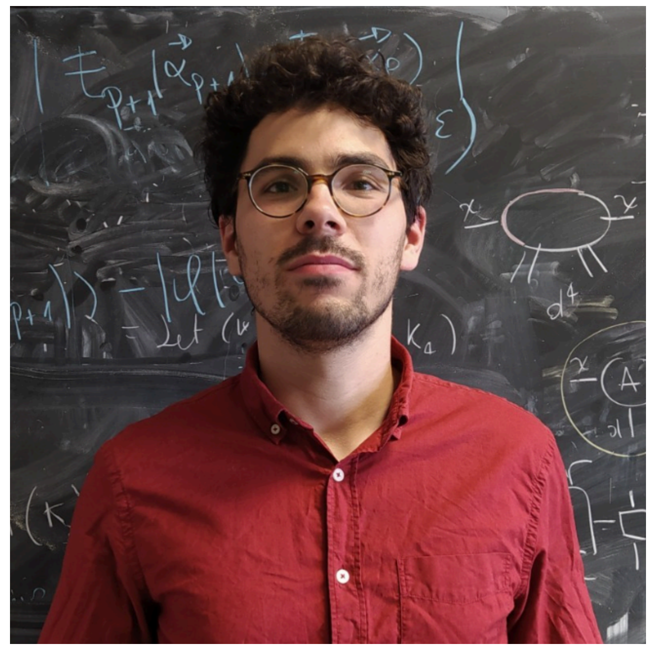
Quantum optimization of Binary Neural Networks

Pietro Torta (SISSA, Trieste)

QTML

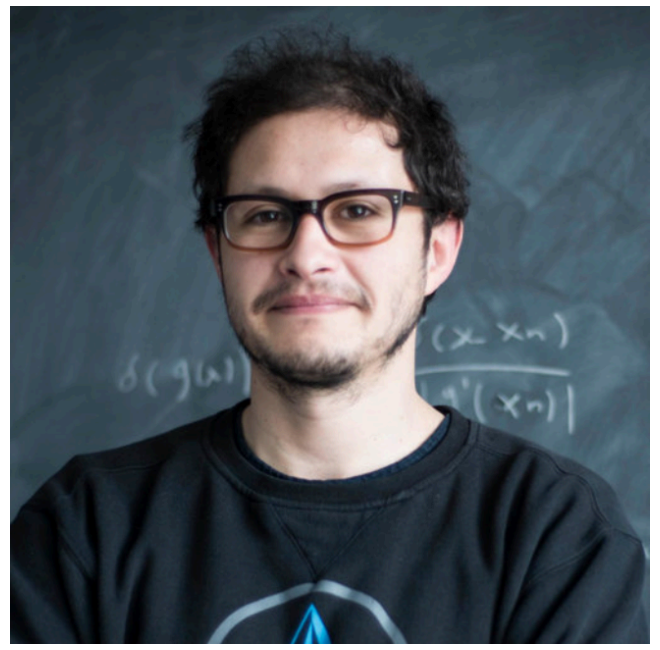
23/11/2023

Guglielmo Lami



SISSA

Juan Carrasquilla

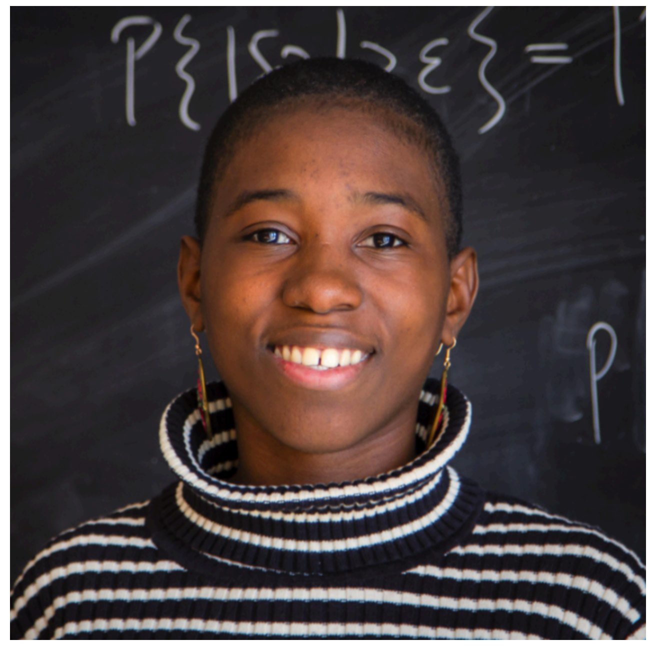


VECTOR INSTITUTE

Mario Collura

SISSA

Estelle Inack



PERIMETER INSTITUTE

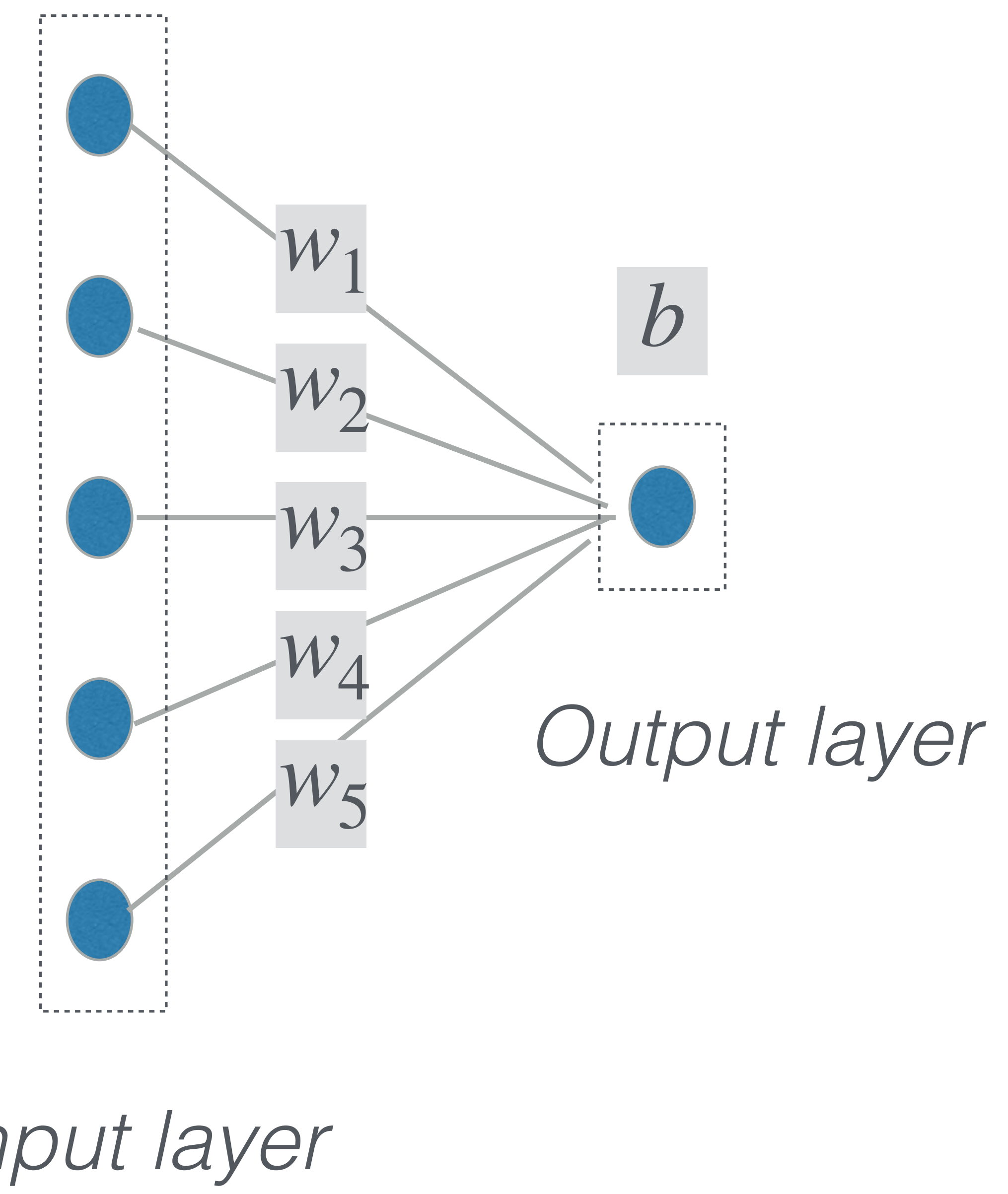
Outline

- Binary Neural Networks (BiNNs for friends): what and why?
- BiNNs training and quantum hypernetworks
- Our proposal: A Variational Quantum Algorithm (VQA) to train BiNNs
- Preliminary results and open challenges

Binary Neural Networks (BiNNs): **what**?

Classical Neural Networks

Simple example:
a binary perceptron



Binary weights and biases: interpret them as classical **spins** $w_j = \pm 1 \quad b = \pm 1$

Binary Neural Networks (BiNNs): **what**?

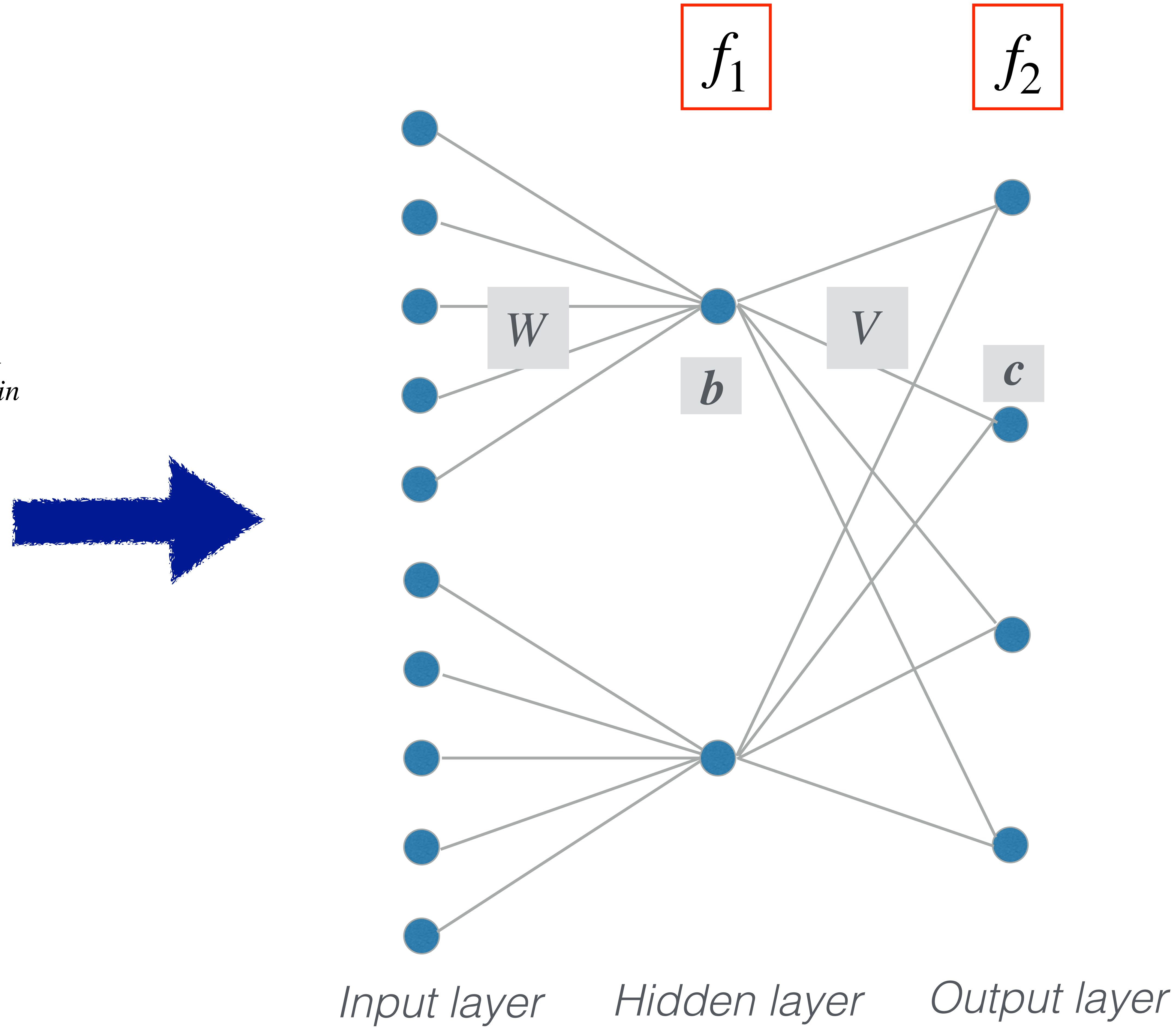
Multi-layer feed-forward BiNNs

Input (data)

$\boldsymbol{x} \in \{-1, +1\}^{N_{in}}$

or

$\boldsymbol{x} \in \mathbb{R}^{N_{in}}$



Output (label)

$$f_2\Big(V\left[f_1\left(W\boldsymbol{x}+\boldsymbol{b}\right)\right]+\boldsymbol{c}\Big) \equiv y$$

Binary Neural Networks (BiNNs): **what**?

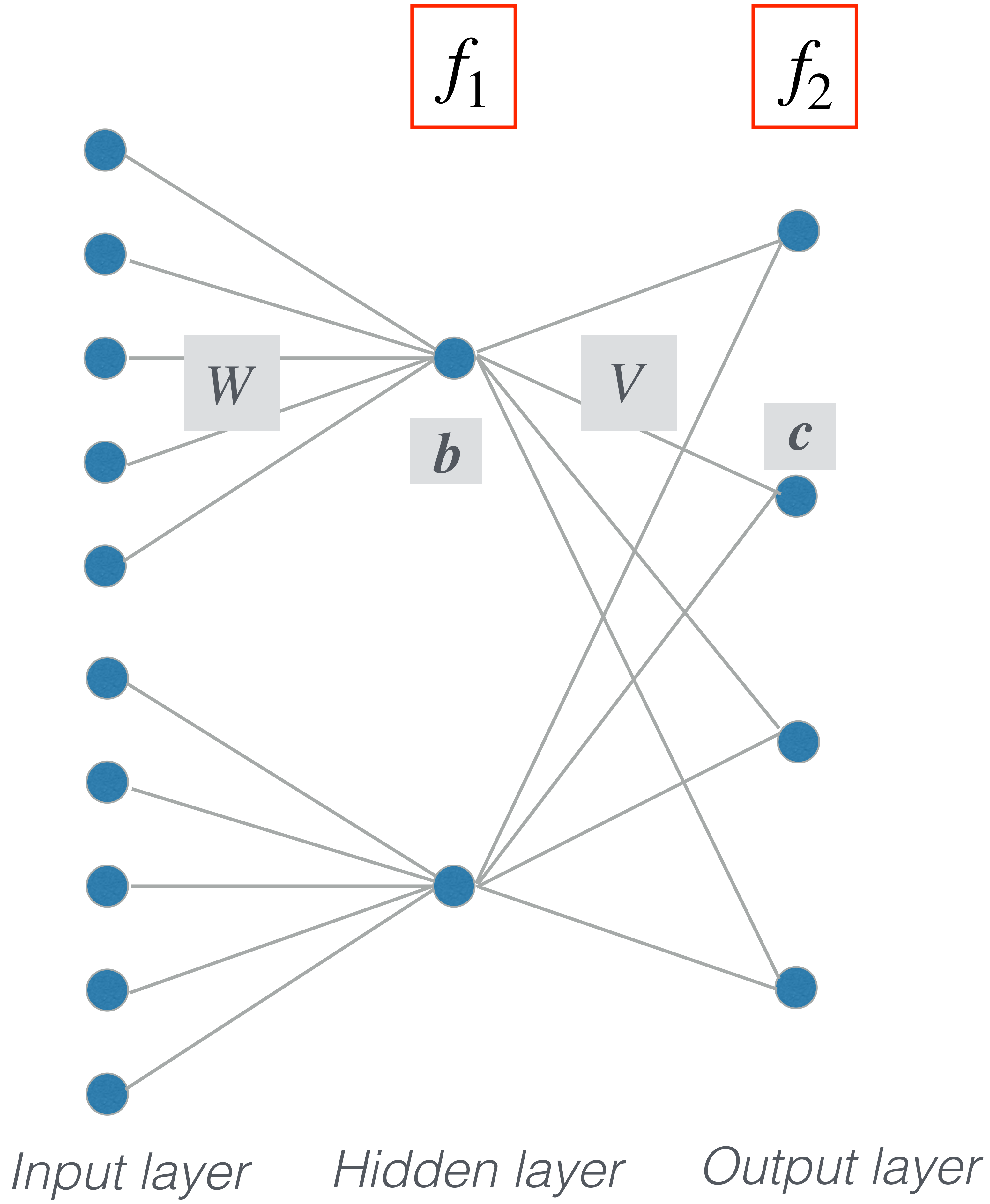
Multi-layer feed-forward BiNNs

Input (data)

$\boldsymbol{x} \in \{-1, +1\}^{N_{in}}$

or

$\boldsymbol{x} \in \mathbb{R}^{N_{in}}$



Output (label)

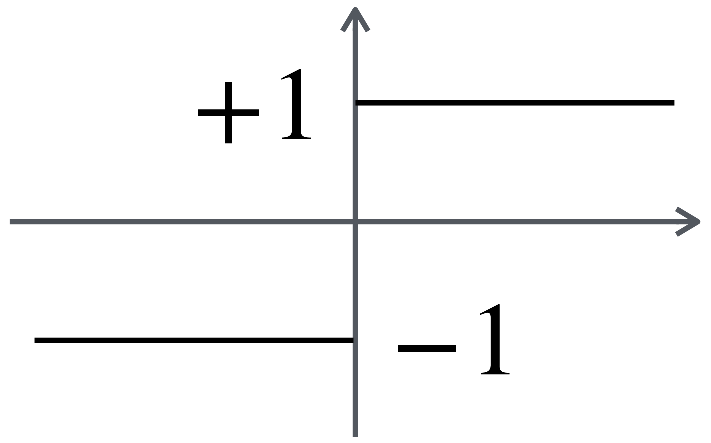
$$f_2\Big(V\big[f_1(W\boldsymbol{x}+\boldsymbol{b})\big]+\boldsymbol{c}\Big) \equiv y$$

with $\{\sigma_\alpha\}_{\alpha=1}^{N} \equiv \{W, \boldsymbol{b}, V, \boldsymbol{c}\}$
N : total number of binary parameters

Binary Neural Networks (BiNNs): **what**?

Multi-layer feed-forward BiNNs

$$\boxed{f_1} = \boxed{f_2} = \mathrm{sgn}(\cdot)$$



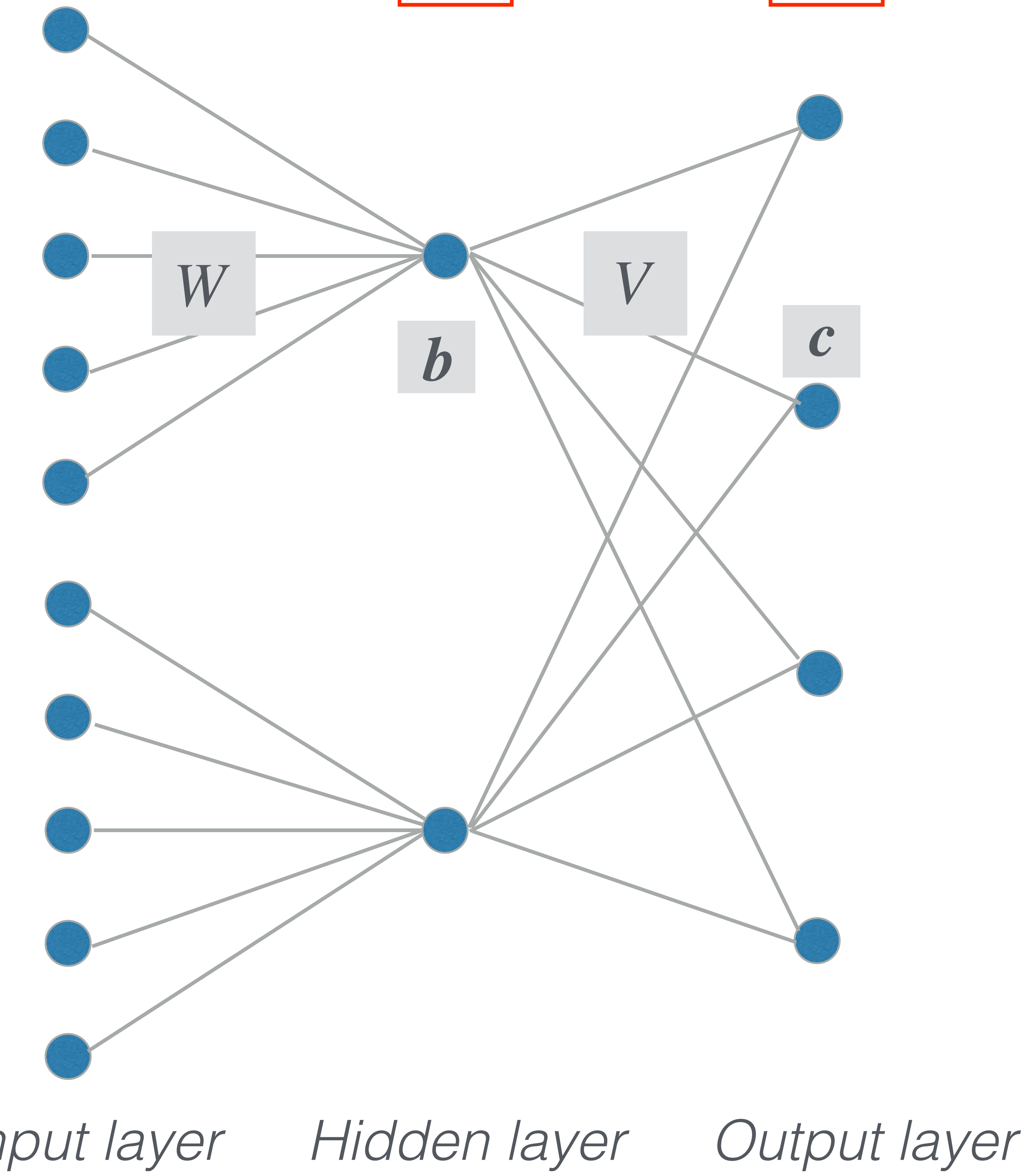
Non differentiable!

Input (data)

$$\boldsymbol{x} \in \{-1,+1\}^{N_{in}}$$

or

$$\boldsymbol{x} \in \mathbb{R}^{N_{in}}$$



Output (label)

$$f_2\Big(V\left[f_1\left(W\boldsymbol{x}+\boldsymbol{b}\right)\right]+\boldsymbol{c}\Big) \equiv y$$

with $\{\sigma_\alpha\}_{\alpha=1}^{N} \equiv \{W, \boldsymbol{b}, V, \boldsymbol{c}\}$
N : total number of binary parameters

Binary Neural Networks (BiNNs): **what**?

Supervised Learning (a.k.a. training)

Training and Test Set $\{\boldsymbol{x}^{(\mu)}, \bar{y}^{(\mu)}\}_{\mu=1}^{N_{train}}$ $\{\boldsymbol{x}^{(\mu)}, \bar{y}^{(\mu)}\}_{\mu=1}^{N_{test}}$

$\boldsymbol{x}^{(\mu)}$ data point (*pattern*) to classify

$\bar{y}^{(\mu)}$ prescribed *label*

Classical **Loss Function to minimize during training**

$$\mathscr{L}\left(\{\sigma_\alpha\}\right) = \sum_{\mu=1}^{N_{train}} l\left(\mathrm{NN}\left(\boldsymbol{x}^{(\mu)}, \{\sigma_\alpha\}\right), \bar{y}^{(\mu)}\right)$$

N-bit real Boolean function $\mathscr{L} : \{0,1\}^N \to \mathbb{R}$

Binary Neural Networks (BiNNs): **why**?

- Deep NN models are **computationally expensive**: **memory, energy…**
- **Require a lot of GPUs:** difficult to deploy on **small devices (a smartphone)**

BiNNs

Pros

- Replace most float-arithmetic operations with bit-wise operations: **from 32 to 1 bit.**
- Can reduce *storage*, *computational cost*, and *energy consumption*
- Implementable on **specialized hardware**
- Robustness against adversarial attacks

Binary Neural Networks (BiNNs): **why**?

- Deep NN models are **computationally expensive**: **memory, energy…**
- **Require a lot of GPUs:** difficult to deploy on **small devices (a smartphone)**

$\boxed{\text{BiNNs}}$

Cons

- **Training** is **challenging**
  1. **Standard backpropagation:** it cannot be applied (non-differentiable activations!)
  2. **Binarization after training on float weights:** it does not work
- **Other challenges: architectural design, hyper-parameter tuning: hard combinatorial optimization tasks**

Binary Neural Networks (BiNNs): **how**?

Our proposal: to use ***Quantum hypernetworks to train BiNNs***

- **Only binary weights during training**
- **Unify the search over parameters, hyper-parameters, and architectures in a single loop**

Quantum Hypernetwork

▶ **Hypernetwork**: a Neural Network used to generate the weights of another Neural Network

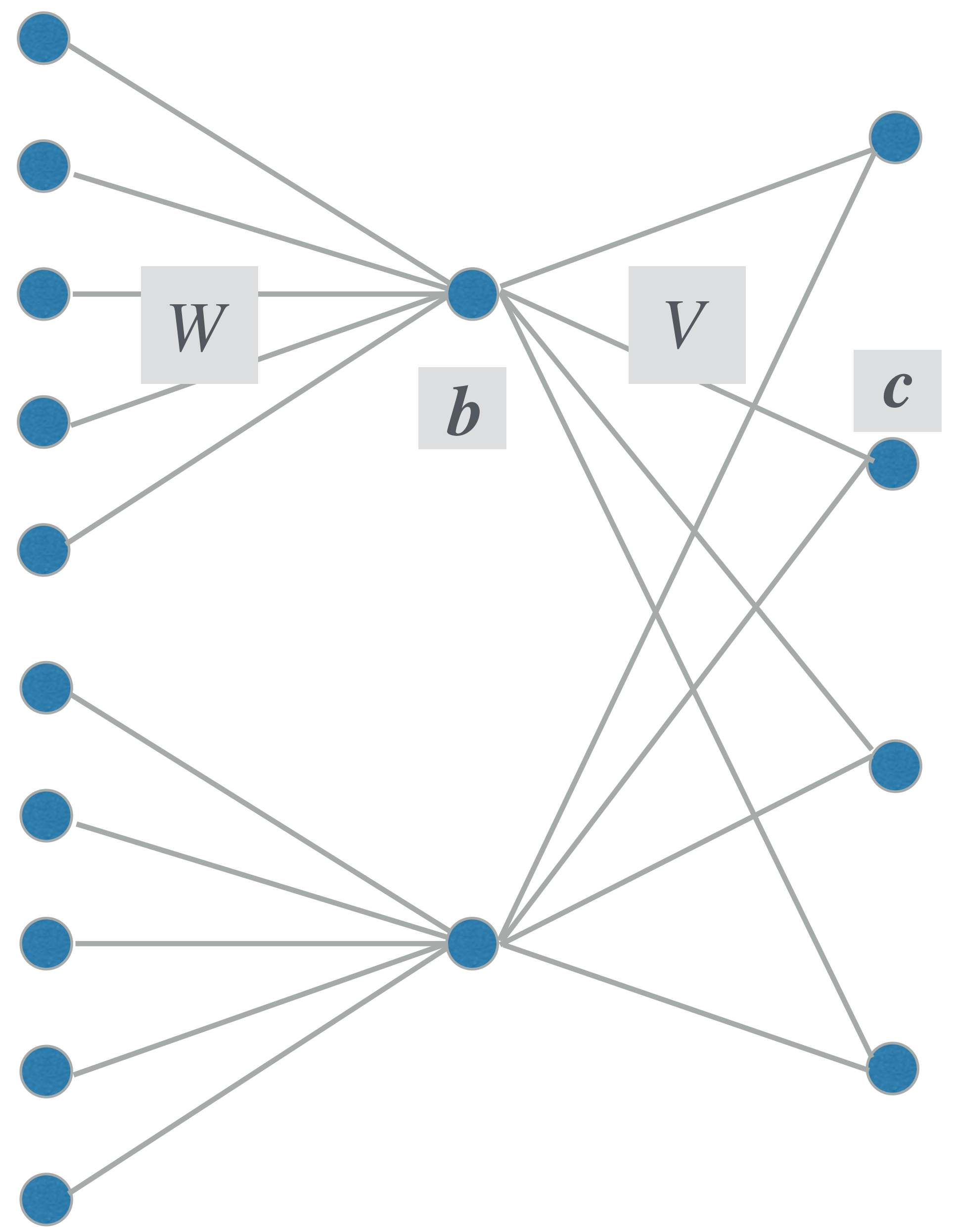
Quantum hypernetwork: quantum state that generates the weights of a classical BiNNs

$$|\psi\rangle = \sum_{i=1}^{N} \psi(\sigma_1, \ldots, \sigma_N) \, |\sigma_1, \ldots, \sigma_N\rangle$$

✦ **Quantum superposition of classical** BiNNs configurations

$$\hat{\sigma}^z_\alpha \, |\sigma_1, \ldots, \sigma_N\rangle = \sigma_\alpha \, |\sigma_1, \ldots, \sigma_N\rangle$$

✦ Gives a BiNNs configuration upon **measurement on** $|\psi\rangle$



Computational basis state corresponds to BiNN configuration

$$\{\sigma_\alpha\}_{\alpha=1}^{N} \equiv \{W, \boldsymbol{b}, V, \boldsymbol{c}\}$$

Quantum Hypernetwork

- We can exploit **quantum superposition** even further: train **different BiNN architectures** at the same time

Binary architectural choice OR binary hyper-parameter selection = additional qubit σ^*

$$|\psi\rangle = \sum_{i=1}^{N} \psi(\sigma_1 \ldots \sigma_N, \sigma^*) \, |\sigma_1 \ldots \sigma_N, \sigma^*\rangle$$

Example of architectural choice:
weights dropout

$\sigma^* = +1$ $\qquad$ $\sigma^* = -1$

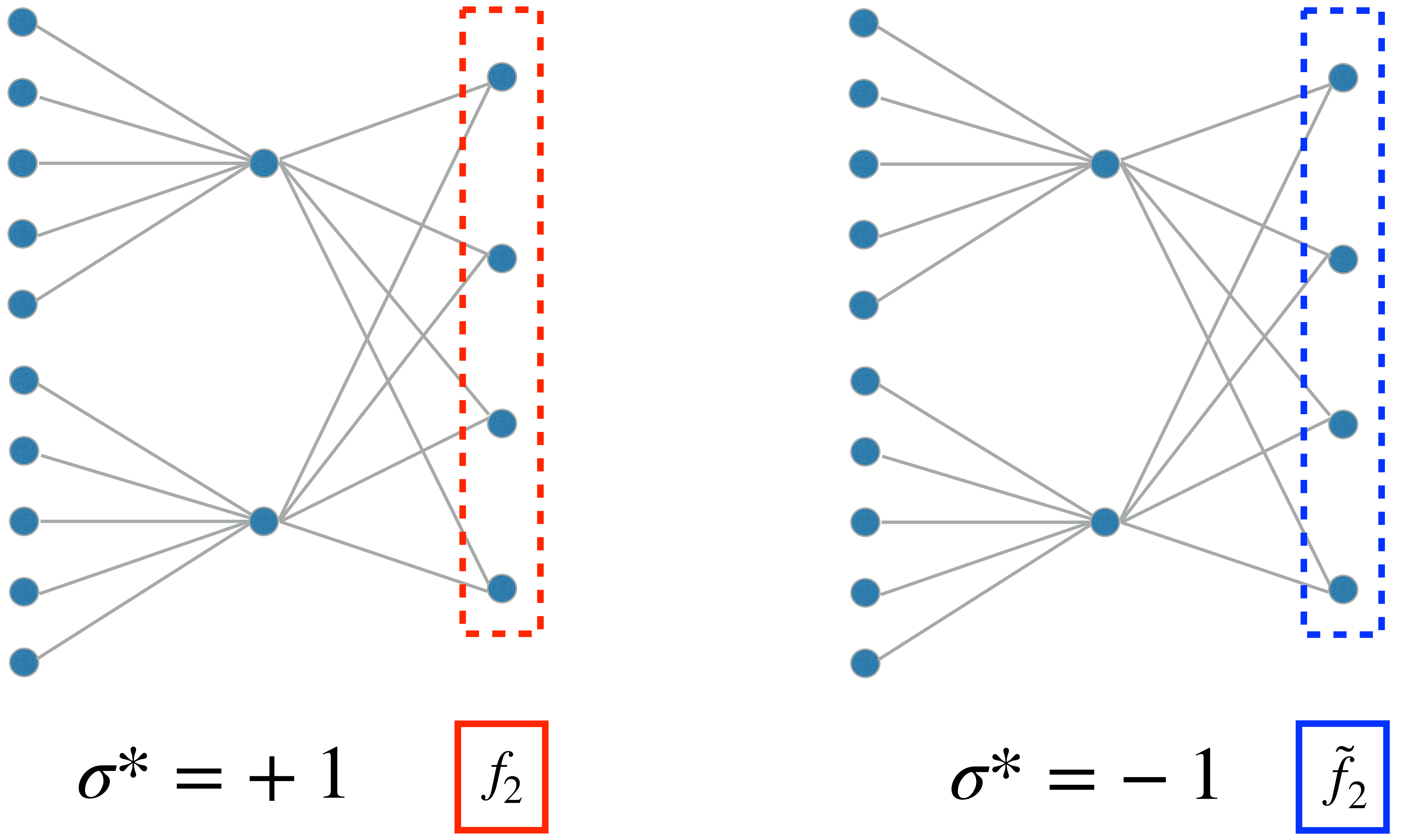
Quantum Hypernetwork

- We can exploit **quantum superposition** even further: train **different BiNN architectures** at the same time

Binary architectural choice OR binary hyper-parameter selection = additional qubit σ^*

$$|\psi\rangle = \sum_{i=1}^{N} \psi(\sigma_1 \ldots \sigma_N, \sigma^*) \, | \sigma_1 \ldots \sigma_N , \sigma^* \rangle$$

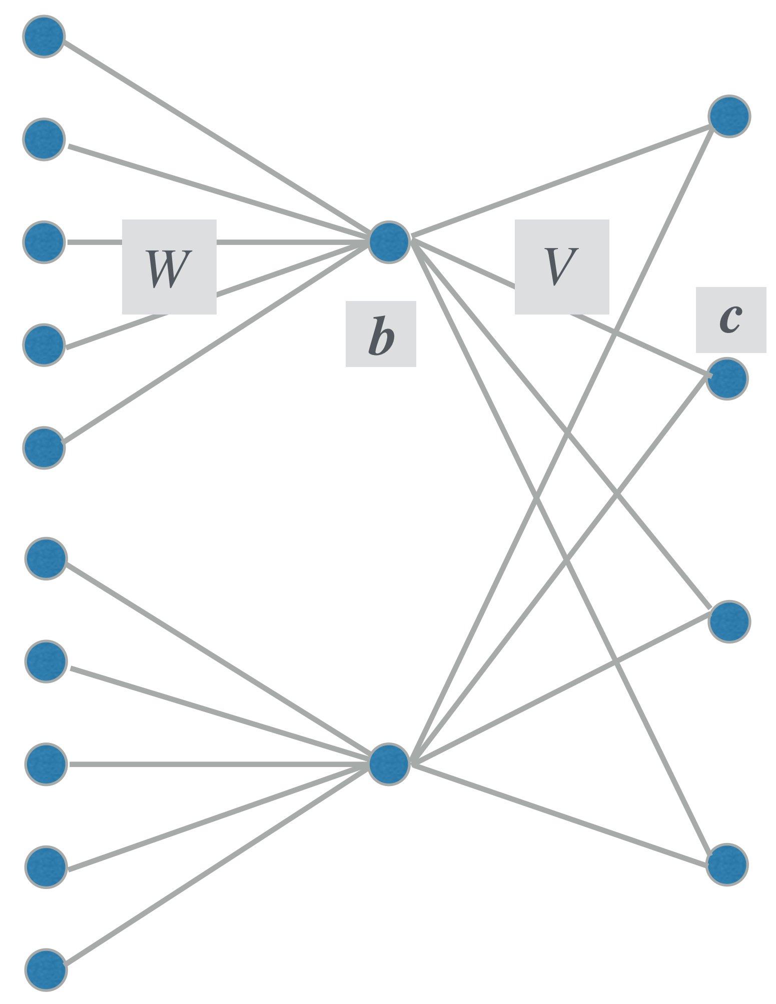
Example of hyper-parameter selection: activation in the last layer



Take-home message

The quantum hyper-network state $|\psi\rangle$ is now a Parameterized Quantum Circuit: we train it in a VQA

BiNN



N binary variables

$$\{\sigma_\alpha\}_{\alpha=1}^{N} \equiv \{W, \boldsymbol{b}, V, \boldsymbol{c}\}$$

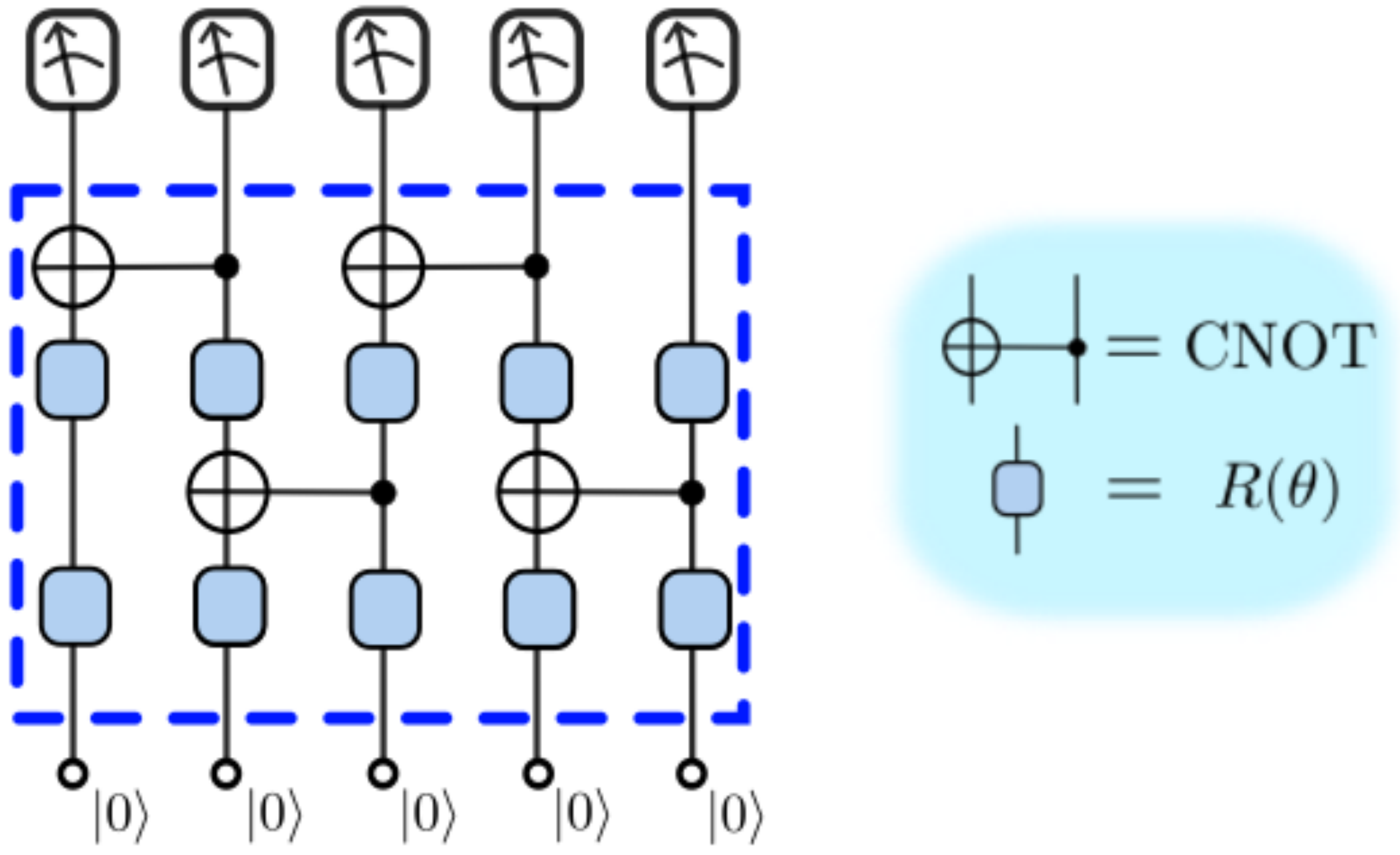
Classical Loss

$$\mathscr{L}\left(\{\sigma_\alpha\}_{\alpha=1}^{N}\right) = \sum_{\mu=1}^{N_{train}} l\left(\mathrm{NN}\left(\boldsymbol{x}^{(\mu)}, \{\sigma_\alpha\}\right), \bar{y}^{(\mu)}\right)$$

Stochastic Relaxation

$$p\left(\{\sigma_\alpha\}\right) = \left|\psi\left(\{\sigma_\alpha\}\right)\right|^2$$

Quantum Hypernetwork



d real angles $\vec{\theta}$

$$\{\theta_j\}_{j=1}^{d}$$

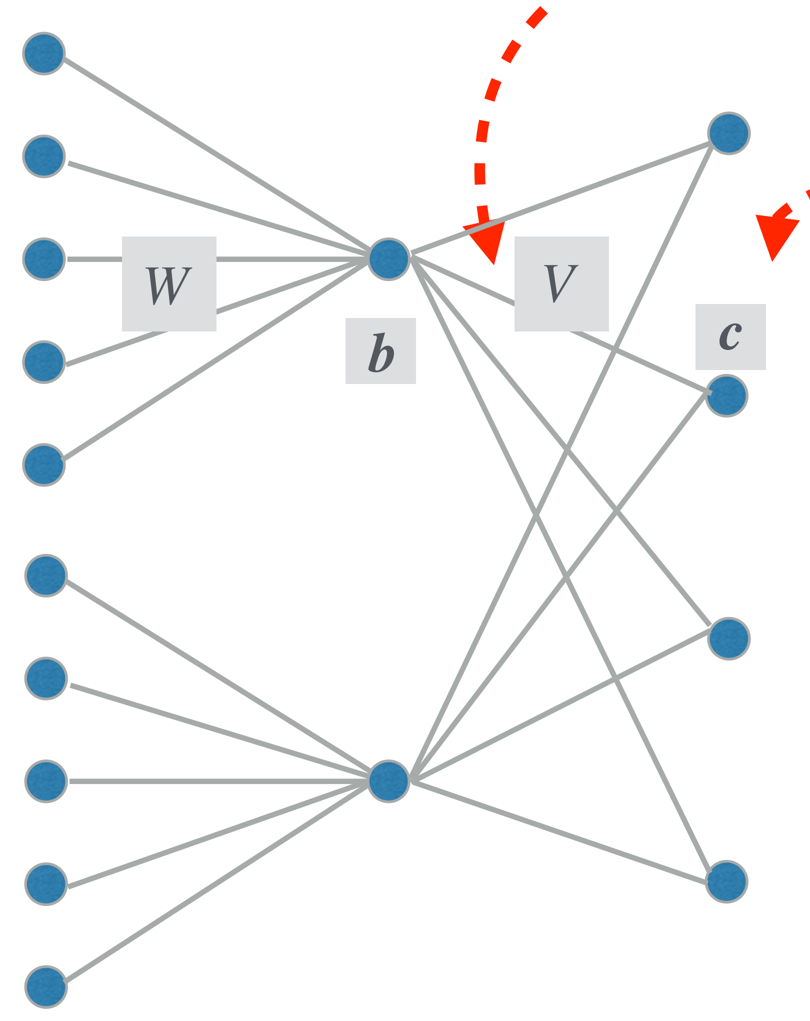
Quantum variational energy

$$E_{var}(\vec{\theta}) = \langle\psi(\vec{\theta})|\mathscr{L}\left(\{\hat{\sigma}_\alpha^z\}\right)|\psi(\vec{\theta})\rangle$$

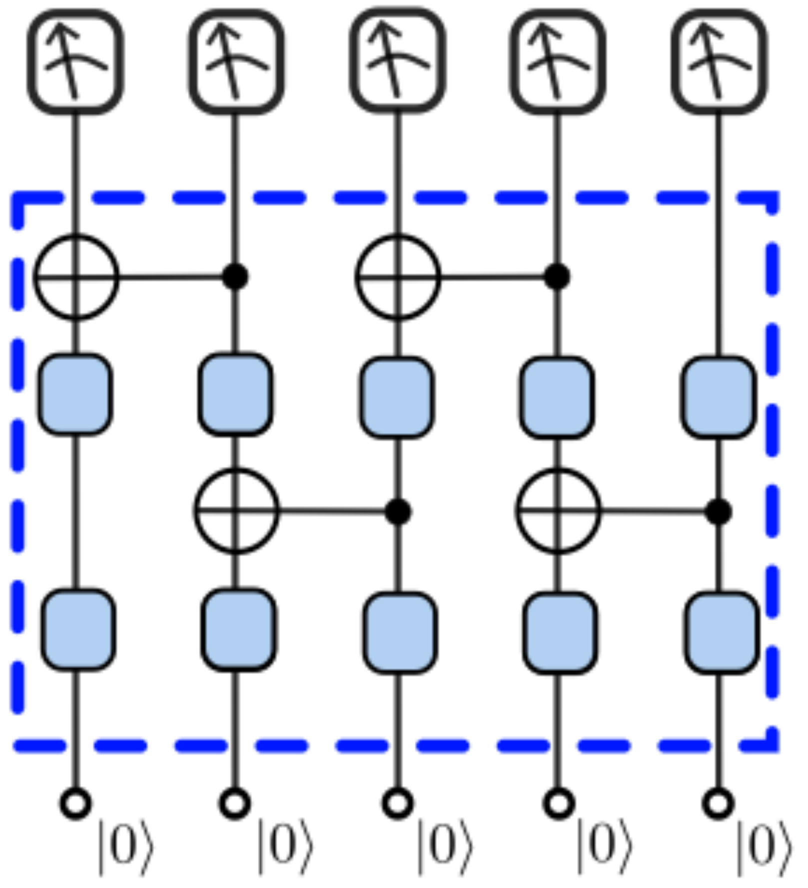
Take-home message

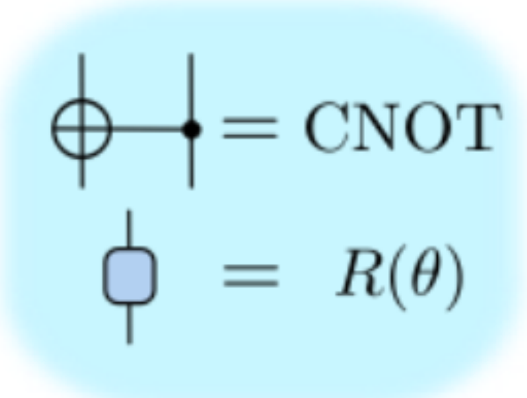
The quantum hyper-network state $|\psi\rangle$ is now a Parameterized Quantum Circuit: we train it in a VQA

BiNN



Quantum Hypernetwork





Stochastic Relaxation

$$p\left(\{\sigma_\alpha\}\right) = \left|\psi\left(\{\sigma_\alpha\}\right)\right|^2$$

N binary variables

$$\{\sigma_\alpha\}_{\alpha=1}^{N} \equiv \{W, \boldsymbol{b}, V, \boldsymbol{c}\}$$

d real angles $\vec{\theta}$

$$\{\theta_j\}_{j=1}^{d}$$

Classical Loss

$$\mathscr{L}\left(\{\sigma_\alpha\}_{\alpha=1}^{N}\right) = \sum_{\mu=1}^{N_{train}} l\left(\mathrm{NN}\left(\boldsymbol{x}^{(\mu)}, \{\sigma_\alpha\}\right), \bar{y}^{(\mu)}\right)$$

Quantum variational energy

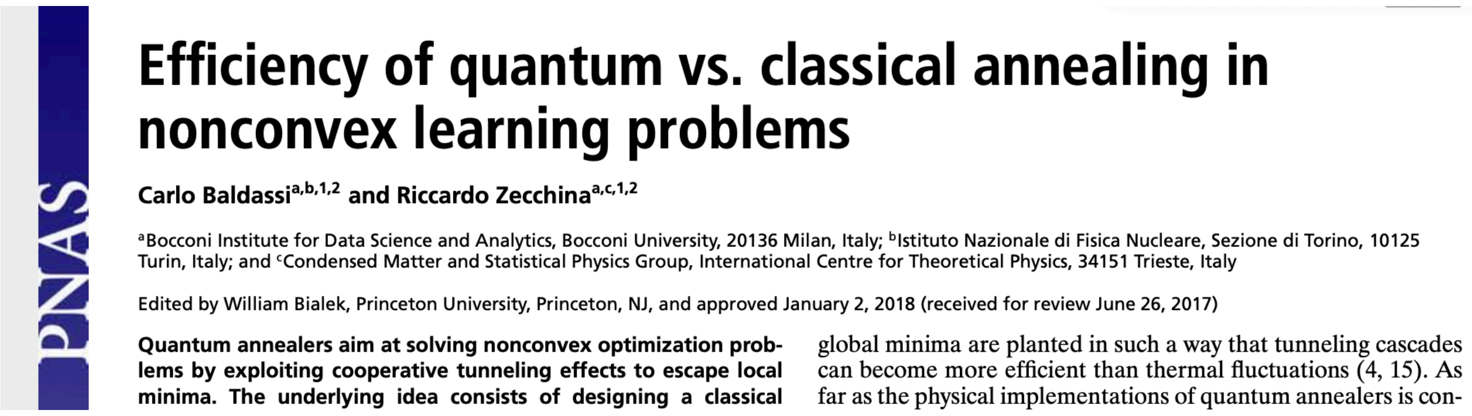
$$E_{var}\left(\vec{\theta}\right) = \langle\psi\left(\vec{\theta}\right)|\,\mathscr{L}\left(\{\hat{\sigma}^z_\alpha\}\right)|\psi\left(\vec{\theta}\right)\rangle$$

A few final comments

▶ Supervised learning of BiNNs**: hard binary optimization** beyond 2-bodies interaction **(beyond QUBO)**

$$\mathscr{L}\left(\{\hat{\sigma}_i^z\}\right) = \sum_{i=1}^{N} a_i \hat{\sigma}_i^z + \sum_{i=1}^{N} \sum_{j>i} b_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z + \sum_{i=1}^{N} \sum_{j>i} \sum_{k>j} c_{ijk} \hat{\sigma}_i^z \hat{\sigma}_j^z \hat{\sigma}_k^z + \dots$$

▶ **Why** should it work?


PNAS

Efficiency of quantum vs. classical annealing in nonconvex learning problems

Carlo Baldassi[a,b,1,2] and Riccardo Zecchina[a,c,1,2]

[a]Bocconi Institute for Data Science and Analytics, Bocconi University, 20136 Milan, Italy; [b]Istituto Nazionale di Fisica Nucleare, Sezione di Torino, 10125 Turin, Italy; and [c]Condensed Matter and Statistical Physics Group, International Centre for Theoretical Physics, 34151 Trieste, Italy

Edited by William Bialek, Princeton University, Princeton, NJ, and approved January 2, 2018 (received for review June 26, 2017)

Quantum annealers aim at solving nonconvex optimization problems by exploiting cooperative tunneling effects to escape local minima. The underlying idea consists of designing a classical

global minima are planted in such a way that tunneling cascades can become more efficient than thermal fluctuations (4, 15). As far as the physical implementations of quantum annealers is con-

- BiNNs optimal configurations $(\sigma_1, \dots, \sigma_N)$ tend to cluster in Hamming Distance
- & Quantum fluctuations are efficient to sample in these clusters

We want to run **large-scale simulations** with **shot noise** (we already have proofs-of-principle)

PT, G. B. Mbeng, C. Baldassi, R. Zecchina, G. E. Santoro, Quantum Approximate Optimization Algorithm applied to the binary perceptron, Physical Review B 107 (9), 094202 (2023)

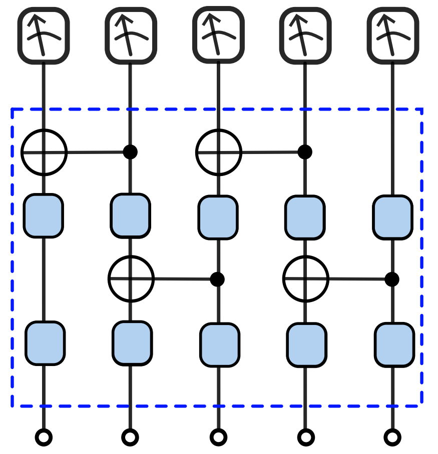
G. Lami, PT, G. E. Santoro, M. Collura, Quantum annealing for neural network optimization problems: A new approach via tensor network simulations, SciPost Phys. 14, 117 (2023)

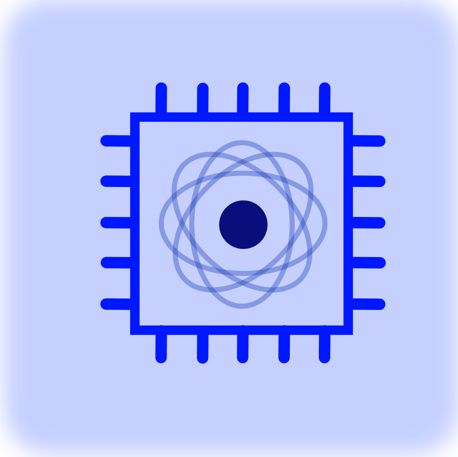
Methods

$$E_{var}(\vec{\theta}) = \langle \psi(\vec{\theta}) | \mathscr{L}\left(\{\hat{\sigma}^z_\alpha\}\right) | \psi(\vec{\theta}) \rangle \approx \frac{1}{N_s} \sum_{k=1}^{N_s} \mathscr{L}\left(\{\sigma_k\}\right)$$

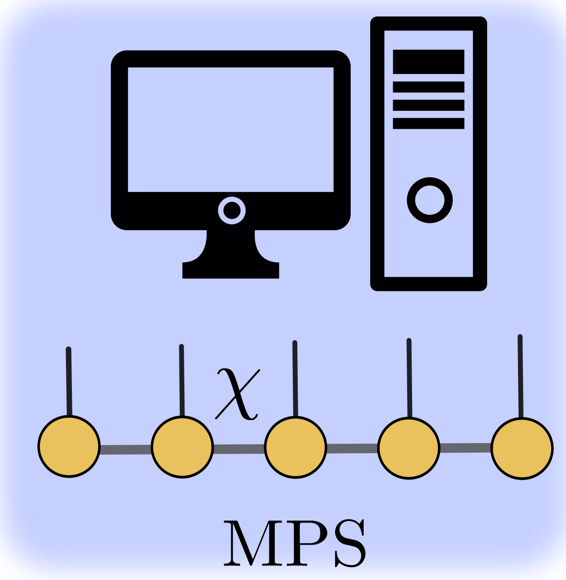
Sample mean over N_s shots

How to get them?





Measure a real quantum state



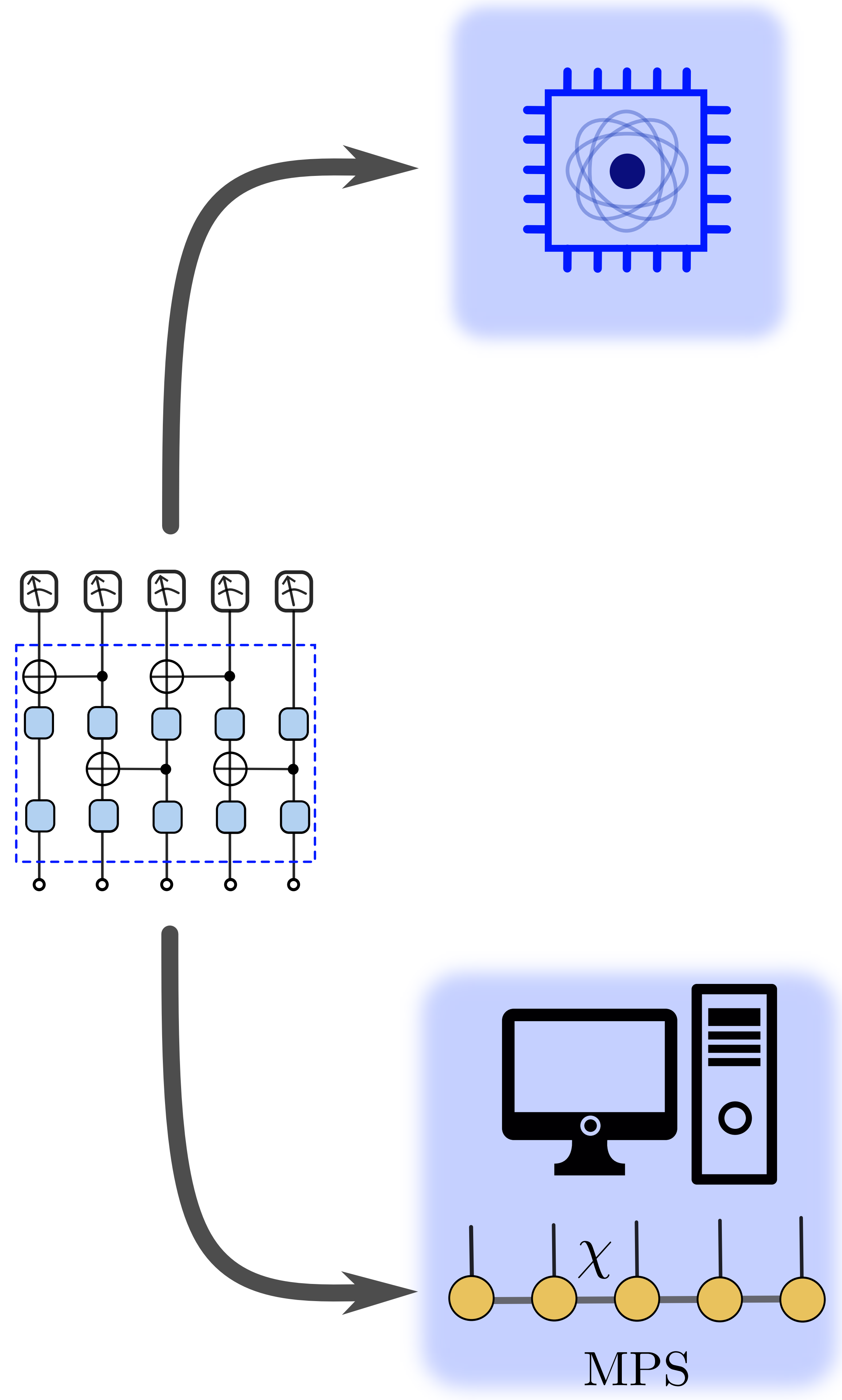
Sample a Matrix Product State

Methods

$$E_{var}(\vec{\theta}) = \langle \psi(\vec{\theta}) | \mathscr{L}\left(\{\hat{\sigma}^z_\alpha\}\right) | \psi(\vec{\theta}) \rangle \approx \frac{1}{N_s} \sum_{k=1}^{N_s} \mathscr{L}\left(\{\sigma_k\}\right)$$

Sample mean over N_s shots

How to get them?



Measure a real quantum state

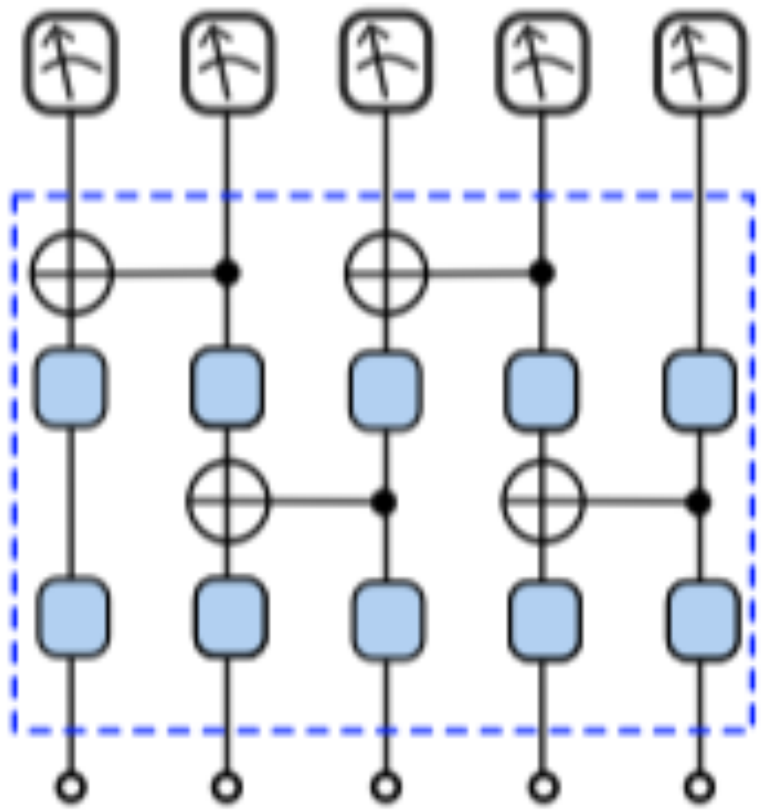
Sample a Matrix Product State

Methods

Sample a Matrix Product State

$\chi \sim e^P$ **Simulation of quantum circuit**

$\chi \ll e^P$ **Classical quantum-inspired algorithm**



$P =$ **number of layers**

$\vec{\theta}$ $=$ **variational parameters**

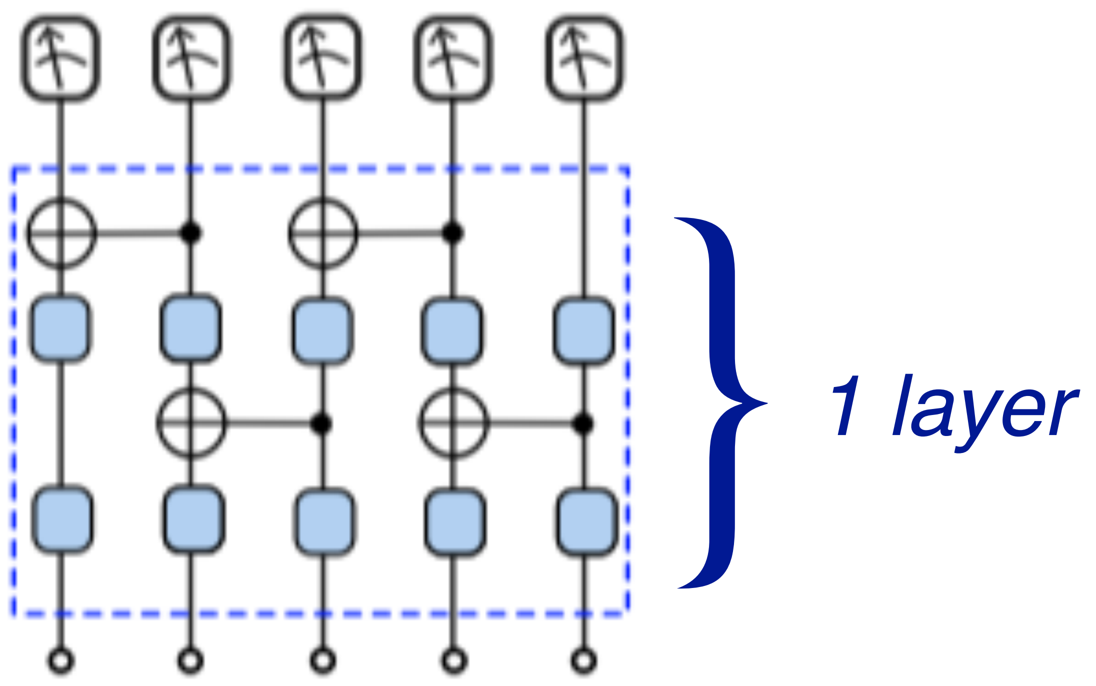
- Unbiased sampling of the MPS wave function with a computation cost of $\mathcal{O}(N_s \chi^2 N)$: **linear** in N

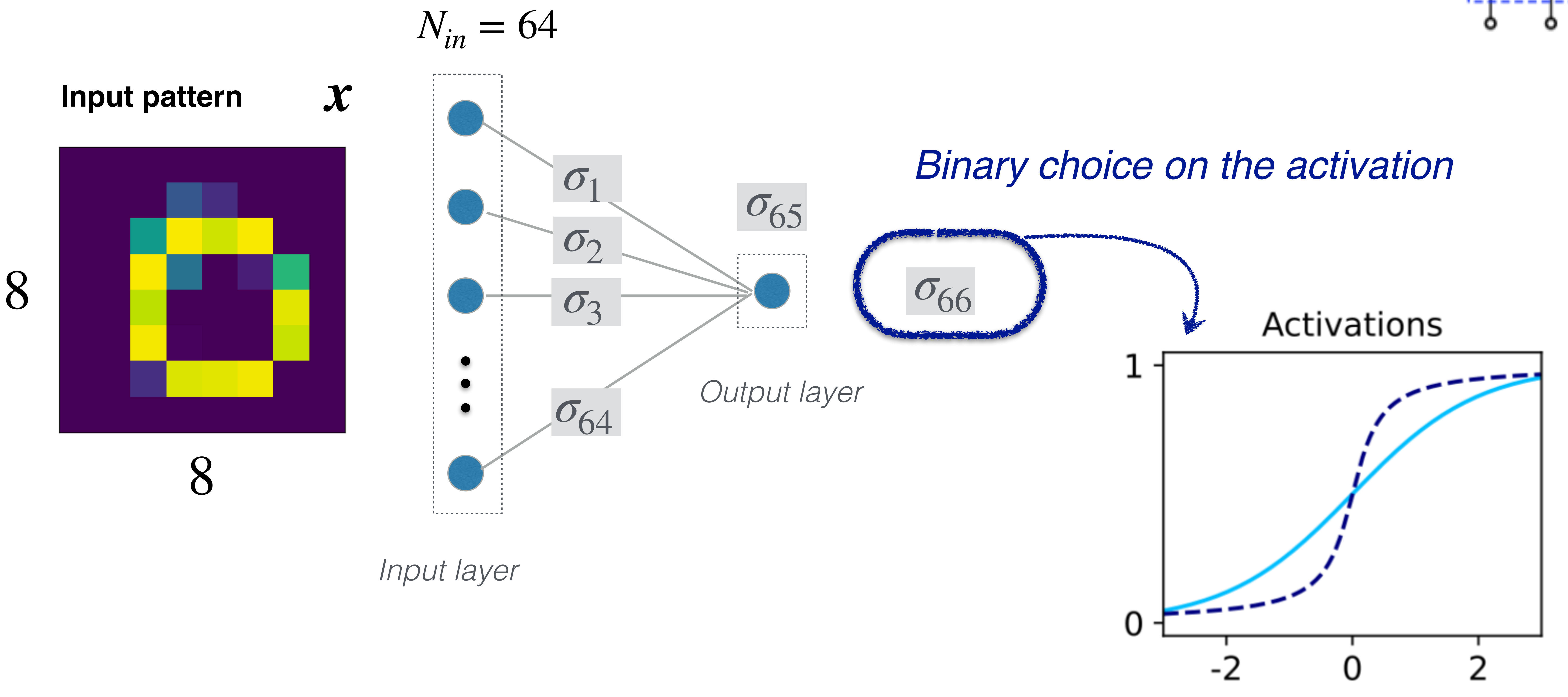
- Optimization routine for $\vec{\theta}$: Quantum Natural - Simultaneous Perturbation Stochastic Approximation (**QN-SPSA**)

J Gacon et al., Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information, Quantum, 5:567, (2021)

Preliminary results

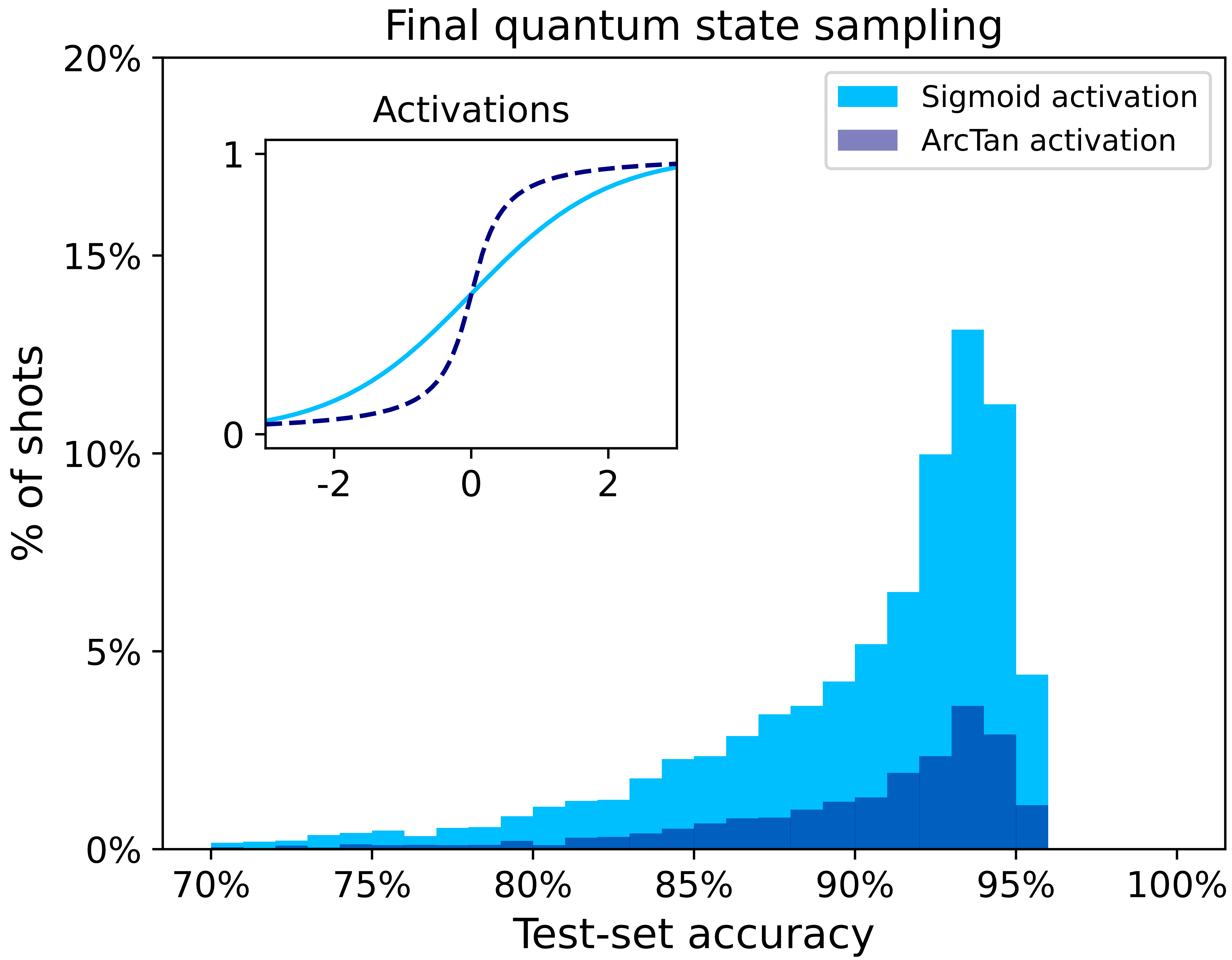
- Scaled-down version of the MNIST: **0 v.s. 1** binary classification
- One-layer BiNN with an hyperparameter for the activation function: **66 qubits**
- Trained with binary cross entropy loss
- $P = 2$ layers of Hardware-Efficient-Ansatz, regime $\chi \sim e^P$ (accurate simulation)





Preliminary results

Perform repeated measurements (shots) on the final state $|\psi(\vec{\theta}_{opt})\rangle$



- $|\psi(\vec{\theta}_{opt})\rangle$ learns batch of BiNN solutions, with both choices of the activation (~Bayesian approaches)
- **Mutual Hamming distance** between optimal BNN: e.g. ≈ 22 for configurations with test accuracy $> 90\,\%$

Outlook

- **Full MNIST**: (100s of qubits) *better Ansatz* is required
  - QN-SPSA seems numerically unstable in this regime
  - Preliminary feature extraction e.g. with **PCA**
- Investigate the role of the **bond dim.** χ as a potential **regularization parameter**
- Generalize to **low-precision NNs**: float with less than 32-bits but more than 1

Thanks for surviving until Friday afternoon :)

Binary Neural Networks (BiNNs): **how**?

State of the art: shortcomings

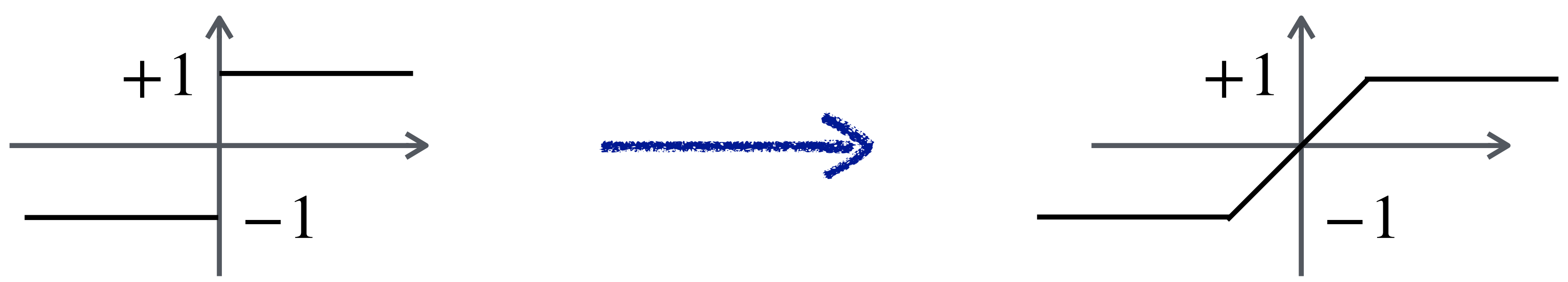
- Existing algorithms often require **full-precision network parameters** in the **training phase**

  E.g.: **straight-through estimator (STE)**

M. Courbariaux et al., BinaryNet: Training Deep Neural Networks with Weights and Activations constrained to +1 or -1
arXiv:1602.02830 (2016)

Forward pass: **binary** weights

Backward propagation: **float** weights



- Two loops:

1. **Outer Loop** (architecture, hyper parameters tuning with a **validation set**)
2. **Inner Loop** (weights and biases training with a **training set**)

Our proposal

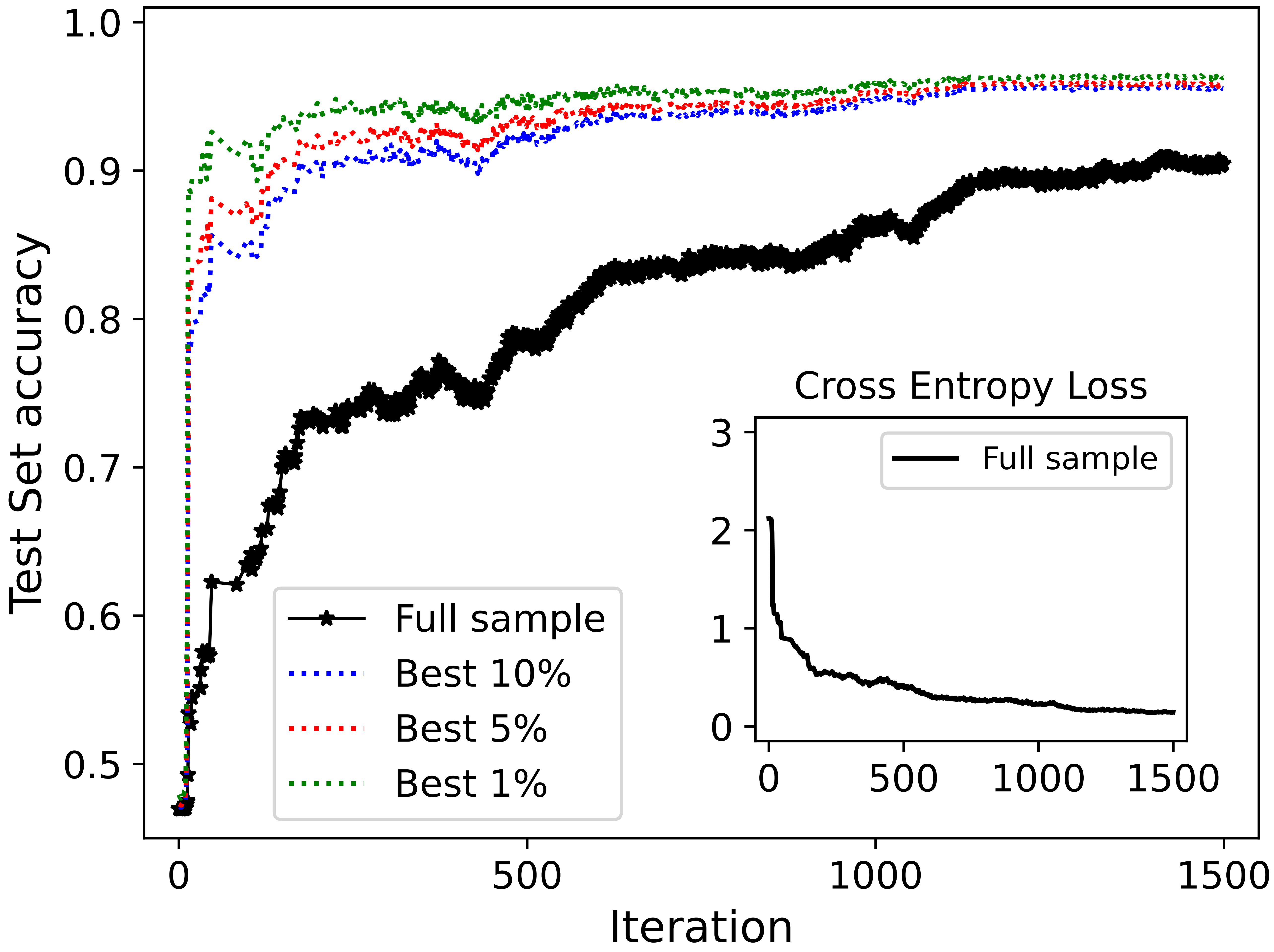
Use **Quantum hypernetworks** to **train BiNNs**

- ✓ **Only binary weights during training**
- ✓ **Unify the search over parameters, hyper-parameters, and architectures in a single loop**

Preliminary results

$$E_{var}(\vec{\theta}) = \langle \psi(\vec{\theta}) \,|\, \mathcal{L}\left(\{\hat{\sigma}_\alpha^z\}\right) |\, \psi(\vec{\theta}) \rangle \approx \frac{1}{N_s} \sum_{n=1}^{N_s} \mathcal{L}\left(\{\sigma_{n_\alpha}\}\right)$$

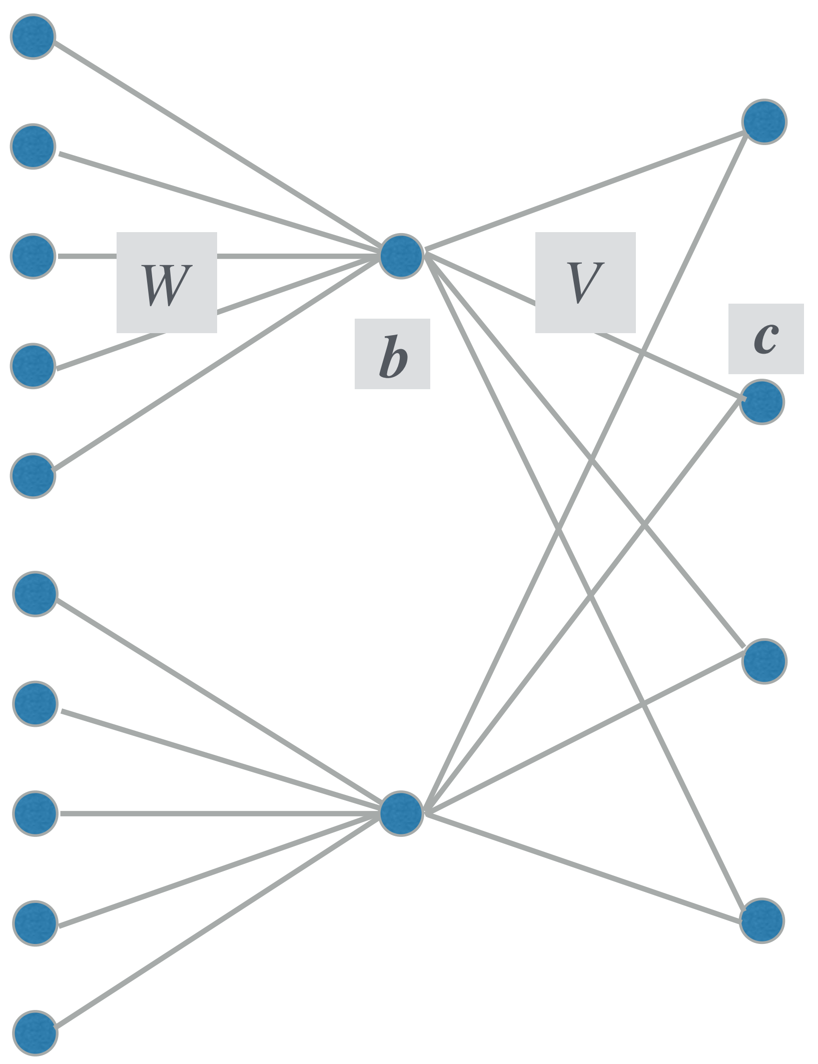
Sample mean over N_s samples (or shots)



Supervised Learning: standard formulation

Input pattern

$\boldsymbol{x}$



Classification: predicted label y

$$y = \mathsf{NN}\big(\boldsymbol{x}, \{\sigma_\alpha\}\big)$$

with $\{\sigma_\alpha\}_{\alpha=1}^{N} \equiv \{W, \boldsymbol{b}, V, \boldsymbol{c}\}$

N : total number of binary parameters

Training and Test Set $\{\boldsymbol{x}^{(\mu)}, \bar{y}^{(\mu)}\}_{\mu=1}^{N_{train}}$ $\{\boldsymbol{x}^{(\mu)}, \bar{y}^{(\mu)}\}_{\mu=1}^{N_{test}}$

$\boldsymbol{x}^{(\mu)}$ data point (*pattern*) to classify

$\bar{y}^{(\mu)}$ prescribed *label*

Classical **Loss Function to minimize during training**

$$\mathscr{L}\Big(\{\sigma_\alpha\}\Big) = \sum_{\mu=1}^{N_{train}} l\left(\mathsf{NN}\big(\boldsymbol{x}^{(\mu)}, \{\sigma_\alpha\}\big), \bar{y}^{(\mu)}\right)$$

N-bit real Boolean function $\mathscr{L} : \{0,1\}^N \to \mathbb{R}$

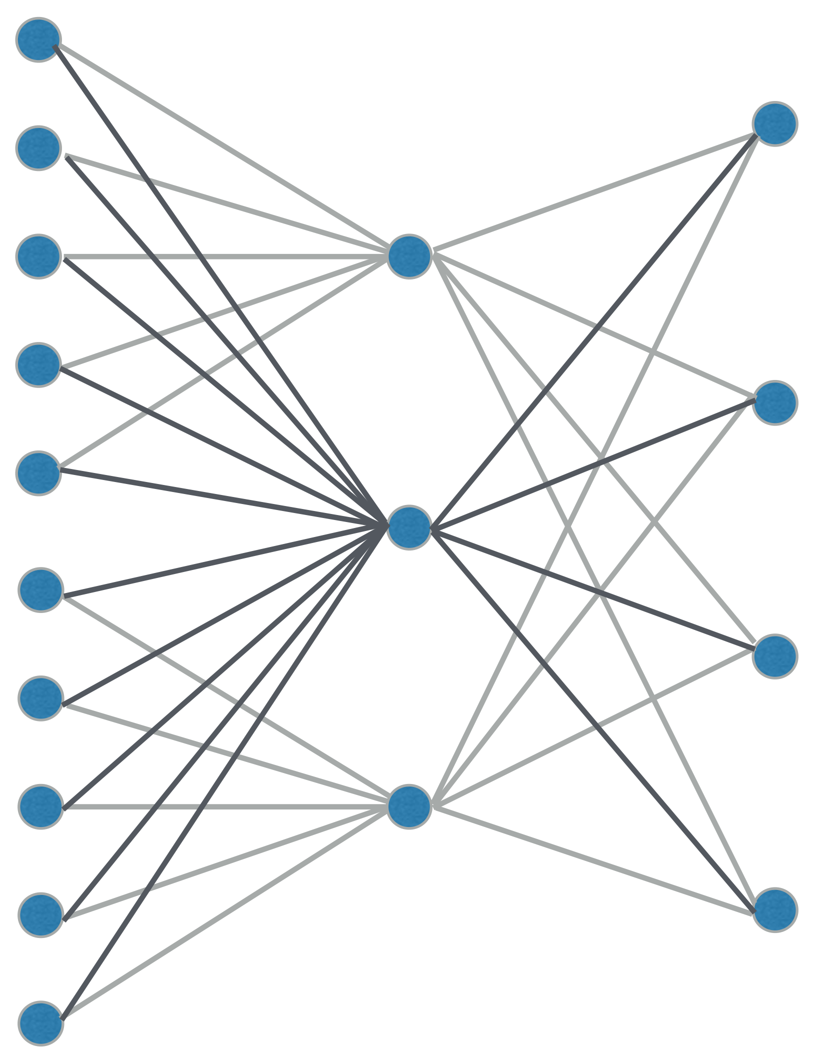
Quantum Hypernetwork

- We can exploit **quantum superposition** even further: train **different BiNN architectures** at the same time

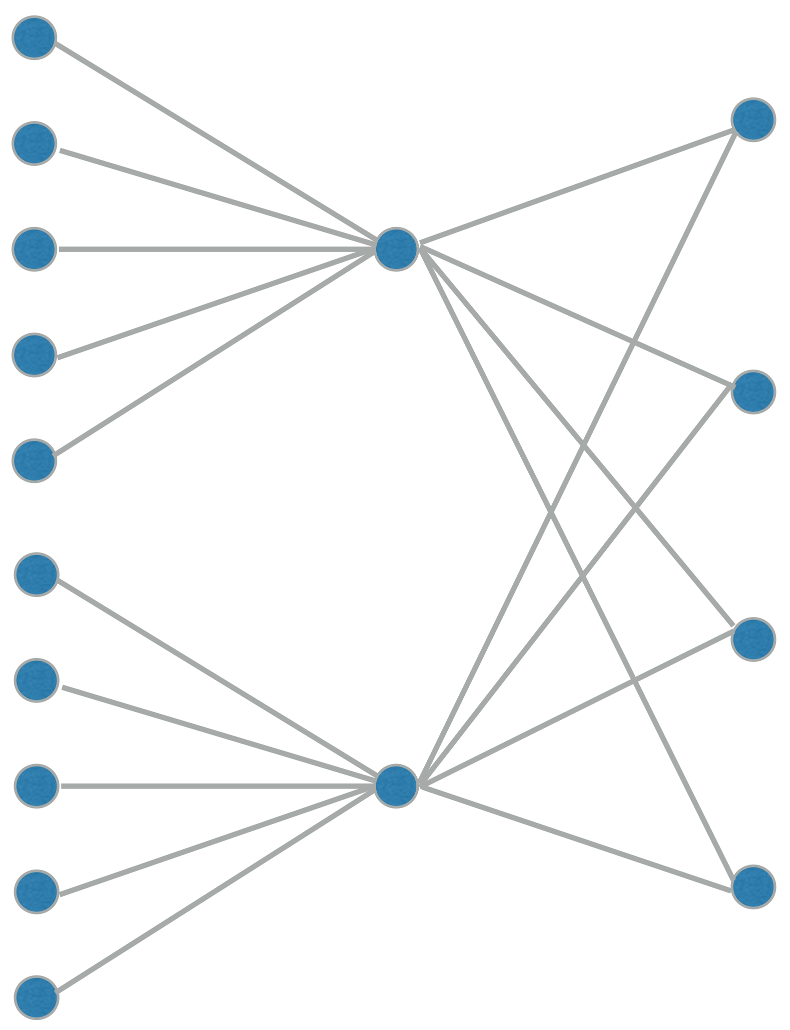
Binary architectural choice OR binary hyper-parameter selection = additional qubit σ^*

$$|\psi\rangle = \sum_{i=1}^{N} \psi(\sigma_1 \ldots \sigma_N, \sigma^*)\,|\,\sigma_1 \ldots \sigma_N\,,\sigma^*\rangle$$

Example of
architectural choice:
remove some neurons



$\sigma^* = +1$



$\sigma^* = -1$

Binary Neural Networks (BiNNs): **why**?

- Deep NN models are **computationally expensive**: **memory, energy…**
- **Require a lot of GPUs:** difficult to deploy on **small devices (a smartphone)**

BiNNs

Cons

- **Training** is **challenging**
  1. **Standard backpropagation:** it cannot be applied (non-differentiable activations!)
  2. **Binarization after training on float weights:** it does not work
- **Other challenges: architectural design, hyper-parameter tuning: hard combinatorial optimization tasks**
- **Worse performance?**

… or maybe not

Y. Zhang et al., Binarized Neural Machine Translation arXiv:2302.04907 (2023)

"one-bit weight-only Transformer can achieve the same quality as a float one […]"