Supplementary Information: A Framework for Demonstrating Practical Quantum Advantage: Comparing Quantum against Classical Generative Models

Mohamed Hibat-Allah,[1,2,3,4] Marta Mauri,[1] Juan Carrasquilla,[5,2,6] and Alejandro Perdomo-Ortiz[1,*]

[1]*Zapata AI Canada Inc., 25 Adelaide St East, M5C 3A1, Toronto, ON, Canada*
[2]*Vector Institute, MaRS Centre, Toronto, Ontario, M5G 1M1, Canada*
[3]*Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[4]*Perimeter Institute for Theoretical Physics, 31 Caroline St N, Waterloo, ON N2L 2Y5, Canada*
[5]*Institute for Theoretical Physics, ETH Zürich, 8093, Switzerland*
[6]*Department of Physics and Astronomy, University of Waterloo, Ontario, N2L 3G1, Canada*

(Dated: February 1, 2024)

I. SUPPLEMENTARY NOTE 1: GENERATIVE MODELS

In this supplementary note, we briefly introduce the generative models used in this study. We also summarize their hyperparameters in Supplementary Tab. I.

A. Quantum Circuit Born Machines (QCBMs)

Quantum circuit Born machines (QCBMs) are a class of expressive quantum generative models based on parametrized quantum circuits (PQCs) [1]. Starting from a fixed initial qubit configuration $|\mathbf{0}\rangle^{\otimes N}$, we apply a unitary $U(\boldsymbol{\theta})$ on top of this state. Projective measurements are performed at the end of the circuit to get configurations that are sampled, according to Born's rule, from the squared modulus of the PQC's wave function. One can use different topologies that describe the qubit connectivity to model the unitary $U(\boldsymbol{\theta})$. Our study uses the line topology, where each qubit is connected to its nearest neighbors in a 1D chain configuration [1, 2]. More details can be found in Ref. [3].

To train our QCBM, we compute the Kullback-Leibler (KL) divergence between the softmax training distribution P_{train} (see Eq. (1) in the main manuscript) and the QCBM probability distribution P_{QCBM} as

$$\text{KL}(P_{\text{train}}||P_{\text{QCBM}}) = \sum_{\mathbf{x}} P_{\text{train}}(\mathbf{x}) \log\left(\frac{P_{\text{train}}(\mathbf{x})}{\max(\delta, P_{\text{QCBM}}(\mathbf{x}))}\right),$$

where $\delta = 10^{-8}$ is a regularization factor to avoid numerical instabilities. The optimization of the parameters $\boldsymbol{\theta}$ is performed using a gradient-free method called CMA-ES optimizer [4, 5] after randomly initializing the parameters of the PQC. In our simulations, we used 8 layers, which provided the best utility U compared to $2, 4$ and 6 layers with line topology. For the QCBM and the following classical generative models, the optimal hyperparameters are obtained with a grid search with the condition of getting the lowest utility after 100 training iterations.

We have also performed simulations with the all-to-all topology [1]; however, they lead to sub-optimal performances compared to the line topology for trainability reasons at the scale of our experiments. In Ref. [3], we did not observe this issue for an all-to-all topology at the scale of 12 qubits. To further improve the trainability of QCBMs, there is a potential for using pre-training of QCBMs with tensor networks as suggested in Ref. [6]. As a final note, since the exact computation of P_{QCBM} is not tractable for large system sizes, one could consider the use of sample-based cost functions such as Maximum Mean Discrepancy (MMD) for a large number of qubits [7]. Extending this study to an experimental demonstration on currently available quantum devices is also crucial to evaluate the impact of noise on the model's performance.

B. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are unique architectures traditionally used for applications in natural language processing such as machine translation and speech recognition [8, 9]. They are known for their ability to simulate Turing machines [10] and for their capability of being universal approximators [11]. RNNs take advantage of the probability chain rule to generate uncorrelated samples autoregressively as follows:

$$P_{\text{RNN}}(\sigma_1, \sigma_2, \dots, \sigma_N) = P_{\text{RNN}}(\sigma_1) P_{\text{RNN}}(\sigma_2|\sigma_1) \dots P_{\text{RNN}}(\sigma_N|\sigma_1, \dots \sigma_{N-1}). \quad (1)$$

Here $(\sigma_1, \sigma_2, \dots, \sigma_N)$ is a configuration of N bits where $\sigma_i = 0, 1$. Each conditional $P_{\text{RNN}}(\sigma_i|\sigma_1, \dots \sigma_{i-1})$ is computed using a Softmax layer as:

$$P_{\text{RNN}}(\sigma_i|\sigma_{<i}) = \text{Softmax}\left(U\mathbf{h}_i + \mathbf{c}\right) \cdot \boldsymbol{\sigma}_i. \quad (2)$$

$\boldsymbol{\sigma}_i$ is a one-hot encoding of σ_i and '$\cdot$' is the dot product operation. The weights U and the biases $\mathbf{c}$ are the parameters of this Softmax layer. The vector $\mathbf{h}_i$ is called the memory state with a tunable size d_{h}, which we call the hidden dimension. For the simplest (vanilla) RNN, $\mathbf{h}_i$ is computed using the following recursion relation:

$$\mathbf{h}_i = f(W\mathbf{h}_{i-1} + V\boldsymbol{\sigma}_{i-1} + \mathbf{b}), \quad (3)$$

where W, V and $\mathbf{b}$ are trainable parameters and f is a nonlinear activation function. Furthermore, $\boldsymbol{\sigma}_0, \mathbf{h}_0$ are initialized to zero. Typically, vanilla RNNs suffer from the vanishing

* aperdomo@post.harvard.edu

gradient problem, which makes their training a challenging task [12]. To mitigate this limitation, more advanced versions of RNN cells have been devised, namely, the Gated Recurrent Unit (GRU) [8, 13]. In our experiments, we use the PyTorch implementation of the GRU unit cell [14].

Similarly to the QCBM, we minimize the KL divergence between the RNN distribution (1) and the training distribution (see Eq. (1) in the main manuscript) without a regularization factor δ. For the optimization, we use Adam optimizer [15]. To find the best hyperparameters that provided the lowest utility, we have conducted a grid search with the learning rates $\eta = 10^{-4}, 10^{-3}, 10^{-2}$ and the hidden dimensions $d_{\text{h}} = 8, 16, 32, 64, 128$. We find that $\eta = 10^{-3}$ and $d_{\text{h}} = 32$ are optimal for both the two values of the training data portion ϵ.

C. Transformers (TFs)

Transformers (TFs) are attention models that have sparked exciting advances in natural language processing [16], namely in applications related to language translation, text summarization, and language modeling. Similarly to the RNN, TFs can have the autoregressive property, and they have been proposed in the literature as a solution to the vanishing gradient problem in RNNs [12]. The attention mechanism in TFs allows handling long-range correlations compared to traditional RNNs [16, 17]. TFs can also process long data sequences compared to traditional neural network architectures.

In our simulations, we use the traditional implementation of TF decoders in Ref. [16] without using a TF encoder in a similar fashion to the RNN. First, we embed our inputs using a multilayer perceptron (MLP) with a Leaky ReLU activation, and then we add positional encoding. Next, we use a one-layered TF with one attention head. Additionally, the outputs are passed to a two-layer feed-forward neural network with a ReLU activation in the hidden layer. To reduce the search space, the size of the hidden dimension in the feed-forward neural network is chosen to be the same as the size of the embedding dimension, which we also denote as d_{h} [16]. The latter is fine-tuned in the range of possibilities $d_{\text{h}} = 8, 16, 32, 64, 128$ along with a range of learning rates $\eta = 10^{-4}, 10^{-3}, 10^{-2}$. For the training, we minimize the KL divergence between the TF probability distribution and the re-weighted training distribution (see Eq. (1) in the main manuscript). We find that $d_{\text{h}} = 64$ and $\eta = 10^{-2}$ provided the lowest utilities for both values of the training data portion ϵ.

D. Variational Autoencoders (VAEs)

Variational autoencoders (VAEs) are approximate likelihood generative models that can learn to represent and reconstruct data [18]. Initially, a VAE encoder maps an input data point to a latent representation $\mathbf{z}$. Next, a VAE decoder uses the latent representation to reconstruct the original input data point. Finally, the VAE compares the reconstruction and the initial input data point and is trained until the reconstruction error is as small as possible. In this case, the VAE cost function corresponds to the evidence lower bound (ELBO) of the negative log-likelihood [19].

In our study, the encoder and the decoder are built as an MLP with two hidden layers with a CELU ($\alpha = 2$) activation [20]. The encoder is followed by two separate MLP layers for the purpose of implementing the reparametrization trick [18, 21]. To reduce the search space of hyperparameters, the size of these hidden layers is taken to be the same as the size of the latent representation. The latter is chosen from the range $8, 16, 32, 64, 128$. For the optimization, we explore different learning rates $\eta = 10^{-4}, 10^{-3}, 10^{-2}$. We also note that the temperature β was not added to the training of VAEs [22]. After conducting a grid search, we find that a latent dimension of 128 and a learning rate $\eta = 10^{-3}$ are the optimal hyperparameters.

E. Generative Adversarial Networks (GANs)

Generative adversarial networks (GANs) are a class of generative models that consists of a generator and a discriminator [18, 23]. The generator is optimized based on a dataset, while the discriminator is trained to distinguish between real and generated data points. The two networks are trained together until reaching Nash equilibrium, where the generator produces data that is very similar to the original dataset while the discriminator becomes much better at distinguishing real from generated data.

In our study, we use Wasserstein GANs (WGAN) [24], which are a variant of GANs with a loss function called the Wasserstein loss (which is also known as the Earth Mover's distance). For the optimization of the loss function, we use Adam optimizer. An interesting feature about WGANs is that they are less susceptible to mode collapse. They can also be used for a wide range of applications, including audio synthesis, text generation, and image generation, and a wide range of areas of science [25].

In our numerical implementation, we choose the generator and discriminator as two feed-forward neural networks with two hidden layers with CELU ($\alpha = 2$) activation [20]. For simplicity, we choose the size of the gaussian prior to being the same as the width of the hidden layers. We fine-tune the hyperparameters with a grid search on the widths $8, 16, 32, 64, 128$ and the learning rates $\eta = 10^{-4}, 10^{-3}, 10^{-2}$. We find that a prior size of 8 and $\eta = 10^{-2}$ are optimal for the data portion $\epsilon = 0.01$, whereas a prior size of 128 with $\eta = 10^{-3}$ works best for $\epsilon = 0.001$.

II. SUPPLEMENTARY NOTE 2: ADDITIONAL QUALITY-BASED GENERALIZATION RESULTS

In this appendix, we report the best values found by our generative models throughout the training shown in Fig. 3 of the main manuscript. The error bars are computed by averaging over the outcome of 10 different training seeds. For

Generative model	Hyperparameter	Value
QCBM	Number of layers	8
	Circuit topology	Line topology
	Optimizer	CMA-ES optimizer with $\sigma_0 = 0.1$
	Initialization	Random initialization between $-\pi/2$ and $\pi/2$
	Total number of parameters	256
RNN	Architecture	GRU
	Number of layers	1
	Optimizer	Adam optimizer
	Learning rate	10^{-3}
	Number of hidden units	32
	Total number of parameters	3456
TF	Number of layers	1
	Number of attention heads	1
	Embedding dimension size	64
	Size of FFNN output	64
	Optimizer	Adam optimizer
	Learning rate	10^{-3}
	Total number of parameters	25538
VAE	Prior size	128
	Encoder architecture	FFNN with CELU ($\alpha = 2$) activation
	Encoder hidden layers width	128
	Number of hidden layers of encoder net	2
	Decoder architecture	FFNN with CELU ($\alpha = 2$) activation
	Decoder hidden layers width	128
	Number of hidden layers of decoder net	2
	Temperature $1/\beta$	0.0
	Optimizer	Adam optimizer
	Learning rate	10^{-3}
	Total number of parameters	87828
WGAN ($\epsilon = 0.01$)	Prior size	8
	Generator architecture	FFNN with CELU ($\alpha = 2$) activation
	Generator hidden layers width	8
	Number of hidden layers of generator net	2
	Discriminator architecture	FFNN with CELU ($\alpha = 2$) activation
	Discriminator hidden layers width	8
	Number of hidden layers of discriminator net	2
	Optimizer	Adam optimizer
	Gradient regularization	10.0
	Learning rate	10^{-2}
	Total number of parameters	573
WGAN ($\epsilon = 0.001$)	Prior size	128
	Generator architecture	FFNN with CELU ($\alpha = 2$) activation
	Generator hidden layers width	128
	Number of hidden layers of generator net	2
	Discriminator architecture	FFNN with CELU ($\alpha = 2$) activation
	Discriminator hidden layers width	128
	Number of hidden layers of discriminator net	2
	Optimizer	Adam optimizer
	Gradient regularization	10.0
	Learning rate	10^{-3}
	Total number of parameters	54933

Supplementary Tab. I. Hyperparameters used to obtain the results reported in this study. FFNN stands for a feed-forward neural network. We also report the total number of parameters for each model, where the QCBM has the lowest number of variational parameters.

$\epsilon = 0.001$, we report the best performances for each metric during the training in Supplementary Tab. II. We also report the metrics for $\epsilon = 0.01$ in Supplementary Tab. III.

III. SUPPLEMENTARY NOTE 3: PRE-GENERALIZATION AND VALIDITY-BASED GENERALIZATION RESULTS

The focus of pre-generalization is to check whether a generative model can generate samples outside the training data. This can be done by generating a set of queries $\mathcal{G} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(Q)}\}$ with size Q to compute the ratio:

$$E = \frac{|G_{\text{new}}|}{Q},$$

where G_{new} is the set of unseen (i.e., out-of-training) samples among the set of generated queries $\mathcal{G}$. This metric is called exploration, and the larger its value, the more our model of interest can explore different configurations outside the training data.

Given a set of constraints, it is also desirable that a generative model generates queries that satisfy these constraints, which are typical in combinatorial optimization problems. In this case, these queries are called valid. We can define the first validity-based metric, called the generalization rate:

$$R = \frac{|G_{\text{sol}}|}{Q},$$

where G_{sol} is the set of valid and unseen queries. This metric quantifies the likelihood of our generated samples that are both unseen and valid, and it can be renormalized to 1 [3]. In this case, we can define the normalized rate as:

$$\tilde{R} = \frac{R}{1-\epsilon}.$$

Additionally, we can define the generalization fidelity as

$$F = \frac{|G_{\text{sol}}|}{|G_{\text{new}}|},$$

which quantifies the likelihood of a generative model to produce valid unseen samples out of the generated unseen samples, rather than invalid ones. This metric is not to be confused with the fidelity measure between two probability distributions or two quantum states.

In some applications [26], it is also important that a generative model provides a diverse set of unseen and valid samples. For this reason, we can define a third metric called the generalization coverage defined as:

$$C = \frac{|g_{\text{sol}}|}{|S| - T} = \frac{|g_{\text{sol}}|}{|S|(1-\epsilon)},$$

where g_{sol} is the set of unique queries that are both unseen and valid. This metric allows quantifying the diversity of solutions as opposed to the fidelity and the rate. The coverage can also be renormalized to 1 by defining a normalized coverage:

$$\tilde{C} = \frac{C}{1 - (1 - 1/(|S|(1-\epsilon))^Q} \approx \frac{|g_{\text{sol}}|}{Q},$$

where the denominator corresponds to the ideal expected value of the coverage [27] and the approximation holds in the regime $Q \ll |S|(1-\epsilon)$. The latter is typical when we have a relatively large solution space. We highlight that it is not necessary to know $|S|$ (or ϵ) in order to compute the coverage: since we are interested in using these metrics to compare models, one can simply utilize coverage ratios among models thus avoiding the need of having prior knowledge about the size of the solution space. Lastly, we specify that we only use the first track T1 to compute the validity-based metrics, since there is no cost involved in the calculation, which makes the second track T2 not applicable.

In this section, we also present the results from the pre-generalization and the validity-based generalization metrics to provide an additional perspective on the results in the main text. In Supplementary Fig. 1, we show the exploration E, normalized rate $\tilde{R}$, the fidelity F and the normalized coverage $\tilde{C}$ for a data portion $\epsilon = 0.01$ in panel (a) and for $\epsilon = 0.001$ in panel (b). A common feature of the results is that QCBMs are competitive with TFs and RNNs on the validity metrics and superior to VAEs and WGANs. Besides the quality-based generalization results in the main text, these observations favor the QCBMs' ability to provide the best compromise for validity-based and quality-based generalization compared to the other generative models. The observation of VAEs being less efficient at this generalization level could be explained by the mode collapse that occurs at zero temperature. Using a finite temperature $1/\beta$ in future studies could be helpful to mitigate this limitation [22]. We also observe that the WGAN has a similar behavior to the VAE, which can also be related to mode-collapse.

T1	C_q	MV	U
QCBM	**7e-4**	**-19**	**-17.30(8)**
RNN	**7e-4**	**-19**	-15.03(13)
TF	**6.9(1)e-4**	**-18.9(1)**	-16.07(42)
VAE	6.7(1)e-4	**-19**	-16.46(7)
WGAN	6.4(2)e-4	**-19**	-15.21(13)

T2	C_q	MV	U
QCBM	**0.04**	-16	**-14.60(5)**
RNN	0.005(2)	-11.5(5)	-10.94(46)
TF	0.024(4)	-14.5(4)	12.98(39)
VAE	0.037(1)	**-16.2(2)**	**-14.58(7)**
WGAN	0.036(1)	-16	-14.14(12)

Supplementary Tab. II. A summary of the best values of quality coverage (C_q), minimum value (MV) and utility (U) for the data portion $\epsilon = 0.001$ from Fig. 3 and for the generative models (QCBM, RNN, TF, VAE and WGAN). Values in bold correspond to the best performance among the different models. Furthermore, the digits in parentheses correspond to the uncertainty over the last digit of the reported numbers. For track T1, the QCBM is the winner, whereas for track T2, QCBM is competitive with the VAE.

T1	C_q	MV	U
QCBM	**7e-4**	**-19**	-18.19(7)
RNN	**7e-4**	**-19**	-15.01(12)
TF	6.7(2)e-4	**-18.8(2)**	-16.12(26)
VAE	**7e-4**	**-19**	**-19**
WGAN	**7e-4**	**-19**	-17.86(46)

T2	C_q	MV	U
QCBM	0.038(1)	-15.8(1)	-14.52(16)
RNN	0.015(2)	-13.5(2)	-12.78(15)
TF	0.027(5)	-14.7(5)	-13.74(41)
VAE	**0.05**	-17	-15.58(5)
WGAN	**0.051(2)**	**-17.7(3)**	**-15.96(19)**

Supplementary Tab. III. A report of the numerical values as in Supplementary Tab. II for the data portion $\epsilon = 0.01$ reported in Fig. 3. Values in bold correspond to the best performance among the different models. For track T1, VAE is the winner, whereas, for track T2, WGAN is superior compared to the other models.

SUPPLEMENTARY REFERENCES

[1] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz, "A generative modeling approach for benchmarking and training shallow quantum circuits," npj Quantum Information **5**, 45 (2019).

[2] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini, "Parameterized quantum circuits as machine learning models," Quantum Science and Technology **4**, 043001 (2019).

[3] Kaitlin Gili, Mohamed Hibat-Allah, Marta Mauri, Chris Ballance, and Alejandro Perdomo-Ortiz, "Do quantum circuit born machines generalize?" Quantum Science and Technology **8**, 035021 (2023).

[4] Nikolaus Hansen, "The cma evolution strategy: A tutorial," arXiv:1604.00772 (2016).

[5] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis, "CMA-ES/pycma on Github," (2019).

[6] Manuel S. Rudolph, Jacob Miller, Danial Motlagh, Jing Chen, Atithi Acharya, and Alejandro Perdomo-Ortiz, "Synergistic pretraining of parametrized quantum circuits via tensor networks," Nature Communications **14**, 8367 (2023).

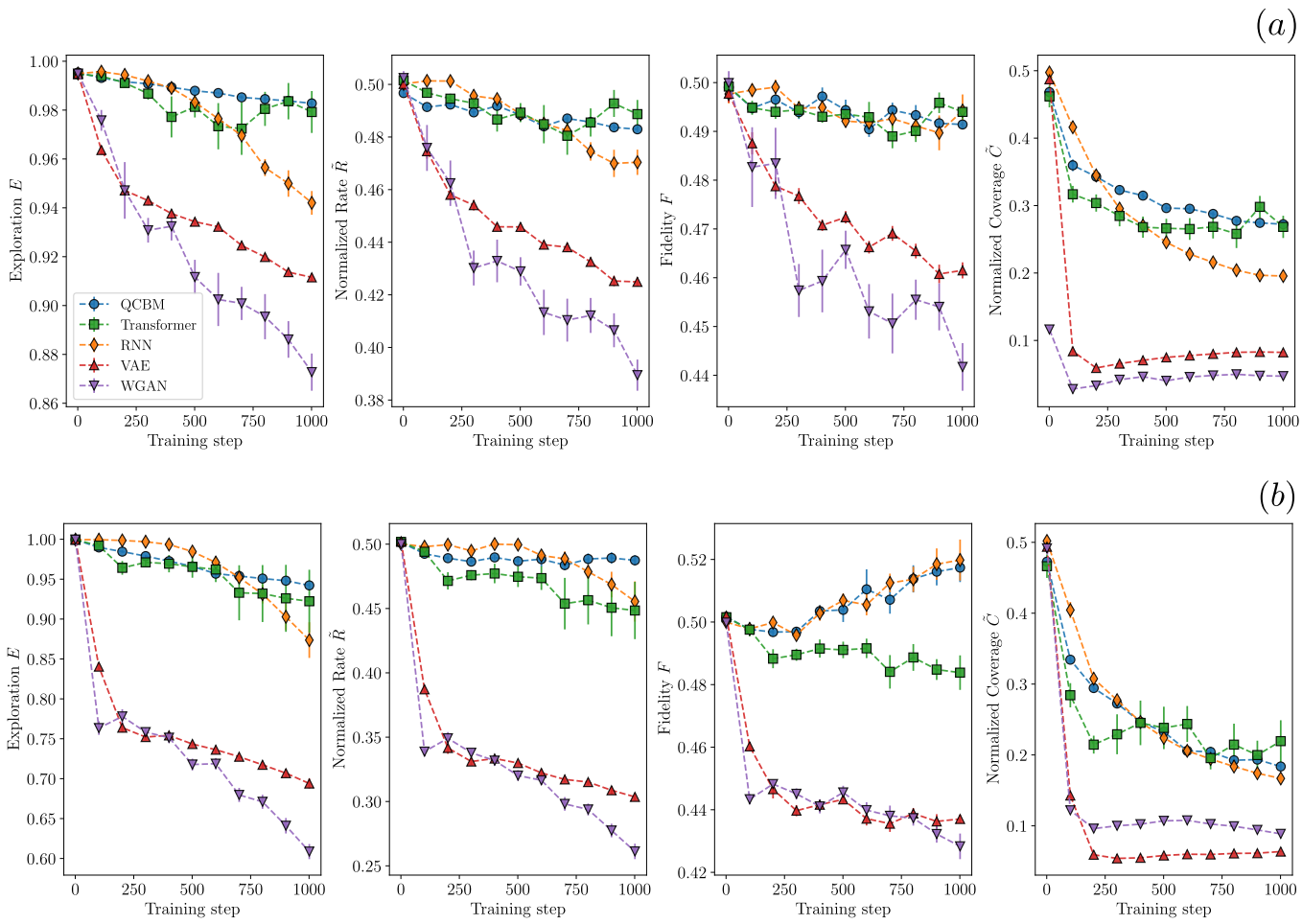
[7] Jin-Guo Liu and Lei Wang, "Differentiable learning of quantum circuit born machines," PRA **98**, 062324 (2018).

[8] Zachary C. Lipton, John Berkowitz, and Charles Elkan, "A critical review of recurrent neural networks for sequence learning," arXiv:1506.00019 (2015).

[9] Ian Goodfellow Yoshua Bengio and Aaron Courville, "Deep learning," MIT Press (2016).

[10] Giovanni S. Carmantini, Peter Beim Graben, Mathieu Desroches, and Serafim Rodrigues, "Turing computation with recurrent artificial neural networks," in *Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches - Volume 1583*, COCO'15 (CEUR-WS.org, Aachen, DEU, 2015) p. 1–9.

[11] Anton Maximilian Schäfer and Hans Georg Zimmermann, "Re-



Supplementary Fig. 1. **Validity-based generalization comparison between QCBMs, RNNs, TFs, WGANs and VAEs** for a system size $N_{\text{var}} = 20$. Here we plot the exploration E, normalized rate $\tilde{R}$, fidelity F and normalized coverage $\tilde{C}$ against the number of training steps for $\epsilon = 0.01$ in panel (a), and for $\epsilon = 0.001$ in panel (b). The models are trained using $N_{\text{seeds}} = 10$ random seeds, and the outcomes of the metrics are averaged over these seeds. The error bars are estimated as the one standard deviation, which can be computed for each metric as $\sqrt{\text{Variance}/N_{\text{seeds}}}$.

current neural networks are universal approximators," in *Artificial Neural Networks – ICANN 2006* (2006) pp. 632–640.

[12] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 28, edited by Sanjoy Dasgupta and David McAllester (PMLR, Atlanta, Georgia, USA, 2013) pp. 1310–1318.

[13] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, edited by Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi (Association for Computational Linguistics, Doha, Qatar, 2014) pp. 103–111.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019) pp. 8024–8035.

[15] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations ICLR 2015* (2015).

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," Advances in neural information processing systems **30** (2017).

[17] Huitao Shen, "Mutual information scaling and expressive power of sequence models," arXiv:1905.04271 (2019).

[18] Ian Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," arXiv:1701.00160 (2016).

[19] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin, "Cyclical annealing schedule: A simple approach to mitigating KL vanishing," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, edited by Jill Burstein, Christy Doran, and Thamar Solorio (Association for

Computational Linguistics, Minneapolis, Minnesota, 2019) pp. 240–250.

[20] Jonathan T. Barron, "Continuously differentiable exponential linear units," arXiv:1704.07483 (2017).

[21] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations ICLR 2014* (2014).

[22] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations* (2017).

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," Advances in neural information processing systems **27** (2014).

[24] Martin Arjovsky, Soumith Chintala, and Léon Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning* (PMLR, 2017) pp. 214–223.

[25] Ankan Dash, Junyi Ye, and Guiling Wang, "A review of generative adversarial networks (gans) and its applications in a wide variety of disciplines: From medical to remote sensing," IEEE Access , 1–1 (2023).

[26] Andrei Cristian Nica, Moksh Jain, Emmanuel Bengio, Cheng-Hao Liu, Maksym Korablyov, Michael M. Bronstein, and Yoshua Bengio, "Evaluating generalization in gflownets for molecule design," in *ICLR2022 Machine Learning for Drug Discovery* (2022).

[27] Kaitlin Gili, Marta Mauri, and Alejandro Perdomo-Ortiz, "Generalization metrics for practical quantum advantage in generative models," arXiv:2201.08770 (2022).