A Deep Learning Architecture for Predictive Control

Steven Spielberg Pon Kumar and Bhushan Gopaluni

University of British Columbia

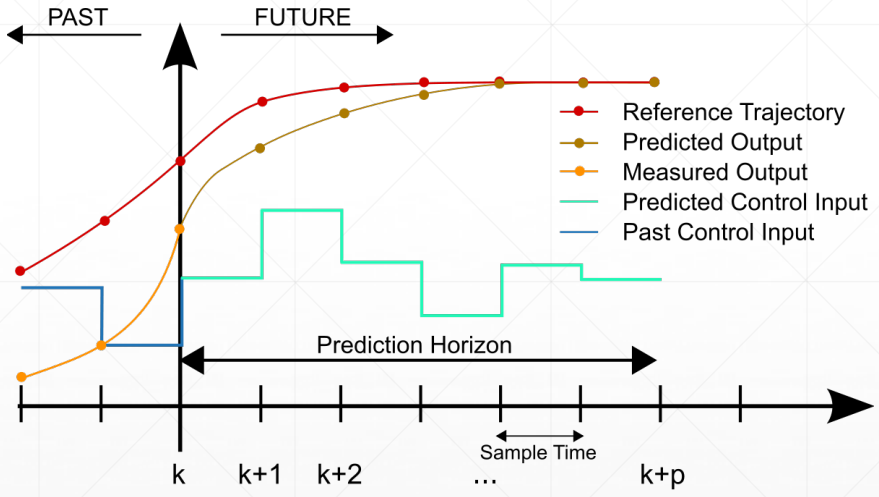
Outline

- Introduction
- Related Works
- Background
- System Setup
- Model Architectures
- Simulation Results
- Observations
- Conclusion

Introduction

Model Predictive Control - Advantages

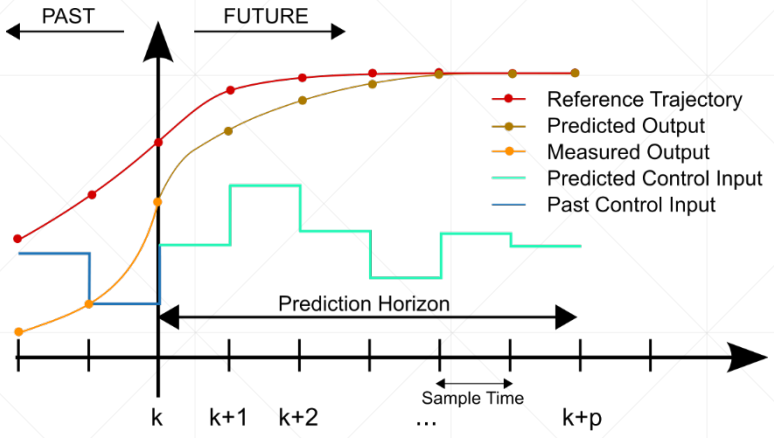
- Look ahead in future to take optimal control actions
- Desired objective can be fed in optimization objective
- Constraint can be imposed in MPC via constraints in optimization objective
- Can handle non-linearities, disturbances, multivariate interactions and model uncertainties
- More degrees of freedom to control the plant optimally compared to P, PI and PID controllers.

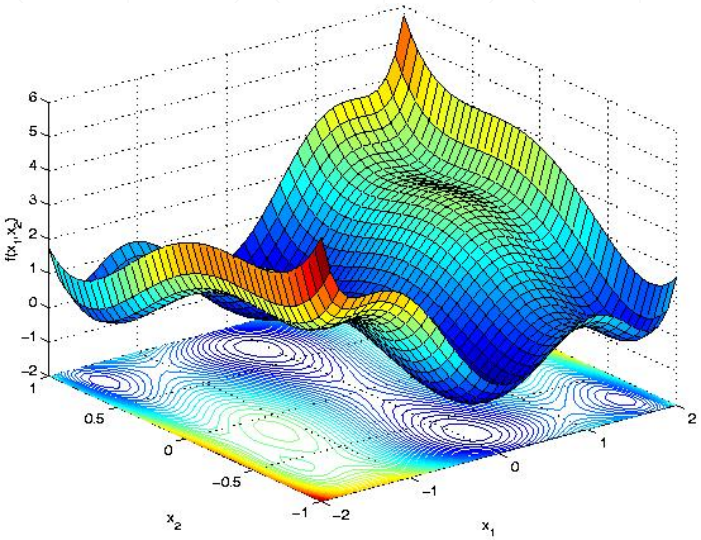


Model Predictive Control - Drawbacks

MPC-Drawbacks

- Solving optimization problem ‘online’ for complex non-linear systems takes time
- Computationally intensive in production stage
- Estimation of hidden states (difficult for complex stochastic non-linear system and for non-observable systems)





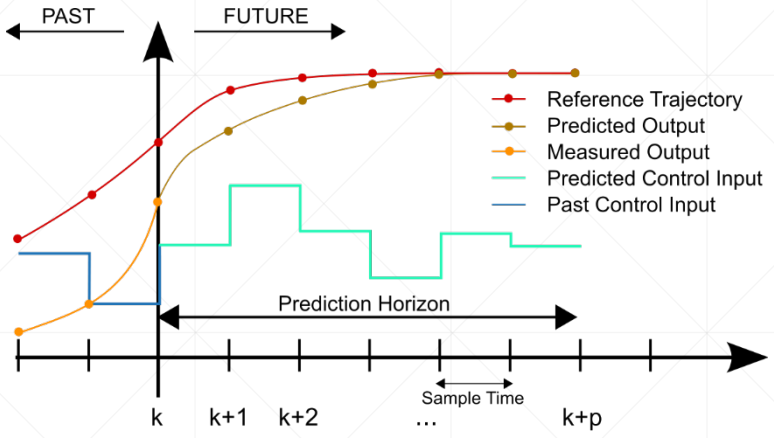
MPC – Existing Approaches to Tackle Drawbacks

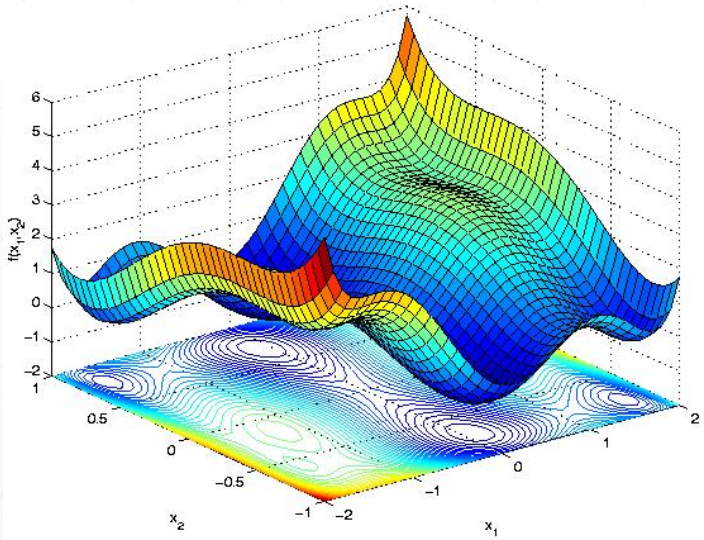
MPC - Drawbacks

- Solving optimization problem 'online' for complex non-linear systems takes time
- Computationally intensive in production stage
- Estimation of hidden states (difficult for complex stochastic non-linear system and for non-observable systems)

Current Approaches to tackle Drawbacks

- Linearize a non-linearize system
- Approximate complex system with simple system

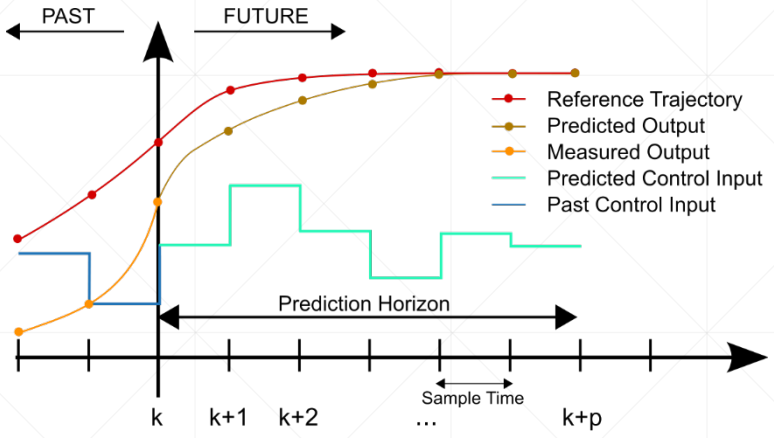




MPC – Existing Approaches to Tackle Drawbacks

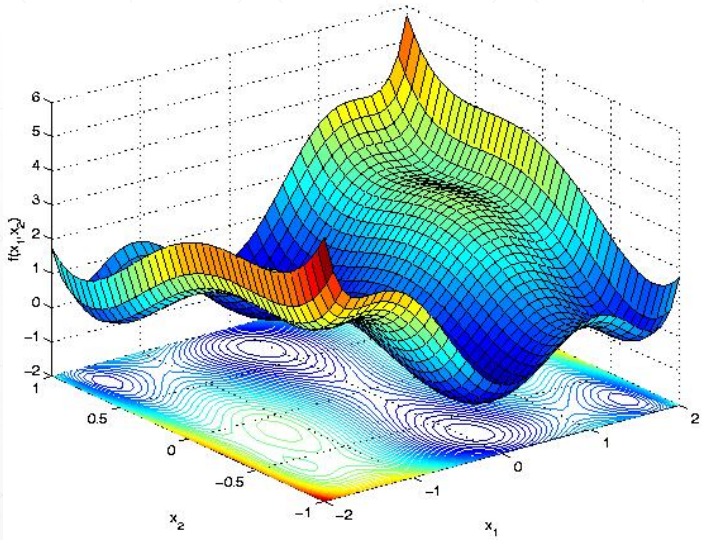
MPC - Drawbacks

- Solving optimization problem 'online' for complex non-linear systems takes time
- Computationally intensive in production stage
- Estimation of hidden states (difficult for complex stochastic non-linear system and for non-observable systems)



Current Approaches to tackle Drawbacks

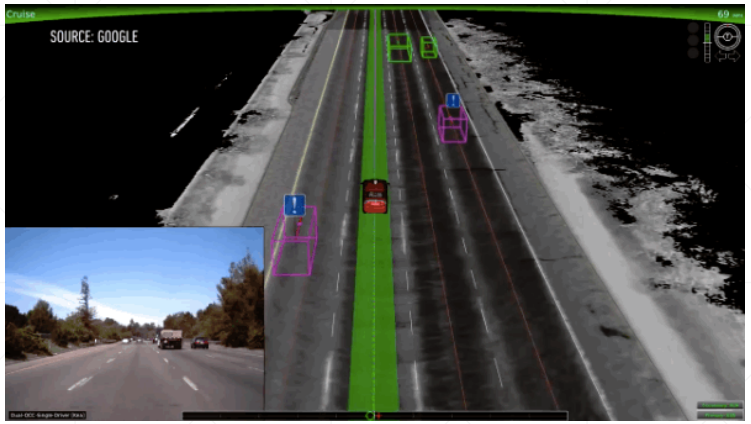
- ~~Linearize a non-linearize system~~
- ~~Approximate complex system with simple system~~



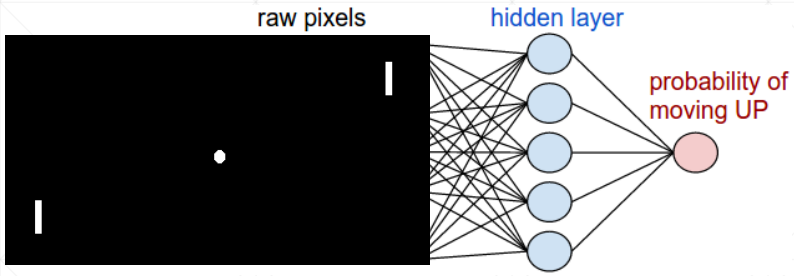
Does not account the full dynamics of the system leading to decrease in performance

Motivation – Learning Control Problems with deep neural networks

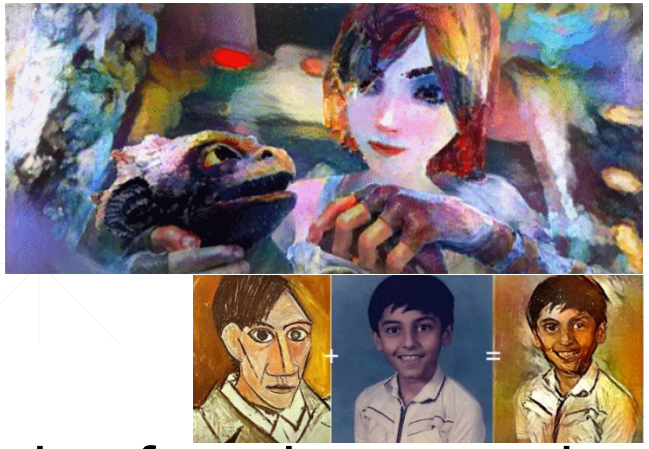
Image Classification + Self Driving Cars



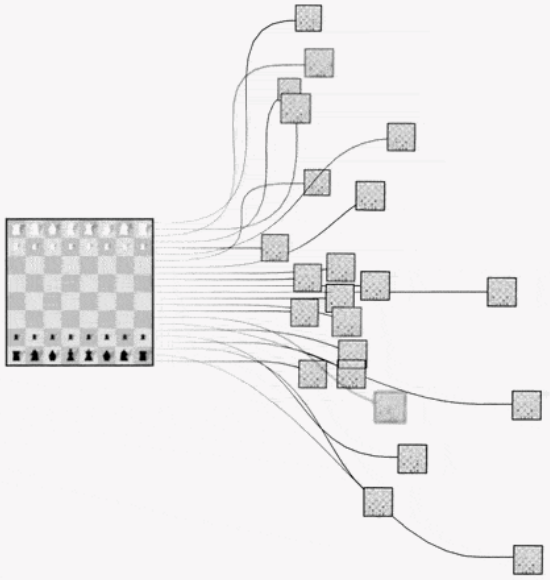
Human Level Control in Games



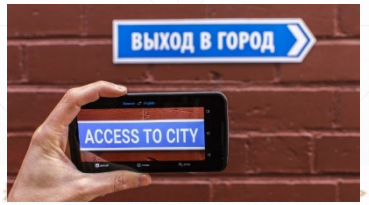
Paint like an Artist

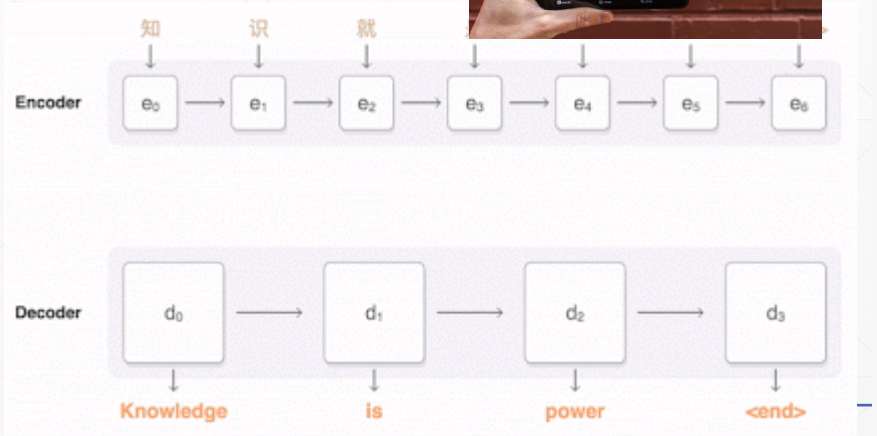


complex Go game (1.74×10^{172} states)

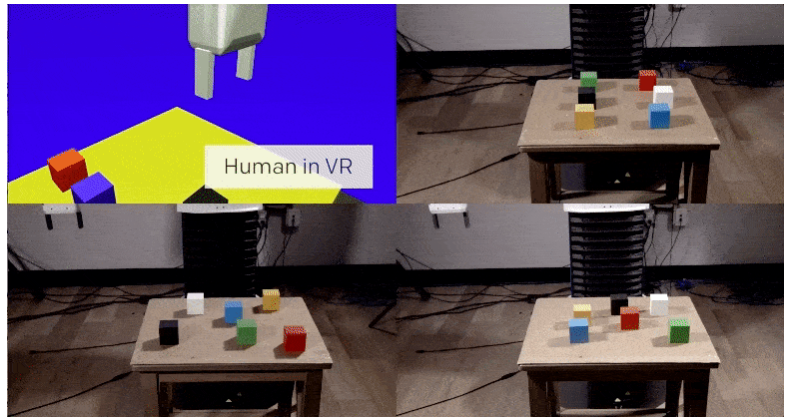


Machine Translation



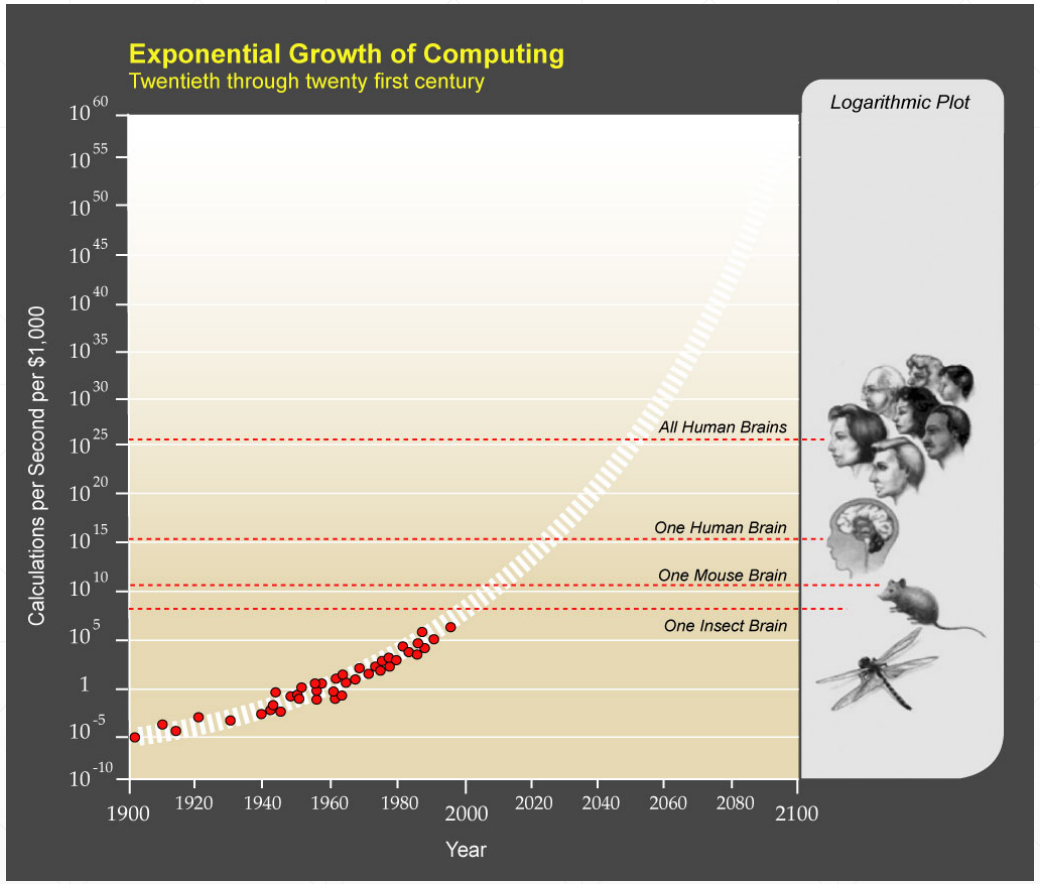


Learning from humans via VR

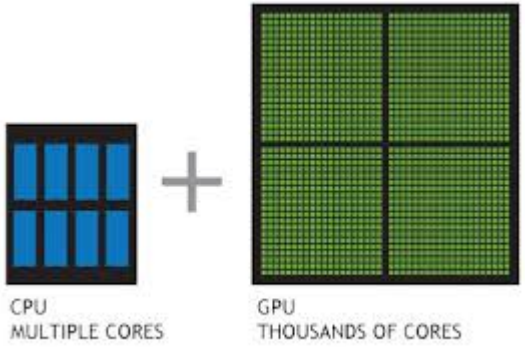


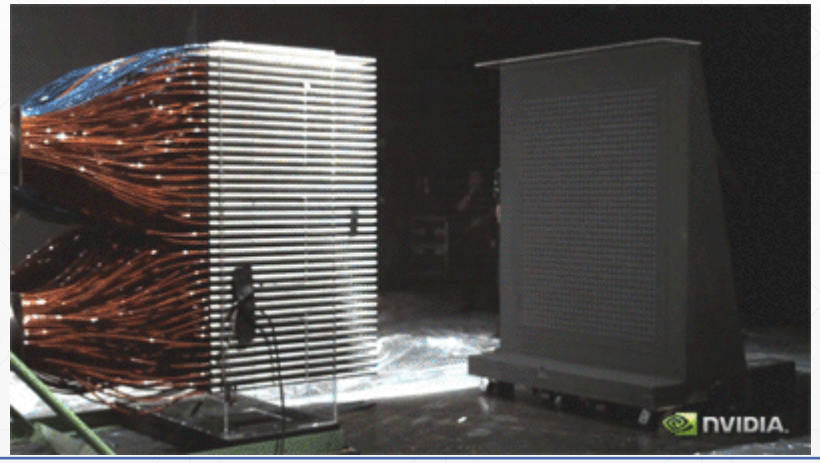
Source: google images

Primary reasons for success of deep learning



- Increase in compute power (Moore's law)
- Use of Graphical Processing Unit for parallel computation





Can we frame this specific control problem as a learning problem?

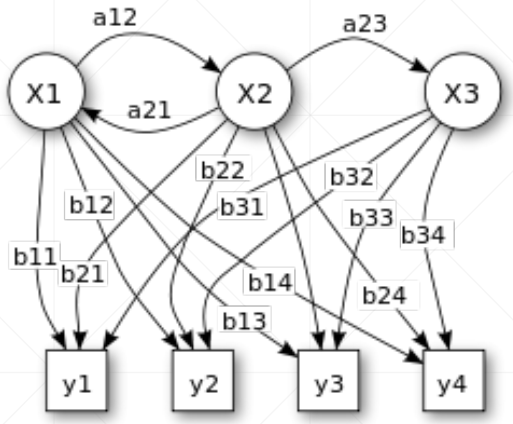
- Artificial Intelligence meets Process Control

- Next Generation exascale Controllers

Our Approach

Our Approach

- Learn complex control behavior of MPC
- Learn function of the mapping from plant output to optimal control input

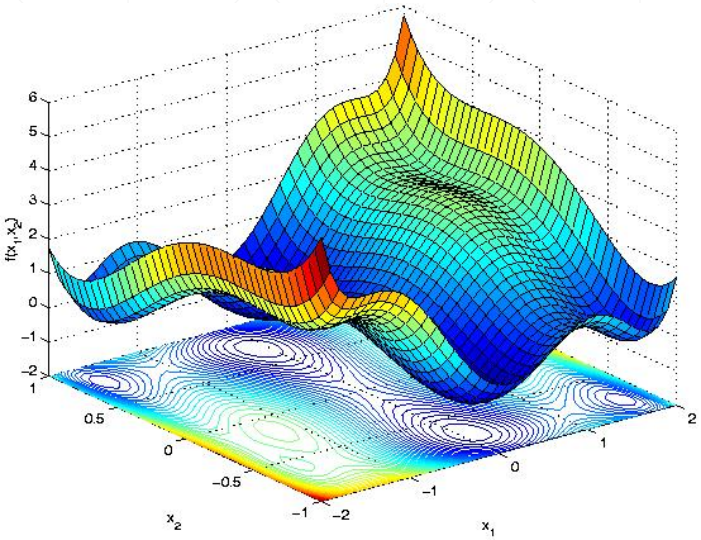


Challenges

- Data collected with respect to time steps are not i.i.d (independently identically distributed)
- Control actions are dependent of current output and past control actions

Novel Architecture

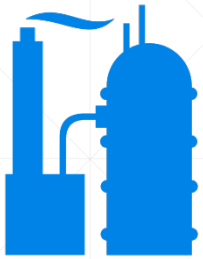
- We propose a novel neural network architecture to learn the complex control behavior



Advantages

- Quick in predicting optimal control actions given plant output
- Does not involve solving optimization objective
- Does not involve estimating the hidden states
- All steps are learnt within hidden units of neural networks

Motivation - Summary



Optimization based control

Drawback

- Computationally demanding (optimization objective)
- Estimating hidden states
- Solving optimization problem 'online' for complex non-linear systems takes time

Current Approach

* Linearize a non-linear system dynamics
* Approximate complex system with simple system
* Does not account full dynamics of the system

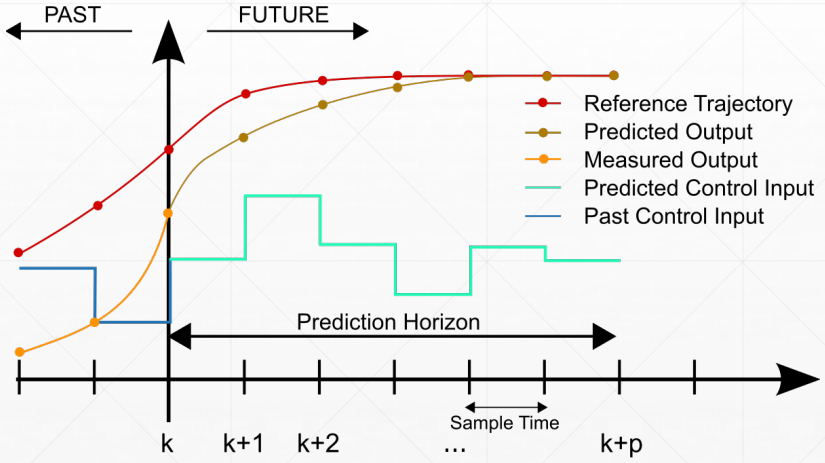
* Learn the complex policies of MPC
* A new novel architecture combining recurrent network and feed forward nets

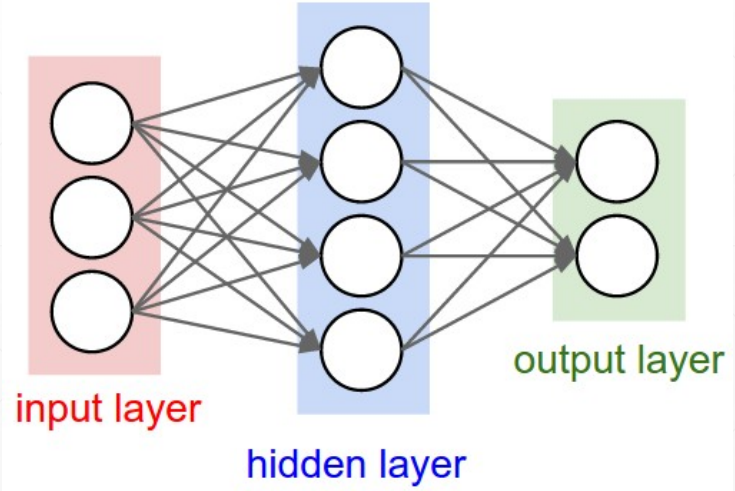
1. Quicker at Test time
2. Does not involve solving optimization problem

Related Works

Related Works

- NN was used to approximate prediction step [1]
- NN was used to approximate non-linear dynamics [2]
- Two tier RNN for solving optimization objective based on linear and quadratic formulations [3]
- NN was used to do the future state prediction steps with policies learnt using Proportional-Integral-Derivative Controller (PID) [4]



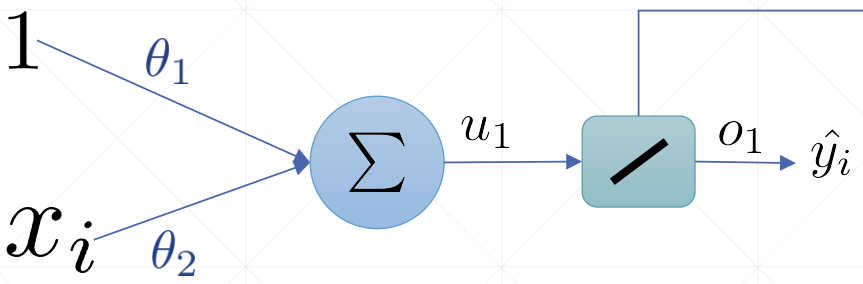


Background

Linear Regression

$$\hat{y}_i = \theta_1 + \theta_2 \times x_i$$

- $\hat{y}_i$ is the predicted output, θ_1 and θ_2 are the parameters (weights) of the linear model



Activation function (linear for linear regression)

Since activation function is linear,

$$\hat{y}_i = o_1 = u_1 = \theta_1 + \theta_2 \times x_i$$

- u_1 - output of neurons before activation
- o_1 - output of neurons after activation

Goal

- Learn the parameters/weights θ_1 and θ_2 using the data we have

Prediction

- If θ_1 and θ_2 are known, Output can be predicted

Learning Algorithm

- The loss is given by,

$$Loss = \sum_{i=1}^{n}(y_i^{true} - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i^{true} - \theta_1 + \theta_2 \times x_i)^2$$

- y_i^{true} is the data we have in our hand
- $\hat{y}_i$ is the data that our model predicts

Find the values of θ_1 and θ_2 where loss is minimum,

$$\underset{\theta_1,\theta_2}{\text{minimize}} \sum_{i=1}^{n}(y_i^{true} - \theta_1 + \theta_2 \times x_i)^2$$

Goal

- We want our predictions to be closer to actual data

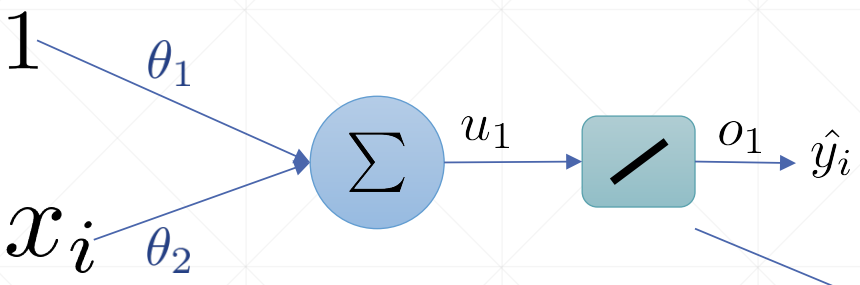
Summary

- If θ_1 and θ_2 are known, output can be predicted

Drawback - Linear Regression

$$\hat{y}_i = \theta_1 + \theta_2 \times x_i$$

- $\hat{y}_i$ is the predicted output, θ_1 and θ_2 are the parameters (weights) of the linear model



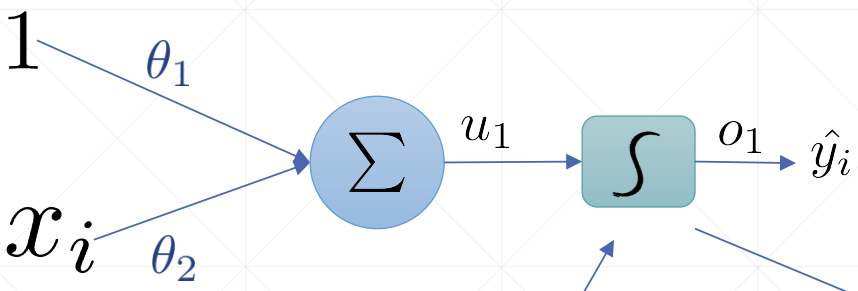
$$\hat{y}_i = o_1 = u_1 = \theta_1 + \theta_2 \times x_i$$

Linear Regression - Output is not bounded to [0,1]

Logistic Regression

$$\hat{y}_i = \frac{1}{1+e^{-(\theta_1+\theta_2 \times x_i)}}$$

- $\hat{y}_i$ is the predicted output, θ_1 and θ_2 are the parameters (weights) of the linear model

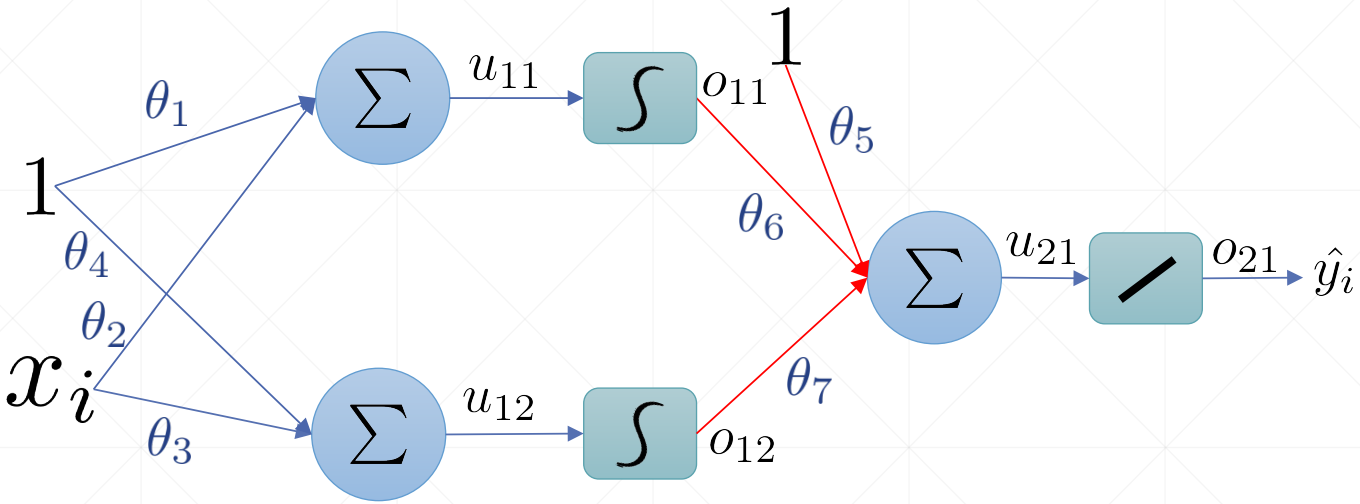


$$\hat{y}_i = o_1 = \frac{1}{1+e^{-u_1}} = \frac{1}{1+e^{-(\theta_1+\theta_2 \times x_i)}}$$

Linear Regression - Output is bounded to [0,1]

Sigmoid function, $f(x) = \frac{1}{1+e^{-x}}$

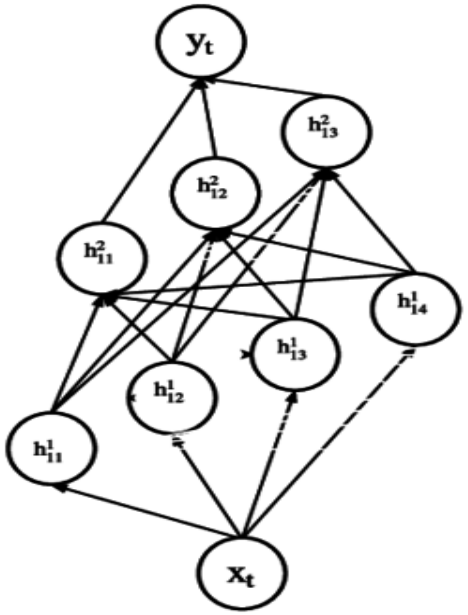
Neural Networks – For Regression

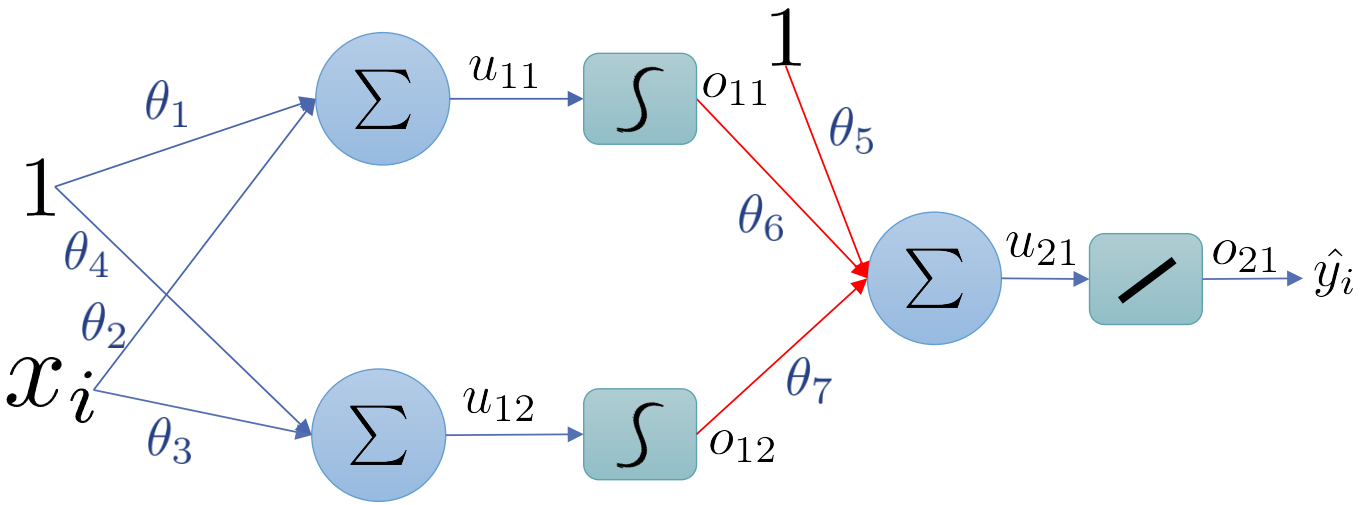


- $u_{11} = \theta_1 \times 1 + \theta_2 \times x_i$
- $u_{12} = \theta_2 \times 1 + \theta_3 \times x_i$
- $o_{11} = \frac{1}{1+e^{-u_{11}}}$
- $o_{12} = \frac{1}{1+e^{-u_{12}}}$
- $u_{21} = \theta_5 \times 1 + \theta_6 \times o_{11} + \theta_7 \times o_{12}$

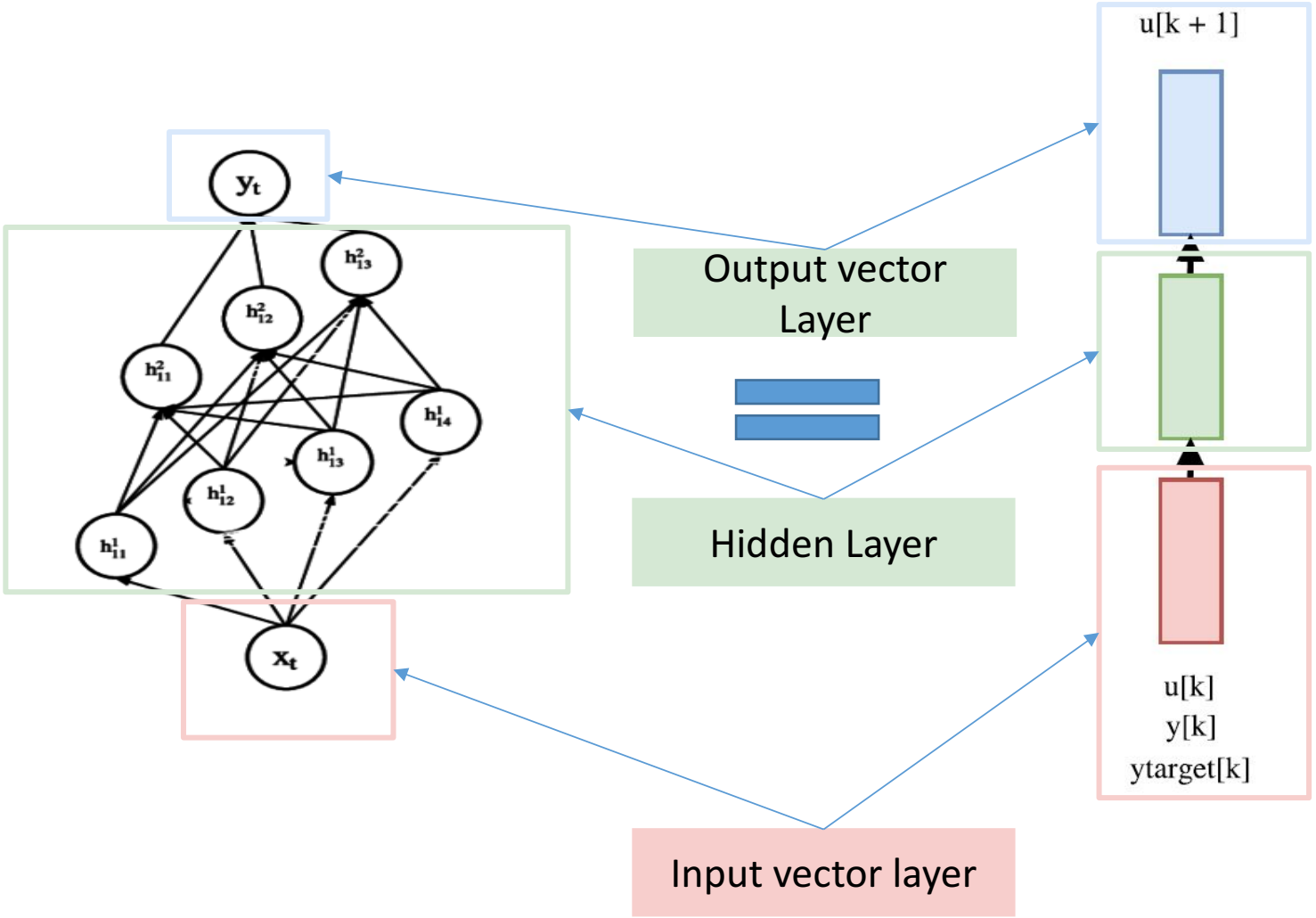
$\hat{y}_i = o_{21} = u_{21}$

Neural Network Compact Representation

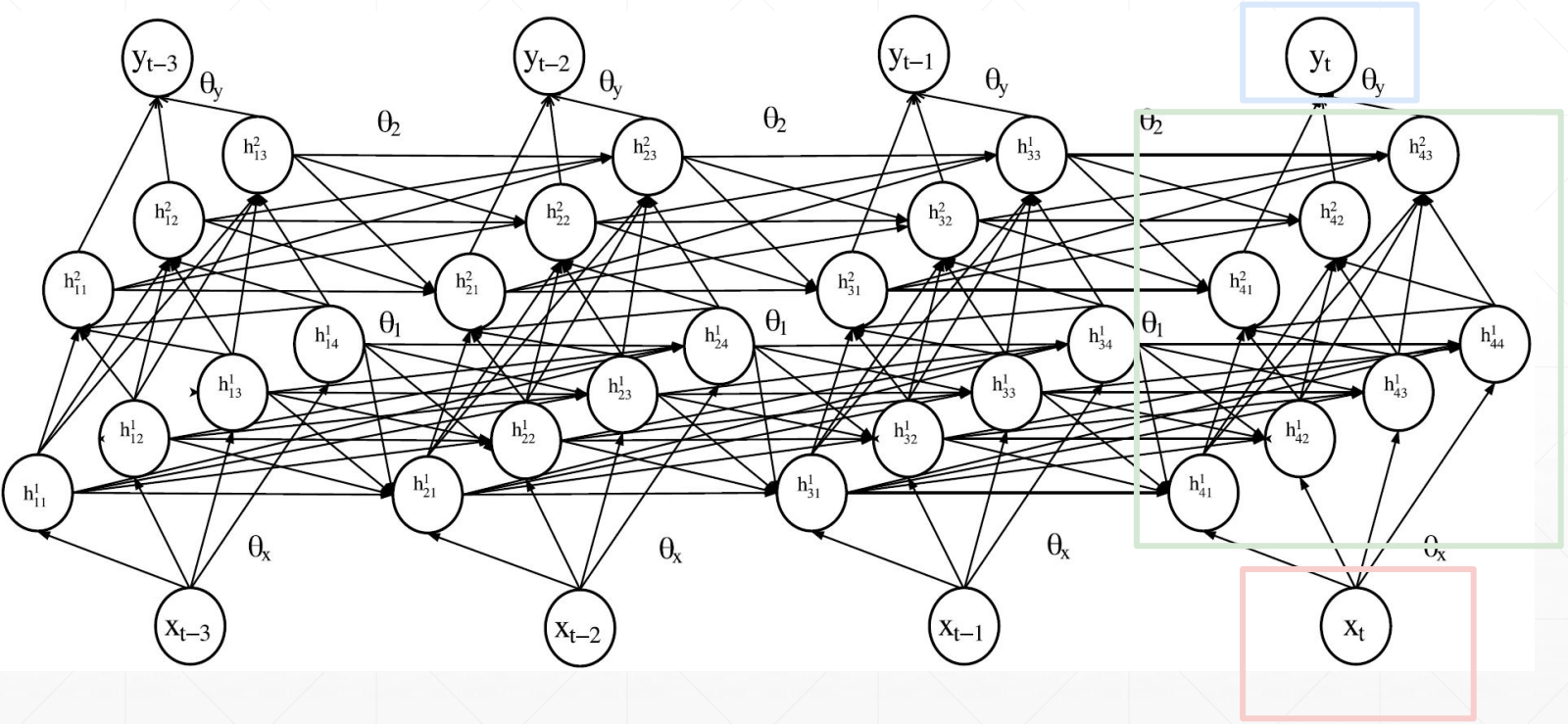


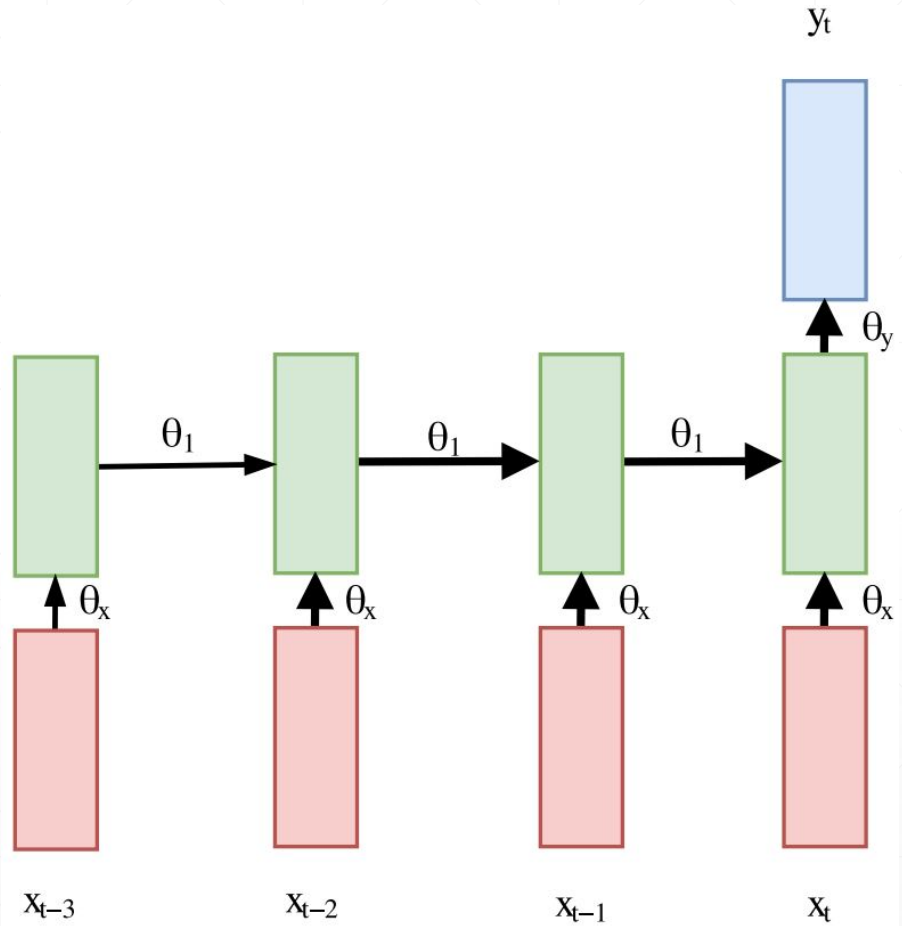


Notation


y_t
h^2_{13}
h^2_{12}
h^2_{11}
h^1_{14}
h^1_{13}
h^1_{12}
h^1_{11}
x_t
Output vector Layer
Hidden Layer
Input vector layer
u[k + 1]
u[k]
y[k]
ytarget[k]

Recurrent Neural Network (RNN)





Drawback of Recurrent Neural Networks

- Because of many layers in time steps,

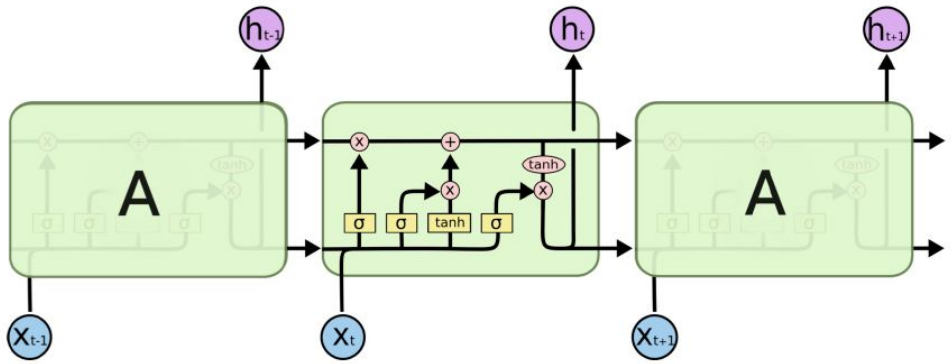
  (i) if gradient is less than zero, while backpropogating to first layer the gradients becomes zero ('vanishing gradient')

  (ii) If gradient is gradient is greater than 1, the gradient explodes ('exploding gradient')

- To overcome that 'Long Short Term Memory' was proposed. In addition to hidden state vector it uses memory vector to overcome the 'vanishing' and 'exploding' gradient problems.

- In terms of architecture, LSTM is similar to RNN but with extra model complexity to handle 'vanishing' and 'exploding' gradient problems

Long Short Term Memory(LSTM)



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad forget\ gate \tag{2}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad input\ gate \tag{3}$$

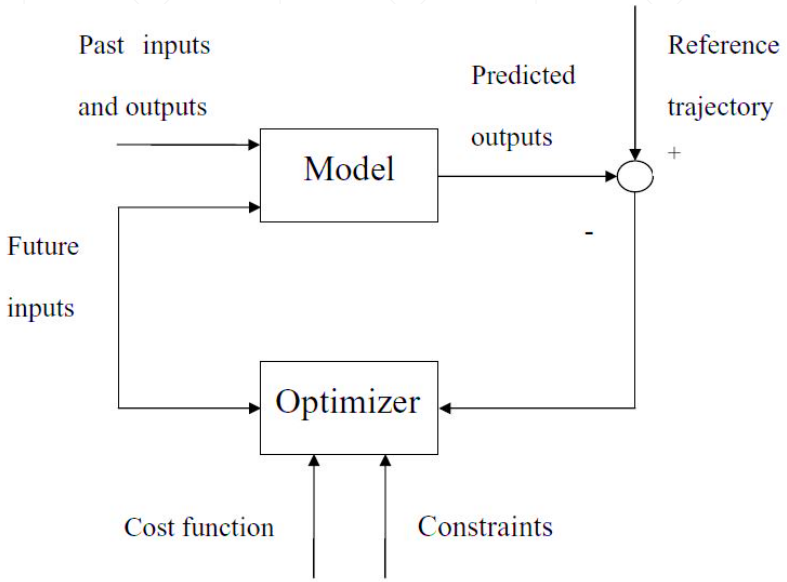
$$\widetilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \quad input\ state \tag{4}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \quad cell\ state \tag{5}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad output\ gate \tag{6}$$

$$h_t = o_t \odot tanh(C_t) \quad hidden\ state \tag{7}$$

Model Predictive Control

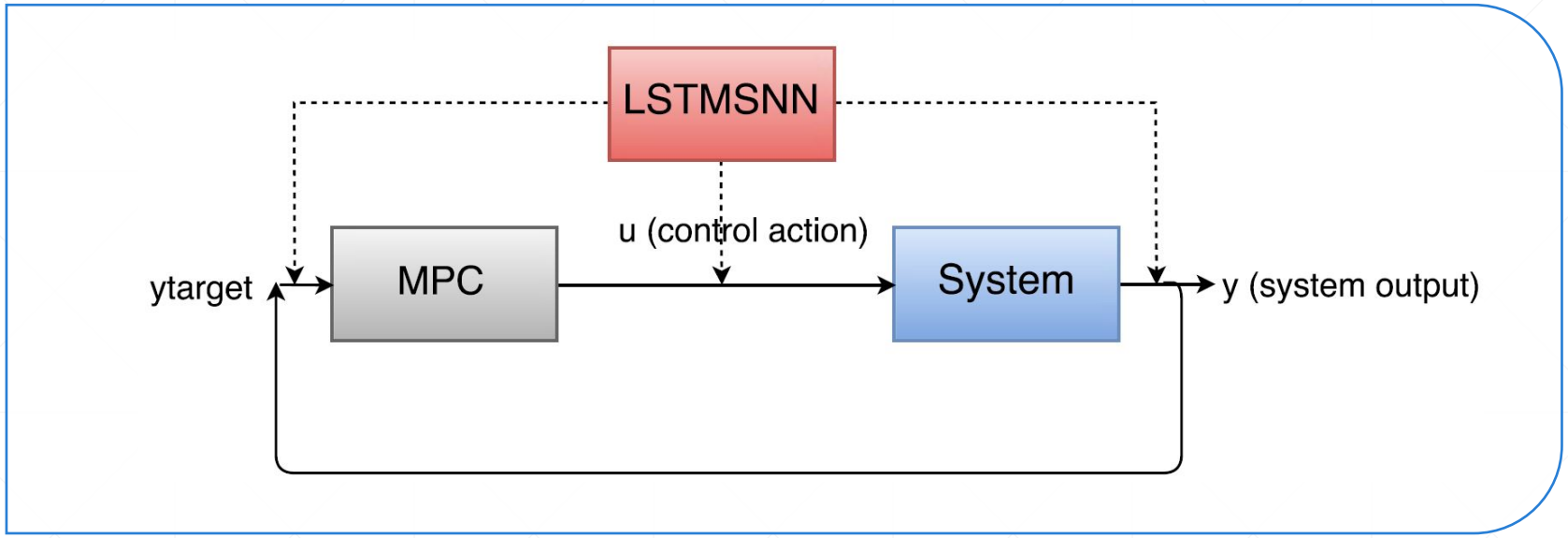


- Objective function: $$\min_{u} \sum_{i=1}^{N} w_{y_i}(ytarget - y_i)^2 + \sum_{i=1}^{N} w_{u_i} \Delta u_i^2$$
- y – outputs, w – weighing coefficient , u – inputs , N – Prediction horizon

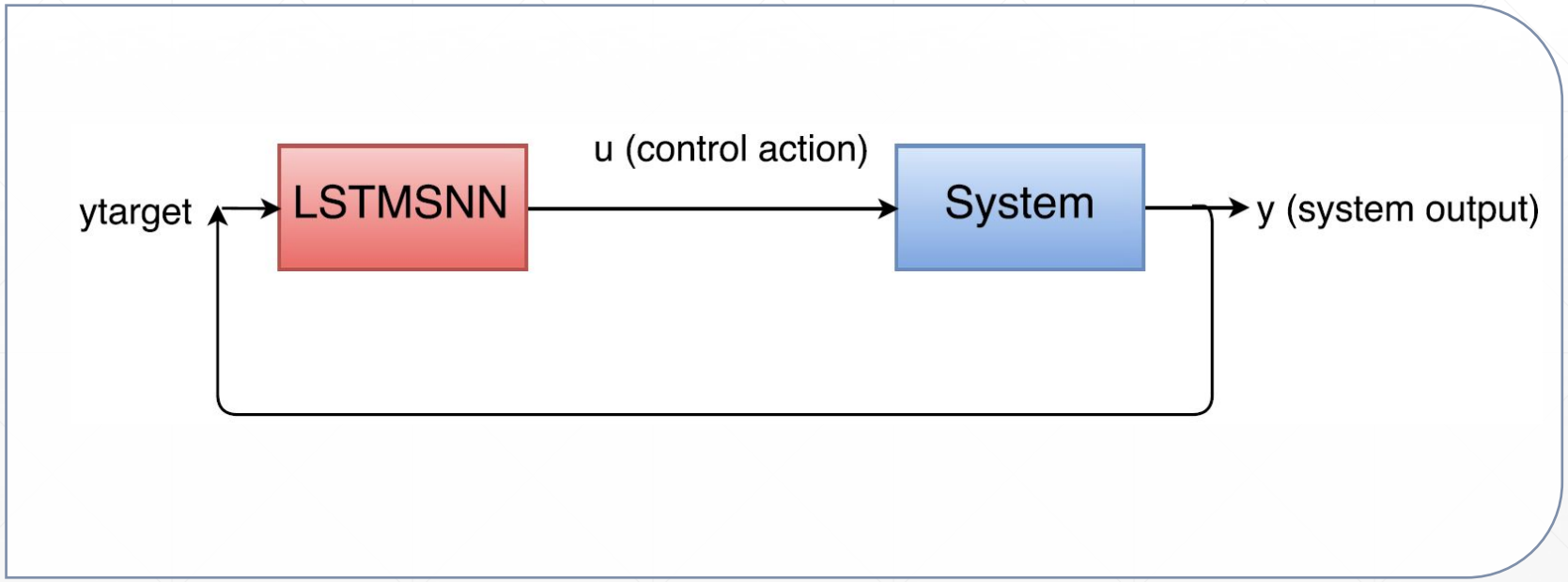
System Setup

System Setup

Training

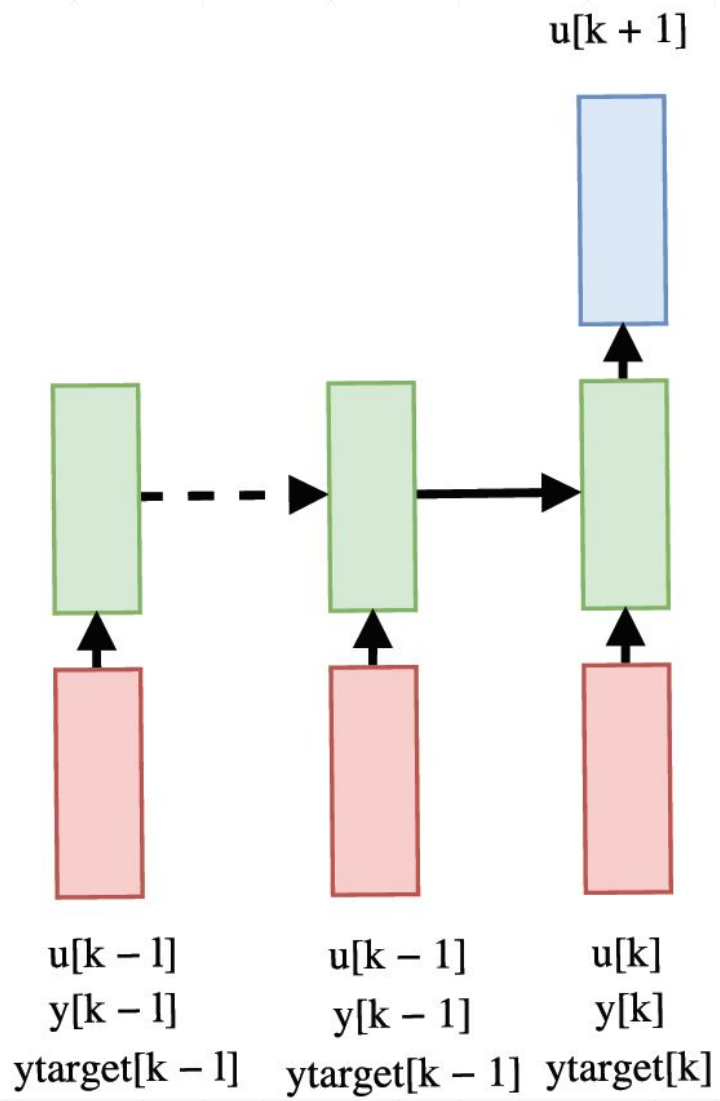


Testing

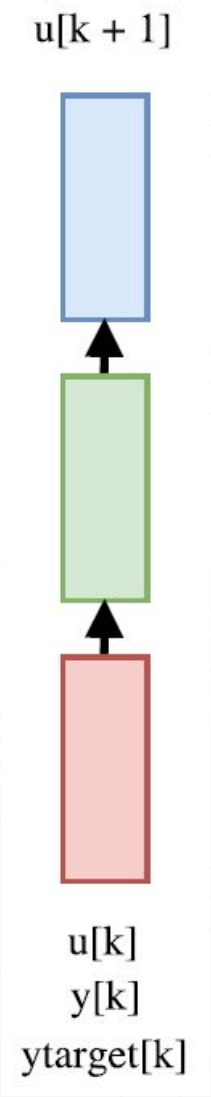


Model Architectures

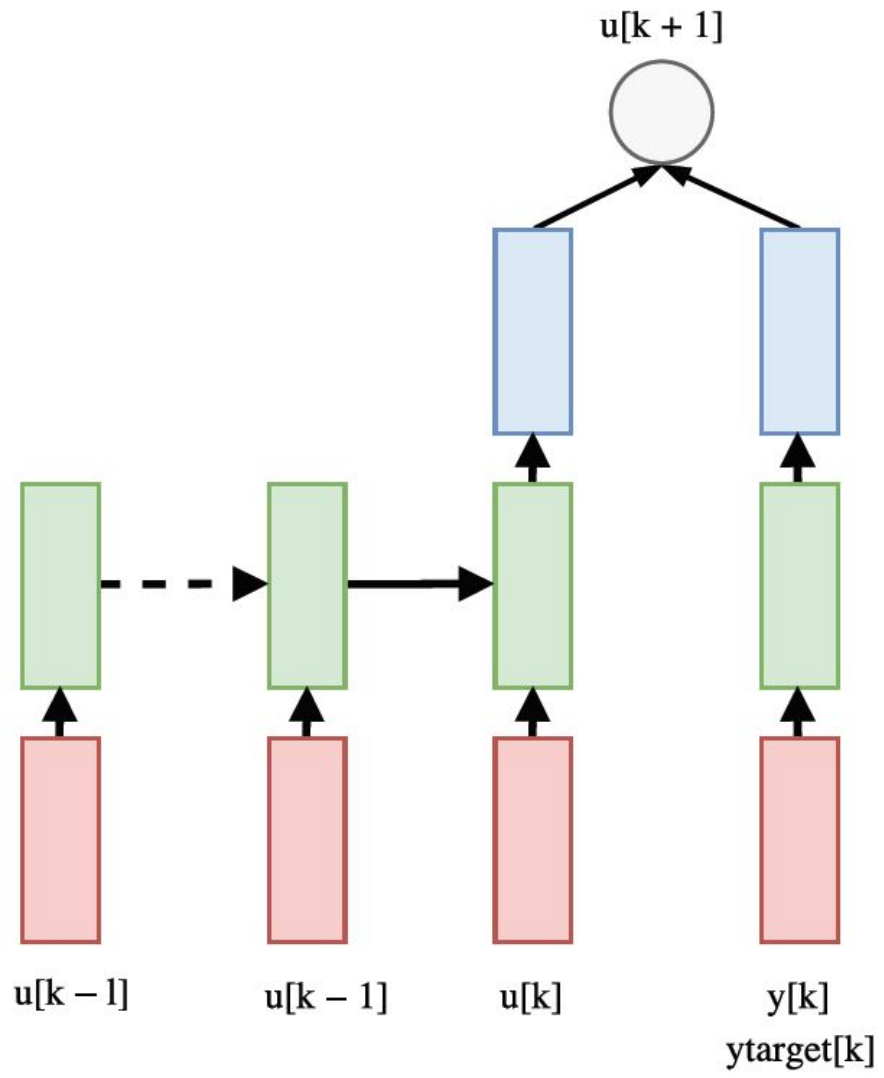
Model 1 – Long Short Term Memory



Model 2 – Neural Networks



Model 3 – New Architecture - LSTMSNN



Simulation Results

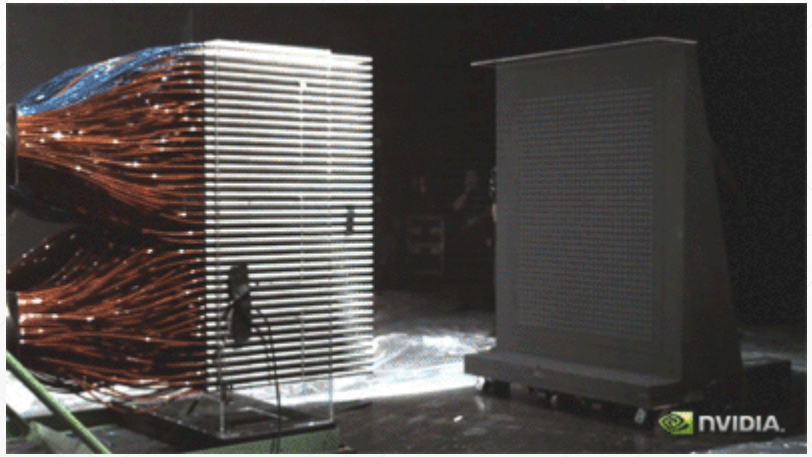
Experimental Setup – Physical System

- Control of manufacture of paper in a paper machine
- Target output, y_{set} , is the desired moisture content of the paper sheet
- Control action, u, is the steam flow rate
- System output, y, is the current moisture content
- Model,

$$G(z) = \frac{0.05z^{-4}}{1 - 0.6z^{-1}}$$

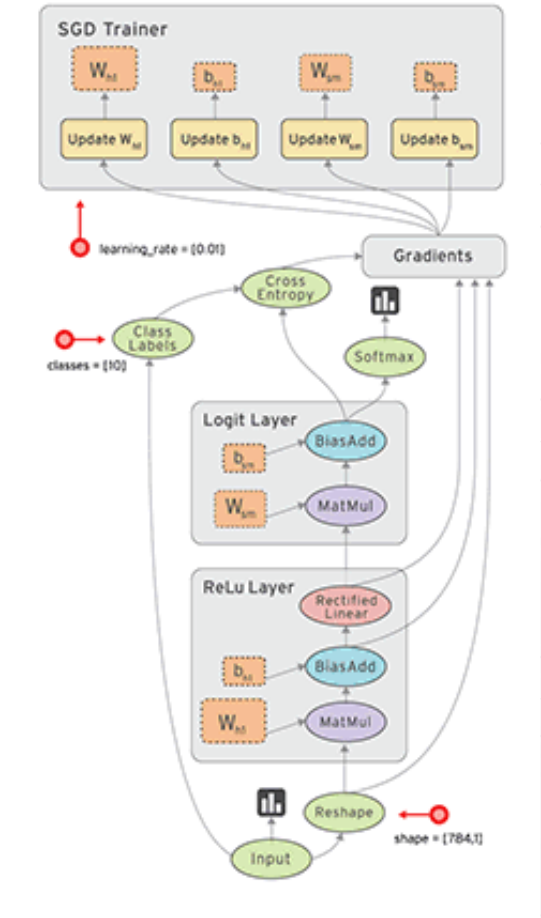
Implementation

- Basic Python numpy implementation with cudapy for GPU computation
- Trained and tested on NVIDIA Geforce 960M 4GB
- MPC simulation data were generated in Matlab
- Extended implementation with tensorflow



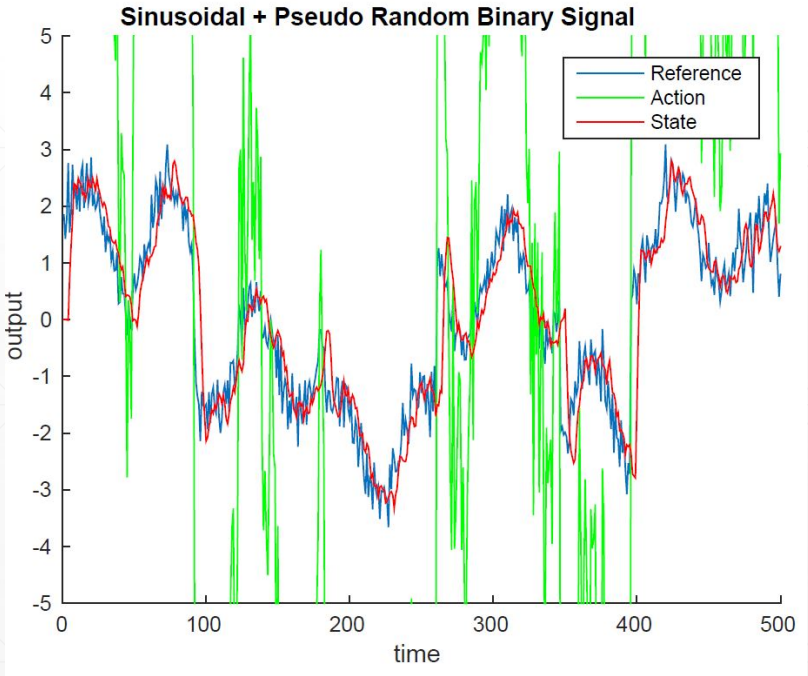
python™

TensorFlow



Data Collection

- Training Samples - 100,000 points
- Random jump – {(-3, -0.2), (0.2,3)}
- sine function with period (10, 1000)
- Gaussian noise with noise to signal ratio of 10
- Sine and Random jump combined fed as a reference trajectory to MPC

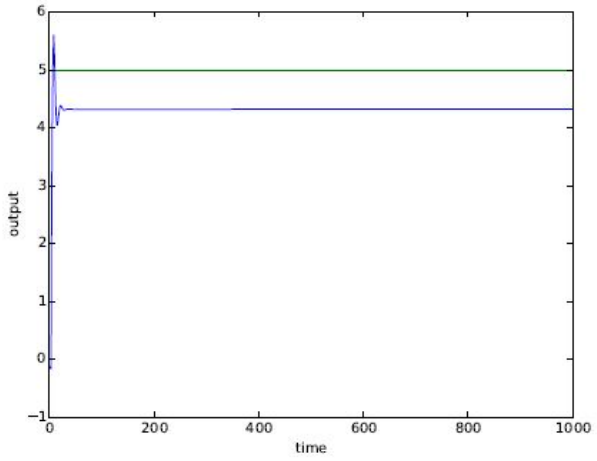


Training

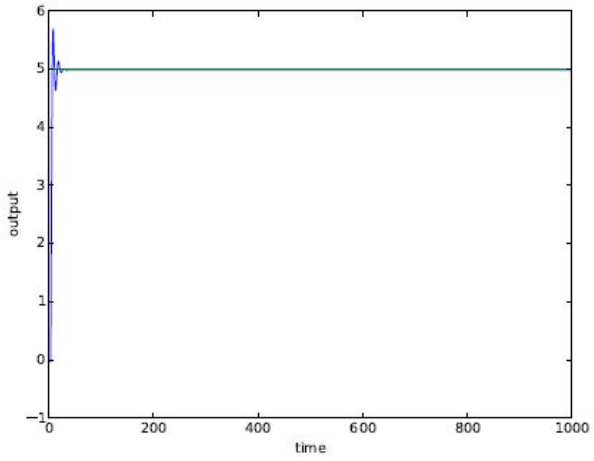
- Trained using RMSprop [4] Optimizer
- Compared LSTM, NN, LSTMSNN
- Details of training

Model	Optimizer	Number of layers	Sequence Length
LSTMSNN	RMSprop	2 layers of LSTM and 4 layers of NN	5
LSTM-only	RMSprop	2	5
NN-only	Stochastic Gradient Descent	3	N/A

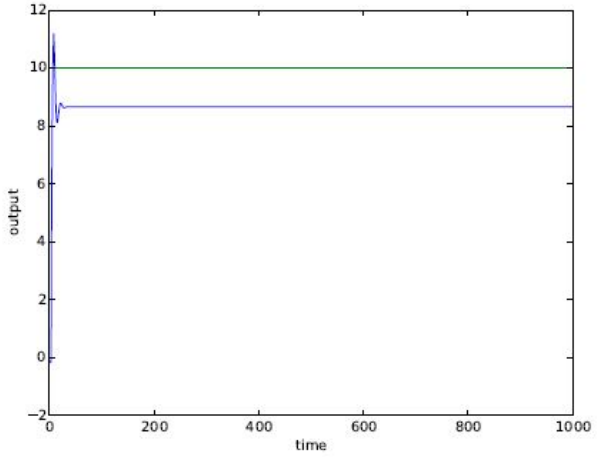
Simulation Results



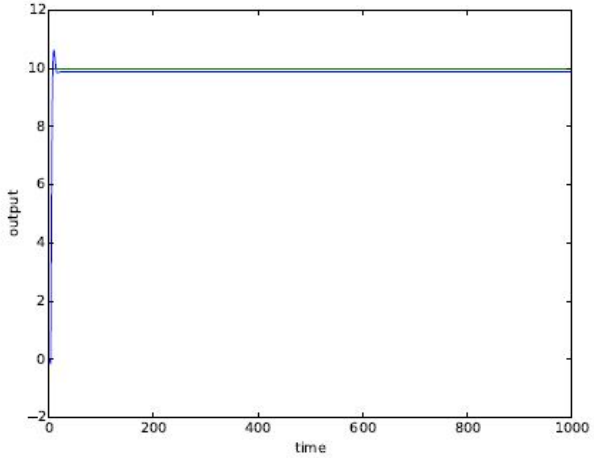
(a) NN-only's system output for target output=5



(b) LSTMSNN's system output for target output=5

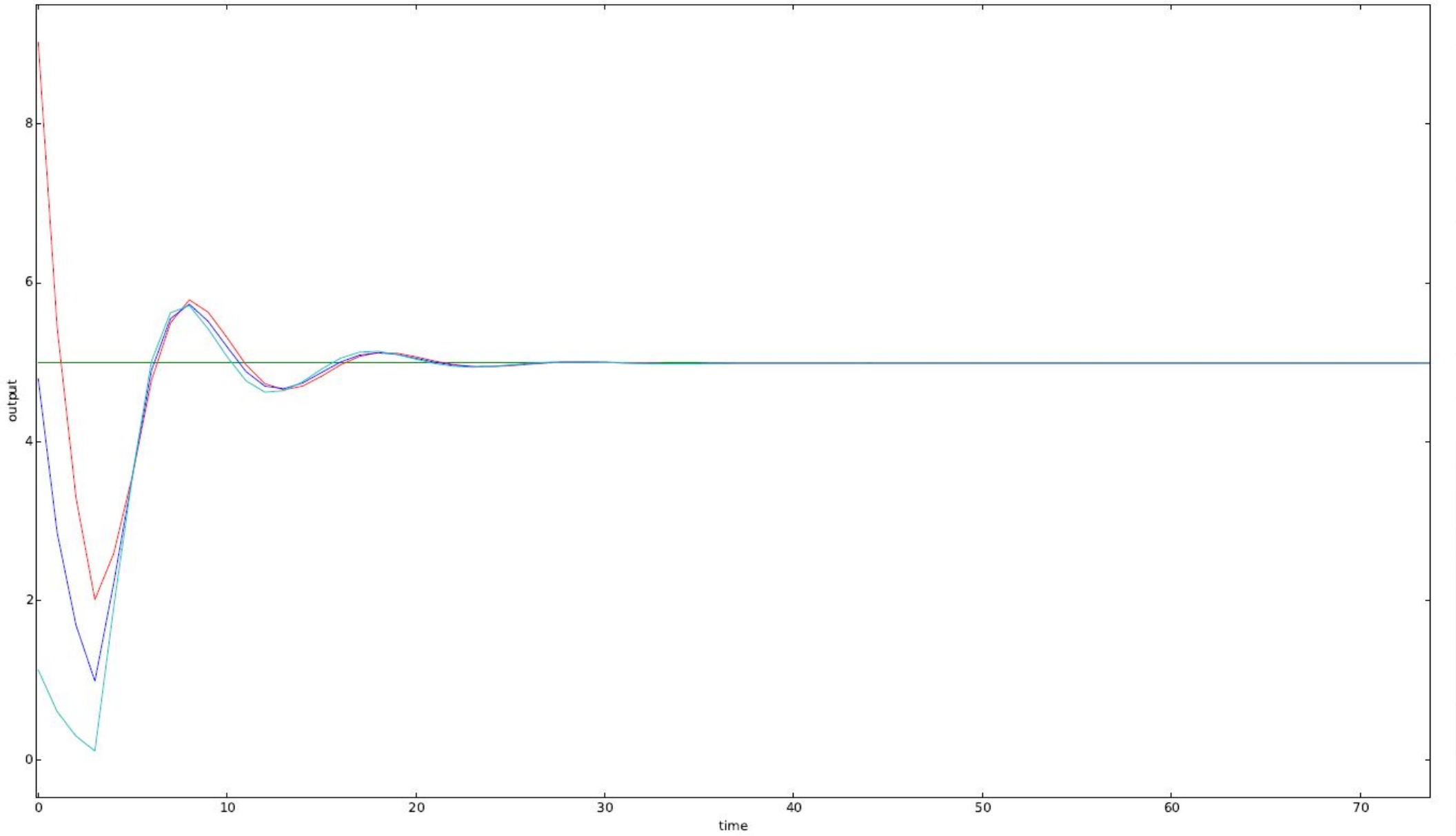


(c) NN-only's system output for target output=10

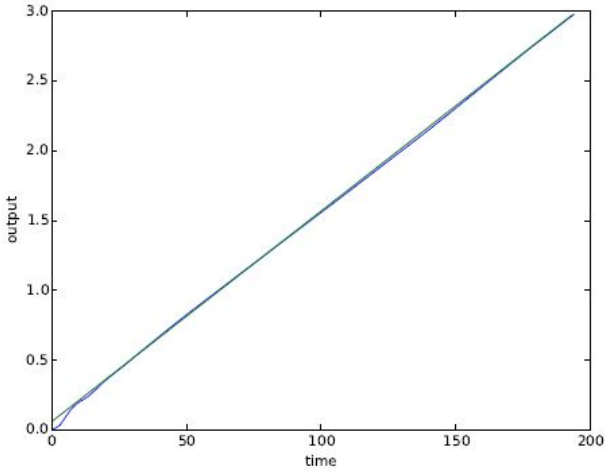


(d) LSTMSNN's system output for target output=10

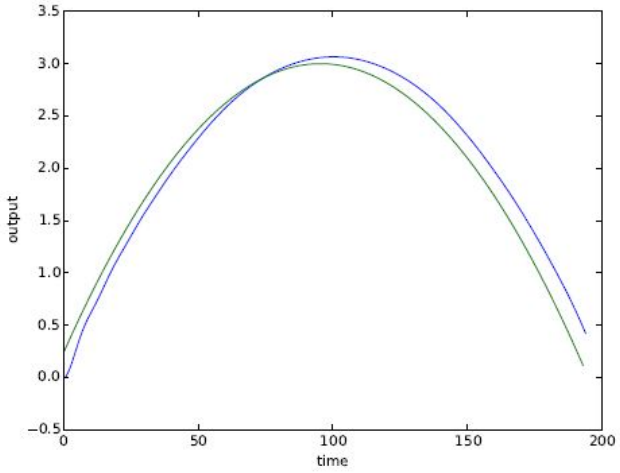
Simulation Results



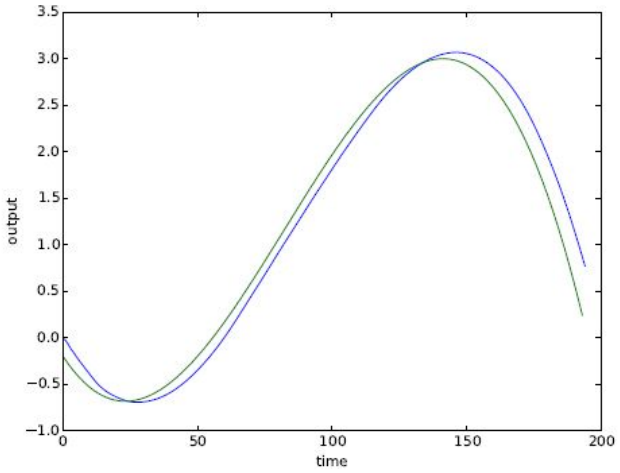
Simulation Results



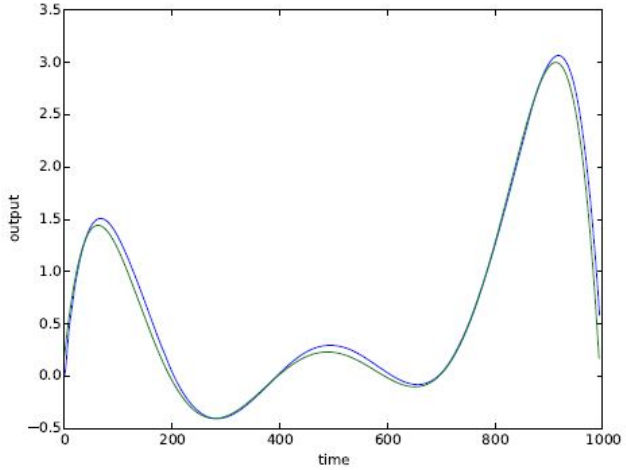
(a) Target output is linear



(b) Target output is quadratic



(c) Target output is cubic



(d) Target output is polynomial of degree is 6

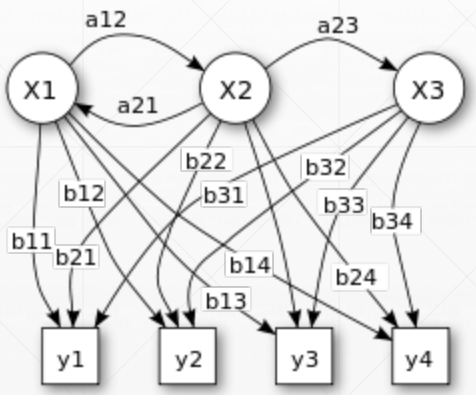
Performance Comparison

Model	Target Output	MSE	OE
LSTMSNN	2	0.06	0.01
NN-only	2	0.07	0.23
LSTM-only	2	5.56	Did not converge
LSTMSNN	5	0.078	0.01
NN-only	5	0.53	0.7
LSTM-only	5	62.42	Did not converge
LSTMSNN	10	0.01	0.01
NN-only	10	2.21	1.3
LSTM-only	10	167.78	Did not converge

Observations

Observations

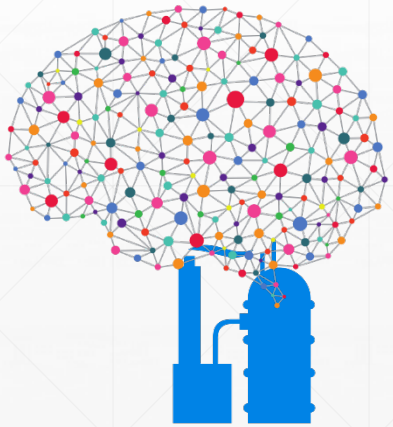
- Predicting hidden states step is learnt as an abstract information in the hidden units of neural network
- Optimization step is learnt as an abstract information in the hidden units of neural network
- MPC prediction step is learnt within the hidden units and now prediction model is learnt directly with plant output without the need of hidden states
- Computing with GPU makes it relatively fast



Conclusion

Conclusion

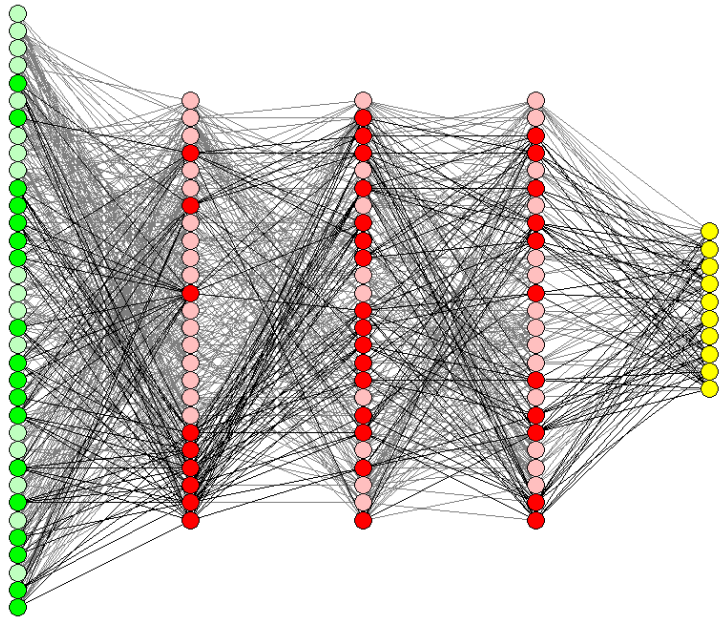
- Combined deep learning with process control
- Different Neural Network architecture ‘LSTMSNN’
- Eliminates burden of state estimation, optimization step, prediction step are learn in hidden layers of neural networks
- Computing with GPU makes it relatively fast
- Trained model can be deployed in production to gain advantage of faster retrieval of optimal inputs

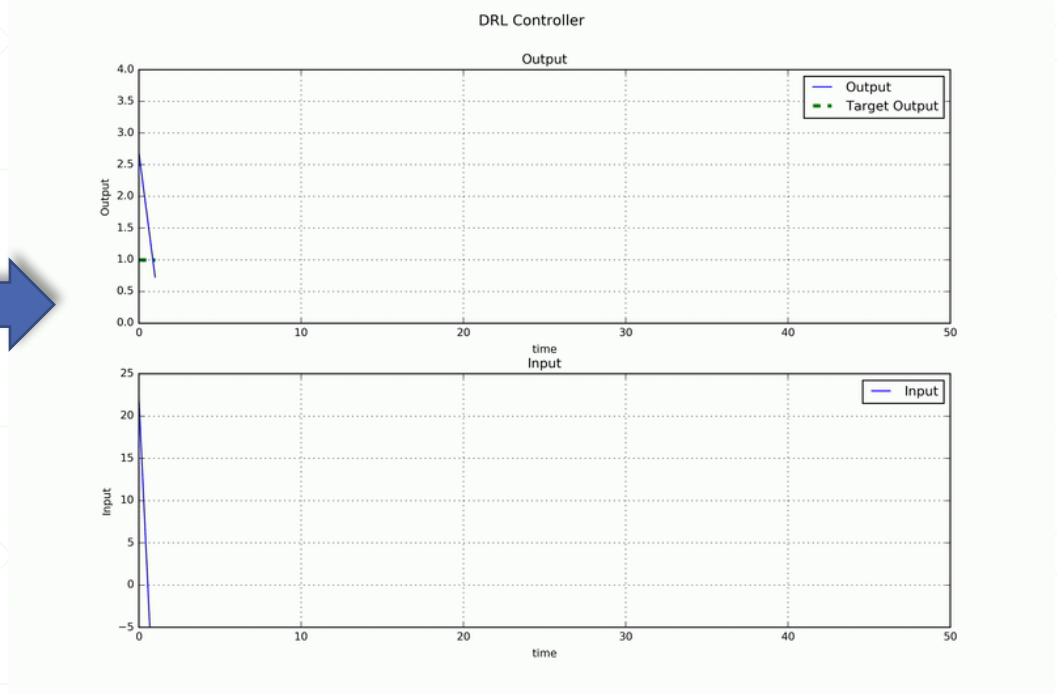


References

1. Paisan Kittisupakorn, Piyanuch Thitiyasook, M.A. Hussain, Wachira Daosud, "Neural Network based model predictive control for a steel picking process", Journal of Process Control
2. Piche, Stephen, et al. "Neural network based model predictive control." Advances in Neural Information Processing Systems. 2000
3. Yunpeng Pan and Jun Wang, "Two Neural Network Approaches to Model Predictive Control" , American Control Conference, 2008
4. Draeger, Andreas, Sebastian Engell, and Horst Ranke. "Model predictive control using neural networks." Control Systems, IEEE 15.5 (1995): 61-66.

Thank you




DRL Controller
Output
Output
Target Output
Output
time
Input
Input
Input
time