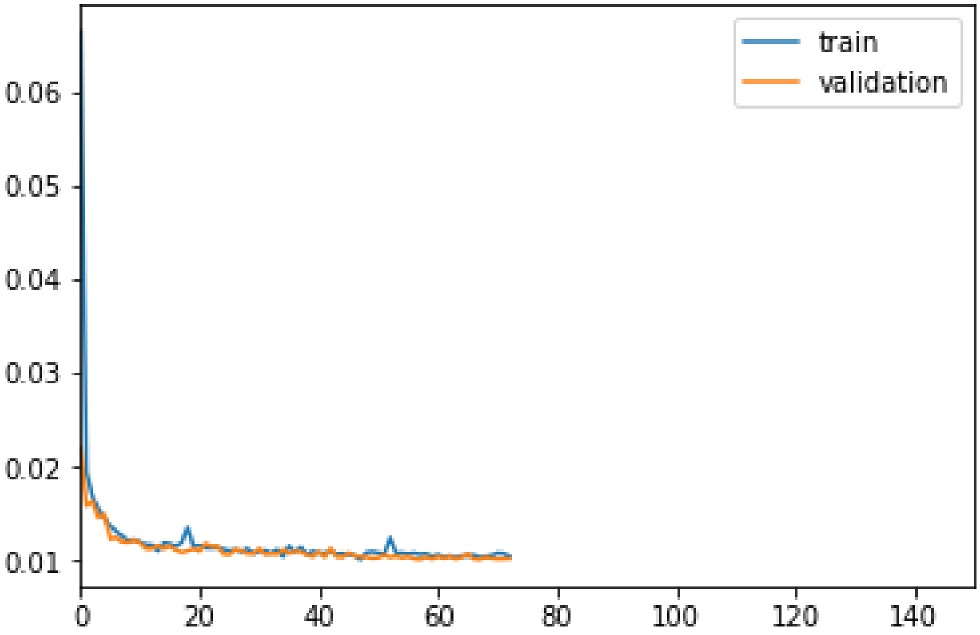
10 steps

```
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)
```

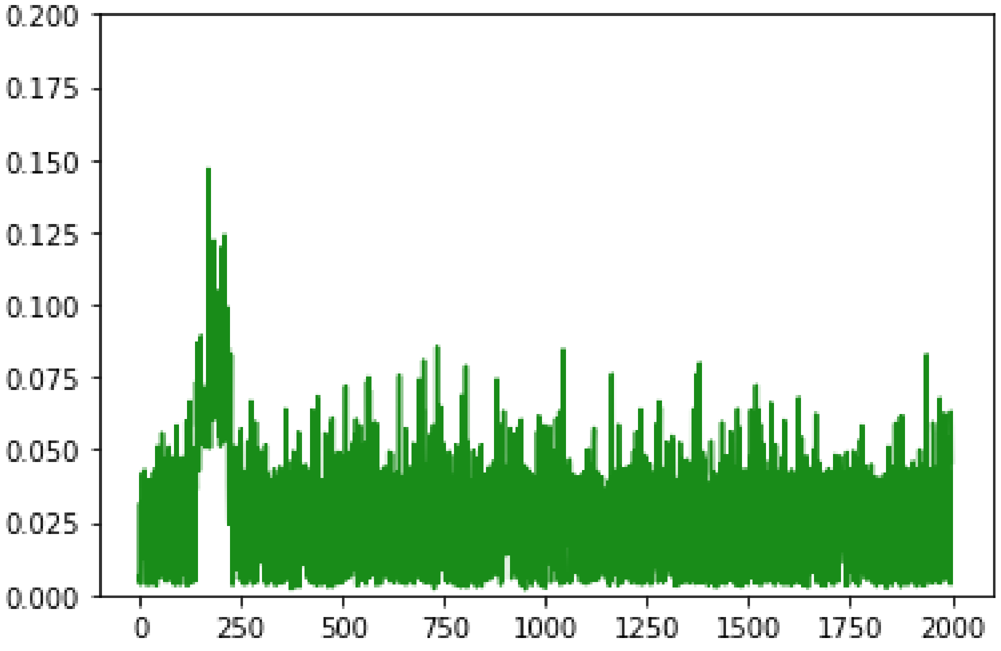
```
In [172]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [173]: train(model, tgen, vgen)
```

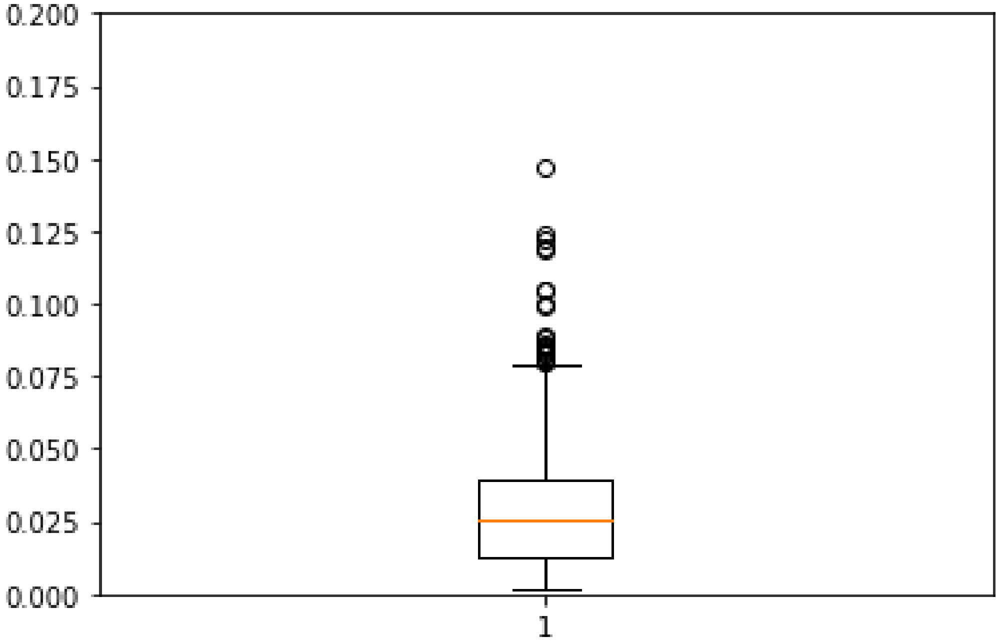


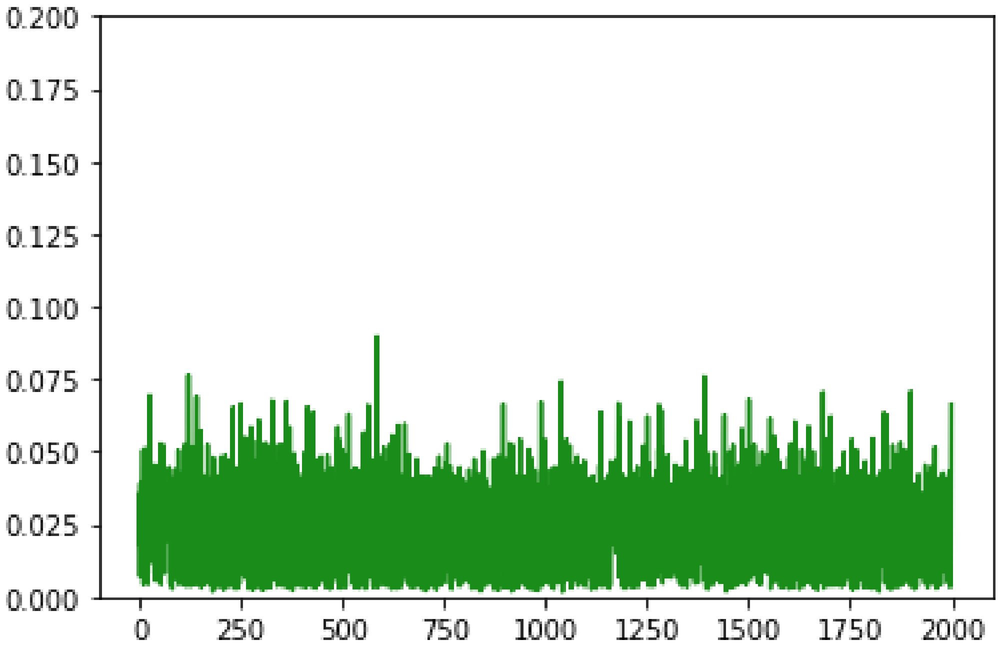
```
0.0104816673028
```

```
In [174]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

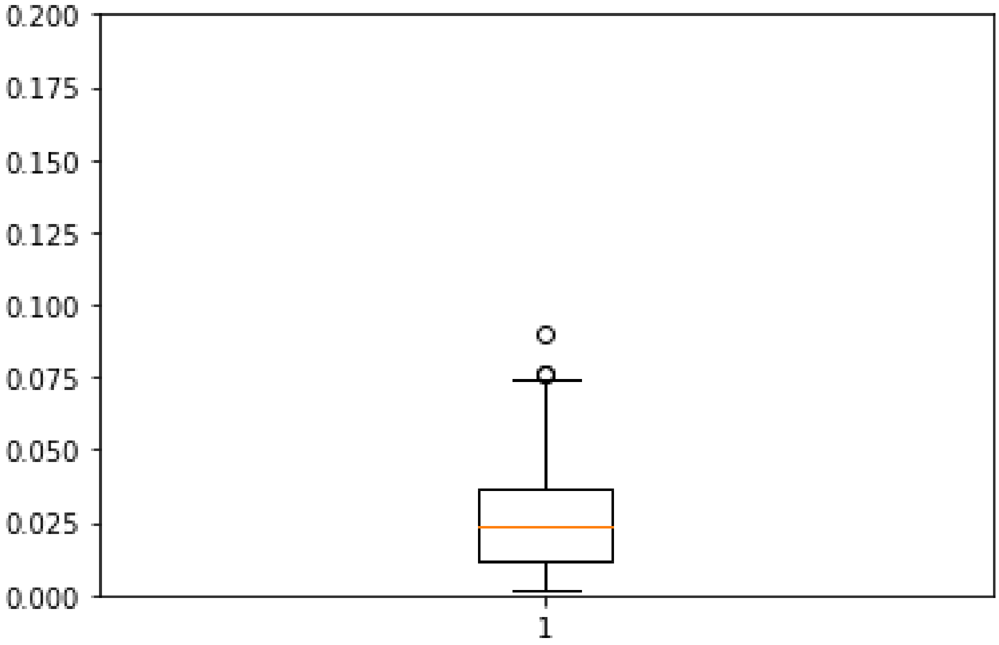


```
0.0277265963166
```





```
0.0247659988424
```



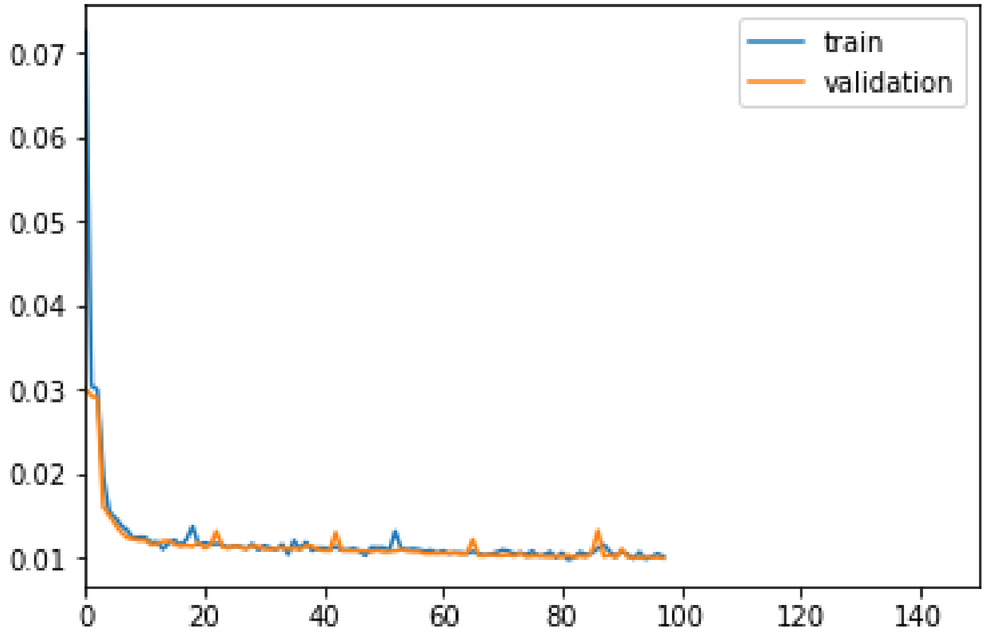
20 steps

```
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)
```

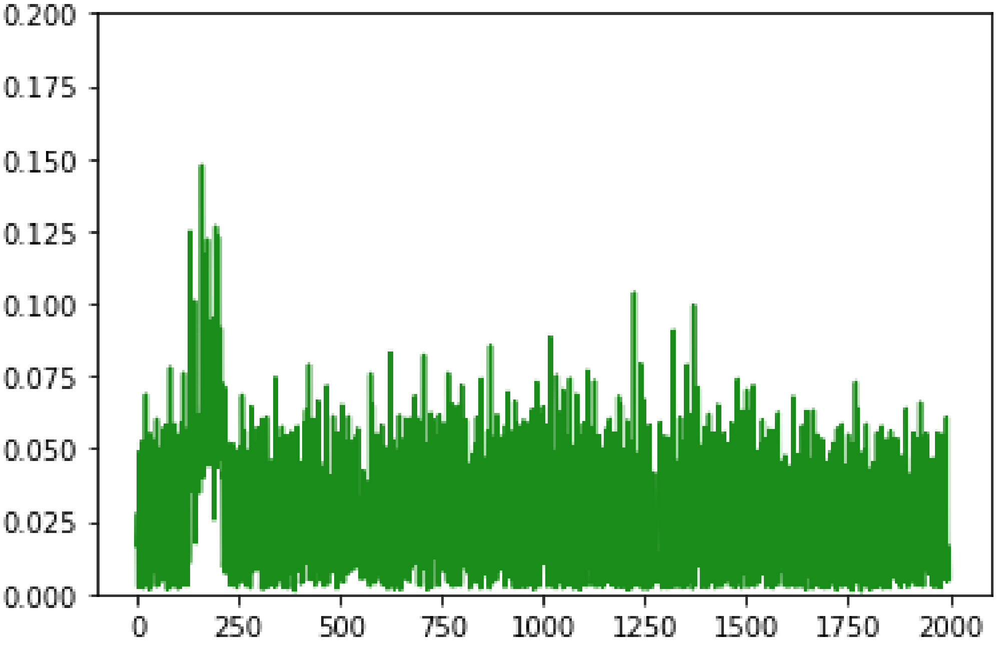
```
In [177]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [178]: train(model, tgen, vgen)
```

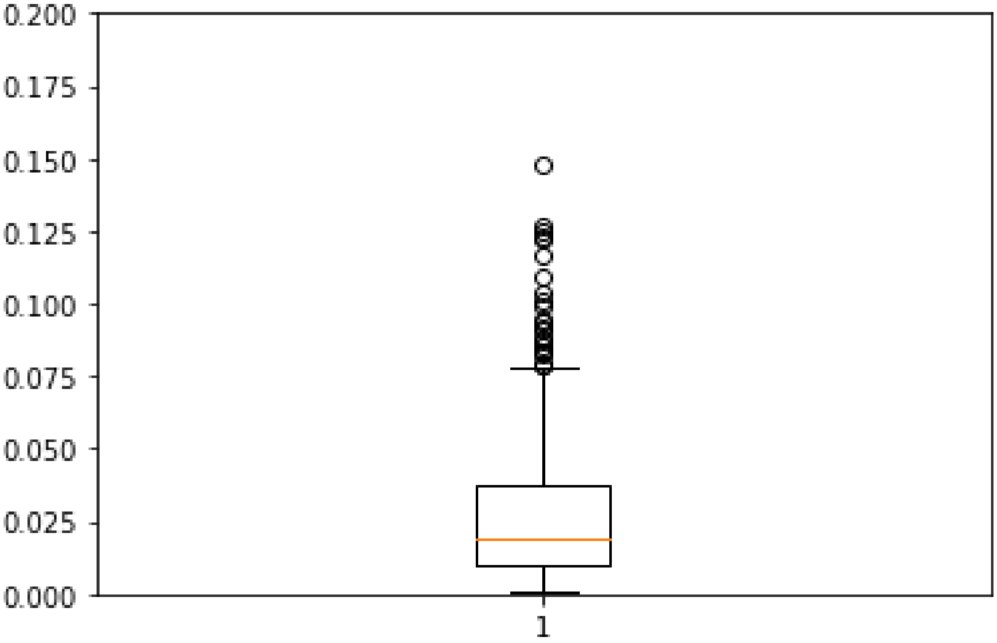


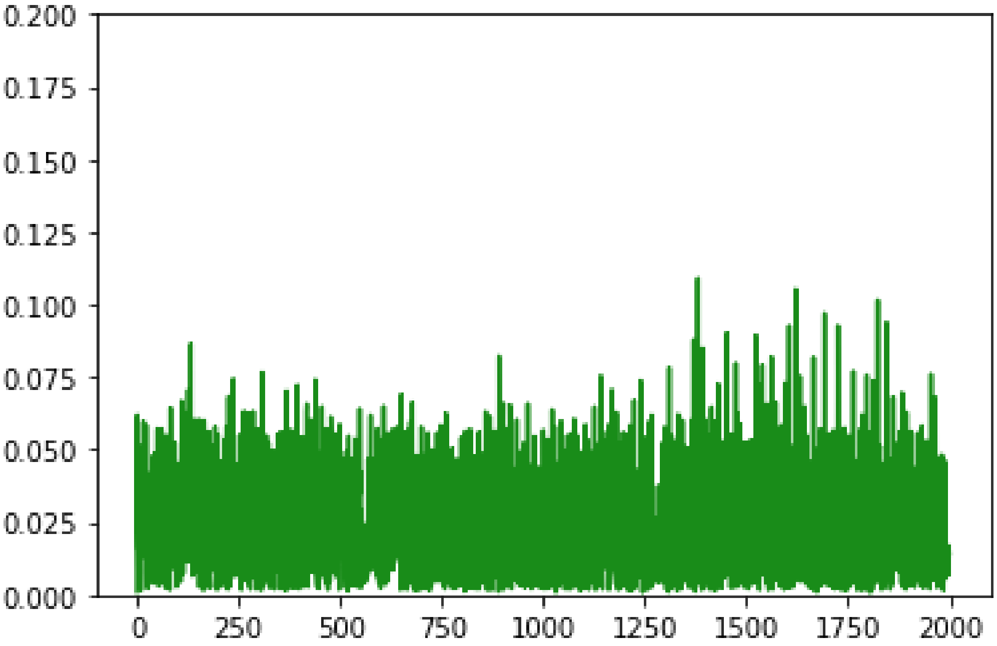
```
0.010202657062
```

```
In [179]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

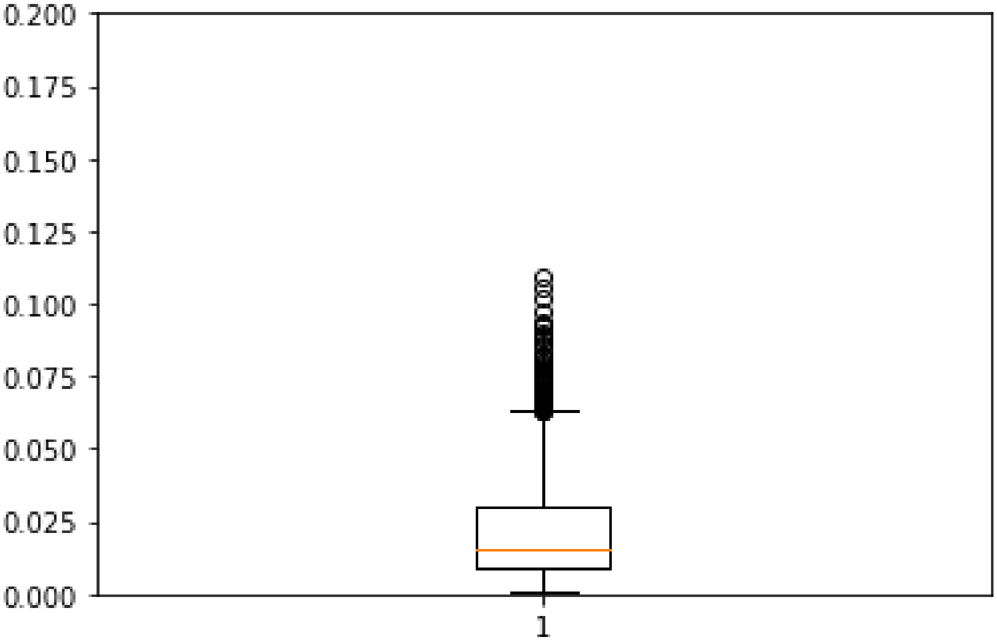


```
0.024973029488
```





```
0.0219135989671
```



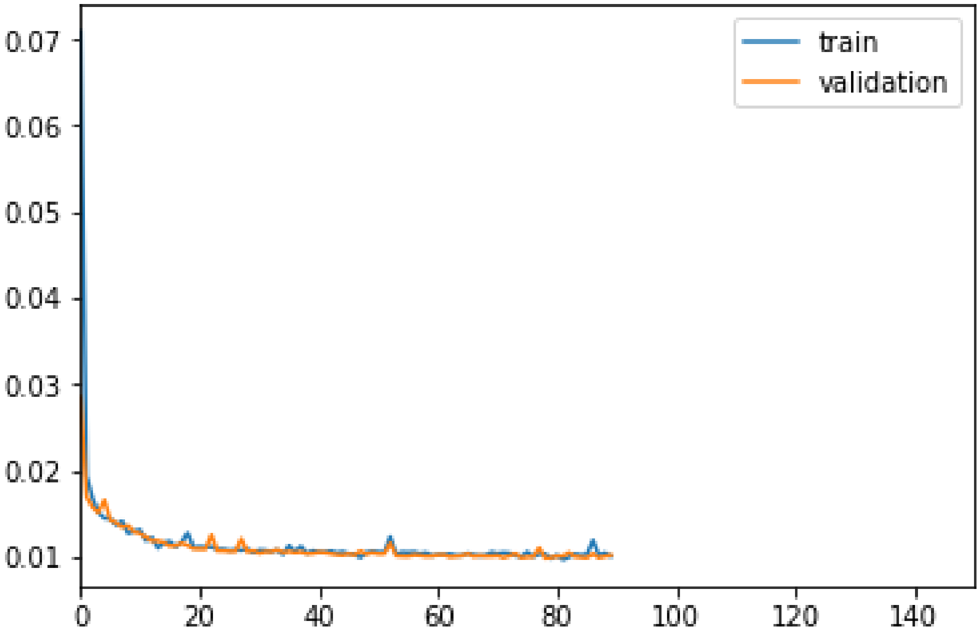
50 steps

```
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)
```

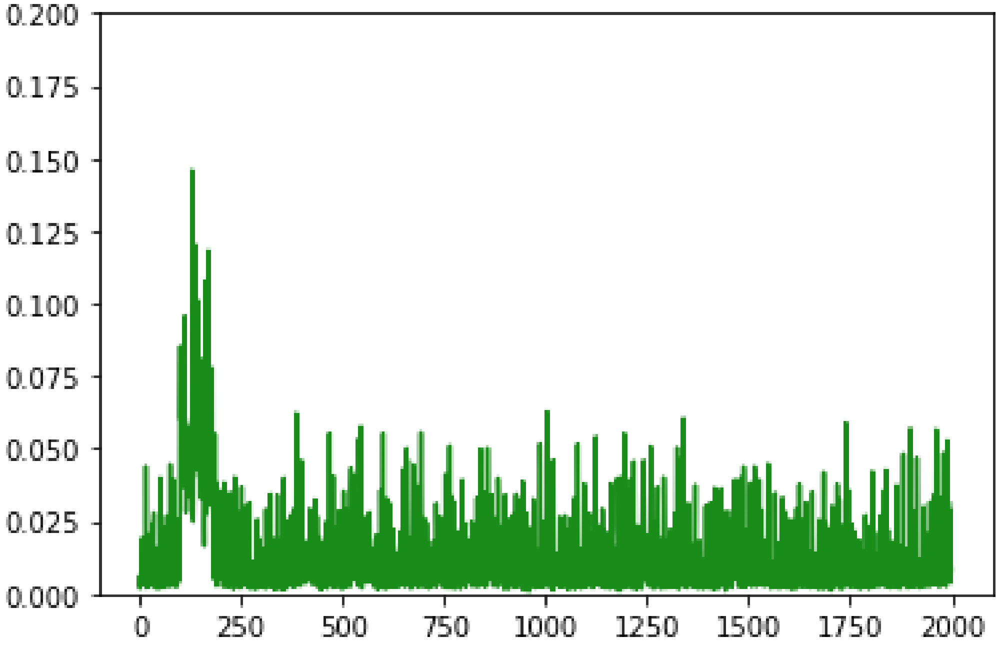
```
In [182]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [183]: train(model, tgen, vgen)
```

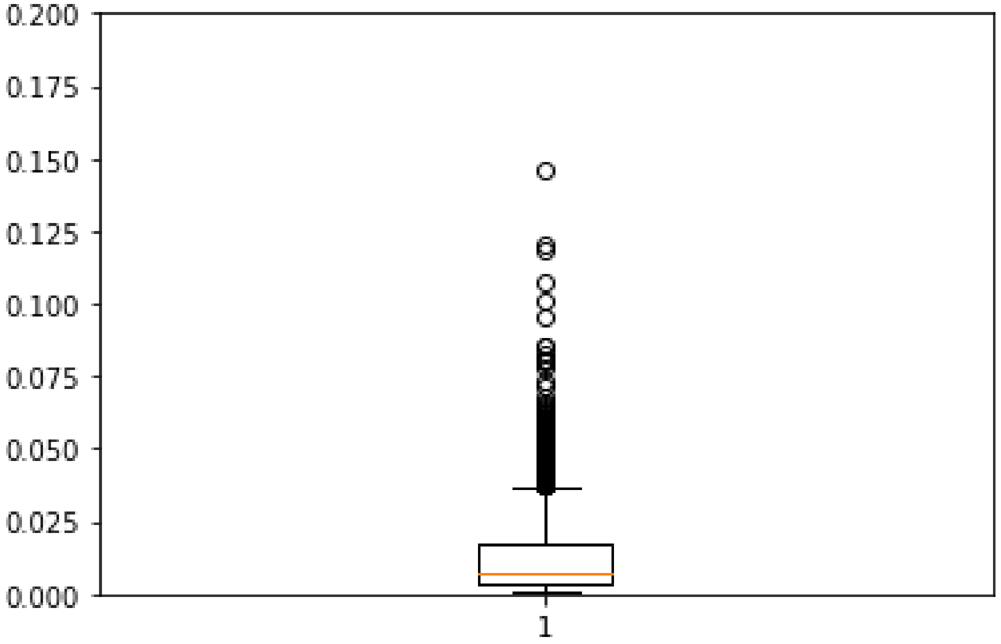


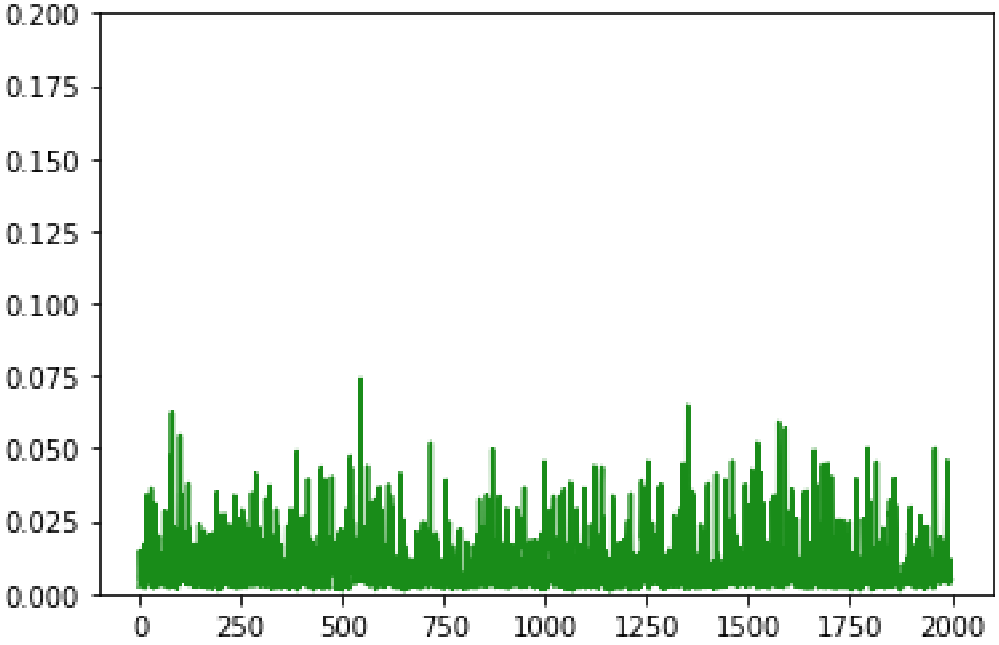
```
0.0101162583093
```

```
In [184]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

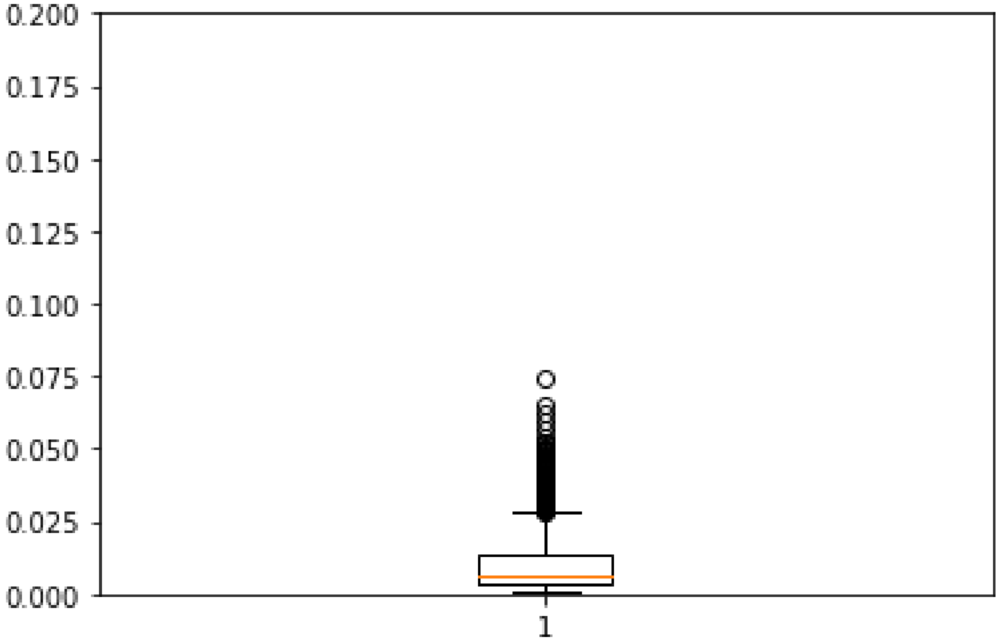


```
0.0130328587091
```





```
0.00978168778487
```



2.1.7 RNN with 3 GRU layers

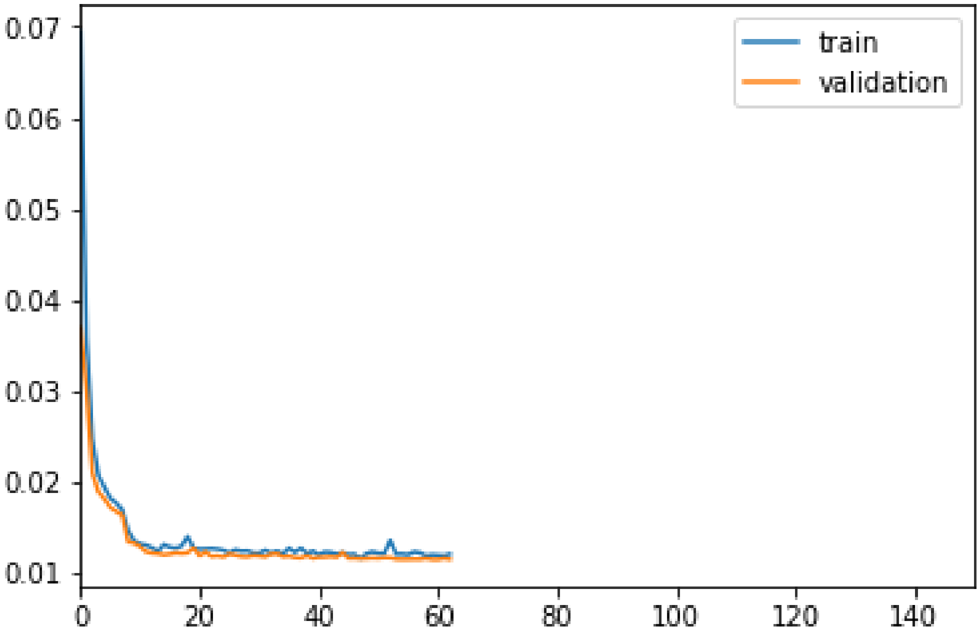
2 steps

```
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)
```

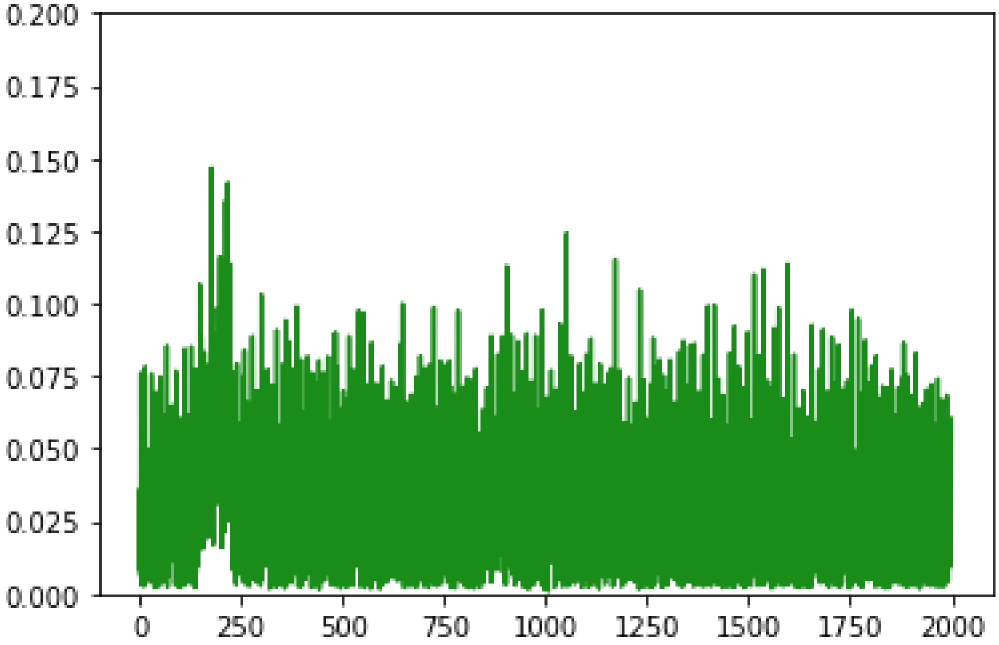
```
In [187]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [188]: train(model, tgen, vgen)
```

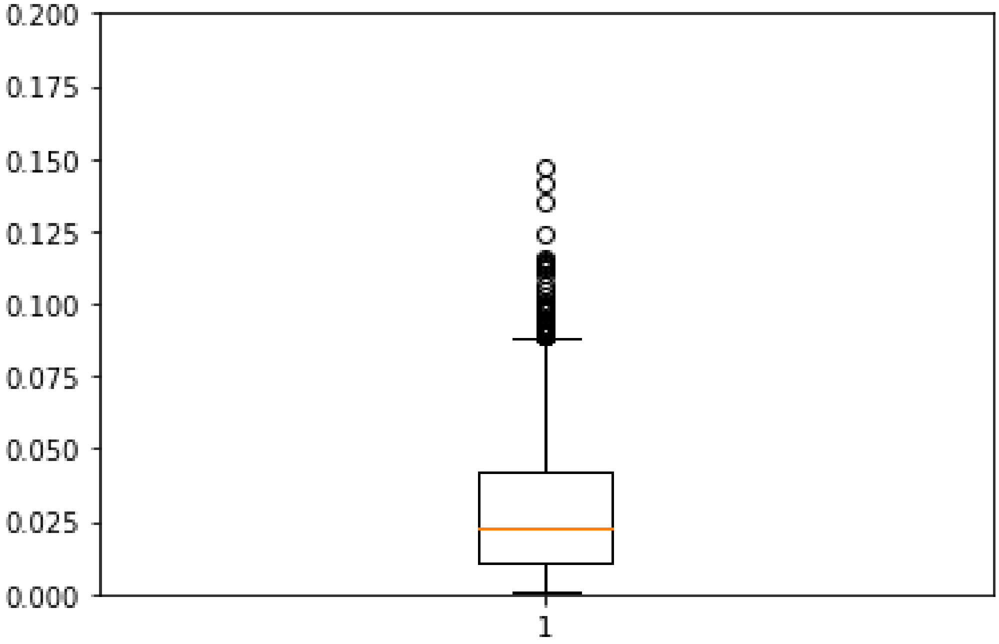


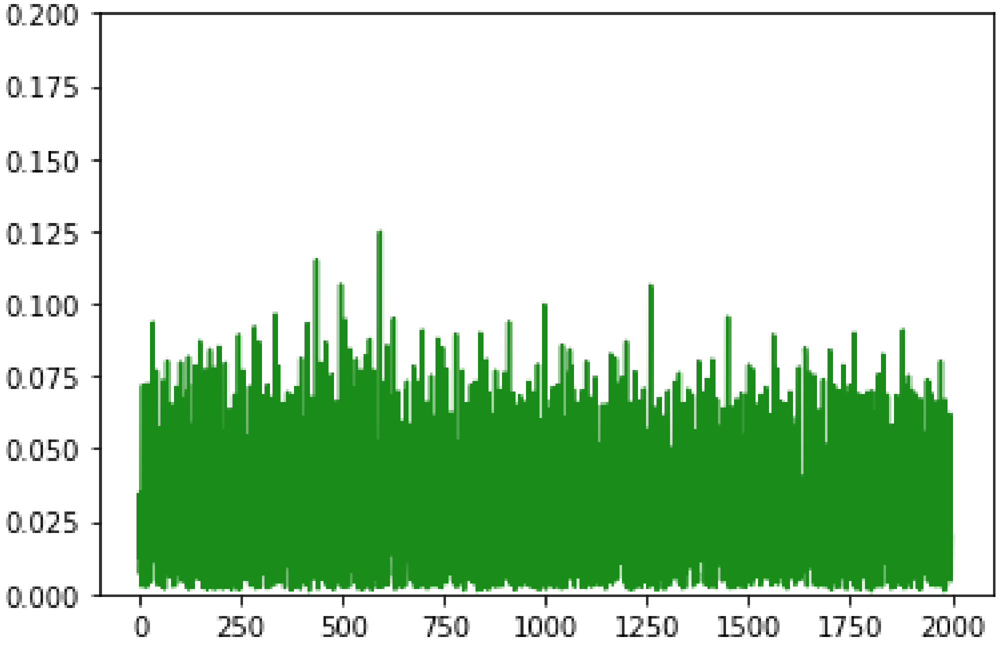
```
0.0120234720293
```

```
In [189]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

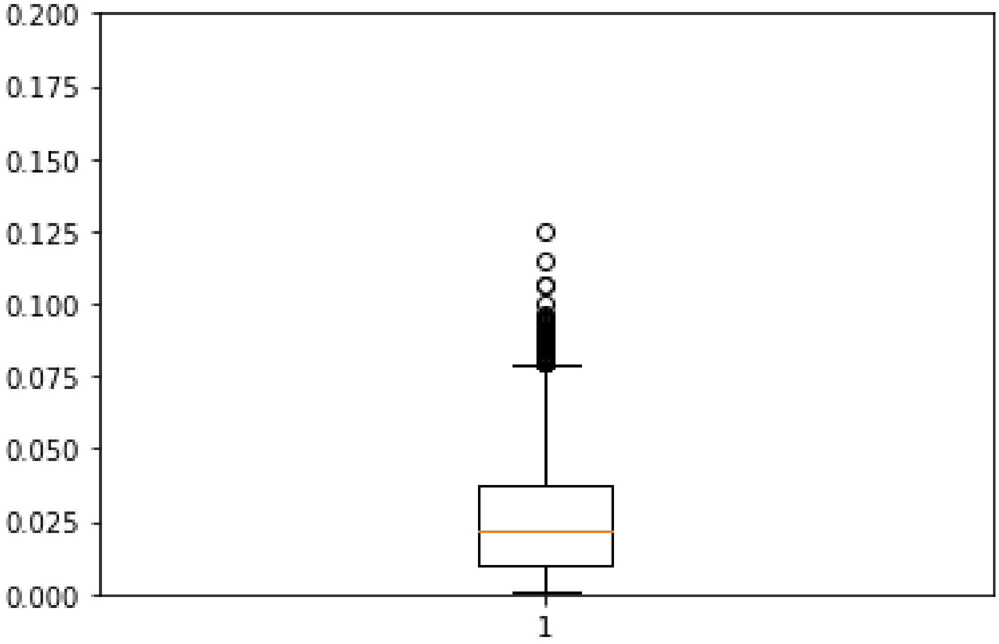


```
0.0295772995954
```





```
0.0270310814877
```



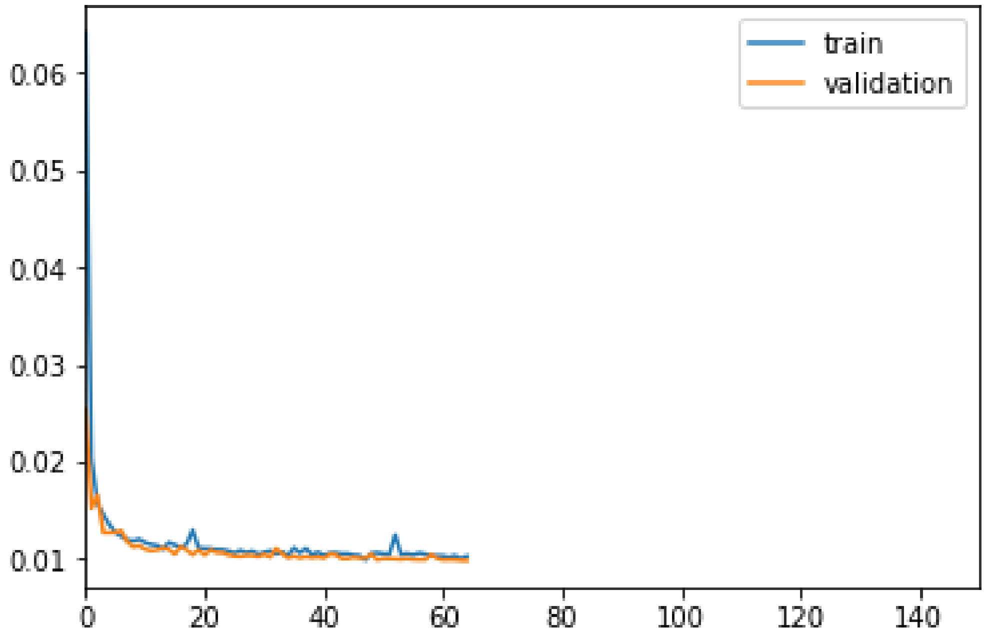
5 steps

```
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)
```

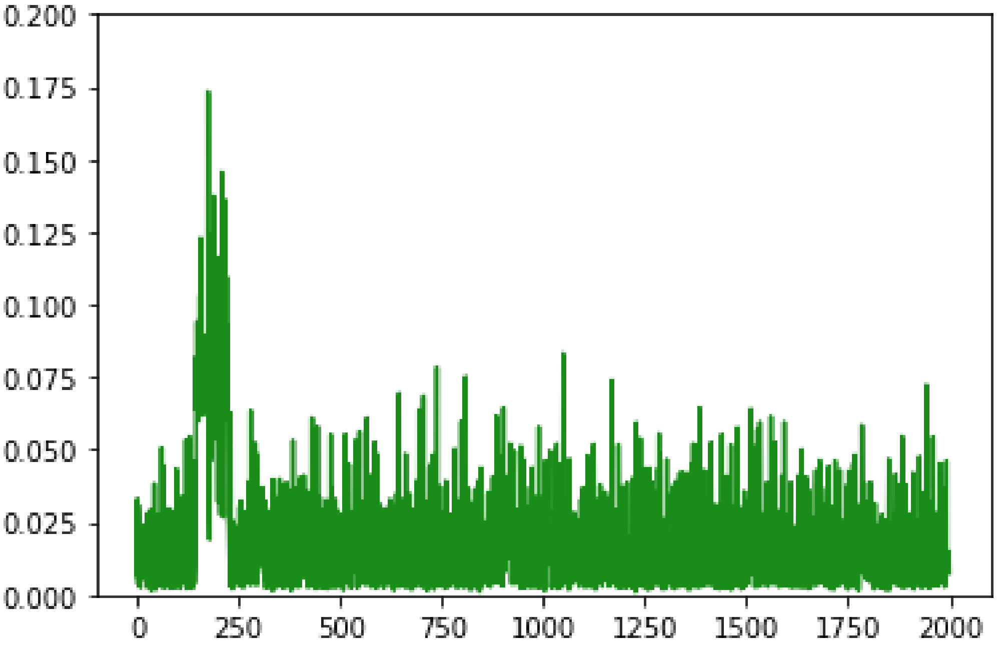
```
In [192]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [193]: train(model, tgen, vgen)
```

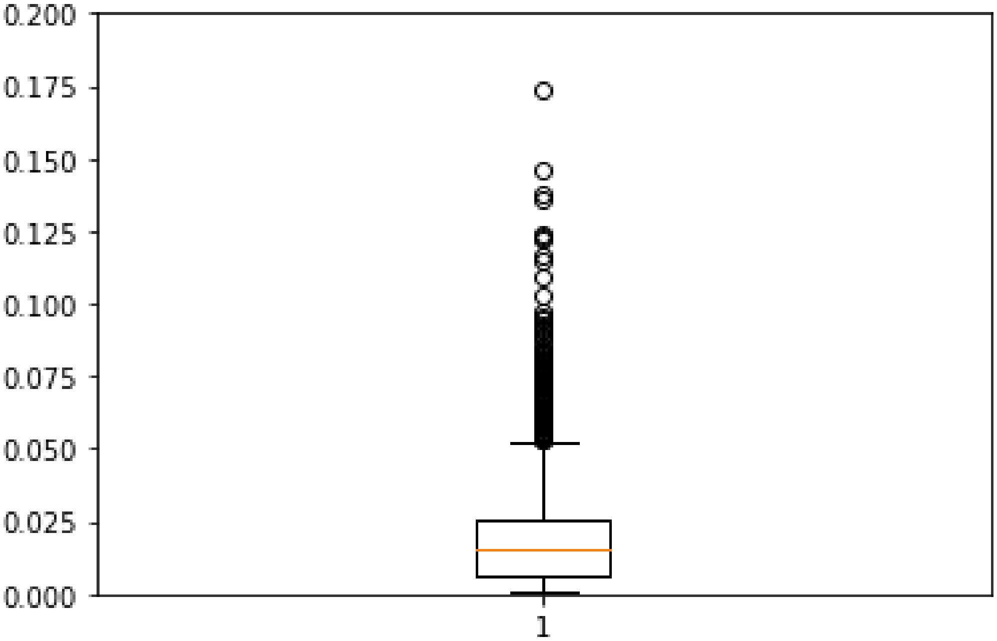


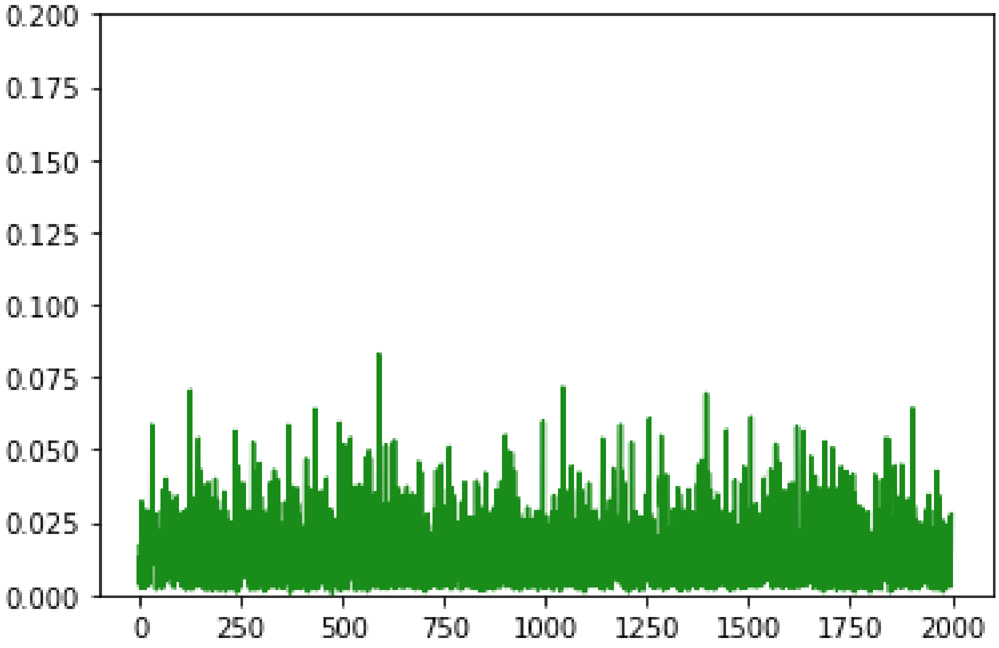
```
0.0103247393542
```

```
In [194]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

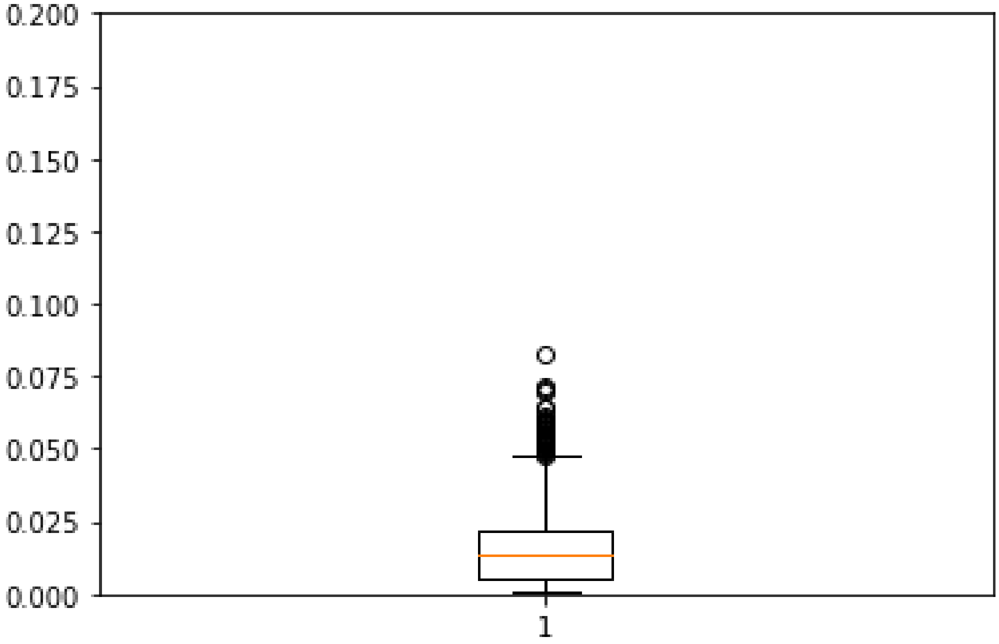


```
0.0192457658947
```





```
0.0155044437412
```



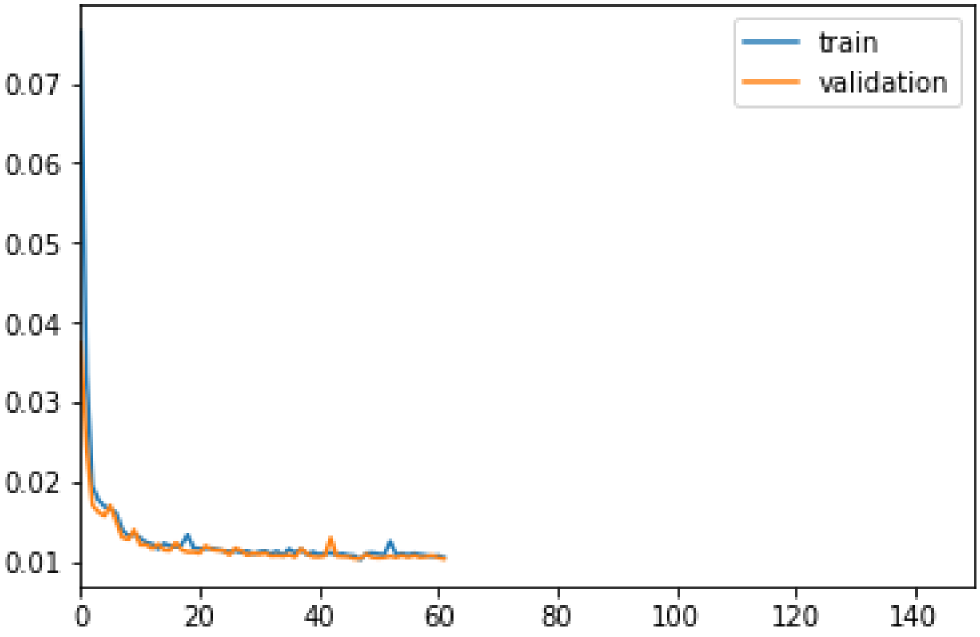
10 steps

```
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)
```

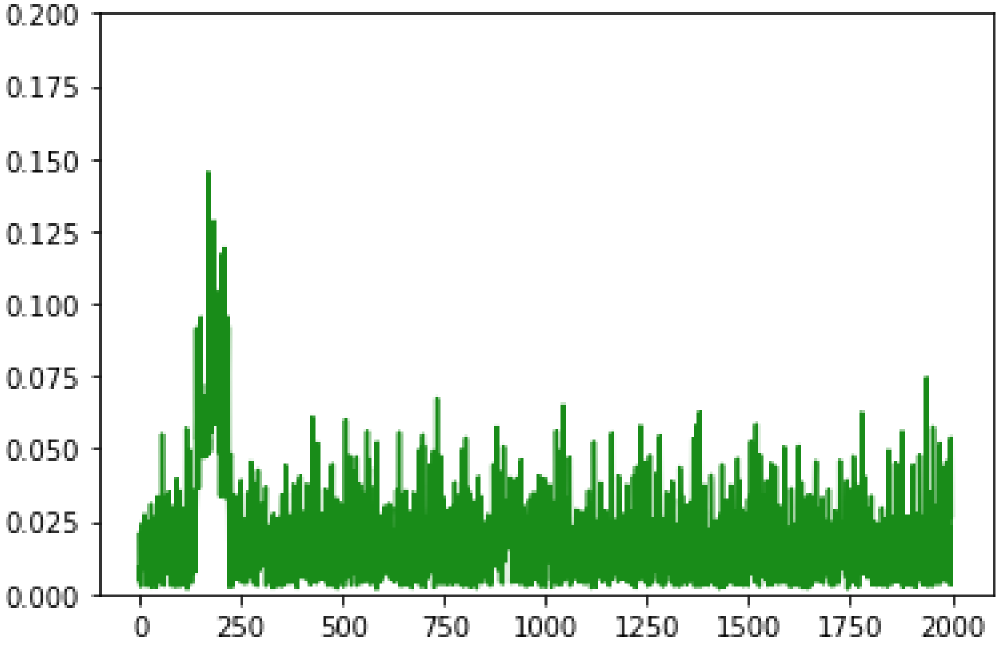
```
In [197]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [198]: train(model, tgen, vgen)
```

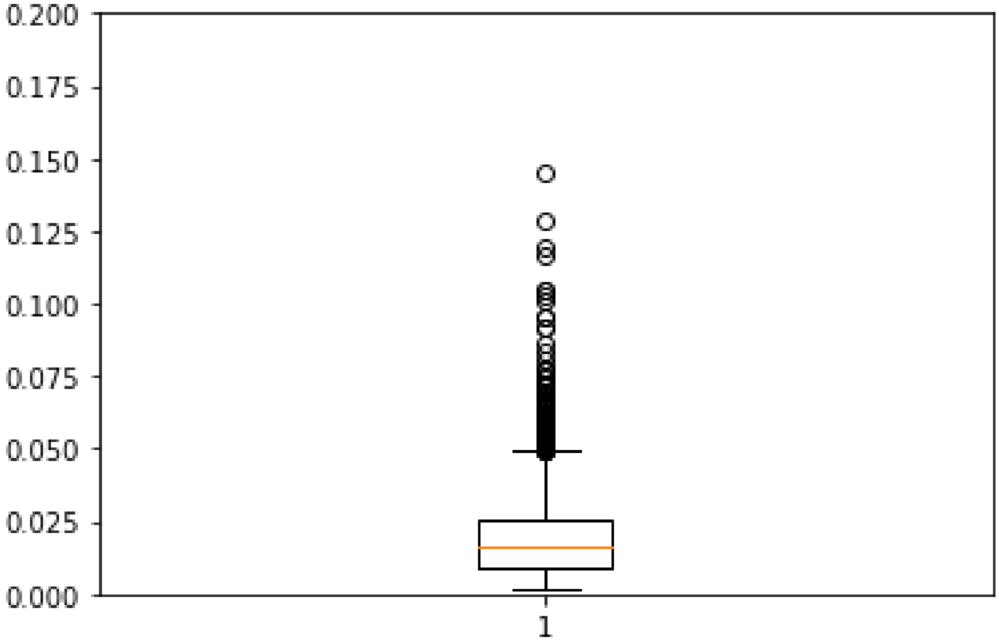


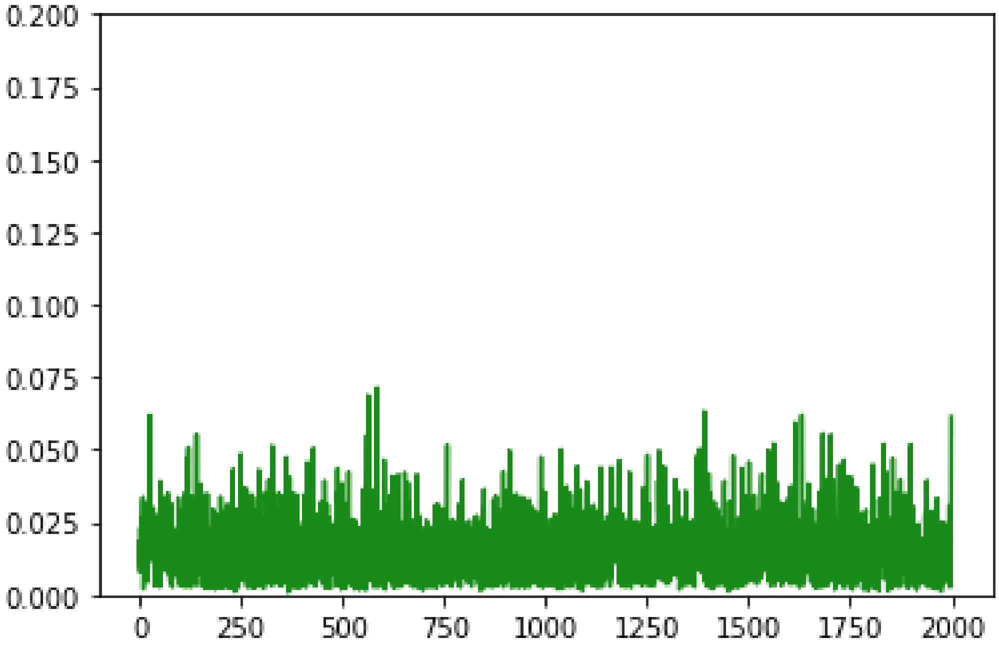
```
0.0105652405741
```

```
In [199]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

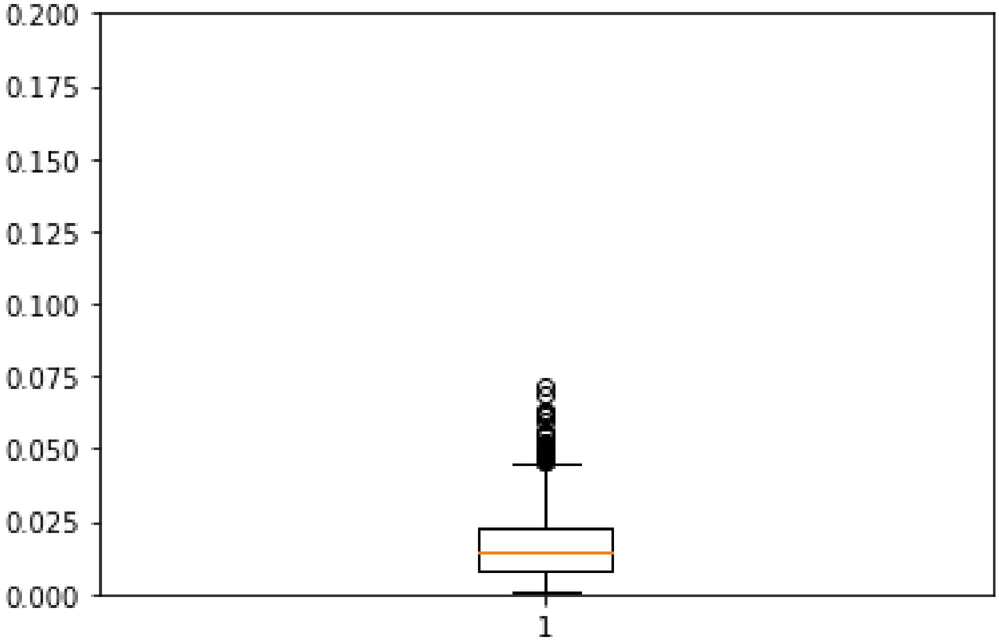


```
0.0197110723512
```



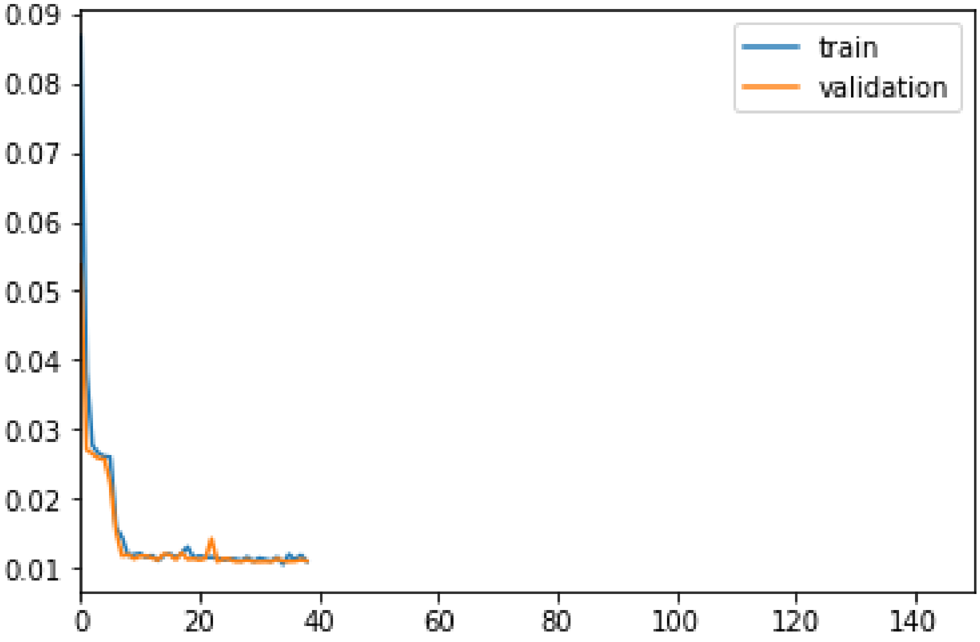


```
0.0163662280787
```



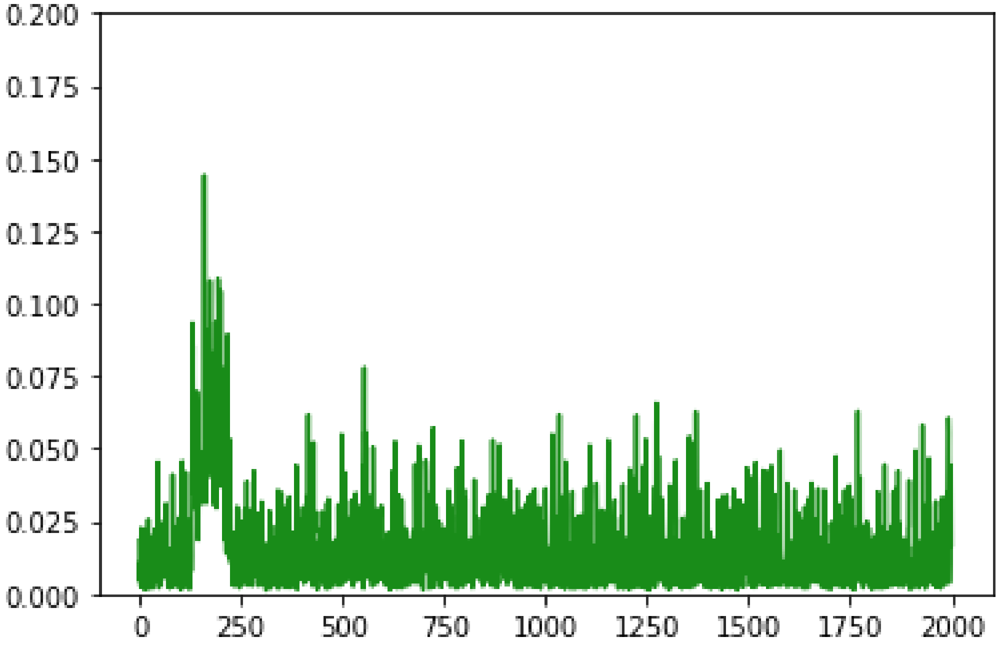
20 steps

```
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)
```

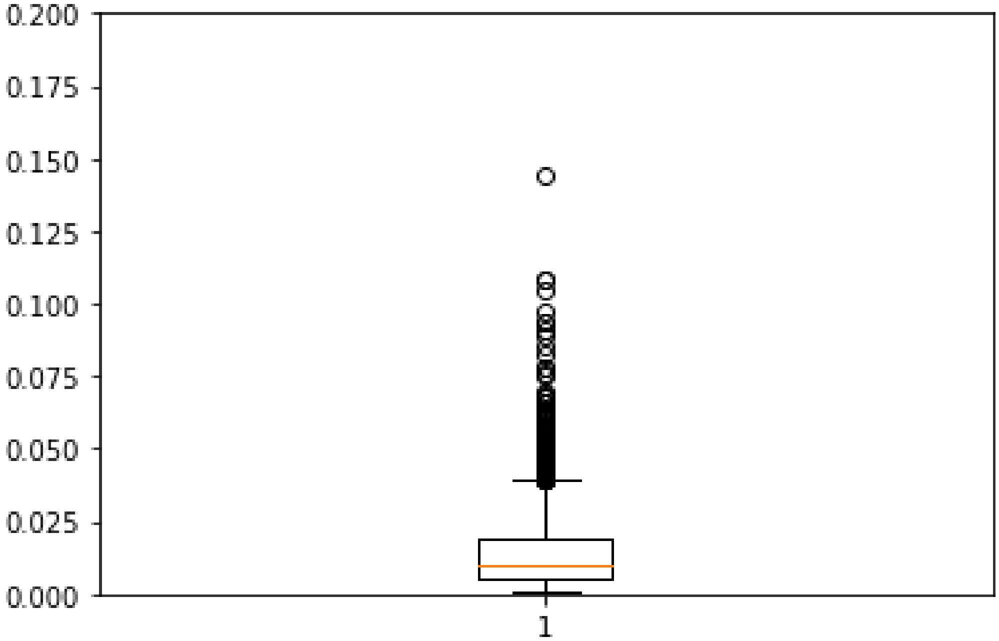


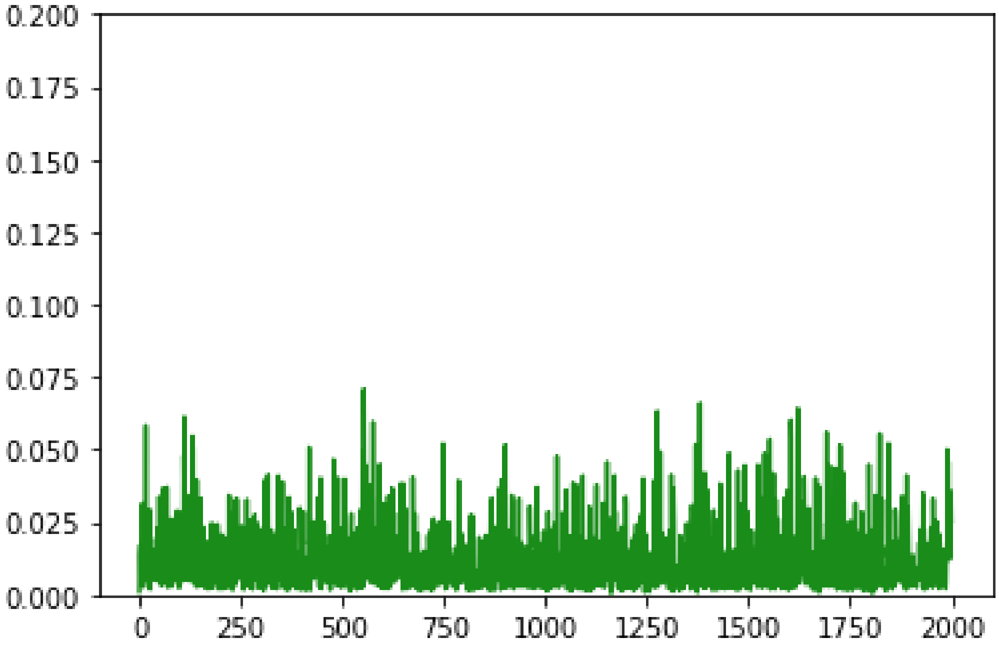
```
0.0109021075461
```

```
In [204]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

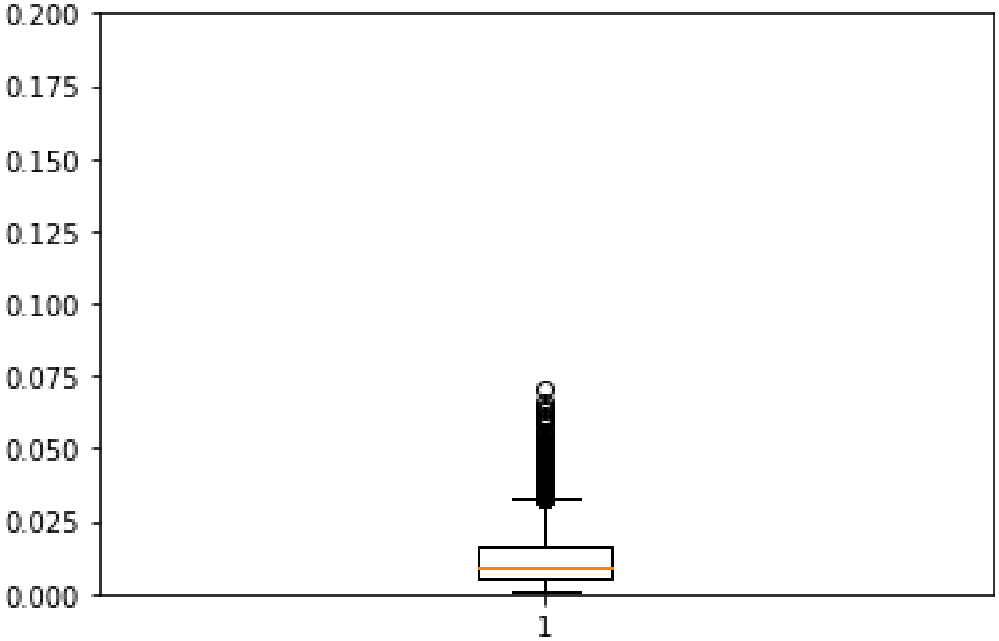


```
0.0149066606174
```





```
0.0124193631963
```



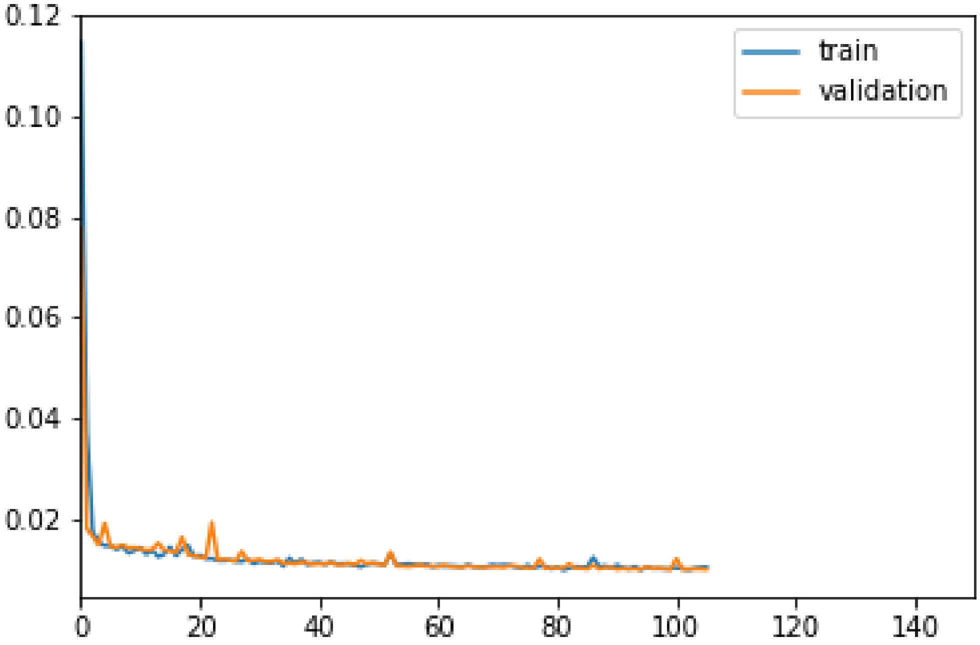
50 steps

```
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)
```

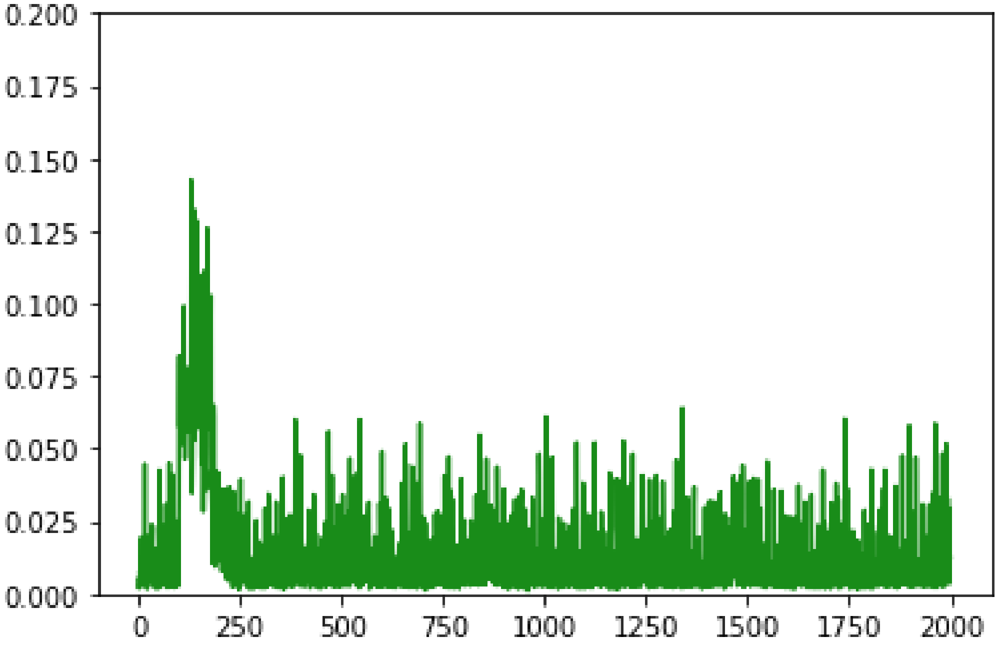
```
In [207]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [208]: train(model, tgen, vgen)
```

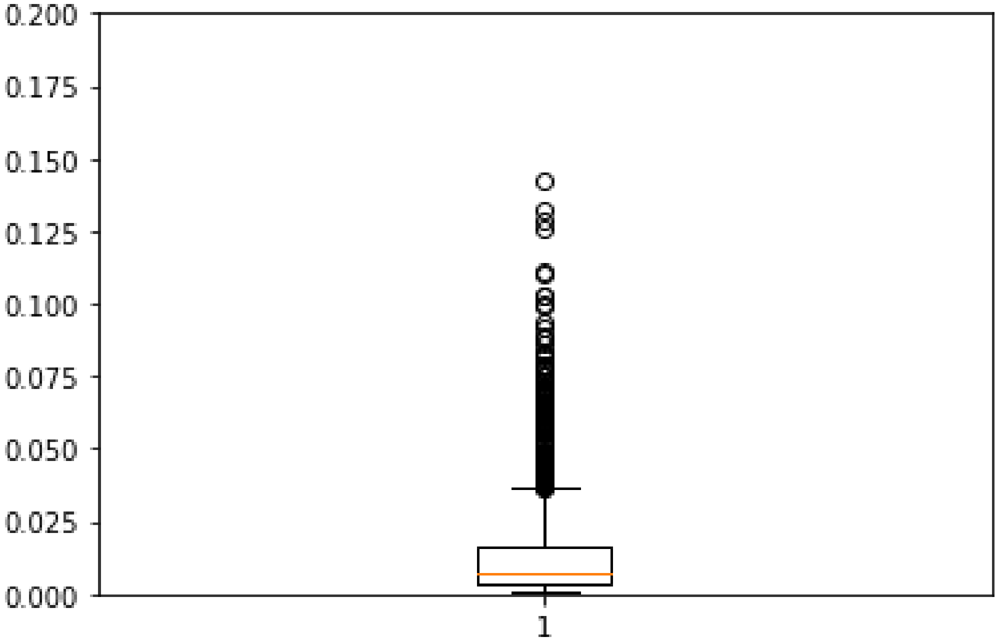


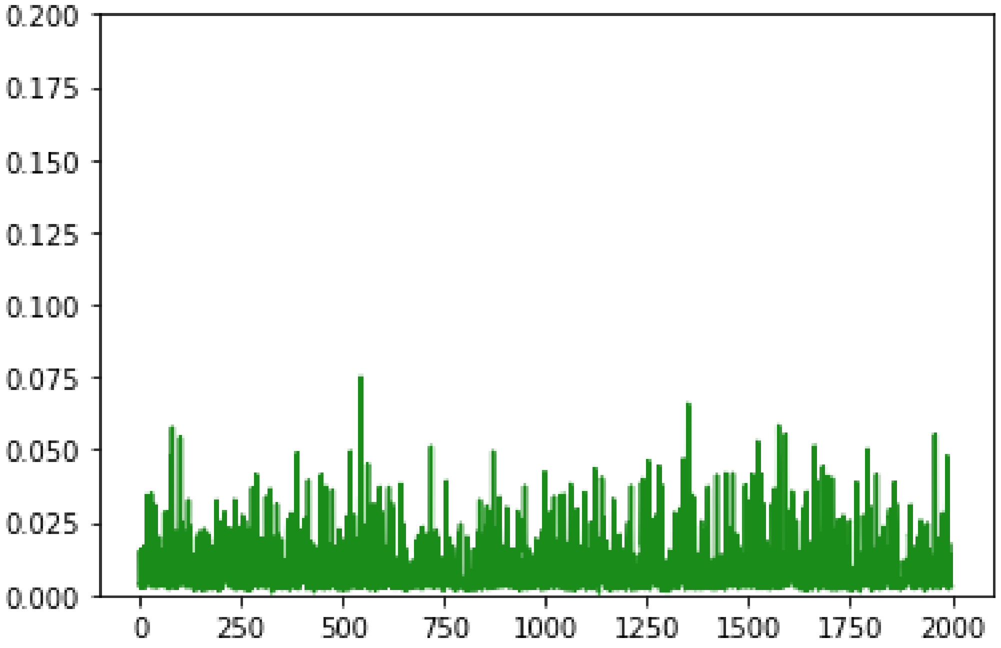
```
0.0104288132561
```

```
In [209]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

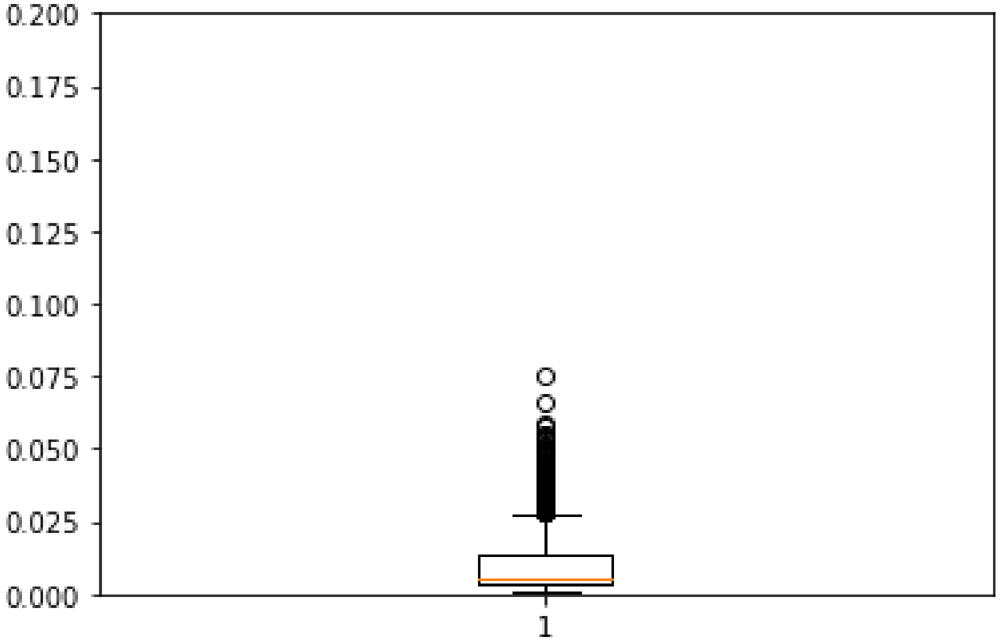


```
0.0136076237793
```





```
0.00976564280366
```



2.1.8 RNN with 4 GRU layers dim compression.

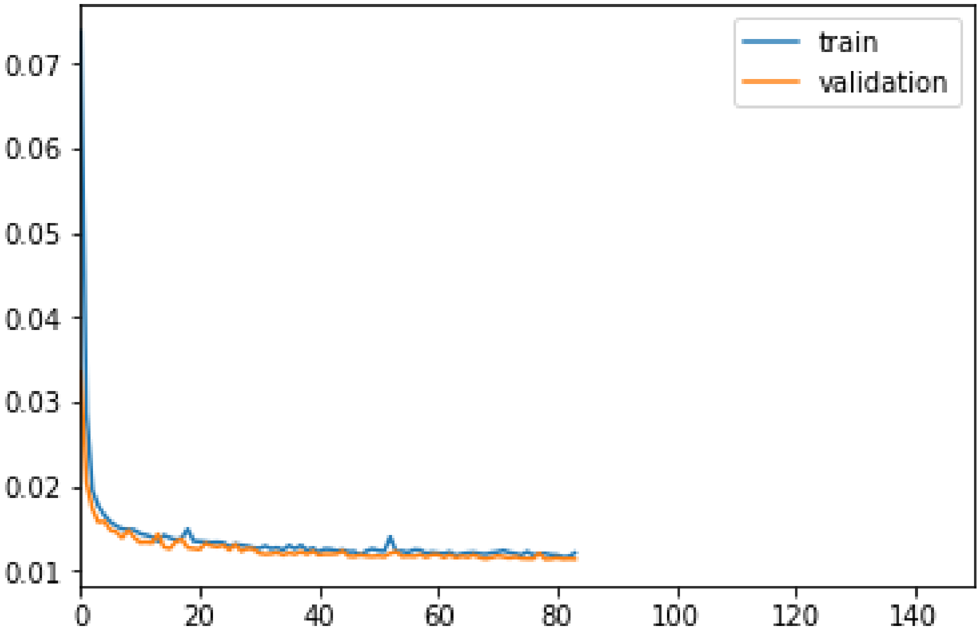
2 steps

```
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)
```

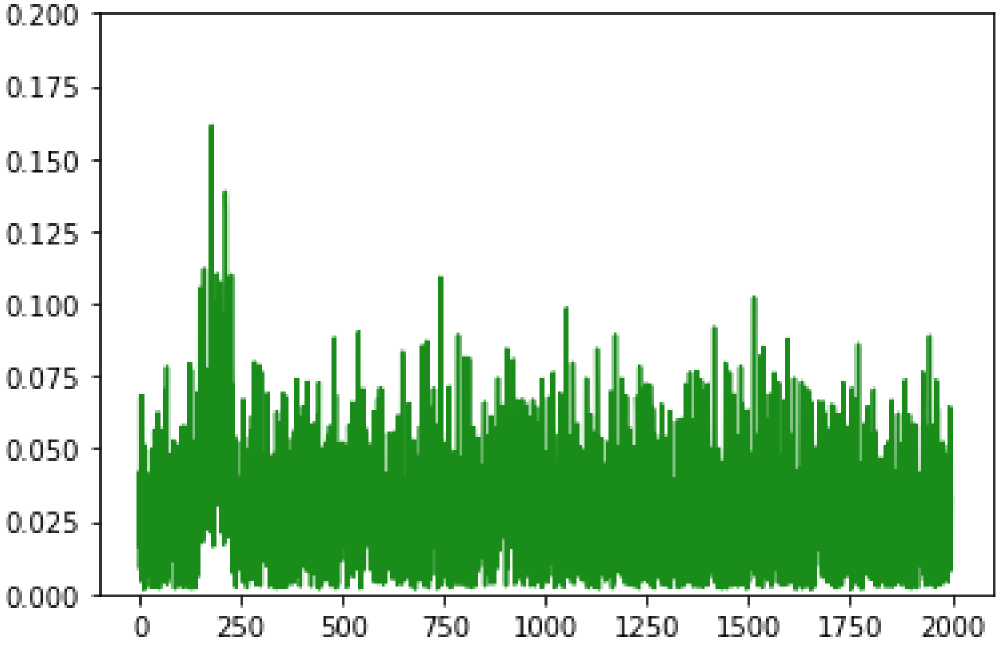
```
In [212]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [213]: train(model, tgen, vgen)
```

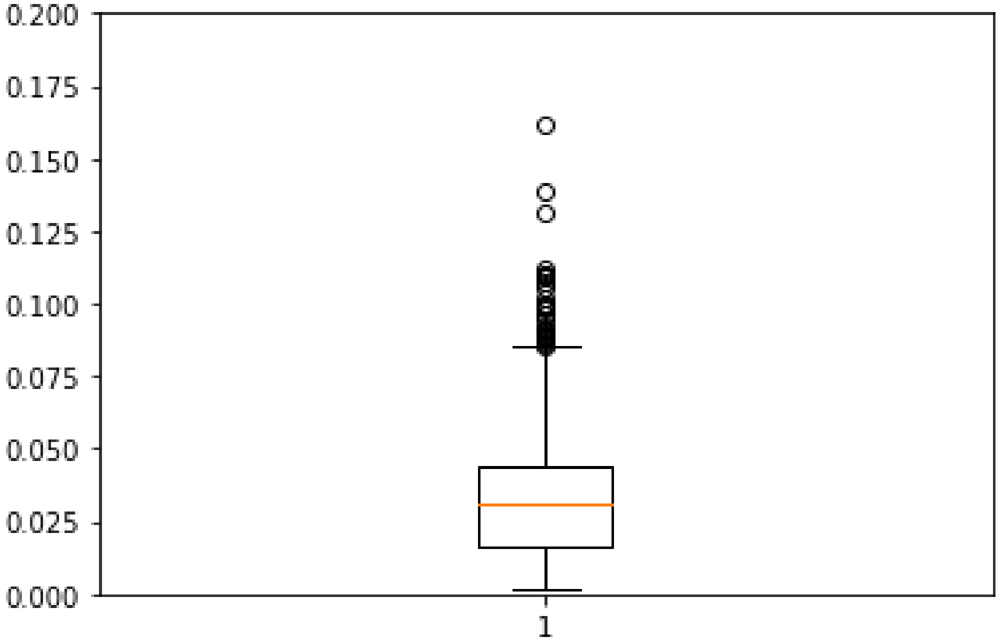


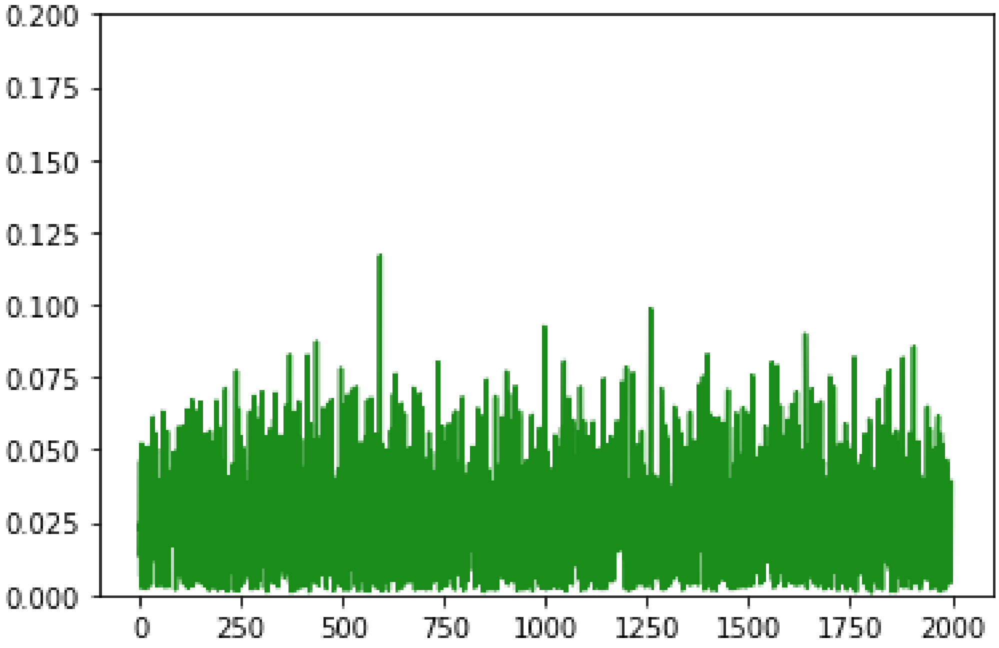
```
0.0121524259411
```

```
In [214]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

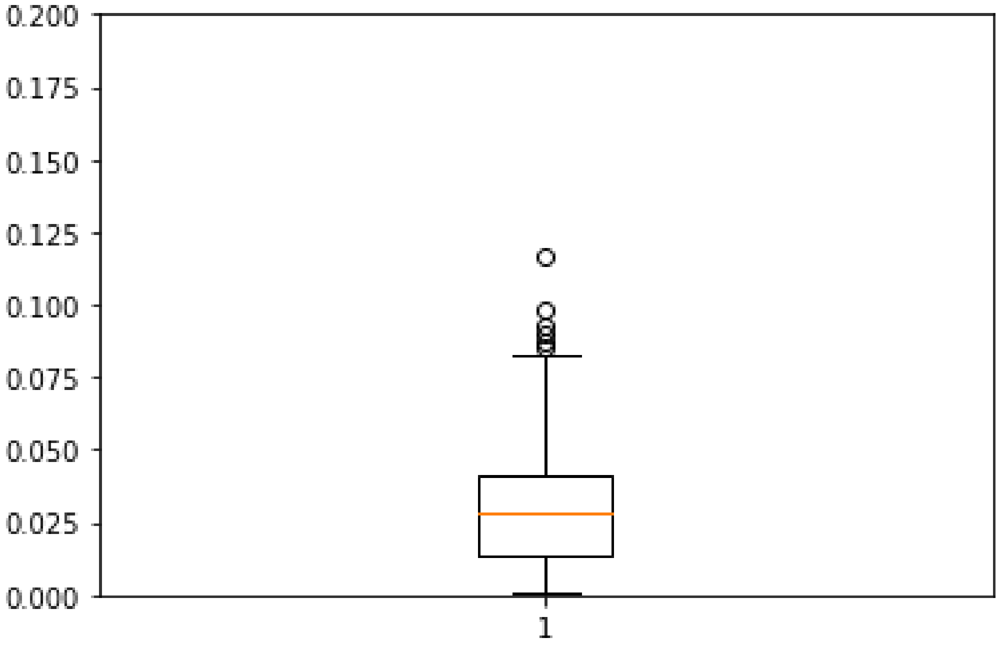


```
0.0320774163956
```





```
0.0292895404746
```



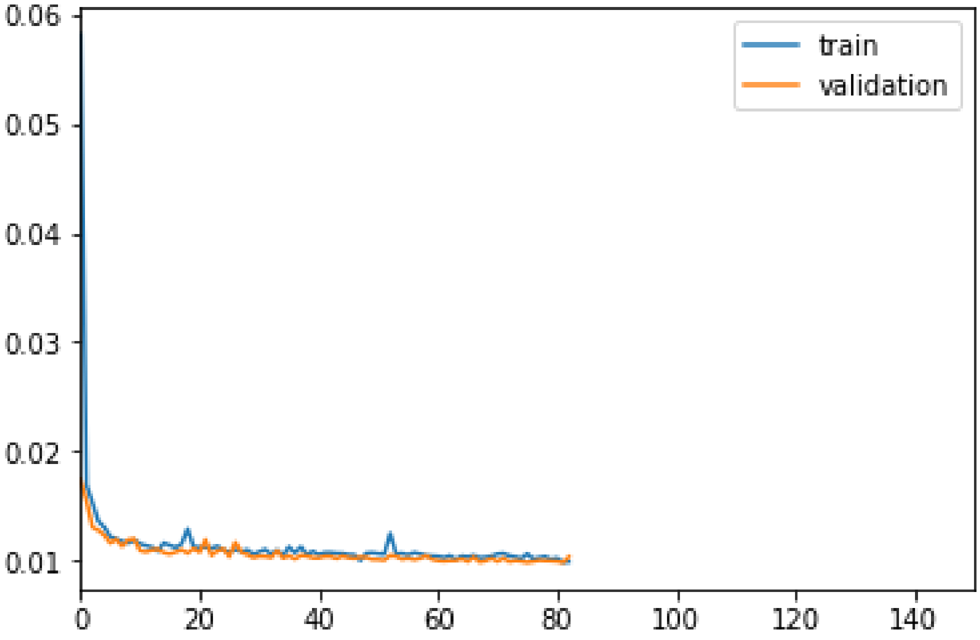
5 steps

```
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)
```

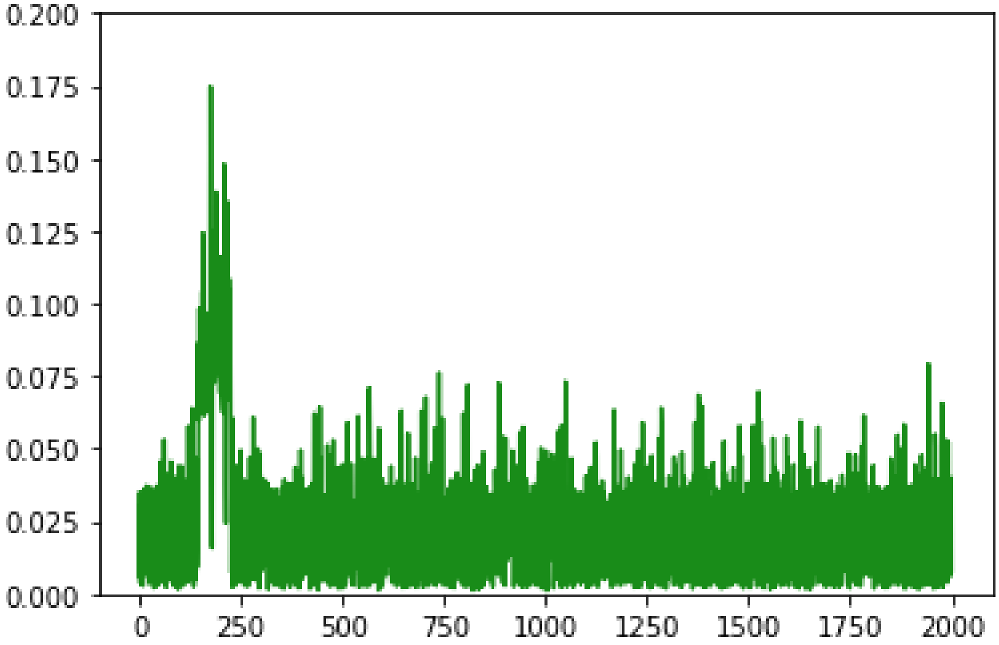
```
In [217]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [218]: train(model, tgen, vgen)
```

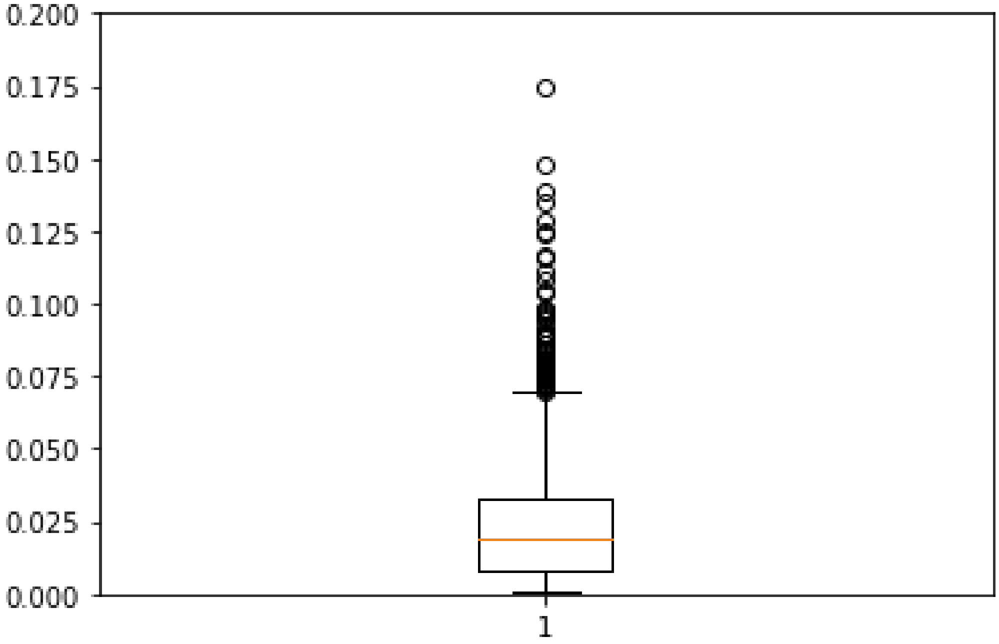


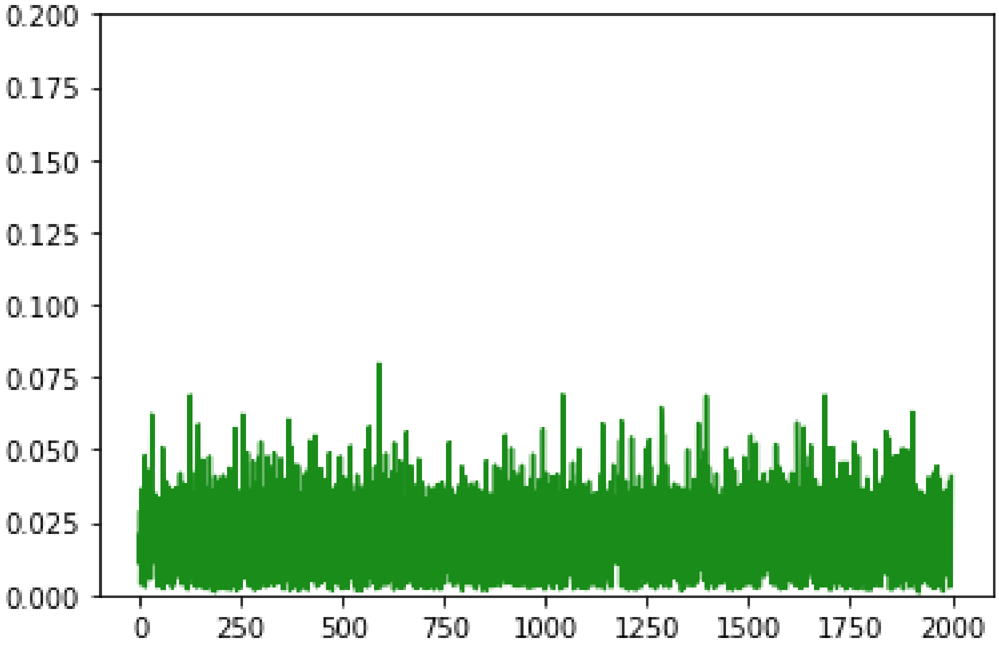
```
0.00989359730587
```

```
In [219]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

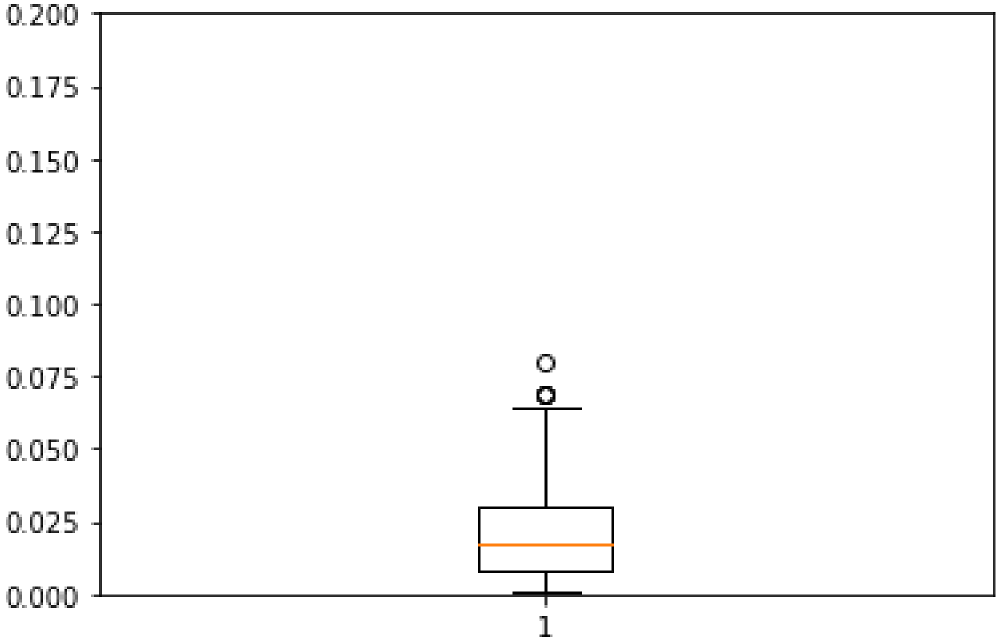


```
0.0232166232826
```





```
0.0199705442353
```



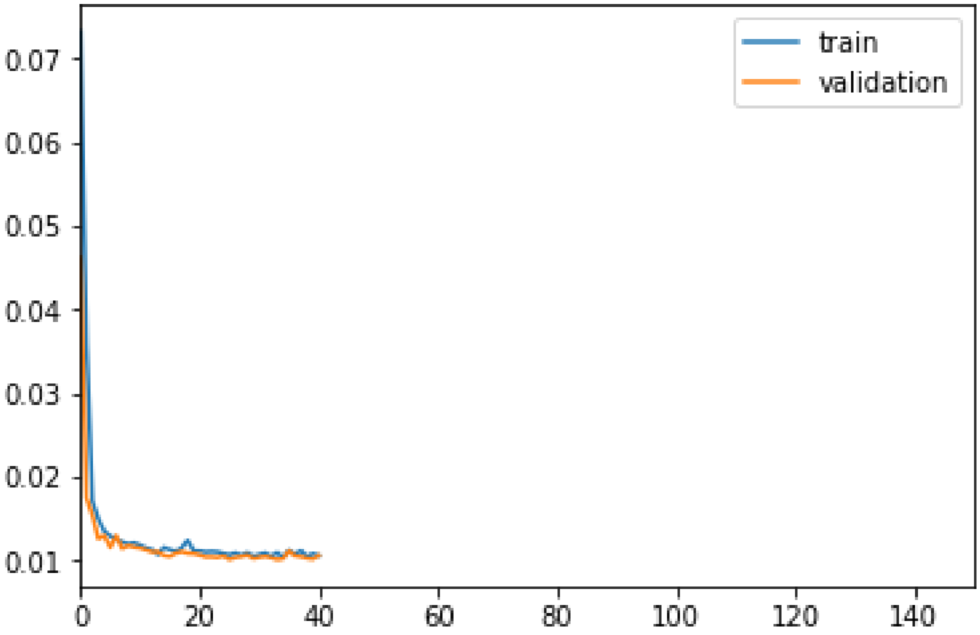
10 steps

```
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)
```

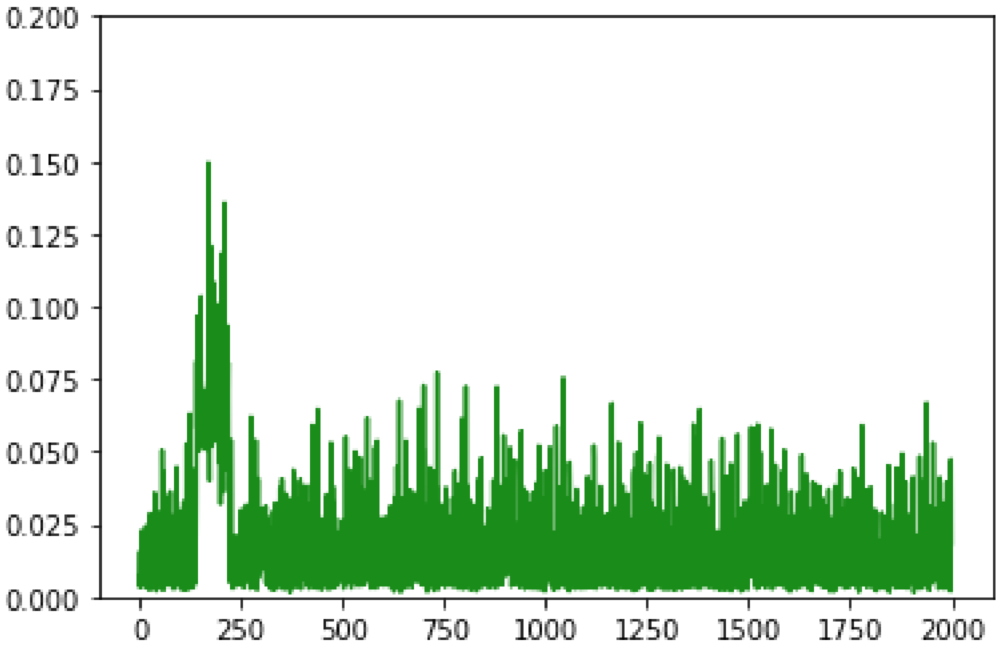
```
In [222]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [223]: train(model, tgen, vgen)
```

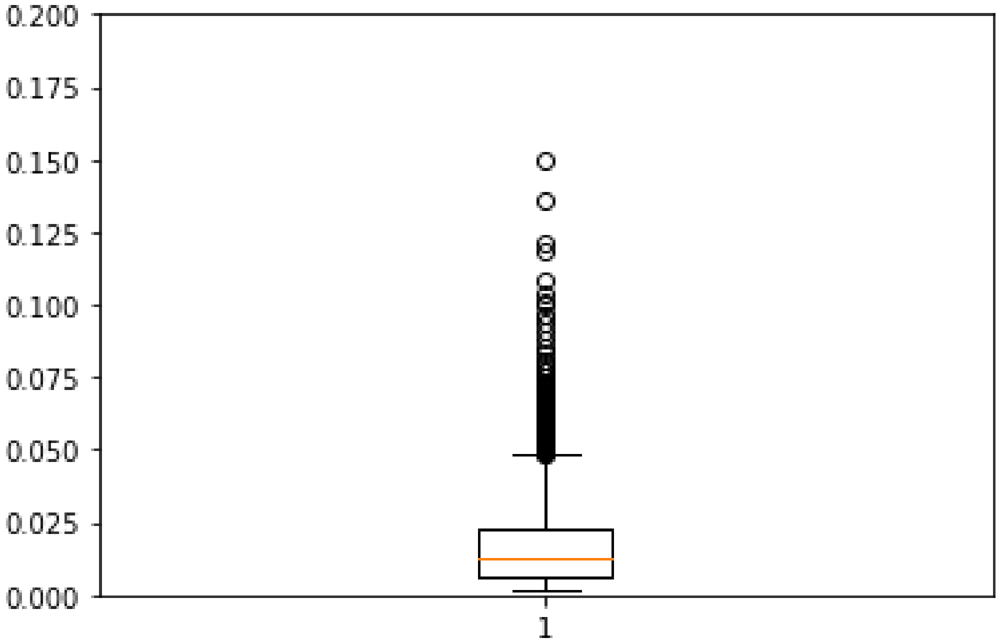


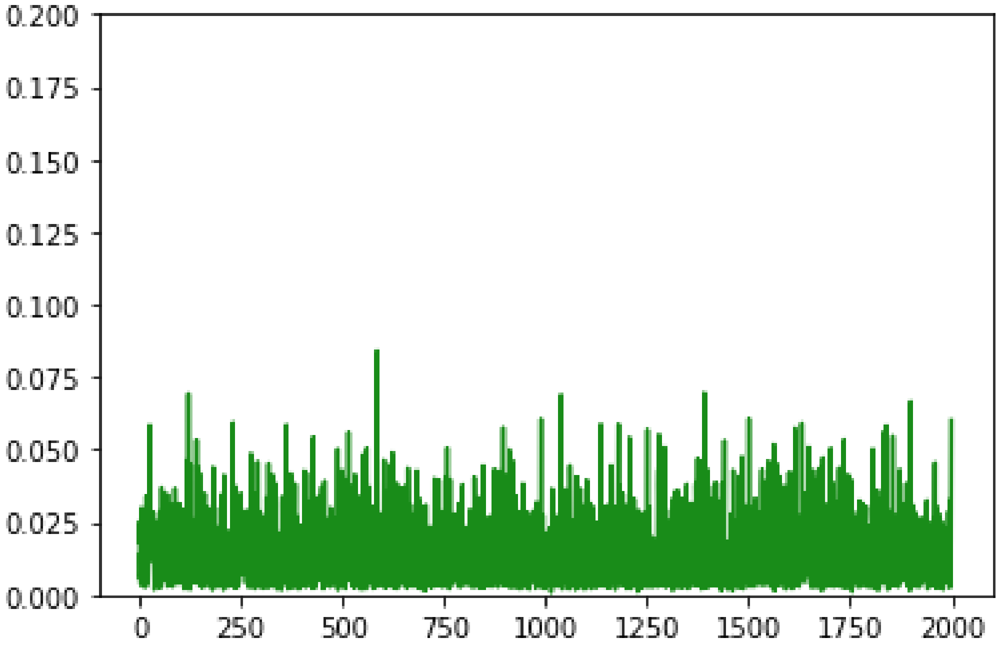
```
0.0105709931646
```

```
In [224]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

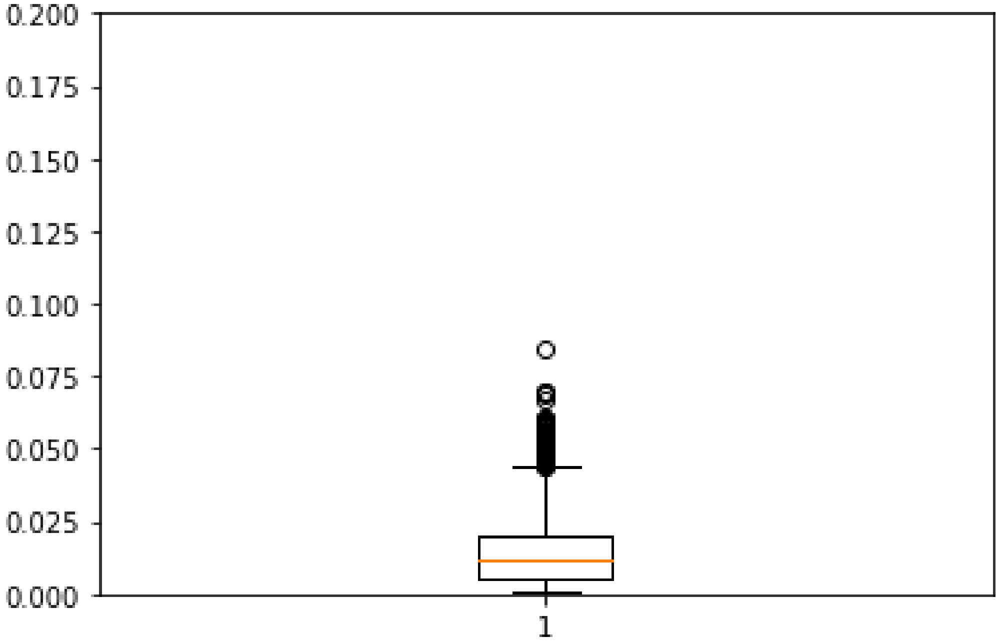


```
0.017791162019
```





```
0.0148359506123
```



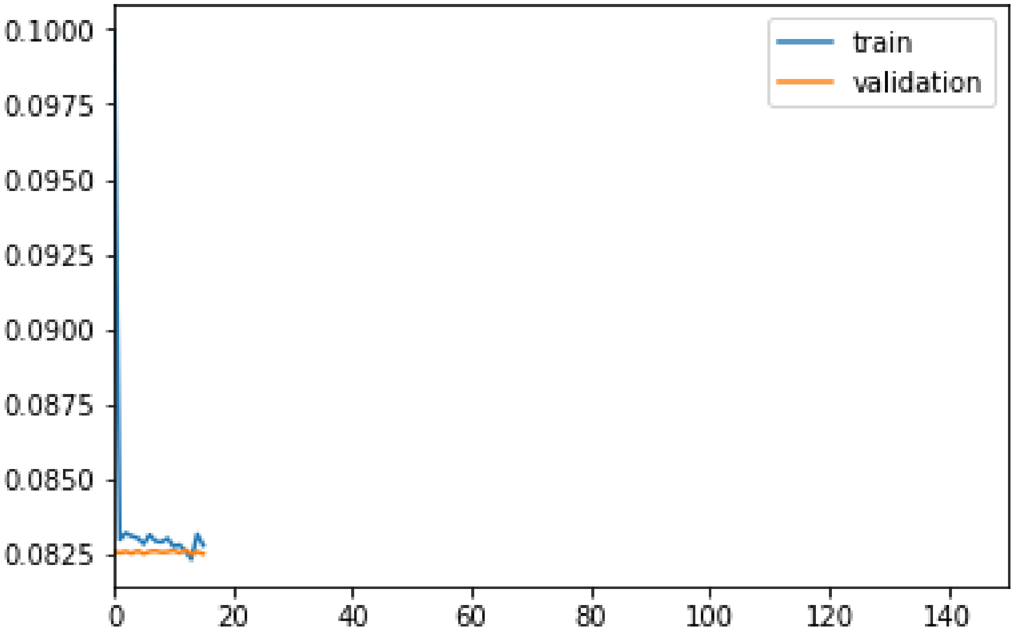
20 steps

```
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)
```

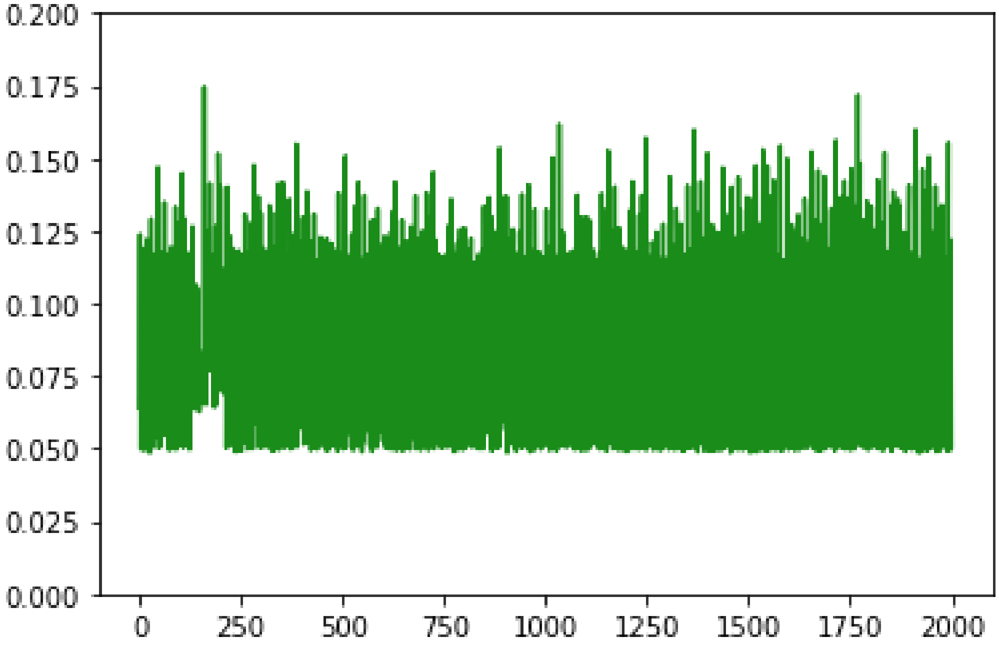
```
In [227]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [228]: train(model, tgen, vgen)
```

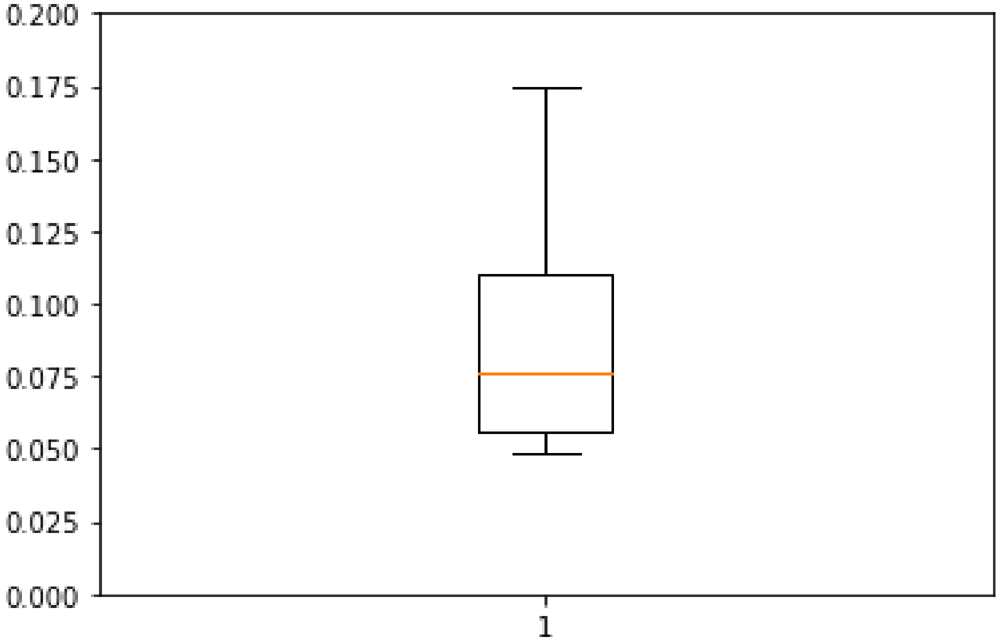


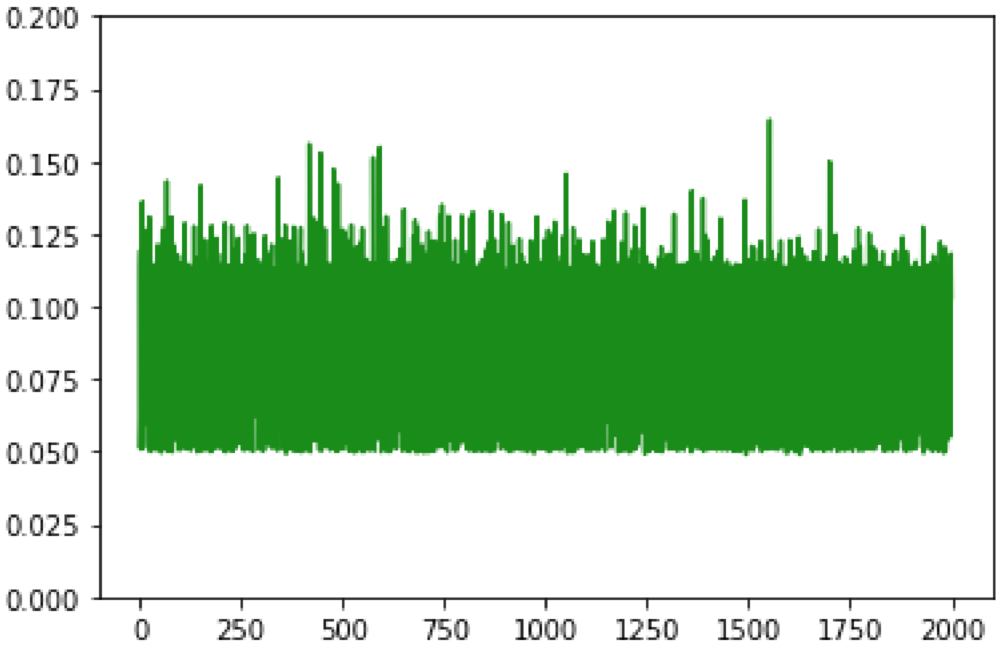
```
0.0828080301918
```

```
In [229]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

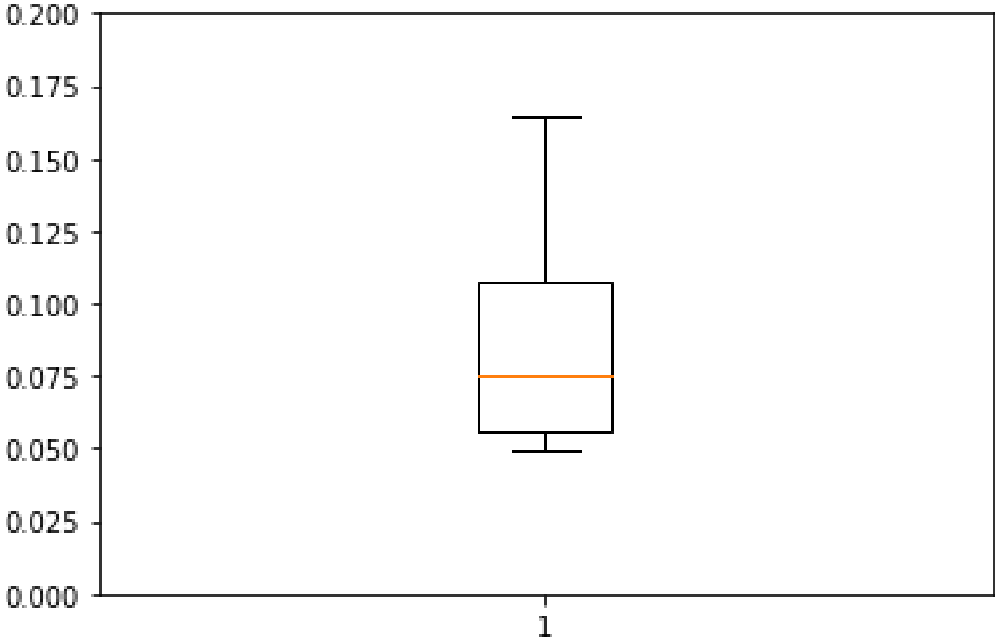


```
0.0831764654175
```





```
0.0815420504655
```



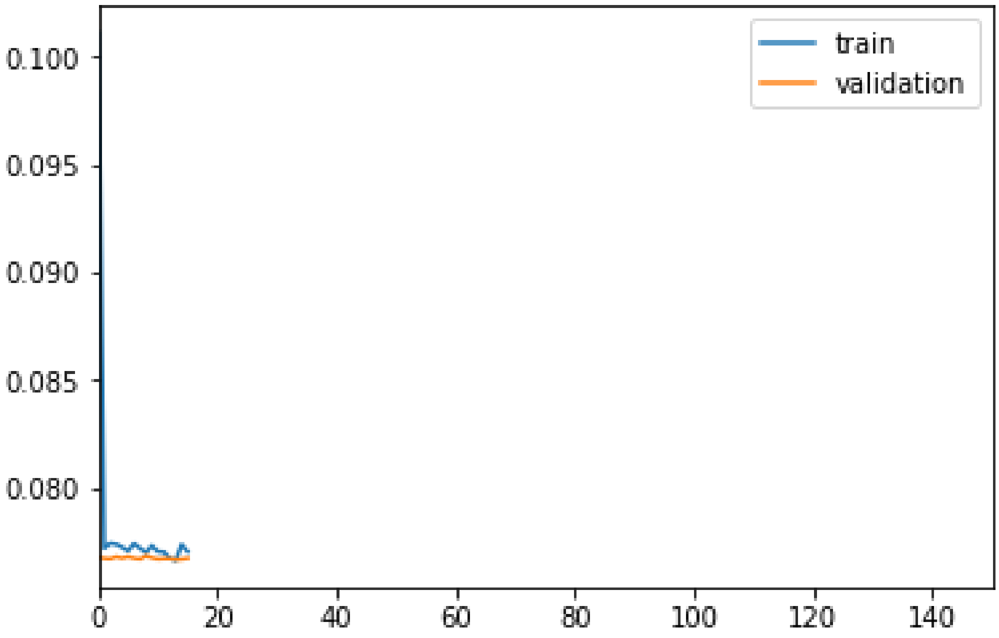
50 steps

```
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)
```

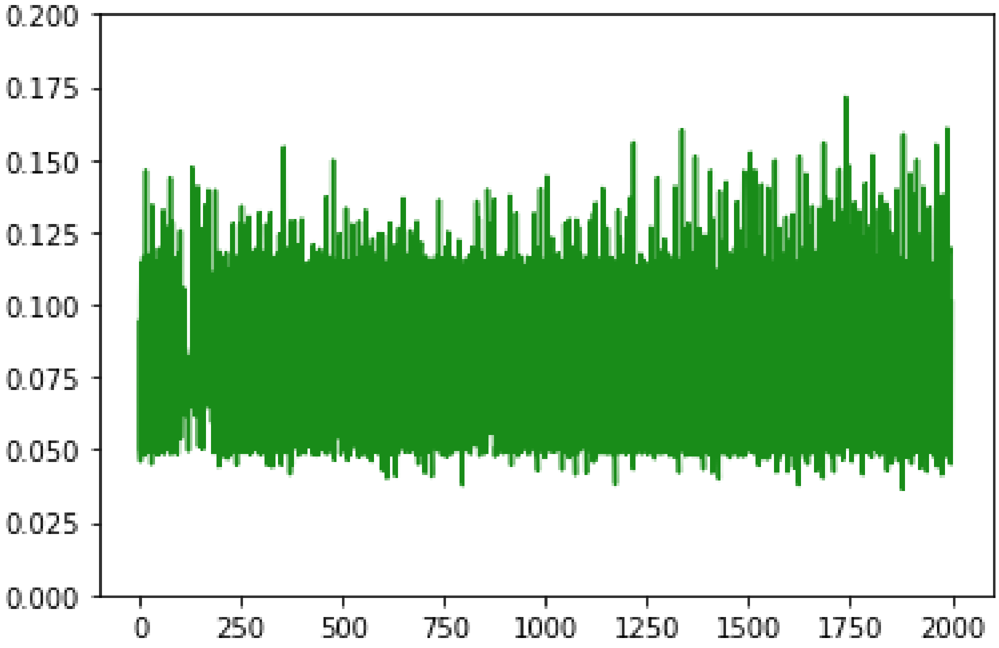
```
In [232]: model = Model(input_layer, output)
          model.compile(loss='mean_absolute_error', optimizer='adam', metrics=['mae'])
```

```
In [233]: train(model, tgen, vgen)
```

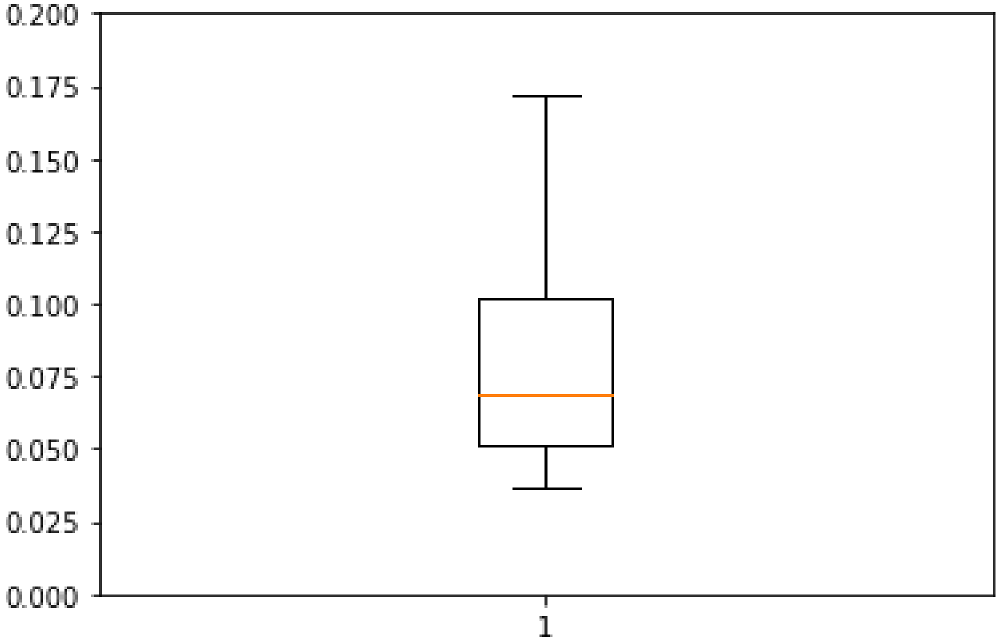


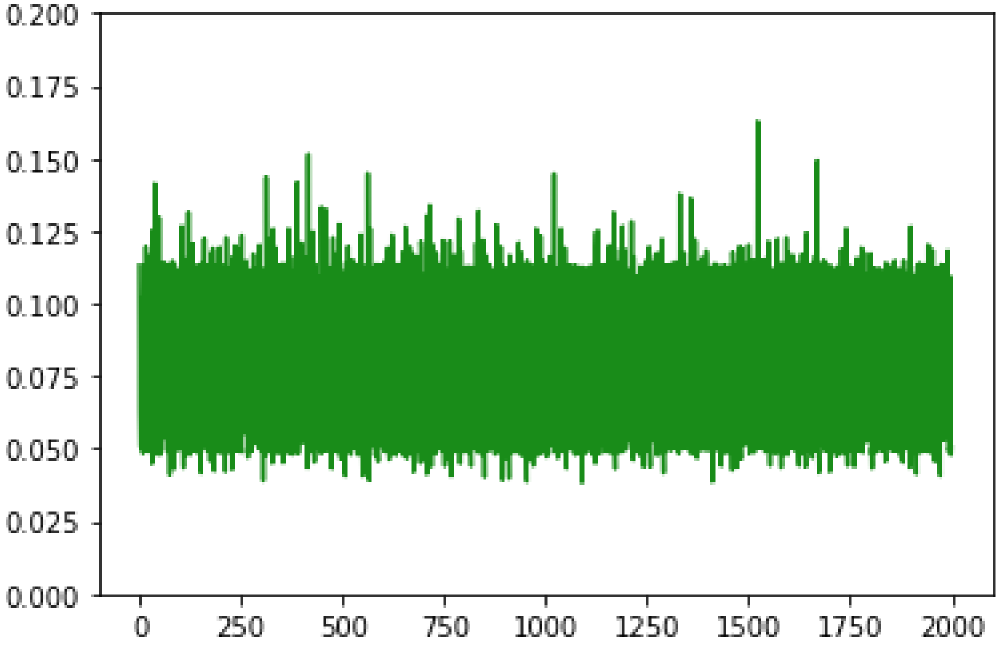
```
0.077035689082
```

```
In [234]: test(model, test_X[0],0)
          test(model, test_X[2],0)
```

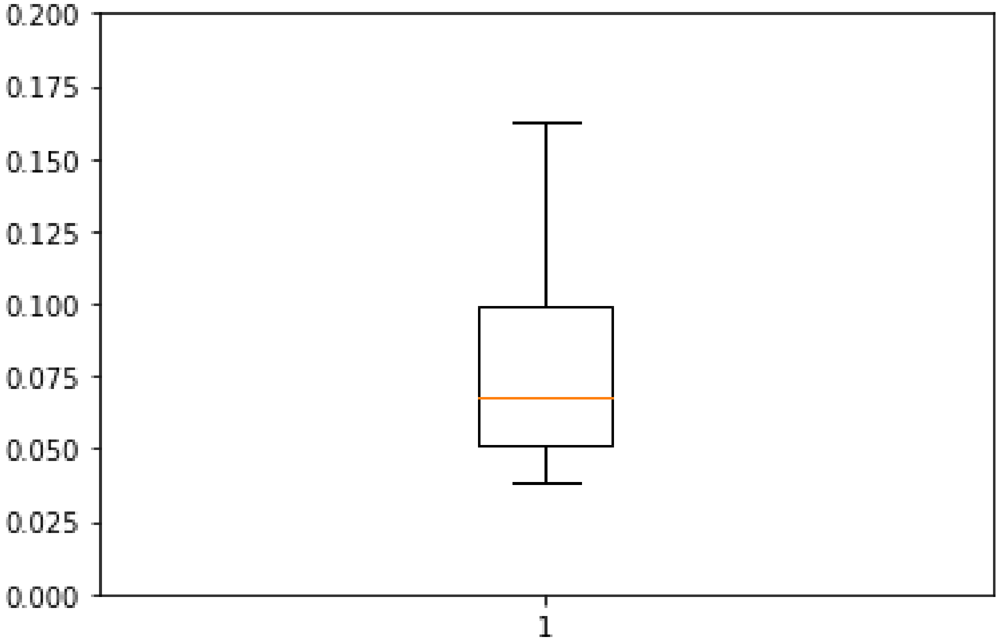


```
0.0776662132452
```





```
0.0753738445685
```


0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
1