Graph Neural Networks for Power Systems Operation

Balthazar DONON - 06/09/2023

Balthazar DONON

Deep Statistical Solvers & Power Systems Applications

- 2018 - 2022 : PhD student @ RTE R&D and Université Paris-Saclay

Graph Neural Networks for Tertiary Voltage Control

- 2022-2024 : Postdoc at Institut Montefiore in a project for RTE R&D

- Talk is about our latest work !
- But it does not include experimental results for now…
- *Topology-Aware Reinforcement Learning for Tertiary Voltage Control,* submitted to PSCC 2024 (power systems conference).
- Gives an idea of how GNNs can be applied for a real-life power systems problem.

- Collaboration between **RTE R&D** and **Institut Montefiore** (Liège Université).

Louis Wehenkel
Professor - ULiège

François Cubelier
PhD student - ULiège

Efthymios Karangelos
Postdoc - ULiège

Laure Crochepierre
Researcher - RTE

Balthazar Donon
Postdoc - ULiège

Background & Motivations

Background & Motivations

High Voltage Issues

- Increase in the frequency and intensity of high voltage events:
  - Can damage assets,
  - Caused by increase of renewables, new electricity uses, etc.
- Operators do not have time for that.

[1] P.Mandoulidis, V.Lampropoulos, C.Vournas, M.Karystianos, G.Christoforidis, A. Neris, and Y. Kabouris, “Controlling long-term overvoltages in lightly loaded transmission systems,” 2022.

Background & Motivations

Long Term Goal

- We want a decision support tool to assist operators.
  - Input : Operating condition x,
  - Output : Voltage setpoints y.
- The tool will not take control of the grid : it can only suggest an action (open-loop control).
- The tool cannot rely on expensive intermediate simulations.

Background & Motivations

Traditional Optimization

- Tertiary Voltage control can be cast as an Optimal Power Flow [2].
  - Mixed Integer Non Linear Problem.
  - Current resolution methods do not scale to real-life power systems.

[2] A. Castillo, “Essays on the ACOPF Problem: Formulations, Approximations, and Applications in the Electricity Markets »

Background & Motivations

Deep Learning

- Why not Deep Learning ?
  - Tremendous successes in various domains.
  - Solves complex problems that require a very high level of abstraction [3] [4] [5].

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,”
[4] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,”
[5] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Diele- man, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,”

Initial Optimization problem

Initial Optimization Problem

Variables

- Let $x \in \mathscr{X}$ be an operating condition (i.e. a snapshot at certain instant).
  - Encompasses both its structure and its numerical features.

- Let $y \in \mathscr{Y}(x)$ be a tertiary voltage control decision.
  - Size depends on the number and nature of assets in x.

Initial Optimization Problem

Problem

- We look for a tertiary voltage control decision that minimizes a real-valued cost function c,

$$y^*(x) = \underset{y \in \mathcal{Y}(x)}{\arg\min}\ c(x, y).$$

- Tertiary voltage control includes both continuous and discrete variables.
- As a first step, we only consider generators voltage setpoints (continuous).

Initial Optimization Problem

Cost function

- As a proxy of what operators do, we consider

$$c(x, y) = c_J(x, y) + c_V(x, y) + c_I(x, y).$$

  - c_J penalizes power losses caused by Joule's effect.
  - c_v penalizes voltage violations.
  - c_I penalizes overflows.
- Computing c requires to run a black-box simulation.

Reinforcement Learning Problem

RL Problem

Distribution of Problem Instances

- We wish to solve the problem for a whole distribution p of operating conditions x.
- Given a certain x, we have to return a y in one step.
  - *Contextual Bandit*, typically addressed in Reinforcement Learning [6].

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*.

RL Problem

Conversion to an RL Problem

We consider a probabilistic control policy Π_θ , such that

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{\substack{x \sim p(\,\cdot\,) \\ y \sim \Pi_\theta(\,\cdot\,|x)}} \Big[c(x, y) \Big].$$

x can be viewed as a state, y as an action, and c as the opposite of a reward.

RL Problem

Policy Model

Control variables being continuous, we consider

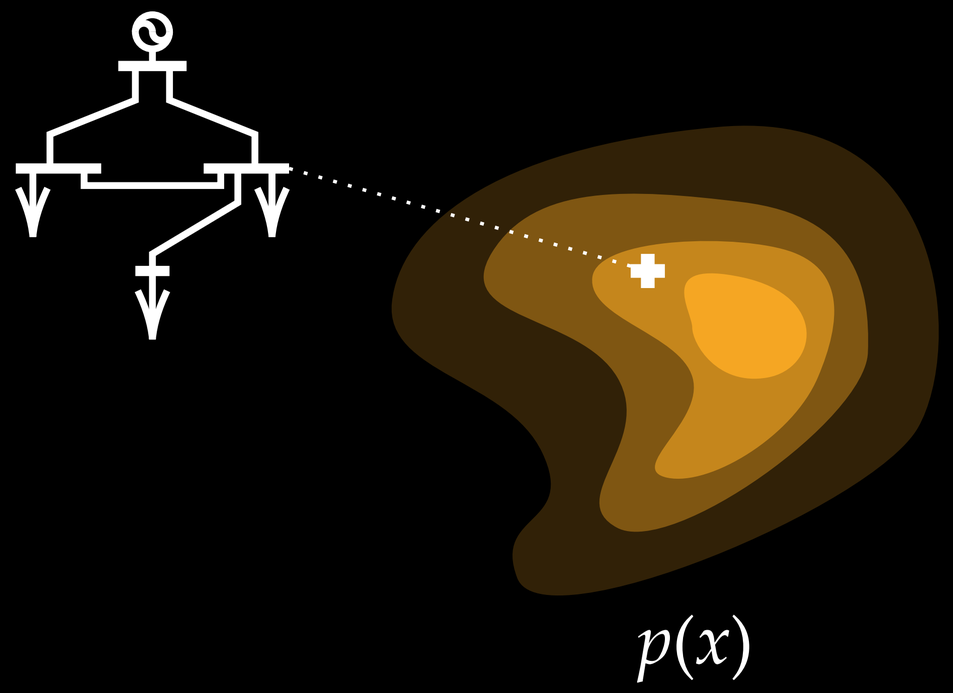
$$\Pi_\theta(\,\cdot\,|\,x) = \mathcal{N}(f_\theta(x), \sigma^2 \mathbb{1}),$$

where $\sigma > 0$ is fixed, $\mathbb{1}$ is the identity matrix, and f_θ is a neural network.

Hyper Heterogeneous Multi Graphs (H2MGs)

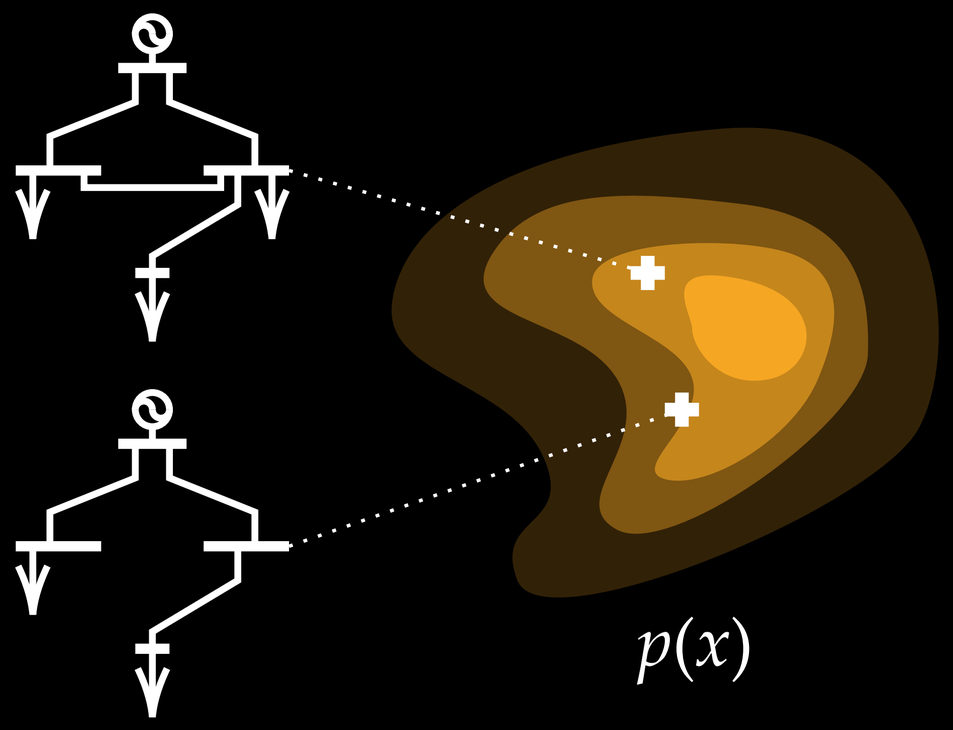
H2MGs

Operating Conditions Distribution



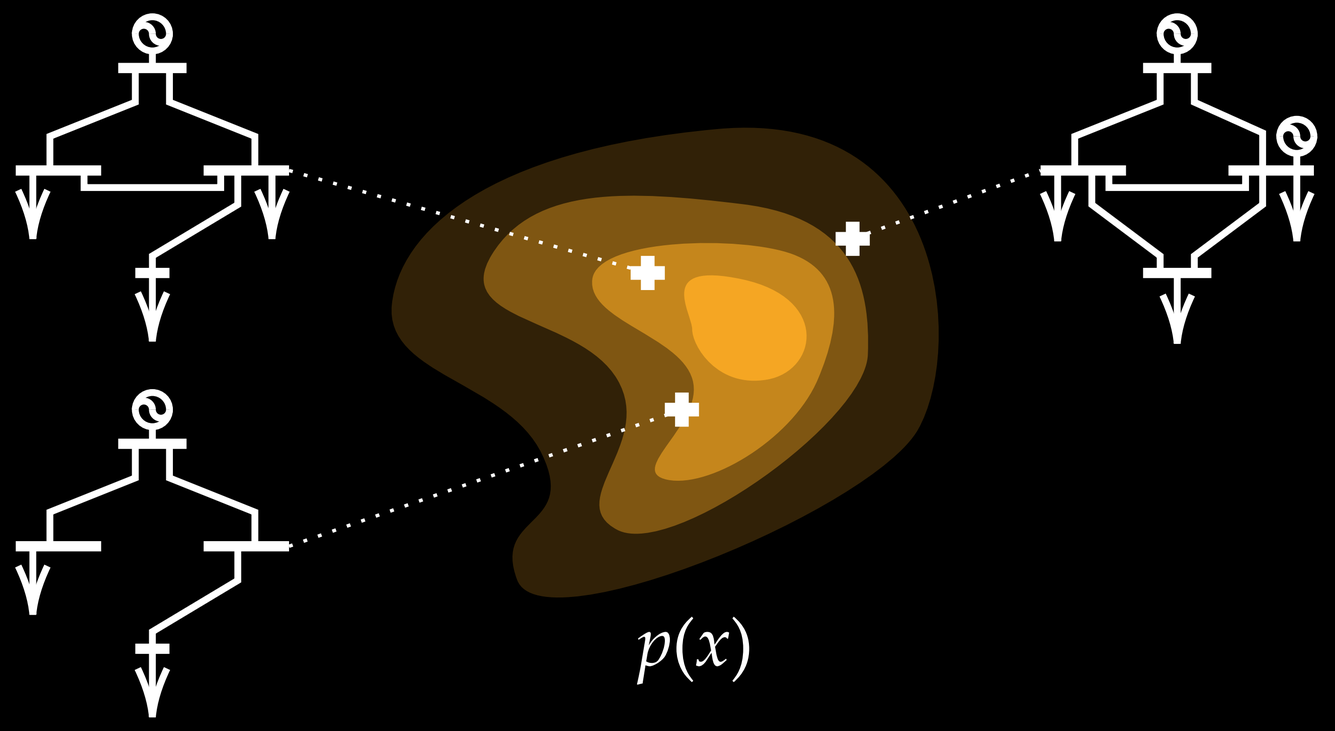
H2MGs

Operating Conditions Distribution



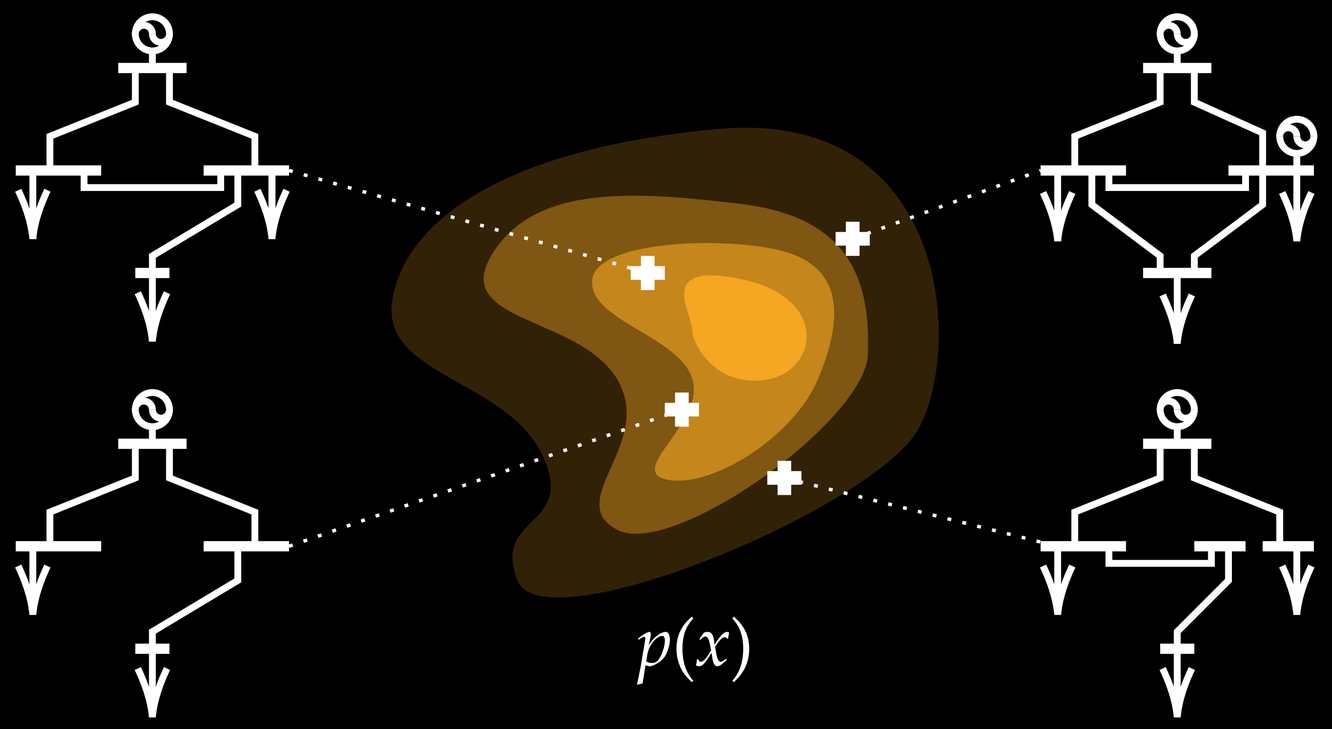
H2MGs

Operating Conditions Distribution



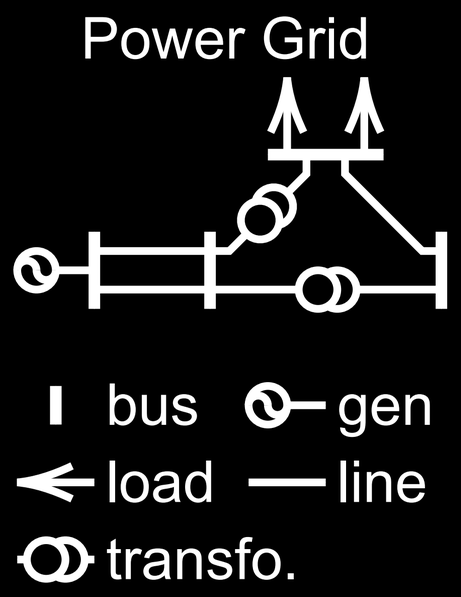
H2MGs

Operating Conditions Distribution



H2MGs

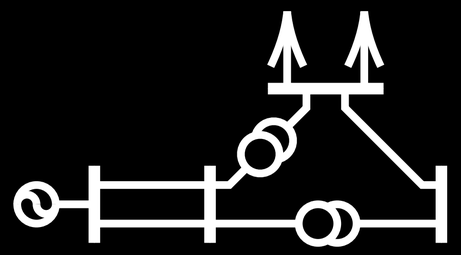
Data Representation



H2MGs

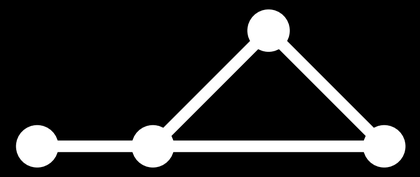
Data Representation

Power Grid



bus gen
load line
transfo.

Standard Graph

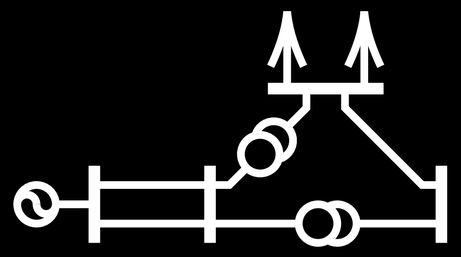


node
edge

H2MGs

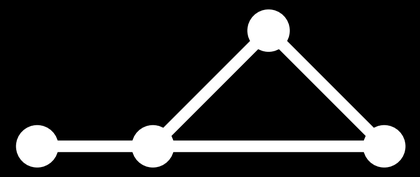
Data Representation

Power Grid



bus gen
load line
transfo.

Standard Graph

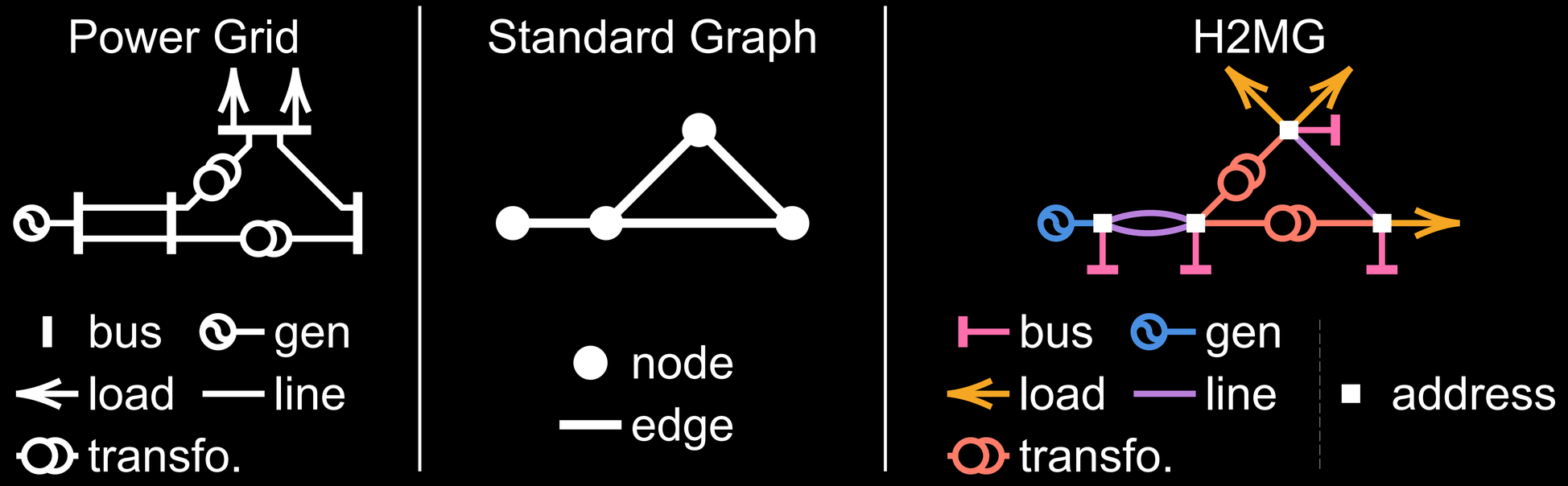


node
edge

Information loss !

H2MGs

Data Representation



H2MGs

Need for a Better Representation

- Actual power grids are :
  - Hyper graphs : Composed of hyper-edges of various orders,
  - Heterogeneous Graphs : Multiple classes of hyper-edges,
  - Multi Graphs : Multiple hyper-edges of the same class can be collocated.

B. Donon, “Deep statistical solvers & power systems applications,”

H2MGs

Graphical Structure

- Assets are referred to as *hyper-edges*.
- We denote by $\mathscr{C} = \{\text{bus, gen, load, line}, \dots\}$ the set of all classes of hyper-edges.
- For all class $c \in \mathscr{C}$, we denote by $\mathscr{E}_x^c$ the set of hyper-edges of class c.
  - Example : $\mathscr{E}_x^{line}$ is the set of transmission lines available in x.

B. Donon, “Deep statistical solvers & power systems applications,”

H2MGs

Graphical Structure

- Hyper-edges in x are interconnected through a set of *addresses* A_x.
- Addresses serve as interface between hyper-edges, and do not bear any numerical features.
- All hyper-edges of the same class c are connected to the same amount of addresses, through their ports $\mathcal{O}^c$.
  - A port $o \in \mathcal{O}^c : \mathcal{E}_x^c \to A_x$ is a mapping from an hyper-edge to an address.

B. Donon, “Deep statistical solvers & power systems applications,”

H2MGs

Graphical Structure

- Let $a \in A_x$ be an address.
- We define its neighborhood by :

$$N_x(a) = \{(c, e, o) \,|\, c \in \mathscr{C}, e \in \mathscr{E}_x^c, o \in \mathscr{O}^c, o(e) = a\}.$$

B. Donon, “Deep statistical solvers & power systems applications,”

H2MGs

Feature Vectors

- All hyper-edges of the same class bear feature vectors of the same size,

$$x = (x_e^c)_{c \in \mathscr{C}, e \in \mathscr{E}_x^c}.$$

B. Donon, “Deep statistical solvers & power systems applications,”

Neural Network Architecture

Neural Network Architecture

Overview

- Our policy Π_θ requires a trainable function f_θ to compute the mean of a Gaussian distribution.
- f_θ should take an operating condition x as input, and return a series of voltage setpoint values for each available generator.

B. Donon, “Deep statistical solvers & power systems applications,”

Neural Network Architecture

Encoding

At first, we encode feature vectors of all hyper-edges.

$$\forall c \in \mathscr{C}, \forall e \in \mathscr{E}_x^c, \qquad \tilde{x}_e^c = \Xi_\theta^c(x_e^c)$$

B. Donon, “Deep statistical solvers & power systems applications,”

Neural Network Architecture

Interaction

Then, we associate all addresses with latent coordinates $(h_a)_{a \in A_x}$, which we initialize at the origin.

$$\forall a \in A_x, \qquad h_a(\tau = 0) = [0,\ldots,0]$$

B. Donon, “Deep statistical solvers & power systems applications,”

Neural Network Architecture

Interaction

- Addresses then evolves in this latent space between $\tau = 0$ and 1 according to the following dynamical system (Neural Ordinary Differential Equation) :

$$\forall a \in A_x, \qquad \frac{dh_a}{d\tau} = \tanh\left(\sum_{(c,e,o)\in N_x(a)} \Phi_\theta^{c,o}(\tilde{x}_e^c, h_e(\tau), \tau) \right).$$

B. Donon, “Deep statistical solvers & power systems applications,”

Neural Network Architecture

Decoding

- Finally, we use the resulting latent state to produce a prediction for all hyper-edges that require one :

$$\forall c \in \mathscr{C}, \forall e \in \mathscr{E}_x^c, \qquad \mu_e^c = \Psi_\theta^c(\tilde{x}_e^c, h_e(1)).$$

B. Donon, “Deep statistical solvers & power systems applications,”

Neural Network Architecture

Overall System

- Here is the whole process,

$$\forall (c,e), \qquad \tilde{x}_e^c = \Xi_\theta^c(x_e^c),$$

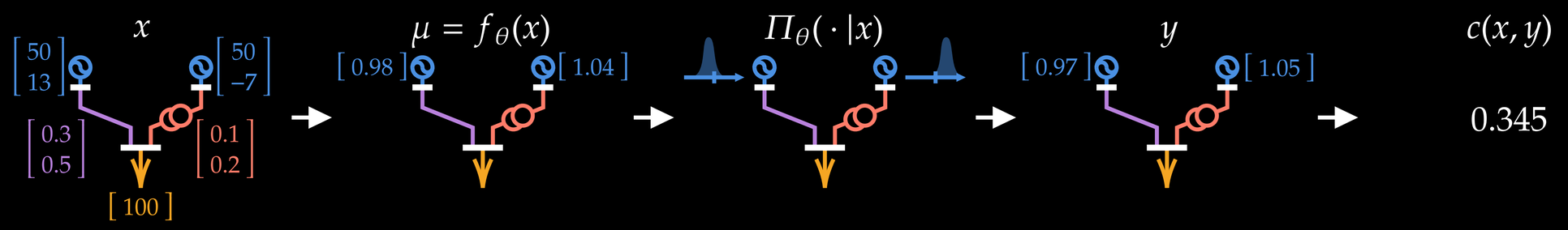
$$\forall a, \qquad h_a(0) = [0,\ldots,0],$$

$$\forall a, \qquad \frac{dh_a}{d\tau} = \tanh\left(\sum_{(c,e,o)\in N_x(a)} \Phi_\theta^{c,o}(\tilde{x}_e^c, h_e(\tau), \tau)\right)$$

$$\forall (c,e), \qquad \mu_e^c = \Psi_\theta^c(\tilde{x}_e^c, h_e(1))$$

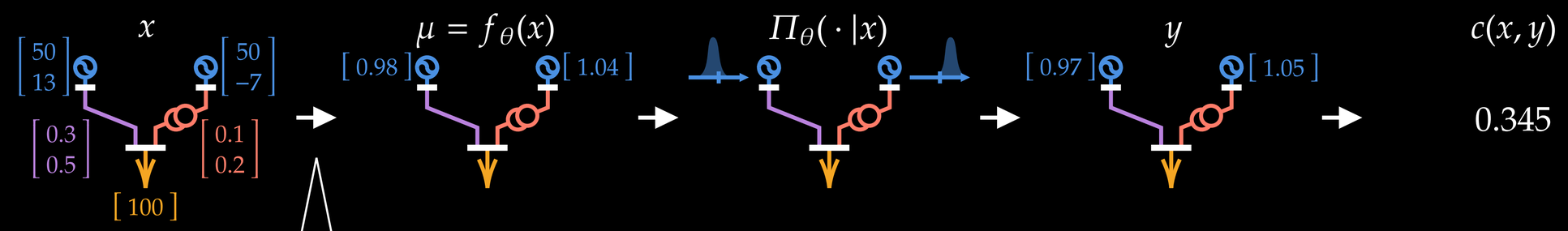
Neural Network Architecture

Pipeline



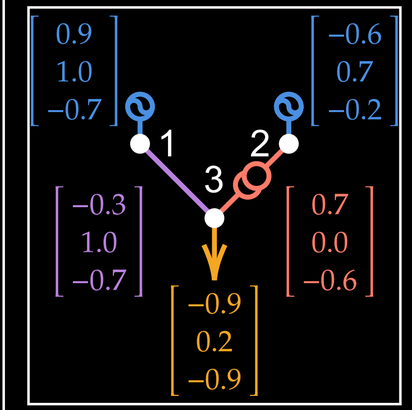
Neural Network Architecture

Pipeline



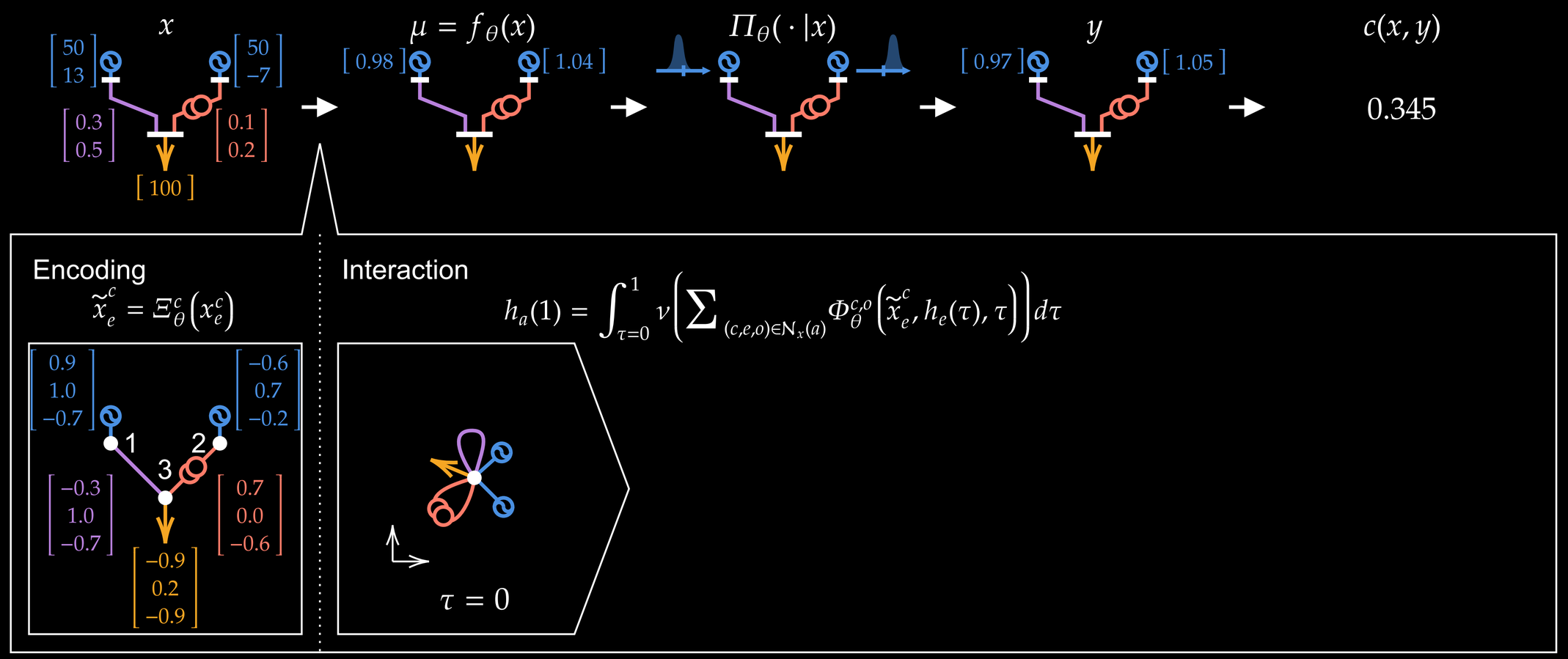
Encoding

$$\tilde{x}_e^c = \Xi_\theta^c\left(x_e^c\right)$$



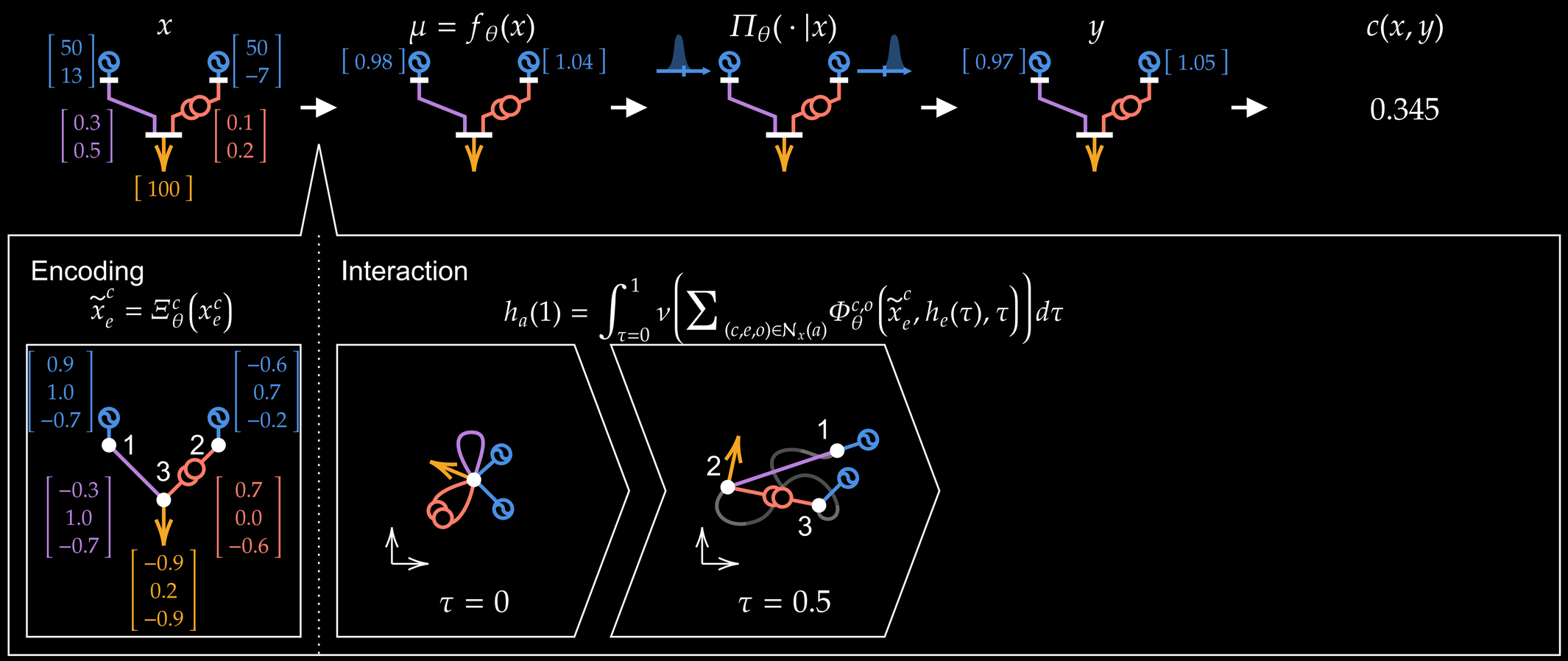
Neural Network Architecture

Pipeline



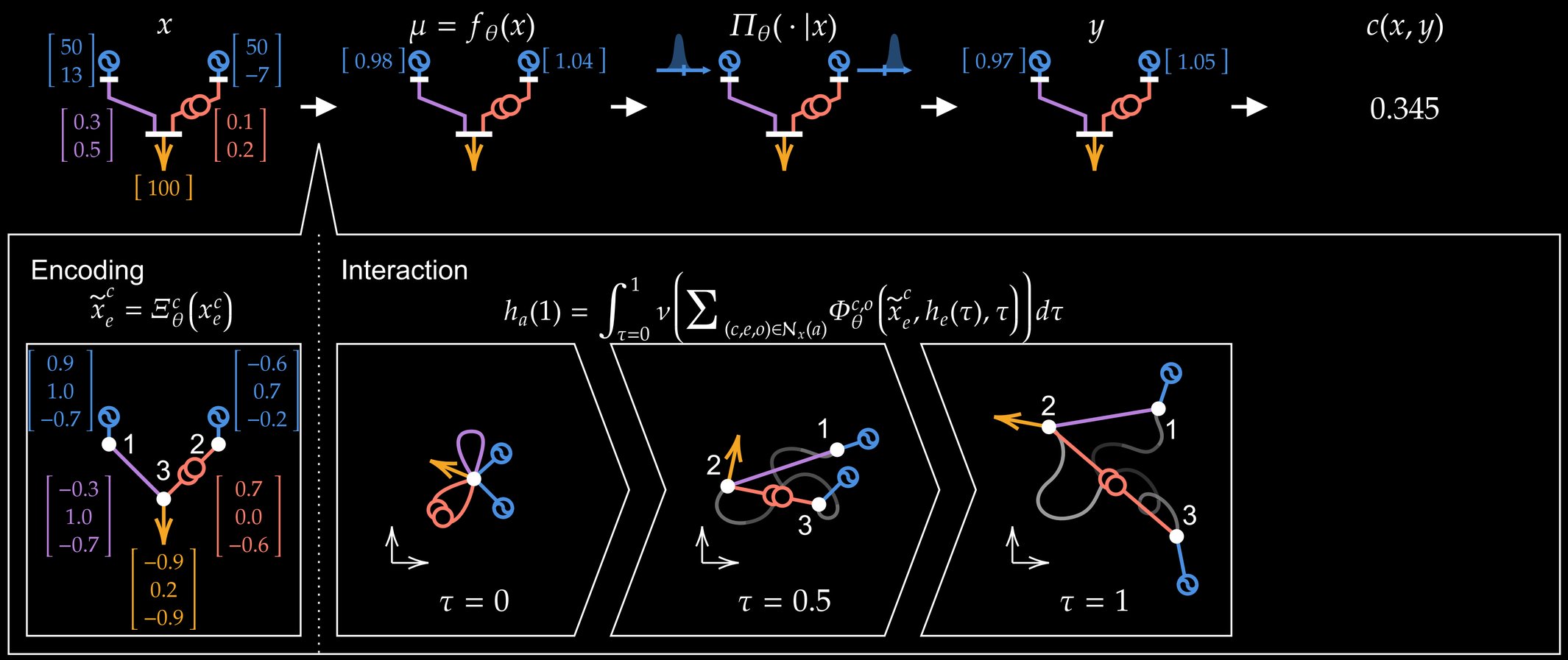
Neural Network Architecture

Pipeline



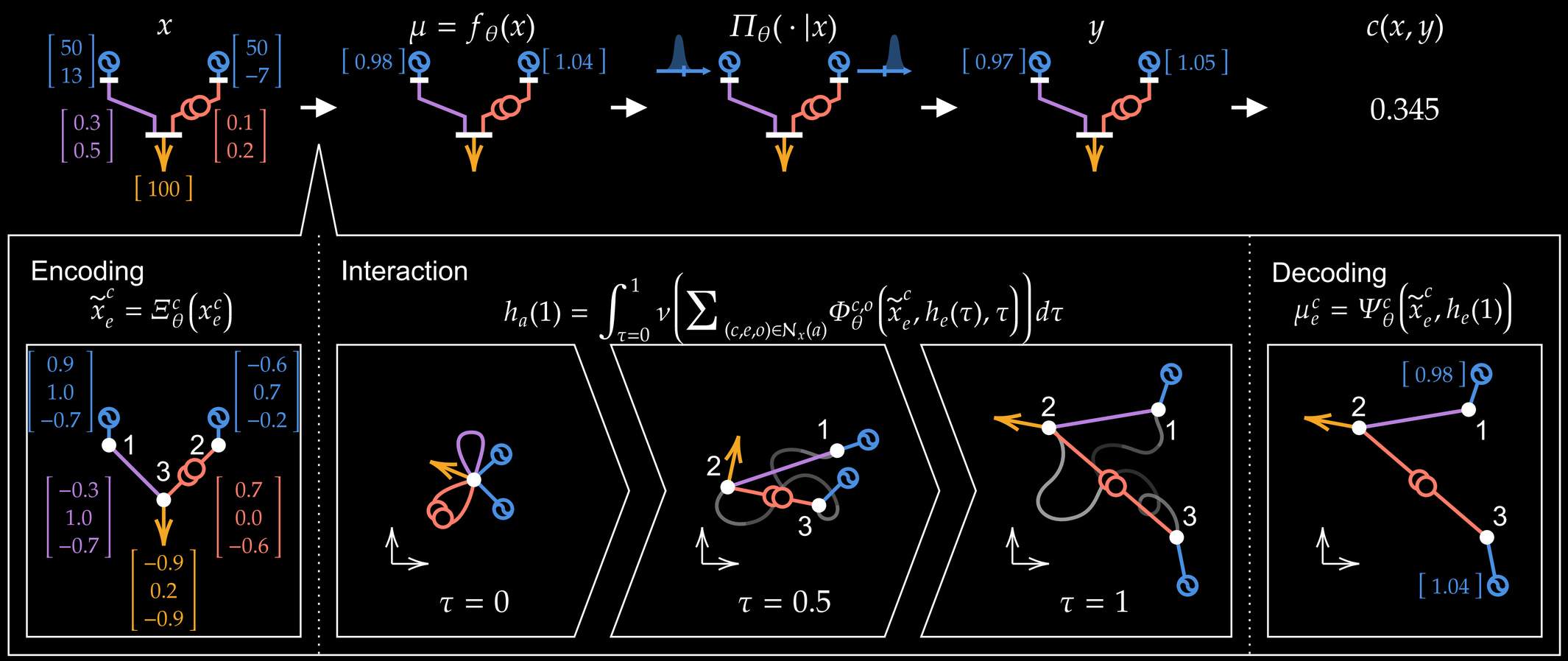
Neural Network Architecture

Pipeline



Neural Network Architecture

Pipeline



Policy Training

Policy Training

Algorithm

- Because of the absence of sequentiality, we have chosen a basic RL method called *REINFORCE*.

- For a certain x, we have

$$\nabla_\theta \mathbb{E}_{y\sim\Pi_\theta(\cdot|x)}\left[c(x,y)\right] = \mathbb{E}_{y\sim\Pi_\theta(\cdot|x)}\left[c(x,y)\,\nabla_\theta \log \Pi_\theta(y\,|\,x)\right].$$

- For a given x, we draw multiple actions $y \sim \Pi_\theta(\,\cdot\,|\,x)$ to obtain an empirical estimation of the right term.

Experiments

Experiments

Missing Data

- Artificial data for now.
- Results have not been published for now, but basically it works quite well !
- It works on datasets with varying number of generators, lines, transformers, etc.
  - Compatible with real-life data !

Future Work

Future Work

Main directions

- Improve sample efficiency of our training pipeline.
- Implement on real-life data.
- Consider discrete variables.
- Address other Power Systems issues.

Thank you !