	Politechnika Śląska Katedra Grafiki, Wizji Komputerowej i Systemów Cyfrowych			
Rok akademicki:	**Rodzaj studiów*: SSI/NSI/NSM**	**Przedmiot (Języki Asemblerowe/SMiW):**	**Grupa**	**Sekcja**
2020/2021	**SSI**	**BIAI**	**GKiO1**	**43**
Skład sekcji:	Marcin **NASTAŁA** Piotr **SOROCIAK** Sebastian **RICHTER**	**Prowadzący**: OA/JP/KT/GD/BSz/GB	**KH**	

Raport końcowy

Temat projektu:

The use of neural network to predict the change of position of a point on the plane on the basis of previous trajectory of its movement

Data oddania: dd/mm/rrrr	**03/07/2021**

1. Project topic

Our task is to implement a program, which uses neural network to predict the change of position of a point on the plane on the basis of previous trajectory of its movement. This network can be used in an object tracking system.

2. Task analysis

Change of the position of the point in time is a typical time series problem, which can be solved using convolutional neural networks (CNN) and recurrent neural networks (RNN). We decided to use RNN, more specifically long short-term memory network (LSTM). RNNs are generally good for time series data, but they suffer from short-term memory due to the vanishing gradient problem. This causes earlier layers to get a small gradient and basically stop learning once it gets too small. This means that the neural network will forget the earlier information in longer sequences. The solution for that was to use LSTM, which can control, which information from the previous steps and the new input should be kept and which forgotten.

3. Data source

Data used to test and train neural network was a series of two numbers separated by tab stored in text files. This was the representation of the coordinates x and y of a point and their change over time. These files represent bike ride.

4. Technologies

Qt library was used to create GUI. In the context of tests we create plots using Python library called Matplotlib. We decided to use Python programming language for our project, because it contains many popular and useful libraries used for creating and training neural networks. Out of them we decided to use PyTorch, because it is both easy to learn and powerful library, thanks to which we are able to create and train a neural network and make predictions without any problems.

5. Software specification

In class Simulation_model we defined the function simulate, where we train the network and make predictions. At first, we declare, how many points we are going to take to train a network (at the beginning we take 50 first points from dataset). Next in while loop we predict next N points (N - number of points to predict) until we achieve the end of dataset. We get first X(50,60,...) points from list of all points to train, convert selected dataset to numpy array and normalize our data to range <-1,1>.

```
normalized_training_data=scaler.fit_transform(training_data.reshape(-1,
1))
```

The next step is to convert our dataset into tensors since PyTorch models are trained using tensors.

```
normalized_training_data=torch.FloatTensor(normalized_training_data).vi
ew(-1)
```

Later we convert our training data into sequences and corresponding labels using create_inout_sequences function. Next we declare number of epochs for training a network (number of points to train + 13 - we can declare more points if we want) and in for loop train a neural network.

```
self.epochs = self.points + 13 # training a network for N+13 epochs
            for _ in range(self.epochs):
                for seq, labels in training_inout_seq:
                    self.optimizer.zero_grad()
                            self.model.hidden_cell = (torch.zeros(1, 1,
self.model.hidden_layer_size),
                                                      torch.zeros(1, 1,
self.model.hidden_layer_size))

                    y_pred = self.model(seq)

                    single_loss = self.loss_function(y_pred, labels)
                    single_loss.backward()
                    self.optimizer.step()
```

After training we get last N samples from a normalized training dataset to make predictions. Next we evaluate a model of our neural network and prepare predictions in a for loop.

```
self.model.eval()

            for _ in range(TEST_DATA_SIZE): #for each point to predict
                seq = torch.FloatTensor(test_inputs[-TEST_DATA_SIZE:])
                with torch.no_grad():
                                self.model.hidden = (torch.zeros(1, 1,
self.model.hidden_layer_size),
                                                     torch.zeros(1, 1,
self.model.hidden_layer_size))
                    test_inputs.append(self.model(seq).item())
```

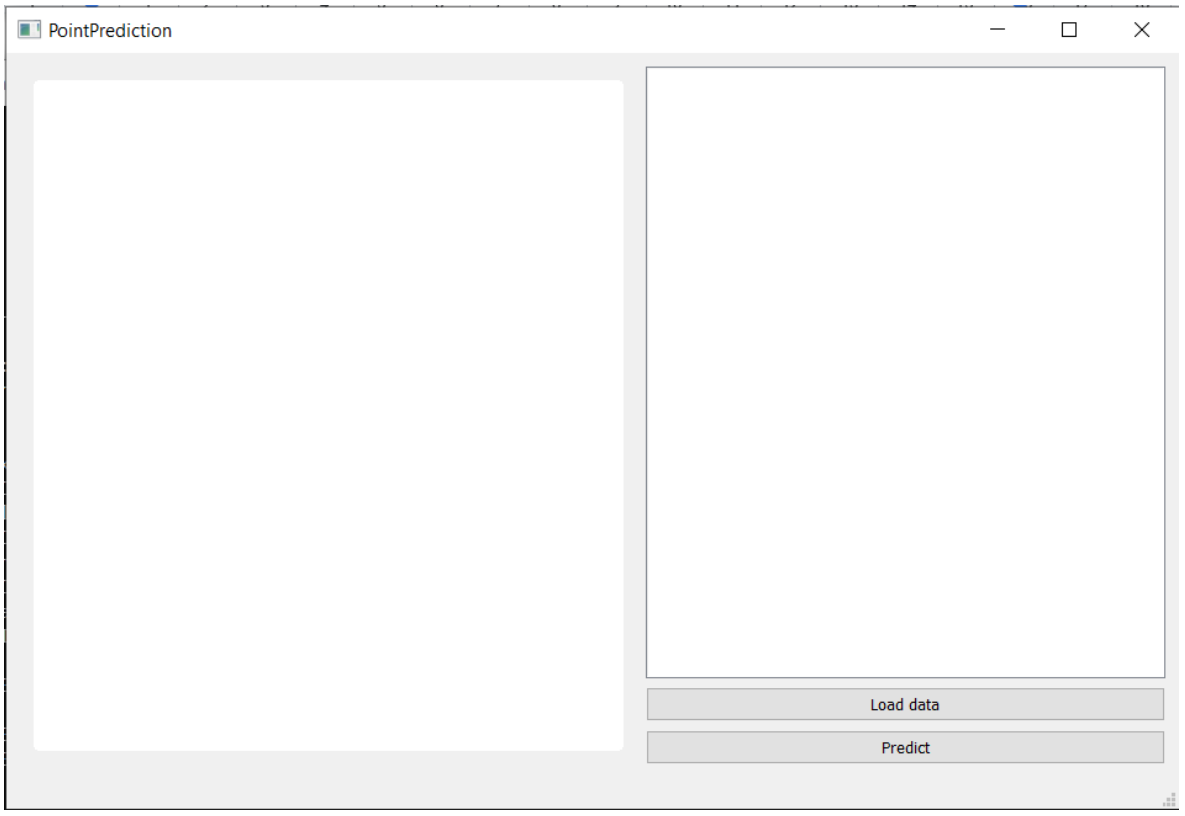
Initially the test_inputs item will contain N items. Inside a for loop these N items will be used to make predictions about the first item from the test set i.e. the point no. 51. The predict value will then be appended to the test_inputs list. During the second iteration, again the last N items will be used as input and a new prediction will be made which will then be appended to the test_inputs list again. The for loop will execute for N times since there are N elements in the test set. At the end of the loop the test_inputs list will contain 2xN items. The last N items will be the predicted values for the test set. Next we convert the normalized predicted values into actual predicted values, add them to the list of all predicted points and increase a number of points in a training dataset by a number of points we are going to predict.

6. Testing

At first we write on the command line:

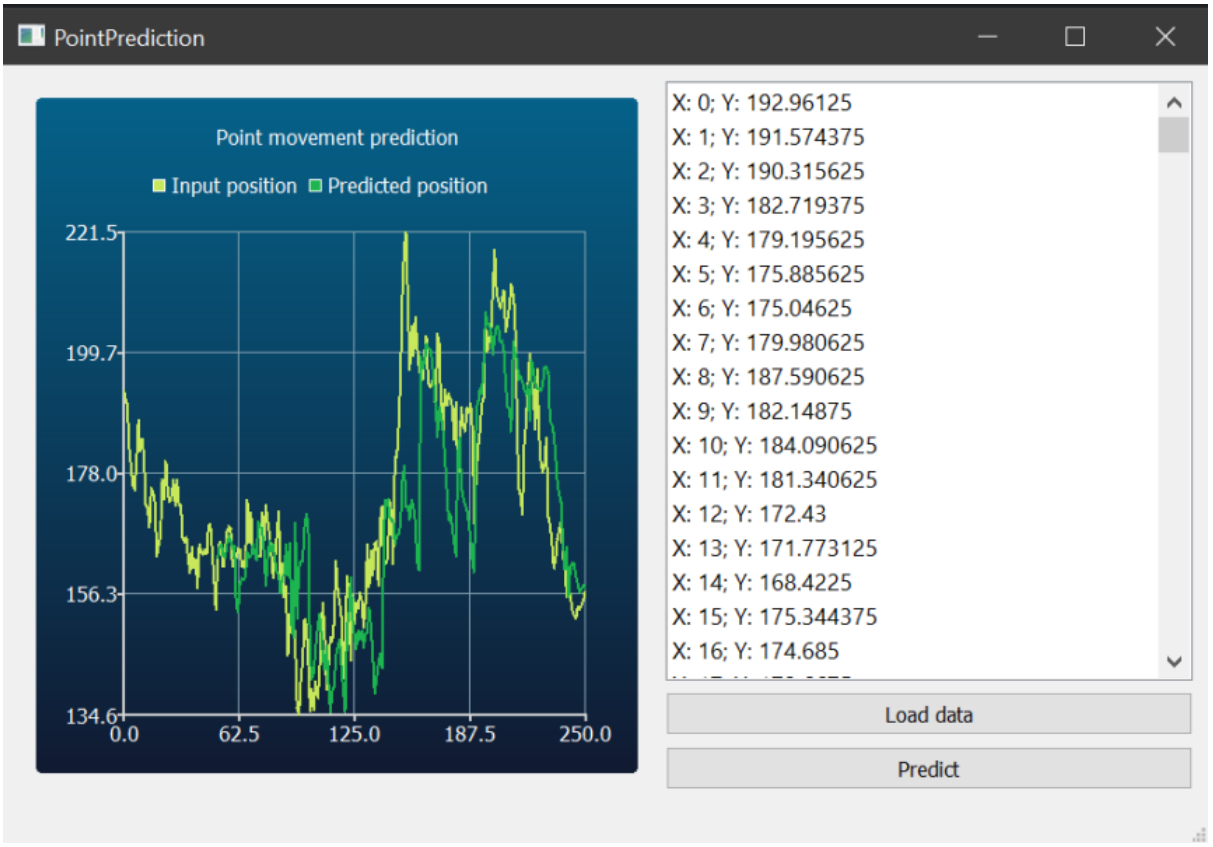
```
(src-gFFGbH7x) C:\BIAI-Projekt\src>python gui.py
```

Later the window of the program should be shown:



We have two buttons: “Load data”, thanks to which we can choose and load dataset of points to analyze and “Predict”, which allow us to train a network and make predictions on the basis of chosen dataset.

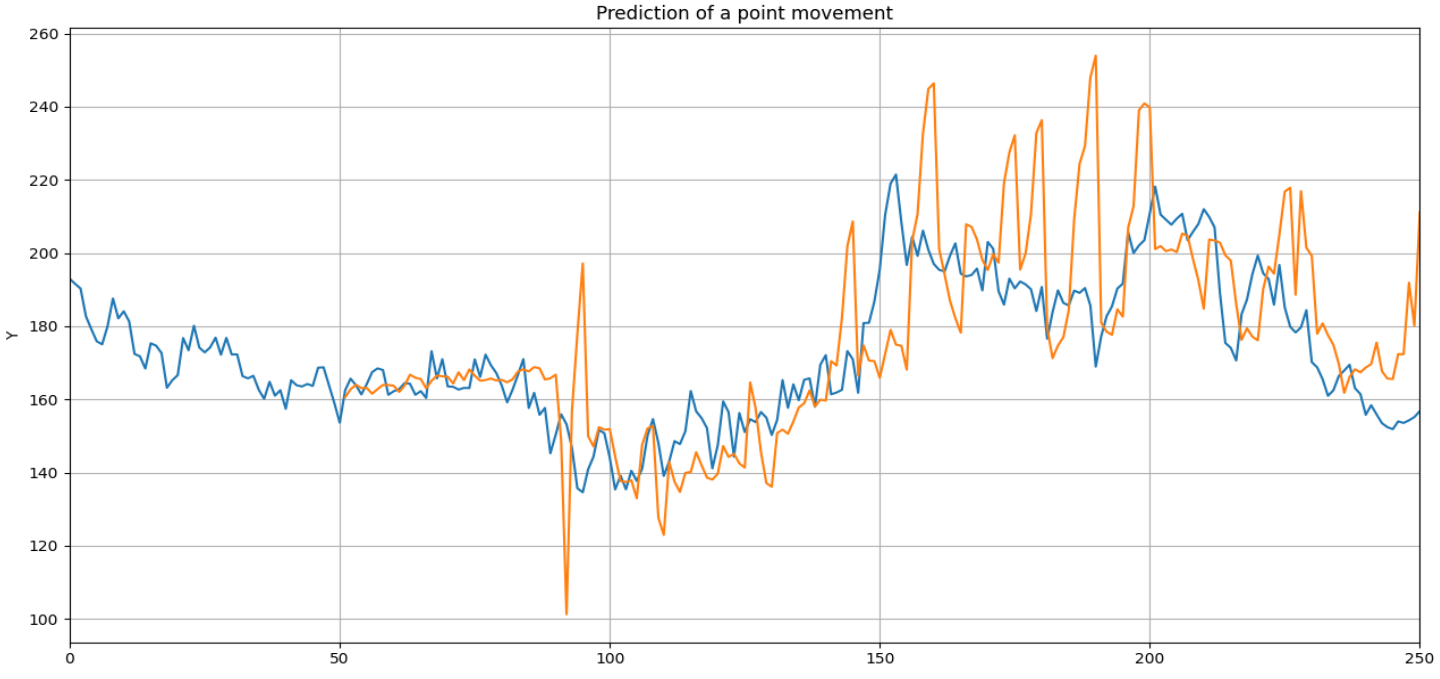
Results should be as similar as shown below:



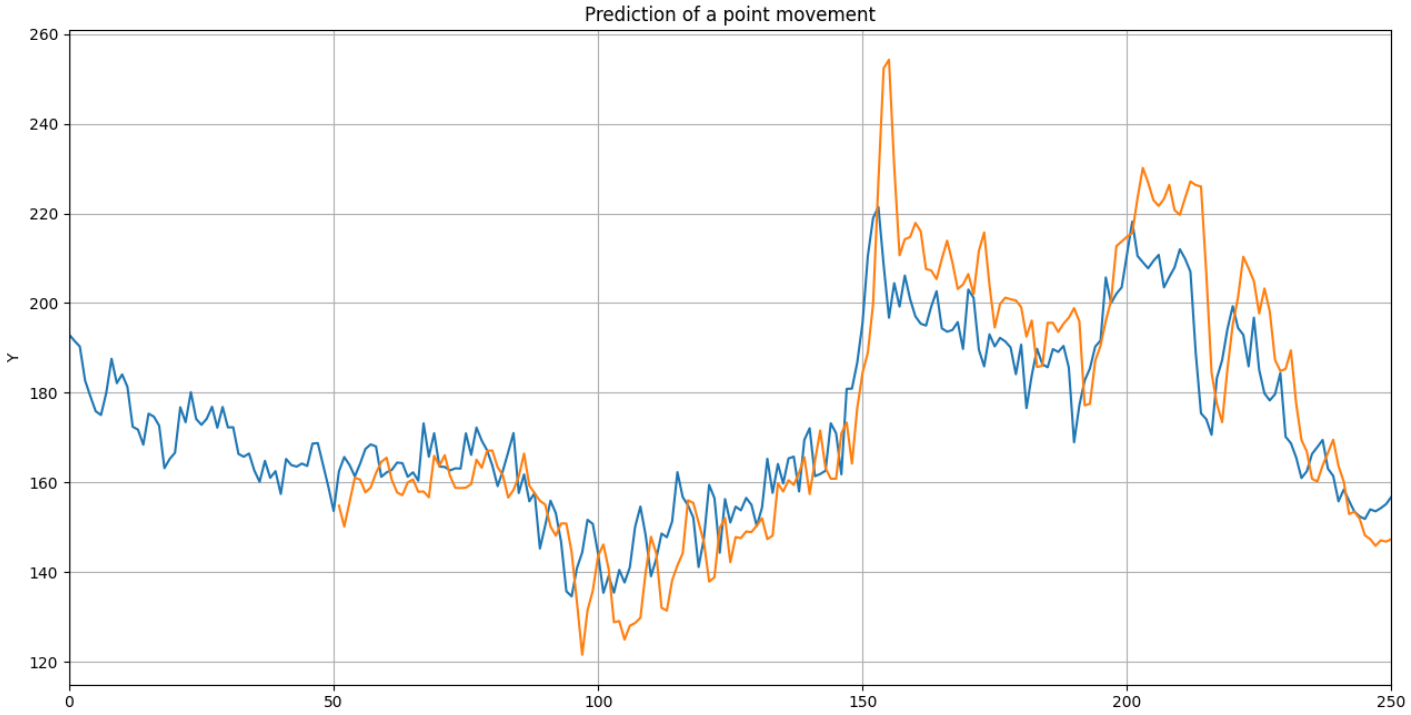
Creating this prediction for each step we predict 10 points in advance, using neural network having 100 neurons (chosen dataset: prosta_12.txt).

In the plot axis X represents number of a point in dataset and axis Y represents a placement of a point on the Y axis in 2D coordinate system.

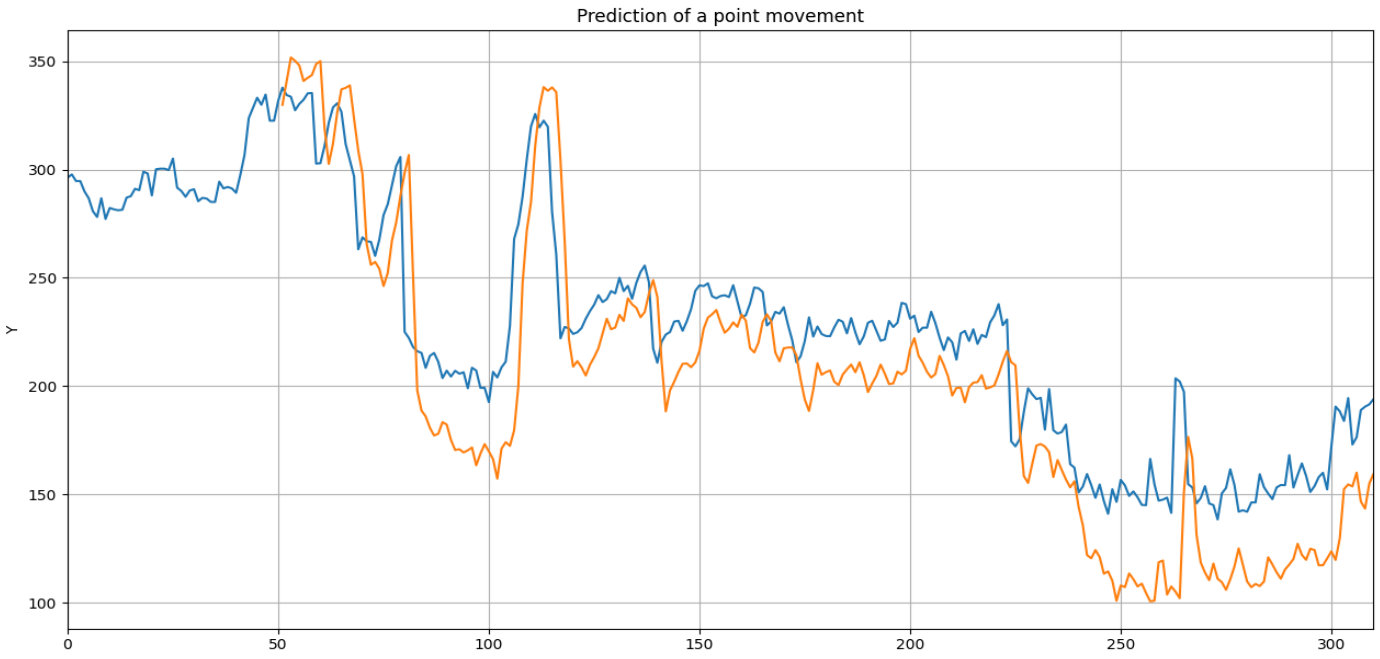
Below it is presented the prediction, where for each step we predict 5 points in advance, using neural network having 100 neurons (for the same dataset).



Below it is presented the prediction, where for each step we predict 1 point in advance, using neural network having 100 neurons (for the same dataset).



Below it is presented the prediction, where for each step we predict 1 point in advance, using neural network having 100 neurons (chosen dataset: prosta_11.txt).



Analyzing presented plots we can claim that the best predictions are made in situation, where for each step (for each iteration of the while loop) we predict 1 point

in advance. The more points we intend to predict in advance, the worse predictions will be made. The neural network is not able to trace well the movement of a point, so results aren't as good as we except. Moreover, the best predictions are prepared when we use a neural network consisting of 100 neurons. If we place e.g. 150 neurons in a network, the plot is starting to "shrink" and got results fall far short of expectations. Similarly the results look like when we place less neurons in a network (e.g. 75-80). Here it is hard to get excellent results, too.

7. Conclusions

During the project we were able to learn methods of creating, training and testing neural networks. Our first results made it seem like the neural network did not learn, but the solution was to predict only a few points in advance (for example 10) based on the previous points (for example 50) from the given dataset and then predict further points after proper increasing the number of points we train our network on. This project isn't as complicated as we supposed before the start of our work, but preparing proper predictions took much time (for one file training a network and making predictions took 4.5 hours, but for another one the time execution of a program was equal 10 hours!). The time execution of a program depends on size of training dataset and number of points to predict (the less points we are going to predict in advance, the we have to wait longer for getting results). Generally, artificial intelligence is pretty interesting and crucial field of science, which affects our life profoundly and has applications in many areas of life (e.g. detecting diseases and predicting number of passengers in airplanes for next 12 months).

8. References

https://stackabuse.com/time-series-prediction-using-lstm-with-pytorch-in-python

Link to GitHub Repository:
https://github.com/sebix354/BIAI-Projekt