Training a Hopfield Variational Autoencoder with Equilibrium Propagation

Tom Van Der Meersch **Johannes Deleu** **Thomas Demeester**
IDLab, Ghent University – imec
Ghent, Belgium
first.last@ugent.be

Abstract

On dedicated analog hardware, equilibrium propagation is an energy-efficient alternative to backpropagation. In spite of its theoretical guarantees, its application in the AI domain remains limited to the discriminative setting. Meanwhile, despite its high computational demands, generative AI is on the rise. In this paper, we demonstrate the application of Equilibrium Propagation in training a variational autoencoder (VAE) for generative modeling. Leveraging the symmetric nature of Hopfield networks, we propose using a single model to serve as both the encoder and decoder which could effectively halve the required chip size for VAE implementations, paving the way for more efficient analog hardware configurations.

1 Introduction

Training with backpropagation (BP) has been essential for the success of deep learning, enabling the training of very large neural networks [1]. However, the pursuit for more biologically plausible neural networks and the promise of extremely energy-efficient, fast and compact analog implementations [2] have led to alternative models and learning strategies, such as the Equilibrium Propagation (EP) algorithm for a class of energy-based models including the Continuous Hopfield Network (CHN) [3]. Despite considerable algorithmic and conceptual extensions [4, 5, 6] since its introduction [7], much of the existing literature on EP remains confined to applications in discriminative settings. Seeking to bridge this gap, in this paper we are pioneering the application of EP within a generative framework, applied to image generation.

A CHN can be described as a neural network consisting of input, hidden, and output neurons, interlinked by *undirected* connections. This undirected nature of the connections justifies the question whether one and the same network could be used to both encode and decode (i.e. generate) an image. In a prior effort, EP has been employed for bidirectional training, yet this previous approach could only generate one canonical image per class [8]. In order to be able to sample from the data distribution instead, we have chosen to build a Variational Autoencoder (VAE) based on Hopfield networks where encoding and decoding can be done with the same undirected network.

Our proof-of-concept results in Section 4 confirm that a Hopfield VAE can be trained with the local learning strategy of EP. Moreover, the same network can indeed serve both as the encoder and decoder. While analyzing different CHN architectures, we also find that a layered model is outperformed by a fully connected model. Although our experiments hardly allow for any general claims in that respect, they highlight a unique property absent in directed networks, warranting future research.

2 Preliminaries

This section summarizes the key concepts the Hopfield VAE is built on, explaining the Continuous Hopfield Network, Equilibrium Propagation training, and the classical Variational Autoencoder.

Continuous Hopfield Networks: The CHN as characterized by Bengio and Fischer [3] is a particular type of Hopfield Network – a recurrent neural network (RNN) whose settling process into an energy minimum is described by an associated Lyapunov function [9, 10, 11]. The CHN possesses continuous states and dynamics, and its energy function $E(\boldsymbol{s})$ is defined as

$$E(\boldsymbol{s}) := \frac{1}{2}||\boldsymbol{s}||^2 - \frac{1}{2}\rho(\boldsymbol{s}^T)\boldsymbol{W}\rho(\boldsymbol{s}) \tag{1}$$

in which the vector $\boldsymbol{s}$ represents the state of all neurons, $\boldsymbol{W}$ a symmetrical weight matrix (with zero-diagonal), and $\rho(\cdot)$ a bounded component-wise non-linearity. Note that we chose not to use a bias for each neuron. This seems to be common practice in previous work on EP [5, 12] and small experiments with an additional bias did not show an improved performance.

The temporal dynamics of the model are governed by the energy minimization:

$$\frac{d\boldsymbol{s}}{dt} = -\nabla_{\boldsymbol{s}} E = -\boldsymbol{s} + \rho'(\boldsymbol{s}) \odot \big(\boldsymbol{W}\rho(\boldsymbol{s})\big) \tag{2}$$

with $\odot$ the component-wise product. This differential equation is integrated numerically, often through the following first-order update scheme for iteration i, with a small time step δt:

$$\boldsymbol{s}_{i+1} = (1-\delta t)\,\boldsymbol{s}_i + \delta t\;\rho'(\boldsymbol{s}_i) \odot \big(\boldsymbol{W}\rho(\boldsymbol{s}_i)\big). \tag{3}$$

The state vector $\boldsymbol{s}$ is typically divided into the *input* neurons $\boldsymbol{x}$, *hidden* neurons $\boldsymbol{h}$, and *output* neurons $\boldsymbol{y}$. During inference, the vector $\boldsymbol{x}$ is clamped to the given input pattern, upon which the system dynamics induce a transition phenomenon towards an equilibrium state, after which the output state vector $\boldsymbol{y}$ is read out. Note that in our experiments, the initial state for all but the input neurons is set to zero.

Equilibrium Propagation: The standard approach to training RNNs is to use BP through time (BPTT) [13, 14]. However, when dealing with energy-based models like CHNs, EP provides an alternative to finding suitable parameter updates. As opposed to back-propagation, EP is a so-called *local* learning scheme, requiring only local activations, and a single computational circuit [7]. It is therefore regarded as a promising candidate training strategy for extremely energy-efficient analog accelerators [15].

For supervised learning with EP, the model first settles into a *free* equilibrium state, given input $\boldsymbol{x}$. This is called the *free phase*. An extra term is then added to the energy function, representing the loss $\mathcal{L}(\boldsymbol{y}, \boldsymbol{y}_{\text{target}})$ of the output neurons w.r.t. the ground truth output $\boldsymbol{y}_{\text{target}}$. The so-called *total energy* F is written as $F = E + \beta\mathcal{L}(\boldsymbol{y}, \boldsymbol{y}_{\text{target}})$, with β a small positive constant. Replacing E by F in (2), leads to the following update rule for the output neurons

$$\boldsymbol{y}_{i+1} = (1-\delta t)\,\boldsymbol{y}_i + \delta t\;\rho'(\boldsymbol{y}_i) \odot \big(\boldsymbol{W}_y\rho(\boldsymbol{s}_i)\big) - \delta t\beta\nabla_{\boldsymbol{y}}\mathcal{L}(\boldsymbol{y}, \boldsymbol{y}_{\text{target}}) \tag{4}$$

with $\boldsymbol{W}_y$ representing the rows of $\boldsymbol{W}$ corresponding to neural connections of the output neurons. In the so-called *weakly clamped phase*, starting from the free equilibrium state, the last term in (4) effectively nudges the output state $\boldsymbol{y}$ towards the desired $\boldsymbol{y}_{\text{target}}$ while the network settles into the *weakly clamped* equilibrium $\boldsymbol{s}_{\boldsymbol{x}}^{\beta}$. Thirdly, a similar phase is executed, after replacing β by its opposite in the total energy, leading to the weakly clamped equilibrium $\boldsymbol{s}_{\boldsymbol{x}}^{-\beta}$.

These phases are executed within each training iteration, whereby every weight w is updated proportionally to the difference between the partial derivatives of the total energy with respect to the weights, given input $\boldsymbol{x}$, in both weakly clamped equilibria

$$\Delta w \propto \frac{1}{2\beta}\left(\left.\frac{\partial F}{\partial w}\right|_{\boldsymbol{s}_x^{\beta}} - \left.\frac{\partial F}{\partial w}\right|_{\boldsymbol{s}_x^{-\beta}}\right) \tag{5}$$

which can be shown to yield a second order approximation to the actual BPTT derivative [5].

Variational Autoencoders: A VAE is a deep generative latent variable model aiming to approximate the data distribution $p(\boldsymbol{x})$ through learning encodings and decodings between data space and a latent space [16]. It employs an encoder network, parameterized by ϕ, to approximate the posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ over latent variables $\boldsymbol{z}$, given data sample $\boldsymbol{x}$. A decoder network, parameterised by θ, produces a distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ to generate data from latent variables. As objective to jointly train encoder and decoder, a lower bound to the marginal likelihood (or model evidence) $p(\boldsymbol{x})$ over the training data is maximized, i.e., the so-called *evidence lower bound* (ELBO), which can be formulated as [17]

$$\text{ELBO} = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\text{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||\, p(\boldsymbol{z})) \tag{6}$$

Here, the first term represents the reconstruction loss, which can be simplified to the negative sum of squared errors (SSE) under specific Gaussian assumptions. The Kullback–Leibler divergence (D_{KL}) term ensures proximity between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ so that after training, samples can be drawn from this prior to generate new data with the decoder. When assuming a standard normal distribution for the prior $p(\boldsymbol{z})$, and for $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ a normal distribution with diagonal covariance matrix, whose means $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ are calculated by the encoder network, D_{KL} can be written as

$$D_{KL}[\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}))\|\mathcal{N}(0, I)] = \frac{1}{2}\sum_{d=1}^{D}\left(-\log \boldsymbol{\sigma}_d^2 - 1 + \boldsymbol{\sigma}_d^2 + \boldsymbol{\mu}_d^2\right) \tag{7}$$

In this paper, we use a variant called the β-VAE, which introduces an additional hyperparameter β_{KL} to control the contribution of the D_{KL} term in (6) [18].

3 Training a Variational Autoencoder with Equilibrium Propagation

Free phase: Finding the free phase equilibrium in our proposed Hopfield VAE is done by treating the encoder and decoder as separate Hopfield networks, each with their own energy function. Upon assigning the input image $\boldsymbol{x}$ to the encoder input states, it settles into an equilibrium without taking into account the decoder. The encoder output can be written as $\boldsymbol{y}_{\text{enc}} = [\boldsymbol{y}_\mu; \boldsymbol{y}_{\log\sigma^2}]$, and allows sampling $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{y}_\mu, \text{diag}(\exp(\frac{1}{2}\boldsymbol{y}_{\log\sigma^2})))$. The latent vector $\boldsymbol{z}$ is used as a fixed input to the decoder which then settles into its own equilibrium after which its output neurons $\boldsymbol{y}_{\text{dec}}$ form the reconstructed image.

Weakly clamped phase (decoder): With the sum of squared errors (SSE) as the reconstruction loss $\mathcal{L}_{\text{SSE}} = \frac{1}{2}\sum(\boldsymbol{y}_{\text{dec}} - \boldsymbol{x})^2$, the following loss gradient is applied as the last term of the update equation (4) for the decoder energy:

$$\nabla_{\boldsymbol{y}_{\text{dec}}}\mathcal{L}_{\text{SSE}}(\boldsymbol{y}_{\text{dec}}, \boldsymbol{x}) = \boldsymbol{y}_{\text{dec}} - \boldsymbol{x} \tag{8}$$

Weakly clamped phase (encoder): The *encoder* parameters also need to be optimized to lower the reconstruction loss at the *decoder* output. This means that during the weakly clamped phase, $\nabla_{\boldsymbol{y}_\mu}\mathcal{L}_{\text{SSE}}(\boldsymbol{y}_{\text{dec}}, \boldsymbol{x})$ and $\nabla_{\boldsymbol{y}_{\log\sigma^2}}\mathcal{L}_{\text{SSE}}(\boldsymbol{y}_{\text{dec}}, \boldsymbol{x})$ need to be estimated. As further explained in Appendix A, the Equilibrium Propagation framework allows directly estimating $\nabla_{\boldsymbol{z}}\mathcal{L}_{\text{SSE}}$, in line with (5):

$$\nabla_{\boldsymbol{z}}\mathcal{L}_{\text{SSE}} = \lim_{\beta\to 0}\frac{1}{2\beta}\left(\left.\frac{\partial F}{\partial \boldsymbol{z}}\right|_{\boldsymbol{s}_{\text{dec}}^{\beta}} - \left.\frac{\partial F}{\partial \boldsymbol{z}}\right|_{\boldsymbol{s}_{\text{dec}}^{-\beta}}\right) \tag{9}$$

in which F denotes the total energy of the decoder, $\boldsymbol{s}_{\text{dec}}^{\beta}$ denotes the weakly clamped decoder equilibrium state and $\boldsymbol{z}$ is the *decoder* input. Furthermore, since $\boldsymbol{z} = \boldsymbol{y}_\mu + \boldsymbol{\epsilon} \odot \boldsymbol{y}_{\boldsymbol{\sigma}}$ (with $\boldsymbol{y}_\sigma = \exp(\frac{1}{2}\boldsymbol{y}_{\log\sigma^2})$, and the components of $\boldsymbol{\epsilon}$ drawn from a standard normal distribution), we can write:

$$\nabla_{\boldsymbol{y}_\mu}\mathcal{L}_{\text{SSE}} = \nabla_{\boldsymbol{z}}\mathcal{L}_{\text{SSE}} \quad \text{and} \quad \nabla_{\boldsymbol{y}_{\log\sigma^2}}\mathcal{L}_{\text{SSE}} = \frac{1}{2}\,\boldsymbol{y}_\sigma \odot \boldsymbol{\epsilon} \odot \nabla_{\boldsymbol{z}}\mathcal{L}_{\text{SSE}} \tag{10}$$

These terms function as the contribution for the reconstruction loss in the update equation (4) for the decoder energy. Besides the reconstruction loss, D_{KL} also needs to be minimized during training. From (7), we find that the following loss gradients need to be added to the update equation of the encoder energy:

$$\nabla_{\boldsymbol{y}_\mu}\mathcal{L}_{D_{\text{KL}}} = \boldsymbol{y}_\mu \quad \text{and} \quad \nabla_{\boldsymbol{y}_{\log\sigma^2}}\mathcal{L}_{D_{\text{KL}}} = \frac{1}{2}(-\mathbf{1} + \exp(\boldsymbol{y}_{\log\sigma^2})) \tag{11}$$

4 Experimental Results

We present three experiments, comparing the four different models listed below. Details on hyper-parameter tuning are provided in Appendix B. The Fréchet inception distance (FID) serves as the optimization target as it captures both the fidelity and diversity of generated samples, corresponding better with human perception than similar metrics like the Inception Score [19].

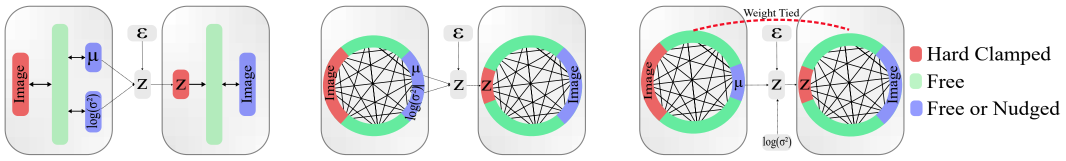

Figure 1: Proposed architectures. Left: H-VAE, middle: DH-VAE, right: TDH-VAE. Note that the encoder output of the TDH-VAE only outputs a μ vector to have a mirrored encoder and decoder.

- *Forward VAE* (F-VAE): Used as a baseline model, it consists of a feedforward encoder and decoder trained with BP. The encoder and decoder have one hidden layer with a sigmoid non-linearity. The output of the decoder goes through a hyperbolic tangent.
- *Hopfield VAE* (H-VAE): Similar to the F-VAE as encoder and decoder both have a single hidden layer without within-layer connections, but now they are CHNs, trained with EP.
- *Dense Hopfield VAE* (DH-VAE): This variation on the Hopfield VAE no longer has a layered structure, and *all* neurons are interconnected (even within the output layer).
- *Tied Dense Hopfield VAE* (TDH-VAE): This model is similar to the DH-VAE but ties the encoder and decoder weights. To have the same dimensions for the encoder and decoder weight matrix, $\log(\boldsymbol{\sigma}^2)$ is no longer predicted but replaced by a tunable but fixed hyperparameter σ.

Figure 1 visualizes the architectures of all EP-trained VAEs and Table 1 presents the FID scores on the MNIST test set. Here are our main insights:

Model	FID
F-VAE	60.5
H-VAE	73.7
DH-VAE	49.6
TDH-VAE	58.8

Table 1: FID scores (lower is better) on the MNIST test set.

Training a Hopfield VAE with Equilibrium Propagation is possible: The FID score of the EP-trained H-VAE is not quite as good as its backprop-trained counterpart F-VAE, but it can be trained. This is visually confirmed by the reconstructed and generated MNIST samples shown in Appendix C.

Dense CHNs yield a better Hopfield VAE: In the first experiment, both models are layered. However, (1) imposes no conditions on the weight matrix. Previous work on CHNs and EP shows that adding random connections to a layered CHN reduces vanishing gradients [20]. Our preliminary tests show that fully connecting all neurons yields optimal results. To measure the effect of this change, we reran the previous parameter sweep on the DH-VAE, resulting in a strongly improved FID. This strong result may be related to the ability of pixel neurons to directly influence each other, or it could be the result of the increased parameter count, and requires additional research. In any case, we consider deviating from the standard layered architectures in forward models a valid path for future research on energy-based models.

A Hopfield VAE with the same network as encoder and decoder works: Finally, given the symmetric nature of CHNs, whereby input and output are only determined by how they are trained, we investigate whether the same set of weights can be used for encoder and decoder. In one direction it can be used to encode an image into the latent space distribution $p(\boldsymbol{z}|\boldsymbol{x})$, whereas in the opposite direction, it can generate an image when provided with a latent space sample $\boldsymbol{z}$. Table 1 shows that the TDH-VAE can indeed be trained and even performs better than both the F-VAE and the H-VAE, although there is a loss in performance compared to DH-VAE. We hypothesize that the latter is due to the simplification of the hidden state distribution, with a fixed value for σ.

5 Conclusions and Future Work

In this paper, we have extended the application of EP to generative tasks, specifically for training VAEs. First, we compared a baseline and a CHN model trained with EP. Moving from a layered to a fully connected model proved to be beneficial. Finally, we leverage the inherent symmetry of Hopfield networks to use the same model as encoder and decoder. While this causes some performance degradation, it offers a promising pathway to optimizing energy-efficient analog VAEs.

VAEs trained through EP still yield poor images (although of similar quality to feedforward models of comparable capacity), necessitating larger, more advanced models. To tackle encountered vanishing gradients, research into initialization strategies and non-linearities is needed. Additionally, using Hierarchical Associative Memories [10], which are still trainable with EP, could add missing inductive biases like convolutions and layer normalization.

Acknowledgments

This research was partly funded by the Research Foundation - Flanders (FWO-Vlaanderen) under grant G0C2723N and by the Flemish Government (AI Research Program). Additional thanks are extended to Félix Koulischer and Cédric Goemaere for their proofreading and constructive comments.

References

[1] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* (1986). DOI: `10.1038/323533a0`.

[2] Jack Kendall et al. "Training End-to-End Analog Neural Networks with Equilibrium Propagation". In: *arXiv* (2020). arXiv: `2006.01981`.

[3] Yoshua Bengio and Asja Fischer. "Early Inference in Energy-Based Models Approximates Back-Propagation". In: *arXiv* (2015). arXiv: `1510.02777`.

[4] Benjamin Scellier et al. *Extending the Framework of Equilibrium Propagation to General Dynamics*. 2018. URL: `https://openreview.net/forum?id=SJTB5GZCb`.

[5] Axel Laborieux et al. "Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing its Gradient Estimator Bias". In: *Frontiers in Neuroscience* (2020). DOI: `10.3389/fnins.2021.633674`.

[6] Axel Laborieux and Friedemann Zenke. "Holomorphic equilibrium propagation computes exact gradients through finite size oscillations". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 12950–12963. URL: `https://papers.nips.cc/paper_files/paper/2022/file/545a114e655f9d25ba0d56ea9a01fc6e-Paper-Conference.pdf`.

[7] Benjamin Scellier and Yoshua Bengio. "Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation". In: *Frontiers in Computational Neuroscience* 11 (2017). DOI: `10.3389/fncom.2017.00024`.

[8] Ahmed Faraz Khan. "Bidirectional Learning in Recurrent Neural Networks Using Equilibrium Propagation". MA thesis. University of Waterloo, Sept. 2018. URL: `https://uwspace.uwaterloo.ca/handle/10012/13957`.

[9] J J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8 (1982), pp. 2554–2558. DOI: `10.1073/pnas.79.8.2554`.

[10] Dmitry Krotov. "Hierarchical Associative Memory". In: *arXiv* (2021). arXiv: `2107.06446`.

[11] Hubert Ramsauer et al. "Hopfield Networks is All You Need". In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=tL89RnzIiCd`.

[12] Benjamin Scellier and Yoshua Bengio. "Equivalence of equilibrium propagation and recurrent backpropagation". In: *Neural computation* 31.2 (2019), pp. 312–329.

[13] Paul Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: `10.1109/5.58337`.

[14] Maxence Ernoult et al. "Updates of equilibrium prop match gradients of backprop through time in an RNN with static input". In: *Advances in neural information processing systems* 32 (2019).

[15] Seokjin Oh et al. "Memristor Crossbar Circuits Implementing Equilibrium Propagation for On-Device Learning". In: *Micromachines* 14.7 (2023). DOI: `10.3390/mi14071367`.

[16] Diederik P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations* (2013). URL: `https://openreview.net/forum?id=33X9fd2-9FyZd`.

[17] Calvin Luo. "Understanding Diffusion Models: A Unified Perspective". In: *arXiv* (2022). arXiv: `2208.11970`.

[18] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017. URL: `https://openreview.net/forum?id=Sy2fzU9gl`.

[19] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[20] Jimmy Gammell et al. "Layer-Skipping Connections Improve the Effectiveness of Equilibrium Propagation on Layered Networks". In: *Frontiers in Computational Neuroscience* 15 (2021). DOI: `10.3389/fncom.2021.627357`.

[21] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29 (2012), pp. 141–142. DOI: `10.1109/MSP.2012.2211477`.

[22] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: `https://www.wandb.com/`.

A Equilibrium Propagation for Input Gradients

While the original formulation of EP was designed for finding the gradient of the loss with respect to the parameters, it can be shown that EP also allows for finding the gradient of the loss with respect to the input. To our best knowledge this has never explicitly been stated or used before. The proof that this is possible is almost identical to the proof in Appendix A of the original EP paper by Scellier and Bengio [7]. The only additional required insight, is that you need to split up the function argument $\boldsymbol{s}_\theta^\beta$ in $\tilde{\boldsymbol{s}}_\theta^\beta$ and $\boldsymbol{x}$ with $\tilde{\boldsymbol{s}}_\theta^\beta$ the part of the state that is not hard clamped (everything except the input) and $\boldsymbol{x}$ the input. In all the proofs, the derivative with respect to the parameters $\frac{\partial}{\partial \theta}$ needs to be changed to $\frac{\partial}{\partial \boldsymbol{x}}$. This can be done without violating any of the assumptions made in the proof. An intuitive explanation is that just like θ, the input $\boldsymbol{x}$ can be seen as some fixed quantity influencing the settling process and therefore it is possible to use EP to find the derivative with respect to this quantity.

B Hyperparameters and Training Details

In this section, we discuss details that might be useful for reproducibility. First, we highlight some of our architecture choices and why they were made, followed by an overview of the hyperparameter ranges used during the Bayesian optimization sweeps.

B.1 Architectural Choices

In all CHNs, a compressed sigmoid was used as a non-linearity, specifically $\rho(s) = \frac{1}{1+e^{-3s}}$, which was empirically chosen during initial experiments. Notice that the derivative $\rho'(s)$ is a symmetric function, which is an intentional choice stemming from the CHN update rule which is given by $\boldsymbol{s}_{i+1} = (1-\delta t)\,\boldsymbol{s}_i + \delta t\; \rho'(\boldsymbol{s}_i) \odot \big(\boldsymbol{W}\rho(\boldsymbol{s}_i)\big)$. The occurrence of $\rho'(\boldsymbol{s}_i)$ means that it is difficult for $\boldsymbol{s}_i$ to take values in the saturation region of ρ. As we want the range of possible μ-values to be symmetrical around zero, we chose the non-linearity to be symmetrical around zero as well so that the saturation region of ρ are symmetrical around zero and the predicted μ values can just as easily be positive as negative.

All experiments were performed on the MNIST dataset which contains grayscale images of handwritten digits with a resolution of 28 by 28 pixels [21]. The data was rescaled to a range of $[-1, 1]$. For reasons stated above, we choose a non-linearity ρ so that ρ' is symmetrical. As the output pixel neurons of the decoder now have a symmetrical non-linearity around zero, it makes sense to also shift the pixel range from $[0, 1]$ to $[-1, 1]$ to make the most extreme outputs equally saturated and therefore equally hard to achieve.

B.2 Hyperparameter Optimization

Every model was optimized using the Bayesian optimization implementation of Weights and Biases [22]. The hyperparameter space explored in this study is summarized in Table 2. The final optimized hyperparameters are detailed in Table 3. Each sweep was run for approximately 17 hours before it was terminated and the model with the best final FID score was kept.

C Visual Examples

In this section, visual examples are given for each of the four discussed models. Figure 2 showcases one original image for each of the ten classes in the MNIST dataset. Figure 3 shows the reconstruction of these original images by each of the four models. Finally, Figure 4 shows ten randomly generated images from each of the four models.

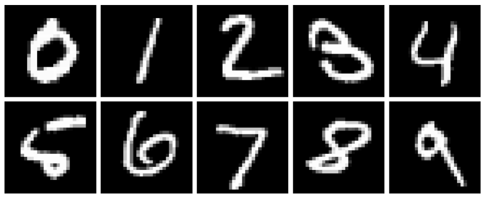

Figure 2: Original images from the MNIST test dataset. One example of each class is shown.

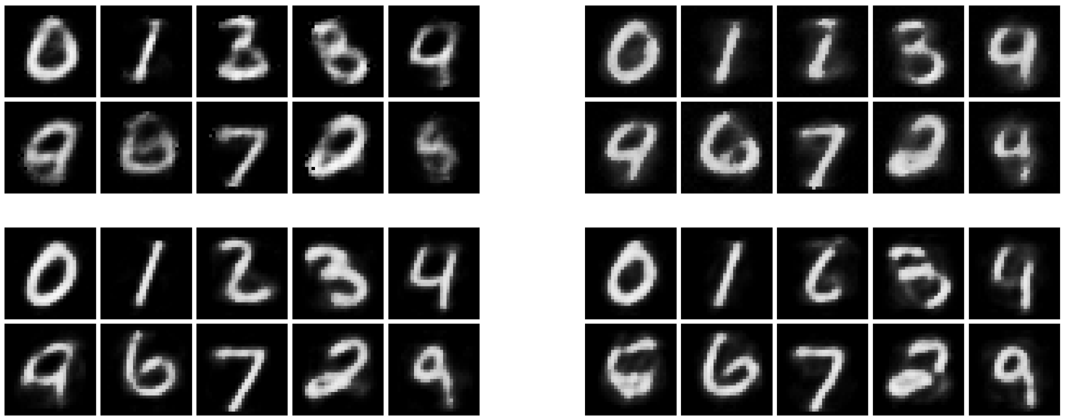

Figure 3: Images from the test set which were reconstructed by the four discussed models. Top left: F-VAE, top right: H-VAE, bottom left: DH-VAE, bottom right: TDH-VAE

Parameter	Minimum Value	Maximum Value	Distribution
β_{KL}	0.5	2 (10)	Logarithmically uniform
Learning rate	10^{-5}	10^{-3}	Logarithmically uniform
Hidden dimension size	10	2000	Logarithmically uniform
Latent dimension size	2	200	Logarithmically uniform
Fixed σ	**0.3**	**1**	**Logarithmically uniform**

Table 2: The sweep ranges for all experiments. Values between brackets indicate that this value was used for the F-VAE case. Bold parameters are only relevant for the TDH-VAE experiments.

Model	β_{KL}	Learning Rate	Hidden Dimension Size	Latent Dimension Size	Fixed σ
F-VAE	5.55	9.91×10^{-4}	1820	10	/
H-VAE	0.517	6.68×10^{-5}	1842	25	/
DH-VAE	0.556	9.41×10^{-4}	1696	25	/
TDH-VAE	1.90	9.55×10^{-4}	833	69	0.429

Table 3: The hyperparameters for each model resulting from each of the sweeps.

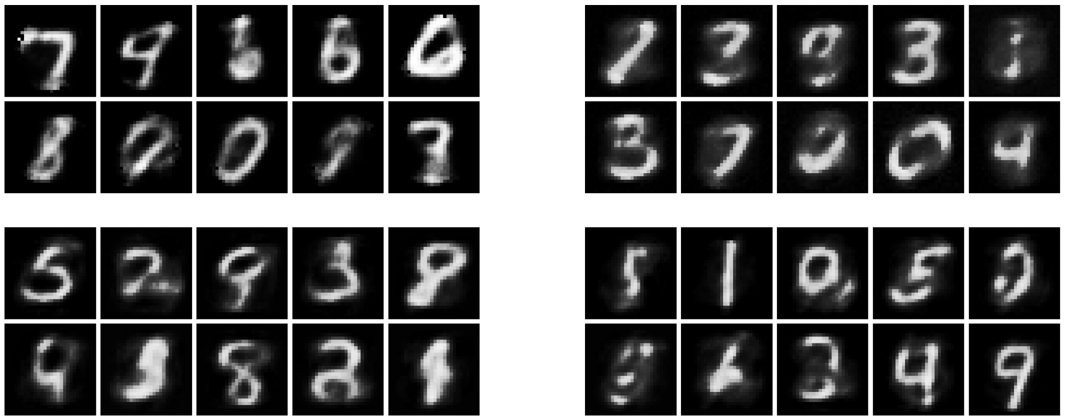

Figure 4: Images generated by the decoder for each of the four models with inputs sampled randomly from a standard normal prior. Top left: F-VAE, top right: H-VAE, bottom left: DH-VAE, bottom right: TDH-VAE