Neural Network Based Model Predictive Control for an Autonomous Vehicle - An Architecture Analysis

Maria Luiza Costa, Éric Goubault and Sylvie Putot

Laboratoire d'Informatique de l'École polytechnique

November 3, 2020

Introduction

- Introduction
- Motivating application
- Supervised Learning
- Reinforcement Learning
- Tests and Results
- Conclusion

Introduction

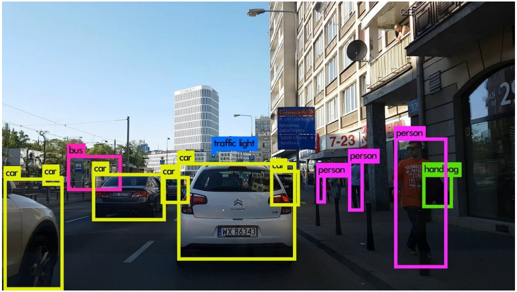
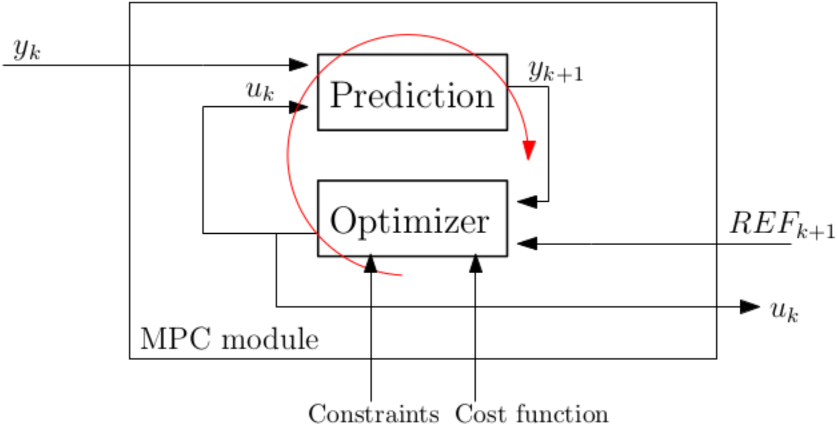


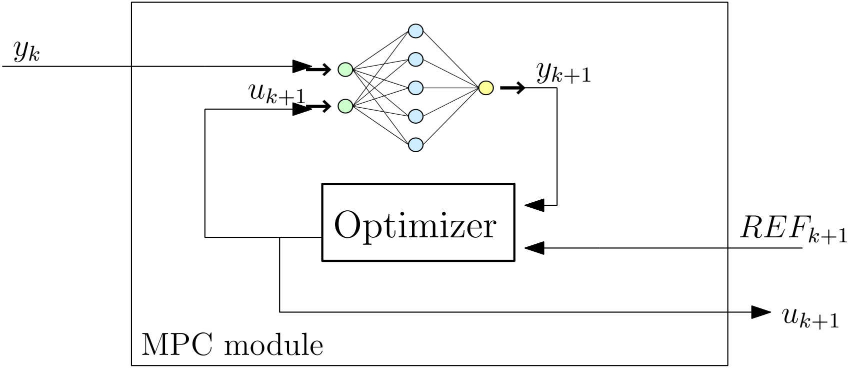
Figure: Image credits: Karol Majek. - Yolo project

Figure: Cruise control symbol

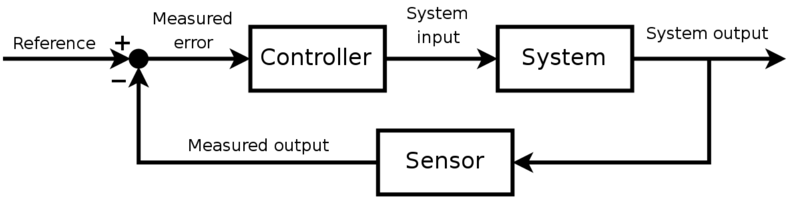
Introduction



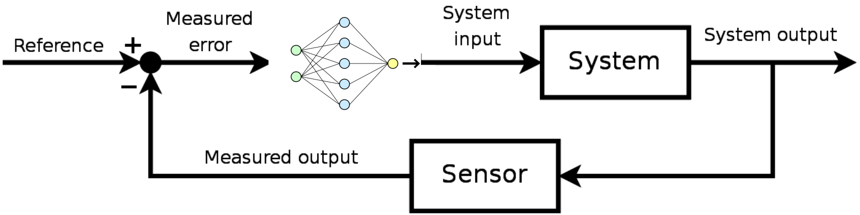
Introduction



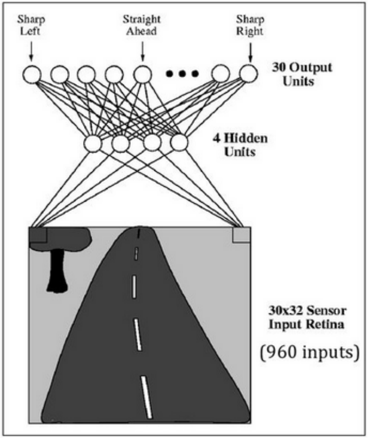
Introduction



Introduction



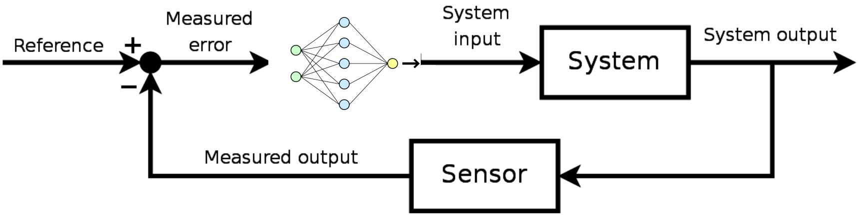
Introduction



- ALVINN learned to steer by *observing a human driver*
- Multiple networks for different roads (e.g. dirt road, two-lane road, highway (up to 70mph!))

Figure: Image credits: Pormelau. - ALVINN project

Introduction



Introduction

Objectives:

- Replace a classical controller for a neural network on the control loop of autonomous systems.
- Assure the stability and precision of neural controllers.

Introduction

Objectives:

- Replace the classical controller for a neural network on the control loop of autonomous systems.
- Assure the stability and precision of neural controllers.
- **Optimize the neural controller architecture to enable an online verification.**

Introduction

Objectives:

- Replace the classical controller for a neural network on the control loop of autonomous systems.
  - Supervised learning
  - Reinforcement learning

What is the simplest architecture we can have and how it affects the network's performance?

Motivating application

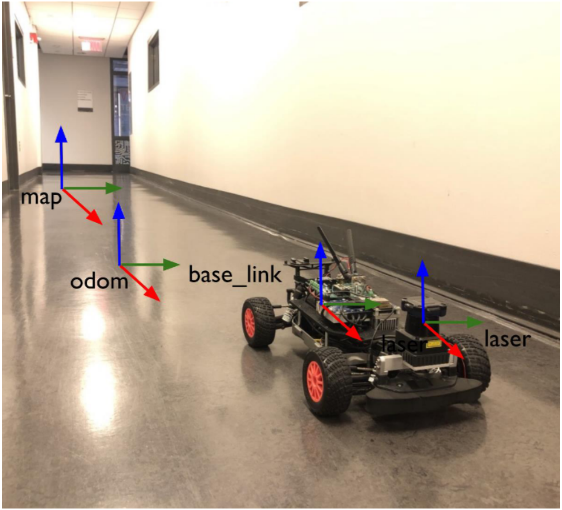


Figure: Image credits: F1tenth competition

Motivating application

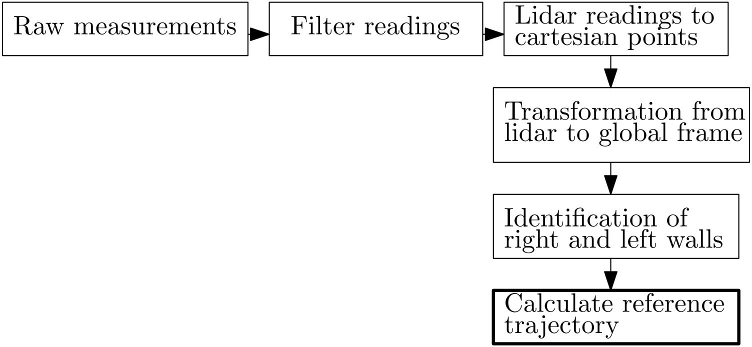


Figure: Image credits: F1tenth competition

Motivating application

$$X^k_{R|W} = \begin{bmatrix} x^k_{R|W} & y^k_{R|W} & \theta^k_{R|W} & v^k_{R|W} \end{bmatrix} \text{ - state vector.}$$

$$v^{k+1}_{R|W} = v^k_{R|W} = v_{R|W}, \text{ for } k = 0, 1, \ldots, T-1.$$

$$X^{k+1}_{R|W} = \begin{bmatrix} x^k_{R|W} + v_{R|W} cos(\theta^k_{R|W})dt \\ y^k_{R|W} + v_{R|W} sin(\theta^k_{R|W})dt \\ \theta^k_{R|W} + v_{R|W} * sin(\delta)/l_f * dt \\ 0 \end{bmatrix}$$

l_f - constant that represents the distance from the vehicle's front to its gravity center .

δ - control command, the desired steering angle.

Supervised Learning

Learning with labels.
Expert : Model Predictive Control (MPC)

- MPC problem - high complexity, not suitable for embedded systems

Supervised Learning

Explicit MPC approach:

- Assumes the MPC problem is Linear Quadratic Problem (LQR).
- Pre-calculates a Continuous Pice-wise Affine Solution. (CPWA)

Advantages of approximating the MPC problem with a neural network:

- Neural networks can represent optimization problems of different sizes.
- Neural networks can represent non linear problems with different complexities.

Supervised Learning

- Feed -Forward Neural Networks,
- Fully connected,
- One continuous output - Linear regression problem,
- Scaled tanh on the output.
- Backpropagation

$$J = \frac{1}{M}\sum_{m=0}^{M}[(u_{NN}^{m} - u_{MPC}^{m})^2$$

Supervised Learning - Training dataset

Dataset1:

- Straight trajectories

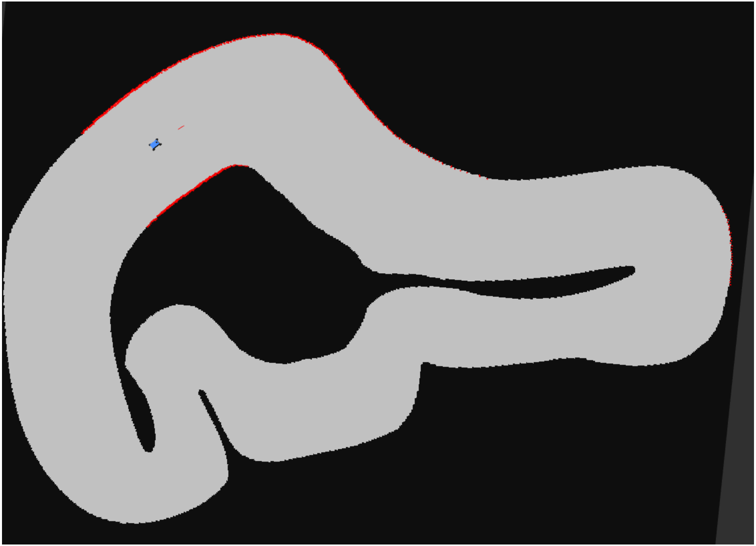
Dataset2:

- Straight trajectories
- Sinusoidal trajectories, $f = f_0$
- Sinusoidal trajectories , $f = 2f_0$

Dataset3:

- Straight trajectories
- Sinusoidal trajectories, $f = f_0$
- Sinusoidal trajectories , $f = 2f_0$
- Spiral trajectories

Supervised Learning - Validation dataset



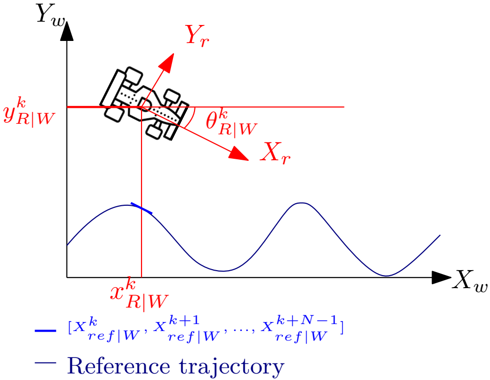
Supervised Learning

- **Inputs**
- Network's architecture
  - Number hidden layers
  - Number of neurons per layer
- Activation function

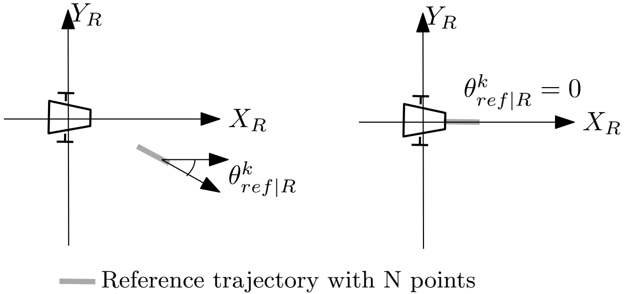
Supervised Learning - Inputs

$$inputs_{MPC}^{k} = \begin{bmatrix} X_{R|W}^{k} & Y_{ref|W}^{k} & \dots & Y_{ref|W}^{k+N-1} \end{bmatrix},$$

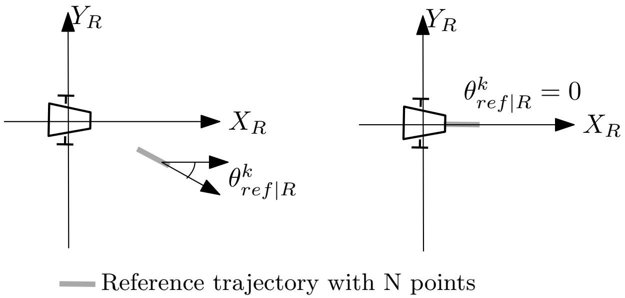
$$Y_{R|W}^{k} = \begin{bmatrix} x_{R|W}^{k} & y_{R|W}^{k} \end{bmatrix}$$



Supervised Learning - Inputs



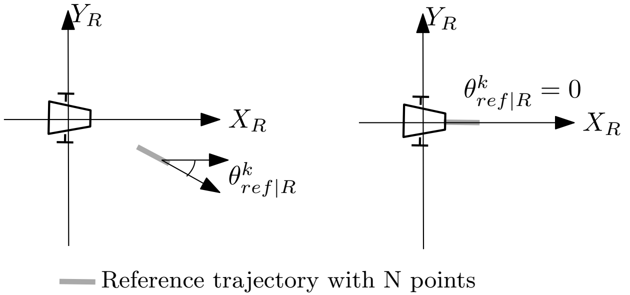
Supervised Learning - Inputs



$$inputs^k_{NN3} = \begin{bmatrix} Y^k_{ref|R} & \theta^k_{ref|R} \end{bmatrix}$$

$$\text{where, } Y^k_{ref|R} = \begin{bmatrix} x^k_{ref|R} & y^k_{ref|R} \end{bmatrix}$$

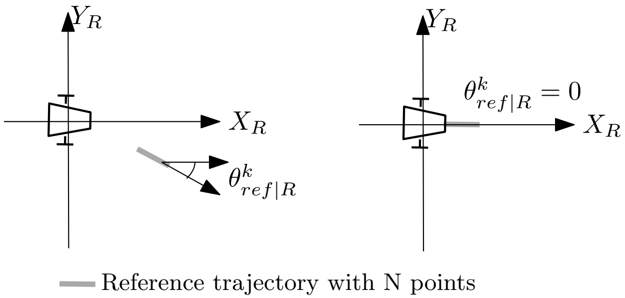
Supervised Learning - Inputs



$$inputs^k_{NN\,N+1} = inputs^k_{NN21} = \begin{bmatrix} y^k_{ref|R} & \cdots & y^{k+N}_{ref|R\,R} & \theta^k_{ref|R} \end{bmatrix}$$

.

Supervised Learning - Inputs



$$inputs^k_{NN2N} = inputs^k_{NN40} = \begin{bmatrix} Y^k_{ref|R} & \dots & Y^{k+N}_{ref|R} \end{bmatrix}$$

$$\text{where, } Y^k_{ref|R} = \begin{bmatrix} x^k_{ref|R} & y^k_{ref|R} \end{bmatrix}$$

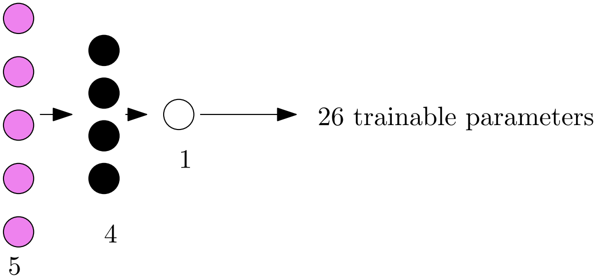
Supervised Learning

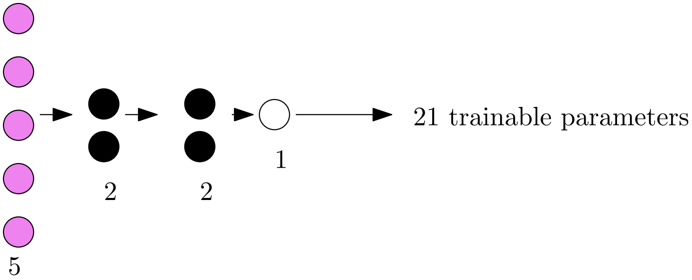
- Inputs
- **Network's architecture**
  - Number hidden layers
  - Number of neurons per layer
- Activation function

Supervised Learning

$$
\begin{aligned}
output_1 &= a_1(W_1 * inputs_{NN} + b_1),\\
output_2 &= a_2(W_2 * output_1 + b_2),\\
&\vdots\\
output_{NN} &= a_{n_h+1}(W_{n_h+1} * output_{n_h} + b_{n_h+1})
\end{aligned}
$$

Supervised Learning

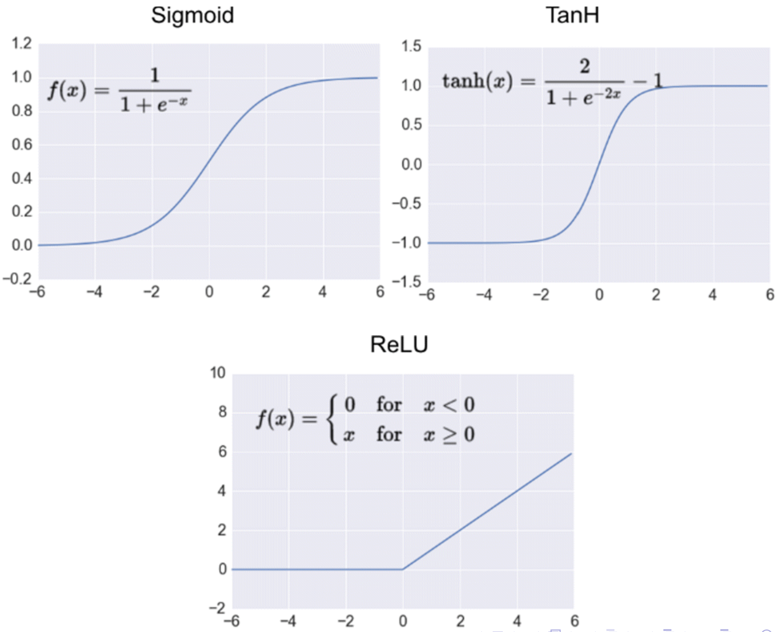




Supervised Learning

- Inputs
- Network's architecture
  - Number hidden layers
  - Number of neurons per layer
- **Activation function**

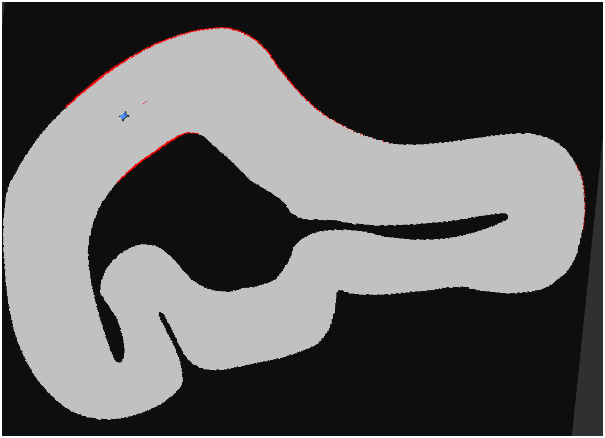
Supervised Learning



Reinforcement Learning

- Deep Deterministic Policy Gradient(DDPG)
- Could not obtain a good performance with small structures.

Tests and Results

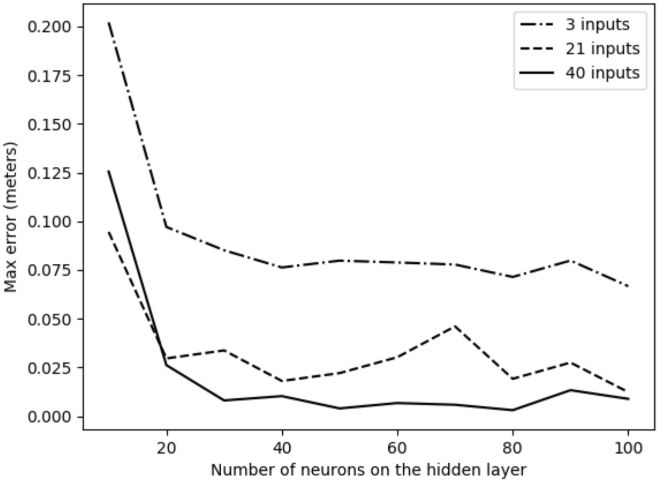


$$J_k = (u^m_{NN} - u^m_{MPC})^2$$

Tests and Results

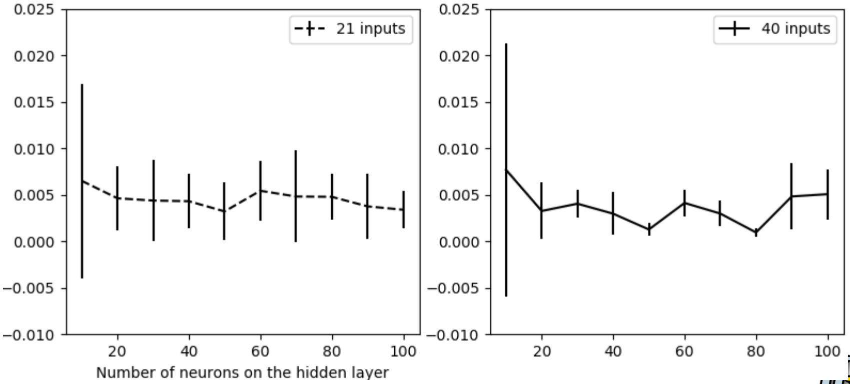
- Inputs : 3,21,40
- Hidden layer : 1
- Number of neurons on the hidden layer : 10::10::100
- Activation function on the hidden layer : relu
- Activation function on the output layer : tanh

Tests and Results



Tests and Results

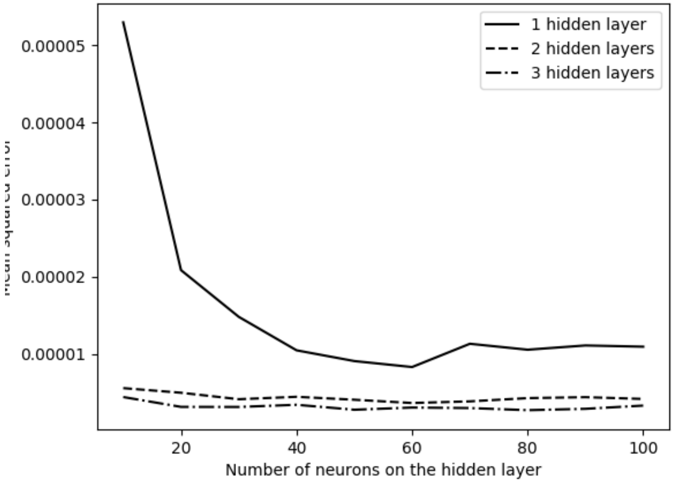
Mean error (meters)



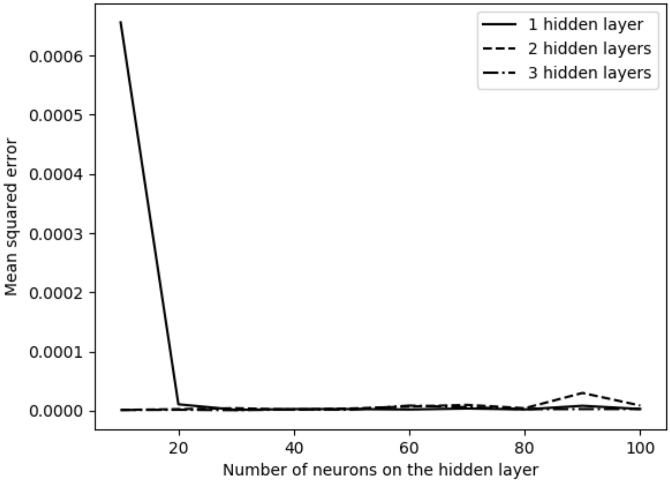
Tests and Results

- Inputs : 40
- Hidden layer : 1,2,3
- Number of neurons on the hidden layers : 10::10::100
- Activation function on the hidden layer : relu
- Activation function on the output layer : tanh

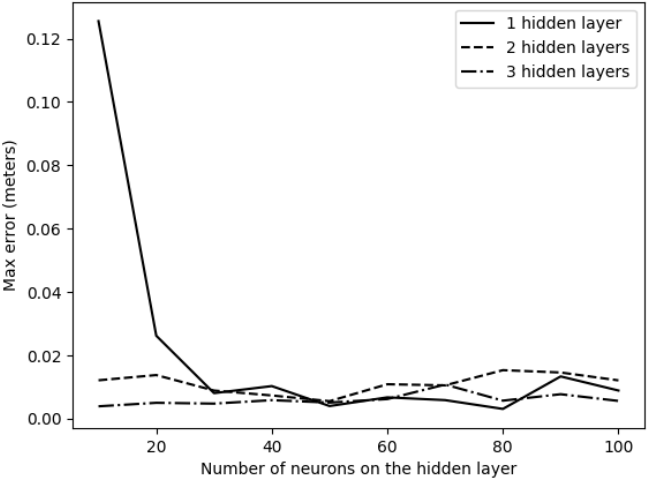
Tests and Results



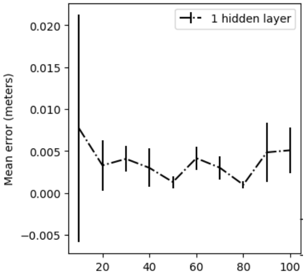
Tests and Results

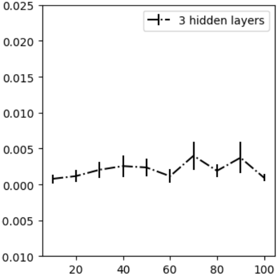


Tests and Results

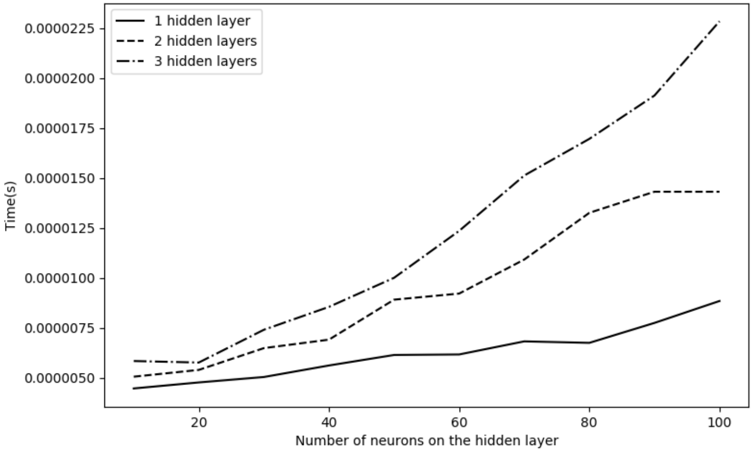


Tests and Results

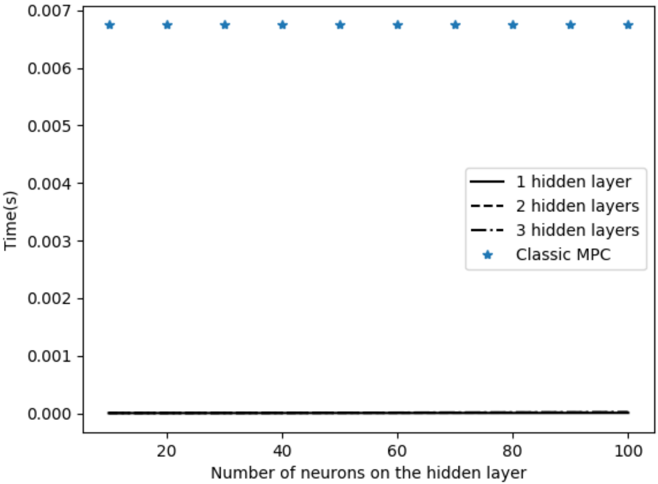




Tests and Results



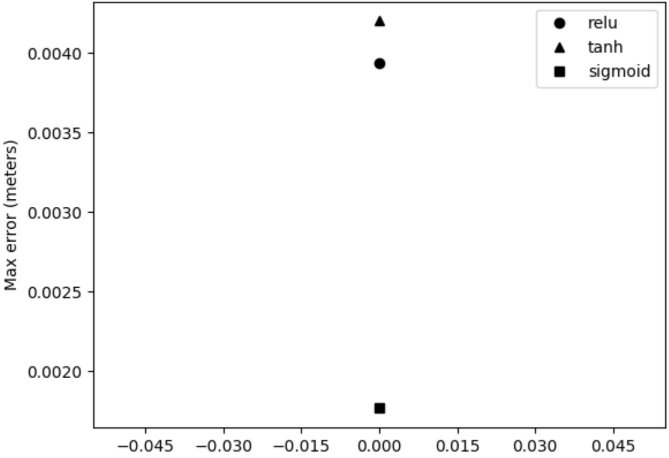
Tests and Results



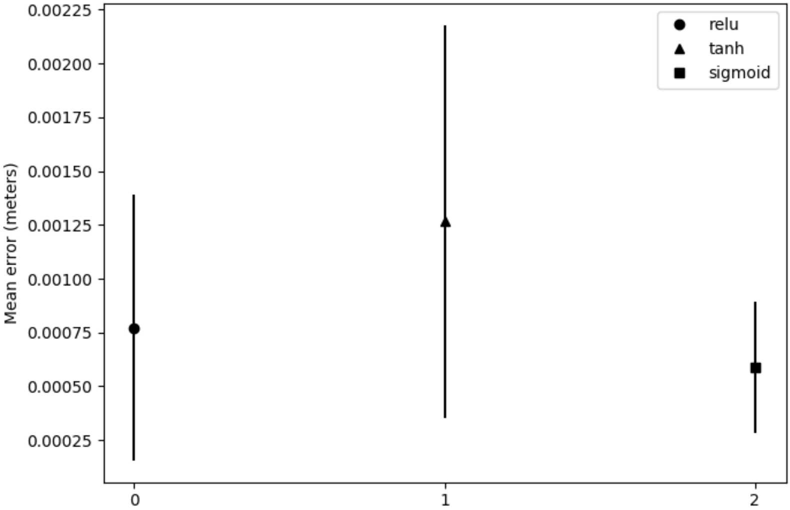
Tests and Results

- Inputs : 40
- Hidden layer : 3
- Number of neurons on the hidden layers :10
- Activation function on the hidden layer : relu,sigmoid,tanh
- Activation function on the output layer : tanh

Tests and Results



Tests and Results



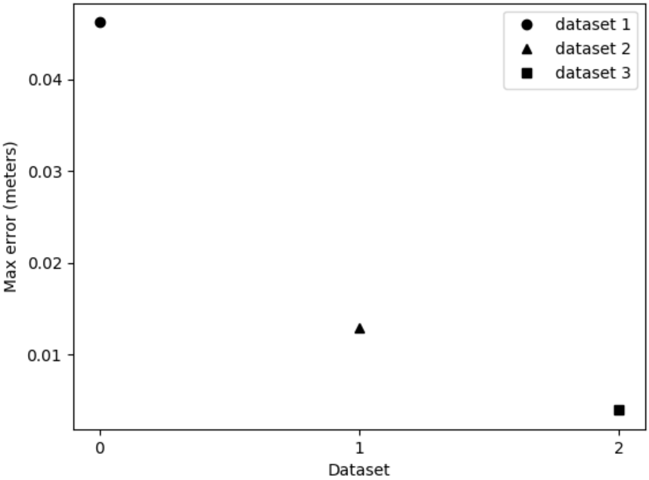
Tests and Results

- Inputs : 40
- Hidden layer : 3
- Number of neurons on the hidden layers :10
- Activation function on the hidden layer : sigmoid
- Activation function on the output layer : tanh

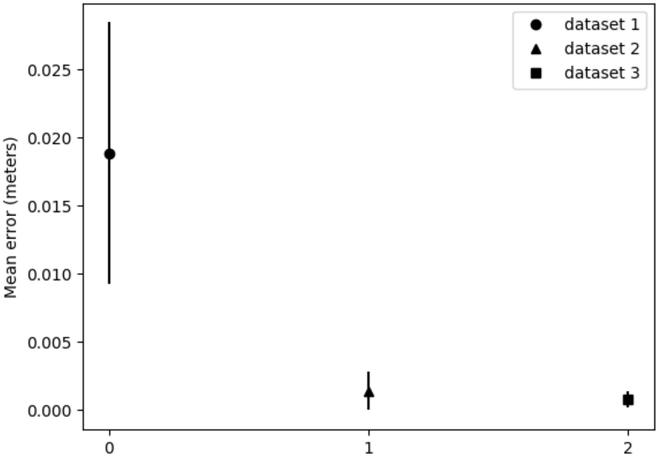
Tests and Results

- Inputs : 40
- Hidden layer : 3
- Number of neurons on the hidden layers :10
- Activation function on the hidden layer : sigmoid
- Activation function on the output layer : tanh
- dataset : 0,1,2

Tests and Results



Tests and Results



Conclusion

Future work :

- Non constant speed
- Apply the neural controller to the real f1tenth competition vehicle.
- Implement the online verification.

Thank you for your attention