# Neural ODE-based Model Predictive Control

Luke Roberto and Anupam Kumar

# Agenda

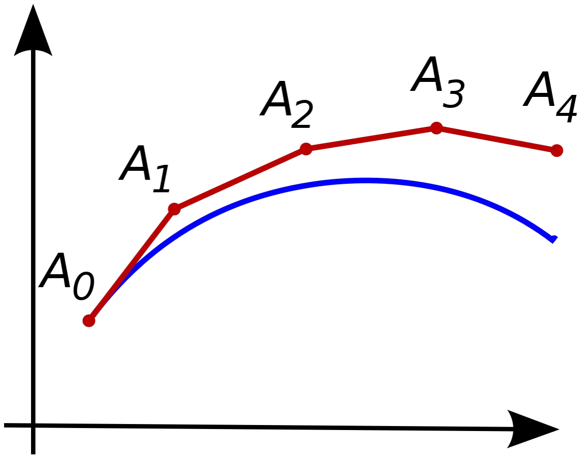
- Introduction
- Literature Review
- Methodology
- Experimental Results
- Discussion

# Introduction

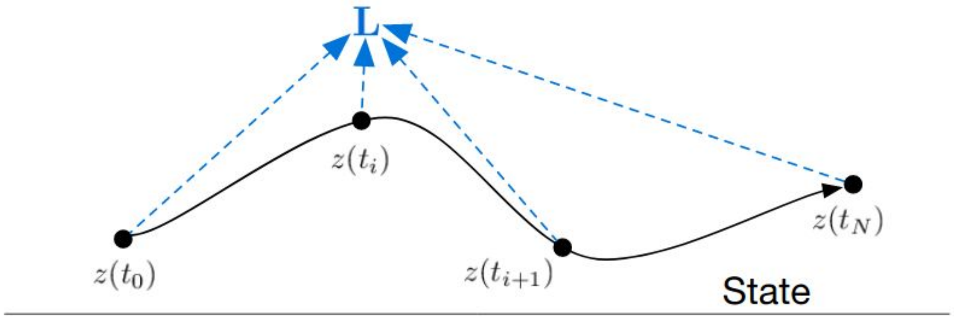
- Robotic planning is difficult
- Need high quality models in order to effectively plan
- Real world is continuous
- How can we encode this inductive bias in our models?

# Literature Review

*Neural ODEs*




RNNs are an Euler approximation to a differential equation




Define a loss on the states and back-propagate ODE solver



Figure reference: Neural Ordinary Differential Equations https://arxiv.org/pdf/1806.07366.pdf

# Literature Review

*Learned Transition Models*

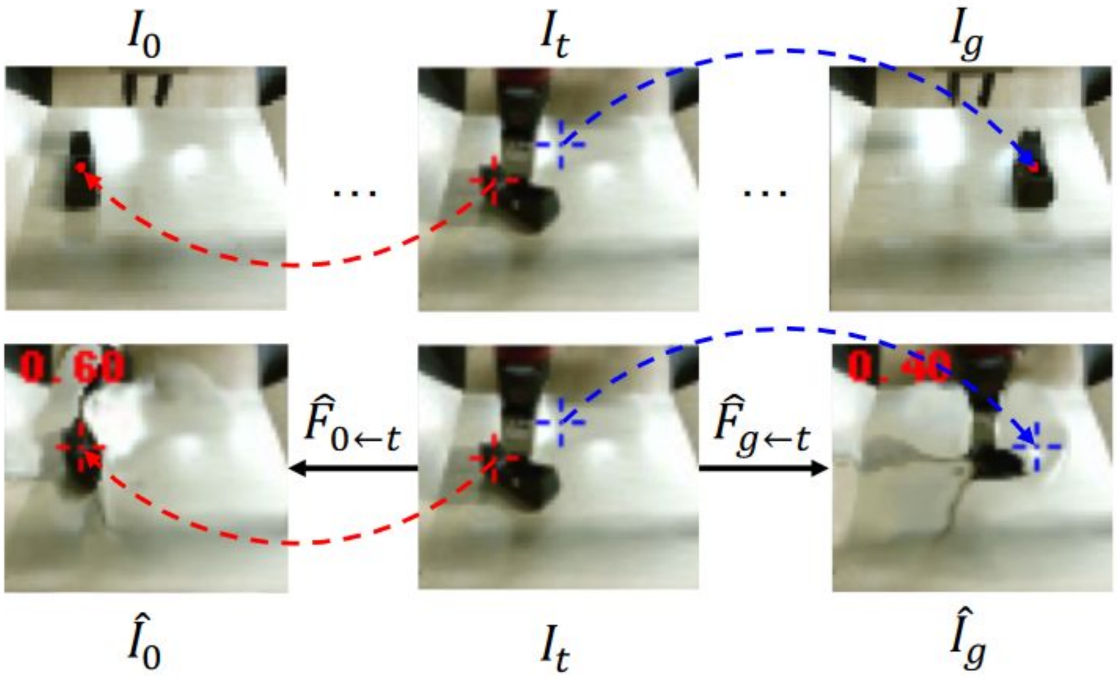
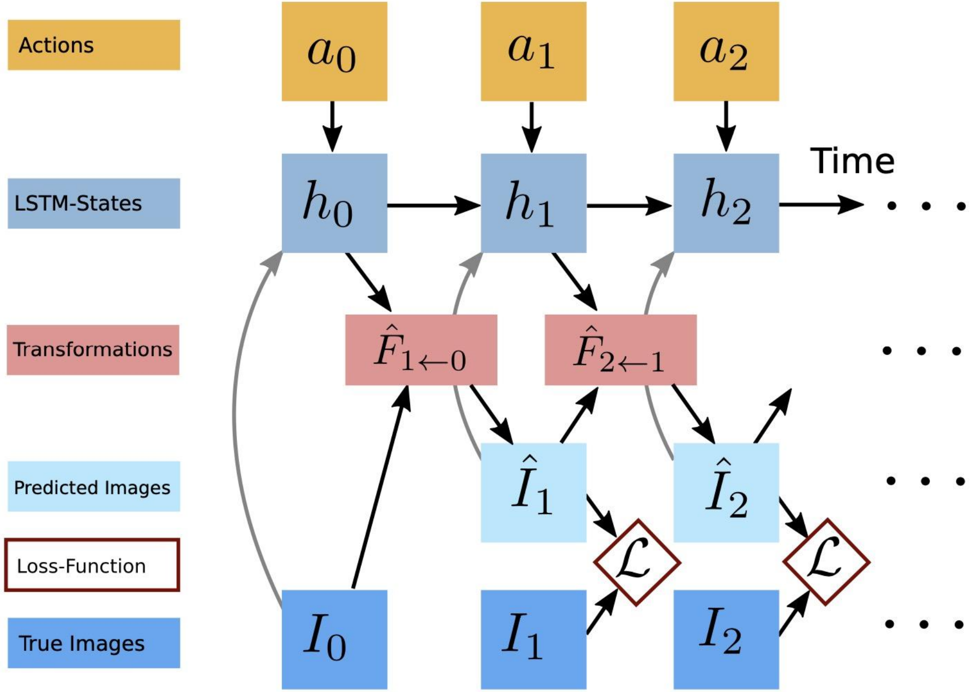





Figure reference: Visual Foresight: Model based deep RL https://arxiv.org/pdf/1812.00568.pdf

# Literature Review

*Learned Transition Models*



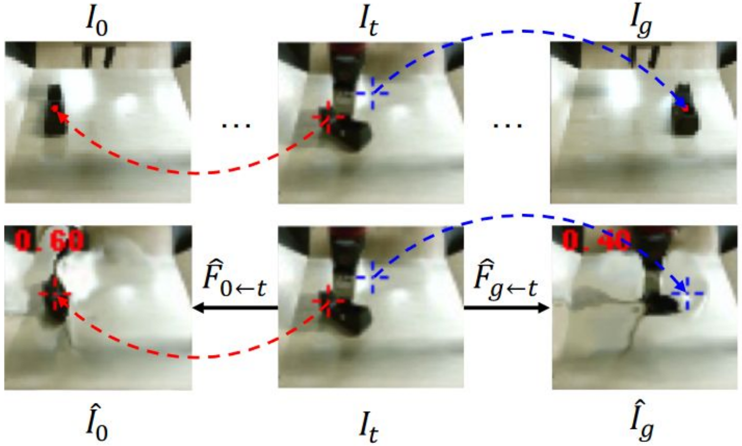
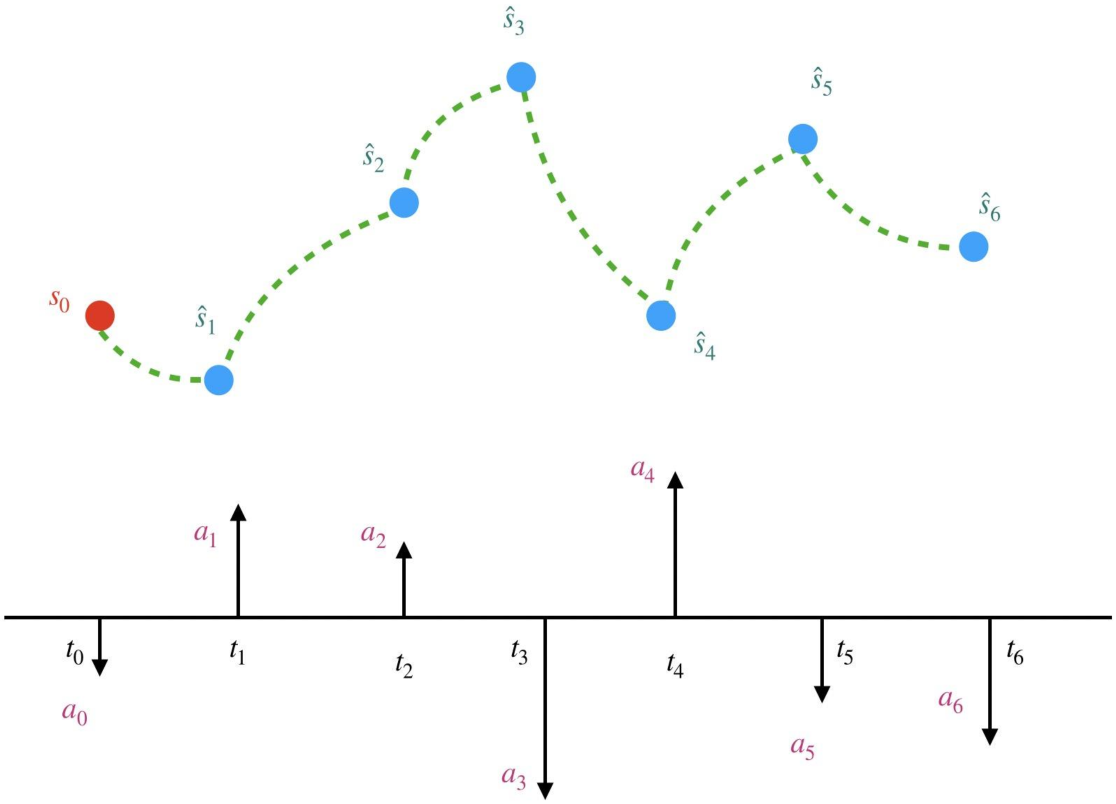






Figure reference: Visual Foresight: Model based deep RL https://arxiv.org/pdf/1812.00568.pdf

# Methodology

*Learning Transition Models*



# Methodology

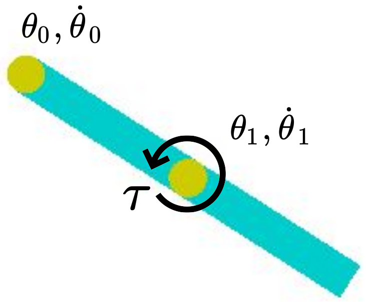
*Model Predictive Control*

**Algorithm 1:** Model Predictive Control

**Input :** Predictive model $g$, planning cost function $c$
**output :** $a_t$, predicted best action
**for** $t = 0 ... T - 1$ **do**
  **for** $i = 0 ... n_{iter} - 1$ **do**
    **if** $i == 0$ **then**
      Sample M actions over the horizon $a_{t:t+H-1}^{(m)}$ from a uniform distribution over our discrete action space ;
    **else**
      Sample M actions over horizon from updated distribution over action space. ;
    **end**
    Check if actions are valid, otherwise resample. ;
    Use $g$ to predict future state sequence $\hat{s}_{t:t+H-1}^{(m)}$. ;
    Evaluate each action sequence with cost function, $c$. ;
    Update action distribution with lowest cost action distribution. ;
  **end**
  return $a_t^*$, first action from best sequence;
**end**

# Experimental Setup

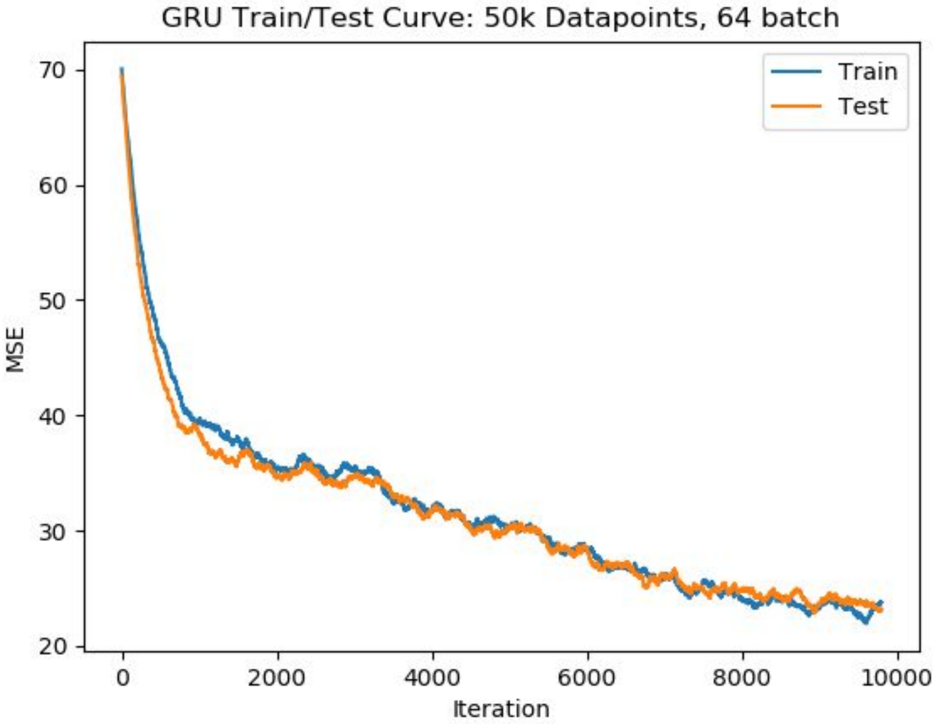
Acrobot

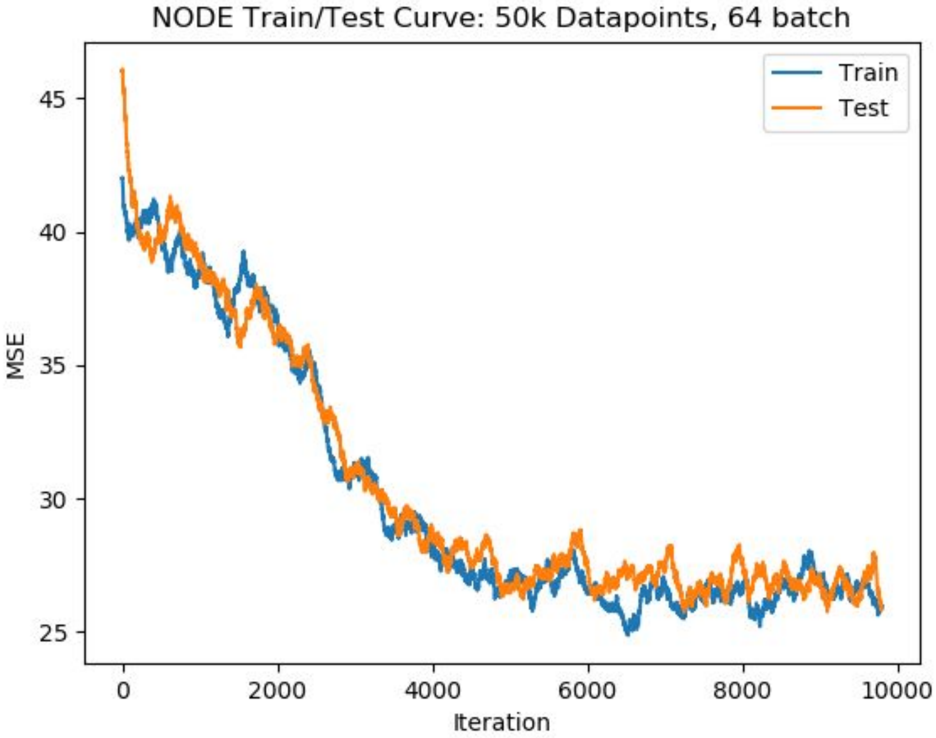



- Double pendulum
- Simple environment, with sufficiently complex dynamics
- Encouraged diversity with random starts

# Experimental Results

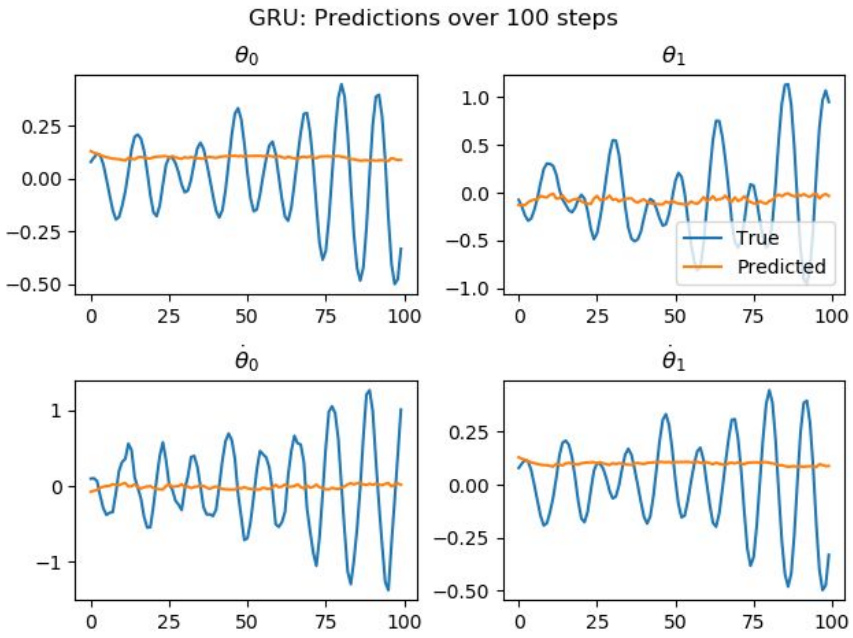
*Model Learning*

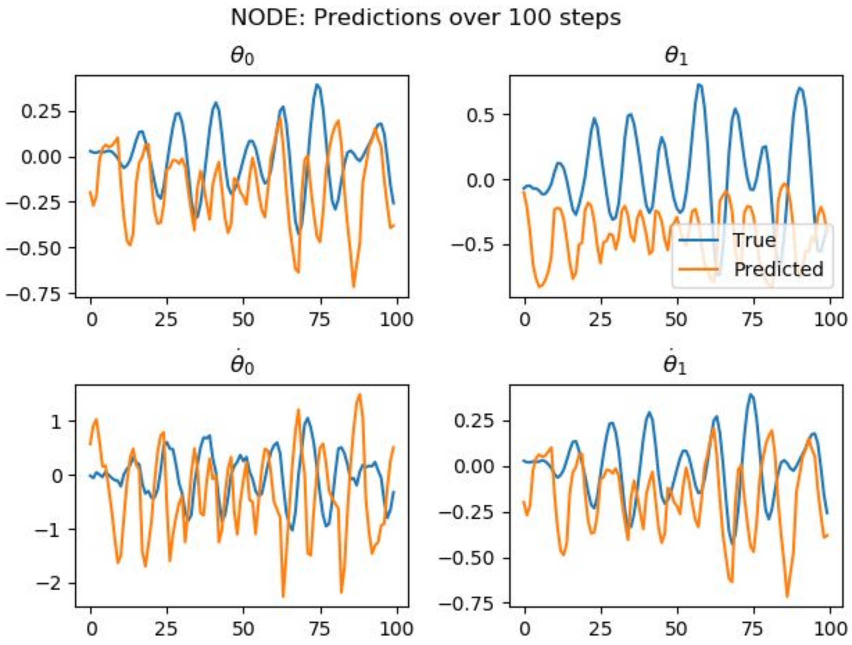


# Experimental Results

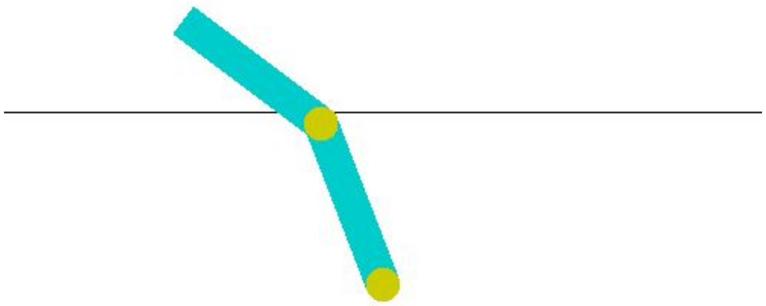
*Model Learning*



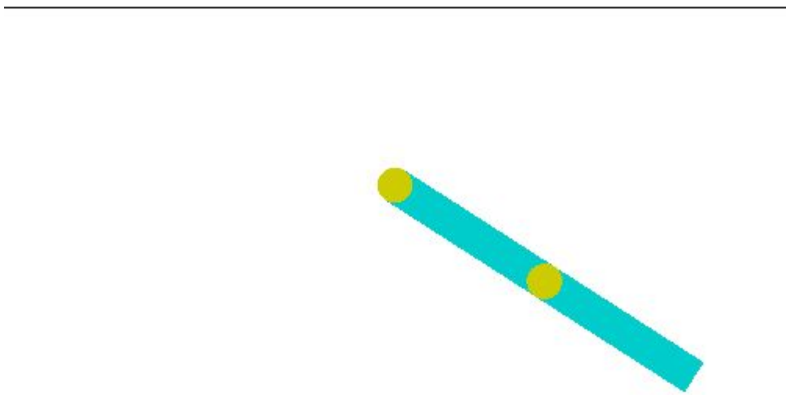



# Experimental Results

*MPC Swing-Up Experiments*



Predicted Trajectory



Actual Trajectory

# What We Learned

*And what we want to try next!*

- Neural odes are hard to train
    - dynamics models in general are hard to learn
- We plan to extend the environments we want to tackle, perhaps those that are non markov
- Use an encoder structure to learn a latent dynamics model to handle pixel level information