POLITECNICO DI TORINO

Master degree course in Electronic Engineering

Master Degree Thesis

Neural network optimization for indoor person localisation using capacitive sensors

Supervisors
prof. Mihai Teodor Lazarescu
prof. Luciano Lavagno

Candidate
Nicoletta Sportillo
ID:242036

Accademic Year 2019-2020

Alla mia famiglia,
in particolare a nonno Tonino, che è
sempre stato un maestro e fonte
d'ispirazione per me e avrebbe gioito
immensamente nel vedermi come sono
diventata e
a nonna Olga, che avrebbe voluto
assistere anche a questo traguardo

Summary

Nowadays one of the main aspects of smart homes and smart cities is human localisation and movement estimation. It can lead to a better quality of life with reduced costs for example by controlling other existent smart technologies as automatic heating and lightning. Moreover, indoor movement monitoring can be really useful in assisted-living for elderly people.

This kind of system must be privacy-aware and easy to use and a tagless passive positioning system can be a good choice. Among the different type of passive sensing, one of the most interesting is the capacitive sensing. This kind of technology has already been used in mobile and wearable devices, but there are few studies on human localisation.

However, the need of cheap hardware and the possibility to analyse the acquired data by a microcontroller make capacitive sensors easy to produce and to prototype.

These sensors are the basis of a long-term research project at DET called "Capacitive sensing for indoor human localisation" within with this thesis work was carried out.

Among the different modes of working of capacitive sensing, the loading mode has been chosen since a single electrode is used. The principle of working is simple: as the person gets closer to the sensor, the capacitive coupling between the body and the sensor increases and by measuring this capacitance, the distance from the sensor can then be inferred.

The experiment has been made in a 3 m x 3 m room using four capacitive sensors. Then an indirect measurement of the capacitance has been performed by measuring the frequency of a 555 timer-based relaxation oscillator. An Arduino board has been used to evaluate the frequency for each sensor, then the measurement has been sent to a base station by a wireless module. Machine Learning classification algorithms have been used for localisation purpose.

Finally, the position of the person within the room has been monitored using two systems.

The first uses four capacitive sensors as in previous experiments, while the second is based on four ultrasound anchors that can localize a mobile tag and is used as accurate reference.

Since the capacitive sensor captures environmental noise, its range and stability over time are limited and the collected data are affected by noise. The reference position and the capacitive sensors readings have been collected concurrently: the former data has been used for training data labelling and inference testing, the latter data has been pre-processed with digital filtering, then neural networks have been used to infer the correct position.

The obtained data set has been used in the thesis work that will focus on different Machine Learning approaches to localise and evaluate the movement of a person in a room.

The whole data set has been divided into three subsets used for the training, validation and testing steps.

For all the simulations TensorFlow and Keras Python libraries have been used, training and validation error curves obtained during the training, validation and test phases have been monitored, to prevent overfitting or underfitting.

Different Neural Networks have been trained and tested to report location estimation as a pair of X/Y coordinates and performance are evaluated in terms of mean squared error (MSE) and Euclidean distance error (EDE).

Three main types of Neural Networks have been analysed:

- Fully Connected Neural Networks: made by a series of fully connected layers, also known as Multilayer Perceptrons (MLP), used as a starting point to better understand the general behaviour of a neural network
- 1D-Convolutional Neural Networks: used to derive features from short segments of the overall data set. This can be useful to analyse time sequences of sensor data and infer the movement from collected data.
- Long Short Time Memory (LSTM): they are recurrent neural network (RNN) architecture able to process sequences of data, and hence potentially useful for movement estimation of a person in a room

Initially, some experiments have been conducted on Neural Networks with only fully connected layers to explore better the main tools used to develop the network and evaluate the results.

Network hyperparameters, such as the learning rate, the number of hidden layers and neurons in the hidden layers have been tuned to improve the network performance. The loss function evaluated during the training and

validation suggested the net was overfitting the data, that means it is difficult to make accurate predictions for new data. This aspect was confirmed in the trend of predicted trajectories: they cannot approximate references curve well.

Since during the design space exploration (DSE) the results with this kind of network have been too high in term of root mean squared error (0.35 m) and Euclidean distance error(0.45 m), the focus moved to the convolutional neural network (CNN) to have a better path recognition.

Among all the types of CNNs, 1D-Convolutional layers have been chosen to build the net, known also as "temporal convolution": the aim was to provide the neural network multiple samples in a time window, to make the network better infer the movement of a person in the room.

Multiple window sizes were considered and hyperparameters tuning has been performed to find a good trade-off between the complexity of the network and performance.

Results have improved over than the previous kind of network, reaching a root mean squared error on the test set of 0.25 m and an Euclidean distance error of 0.29 m. On the other hand, the loss functions suggested that also this network was overfitting the training set.

Since the overfitting can be due to the small data set and a too complex neural network, different tries on really small 1D-CNN have been performed and it has been found that the results, in terms of MSE and EDE, are similar to those achieved with more complex networks.
However, the overfitting remained even if it was attenuated. Indeed, the predicted curves were smoother but still could not match some parts of the reference curves. This result suggests that the problem can be related to the small and noisy data set used for the experiment.

As last step, a recurrent neural network has been explored by using the Long-Short Term Memory network, known to be able to analyse data sequences: they provide a flexible model due to their internal self-connections, so that current predictions can take into account the previous ones.

Design space exploration through hyperparameters tuning has been executed as well, finding good results with a root mean square error on the test of 0.24 m and an Euclidean distance error of 0.30 m. Overfitting remained at a lower level and it was confirmed by the quality of the predicted trajectories.

In general, the results suggest that small networks, in particular the LSTM ones, are good for movement recognition from noisy capacitive sensors readings. The quality of the data set is very important, especially to have sufficient samples for adequate neural network training and a high signal-to-noise ratio.

Future work can be devoted to collect less noisy sensor data to confirm the quality of the networks analysed here, or by trying the sensor fusion approach, which means processing data from several types of localisation sensors to collect more information on position.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are different technologies capable to locate the position of a person (some of them are pressure sensors, ultrasound wave sensors or infrared sensor), among all the possible choices, a capacitive sensing system has been chosen and in this section, the main reasons have been described.

1.1 Capacitive sensing

Capacitive sensing has been chosen to collect data in order to eventually produce a simple and cheap product to localise people in a room.

This kind of technology has been used in the production of devices such as capacitive touchscreen, that can locate the position of the finger in the screen area thanks to the capacitance that is created between the screen and the finger [11].

A capacitance exist between two electrodes separated by some distance, defined as the charge ratio between the change of electrical Q and the change of potential V:

$$C = \frac{Q}{V} \tag{1.1}$$

As explained in [8] human body is conductive, so capacitances exist naturally between people and the environment (Figure 1.1), then to locate or evaluate the proximity of users it is sufficient to measure the changes in capacitive coupling between sensor plane and human body.

There are three ways to sense the nearness of a body with capacitive sensors as explained in [22]: shunt, transmit and load mode (Figure 1.2). For indoor localization, the most interesting is the last one, because of its simplicity and

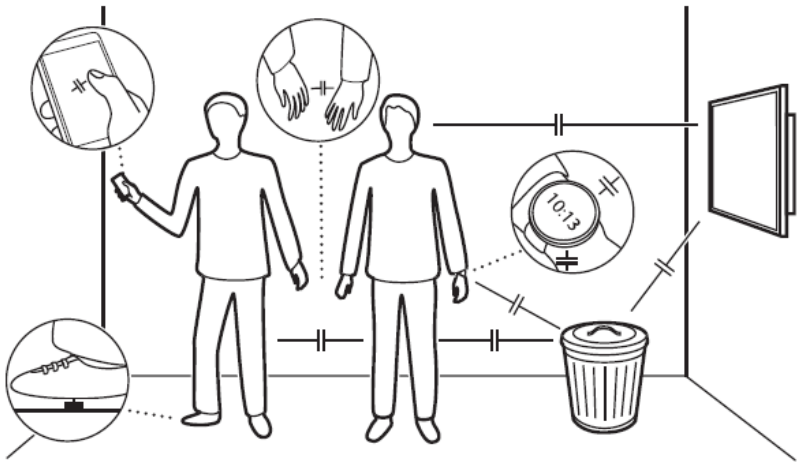


Figure 1.1: Typical capacitances in a room environment [8]

good measurement results. Indeed, the load mode uses one electrode, the human body loads the electrode and the proximity is evaluated by considering that the capacitive coupling increases as the body gets close to the electrode.

Several studies have been conducted on this topic, so limitations and advantages have been found.

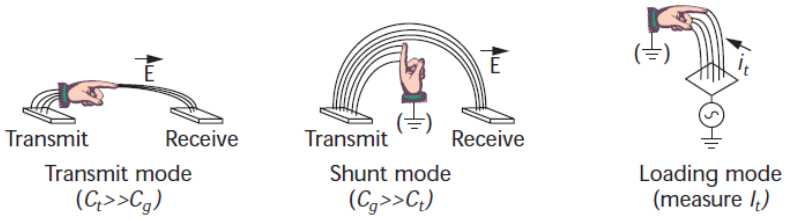


Figure 1.2: Main modes of capacitive sensing [22]

The [8] and [26] discuss some limitations, that consist mostly of the acquisition and quality of data. The main problem is related to the limited tracked range. The empirical model is:

$$C = \frac{\epsilon A}{d^x} \tag{1.2}$$

where x depends on the environment, A is the capacitor plate area and d

is the distance between the sensor and the object to track. Since distance sensing depends on the plate size of the capacitive sensor, distant objects can be tracked by using more capacitive plates so that the object is always close to a sensor [4].

The other important issue is the external influences such as temperature and humidity changes or the presence of conductive objects that should not be tracked but can change the dielectric properties leading to ambiguous or erroneous measurements. The sensitivity to both signal and environmental noise depends especially on A, the effective area of capacitor plates.

On the other hand, capacitive sensors can be used also in the presence of non-conductive objects in the line-of-sight since their influence on dielectric properties is low, so they can be installed in walls. Moreover, the cheap hardware required (electrodes made by different materials and shape) and the possibility to analyse the acquired data by a microcontroller make capacitive sensors easy to produce and to install.

1.2 Previous work

The main project aims to develop a human localisation system to be installed in an indoor environment for the assisted living of people, especially the elderly ones.

As explained in [19] four capacitive sensors has been placed in a 3 m x 3 m room to perform an indirect measurement of the capacitance by measuring the frequency of a 555 timer-based relaxation oscillator. For each sensor the frequency is evaluated on an Arduino board, then the measurement is sent to a base station by a wireless module.

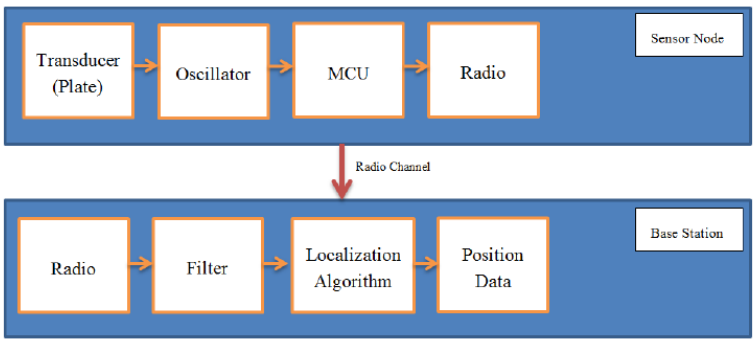


Figure 1.3: Building blocks of the system from [2]

The movement of a person in the room is translated into capacitance variations then converted to frequency variation through the following formula:

$$F = \frac{1}{0.7(R_1 + 2R_2)C} \tag{1.3}$$

Several plate size have been analysed in [19], the resistance values have been chosen to have a good trade-off between sensitivity and sensor size.

Once the frequency measurement has been sent to the base station, post-processing of data and person localisation can be executed. The main building blocks of the system are shown in Figure1.3.

In [2] different kind of Machine Learning classification algorithms taken from Weka collection have been tested on the collected data to understand how it can reduce the effect of the environment on the localisation.

Figure 1.4 shows the setup of the experiment: a person stood in each position, changing the orientation of the body, for more than one samples. Several samples have been collected among the 16 positions.

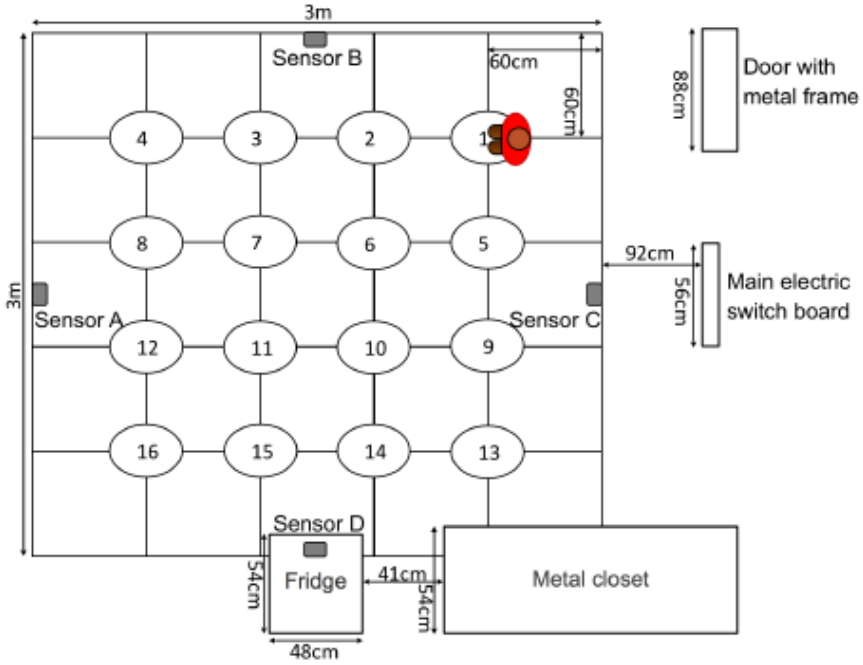


Figure 1.4: Organization of the experiment in [2]

It has been found that LogitBoost and AdaBoostM1 running on top of Random Forest achieved the best results, with an average distance error of 0.06m

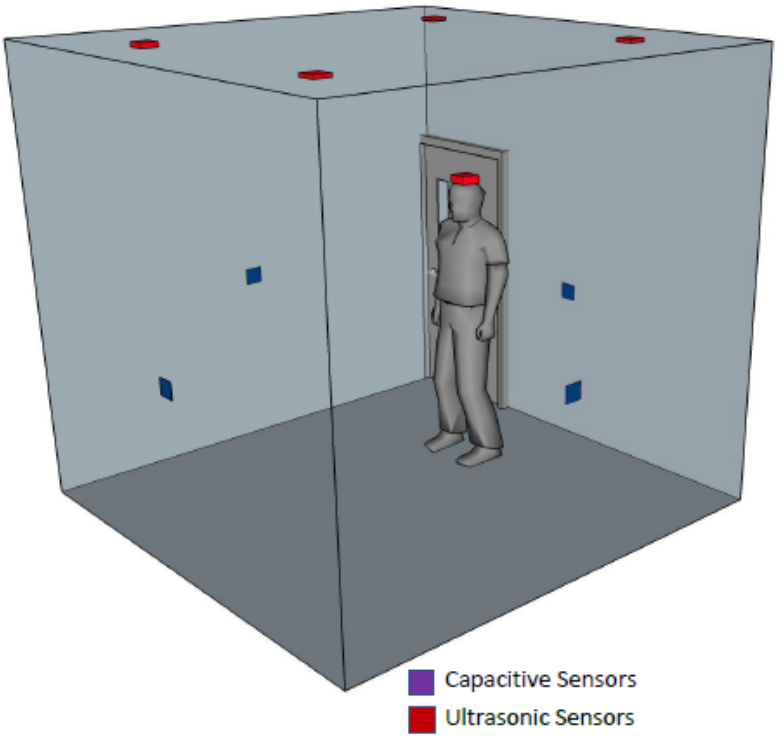


Figure 1.5: Sensor installed in the center of the walls to monitor the movement of a person and anchor system on the roof from [23]

and 0.07m respectively.

The in the most recent experiment the position of the person within the room has been monitored using two systems [23].

The first uses four capacitive sensors installed in the centre of a "wall" of the virtual room, providing readings three times per second.

The other system (from Marvelmind Robotics [12]) is based on four ultrasound anchors that can localize a mobile tag with +/-2 cm accuracy at 15 Hz and has been used as reference for neural network training because of its accuracy. The capacitive sensors are made with plates of 16 cm x 16 cm, similarly to previous experiments [19], [2]. Figure1.5 shows the whole system.

The reference position and the capacitive sensors readings have been collected concurrently to use the former data for training data labelling and inference testing, while the latter data has been pre-processed with digital filtering, then neural networks have been used to infer the correct position.

Since the capacitive sensor front-end is sensitive to environmental noise,

its range and stability over time are limited, so the collected are affected by noise. Data processing includes the following steps [23]:

- translate the average of capacitive sensors reading to zero
- use a Median Filter with a wide window (50s) to extract slow drift
- use a Low-Pass Filter with a pass-band of 0.1 Hz and stop-band of 0.6 Hz to reduce high pitch noise.
- subtract the median filter from the low pass filter output
- normalize the obtained values to (0;1) range

Best values for the window and the cutoff frequency were found empirically. Sensor data tuples are composed of four values, one for each sensor, collected at the same time at 3 Hz frequency. Each tuple is labelled with the corresponding ultrasound sensor reading. The obtained data set has been used in the thesis work.

1.3 Thesis contribution

Starting from the good results obtained in [2] and the work in progress in Dipartimento di Elettronica e TelecomunicazioniI (DET) of Politecnico di Torino [23], the thesis work will focus on different machine learning approaches to handle the noise in the collected data and infer the movement of a person in a room.

Different Neural Networks have been trained and tested to report location estimation and performance have been evaluated in terms of mean squared error (MSE) and Euclidean distance error (EDE). Moreover, to prevent overfitting or underfitting, training and validation error curves obtained during the training, validation and test phases have been monitored.

Three main types of data analysis based on machine learning approaches have been explored:

- Fully connected Neural Networks
- 1D Convolutional Neural Networks
- Long Short Time Memory (LSTM)

The design space exploration (DSE) for all of these structures has been performed to find a good trade-off between the complexity of the network and the MSE.

In Chapter 2, some basic knowledge about Machine Learning and Neural Networks have been discussed, then in Chapter 3 the main step in the design space exploration (DSE) have been explained and some considerations on the all different cases of study has been done in Chapter 4.

Chapter 2

Machine Learning

With experiments conducted in [2], it has been found that machine learning approaches are suitable for the sake of position identification in a room. Indeed they can attenuate sensor data variability and noise due to environmental conditions, obtaining good results in position estimation.

Then, this method has been used in this thesis to estimate the movements of a person in a room. In this chapter, some of the main topics about Machine Learning will be analysed to introduce basic concepts that have been used in the thesis work.

2.1 Some basic knowledge

The term "Machine Learning" has been introduced by Arthur Samuel in 1959 in the article "Some Studies in Machine Learning Using the Game of Checkers"[20]. He described the machine learning as:

> *"the field of study that gives computers the ability to learn without being explicitly programmed."*

A more formal definition has been provided by Tom Mitchell in [13] :

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

Then, it can be said that a machine is learning if it can change its parameters to improve its performance.
As explained in [14] and [21], a machine learning problem can be faced as:

- **Supervised learning**: where inputs x_i and output y_i samples are given to the learning algorithm that has to learn the function f that relates input to output such that $f(x_i) = y_i$. Depending on the type of y_i values it can be distinguished in other two kind of supervised learning problems:
  - *classification* problem: y_i are among a discrete and finite group of values
  - *regression* problem: y_i is a continuous variable

  For historical reasons, the function f is also represented as h and it is called hypothesis [14]. When a new input set is given, the hypothesis h obtained can be used to predict the output values.

- **Unsupervised learning**: only a collection of x_i values is given to the learning algorithm which has to derive the structure from them [14]. This can be done in two ways:
  - clustering approach: used to group the samples in clusters that share some regularities.
  - non-clustering approach: used to find a structure in a chaotic environment.

  No feedback on the prediction results is present

Since supervised learning has been the approach used in the thesis work, more explanations are given in the following, also a simple scheme is reported in Figure 2.1.

In supervised learning there are two main steps:

1. train: a step in which the algorithm that has to learn the hypothesis function that describes the data used. Sometimes also a validation phase can be included during this step.

2. test: a step in which the generalization capability of the algorithm are tested.

In the training step, a set of input and target samples (x_i, y_i), called *training examples* creates a collection called *training data set* that is provided to the algorithm. The performance is evaluated through the cost function

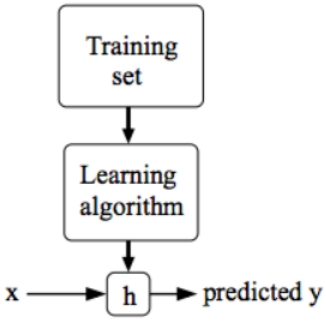


Figure 2.1: Supervised learning scheme from [14]

$J_{train}(\theta)$, that measures the difference between the targets and the outputs of the algorithm [14].

To minimize the cost function, several steps are performed by the algorithm, in which parameters θ are tuned to reach the best result, so when the target and the output values are the nearest. Then parameters values are stored and the training step ends. When the validation is used, also a *validation set*, smaller than the training one, is included in this procedure to identify the minimum of the cost function by using validation error $J_{val}(\theta)$: the model with the lowest validation error is selected.

In the testing step, a *test set* is used, made by a lower number of samples respect to the previous training set. Since generalization is the capability of the algorithm to act well on unknown data, this time targets are not shown to the algorithm: it has to predict the output starting from input data using the parameters saved at the end of the training [14][21].

Also in this step the cost $J_{test}(\theta)$, called "error", is evaluated through the comparison between the computed outputs and the known targets. The test error is usually higher than the training one on which the model is calibrated. Test error can be close to the training one only if the model has reached a good generalization.

It is important to take into account the complexity of the model: it should match the complexity of the function that describes the relation between input and output data to have good performances.

Otherwise, as explained in [14], generalization will be poor and two situation can happen:

- underfitting the data: when model complexity is lower than the actual

function. The error on the training set and test set are both high.

- overfitting the data: when the model is too complex than the actual function. The models fits the samples perfectly but it is not the same for the test set, that results outside the fit curve.

Among all the available machine learning algorithms, neural networks have been chosen to analyse the data collected from capacitive sensors, described in the following section.

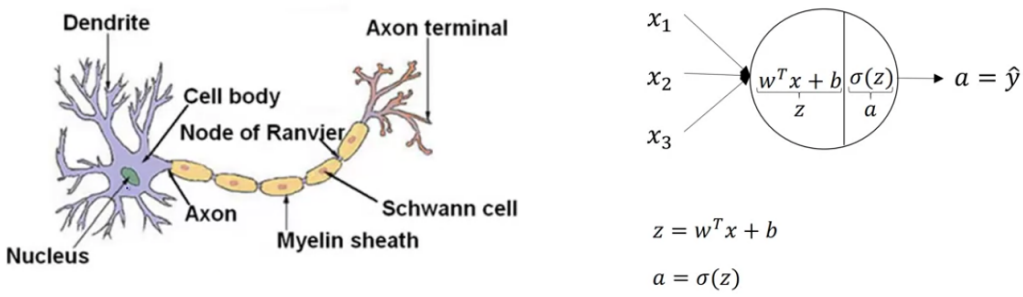
2.2 Neural Networks

Neural networks applications belongs to different fields such as speech recognition, image recognition, and machine translation software [24].

This make them a suitable algorithm to use for the thesis objective: infer the movements of a person in a room for assisted-living sake.

Neural network is a machine learning algorithm based on brain model [16] : to model biological neuron (Figure 2.2a), non-linear elements are used that use weighted sum of the outputs of other elements as their inputs.

The basic element of every Neural Network is a neuron that can be considered computational units that take inputs (dendrites) that are led to outputs (axons). In Figure 2.2b a neuron with three inputs is shown.



(a) Biological neuron from [14]

(b) Neuron model from [15]

Figure 2.2: Real neuron (a) and artificial neuron (b)

The circle is the computational part, made by two main part: the sum between all the weighted inputs $w^T x$ and the bias b and the *activation function*, represented as the sigmoid function of the first part computation, that evaluate the neuron output[15].

The structure presents two layers: the input nodes is the first layer, called *input layer*, the computational node is the second layer, called *output layer*.

In the middle intermediate layers can be added, called *hidden layers*, an example is reported in Figure 2.3. This type of structure is called feedforward neural network.

By combining different neurons a neural network can be implemented.

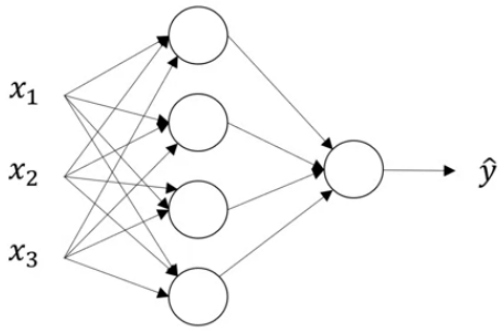


Figure 2.3: Neural network with one hidden layer from [15]

Depending on the purpose of the network to be built, different activation functions can be used, the most common are:

- Sigmoid function: defined as $\sigma = \frac{1}{1+e^{-z}}$ is generally used in the hidden layers
- ReLU function: Rectified linear unit, defined as $a = max(0, z)$ is generally used in the output layers

After creating the network all weights and biases have to be initialized, then they are tuned during the training to optimize network performance. As introduced in Section 2.1, the performance of the network is evaluated through the *cost function* or *loss function*, that shows how much targets and outputs are different, in the best case they result as close as possible, make the loss function reaching the 0.

Optimization algorithm are needed to minimize the cost function. They are used in combination with the backpropagation algorithm that works by computing the gradient of the loss function respect to each weight, computing the gradient one layer at a time, iterating backwards from the last layer to avoid redundant calculations of intermediate terms in the chain rule [15].

One interesting parameter of the optimization algorithm is the *learning rate*, by changing its value the optimization algorithm converges in different

ways: in general its value should be nor too high (otherwise it can overshoot the minimum and fail to converge) nor too small (otherwise it reaches the minimum very slowly) [14].

In the thesis, Mean Square Error is used to estimate the error and two main optimization algorithms have been used: Stochastic Gradient Descent (SGD) and Adamax. Mean Square Error is defined as follows:

$$MSE = \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{N} \tag{2.1}$$

where y is target, $\hat{y}$ is output, N is the number of samples. It can be generalized also to multi-output problems.

As explained in [17] one of the main limitations of traditional forms of feedforward neural network is related to the computational complexity. For example, in coloured image analysis, the number of weights on just a single neuron of the first layer significantly increases.

Indeed, the size of the image (for example, by considering a multidimensional vector, the size can be 64x64 pixels), the number of feature of the image (3 in the case of RGB images) and the computation have to be taken into account: a simple multiplication between these parameters strongly increases the size of the output (output will be 12'228).

This is bad for computational resources and time required, but also for the overfitting problem: the model can easily overfit because of its huge complexity.

2.3 Convolutional Neural Networks

CNN overcome some limitations of the feedforward networks and find numerous applications, such as time series forecasting [3]: they perform well on learning time sequence dependencies, making them suitable for trajectories prediction.

The name of these networks derives from the mathematical operation called convolution, used in their layers [6][25].

In [17] a good explanation of convolutional neural networks is presented by outlining the basic concepts, the layers required to build this kind of network. The convolutional neural network architecture is composed of three types of layers:

- Convolutional layers: it computes the output of neurons by using the scalar product between the input and the kernel, the result is processed

through an activation function and the output feature map is created [10].

- Pooling: it performs downsampling of the given input to reduce the number of parameters and computational complexity.

- Fully-connected layers: it is used in the end for classification. ReLU is used between these layers to improve performance.

A CNN architecture is shown in Figure 2.4.

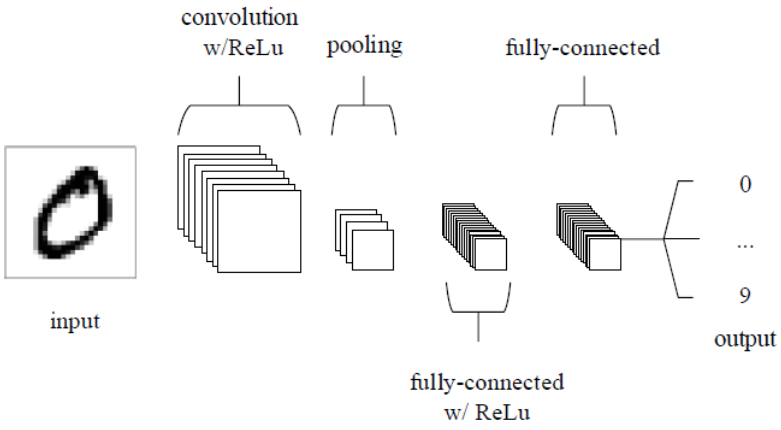


Figure 2.4: Convolutional neural network structure example from [10]

In general, CNNs are trained in a supervised method similar to the fully connected approach explained in Section 2.2. This time, the gradient of each network (then both the convolution and fully-connected layers) parameters are computed. The parameter are updated until a certain stopping criterion is achieved [10].

The number of hidden CNN layers and kernel size per CNN layer can be tuned to obtain the desired structure. These parameters, together with the number of fully connected layers and the choice of activation functions are the main hyperparameters of this type of networks.

In the thesis, 1D convolutional neural networks, also called temporal convolutional neural network, have been used. It is found that they are effective with fixed-length sequences of data. In this kind of CNN, 1D arrays replace 2D matrices for both kernels and feature maps, leading to a low computational complexity CNN [10].

As explained in [6], even if convolution across 1D temporal sequences tries to share parameters across time, this evaluation results superficial since only of a small number of nearby segments of the input is considered in computing the output. A more suitable neural network family that can analyze data over time are Recurrent networks, they will be discussed in the next section.

2.4 Long Short Term Memory (LSTM)

Recurrent neural networks are known for their ability in finding correlations between data, then they are used for tasks as speech recognition [7]. They are also used in time series prediction [18], make them suitable for the main task of this thesis.

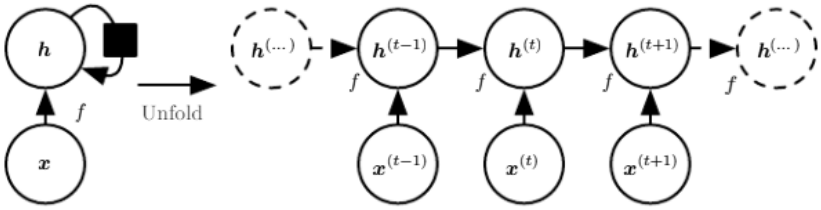


Figure 2.5: Recurrent neural network structure and its unfold version from [6]

In Recurrent Neural Networks (RNN) structure, each output is a function of the previous members of the output and they are computed in the same way. In this way, parameters are shared through a very deep structure

To understand better how RNNs work it is useful to "unfold" the generic structure, as represented in Figure 2.5. With this representation, it can be easily shown that the last state has information about the whole sequence [6].

A complete overview of RNNs types and structures is made in [6], among them, Long Short Term Memory (LSTM) networks are the most used in practical applications. They have been employed in the thesis work and will be discussed in the following.

LSTM networks are a type of recurrent neural networks that have been demonstrated to be particularly useful for learning from sequences of data [9]. They are composed of LSTM cells (Figure 2.6) which use an internally gated self-loop and three gating units that controls the information flow by deciding what should be throw away. All these units have:

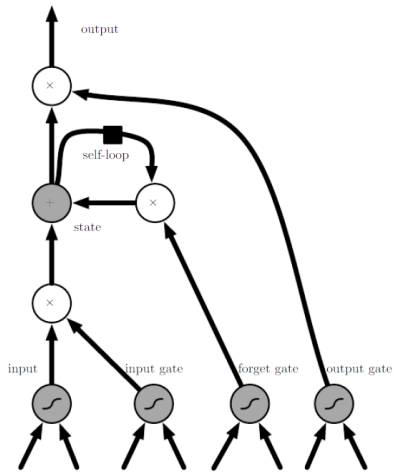


Figure 2.6: LSTM cell from [6]

- input gate: allows the accumulation of the input value into the state depending on its value
- forget gate: controls the weights of the state linear self-loop
- output gate: can extinguish the output of the cell

A more detailed explanation of mathematics that manage this kind of network and some variants can be found in [6].

Also LSTM networks, similarly to CNN and fully connected networks, can be trained in a supervised way, with a training set and, an optimization algorithm with backpropagation.

Chapter 3

Neural network optimization

As explained before in Chapter 1 Section 1.3, the thesis work begins with the data collected and used in [23], some studies about machine learning and neural networks have been examined and finally, three main types of data analysis have been explored:

- Fully connected Neural Networks
- 1D Convolutional Neural Networks
- Long Short Time Memory (LSTM)

The data set used in all the experiments is the same, composed by four values that came from the capacitive samples and two values from the ultrasound sensors, that are related to X and Y position in the room. In total, 1626 labelled samples have been collected and divided into 3 sets to execute a supervised learning:

- Training set: composed of 976 training examples presented to the algorithm in the learning phase
- Validation set: composed of 325 validation samples, used to choose the best model
- Test set: composed of 325 test samples, used in the test phase to evaluate the performance of the chosen model

The training is executed for 1000 epochs and early stopping method has been used to avoid overfitting: the model that performs better on the validation set over the epochs is saved and used in the test phase. During the learning some graphs have been produced:

- the loss function curve of learning and validation, to check the presence of overfitting or underfitting
- the predicted trajectory and the ground truth curves, to visualize the results of the prediction over the actual path

Moreover, some data to evaluate the network have been saved in a comma separated values (CSV) format file:

- best MSE obtained in both the training and testing phase: evaluated as explained in Chapter 2 Section 2.2 with the formula (2.1).
- the best epoch: the epoch in which the model used on the training set is saved
- an estimation of the complexity of the network expressed in "number of parameters"
- the estimation of the distance error as euclidean distance, as the length of the line segment connecting the predicted and the actual points. Then, given the reference trajectory and the predicted trajectory, each described as n ordered points

$$R = (xr_1 ... xr_n; yr_1 ... yr_n), P = (xp_1 ... xp_n; yp_1 ... yp_n) \tag{3.1}$$

  the Euclidean distance between them is the sum of the Euclidean distance between each couple of points that composed the trajectories

$$DE = \sum_{i=1}^{n} \sqrt{(xp_i - xr_i)^2 + (yp_i - yr_i)^2} \tag{3.2}$$

  where n is the number of points that describes the curves

The design space exploration (DSE) of all the networks analysed is discussed in terms of two of the previous parameters:

- *Test MSE* (m^2)
- Neural Network Complexity (*number of parameters*)

TensorFlow has been used to build and train models through the high-level Keras API, which makes getting started with machine learning simple [1]. The very starting point was to learn the usage of TensorFlow and Keras to write the python code and conduct all the experiments.

In the following sections, all the steps of the experiments will be explained and results will be discussed separately.

3.1 Multilayer Perceptron Neural Networks

Generally, fully connected networks made by multiple hidden layers are called Multilayer Perceptron (MLP) Neural Networks. The main steps in MLP networks analysis are:

1. Analysis of a first larger model: the first model was the more complex, some experiment have been conducted to study how it behaves on the data set used. Starting from the results two approaches have been explored:
   - Use a different optimizer: an optimizer with a higher learning rate has been used
   - Increase/decrease the number of neurons: the structure of the network has been changed to check if prediction improves.
2. Analysis on smaller network: a smaller network with just one hidden layer has been tested. Again the number of neurons in the hidden layer has been changed to check if prediction improves.

It is known that the performances of the network depends also on weights and biases, that are randomly generated at the beginning of the training. Since this can lead to different solutions for the same network, each experiment has been executed for 10 times, to be sure that results achieved on one execution aren't sporadic.

3.1.1 Analysis of a larger model

The first MLP network described and analysed was composed of an input layer with 4 input neurons, each corresponding to one capacitive sensor reading, five hidden layers with 64 neurons per layer and an output layer that computes two values (Figure 3.1). The NN model is sequential, described layer by layer in Keras. The optimizer Adamax has been used to train the

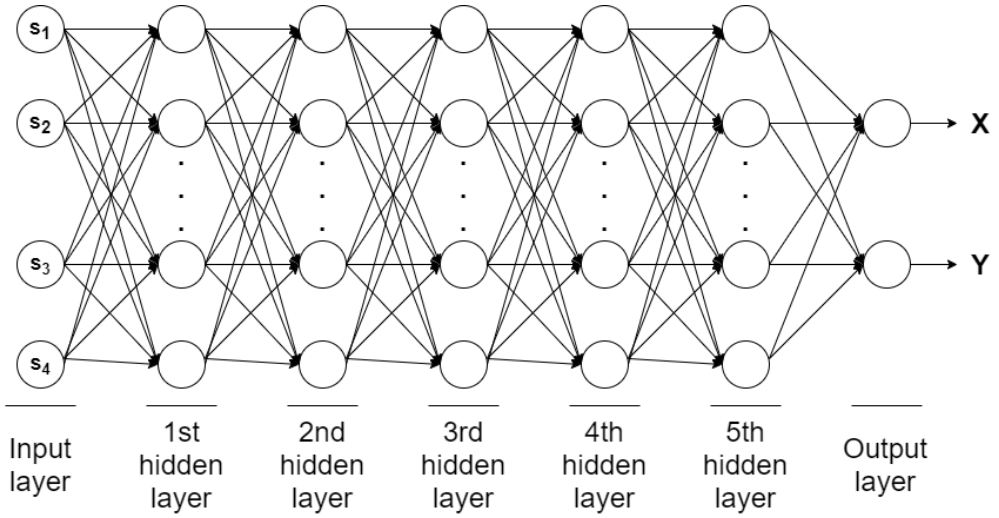


Figure 3.1: MLP network structure

network. More information about models and optimizers can be found in Keras Documentation [5].

The input layer receives four values from the data set and the output layer computes the estimated X-axis and Y-axis values corresponding to human presence in the room.

In the Table 3.1 the main results are reported. It can be noticed that the distance error of ~0.44m can be too high for assisted-living. Moreover, neural network learning seems to stop too early since the best epoch is the 9th out of the 1000 available.

Neurons per hidden layers	Best Epoch	Test MSE (m^2)	Distance Error (m)	Complexity (number of parameters)
64	9	0.126	0.443	17'090

Table 3.1: Results of MLP with 5 hidden layers with 64 neurons per layer

The resulting loss functions for training and validation set are shown in Figure 3.2.

When the training and validation errors steadily decrease, the NN learns useful information about the problem and it is able to generalize it. The training should stop when the validation error starts to increase, which is an indication of NN is overfitting the training data and loosing generalization capability. It can be easily seen that there is a generalization problem: while

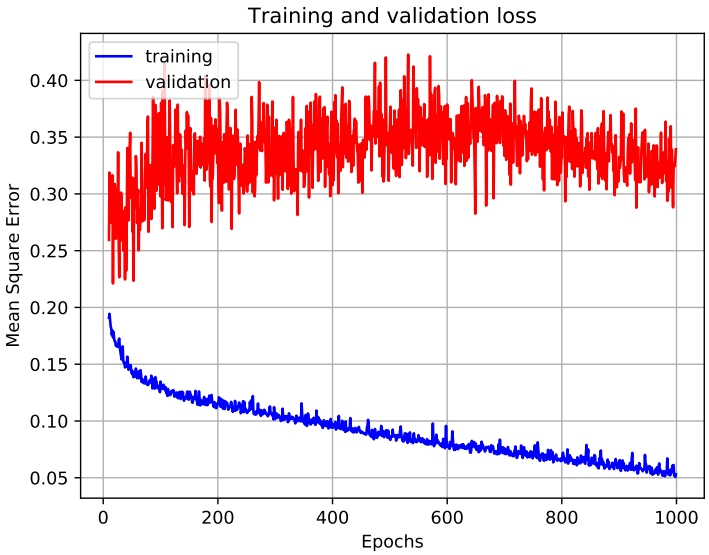
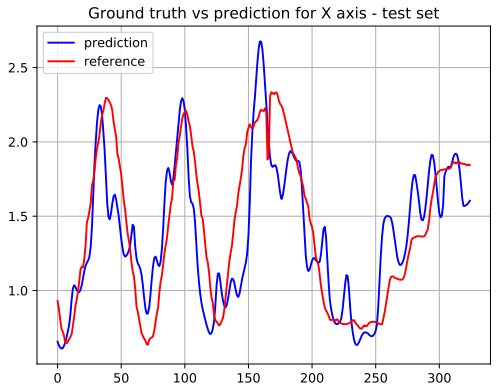
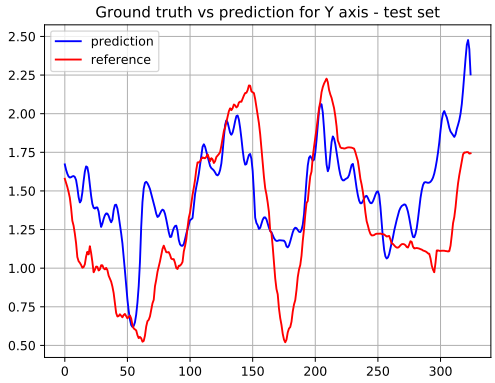


Figure 3.2: Loss function MLP with 5 hidden layers with 64 neurons per layer



(a) X trajectory and X ground truth



(b) Y trajectory and Y ground truth

Figure 3.3: Test trajectory and ground truth of 5 hidden layers with 64 neurons per layer

the training loss decreases over the epochs, the validation loss curve does not follow the train loss curve, on the contrary, it is noisy and it strays very soon from the expected behaviour. This means that the model is learning from the training set but it is not able to generalize well on the validation set.

The bad generalization on the training set is confirmed in the testing phase by the graphs in Figure 3.3 where the X and Y axes prediction is shown.

It is clear that the predictions are not sufficiently accurate: too many

spikes are present over a smoother reference curve.

Then, the model has been changed: SGD optimizer has been used that by default provides a learning rate higher than the Adamax.

Obtained results are reported in the following.

Neurons per hidden layers	Best Epoch	Test MSE (m^2)	Distance Error (m)	Complexity (number of parameters)
64	122	0.125	0.451	17'090

Table 3.2: Results of MLP with 5 hidden layers with 64 neurons per layer and SGD optimizer

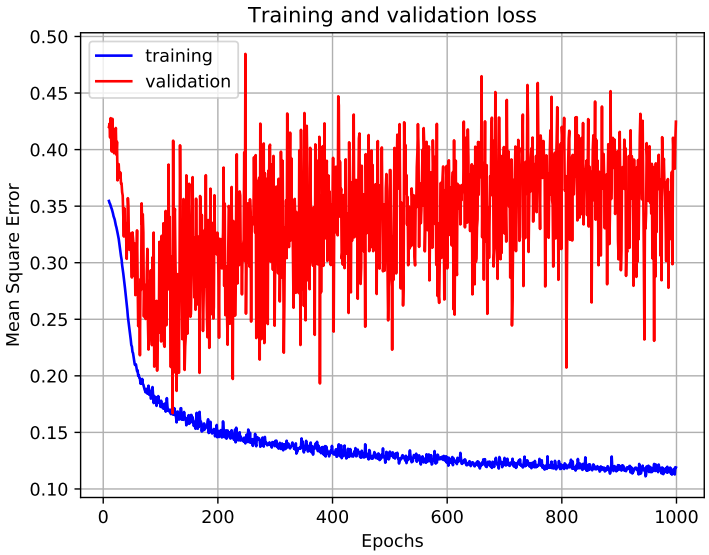
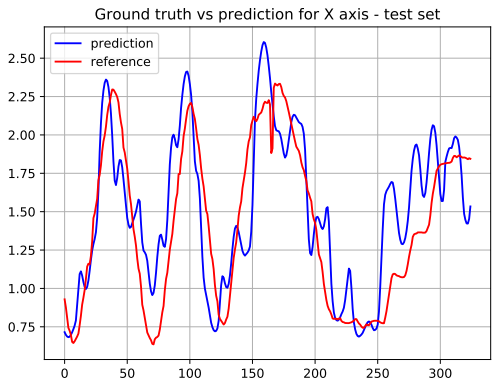


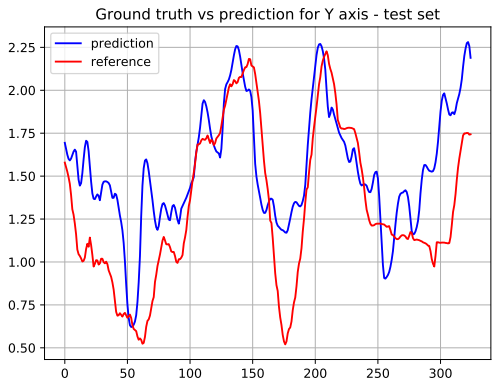
Figure 3.4: Loss function MLP with 5 hidden layers with 64 neurons per layer and SGD optimizer

Since the optimizer just describes the learning method network, its change does not affect the number of parameters. However, results are close to the previous ones, as can be seen from Table 3.2. The loss function has a better behaviour in the beginning but it has a worse behaviour in the following as it can be seen in Figure 3.4.

In Figure 3.5 it is shown that also the prediction curves are similar to the previous experiment. Therefore, it can be said that this type of changes



(a) X trajectory and X ground truth



(b) Y trajectory and Y ground truth

Figure 3.5: Test trajectory and ground truth of 5 hidden layers with 64 neurons per layer and SGD optimizer

is not very effective to reach better performances. This could be due to the higher learning rate of the optimizer used, which results probably too high.

Then, the optimizer has been changed again to the Adamax and the focus moved on changes concerning the network structure, starting from the number of neurons per layer.

The next structure used is the same as Figure 3.1 save for the number of neurons that has been increased to 128. In the Table 3.3 the main values are reported. Results on MSE and distance error are worse than the previous ones.

Neurons per hidden layers	Best Epoch	Test MSE (m^2)	Distance Error (m)	Complexity (number of parameters)
128	24	0.184	0.523	66'946

Table 3.3: Results of MLP with 5 hidden layers with 128 neurons per layer

Loss functions (Figure 3.6) seem to have a comparable behaviour to the one in Figure 3.2 and the problem of bad generalization is not overcome with these changes. Indeed, predicted trajectories, shown in Figure 3.7, confirm the bad generalization.

Since increasing the number of neurons did not lead to better results, other two network configurations with a lower number of neurons per layers have been tried: one with 32 neurons per hidden layer and the other with 8

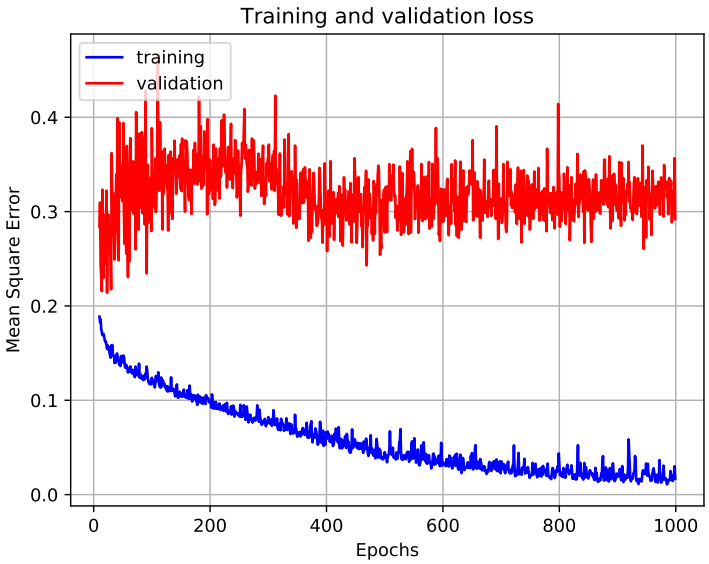
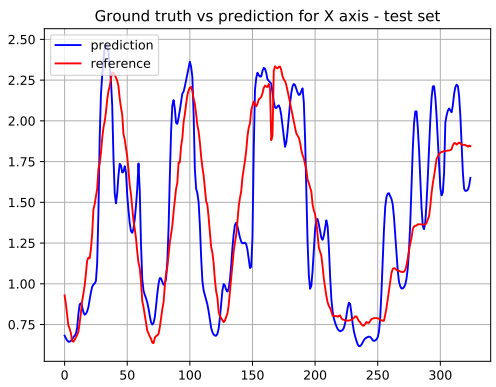
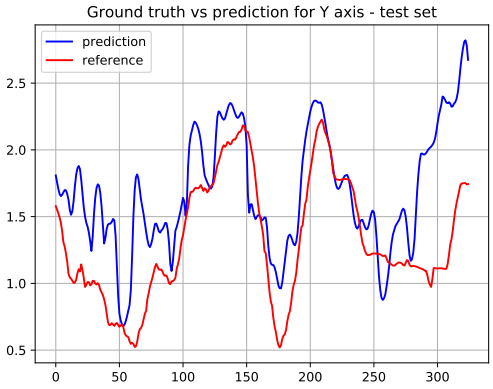


Figure 3.6: Loss function MLP with 5 hidden layers with 128 neurons per layer



(a) x trajectory and x ground truth



(b) y trajectory and y ground truth

Figure 3.7: Test trajectory and ground truth of 5 hidden layers with 128 neurons per layer

neurons per layer.

Results for both of them are reported in the Table 3.4.

The distance errors are close to the results obtained in the network with 64 neurons per layer. This means that performances comparable to complex networks can be reached with a lower complexity network.

However, the problem of overfitting is always present, as it can easily seen

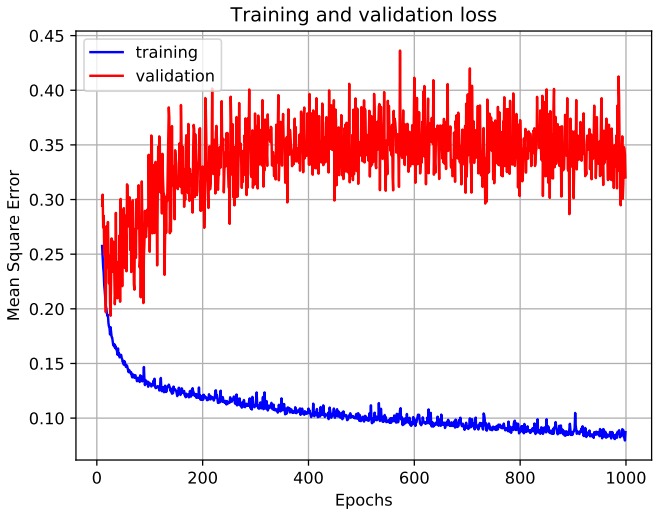
Neurons per hidden layers	Best Epoch	Test MSE (m^2)	Distance Error (m)	Complexity (number of parameters)
32	27	0.135	0.446	4'450
8	120	0.129	0.435	346

Table 3.4: Results of MLP with 5 hidden layers with 32 and 8 neurons per layer

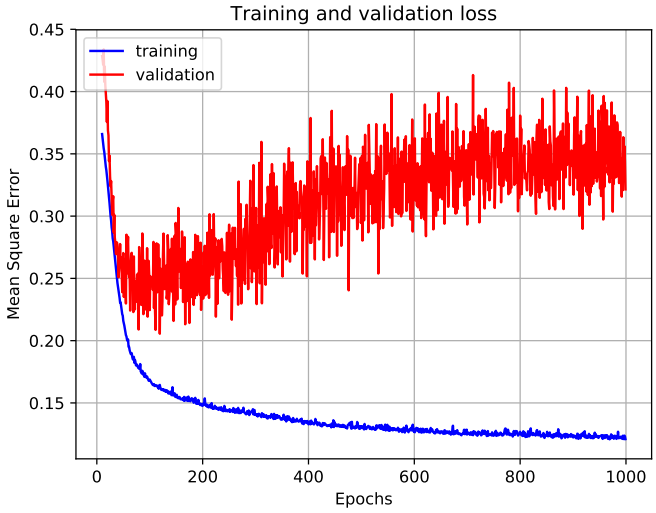
in Figure 3.8: the validation loss of the network with 8 neurons per layer has a slightly better trend respect to others, but still not sufficiently good to avoid bad generalization.

Again the problem of generalization is confirmed by the prediction trajectory curves: in both of the networks, the accuracy in the prediction is not sufficient since the prediction curves do not approximate well the reference values (Figure 3.9).

From these simulations resulted that less complex network performances are similar to more complex ones.

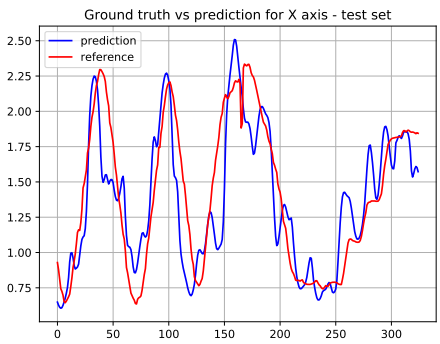


(a) Loss functions for network with 32 neurons per layer

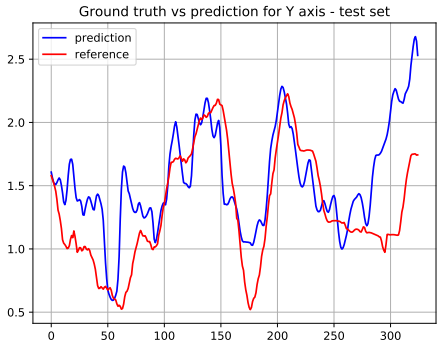


(b) Loss functions for network with 8 neurons per layer

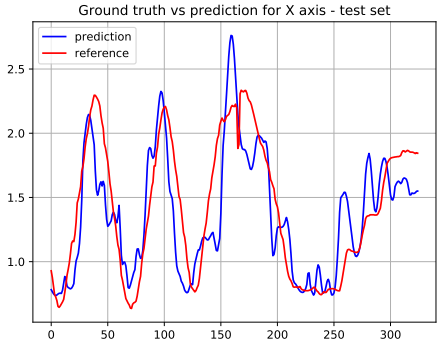
Figure 3.8: Loss functions of 5 hidden layers with 32 and eight neurons per layer



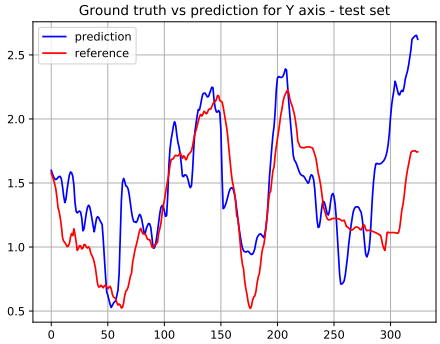
(a) X trajectory and X ground truth for network (32 neurons)



(b) Y trajectory and Y ground truth for network (32 neurons)



(c) X trajectory and X ground truth for network (8 neurons)



(d) Y trajectory and Y ground truth for network (8 neurons)

Figure 3.9: Test trajectory and ground truth of 5 hidden layers with 32 and 8 neurons per layer

3.1.2 Analysis on smaller network

The model has been changed again: just one hidden layer has been used and different simulations with different number of neurons have been executed. The results are summarized in the Table 3.5.

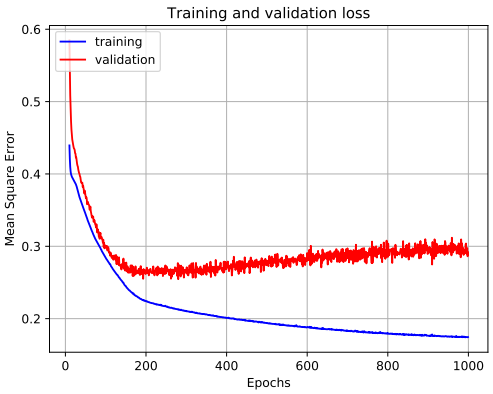
Neurons per hidden layers	Best Epoch	Test MSE (m^2)	Distance Error (m)	Complexity (number of parameters)
8	281	0.152	0.485	58
32	***196***	***0.125***	***0.433***	***226***
64	199	0.132	0.447	450
128	210	0.136	0.457	898

Table 3.5: Results of MLP with 1 hidden layers

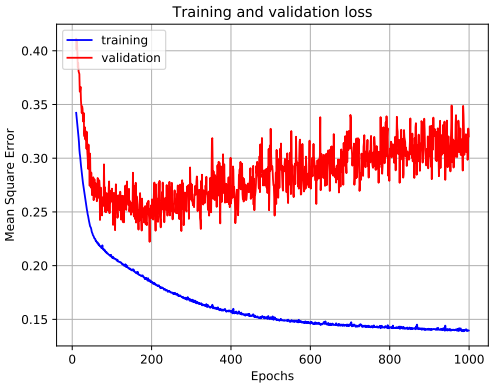
It can be easily seen that the best result in terms of distance error is obtained with an MLP network with 32 neurons in the hidden layer. This result is really close to the one obtained in the beginning with a more complex network which results are reported in Table 3.1.

Additionally, loss functions shown in Figure 3.10 present a behaviour close to the previous simulations: the validation loss function diverge from the training loss function soon, the generalization is poor.

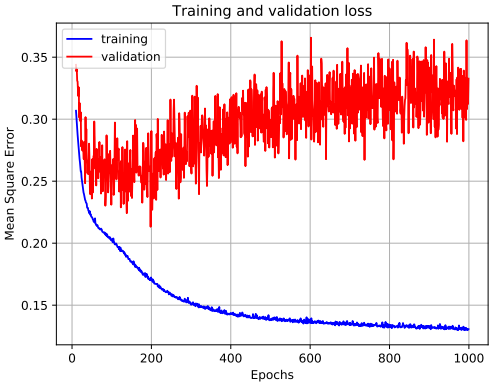
Trajectory predictions show similar behaviour between the different network analyzed, so they are not reported here.



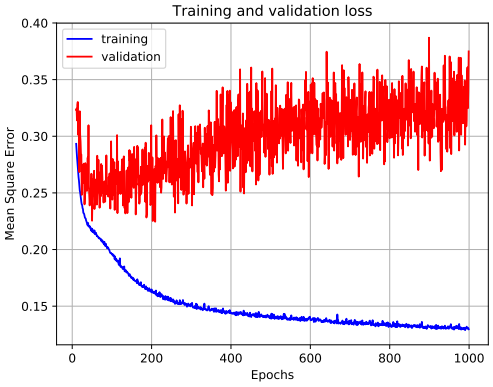
(a) Loss function NN 8 neurons



(b) Loss function NN 32 neurons



(c) Loss function for network with 64 neurons



(d) Loss function for network with 128 neurons

Figure 3.10: Loss functions of 1 hidden layers networks

3.1.3 MLP Summary

To summarise, the main results of this kind of network has been reported in Table 3.6. The design space exploration plot has been shown in Figure 3.11 as the MSE vs number of parameters plot.

Optimizer	Hidden layers	Neurons per hidden layers	Best Epoch	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
Adamax	5	64	9	0.126	0.443	17090
SGD	***5***	***64***	***122***	***0.125***	0.***451***	***17090***
Adamax	5	128	24	0.184	0.523	66946
Adamax	5	32	27	0.135	0.446	4450
Adamax	5	8	120	0.129	0.435	346
Adamax	1	8	281	0.152	0.485	58
Adamax	***1***	***32***	***193***	***0.125***	***0.433***	***226***
Adamax	1	64	199	0.132	0.447	450
Adamax	1	128	210	0.136	0.457	898

Table 3.6: Summary table of MLP networks analyzed

Then, it can be finally said that:

- this kind of network can not produce better results starting from this data set
- in general, simple networks can produce results similar to those obtained with complex network.

Another approach to analyze the data for movements recognition is to use data collected in a certain window of time so that a correlation between data can be found to infer movements. This method has been exploited by using convolutional neural networks, in particular 1D-CNN, that will be discussed in the next Section.

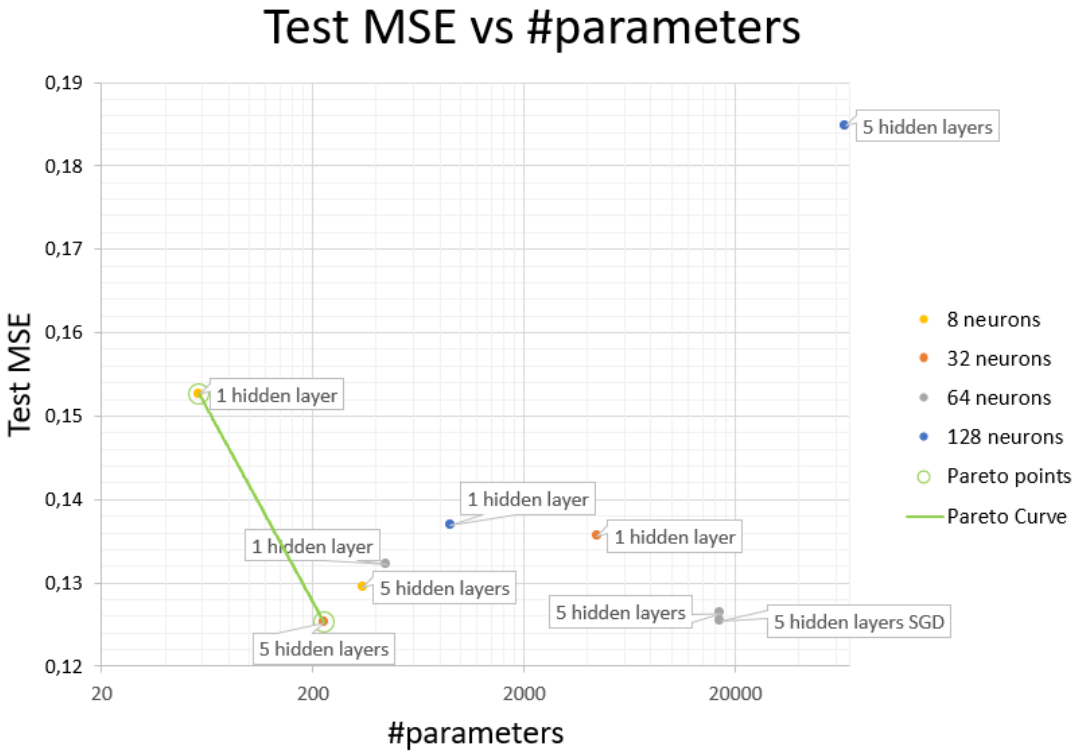


Figure 3.11: Summary of the MLP networks analyzed

3.2 Convolutional Neural Networks

The more suitable convolutional approach for this case of study consists of the usage of the 1D convolutional layers. They are known for their effectiveness on sequences of data analysis, such as audio processing.

The data set has been divided in sequences: after choosing the size of the temporal window to be considered, the CNN receives all the tuples that belong to the temporal window, which are labelled with the position coordinates of the window size middle.

The objective is to analyze these tuples in sequences and infer the position coordinates. Because of the position refers to the middle of a window, the predicted trajectories starts half a window after the beginning of reference trajectory and half a window before its end.

The process is computed by the filters and the kernel that are defined in the CNN: the kernel moves inside the window by one tuple at a time, filters analyze the content of the kernel. When the edge of the window is reached, the window "moves": the very first tuple is discarded and a new tuple is taken into account. The new window obtained is analyzed in the same way described before.

Initially, the window size used was set to 5 s, but later also some experiments with a window size of 10 s have been conducted. All the networks explored have been trained for 1000 epochs. Again, each experiment has been executed for 10 times, to be sure that results achieved on one execution are not sporadic.

The design space exploration on 1D-CNNs can be divided into three main parts:

1. Larger 1D-CNN analysis: the starting point was a large 1D-CNN suggested by the research team. Experiments to perform the DSE have been conducted and results analysed. Main steps are:
    - the study of the behaviour of the starting network
    - increasing/decreasing number of CNN filters and kernel size to check if prediction improves
    - decreasing the number of neurons in the MLP layers: an optimization on the last part of the network has been conducted to check if prediction improves
    - reduction of CNN layers: it has been chosen to reduce the complexity of the network. Only two CNN layers have been used in this stage.

2. Exploration on regularization: L1 and L2 regularizations have been added to check improvements on performances

3. Smaller networks analysis: a network with one convolutional layer and one MLP layer has been used to check if prediction improves.

3.2.1 Larger Network results

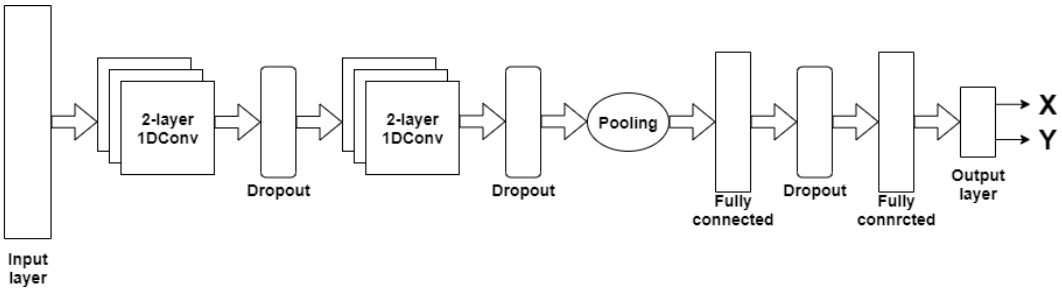


Figure 3.12: Larger 1D-CNN structure

The first convolutional neural network described and analysed has been suggested by the research team, it is composed of two convolutional hidden stages made by two convolutional layers each followed by a dropout layer of 0.5, a pooling layer and two fully connected hidden layers of 64 neurons each, separated by a dropout layer (Figure 3.12).

Dropout is used for regularization to prevent overfitting. It randomly sets a percentage of input data to zero: in this network, since the dropout is set to 0.5, half of the inputs of the second convolutional stage are set to zero.

The pooling size is needed in every convolutional network to reduce the number of parameters. While the last part of MLP networks is used to compute the final outputs.

A complete design space exploration has been conducted by changing the number of filters and kernel size. The network has been trained over 1000 epochs for 10 times with a window size of 5 s, the results are stored in the Table 3.7 and Table 3.8.

Results are better than those obtained with the simple MLP, in part because the CNN analyzes sequences of data. On the other hand, the complexity of the network has increased. Best results have been achieved with a kernel size of 3 and the number of filters of 8 (highlighted in Table 3.7).

Given these good results, also a better behaviour of the loss function is expected. Every loss function obtained for this network have been analysed

	Number of filters					
	8			16		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	***0.064***	***0.297***	***7618***	0.078	0.354	12034
5	0.079	0.344	8066	0.079	0.341	13698
7	0.100	0.393	8514	0.093	0.369	15362
9	0.098	0.367	8962	0.069	0.323	17026

Table 3.7: Results of 1D-CNN 8 and 16 filters

	Number of filters					
	32			64		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.091	0.372	24322	0.084	0.358	62722
5	0.094	0.371	30722	0.090	0.374	87810
7	0.081	0.346	37122	0.083	0.361	112898
9	0.085	0.358	43522	0.091	0.367	137986

Table 3.8: Results of 1D-CNN 32 and 64 filters

and the general trend of the iteration that obtained the reported results has been confirmed.

In Figure 3.13 the best loss function is represented. It can be easily seen that the problem of the bad generalization is still present. On the other hand, the validation curve seems to follow the training loss trend in a better way, compared to the one obtained in the MLP experiments, hence the NN is able to generalize better.

Then, the trajectory prediction curves have been observed, they are reported in Figure 3.14.

In general, the prediction follows the reference more smoothly with less spikes than those obtained in the MLP experiments. However, some parts of the prediction curves do not follow closely the reference ones.

It seems that on the X-axis there is an underestimation of peaks, while in Y-axis several incongruences are present in the initial and final parts.

Some changes in the convolutional stages have been executed to check

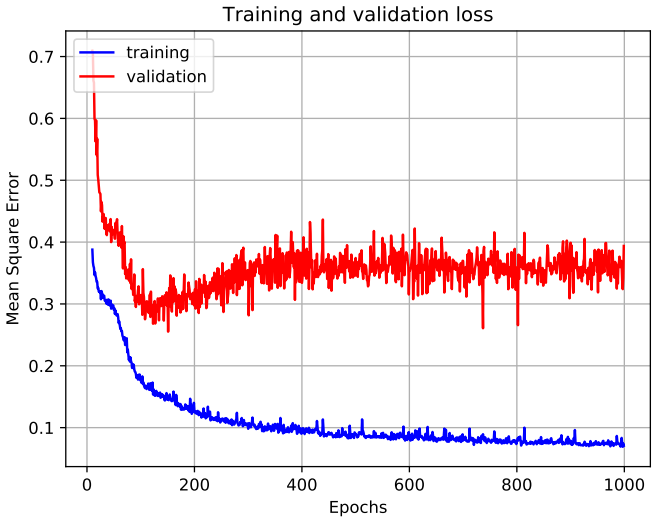
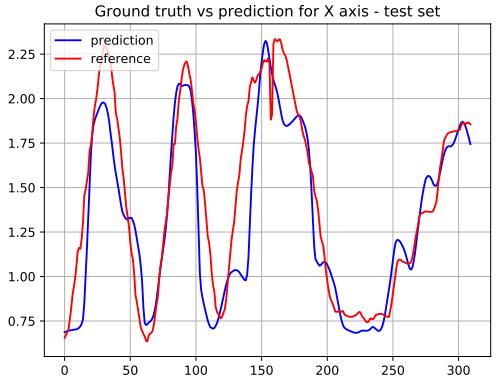
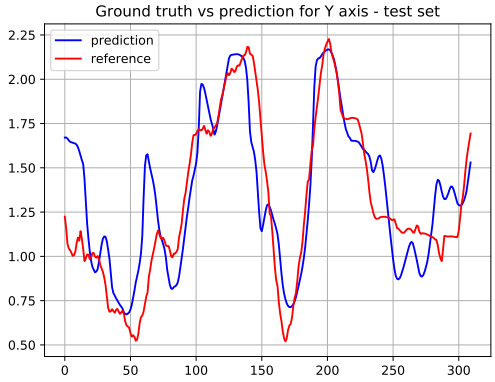


Figure 3.13: Loss function 1D-CNN with 8 filters and kernel size of 3



(a) x trajectory and x ground truth



(b) y trajectory and y ground truth

Figure 3.14: Test trajectory and ground truth of 1D-CNN with 8 filters and kernel size of 3

if prediction improves. New experiments have been conducted by changing the number of filters and kernel size for each layer, but results were not significantly better, so they have been discarded and are not discussed here.

Then, it has been decided to optimize the structure of the network. The number of neurons in the first MLP layer of the network final stage has been reduced from 64 to 32. The complete design space exploration has been executed with different kernel sizes and number of filters. First results with a kernel size of 9 were not promising, hence it has been excluded from the

experiment.

The obtained results have been reported in the Table 3.9 and Table 3.10.

	Number of filters					
	8			16		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	***0.073***	***0.322***	***4258***	0.078	0.340	7394
5	0.098	0.370	4706	0.103	0.399	9058
7	0.087	0.351	5154	0.092	0.371	10722

Table 3.9: Results on 1D-CNN with 32 neurons in the first MLP - results with 8 and 16 filters

	Number of filters					
	32			64		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.094	0.379	17122	0.095	0.384	50402
5	0.094	0.382	23522	0.089	0.365	75490
7	0.090	0.375	29922	0.110	0.407	100578

Table 3.10: Results on 1D-CNN with 32 neurons in the first MLP - results with 32 and 64 filters

Again, the best results have been obtained with the kernel size of 3 and the number of filters of 8, highlighted in Table 3.9.

Since results in terms of Test MSE are a bit worse than the previous try, it is expected that both loss function and predicted trajectories are similar or worse. All the loss functions of the network with the best results have been analysed, the behaviour does not seem sporadic. The loss function is reported in Figure 3.15. The validation loss curve stops following the training loss trend earlier than the previous experiment in Figure 3.13.

They are similar to the previous experiment, with a slightly worse behaviour especially in the end part, between samples 250 and 300. The predicted trajectories are shown in Figure 3.16.

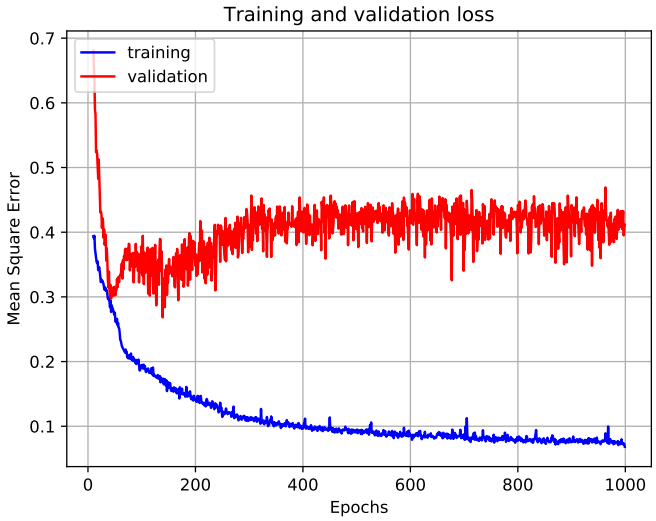
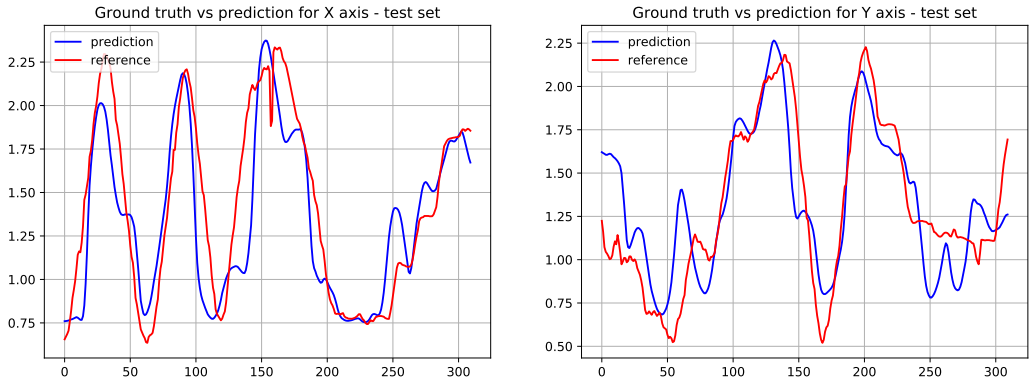


Figure 3.15: Loss function 1D-CNN with 32 neurons in the first MLP, 8 filters and kernel size of 3



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.16: Test trajectory and ground truth of 1D-CNN with 32 neurons in the first MLP, 8 filters and kernel size of 3

Again, a reduction of the network complexity obtains slightly worse results in prediction, as in the case of MLP networks analysed in Section 3.1.

In the next experiment also the last MLP layer was set with 32 neurons, only kernel size of 3 and 5 have been considered and the number of filters has been set to 4 and 8 since best results have been obtained with a low number of filters. Results are reported in Table 3.11.

	Number of filters					
	4			8		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.092	0.374	2002	***0.083***	***0.350***	***3138***
5	0.088	0.370	2130	0.088	0.363	3586

Table 3.11: Test trajectory and ground truth of 1D-CNN with 32 neurons in the first MLP

Results are worse than previous experiment, best results are highlighted. Loss functions presents a smoother trend, but the validation loss always seems to flat around 0.4 m^2 MSE (Figure 3.17).

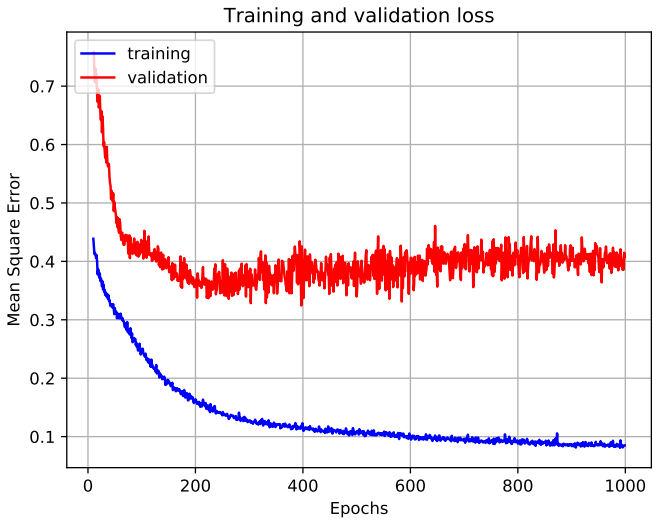
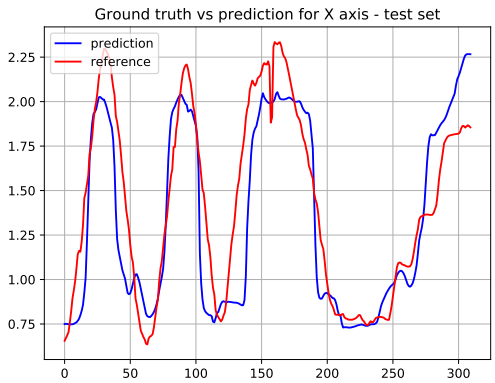


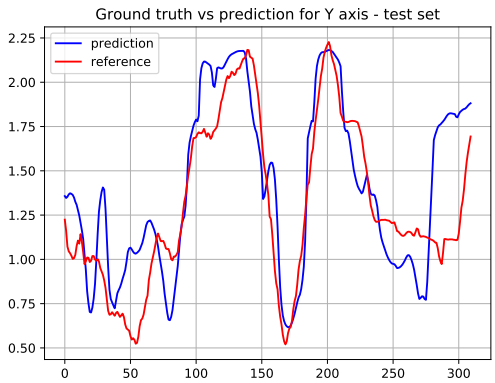
Figure 3.17: Loss function 1D-CNN with 32 neurons in MLP layers, 8 filters and kernel size of 3

While predicted trajectories do not show any improvement, on the contrary, Y-axis prediction is worsened in the initial and final parts and X-axis prediction underestimates peaks as noticed in the very first experiment (Figure 3.14). Prediction curves are shown in Figure 3.18.

Then, the next experiment is focused on the reduction of complexity: the second two 1D-CNN hidden layers and dropout have been removed and the number of neurons of MLP layers have been set to 32. Again, only kernel size



(a) x trajectory and x ground truth



(b) y trajectory and y ground truth

Figure 3.18: Test trajectory and ground truth of 1D-CNN with 32 neurons in MLP layers, 8 filters and kernel size of 3

of 3 and 5 have been considered and the number of filters has been set to 8. The network has been trained over 1000 epochs for 10 times and a window size of 5 s, as usual. Results are shown in Table 3.12. Good results have been

	8 filters		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	***0.070***	***0.329***	***2738***
5	0.090	0.362	2930

Table 3.12: Results of 1D-CNN with 32 neurons in the first MLP and a window size of 5 s

obtained for the network with a kernel size of 3, they are slightly worse than hose highlighted in Table 3.7 of the first network analyzed, but it seems a good trade-off between number of parameters required and Test MSE.

The loss function is shown in Figure 3.19. The validation loss function follows the trend of the training loss function for a short time, then it stays around the 0.3 m^2 MSE: this means that the validation loss function is closest one obtained until now, but the generalization is poor.

The prediction curves (Figure 3.20) present some spikes, mostly in the initial and last part of the Y-axis prediction.

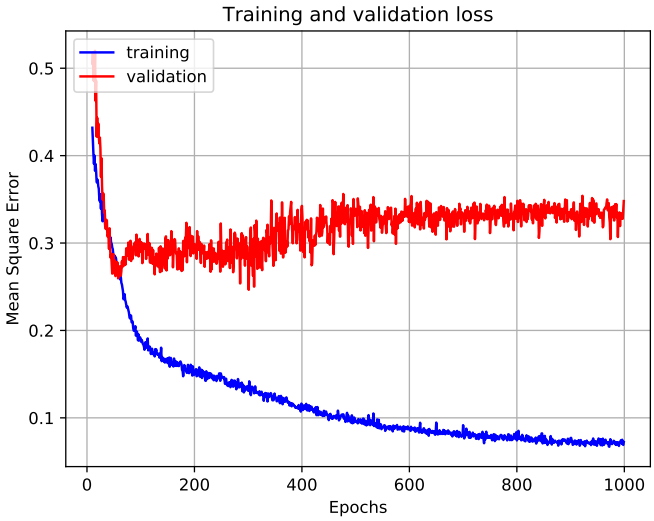
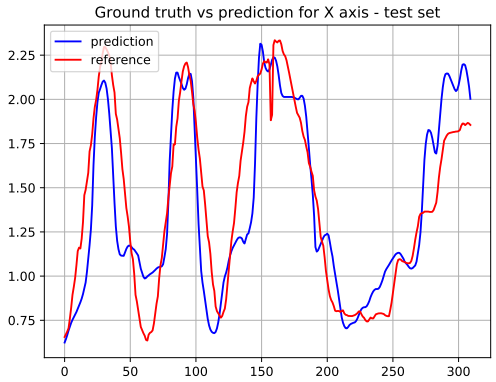
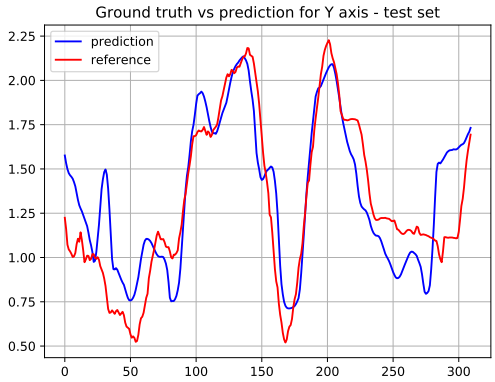


Figure 3.19: Loss functions 2-layers 1D CNN and 32 neurons in MLP layers



(a) x trajectory and x ground truth



(b) y trajectory and y ground truth

Figure 3.20: Test trajectory and ground truth of 2-layers 1D-CNN with 32 neurons in MLP layers, 8 filters and kernel size of 3

A way to reduce the overfitting can be the usage of other kinds of regularization. Starting from this last result, L1 and L2 regularizations have been added to the convolutional layers to check if prediction improves. These experiments are discussed in Subsection 3.2.2.

3.2.2 Exploration on regularization

The network structure is shown in the Figure 3.21.

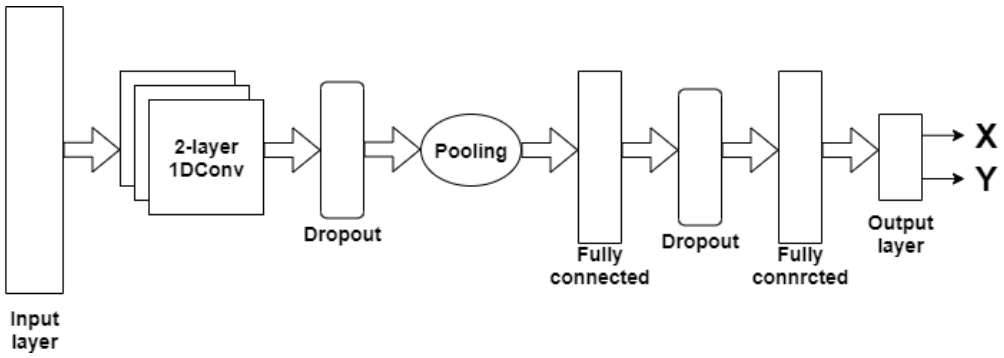


Figure 3.21: 1D-CNN structure used for regularization exploration

The regularization is a way to prevent overfitting or reduce the error in the network, some basic information about different types of regularizations can be found in [6]. Two kinds of regularizations have been applied to the experiments yet: dropout (described in Subsection 3.2.1) and early stopping (described in Section 3).

By reading about 1D convolutional layers on [5], it has been found that L1 or L2 regularization are provided, then also these solutions have been explored. The main difference between these two kinds of regularization is in the way they act on the model to reduce computation complexity, a detailed explanation can be found in [6]. Several experiments have been conducted:

- Adding L1 regularization to the structure with window size to 5 s
- Adding L2 regularization to the structure with window size to 5 s
- Adding L1 regularization to the structure without the dropout layer, with window size to 5 s
- Adding L2 regularization to the structure without the dropout layer, with window size to 5 s
- Increasing the window size to 10 s in all the previous listed experiments

Increasing the window size can lead to an improvement on predictions: since more data are analysed inside the window, a better correlation between data can be implemented.

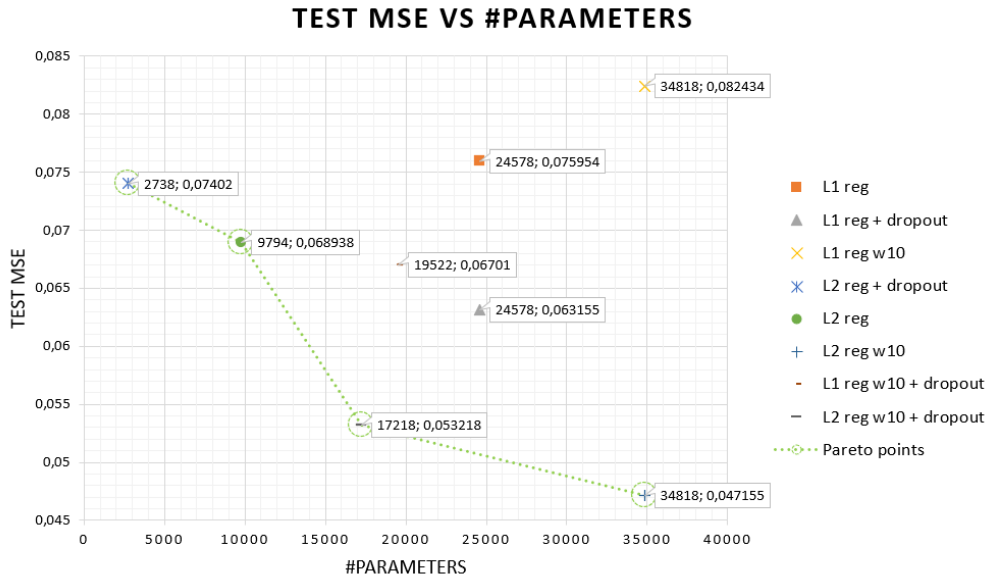


Figure 3.22: Main results of regularization exploration

Since controlling the model complexity is not related to only find the right number of parameters, it is also possible that a large model is the best one if well regularized [6]. Then, a complete design space exploration has been conducted with different kernel sizes and number of filters.
The best results obtained are shown in the Figure 3.22. It can be noticed that L2 regularization Pareto dominate the L1 one. In the following, the highlighted Pareto points will be discussed, starting from the left side of the graph.

The first results reported in Table 3.13, are those of L2 regularization with the dropout layer. The kernel size and the number of filters of the best

	8 filters		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.074	0.333	2738

Table 3.13: L2 regularization results with dropout - window size of 5 s

result obtained with L2 regularization are the same as the best results of the networks analysed in Subsection 3.2.1. The MSE is worse than that obtained

for this network structure without regularization. To have a better idea of the prediction behaviour, also loss functions and prediction trajectories will be analysed. They are shown in Figure 3.23 and Figure 3.24.

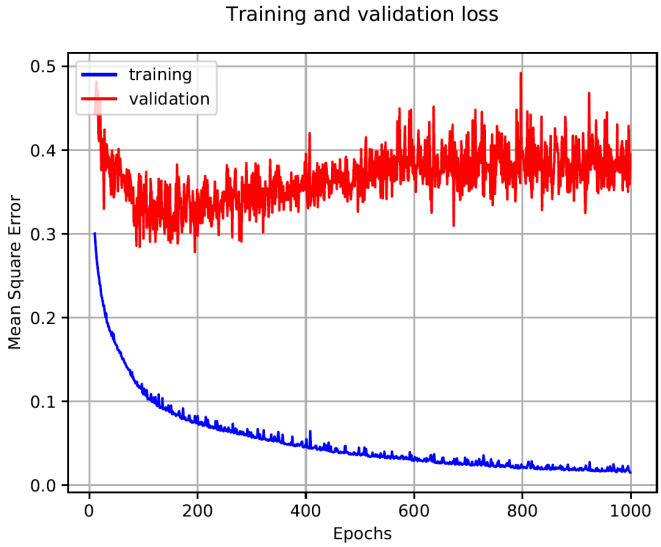
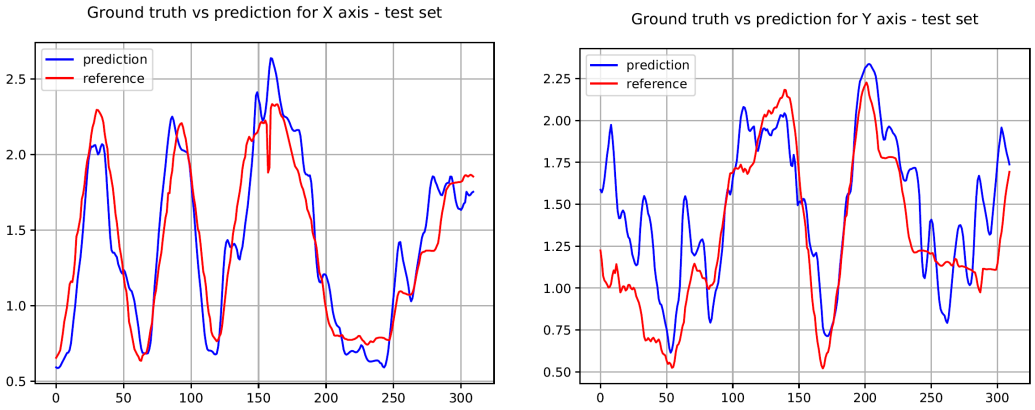


Figure 3.23: Loss functions with L2 regularization with dropout and window size of 10 s



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.24: Test trajectory and ground truth with L2 regularization. dropout and window size of 10 s

No improvement can be noticed in the loss function. On the contrary, it has a worse behaviour than the network without regularization, that is

confirmed by the prediction trajectories. In fact, they present more spikes than other experiments conducted on CNN structures and a very bad trend on y-axis prediction.

The next network to analyse is that with only L2 regularization and a window size of 5 s. Results are in Table 3.14. The Test MSE is slightly better

	32 filters		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.069	0.317	9794

Table 3.14: L2 regularization results - window size of 5s

than previous results, but the number of parameters has increased because of the higher number of filters employed.

The loss function is shown in Figure 3.25 The general trend is worse than

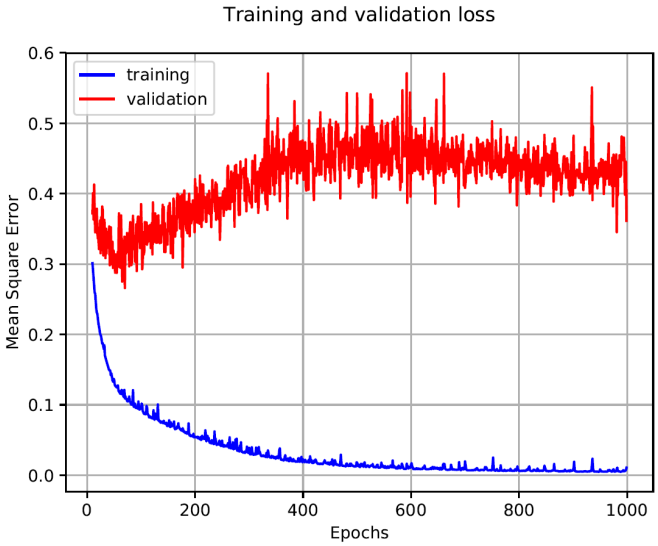
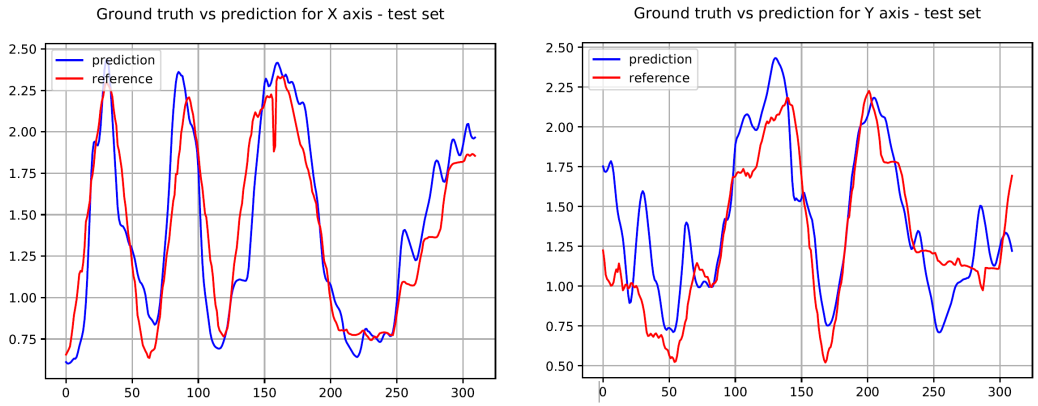


Figure 3.25: Loss functions with L2 regularization and window size of 10 s

previous one: the validation loss diverges soon, then it decreases and it flats around 0.4 m^2 MSE.

Predicted trajectories are shown in Figure 3.26.

It can be easily seen that trajectories are both smoother than previous ones. In the X-axis graph, the general trend of predicted trajectory follows the reference one, with some higher spikes in the last part of the curve.



(a) x trajectory and x ground truth (b) y trajectory and y ground truth

Figure 3.26: Test trajectory and ground truth with L2 regularization and window size of 5 s

In the Y-axis graph more spikes are present especially in the initial and final part of predicted curve while, a better behaviour can be noticed in the middle of the graph.

In general, it can be said that predictions are better than previous results.

The next Pareto point to be analysed is the L2 regularization with dropout and a window size of 10s.

Results are reported in Table 3.15.

	32 filters		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
5	0.053	0.274	17218

Table 3.15: L2 regularization results - window size of 10 s

This time both the number of filters and kernel size is higher, but results in term of MSE are good.

Unfortunately loss functions evolution is suboptimal. They are reported in Figure 3.27.

The validation loss function diverges immediately from the training loss function, but it flats around 0.4 m^2 MSE as in the previous experiments.

The trajectories are shown in Figure 3.28. It can be noticed that the trend of the prediction is smoother than in the previous L2 experiments analysed

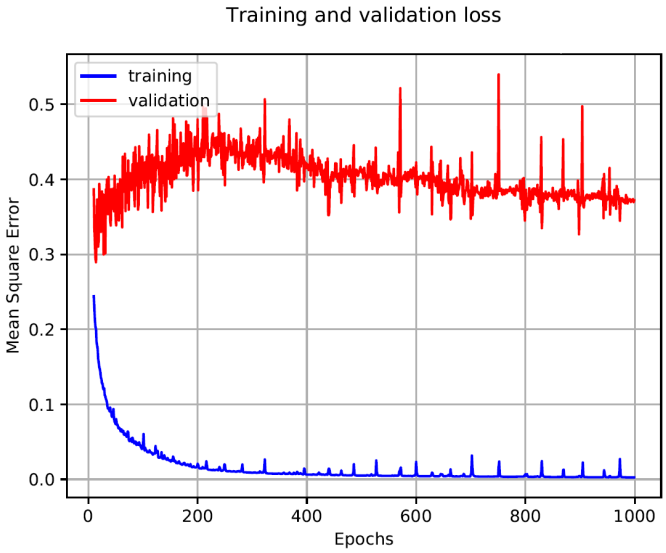
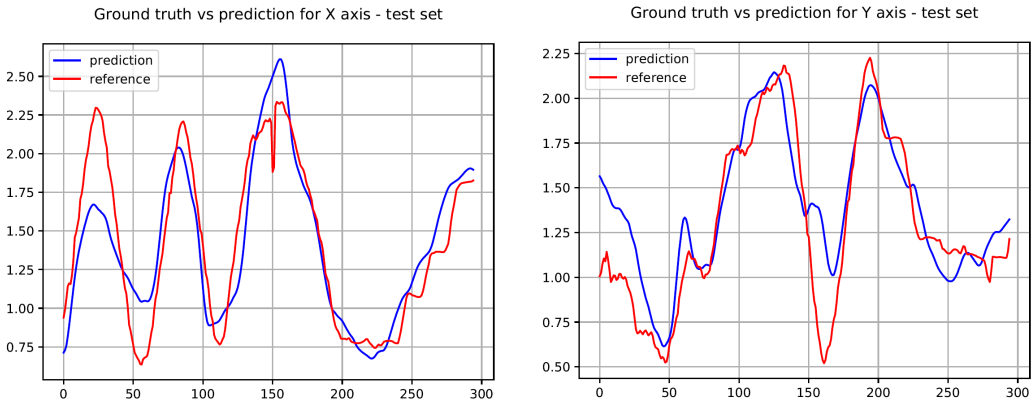


Figure 3.27: Loss functions with L2 regularization and window size of 10 s



(a) x trajectory and x ground truth (b) y trajectory and y ground truth

Figure 3.28: Test trajectory and ground truth with L2 regularization and window size of 10s

but do not predict peaks as well as in the experiments without this kind of regularization.

Mispredictions are concentrated in the initial part of both predicted trajectories. Indeed, the two prediction trajectories cannot follow the references and underestimate peaks but they have a better trend in the final part.

Finally, the experiment with L2 regularization without the dropout is analysed.

The results are reported in Table 3.16. This is the best result obtained in term of Test MSE.

	64 filters		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.047	0.273	34818

Table 3.16: L2 regularization without dropout results - window size of 10s

This time the number of filter is 64 and the kernel size is 3, that leads to a significant increase in the number of parameters.

In the Figure 3.29 and Figure 3.30 loss functions and trajectory prediction are presented.

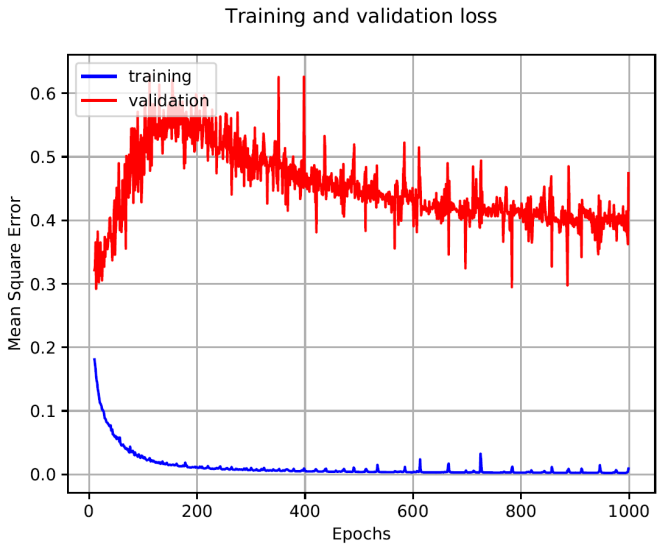
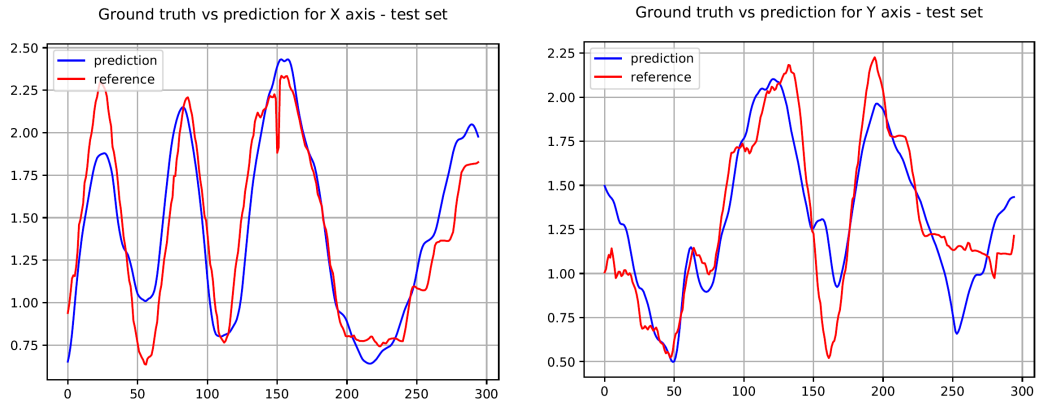


Figure 3.29: Loss functions with L2 regularization without dropout and window size of 10 s

Despite the promising results on Test MSE, the loss functions present a bad trend. The validation loss diverges soon as in the case of L2 regularization with 10 s window and dropout. Nevertheless, it reaches a higher peak, then decreases ands flat around 0.4 m^2 MSE as before. Predicted trajectories have a smoother trend than before, but mispredictions in peaks and around the initial and final Y-axis curve remained.



(a) x trajectory and x ground truth (b) y trajectory and y ground truth

Figure 3.30: Test trajectory and ground truth with L2 regularization without dropout and window size of 10 s

To conclude this analysis, it can be said that L1 and L2 regularizations add few improvements to the network, producing smoother trajectories. On the other hand, some errors have been observed, especially in peaks prediction. The last experiment conducted on 1D-CNNs focused on simplifying the network as much as possible. This will be discussed in the Subsection 3.2.3.

3.2.3 Simple Network analysis

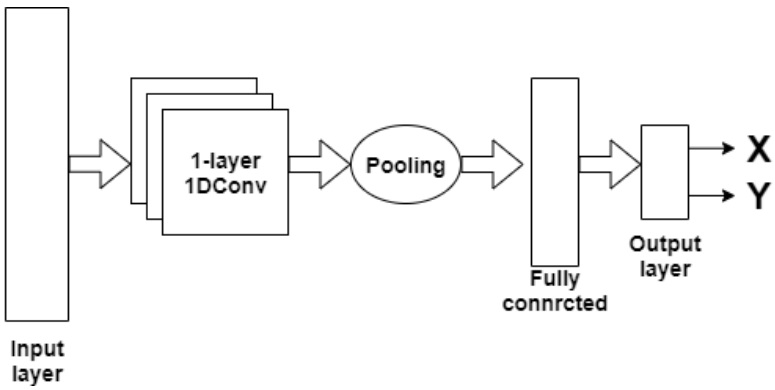


Figure 3.31: Simple 1D-CNN structure

The last step in 1D-CNNs analysis consists in experiments on very simple networks to avoid overfitting the training data. The structure used is shown in Figure 3.31.

The first network analysed is composed of one layer 1D-CNN with a kernel size of three, the number of filters equal to four and an MLP of two neurons. Only early stopping is used as a regularization method. The number of repetition has been increased to 100 for a more accurate analysis of loss functions. The network has been trained over 1000 epochs for 100 times with a window size of 5 s. Results are shown in the Table 3.17.

	4 filters		
Kernel size	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
3	0.079	0.343	100

Table 3.17: Results one layer 1D-CNN with kernel size of three, number of filters equal to four and an MLP of two neurons

The saving in terms of the number of parameters is significant. Moreover, it has similar results to the third network described in Section 3.2.1 (Table 3.11), but with less complexity. The loss function and trajectory predictions are shown in Figure 3.32 and Figure 3.33.

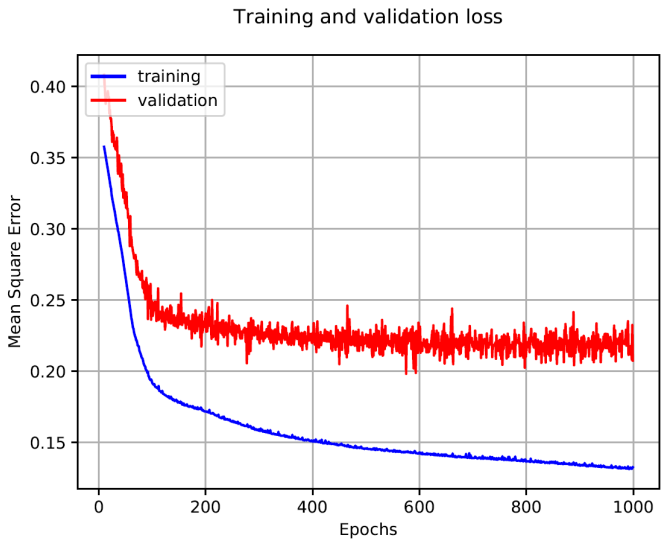
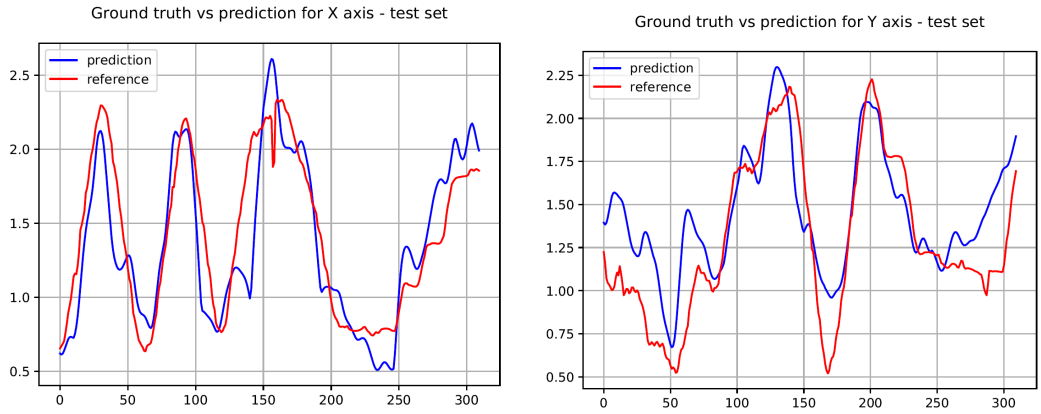


Figure 3.32: Loss functions one layer 1D-CNN with kernel size of three, number of filters equal to four and an MLP of two neurons

The validation loss function is not sporadic and presents a better trend, very similar to the training loss. Moreover, it flats around a 0.22-0.23 m^2



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.33: Test trajectory and ground truth one layer 1D-CNN with kernel size of three, number of filters equal to four and an MLP of two neurons

MSE instead of 0.4 m^2 MSE obtained in previous experiments. Prediction trajectories present some spikes, but in general, the behaviour is quite similar to the predictions obtained in the Section 3.2.1. This means that a simple network structure can infer trajectory in a similar way of a more complex one. In the following, to refer to this first network the name "SimpleNet" will be used. The complexity of this network has been gradually increased to improve the results. Several experiments have been conducted, they are listed in the order of execution:

- SimpleNet with lower learning rate: the same optimizer Adamax has been used but with a lower learning rate. Since this lead to worse results, an experiment on a different optimizer has been conducted

- SimpleNet and different optimizer: SGD has been used instead of Adamax, with a different learning rate. The results were worse than the previous step, then a different way has been followed in the next

- SimpleNet with 8 neurons in the MLP: because of previous worse results, Adamax was chosen again as optimizer and the number of neurons in MLP layer has been increased. Results were better than the previous try.

- SimpleNet with 8 neurons in the MLP and 8 filters: since the incrementation of neurons gave a good result, the number of filters has been

increased as well. This led to a slightly worse result in terms of Test MSE. Then, in the following experiments the number of filters has been changed to 4 again

- SimpleNet with 16 neurons in the MLP: since increasing the number of neurons led to better results, another experiment with 16 neurons in the MLP layer has been conducted. Slightly worse results in terms of Test MSE have been reached respect to the previous experiments in this Subsection

- SimpleNet with 8 neurons in the MLP and window size of 10s: starting from the good result with 8 neurons, it has been chosen to widen the window size. This led to similar results to the previous ones in terms of Test MSE.

All the results are summarized in the Table 3.18 and a resume graph shows the Pareto curve Figure 3.34. The SimpleNet is a Pareto point of the design space.

window size	Kernel size	Filters	MLP number of neurons	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
5	3	4	2	0.086	0.358	100
5	3	4	2	0.089	0.365	100
5	***3***	***4***	***8***	***0.070***	***0.307***	***238***
5	3	8	8	0.079	0.340	450
5	3	4	16	0.075	0.331	422
10	3	4	8	0.072	0.338	398

Table 3.18: Results of all the experiment on SimpleNet

The other network, with results highlighted in Table 3.18, will be analyzed in the following. Again, the result obtained is close to the one obtained with a complex network. This probably means that the real problem is related to the data set: a lower complexity network matches the complexity of the data set and produce results similar to the ones obtained with a more complex and regularized network that tend to overfit and are not able to generalize well on this data set.

In Figure 3.35 and Figure 3.36 the loss function of the second Pareto point are shown.

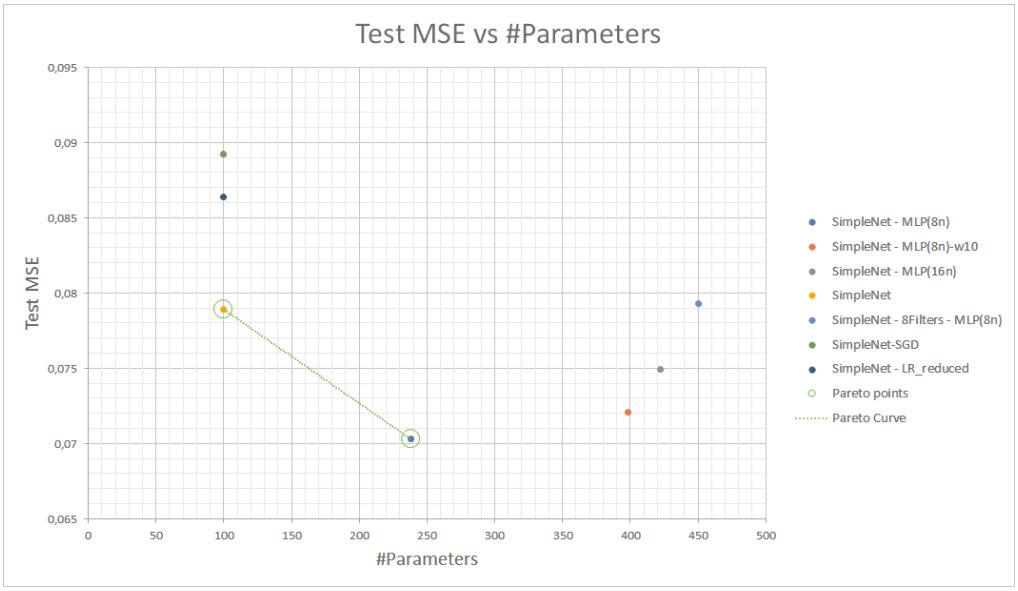


Figure 3.34: DSE of the experiment on SimpleNet

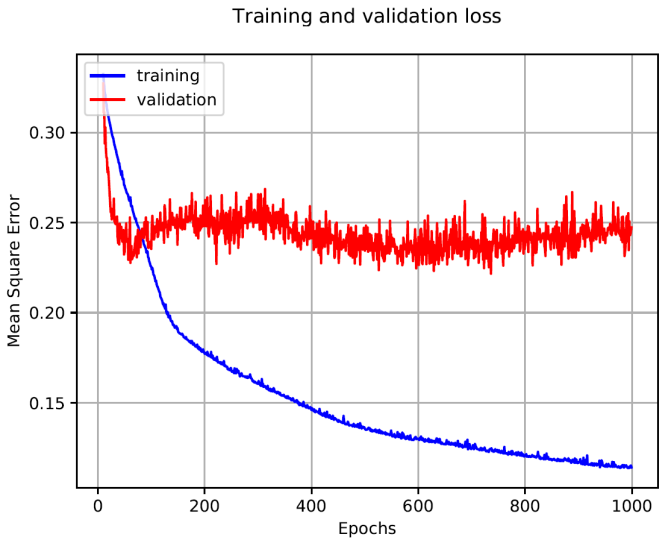
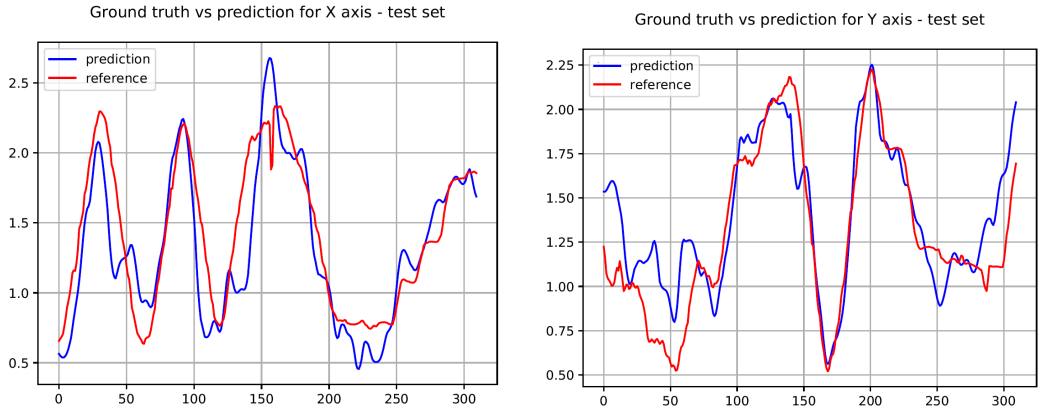


Figure 3.35: Loss functions one layer 1D-CNN with kernel size of 3, number of filters equal to 4 and an MLP with 8 neurons

The validation loss function presents better results than the ones obtained by the training in the beginning. Then, it flats around 0.25 m^2 MSE, that is a similar result to the SimpleNet.

Prediction trajectories follow the trend of the reference curves with some spikes in the initial part of Y-axis prediction and the last part of the X-axis

prediction.



(a) x trajectory and x ground truth (b) y trajectory and y ground truth

Figure 3.36: Test trajectory and ground truth one layer 1D-CNN with kernel size of 3, number of filters equal to 4 and an MLP with 8 neurons

3.2.4 CNNs summary

To conclude, in the following a summary of all the experiments on CNNs is presented in Table 3.19, Table 3.20, Table 3.21. In Figure 3.37 a graphical representation is shown, with Pareto points highlighted. In the legend there are the name of the networks analysed:

- 4conv-MLP64-MLP64-w5: the network with four convolutional layers, 2 MLP (both with 64 neurons) and a window size of 5 s
- 4conv-MLP32-MLP64-w5: the network with four convolutional layers, 2 MLP (the first with 32 neurons, the second with 64 neurons) and a window size of 5 s
- 4conv-MLP32-MLP32-w5: the network with four convolutional layers, 2 MLP (both with 32 neurons) and a window size of 5 s
- 2conv-MLP32-MLP32-w5: the network with two convolutional layers, 2 MLP (both with 32) and a window size of 5 s
- L2-8filters-kernel3-w5: the L2-regularized network with two convolutional layers (with 8 filters and a kernel size of 3), 2 MLP (both with 32 neurons) and a window size of 5 s

- L2-32filters-kernel5-w10: the L2-regularized network with two convolutional layers (with 32 filters and a kernel size of 5), 2 MLP (both with 32 neurons) and a window size of 10 s
- L2-64filters-kernel3-w10: the L2-regularized network with two convolutional layers (with 64 filters and a kernel size of 3), 2 MLP (both with 32 neurons) and a window size of 10 s
- 1conv-MLP2-4filters-kernel3-w5: the L2-regularized network with one convolutional layers (with 4 filters and a kernel size of 3), 1 MLP with 2 neurons and a window size of 5 s
- 1conv-MLP8-4filters-kernel3-w5: the L2-regularized network with one convolutional layers (with 8 filters and a kernel size of 3), 1 MLP with 8 neurons and a window size of 5 s

Window size (s)	1st MLP neurons	2nd MLP neurons	Kernel size	N. of filters	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
5	***64***	***64***	***3***	***8***	***0.064***	***0.297***	***7618***
5	32	64	3	8	0.073	0.322	4258
5	32	32	3	8	0.083	0.350	3138
5	32	32	3	8	0.070	0.329	2738

Table 3.19: Large structure summary: all networks have four 1D-CNN layers excepted for the last one that has two 1D-CNN layers

Window size (s)	1st MLP neurons	2nd MLP neurons	Kernel size	N. of filters	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
5	32	32	3	8	0.074	0.333	2738
10	32	32	5	32	0.053	0.274	17218
10	***32***	***32***	***3***	***64***	***0.047***	***0.273***	***34818***

Table 3.20: 1D-CNN 2-layers structure with L2 regularization summary

Window size (s)	MLP neurons	Kernel size	N. of filters	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
5	2	3	4	0.079	0.343	100
5	***8***	***3***	***8***	***0.070***	***0.307***	***238***

Table 3.21: Small structure summary with one 1D-CNN layer

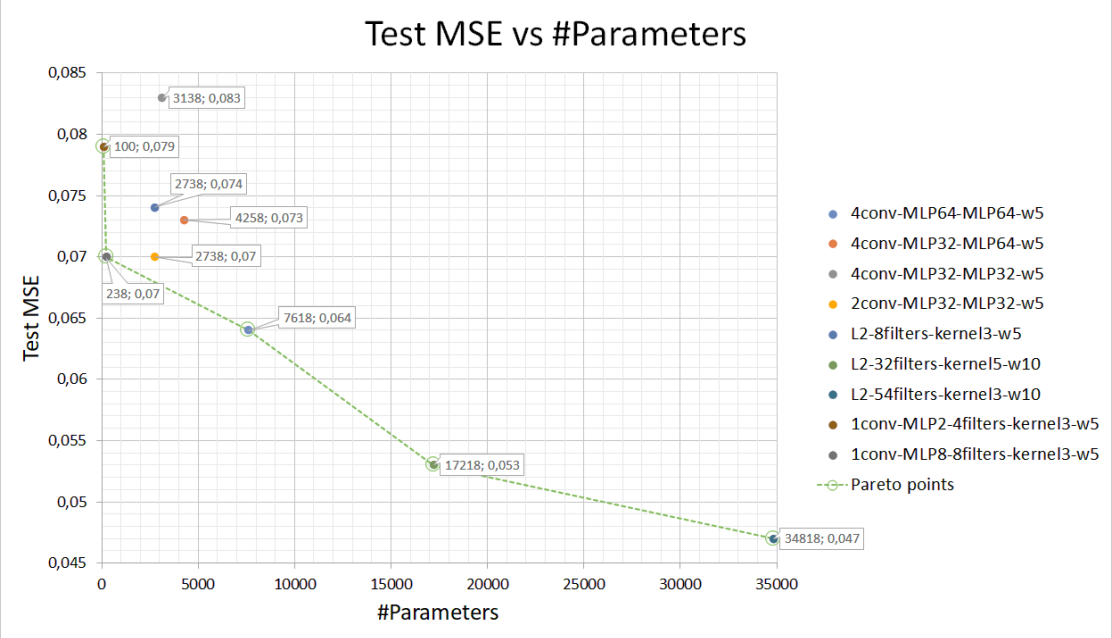


Figure 3.37: Summary CNN DSE

3.3 Long-Short Term Memory (LSTM)

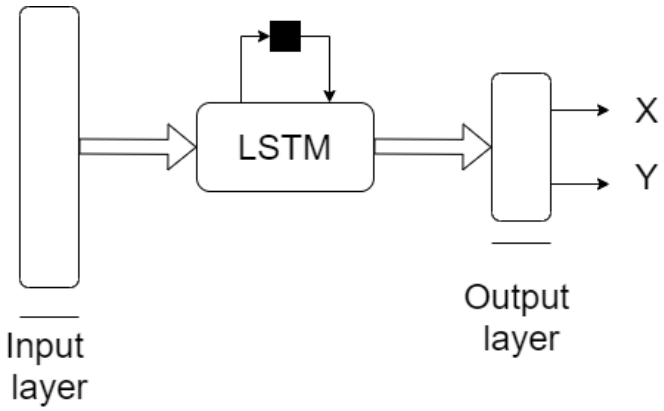


Figure 3.38: LSTM network

Among recurrent NNs, LSTM has been chosen because they are largely used in data sequences analysis, as described in Chapter 2 Section 2.4. The structure of the network analysed is shown in Figure 3.38.

The main parameters changed during the design space exploration are:

- number of neurons in an LSTM stage
- number of LSTMs in the network

The data set has been divided in sequences as already done for CNNs, the process has been explained in Section 3.2.

The initial window size has been set to 5 s, then also experiments with a different window size of to 10 s have been conducted. The network has been trained over 1000 epochs and early stopping regularization has been implemented.

The number of runs on the same structure has been increased to 100, to better check the general behaviour of the network.

The main steps executed during the optimization of this network for the data set used are:

1. analysis of a small network with one LSTM layer:
    - a simple experiment has been executed to study the behaviour of the network.

- the window size has been increased to allow the network to analyse more data to create correlations between them and check if prediction improves.

2. analysis on a regularized LSTM network:

- a dropout regularization between LSTM and output layer has been used to check if the prediction improves.
- the number of LSTM layers has been increased to two to check if the prediction improves.

A summary on the DSE in terms of Test MSE and the number of parameters is presented at the end of this section.

3.3.1 Small network with one LSTM layer

A first experiment on a network with one LSTM layer has been conducted. Results obtained are in Table 3.22.

Window size	Neurons in LSTM layer	Test MSE (m^2)	Distance error (m)	Complexity(N. of parameters)
5 s	2	0.076	0.334	62
5 s	***4***	***0.074***	***0.321***	***154***
5 s	8	0.077	0.336	434
5 s	16	0.085	0.355	1387

Table 3.22: 1-layer LSTM results with a window size of 5 s

Test MSE results are similar to those obtained with 1D-CNNs, but the general saving in terms of the number of parameters is significant. The best result highlighted will be analysed in more detail in the following. Loss function is represented in Figure 3.39.

This time, training loss function keeps decreasing and validation loss function trend flattens around 0.27 m^2 MSE.

General trend among the different execution has been analysed: all training loss functions have a decreasing trend and the majority of validation loss function flattens around 0.25-0.30 m^2 MSE.

Predicted trajectories are shown in Figure 3.40. The X-axis prediction has a good smooth trend over the reference, while the Y-axis trend presents more

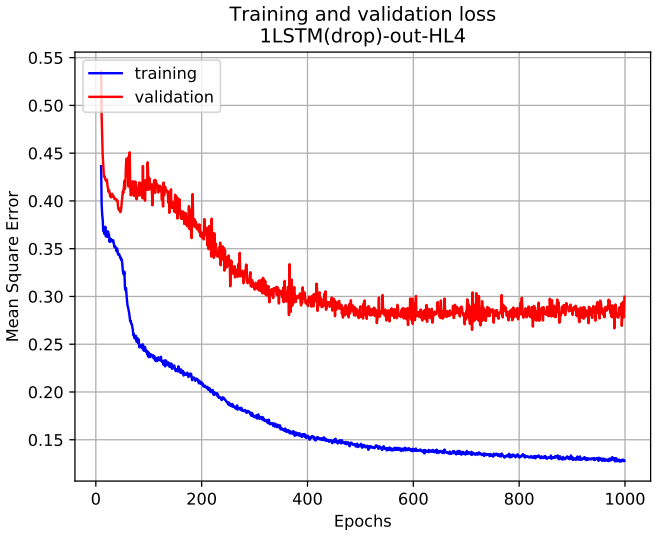
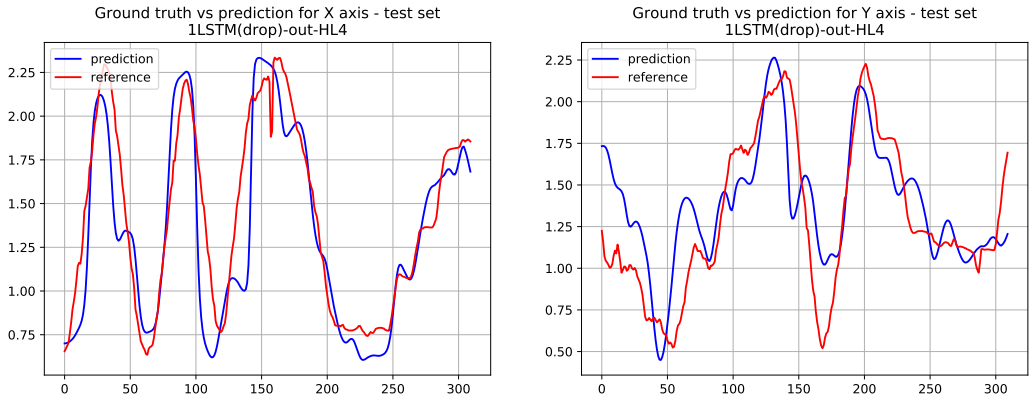


Figure 3.39: Loss functions of one layer LSTM network with four neurons



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.40: Test trajectory and ground truth of one layer LSTM network with four neurons

spikes.
In the Y-axis graph, it can be noticed that the lower peak in the reference curve is underestimated by the predicted trajectory.

Since results were good, then it has been chosen to increase the window size to check if the prediction improves.

Results are in the Table 3.23. Test MSE results are better than previous ones for all values of neurons in the LSTM layer. The highlighted best result

Window size	Neurons in LSTM layer	Test MSE (m^2)	Distance error (m)	Complexity(N. of parameters)
10 s	***2***	***0.056***	***0.301***	***62***
10 s	4	0.059	0.302	154
10 s	8	0.066	0.320	434
10 s	16	0.058	0.299	1387

Table 3.23: 1-layer LSTM results with a window size of 10 s

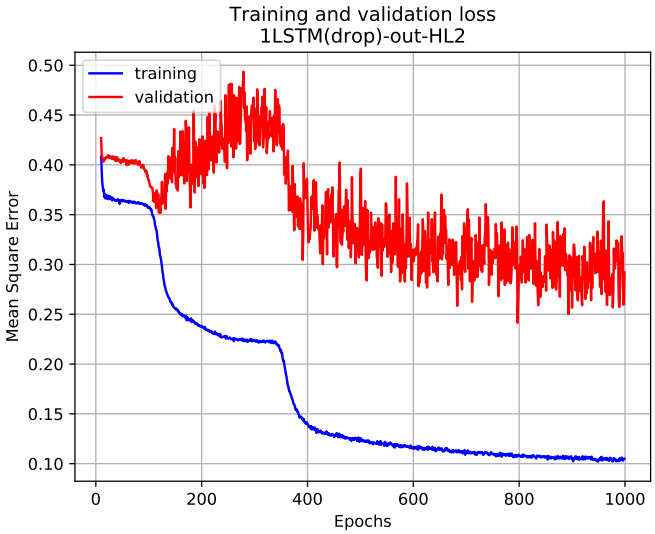


Figure 3.41: Loss functions of one layer LSTM network with four neurons

loss function is shown in Figure 3.41.

The general trend is similar to the previous: despite some initial flat parts, training loss keeps decreasing and the validation loss flattens around 0.30 m^2 MSE.

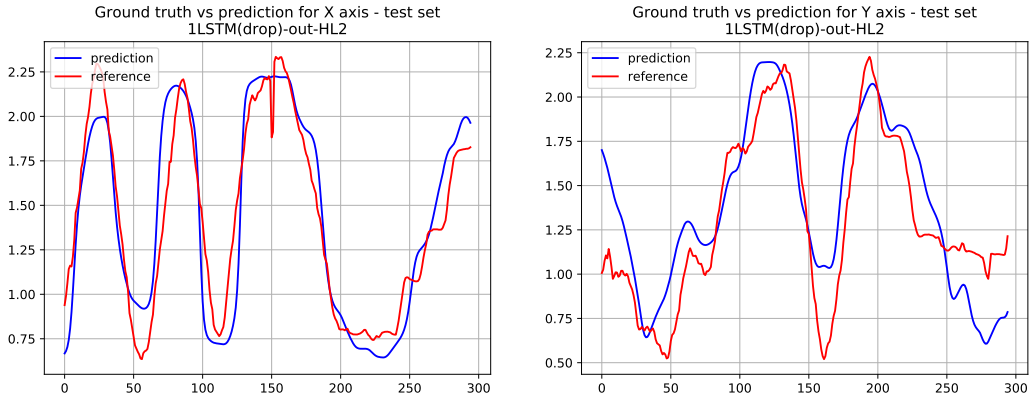
This general behaviour is confirmed by other execution loss functions, a worse prediction behaviour is expected.

The predicted trajectories are shown in Figure 3.42. Prediction trajectories are both smoother than previous ones.

The Y-axis predicted trajectory trend presents less spikes, but the lower peak is not approximated well. The X-axis prediction trajectory trend do not present spikes but cannot approximate well the peaks.

Since the increase of the window size produced overall better results, it has been chosen to keep it and use a simple dropout regularization immediately

after the LSTM layer to reach better results.



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.42: Test trajectory and ground truth of one layer LSTM network with four neurons

3.3.2 Analysis on a regularized LSTM network

Dropout regularization has been added between the LSTM layer and output layer. The experiment has been executed with a window size of 10 s. Results are reported in Table 3.24. Results are similar to the ones without the dropout

Window size	Neurons in LSTM layer	Test MSE (m^2)	Distance error (m)	Complexity(N. of parameters)
10 s	2	0.093	0.388	62
10 s	4	0.069	0.330	154
10 s	***8***	***0.056***	***0.294***	***434***
10 s	16	0.065	0.309	1387

Table 3.24: Results of 1-layer LSTM and dropout with window size of 10 s

layer, the best result has been reached with an LSTM number of neurons equal to eight. Loss functions are represented in Figure 3.43.

Loss functions have a more regular behaviour: the training one has a decreasing trend, while the validation loss function has a diverging drift but it keeps still around 0.25-0.30 m^2 MSE in the final part of the graph.

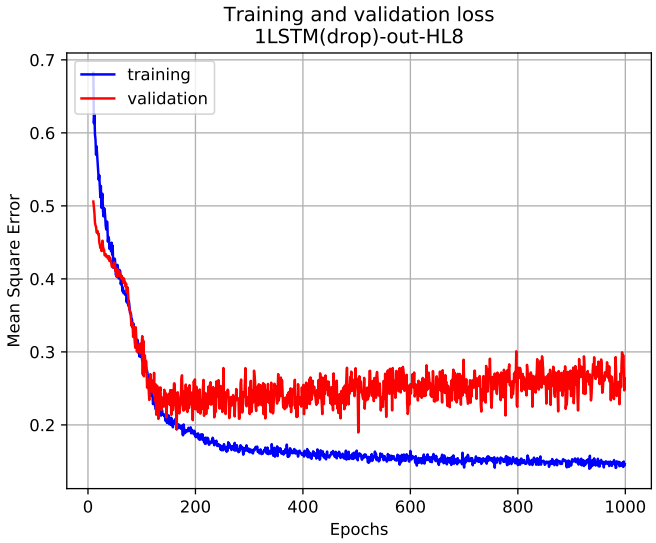
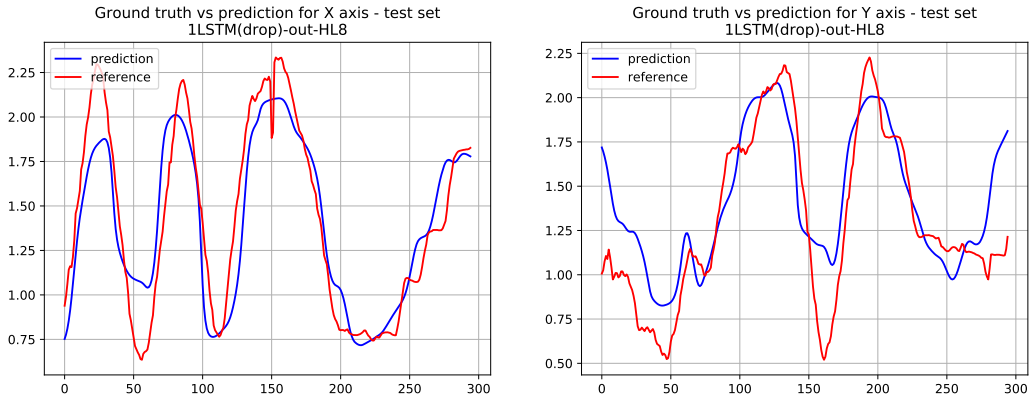


Figure 3.43: Loss functions of one layer LSTM network with eight neurons and dropout



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.44: Test trajectory and ground truth of one layer LSTM network with eight neurons and dropout

The predicted trajectories are shown in Figure 3.44. Both of them have a smooth trend that follows the reference curves but they cannot predict the peaks as well as previous experiments. It can be noticed that Y-axis prediction is less accurate than the X-axis one.

Finally, it has been chosen to increase the size of the network by adding another LSTM layer to the structure while maintaining the dropout layer.

Results are stored in Table 3.25. In general, results are slightly worse than

Window size	Neurons in LSTM layer	Test MSE (m^2)	Distance error (m)	Complexity(N. of parameters)
10 s	2	0.100	0.390	102
10 s	4	0.079	0.337	298
10 s	***8***	***0.066***	***0.304***	***978***
10 s	16	0.075	0.323	3490

Table 3.25: Results of 2-layer LSTM network with dropout and window size of 10 s

previous ones. Best results have been reached by the network with 8 neurons in the LSTM. The loss functions are reported in Figure 3.45.

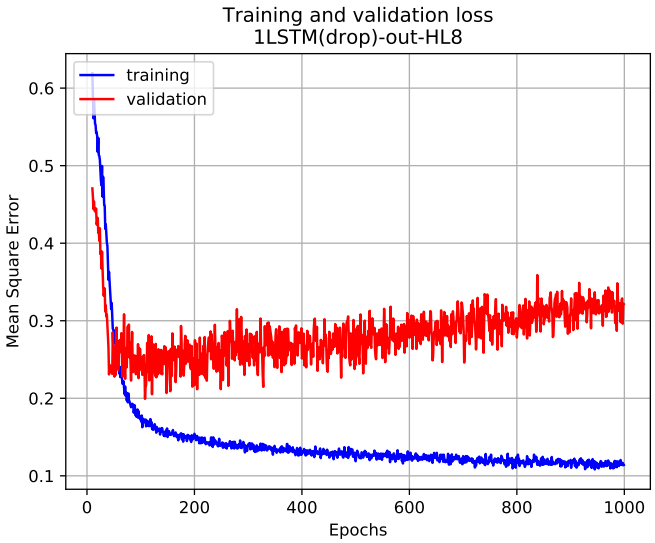
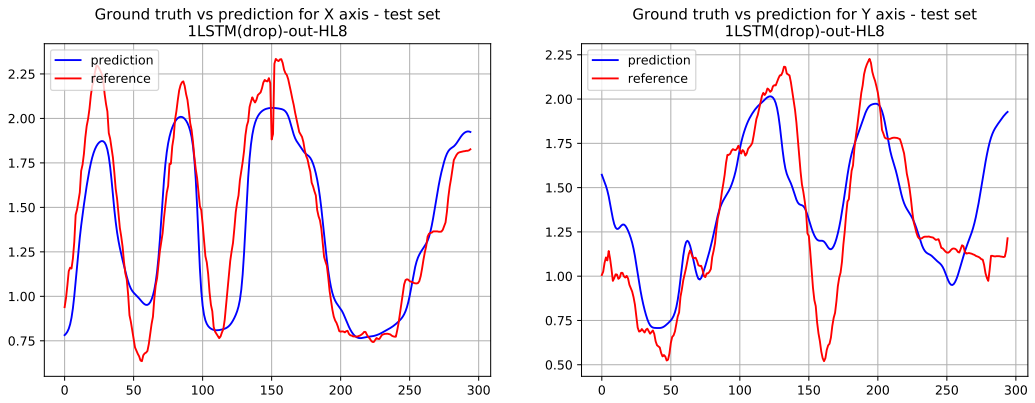


Figure 3.45: Loss functions of one layer LSTM network with eight neurons and dropout

Validation loss function presents a trend similar to the previous one: it slightly diverges instead of flattening. In Figure 3.46 the predicted trajectories are shown. Their trend is slightly smoother than the previous experiment, but the generalization is poor.



(a) x trajectory and x ground truth

(b) y trajectory and y ground truth

Figure 3.46: Test trajectory and ground truth of one layer LSTM network with eight neurons and dropout

3.3.3 LSTMs summary

To conclude LSTM network analysis, a summary is presented in Table 3.26 and a plot of the DSE executed is presented in Figure 3.47. In the legend there are the name of the networks:

- 1LSTM(4)-w5: network with one LSTM layer of 4 neurons and a window size of 5 s
- 1LSTM(2)-w10: network with one LSTM layer of 2 neurons and a window size of 10 s
- 1LSTM(8)-w10: network with one LSTM layer of 8 neurons, dropout regularization of 0.5 and a window size of 10 s
- 2LSTM(8)-w10: network with two LSTM layer of 8 neurons, dropout regularization of 0.5 and a window size of 10 s

Window size (s)	N. of LSTM layers	Neurons in LSTM layers	Dropout value	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
5	1	4	n.d.	0.074	0.321	154
10	***1***	***2***	***n.d.***	***0.056***	***0.301***	***62***
10	1	8	0.5	0.056	0.294	434
10	2	8	0.5	0.066	0.304	978

Table 3.26: Summary table of LSTM best networks

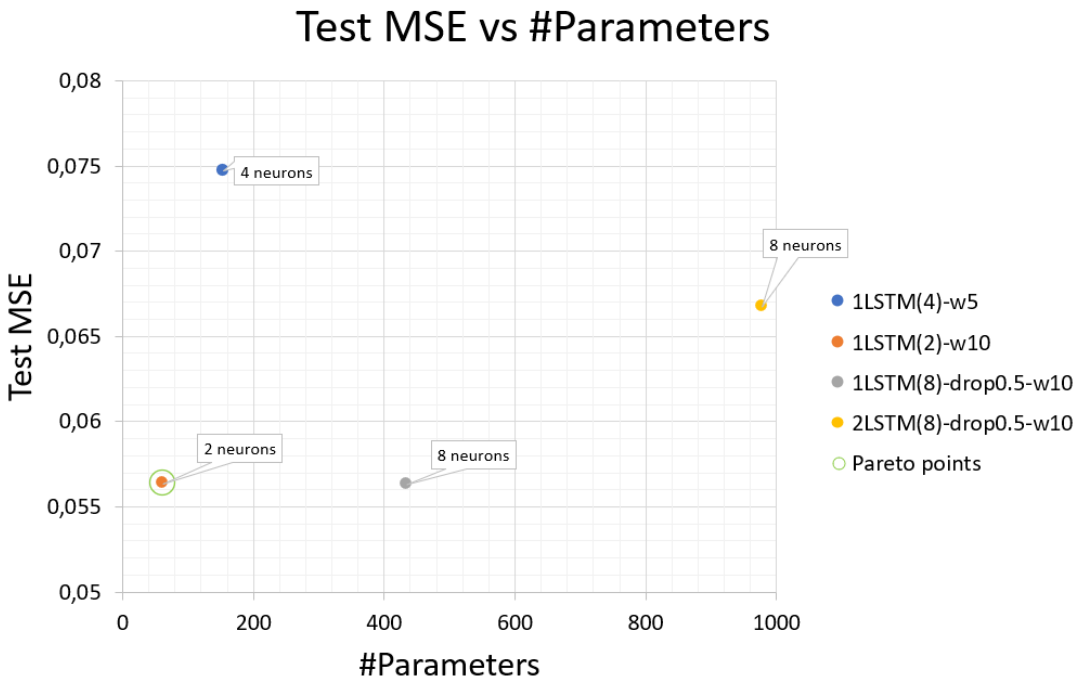


Figure 3.47: DSE of LSTM best networks

Chapter 4

Result Summary

In this chapter, a general summary on all results obtained will be conducted, with a particular focus on the predicted trajectories.

One network for each type analysed has been chosen according to the following criteria based on the two main parameters used for the DSE:

- Test MSE: the network with the minimum MSE is chosen
- Complexity of the network: expressed in "number of parameters". In case of equal MSE, the smallest network has been chosen

In Table 4.1 the best results for each kind of network are summarized, while the final DSE is shown in Figure 4.1.

Network	Window size (s)	Test MSE (m^2)	Distance error (m)	Complexity (N. of parameters)
MLP large	n.d	0.125	0.451	17090
MLP small	n.d.	0.125	0.433	226
CNN large	5	0.064	0.297	7618
CNN regularized	10	***0.047***	0.273	34818
CNN small	5	0.070	0.307	238
LSTM small	10	0.056	0.301	***62***
LSTM large	10	0.066	0.304	978

Table 4.1: Optimization network results summary

The best result on Test MSE has been obtained with the 2-layer 1D-CNN that provides L2 regularization (called "CNN regularized" for simplicity),

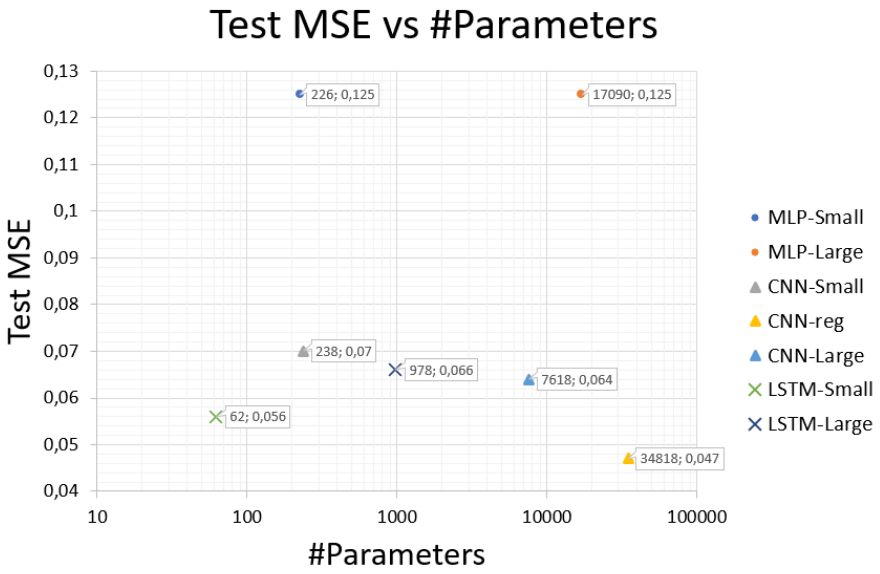
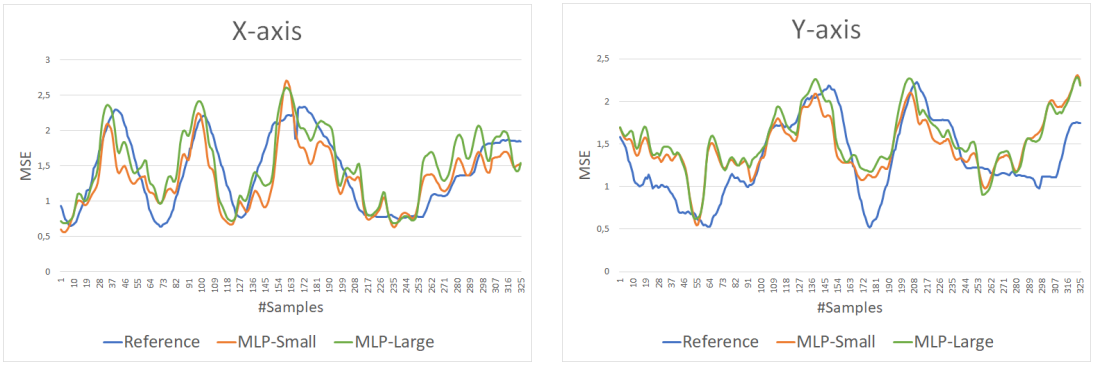


Figure 4.1: DSE summary with detail on lower number of parameters

while the smallest network obtained is the LSTM small, composed by one LSTM layer without dropout regularization.

MLP networks are the worst in general, as it can be easily seen from the DSE graph: as expected window-based networks can perform better on trajectory prediction, and infer the movement of a person in a room. For these reasons, MLP networks have been discarded for the final evaluation on the best trade-off net between the number of parameters and Test MSE. However, in Figure 4.2 a graph of the predicted trajectories of MLP reported



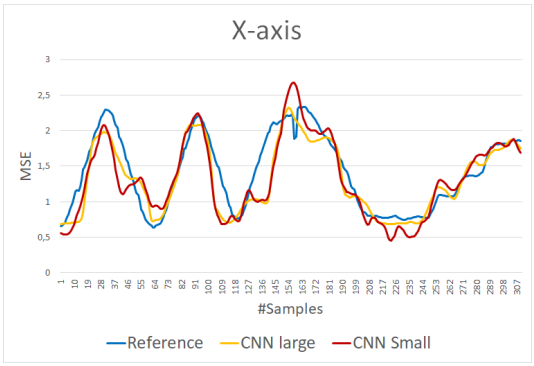
(a) x trajectories and x ground truth

(b) y trajectories and y ground truth

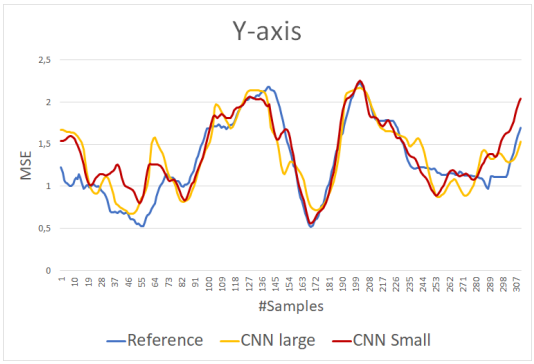
Figure 4.2: Test trajectory and ground truth of two best MLP networks

in the table is shown. In the figure it is confirmed their bad behaviour on this task, as observed in Chapter 3 Section 3.1: the strong presence of spikes in the inferred trajectories cannot allow a clear and good prediction on movements. Better results have been achieved with other networks.

In Figure 4.3 and 4.4 the predicted trajectories of CNNs networks are shown over reference. Because their window size is different they have to be shown in different graphs to understand the behaviour. This is due to the data set slicing needed to analyse this kind of network: the procedure has been described in Chapter 3 Section 3.2.

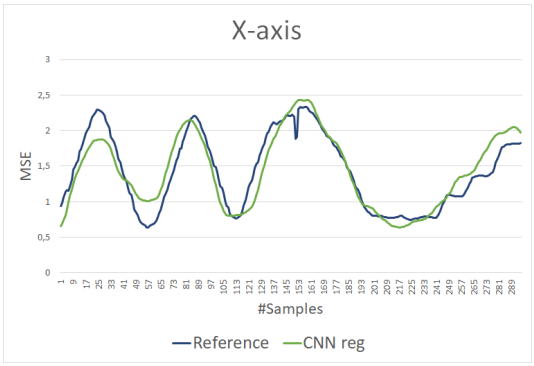


(a) x trajectories and x ground truth

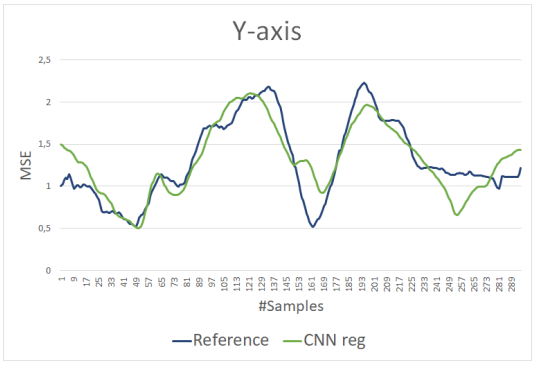


(b) y trajectories and y ground truth

Figure 4.3: Test trajectory and ground truth of best CNN networks with window size of 5 s



(a) x trajectories and x ground truth

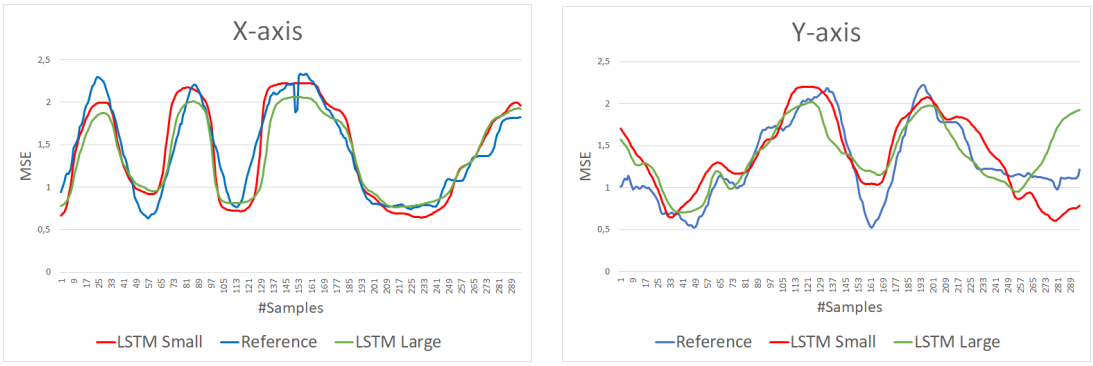


(b) y trajectories and y ground truth

Figure 4.4: Test trajectory and ground truth of best CNN network with window size of 10 s

The general behaviour is better than MLP networks: the predicted trajectories are smoother and follow the reference trend. Spikes are still present in CNN large and small, while they are attenuated in the CNN with L2 regularization. This better results demonstrate that the 1D-CNNs are more suitable than the MLP for person movement prediction.

LSTM prediction trajectories are shown in Figure 4.5. The general trend



(a) x trajectories and x ground truth (b) y trajectories and y ground truth

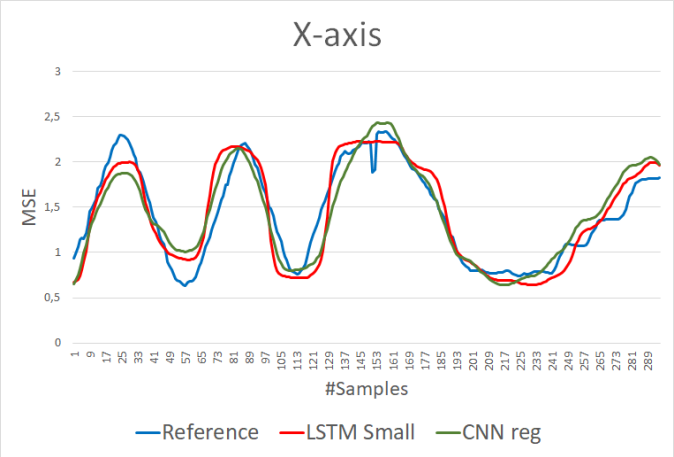
Figure 4.5: Test trajectory and ground truth of best LSTM networks

of these networks is smooth as the one obtained for CNNs. The main problem is related to peaks detection: the predicted trajectories cannot infer well the peaks of reference curve. However, it can be said that small LSTM can follow better the reference with respect to large LSTM.

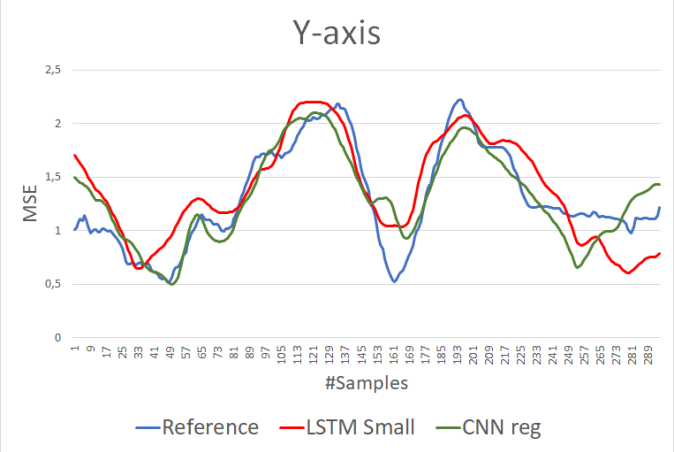
From this discussion on trajectories and obtained results, best network are the regularized CNN and the small LSTM. To compare them directly a common graph has been plotted in Figure 4.6.

Despite the good result obtained with the regularized CNN in terms of Test MSE, it has to be taken into account the size of this network: this is the largest one among all the best results.

Then it can be said that the LSTM is the best trade-off network that can analyse the data set. Indeed, it reaches slightly worse results but with a significant saving in terms of complexity.



(a) x trajectories and x ground truth



(b) y trajectories and y ground truth

Figure 4.6: Test trajectory and ground truth of two best networks

Chapter 5

Conclusion and future work

The main results of the neural networks optimization work are summarized in the following:

- for all network analysed, small structures can reach results similar to larger ones, as noticed several times in Chapter 3.
- networks able to analyse sequences of data can infer better the movement of a person in a room, as noticed by comparing results between MLP against 1D-CNN and LSTM
- smaller LSTMs can reach similar results as CNNs, with a significant number of parameters saving, as remarked in Chapter 4

In general, the error distance obtained with the best networks is around 0.27-0.30 cm. Then, it can be said that the objective find a way to analyse data for assisted-living is achieved. Indeed, this kind of error can be suitable for a generic monitoring of person movements in a room since a fine precision is not strictly required.

The error in the analysis can be related to the collected data set, which was small and noisy despite the digital filtering applied. Therefore, it has to take into account that better results can be achieved with a more accurate measurement system. Future work can be devoted to:

- collect a new less noisy data set to confirm the quality of the networks analysed

- try on sensor fusion approach, which means processing data from several types of localisation sensors to collect more information on position and better infer trajectories

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. https://tensorflow.org, 2015.

[2] Osama Bin Tariq, Mihai Teodor Lazarescu, Javed Iqbal, and Luciano Lavagno. Performance of Machine Learning Classifiers for Indoor Person Localization With Capacitive Sensors. *IEEE Access*, 5:12913–12926, 2017.

[3] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. Conditional time series forecasting with convolutional neural networks, 2017.

[4] Andreas Braun, Reiner Wichert, Arjan Kuijper, and Dieter W. Fellner. Capacitive proximity sensing in smart environments. *Journal of Ambient Intelligence and Smart Environments*, 7(4):483–510, July 2015.

[5] François Chollet et al. Keras. https://keras.io, 2015.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks, 2013.

[8] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. Finding common ground: A survey of capacitive sensing in human-computer interaction. In *Proceedings of the 2017 CHI*

Conference on Human Factors in Computing Systems - CHI '17, pages 3293–3315. ACM Press, 2007.

[9] S. Hochreiter and J. Schmidhuber. Long Short Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1D Convolutional Neural Networks and Applications – A Survey. *arXiv preprint arXiv:1905.03554*, page 20, 2019.

[11] Jeffrey Lee, Matthew Cole, Jackson Lai, and Arokia Nathan. An analysis of electrode patterns in capacitive touch screen panels. *Display Technology, Journal of*, 10:362–366, 05 2014.

[12] Marvelmind. "starter set hw v4.9.". https://marvelmind.com/product/starter-set-hw-v4-9/.

[13] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[14] Andrew Ng. Machine learning. https://www.coursera.org/learn/machine-learning.

[15] Andrew Ng. Neural networks and deep learning. https://www.coursera.org/learn/neural-networks-deep-learning.

[16] N. J. Nilsson. "introduction to machine learning". https://ai.stanford.edu/~nilsson/mlbook.html, 1998.

[17] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks. *arXiv:1511.08458 [cs]*, December 2015.

[18] Gábor Petneházi. Recurrent neural networks for time series forecasting, 2019.

[19] Alireza Ramezani Akhmareh, Mihai Lazarescu, Osama Bin Tariq, and Luciano Lavagno. A tagless indoor localization system based on capacitive sensing technology. *Sensors*, 16(9):1448, 2016.

[20] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[21] Jude W Shavlik and Thomas Glen Dietterich. *Readings in machine learning*. San Mateo, Calif. : Morgan Kaufmann Publishers, 1990.

[22] J. Smith, T. White, C. Dodge, J. Paradiso, N. Gershenfeld, and D. Allport. Electric field sensing for graphical interfaces. *IEEE Computer Graphics and Applications*, 18(3):54–60, June 1998.

[23] Osama Bin Tariq, Mihai Teodor Lazarescu, and Luciano Lavagno. Neural network-based indoor tag-less localization using capacitive sensors.

In *Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers - UbiComp/ISWC '19*, pages 9–12, London, United Kingdom, 2019. ACM Press.

[24] P. D. Wasserman and T. Schwartz. Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE Expert*, 3(1):10–15, Spring 1988.

[25] Wikipedia. Convolutional neural network. https://en.wikipedia.org/wiki/Convolutional_neural_network.

[26] Raphael Wimmer, Matthias Kranz, Sebastian Boring, and Albrecht Schmidt. A capacitive sensing toolkit for pervasive activity detection and recognition. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*, pages 171–180, 2007.

Acknowledgements

Scrivere la tesi sicuramente non è stato facile, ma chi mi conosce sa bene che, per me, la parte più difficile da scrivere sono i ringraziamenti. Non per una questione di ingratitudine, ma perché tendo ad essere sintetica in questo, spesso mi limito a scrivere brevissime frasi, come "grazie a tutti".
Ebbene, è arrivato il momento di imparare a fare anche questa cosa!

Cominciamo dal principio, quando questo percorso di laurea magistrale è cominciato. Ringrazio tutte le persone che mi hanno accompagnato negli studi, colleghi speciali come Nico, Gianlu e Riki che ho conosciuto nei primi mesi della magistrale e che mi hanno aiutata tanto nel percorso e nei laboratori affrontati insieme. Conoscenti che sono diventati cari amici da contattare per provare qualche posto nuovo per mangiare.
Ringrazio anche Riccardo, una conoscenza della triennale che è diventata una bella amicizia in un momento difficile... mi ha anche insegnato a saldare!

E poi, soprattutto Irene, con cui ho affrontato molti se non tutti gli esami! Grazie per avermi sopportata nei mesi di sessione intensiva, in balia di crisi esistenziali! Finalmente siamo arrivate al traguardo e non dovremo più vederci... per lo studio! Non vedo l'ora di festeggiare anche con te, che ormai più che una collega, sei una grande amica.

Ringrazio anche il mio relatore, il prof. Lazarescu e il mio correlatore il prof. Lavagno, che mi hanno saputa indirizzare nello svolgimento della tesi, rispondendo alle mie mail a qualsiasi ora del giorno e della notte!
Un ringraziamento ad Osama, che con pazienza ha risposto a tutte le domande e dubbi che ho affrontato durante il percorso della tesi.
Grazie davvero, di tutto!

Vorrei ringraziare anche tutti gli altri professori che mi hanno accompagnato in questo percorso, in particolare il prof. Passerone che si è sempre dimostrato disponibile e pronto a supportare non solo me ma tutti i suoi studenti.

Ma le persone più importanti da ringraziare sono quelle che mi hanno supportato (o sopportato?) nel quotidiano e a distanza.

Inizio ringraziando la mia famiglia, tutti i miei parenti che mi hanno supportata con infinito affetto, chiamandomi e ricordandomi sempre che la lontananza non influisce sull'amore che può legare una famiglia. Quindi grazie a tutti i miei zii, zie e cugini ma specialmente nonna Ninetta che rappresenta da sempre il mio supporto spirituale e morale.

In particolare, vorrei ringraziare mamma e papà, che con tanti sacrifici mi hanno permesso di studiare qui, al Politecnico di Torino, distante 1200 km da casa, ma che sono sempre stati presenti come fossero con me. Mamma Mariolina con il suo supporto da cheerleader che si può esprimere con due motti peculiari: "se insisti e persisti, raggiungi e conquisti" e "piuttosto in bicicletta, ma mia figlia sulla vetta". Papà Sandro con il suo essere sempre fonte di supporto morale in maniera più silenziosa ma comunque efficace.

Un ringraziamento speciale va a mia sorella, che presto diventerà santa, non perchè voglia farsi suora, ma perché deve sopportare il mio carattere ansioso giorno e notte, dato che abita con me! Nei momenti di difficoltà di questo percorso è stata la luce che mi ha aiutato ad andare avanti, anche solo con un abbraccio poderoso o una mandata a quel paese, che tutto sommato ci sta sempre.

Vorrei ringraziare di cuore le AoR, anche se non usiamo più tanto questo appellativo, con cui ho condiviso un periodo ben più lungo della sola laurea magistrale. Gi, "coinquilina" ma soprattutto amica/sorella acquisita, che non perde l'occasione per creare scompiglio nella mia vita ma che mi propone sempre nuove sfide da affrontare e sa ascoltarmi quando ne ho bisogno. Cesp, che ha saputo starmi accanto in momenti difficili, standomi vicino o anche solo autoinvitandosi a casa e preparando qualcosa di buono. Infine, Rossella, che anche da lontano mi ha supportata, la sua è una di quelle amicizie che non invecchiano.

Ringrazio anche le mie coinquiline, Candy amica ormai storica con cui si può sempre parlare di argomenti interessanti e Giulia per la sua vivacità che movimenta la casa e riesce a strapparti sempre un sorriso.

Un ringraziamento va anche a Raffaella, amicizia consolidata, che è sempre presente e disponibile per pranzi e/o serate di giochi di società.
Grazie anche a Serena, una cara amica, con cui parlare sempre e organizzare gite all'Ikea o smezzare una cesta di pollo al KFC.

Infine vorrei ringraziare il gruppo Taranto, composto da nuove e vecchie amicizie che si sono consolidate in questi ultimi anni, che mi hanno supportato e aiutato a staccare nei periodi trascorsi a Taranto con le serate di gioco e chiacchiere. Quindi un grande grazie a Serena, Giuseppe, Andrea, Veronica, Alessandro, Valentina, Francesca, Frank ed Emanuele!

Concludo i ringraziamenti con un haiku di Kobayashi Issa, perché mi sembra una buona idea!

Non scordare:
noi camminiamo sopra l'inferno,
guardando i fiori.

Ora, con questo non voglio dire che la laurea magistrale sia stata un inferno, anzi, è stato un percorso più piacevole della triennale! Volevo solo dire che voi tutti siete stati fiori che hanno abbellito questo percorso.