University of Stuttgart
Institute for Computational Physics

SimTech

Topical Meeting – Machine Learning

Daniel Fink

Simulating Stochastic Processes with Quantum Devices

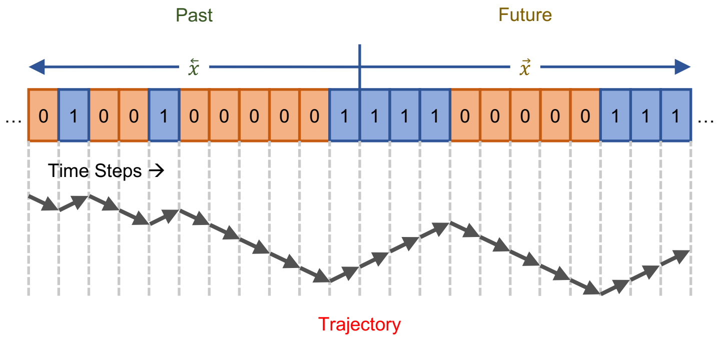
\+

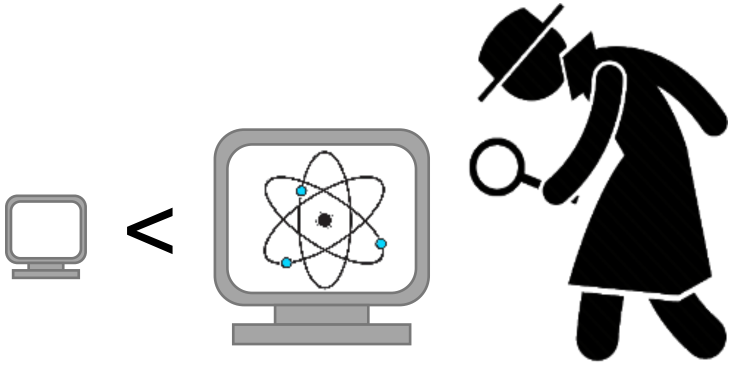
Robust Quantum Algorithms

October 5th, 2022

Agenda

- Stochastic Processes
  - Quick repetition
  - Obstacles
  - Solutions
  - Results
- Robust Algorithms
  - Idea





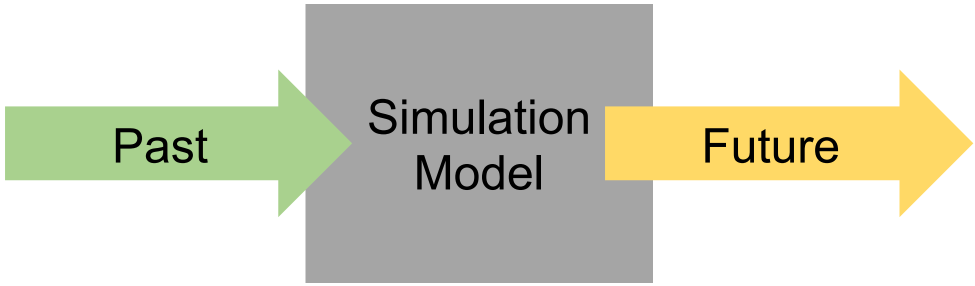
Simulating Stochastic Processes

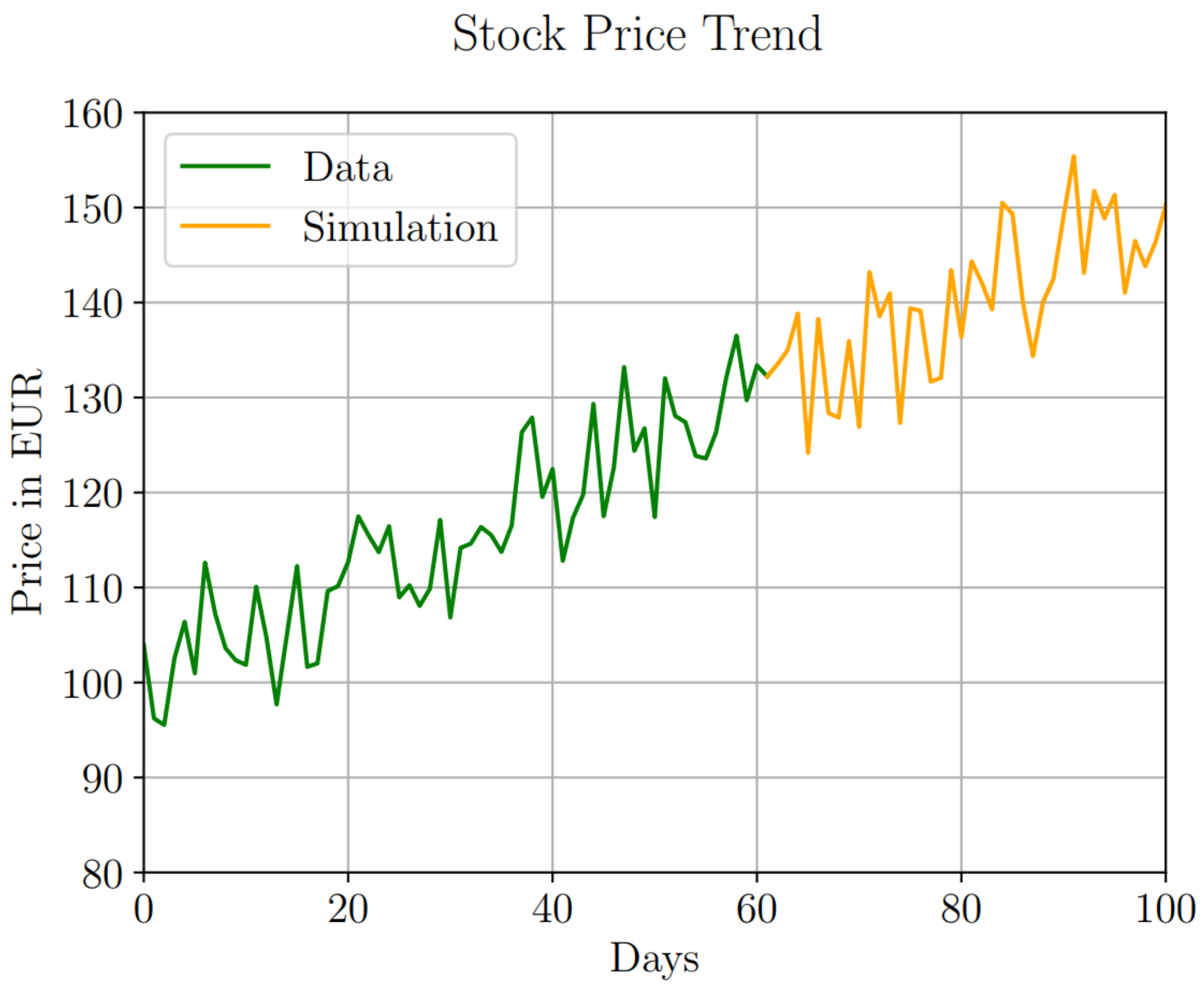
-

Repetition

Stochastic Processes

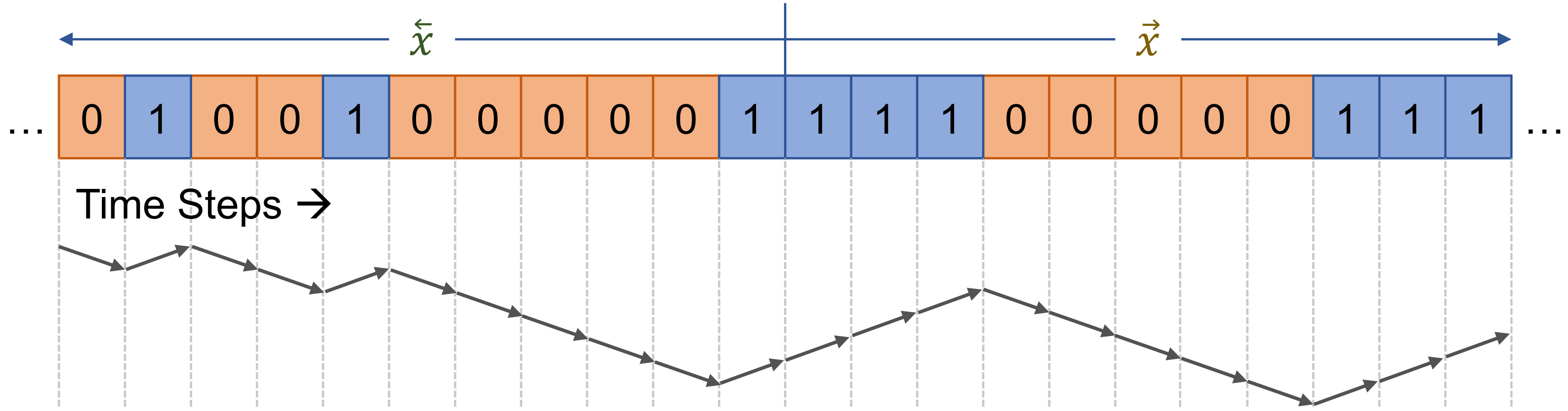
Assume data drawn
by a stochastic process





Stochastic Processes

- Simulating = sampling trajectories
- Trajectory is governed by $P(\vec{X}|\overleftarrow{X})$



Trajectory

Stochastic Processes

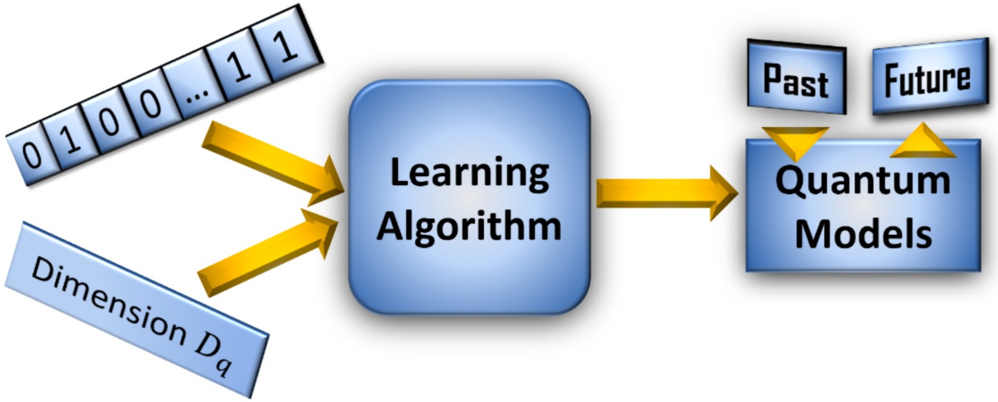
Theoretical statement: Quantum Models are "**better**"

→ Use less memory, can be more accurate, …

Big Picture

Yang et al., arXiv:2105.14434

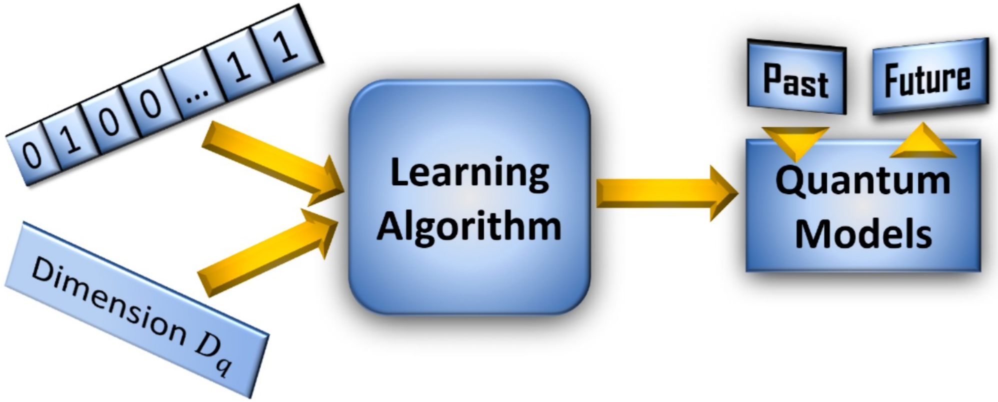
1)



Big Picture

SimTech

Yang et al., arXiv:2105.14434

1) 

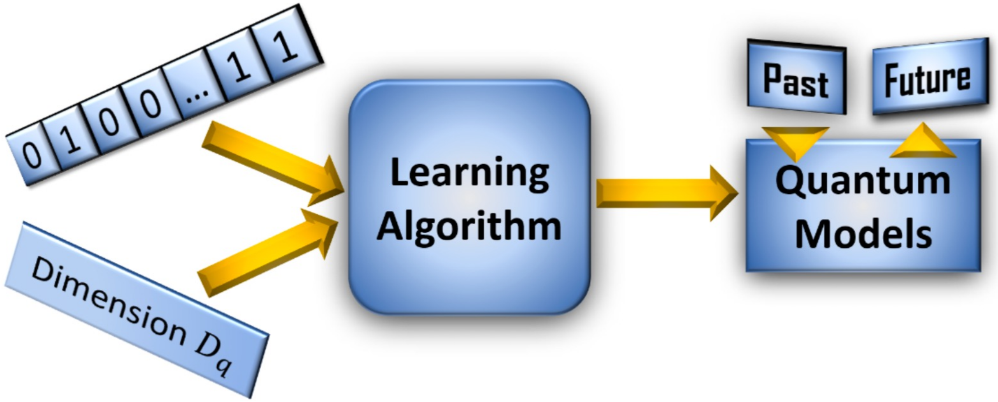
2) Lower bound for the KL divergence

Yang et al., arXiv:2105.14434

Big Picture

Yang et al., arXiv:2105.14434

1)



2) Lower bound for the KL divergence

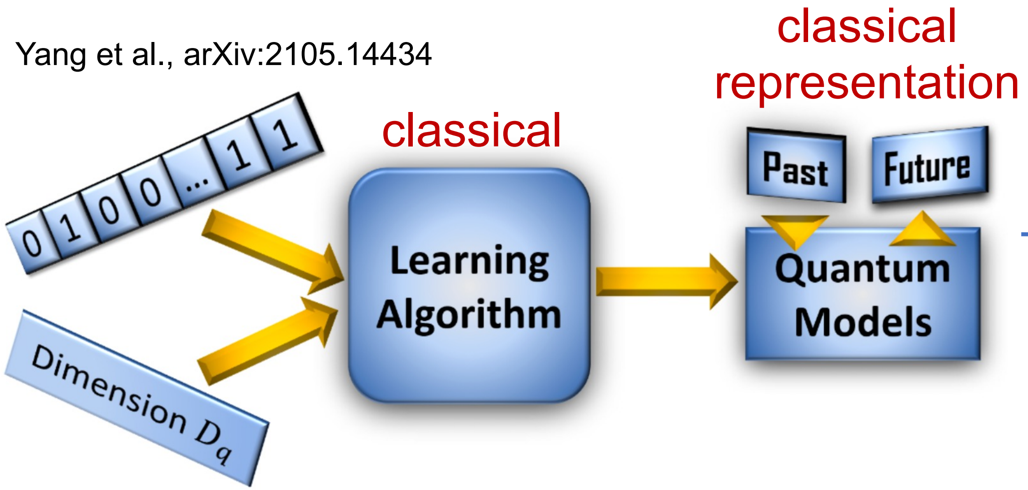
Yang et al., arXiv:2105.14434

Learned Q-Models are <u>provably</u> better

Big Picture

Yang et al., arXiv:2105.14434

1)



memory inefficient

Learned Q-Models are provably better

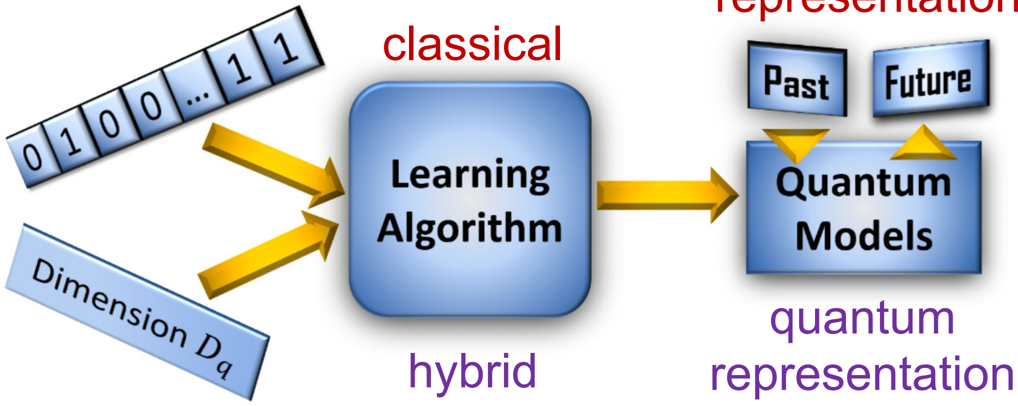
2) Lower bound for the KL divergence

Yang et al., arXiv:2105.14434

Big Picture

Yang et al., arXiv:2105.14434

1)



classical representation

classical

hybrid (VQA)

quantum representation

memory inefficient

Learned Q-Models are <u>provably</u> better

Master Thesis
proof of concept

2) Lower bound for the KL divergence

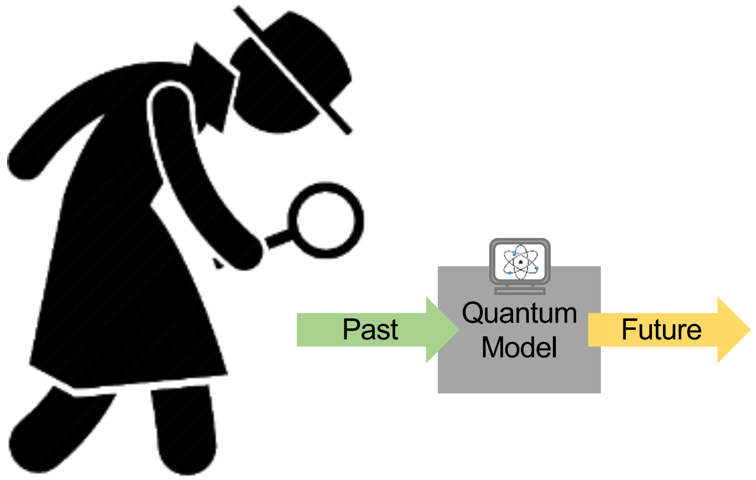
Yang et al., arXiv:2105.14434

Missing:

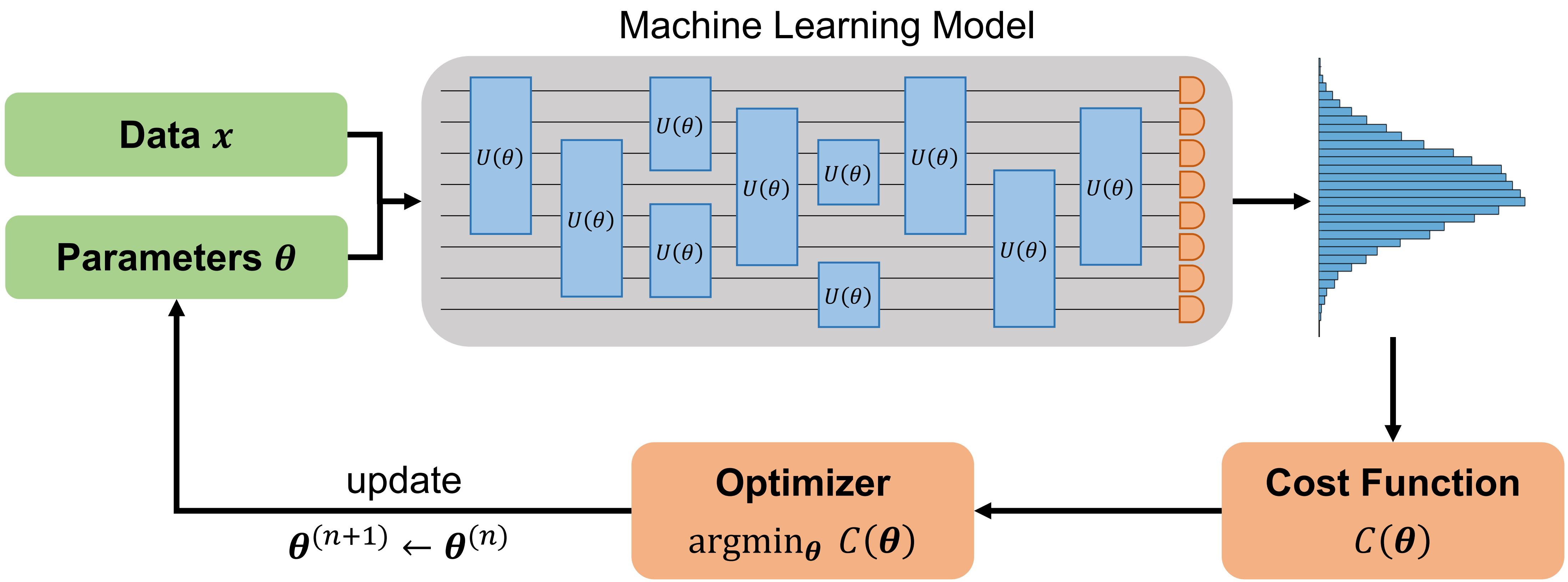
- more complex processes
- experiments

Problem

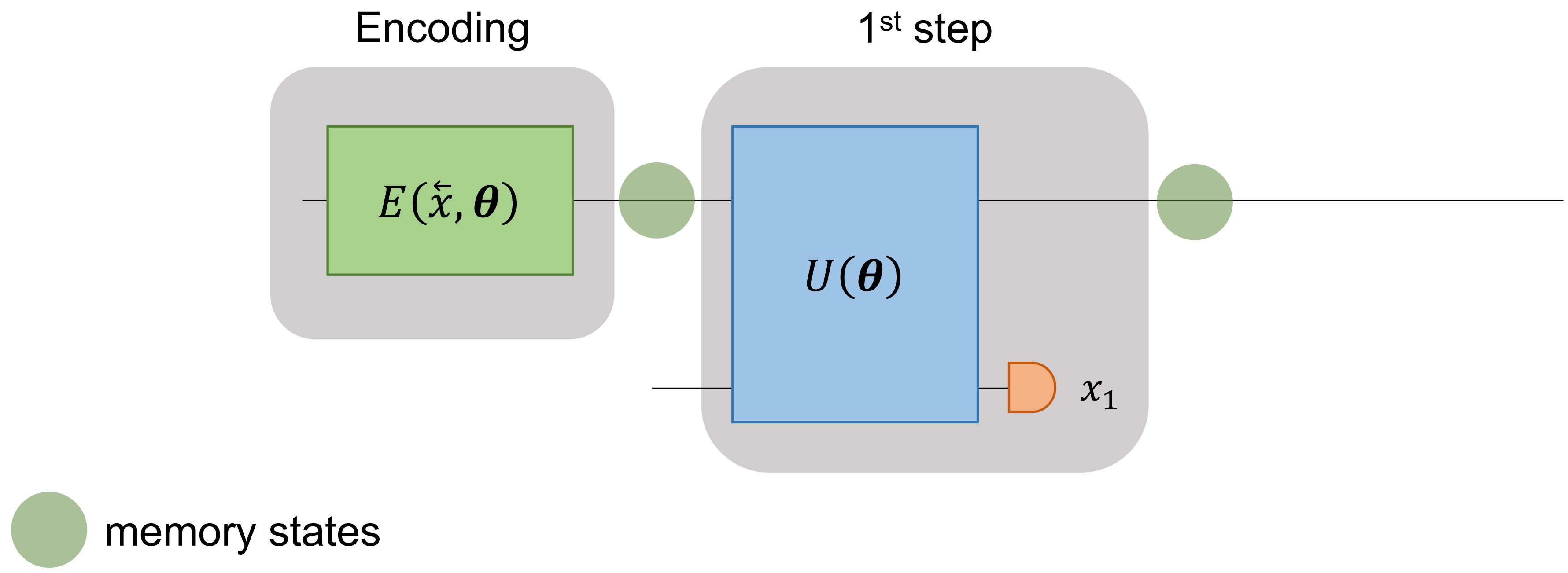
The models are hard to find / learn

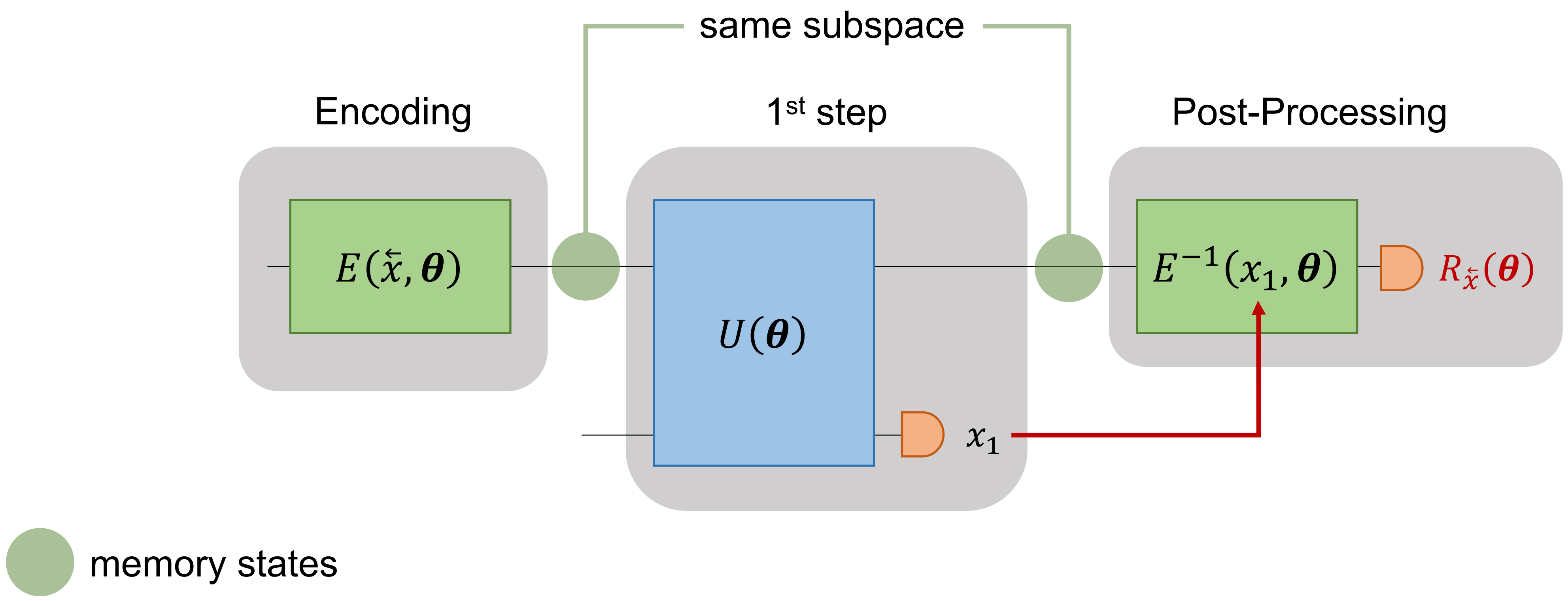


Variational Quantum Algorithm



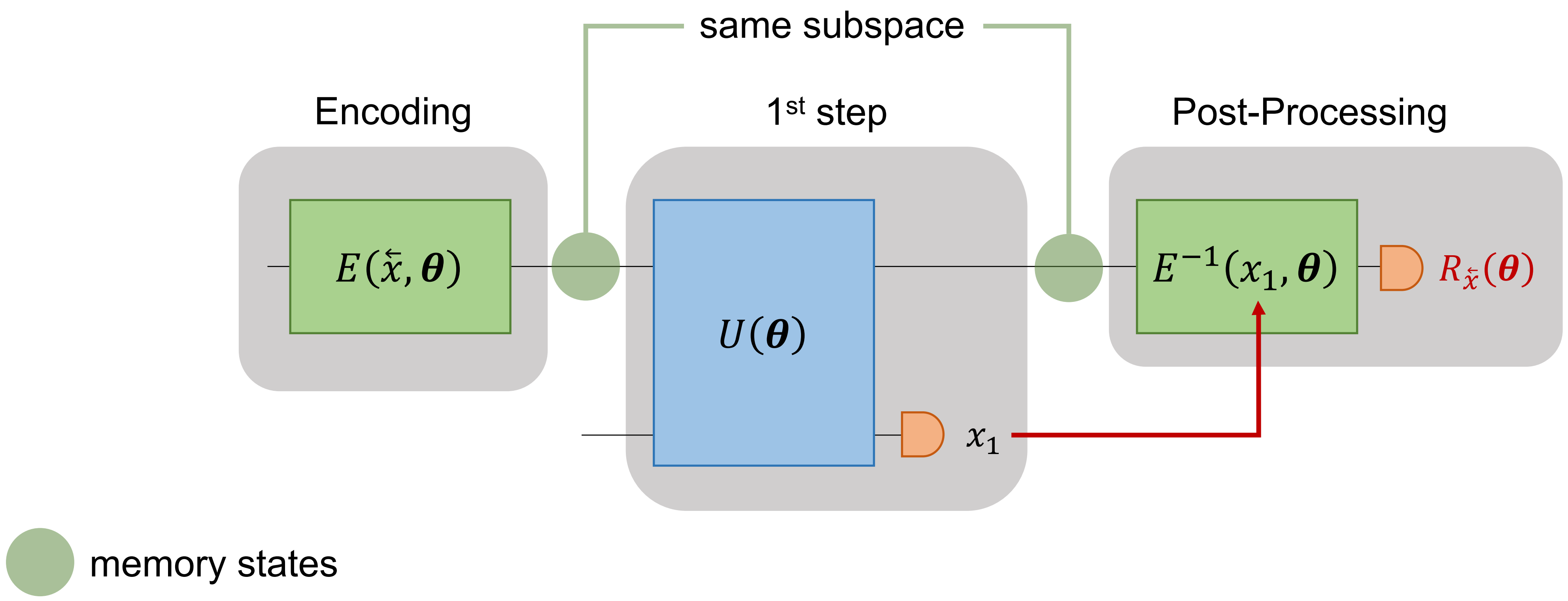
Cost Function





same subspace
Encoding
1st step
Post-Processing
$E(\overleftarrow{x}, \boldsymbol{\theta})$
$U(\boldsymbol{\theta})$
$E^{-1}(x_1, \boldsymbol{\theta})$
$R_{\overleftarrow{x}}(\boldsymbol{\theta})$
x_1
memory states

Cost Function



$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

Simulating Stochastic Processes

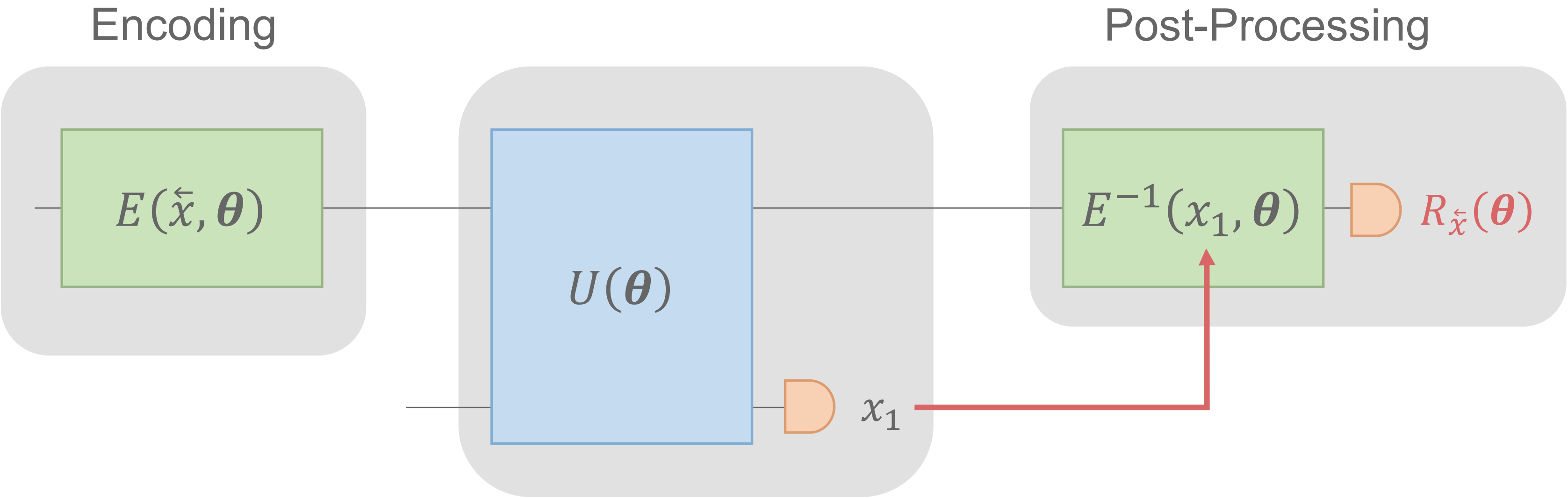
-

Obstacles

Obstacles

only simple processes

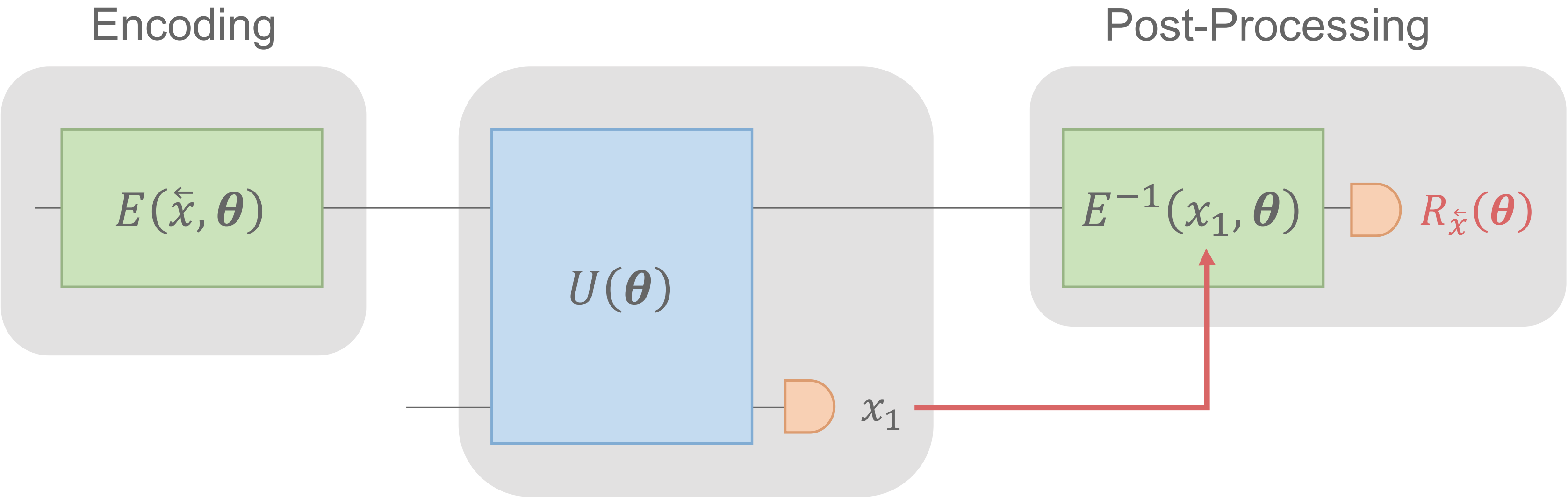
0	1	1	1	1	0	0	0	0	0	0	1



$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

Obstacles

only simple processes



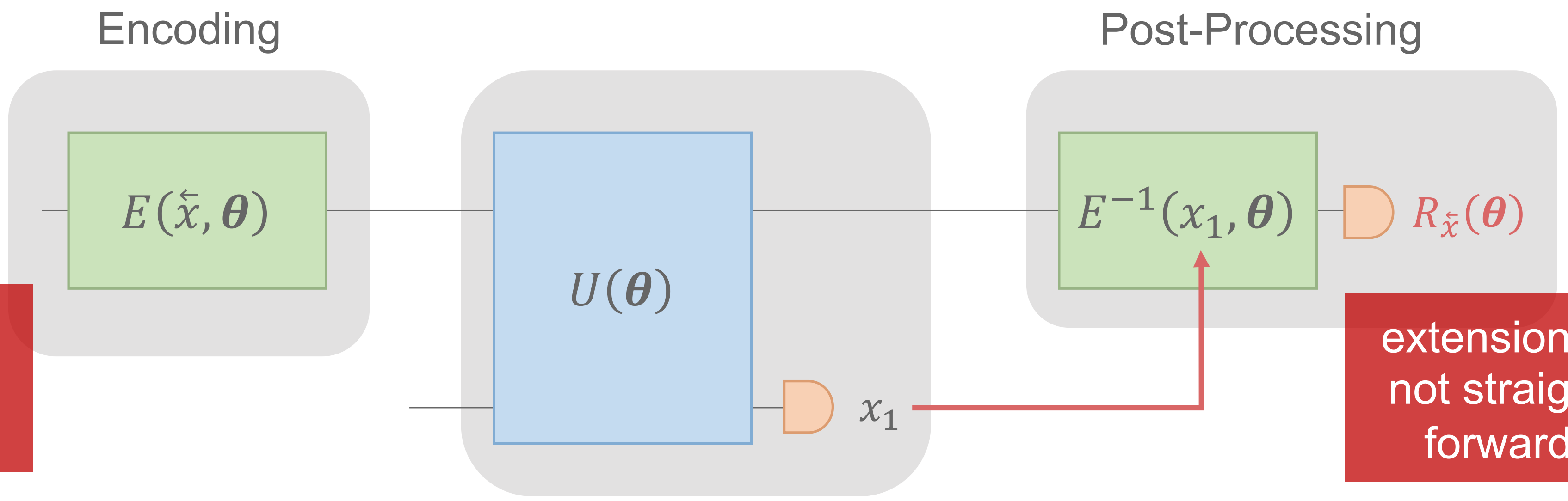
$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

gradient is complicated

Obstacles

only simple processes

0	1	1	1	1	0	0	0	0	0	1



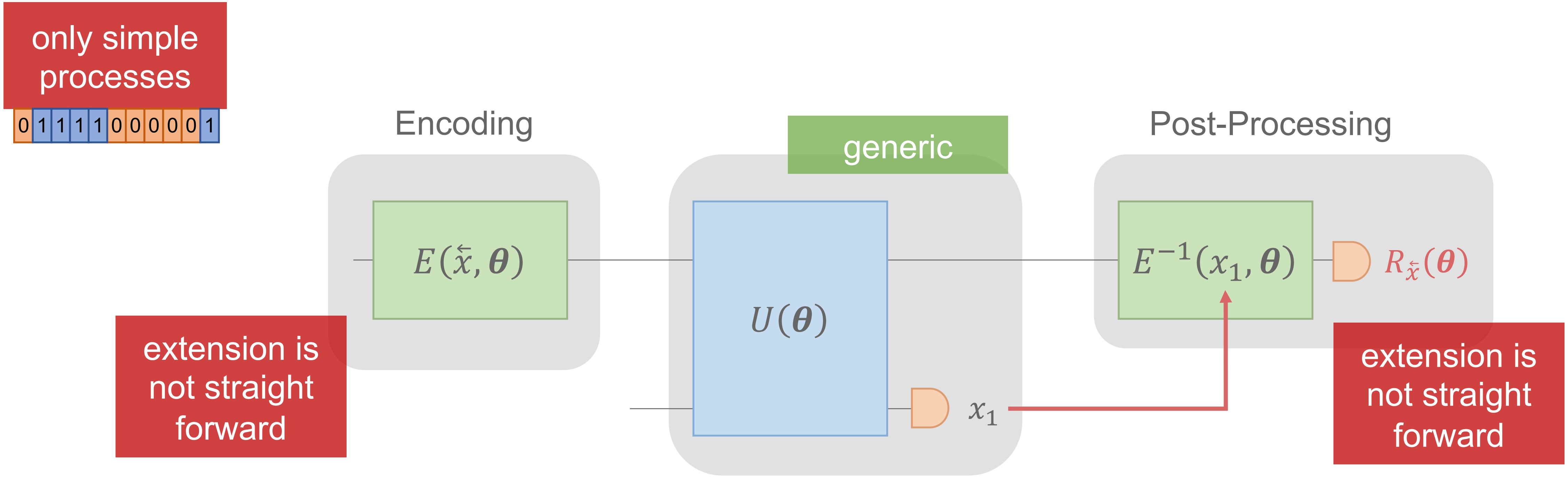
extension is not straight forward

extension is not straight forward

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

gradient is complicated

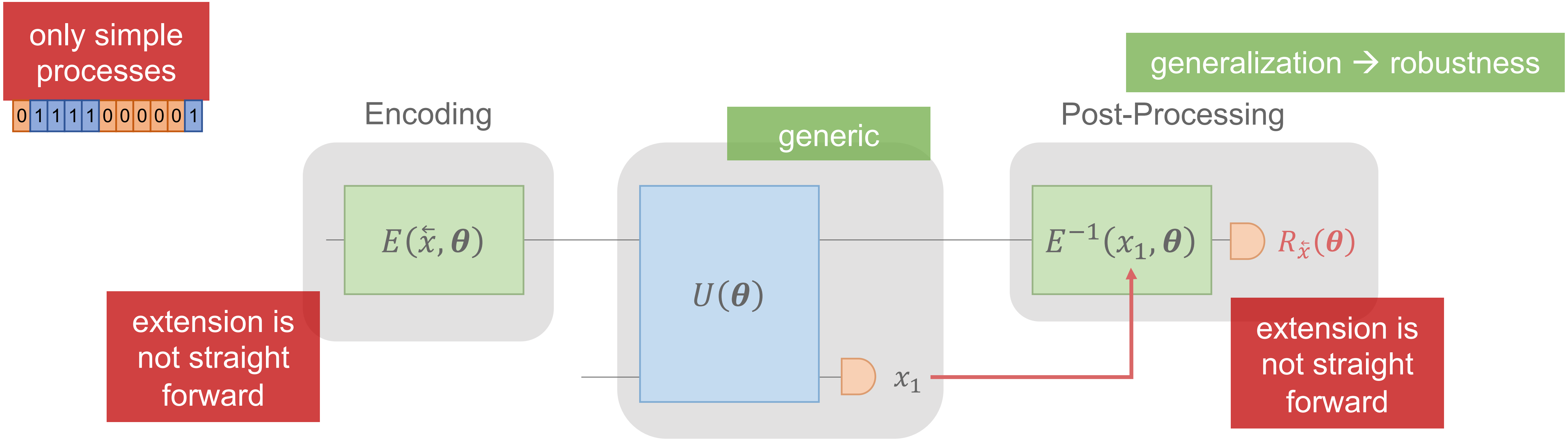
Obstacles



$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

gradient is complicated

Obstacles



$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

gradient is complicated

Simulating Stochastic Processes

-

Solutions

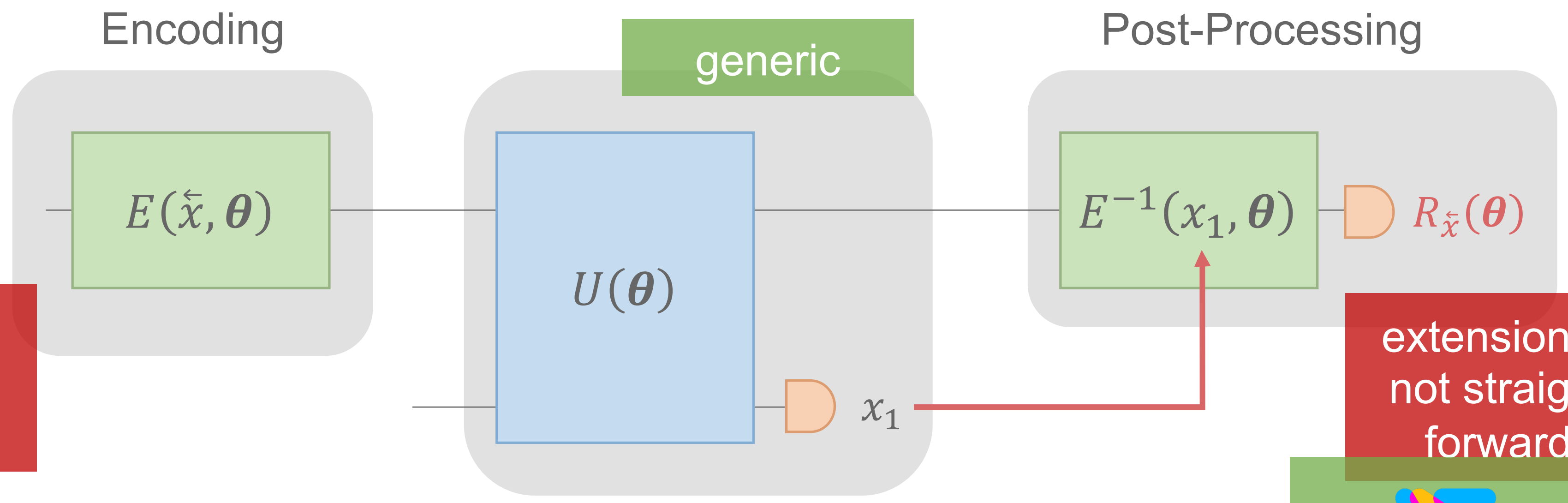
Obstacles

only simple processes

0 1 1 1 1 0 0 0 0 0 1

generalization → robustness

generic



extension is not straight forward

extension is not straight forward

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

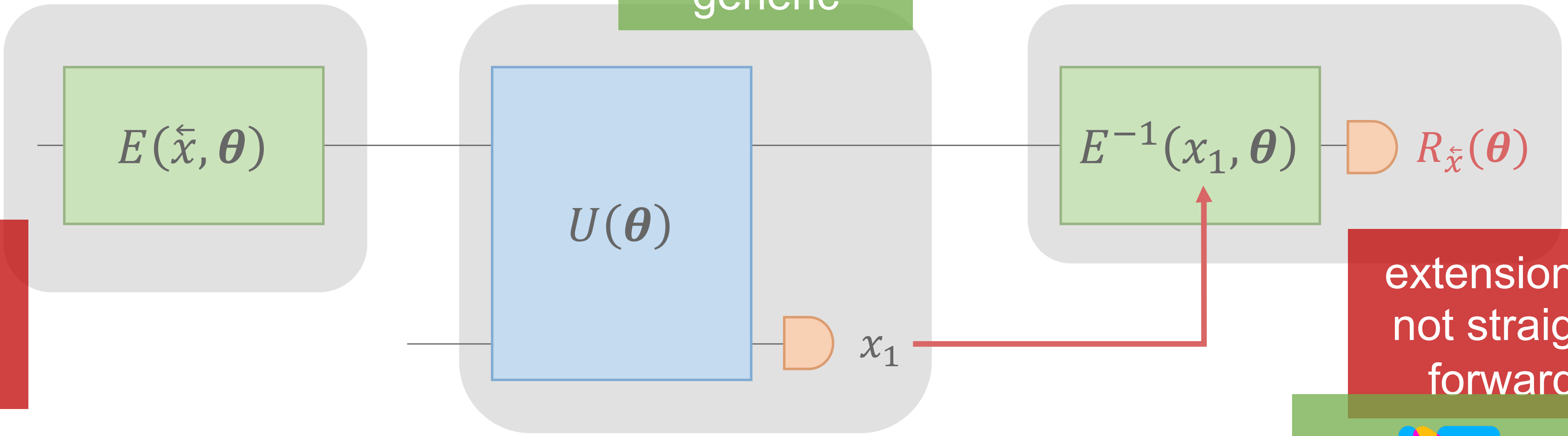
gradient is complicated

Obstacles

only simple processes

0 1 1 1 1 0 0 0 0 0 1

generalization → robustness



extension is not straight forward

extension is not straight forward

PENNYLANE

PyTorch

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}} | \overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

gradient is complicated

Obstacles

SimTech

only simple processes

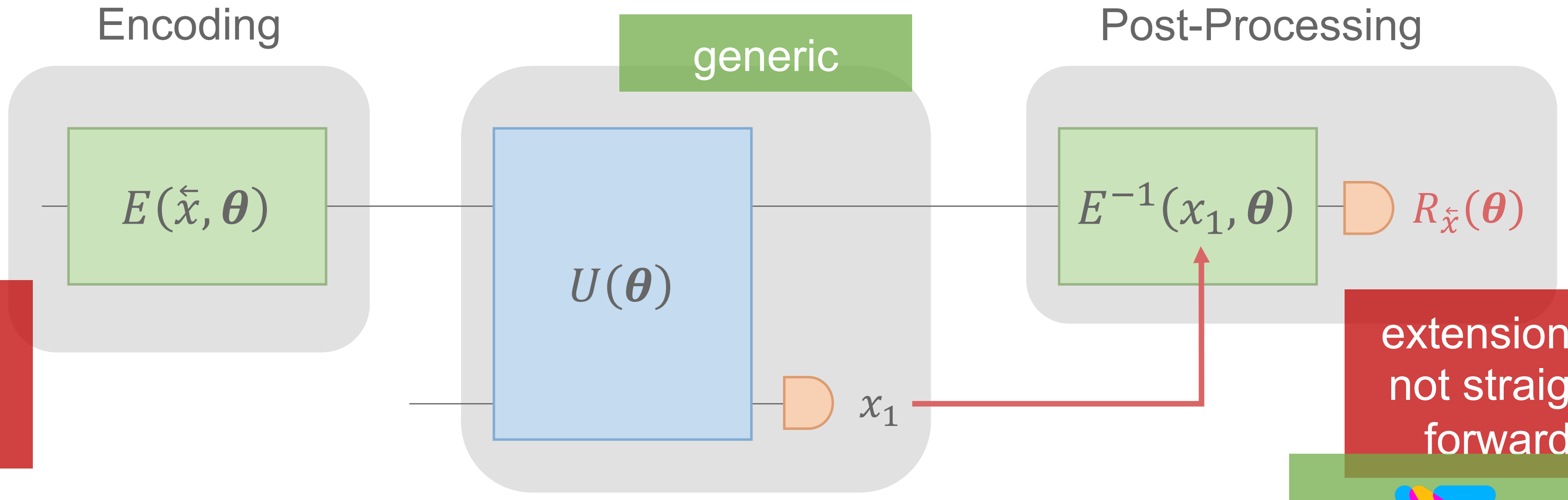
0 1 1 1 1 0 0 0 0 0 1

try and error

extension is not straight forward

generalization → robustness

generic



extension is not straight forward

PENNYLANE

gradient is complicated

PyTorch

$$C(\boldsymbol{\theta}) = \sum_{\overleftarrow{x}} w_{\overleftarrow{x}} \cdot \mathrm{MMD}^2[\mathrm{P}, \hat{P}_{\boldsymbol{\theta}}|\overleftarrow{x}] + R_{\overleftarrow{x}}(\boldsymbol{\theta})$$

Simulating Stochastic Processes

-

Results of the Refactoring

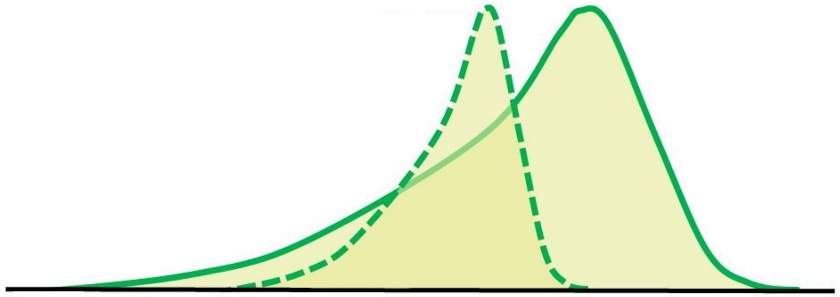
Metrics

Maximum Mean Discrepancy: (MMD)

$$MMD(P,\hat{P}) = \sup_{f \in F}\left[\mathbb{E}_{x\sim P}\, f(x) - \mathbb{E}_{y\sim \hat{P}} f(y)\right]$$

Kullback-Leibler divergence: (KL)

$$D_{KL}(P,\hat{P}) = \sum_x P(x)\log_2 \frac{P(x)}{\hat{P}(x)}$$



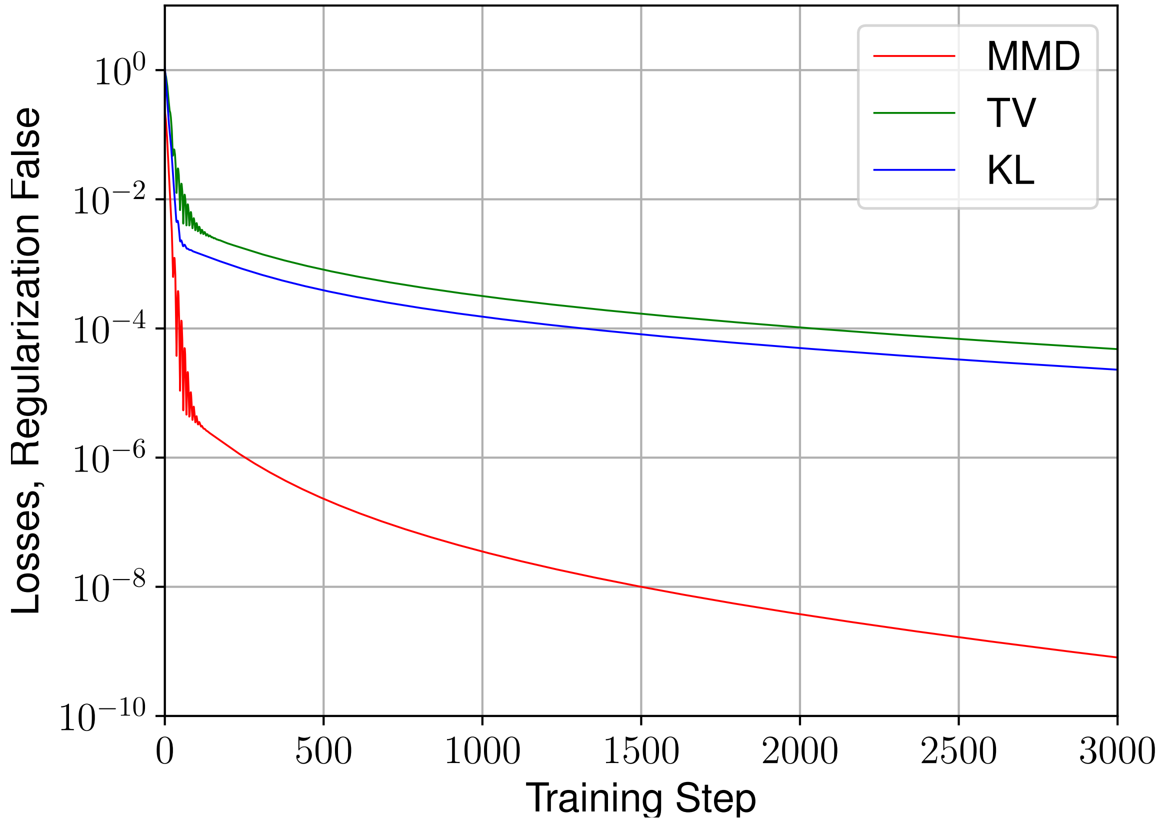
Total Variation Distance: (TV)

$$D_{TV}(P,\hat{P}) = \frac{1}{2}\sum_x \left|P(x) - \hat{P}(x)\right|$$

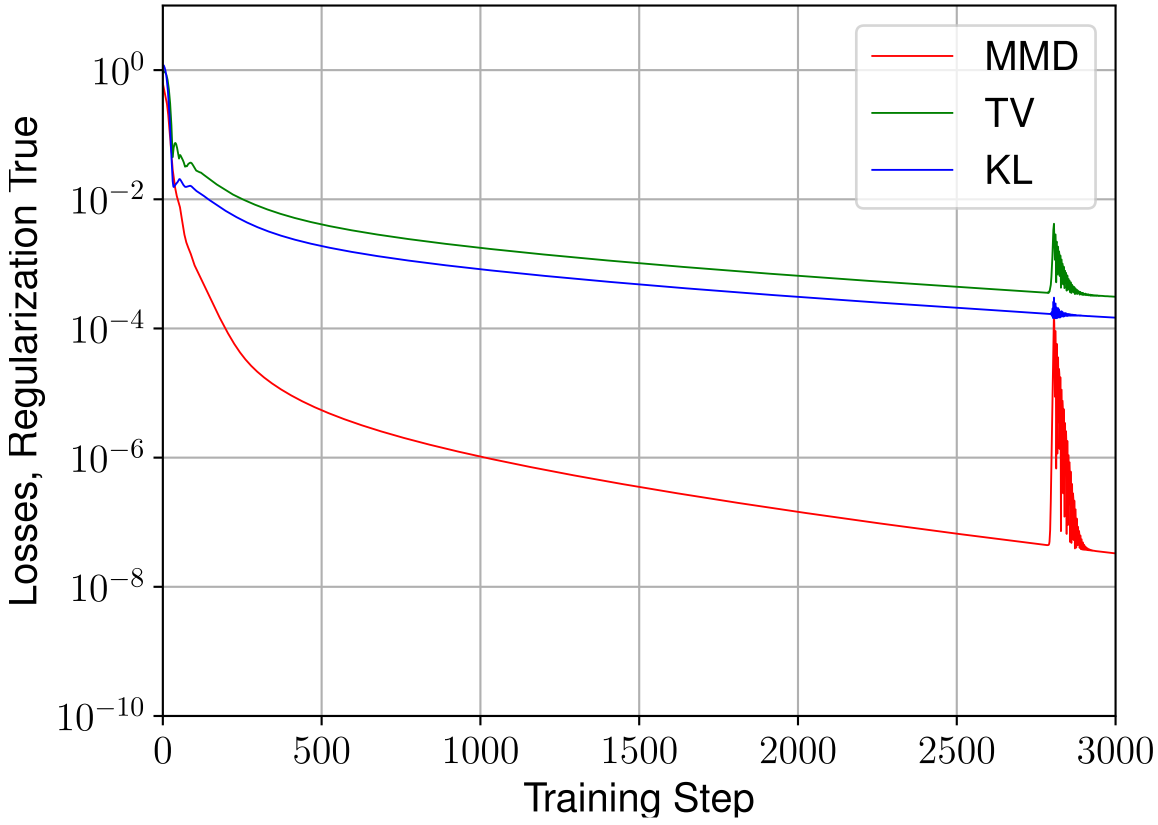
Results – 1 Validation Step

SimTech

Without Regularization

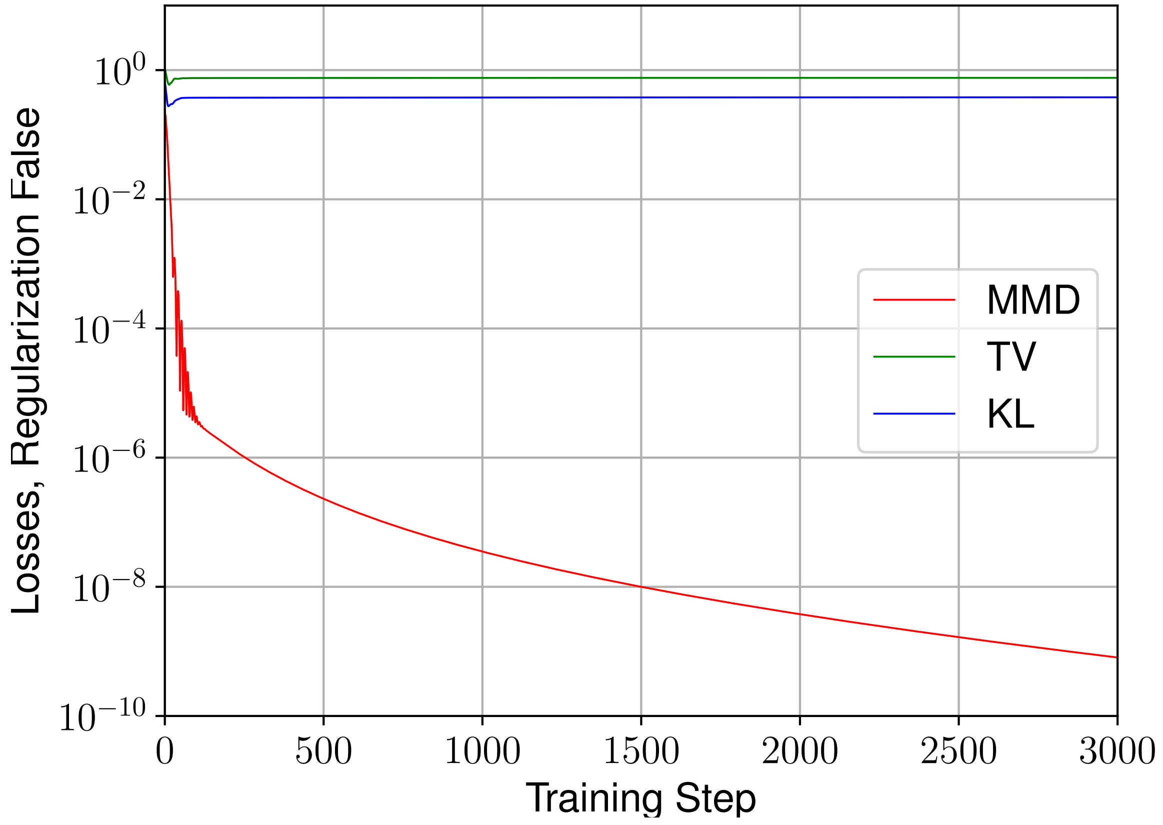


With Regularization

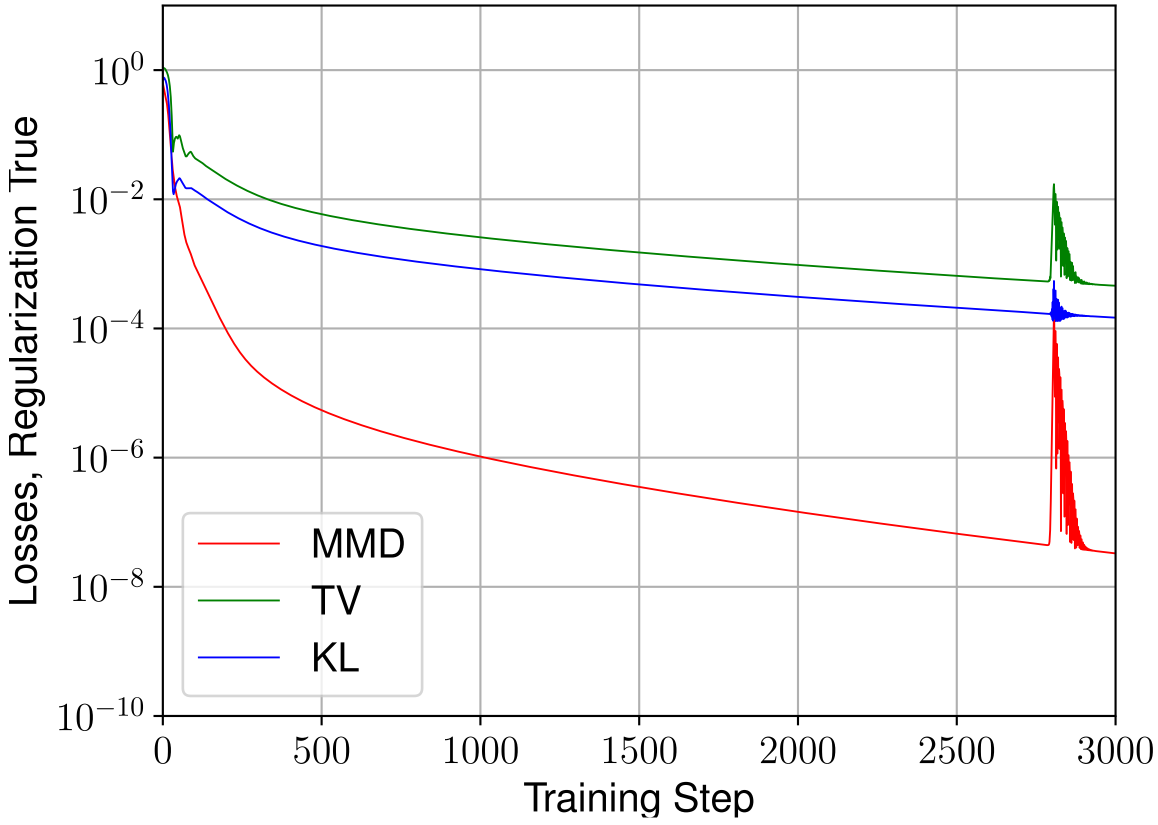


Results – 2 Validation Steps

Without Regularization

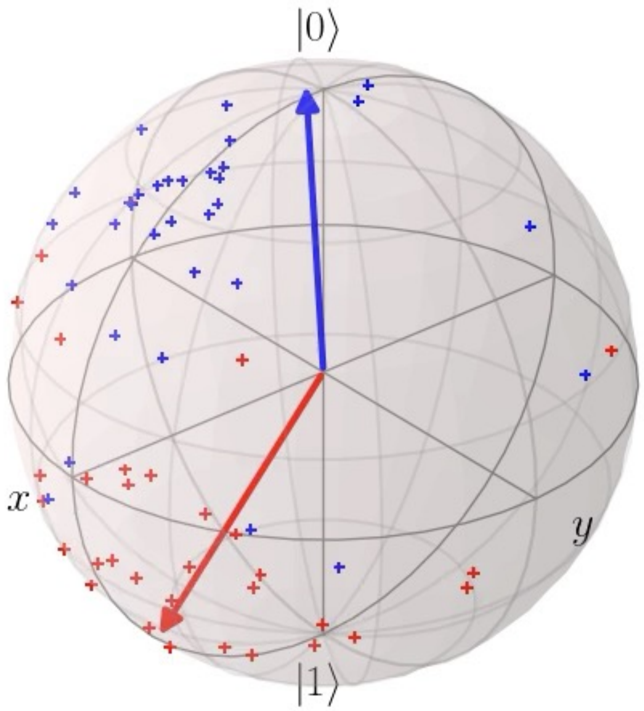


With Regularization

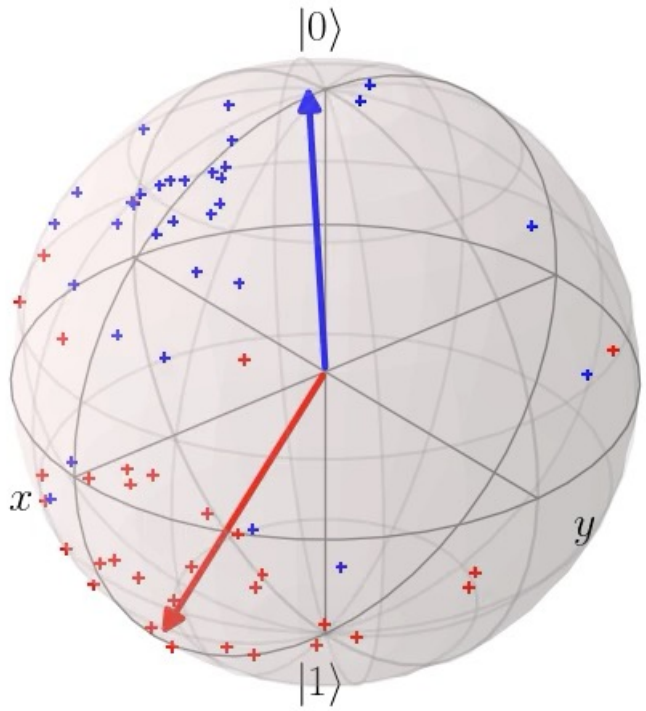


Results – 5 Validation Steps

Without Regularization



With Regularization



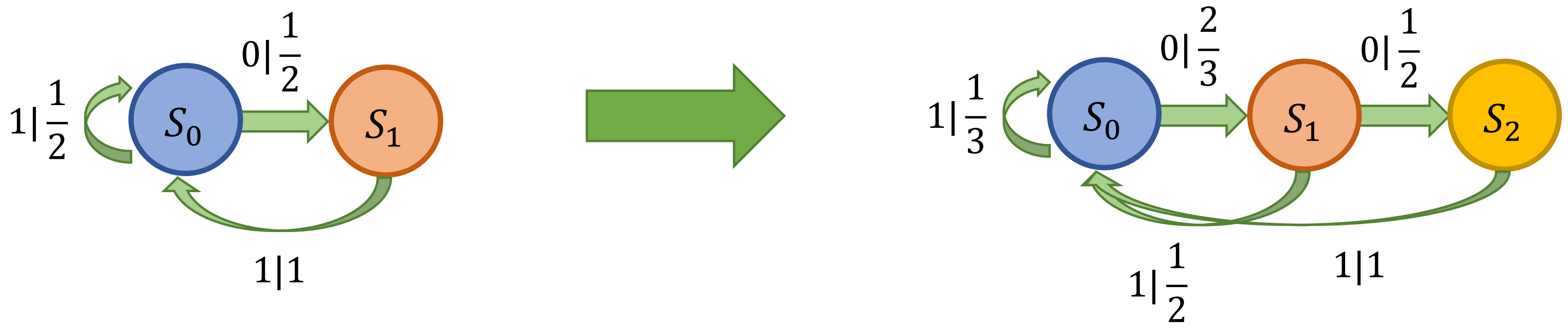
Simulating Stochastic Processes

-

A more complicated process

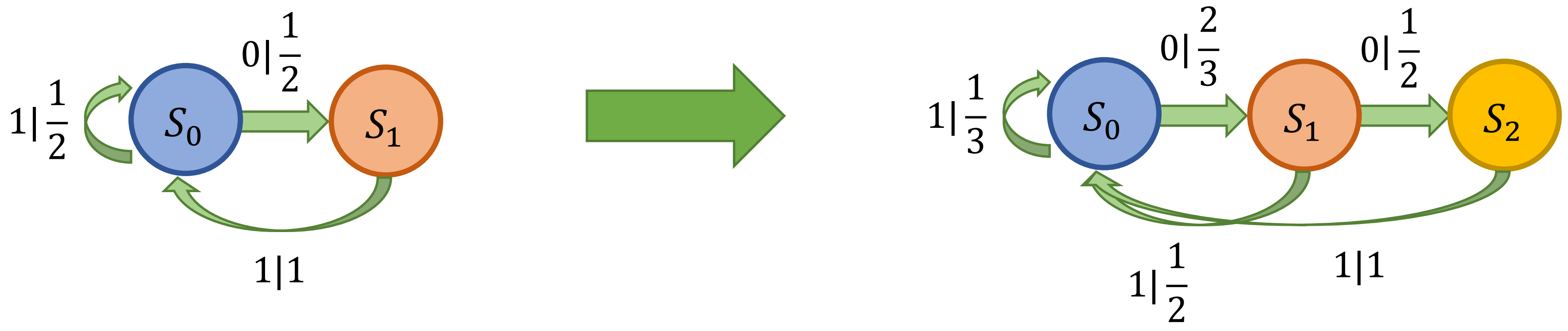
Another Process

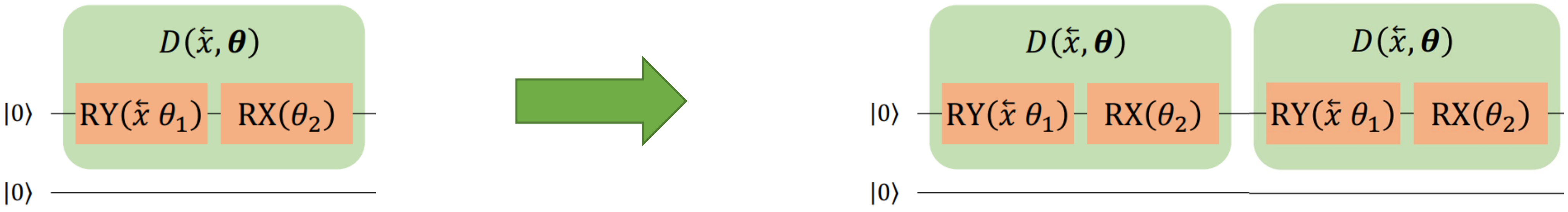
- Use a slightly more complicated stochastic process



Another Process

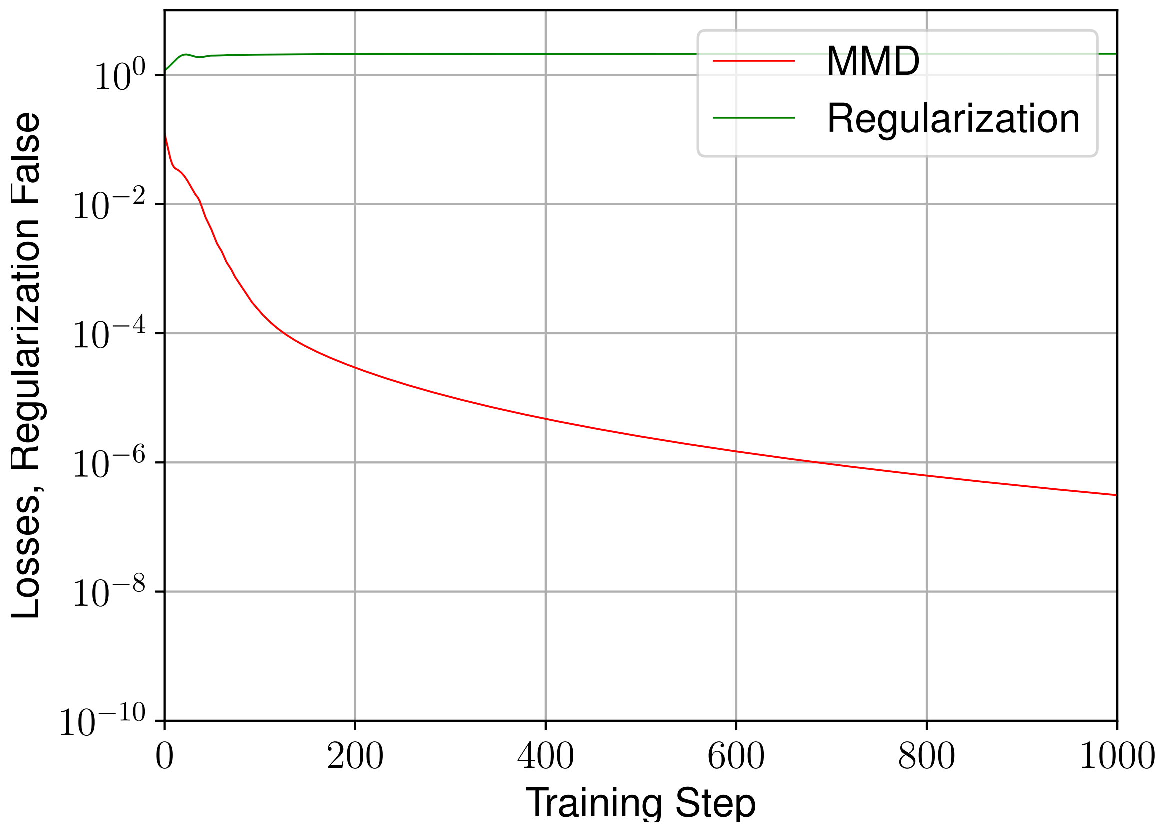
- Use a slightly more complicated stochastic process



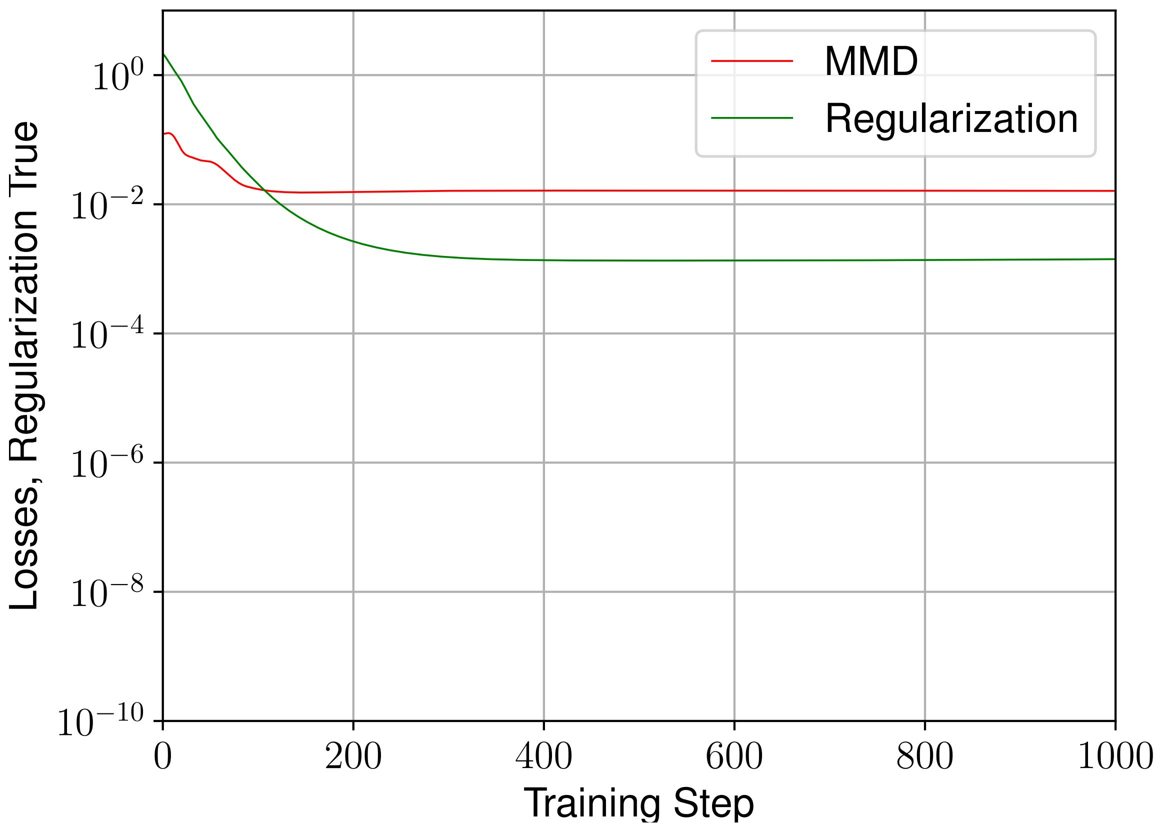


Results – Training

Without Regularization

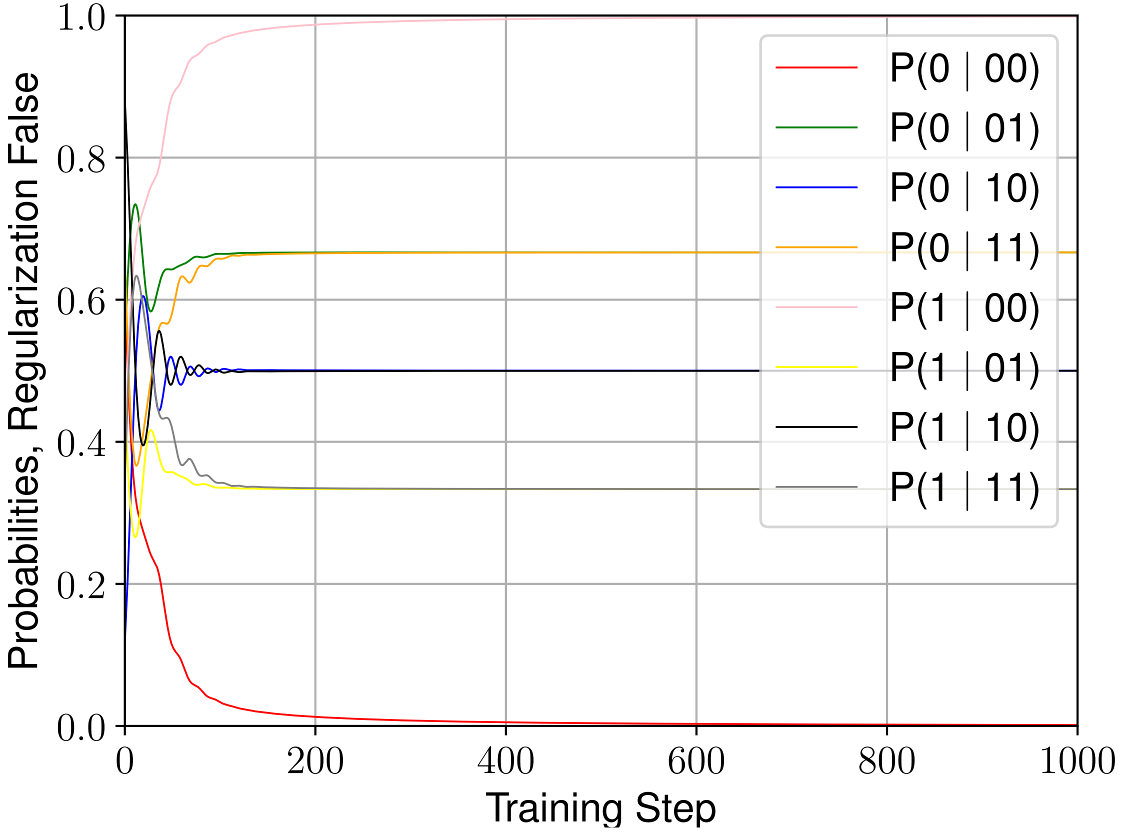


With Regularization

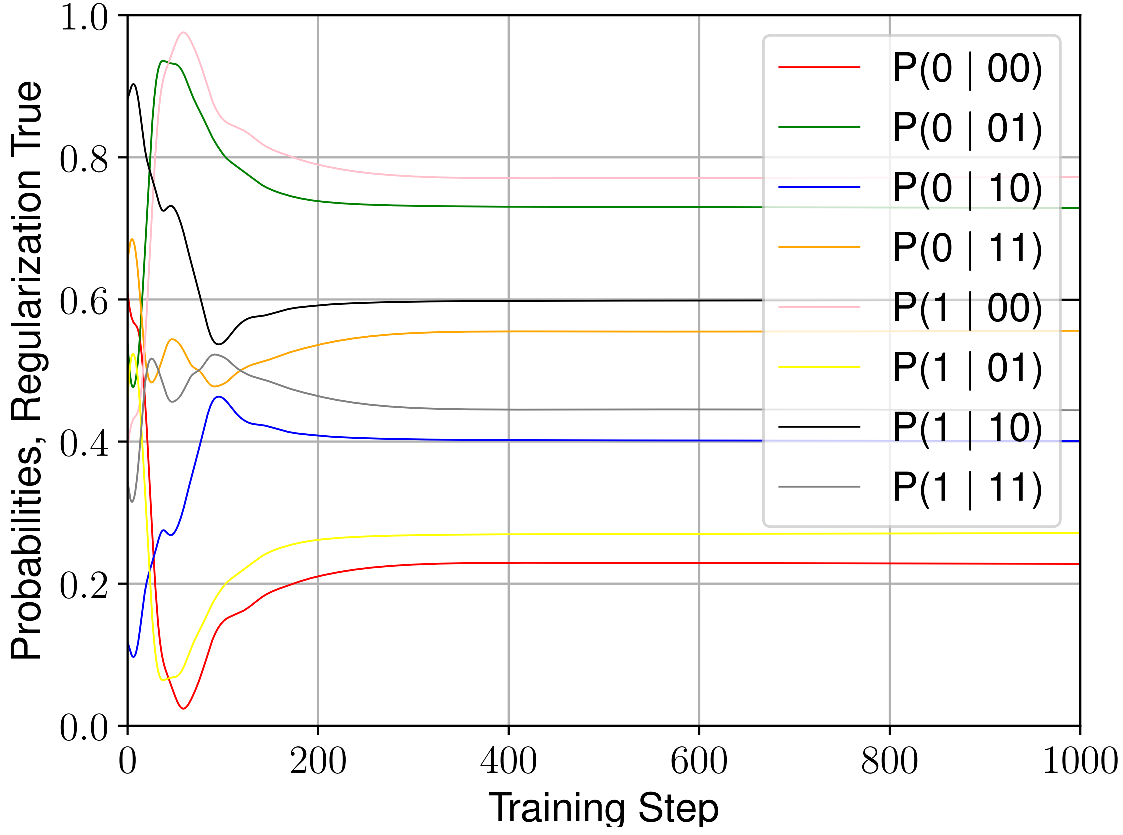


Results – Training

Without Regularization

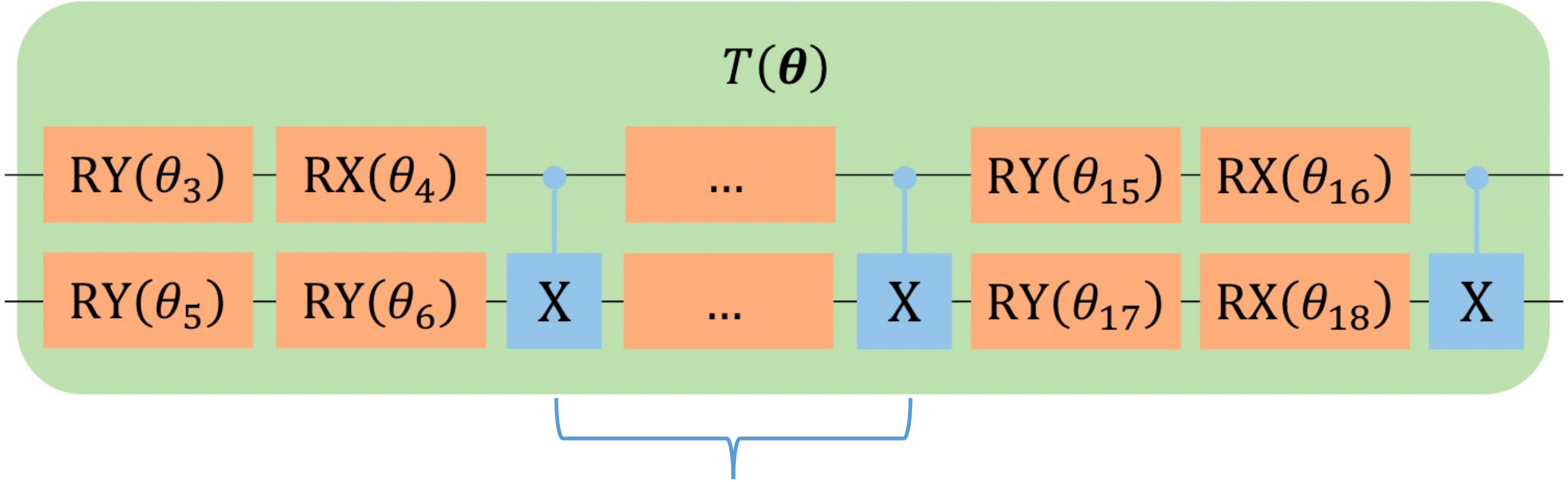


With Regularization



Next Steps

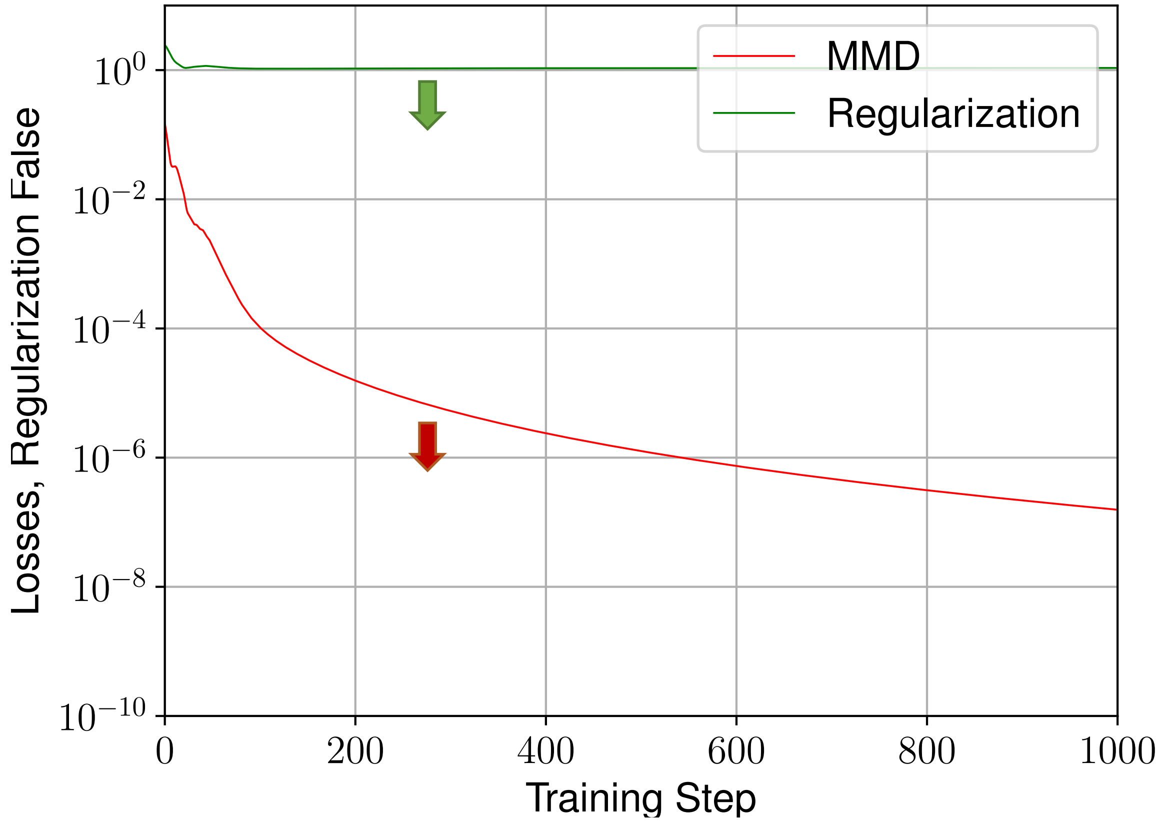
- Make the unitary U more expressive



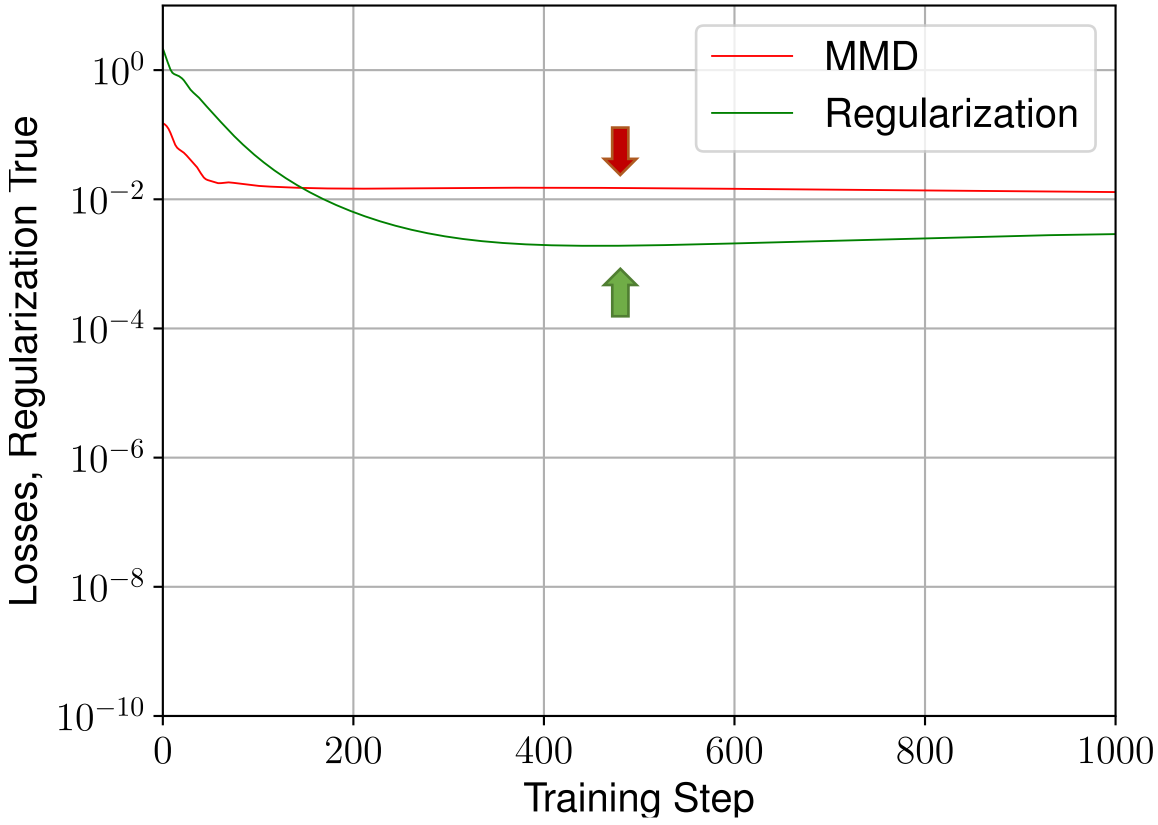
4 layers → 8 layers

Results – Training

Without Regularization

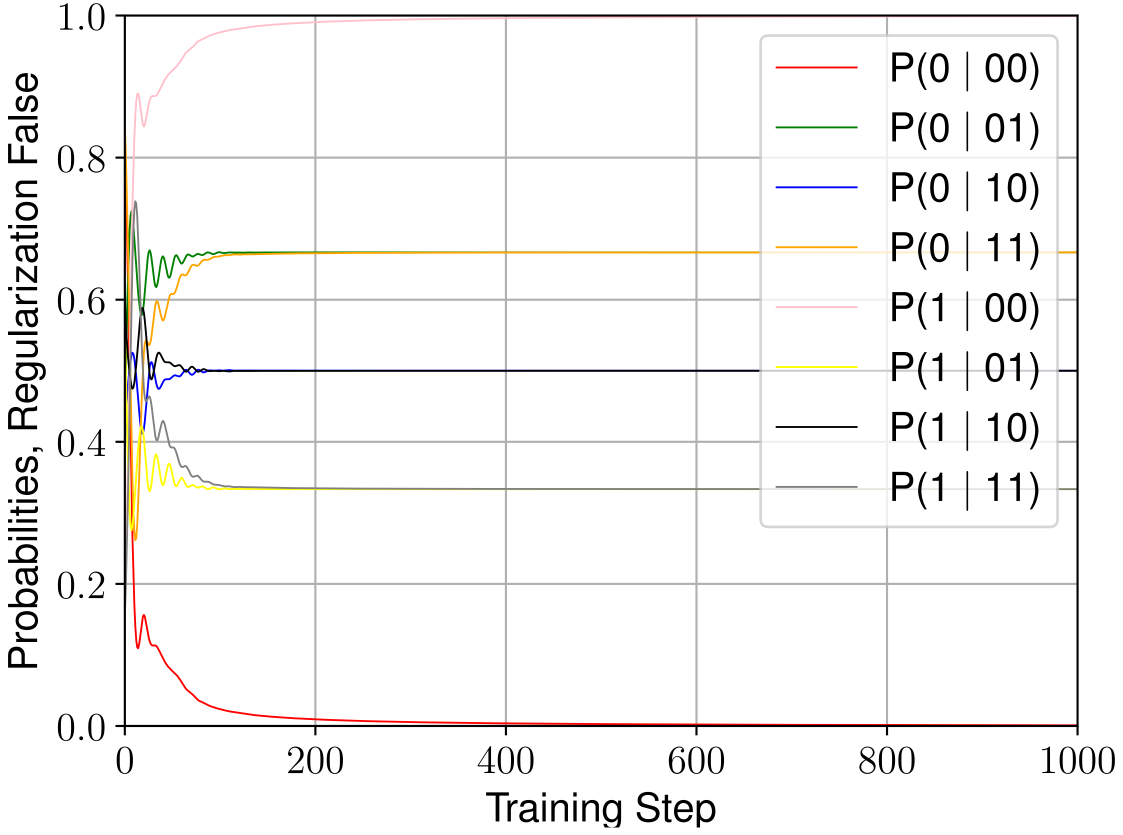


With Regularization

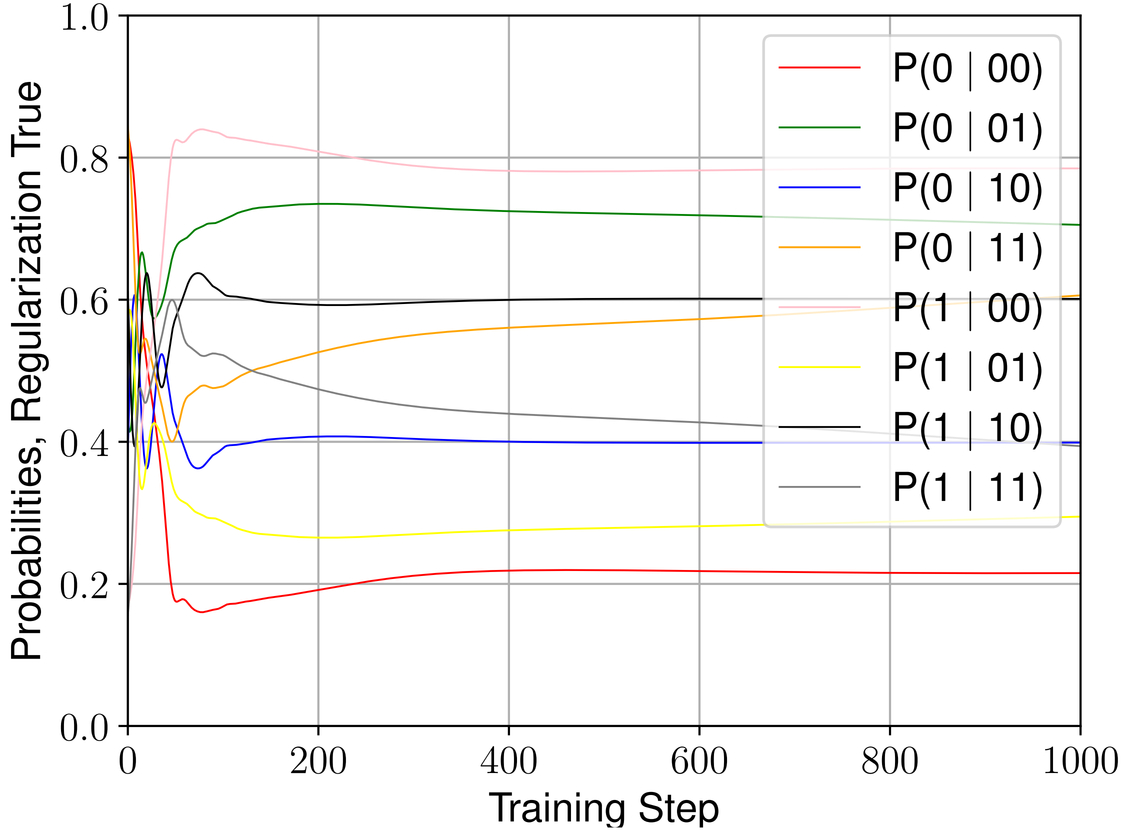


Results – Training

Without Regularization

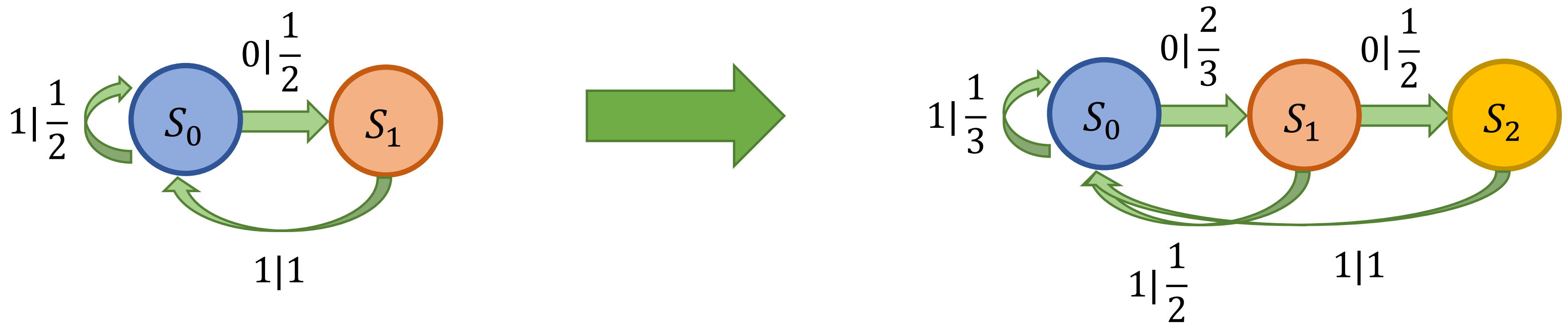


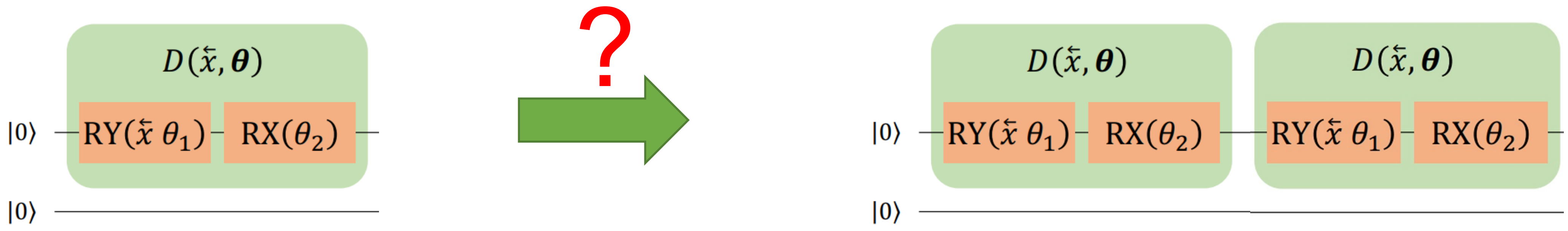
With Regularization



Next Steps

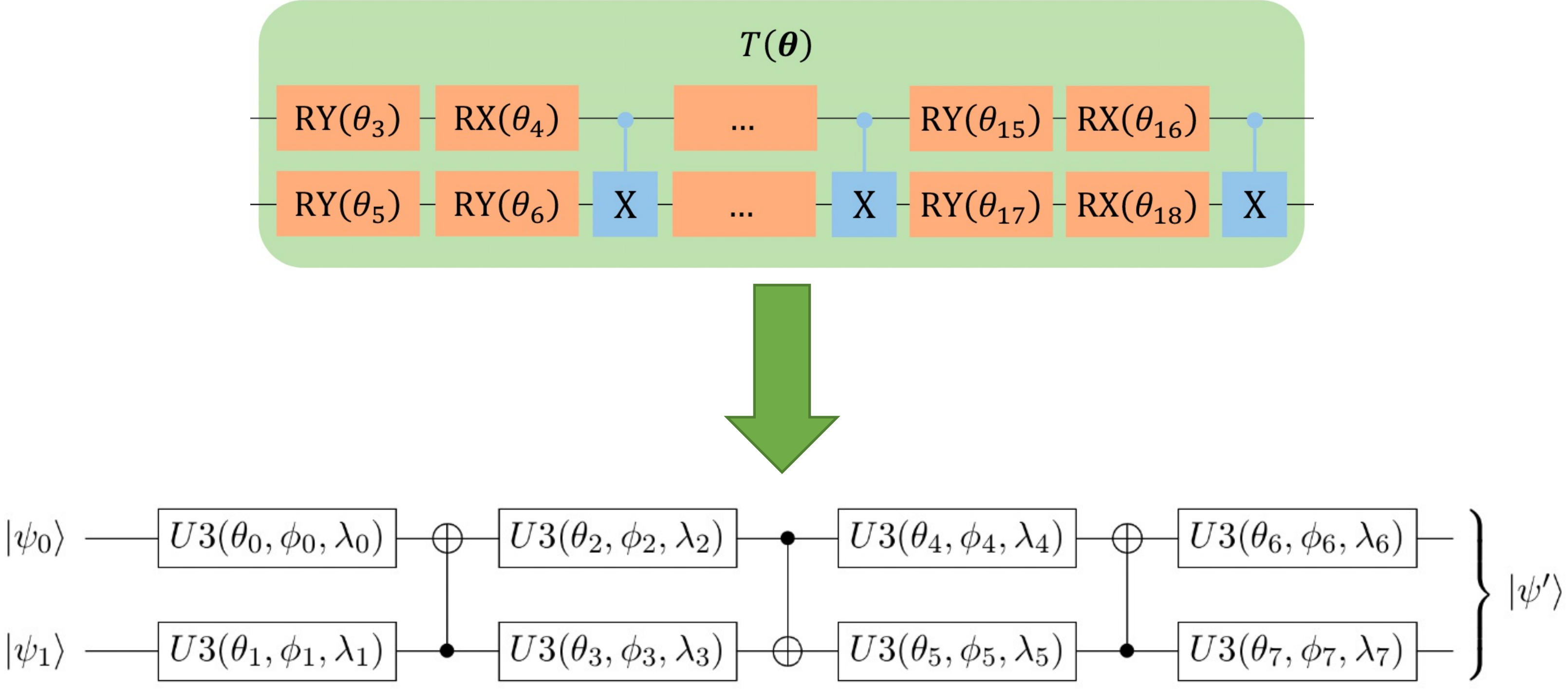
- Create a more sophisticated encoding technique





Next Steps

- Use a universal 2 qubit operator



Shende et al. arXiv:quant-ph/0308033

Robust Quantum Algorithms

Designing noise-resilient Q-Algos

A function $f: X \to Y$ is Lipschitz iff

$$\forall x, x': \ \|f(x) - f(x')\| \leq C\|x - x'\|$$

Designing noise-resilient Q-Algos

A function $f: X \rightarrow Y$ is Lipschitz iff

$$\forall x, x': \ \|\underbrace{f(x) - f(x')}_{\text{max output disturbance}}\| \leq C\|\underbrace{x - x'}_{\text{max input disturbance}}\|$$

Designing noise-resilient Q-Algos

A function $f: X \to Y$ is Lipschitz iff

$$\forall x, x': \ \|f(x) - f(x')\| \leq C\|x - x'\|$$

max output disturbance

max input disturbance

A quantum algo. can be written as

$$f_\theta(x) = \langle \psi(\theta, x)| M_\theta |\psi(\theta, x)\rangle$$

Designing noise-resilient Q-Algos

SimTech

A function $f: X \to Y$ is Lipschitz iff

$$\forall x, x': \ \|f(x) - f(x')\| \leq C\|x - x'\|$$

max output disturbance

max input disturbance

A quantum algo. can be written as

$$f_\theta(x) = \langle \psi(\theta, x) | M_\theta | \psi(\theta, x) \rangle$$

The Lipschitz constant of f_θ defines a robustness measure for the quantum algorithm.

Designing noise-resilient Q-Algos

A function $f: X \to Y$ is Lipschitz iff

$$\forall x, x': \; \|f(x) - f(x')\| \leq C\|x - x'\|$$

max output disturbance

max input disturbance

A quantum algo. can be written as

$$f_\theta(x) = \langle \psi(\theta, x) | M_\theta | \psi(\theta, x) \rangle$$

The Lipschitz constant of f_θ defines a robustness measure for the quantum algorithm.

The constant can be adjusted via suitable regularizations.

Next Steps

- Create a simple QNN
- Train on noise-free circuits vs. with noise (with/without regularization)
- We expect a better accuracy for noisy models with regularization

Thanks!

Let’s
discuss!