Variational framework for partially-measured physical system control

Babak Rahmani*
EPFL
babak.rahmani@epfl.ch

Demetri Psaltis
EPFL
demetri.psaltis@epfl.ch

Christophe Moser
EPFL
christophe.moser@epfl.ch

Abstract

To characterize a physical system to behave as desired, either its underlying governing rules must be known a priori or the system itself be accurately measured. The complexity of full measurements of the system scales with its size. When exposed to real-world conditions, such as perturbations or time-varying settings, the system calibrated for a fixed working condition might require non-trivial re-calibration, a process that could be prohibitively expensive, inefficient and impractical for real-world use cases. In this work, we propose a learning procedure to obtain a desired target output from a physical system. We use Variational Auto-Encoders (VAE) to provide a generative model of the system function and use this model to obtain the required input of the system that produces the target output. We showcase the applicability of our method for two datasets in optical physics and neuroscience.

1 Introduction

In physical system characterization, a fundamental challenge is finding the proper continuous space input to a system that yields a desired functional output. For example, an open question in sensory/motor neuroscience is how to determine the input stimulation able to induce a desired behavior. So is controlling the output of an optical system, such as a turbid medium used for imaging, that could be non-linear and time-varying. In a linear physical system, the problem of finding the input that produces a desired output can be determined by monitoring its response to a series of arbitrary inputs and then computing the inverse of the system's transmission matrix (a mapping from inputs to outputs). This entails measuring the responses of the system fully. In practice, physical systems can only be partially measured and, more importantly, are nonlinear. So the transmission matrix formalism cannot be used. Even though the forward path of the system could be fully characterized, obtaining its inverse for large scale systems involving millions of variables is computationally intensive if not entirely intractable. Hence, resorting to data-driven methods that do not require full-measurements or linear approximation of the system, such as deep learning approaches, is inevitable. Deep learning techniques proposed for these tasks McIntosh et al. [2016], Rahmani et al. [2018] mostly take advantage of labeled data to do supervised training. For applications that require control over the response of one or an ensemble of targets, end-to-end supervised learning can fail due to the lack of labeled data within the distribution of desired target responses as well as inherent sensitivity of supervised approaches to perturbations of out-of-training-distribution data. Therefore, we instead propose a learning framework based on generative probabilistic models Mirza and Osindero [2014] and in specific, VAEs Kingma and Welling [2013] which involves in construction of a forward estimator of the possibly partially measured system. Once the forward model is obtained, a second estimator is trained to provide the required input of the system for producing the desired output. This latter estimator could be constraint so as to promote certain solutions. Therefore, contributions of this work are as follows:

*Department of Electrical Engineering.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

- Using the variational generative models, we provide a training algorithm for learning the distribution of the system's inputs that are needed to obtain a desired output of the system.
- Using the sampling feature of the learned forward VAE model, we illustrate how our training algorithm learns to iteratively move towards the correct distribution of the inputs.

Related works: As opposed to the inference problem of estimating the input of the system from noisy sensory outputs in experimental disciplines such as microscopy Rivenson et al. [2017], optical tomography Würfl et al. [2016] and neuroscience Parthasarathy et al. [2017] that supervised deep learning approach is a fairly well-established technique, learning methods for control applications in these fields have yet to be matured. Closed-loop techniques based on deep networks have been proposed for a number of applications, such as for brain neuroscience Bashivan et al. [2019] wherein authors control the activity of individual neuronal sites in V4 area by optimizing single input stimuli. Likewise, for optical turbid-medium imaging, authors have used ML-based estimators for controlling the optical fields Rahmani et al. [2020]. As opposed to the previous works, we propose joint learning of the forward and backward models of the system with VAEs to implicitly impose compatibility of the sought-after solutions with the underlying physics of the problem. The latter, in essence, is akin to technique of untrained neural networks Van Veen et al. [2018], Ulyanov et al. [2018], Heckel and Hand [2018] in denoising and inpainting.

2 Generative modelling

Problem scenario In the most general form, we assume that a given input of a system, x_i, is mapped to its output via the function f as in $y_i = f(x_i)$. Therein, $x_i \in \mathbb{C}^n$ and $y_i \in \mathbb{C}^m$ in the most general case. All known about f is that it could be a (non)linear time-varying function. We assume that all the noise sources are incorporated in f. Additionally, f can be sampled as many times as needed. In other words, exact output of the system, i.e. y_i, is available for any given input x_i. Yet, f is never measured nor analytically derived. Moreover, f might only be partially measured for which, the input-output relationship is modified to $y_i = \varphi[f(x_i)]$ where φ is either identity (fully measured system) or some other functions (for example modulus $|.|^2$ function).

We seek to find the x_i^* that would produce a desired y_i^*. It is worth emphasizing that the experimentalist might only have access to the partially measured system while the objective is to obtain the desired output in the fully measured system. The problem, in its most general form, can be formulated as follows.

$$\mathcal{L} = \min_{\xi,\zeta} \mathbb{E}_{x_i,y_i,z}\Big[D[y_i, M_\zeta(x_i, z)]\Big] + \mathbb{E}_{x_i^*,y_i^*,z}\Big[\sigma[M_\zeta(A_\xi(x_i^*, y_i^*), z), y_i^*]\Big] \tag{1}$$

where $M_\zeta : \mathbb{C}^{n\times l} \to \mathbb{C}^m$, referred to henceforth as the Model, is a differentiable representation of f parameterized by ζ and $A_\xi : \mathbb{C}^{n\times m} \to \mathbb{C}^n$, referred to henceforth as Actor, is a mapping that produces the input for M_ζ. Therein, D is the distance between outputs y_i sampled (experimentally) from $y_i = f(x_i)$ and the output of M_ζ. σ is the distance between the desired target y_i^* and predicted output of M_ζ given the output of A_ξ. $z = \{z_i\}_{i=1}^l$ is the latent space vector of size l. The two-term loss function $\mathcal{L}$ is then optimized with respect to the parameters ζ and ξ. We denote the first RHS term in Eq. 1, as the Model's loss $\mathcal{L}_{M_\zeta}$ and the second RHS term as the Actor's loss $\mathcal{L}_{A_\xi}$.

Forward estimator learning The forward mapping M_ζ is estimated as a generative probabilistic VAE. The reason for this choice of model is two-fold. First, forward models that are fundamentally stochastic in nature (see example 2 in Results section) could be better represented by a probabilistic model rather than a ML estimator trained in a supervised learning manner. Additionally, even if f is deterministic, noise sources incorporated into f make it stochastic in practice. Second, the generative sampling feature of VAEs could be conveniently used to demonstrate how the correct control input x_i^* (that is required to generate y_i^*) could be obtained iteratively.

The VAE M_ζ consists of two networks, an encoder and a decoder. The former is trained to transform input x_i conditioned on the system's output y_i onto the latent vector z that is enforced to be close to a normal distribution $\mathcal{N}(0, \mathrm{I})$; effectively learning the conditional distribution $q_\Phi(z|y_i)$ parameterized by Φ. The decoder, on the other hand, takes the latent vector z- drawn from the encoder distribution $\mathcal{N}(\mu_{enc}, \sigma_{enc})$ using reparameterization trick- to generate output $\hat{y}_i$; effectively learning

the conditional distribution $p_\theta(y_i|z, x_i)$ parameterized by θ. The training of the VAE is carried out by optimizing the following loss function w.r.t. $\zeta : \{\theta, \Phi\}$ Higgins et al. [2016].

$$\mathcal{L}_{M_\zeta} = \min_{\zeta:\{\theta,\Phi\}} -\mathbb{E}_{x_i\sim\rho(x)}\Big[\mathbb{E}_{z\sim q(.|y_i)}\left[\log[p_\theta(y_i|z,x_i)]\right] - \beta\,\mathbb{E}_{y_i\sim\rho(.|x_i)}\left[D_{\mathrm{KL}}(q_\Phi(z|y_i)\,||\,\mathcal{N}(0,\mathrm{I}))\right]\Big] \tag{2}$$

where β is the weighting factor between the two terms in the loss function and ρ is to denote a general purpose probability distribution.

Training algorithm A sketch of the networks and gradient flows is depicted below. Algorithm 1 presents the learning procedure for the system control. It involves in computing the variational updates of the forward model followed by training of the backward model. In particular, first the forward model M_ζ is learned through optimizing $\mathcal{L}_{M_\zeta}$ in the general loss function in Eq. 1 for some number of steps K_1 (note the gradient flow in the sketch). Once done, the backward model A_ξ is learned via optimizing the loss function $\mathcal{L}_{A_\xi}$ for K_2 number of steps (gradients flow through the fixed Model to reach the Actor).

After $K_1 + K_2$ steps, we asses the performance by sampling some targets $(x_i^*,\, y_i^*)$ and using the learned Actor $A_\xi(x_i^*, y_i^*)$ to obtain system inputs $\hat{x}_i$. Finally, we compute an empirical performance metric between the outputs generated through the experimental system $\hat{y}_i$ by the control inputs provided by the algorithm and the targets and reiterate the entire process if the performance is not satisfactory.

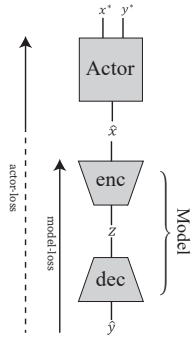
Algorithm 1

Input: Data tuples (x_i, y_i) sampled randomly from partially measured system $y_i = \varphi[f(x_i)]$, target outputs y_i^*, K_1 and K_2 (number of training steps for M_ζ and A_ξ, respectively), I (number of going back and forth between training the entire networks and experimenting the obtained solution in the true system)

Output: The control input x_i^* required for generating y_i^*

Initialization Variational parameters $\zeta : \{\theta, \Phi\}$ and ξ
for iter $\in \{1, 2, 3, ...I\}$ **do**
  for i $\in \{1, 2, 3, ...K_1\}$ **do**
    $\zeta \leftarrow \zeta - \alpha\nabla_\zeta\mathcal{L}_{M_\zeta}(x_i, y_i, z)$
  end for
  for i $\in \{1, 2, 3, ...K_2\}$ **do**
    $\xi \leftarrow \xi - \alpha\nabla_\xi\mathcal{L}_{A_\xi}(x_i^*, y_i^*)$
  end for
  Sample new (x_i, y_i) from $x_i \leftarrow \hat{x}_i = A_\xi(x_i^*, y_i^*)$, and $y_i \leftarrow \hat{y}_i = f(A_\xi(x_i^*, y_i^*))$
  Calculate empirical performance metric $\frac{1}{N}\sum_{i=1}^{N}\sigma(\hat{y}_i, y_i^*)$
  if System's desired performance is achieved **then**
    End training
  end if

end for



3 Results

Phase retrieval for optical system control The first example involves in characterization of a slowly time-varying, nonlinear physical system featuring random scramblers. The objective of this experiment is to find the appropriate complex input vector of the system, $\mathbf{X}^* = \{x_i^*\} \in \mathbb{C}^n$, that produces a target output, $\mathbf{Y}^* = \{y_\mu^*\} \in \mathbb{R}^m$ (sampled from a desired distribution ρ) given the partial measurements of the system as in $y_\mu^* = \left|\sum_i F_{\mu i}x_i^*\right|^2$, where x_i^* (and respectively y_μ^*) are elements of the input (output) vector and $F_{\mu i}$ is the complex-value measurement matrix. Although the problem in essence is a phase retrieval (PR) of the system's input, key differences with the conventional PR settings renders it more challenging. In particular, in the the original PR problem, F is entirely known

a priori. In the current setting, F is not measured and therefore is unknown. Instead, tuples of an arbitrary input $\mathbf{X}$ and its corresponding output $\mathbf{Y}$ is available. Secondly, while in the conventional PR, outputs $\mathbf{Y}$ (generated via a teacher model) always belong to the support of F, the target output $\mathbf{Y}^*$ may not belong to the support of F which requires finding the input that produces the closest output to the target in some metric. The optimization problem and network architecture is explained in detail in the Appendix A.

We tested our algorithm with target outputs $\mathbf{Y}^*$ sampled from MNIST dataset Cohen et al. [2017]. Fig. 1 (a) plots the empirical l_2 norm as well as 2D Pearson correlation between the system's outputs and targets versus the iteration number I in Algorithm 1. It can be seen that the algorithm almost reaches the 2D correlation (~ 0.9) obtained with the gold-standard full-measurement techniques such as Loterie et al. [2015]. We note that this is the upper bound of the problem because the experimentalist has access to full information phase and amplitude of the system's output (the same problem but without the modulus $|.|^2$). Examples of the experimentally generated outputs using the proposed algorithm are also provided in Appendix A.

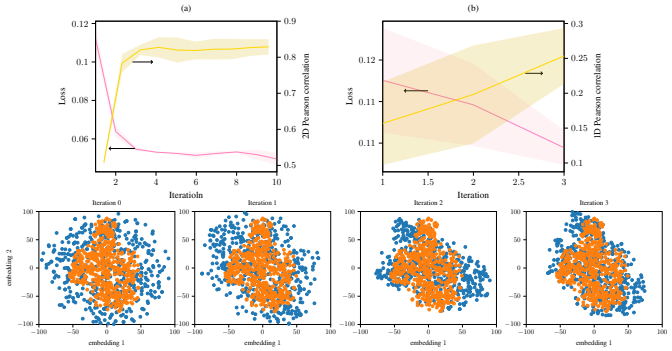


Figure 1: Performance metric of the algorithm (loss: left axis and Pearson correlation: right axis) versus iteration number for phase retrieval (a) and vision neuroscience task (b). The Latent vector evolution of the latter task as 2D embedding (blue). Orange dots denote the latent vector of the true system f.

Vision neuroscience In the second example, we apply the algorithm to a dataset comprising sequences of natural images $x^*_{ij} \in \mathbb{R}^{n\times t}$ and their corresponding count data $y^*_{ij} \in \mathbb{N}^{m\times t}$. These images and the count data are, respectively, the stimuli entering the retina in Salamander and the elicited time-series count responses of a number of Retina Ganglion Cells (RGCs). Approximating the system as a Poisson process, the system f is defined as the function that takes the image sequences as input and gives a time-varying posterior mean as output. Models based on Convolutional Neural Networks (CNNs) have been recently proposed for this modeling McIntosh et al. [2016]. Given f, we intend to find a transformed version of the input images that while are constraint to be of lower resolution, still elicit similar neuronal responses (in some metric) to those of the original input images (refer to appendix B). This constraint is imposed implicitly by architecture of the Actor network explained in more details in the appendix C.

Fig. 1 plots the performance metric evolution of this task. It can be seen that the algorithm almost reaches the maximum possible performance of the system (1D correlation ~ 0.3) within three iterations. The latent vector of the forward VAE Model of our algorithm is sampled at each iteration and projected to a 2-dimensional (2D) embedding using t-SNE. The true latent vector distribution required for obtaining the desired outputs is also shown. The network architecture and optimization scheme is further explained in the Appendices.

4 Discussion and conclusion

We proposed a framework based on VAEs for system control. We also demonstrated how VAEs could illustratively show iterative convergence of the posterior latent variables to those required for obtaining the target outputs. The relevance of the approach was showcased for two applications. The applicability of the method to problems that are chaotic or rapidly time-varying is interesting and

perhaps more challenging due to their difficulty of latent space learning. We note that black-box treatment of the physical system by the algorithm should be treated with caution and further studied in future work.

Acknowledgments and Disclosure of Funding

Authors thank Dr. Damien Loterie for his help with preparing some data used in the paper.

References

P. Bashivan, K. Kar, and J. J. DiCarlo. Neural population control via deep image synthesis. *Science*, 364(6439), 2019.

G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

R. Heckel and P. Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. *arXiv preprint arXiv:1810.03982*, 2018.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. Loterie, S. Farahi, I. Papadopoulos, A. Goy, D. Psaltis, and C. Moser. Digital confocal microscopy through a multimode fiber. *Optics express*, 23(18):23845–23858, 2015.

L. McIntosh, N. Maheswaranathan, A. Nayebi, S. Ganguli, and S. Baccus. Deep learning models of the retinal response to natural scenes. *Advances in neural information processing systems*, 29:1369–1377, 2016.

M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

N. Parthasarathy, E. Batty, W. Falcon, T. Rutten, M. Rajpal, E. Chichilnisky, and L. Paninski. Neural networks for efficient bayesian decoding of natural images from retinal neurons. *Advances in Neural Information Processing Systems*, 30:6434–6445, 2017.

B. Rahmani, D. Loterie, G. Konstantinou, D. Psaltis, and C. Moser. Multimode optical fiber transmission with a deep learning network. *Light: Science & Applications*, 7(1):1–11, 2018.

B. Rahmani, D. Loterie, E. Kakkava, N. Borhani, U. Teğin, D. Psaltis, and C. Moser. Actor neural networks for the robust control of partially measured nonlinear systems showcased for image propagation through diffuse media. *Nature Machine Intelligence*, 2(7):403–410, 2020.

Y. Rivenson, Z. Göröcs, H. Günaydin, Y. Zhang, H. Wang, and A. Ozcan. Deep learning microscopy. *Optica*, 4 (11):1437–1443, 2017.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. arxiv 2015. *arXiv preprint arXiv:1505.04597*, 2019.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

D. Van Veen, A. Jalal, M. Soltanolkotabi, E. Price, S. Vishwanath, and A. G. Dimakis. Compressed sensing with deep image prior and learned regularization. *arXiv preprint arXiv:1806.06438*, 2018.

T. Würfl, F. C. Ghesu, V. Christlein, and A. Maier. Deep learning computed tomography. In *International conference on medical image computing and computer-assisted intervention*, pages 432–440. Springer, 2016.

5 Checklist

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **Response**: Yes, please refer to Results section.

(b) Have you read the ethics review guidelines and ensured that your paper conforms to them? **Response**: Yes.

(c) Did you discuss any potential negative societal impacts of your work? **Response**: Yes, please refer to the discussion and conclusion section.

(d) Did you describe the limitations of your work? **Response**: Yes, please refer to the discussion and conclusion section.

2. If you are including theoretical results... (a) Did you state the full set of assumptions of all theoretical results? **Response**: n/a

(b) Did you include complete proofs of all theoretical results? **Response**: n/a

3. If you ran experiments... (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **Response**: Yes, please refer to the supplementary section.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **Response**: Yes, please refer to the supplementary section.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **Response**: Yes, please refer to Fig. 1.

(d) Did you include the amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **Response**: Yes, please refer to the supplementary section.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets... (a) If your work uses existing assets, did you cite the creators? **Response**: Yes, please refer to the results section.

(b) Did you mention the license of the assets? **Response**: No. License is not required for usage or is data is publicly available.

(c) Did you include any new assets either in the supplemental material or as a URL? **Response**: Yes, assets are provided in the submission material and will be made public ally available upon publication.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **Response**: n/a

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **Response**: n/a

5. If you used crowdsourcing or conducted research with human subjects... (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **Response**: n/a

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **Response**: n/a

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **Response**: n/a

Appendices

A Phase retrieval for optical system control problem setting

Physical system The system in the optical control task is the experimental setup that consists of an input modulator (spatial light modulator), turbid medium (a 50 μm core size step-index multimode fiber of length 75 cm) and a receiver (a CMOS camera) working at light wavelength 532 nm. An example of a random input and its corresponding system's output is depicted in Fig. 2.

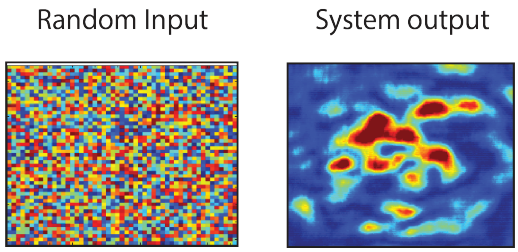


Figure 2: An example of a random input to the system and its corresponding output.

Optimization setting The task of phase retrieval for optical system control is schematically depicted in 3. The optimization problem can then be written as:

$$\mathcal{L} = \min_{\xi,\zeta} \mathbb{E}_{\mathbf{X},\mathbf{Y},z} \left\| \mathbf{Y} - M_\zeta(\mathbf{X}, z) \right\|_{l_2}^2 + \mathbb{E}_{\mathbf{X}^*,\mathbf{Y}^*,z} \left[\| \mathbf{Y}^* - M_\zeta(A_\xi(\mathbf{Y}^*), z) \|_{l_2}^2 \right] \tag{3}$$

where we choose l_2 norm for the forward and backward metrics.

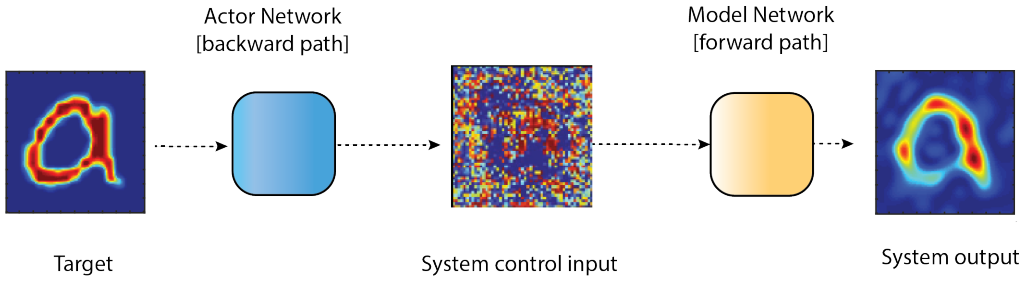


Figure 3: Input control pattern inference in the phase retrieval for optical system control task.

System's output Some examples of system's output obtained with the input found with the proposed algorithm is depicted in Fig. 4.

B Vision neuroscience problem setting

Physical system Instead of the experimental system, we use a CNN-based network trained with the *entire* dataset of the input image stimuli and their corresponding neuronal responses as the proxy for the true system f. Therefore, to be fair, only a third of the same dataset, randomly selected, is made available to our training algorithm. The architecture of f is identical to that of the forward model. An example of the input-output of this system is depicted in Fig. 5.

Optimization setting The dimensionality reduction process is schematically depicted in Fig. 6. The backward model in the task, which has a U-net architecture Ronneberger et al. [2019], is constraint to find solutions that are of lower resolutions than the original high resolution stimuli. This is achieved by adjusting the bottleneck size in the network architecture (denoted in Table 3). Lower sizes for the bottleneck provide lower resolution solutions. The loss function for this optimization problem reads as follows:

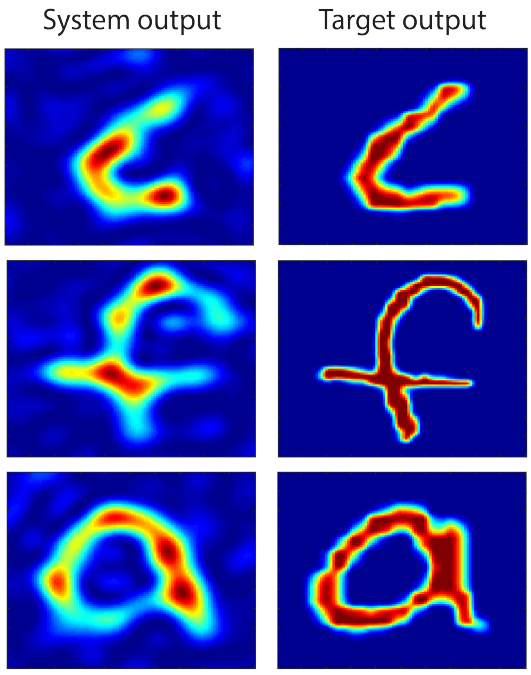


Figure 4: Examples of the experimental system's outputs obtained with the input solutions found by the algorithm.

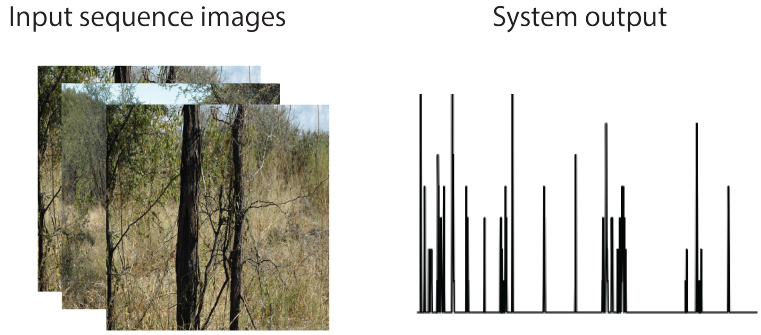


Figure 5: An example of the input image sequence and its corresponding output spike train of the vision system.

$$\mathcal{L} = \min_{\xi,\zeta} \mathbb{E}_{x^*_{ij},y^*_{ij},z}\Big[\log \mathrm{Po}(y^*_{ij}|M_\zeta(x^*_{ij},z))\Big] + \mathbb{E}_{x^*_{ij},y^*_{ij},z}\Big[\sigma[M_\zeta(A_\xi(x^*_{ij}),z),y^*_{ij}]\Big] \tag{4}$$

where we choose Poisson loss both for the forward and backward mappings.

C Network architecture and optimization

The hyperparameters of the forward and backward networks, optimizers as well as training epochs used for training is summarized in Table 1. Architecture of the networks is presented in Table 2 and 3. Hyperparameters were chosen such that a balance between the two terms of losses in Eq. 1 is achieved.

D Code repository

All models, implemented in Tensorflow v. 2.1. on Nvidia GPU 2080 Ti, are available on Github: `https://github.com/Babak70/Variational-framework-for-partially-measured-physical-system-control`.

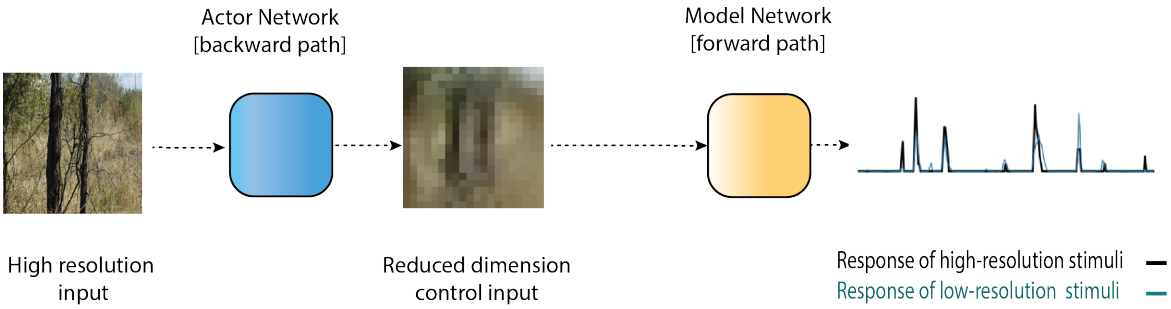


Figure 6: Dimensionality reduction of the input stimuli in the vision neuroscience task.

	Task 1	Task 2
Optimizer	Adam	Adam
Learning rate	10^{-4}	10^{-4}
VAE's β	500/450	10
Latent space dim.	100	15
Train/val/test batch size	20/-/10^3	10^3/10^3/10^3
Train/val/test batch num.	10^3/-/1	287/72/5

Table 1: Training details

Actor	Encoder	Decoder
Input 100×100 imgs F.C. output 51×51 Sigmoid	Input 51×51 imgs F.C. output $2\times$ latent dim. no activ.	Input latent dim. vector F.C. output 51×51 Sigmoid F.C. output 100×100 Sigmoid

Table 2: Task 1 network architecture

Actor	Encoder	Decoder
Input $50 \times 50 \times 1000$ seq. of imgs 3×3 conv. 64 s. 1 same Relu 2×2 maxpool 3×3 conv. 32 s. 1 same Relu 2×2 maxpool 3×3 conv. 16 s. 1 same Relu F.C. output Bottleneck(1/4/9) No activ. F.C. output $16 \times 12 \times 12$ 3×3 conv. 32 s. 1 same Relu 2×2 Upsampling 4 sided zero pad. 3×3 conv. 64 s. 1 same Relu 2×2 Upsampling 3×3 conv. 1 s. 1 same Sigmoid	Input $50 \times 50 \times 1000$ seq. of imgs 3×3 conv. 64 s. 1 same Relu 2×2 maxpool 3×3 conv. 32 s. 1 same Relu 2×2 maxpool 3×3 conv. 16 s. 1 same Relu F.C. output $2\times$ Latent dim. No activ.	F.C. output $16 \times 12 \times 12$ Relu 3×3 conv. 32 s. 1 same Relu 2×2 Upsampling 4 sided zero pad. 3×3 conv. 64 s. 1 same Relu 2×2 Upsampling 3×3 conv. 1 s. 1 same Sigmoid 21×21 conv. 4 s. 1 no pad. no activ. 40×1 1D-conv. 4 s. 1 same Relu 15×15 conv. 4 s. 1 no pad. Relu F.C. output 9 Exponential activ.

Table 3: Task 2 network architecture