RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] Variational Neural Cellular Automata

Albert Aillet[1] and Simon Sondén[1]
[1]Equal contributions, KTH Royal Institute of Technology

Edited by
Koustuv Sinha,
Maurits Bleeker,
Samarth Bhargav

Received
04 February 2023

Published
20 July 2023

DOI
10.5281/zenodo.8173729

Reproducibility Summary

Scope of Reproducibility – The main claim of the paper being reproduced is that the proposed Variational Neural Cellular Automata (VNCA) architecture, composed of a convolutional encoder and a Neural Cellular Automata (NCA)-based decoder, is able to generate high-quality samples. The paper presents two variants of this VNCA decoder: the doubling VNCA variant that is claimed to have a simple latent space, and the non-doubling VNCA variant that is claimed to be optimized for damage recovery and stability over many steps.

Methodology – To reproduce the results, we re-implemented all of the VNCA models and a fully-convolutional baseline in JAX, by using the descriptions given in the paper. We then followed the same experimental setup and hyperparameter choices as in the original paper. All of the models were trained on a TPU v3-8 provided by Kaggle, with a total budget of around 4 TPU hours, not counting unreported experiments.

Results – All but one of the figures and results from the original study were possible to reproduce. The obtained Evidence Lower Bound (ELBO) of the doubling VNCA was within 0.3% of the stated and for the non-doubling VNCA the ELBO was within 1.8% and the observed damage recovery was similar. We were however not able to reproduce the t-SNE reduction experiment for the baseline and were therefore not able to show the VNCA decoder having a cleaner t-SNE separation than the baseline.

What was easy – The implementation of the convolutional baseline and most parts of the NCA-based decoder were straightforward to re-implement based on the description provided in the paper.

What was difficult – One of the difficulties faced in the reproduction study was obtaining the labels of the binarized MNIST dataset used in the original paper since it officially is provided without labels. This made it unclear how the original paper got the labels for the latent space visualization. Additionally, implementing the pool for the non-doubling version of the VNCA model with a distributed training setup was challenging as it required operations between TPU cores.

Communication with original authors – No communication was made with the original authors.

1 Introduction

This report presents a reproduction of a part of the results from the paper "Variational Neural Cellular Automata" [1] published in ICLR 2022. The authors of the original paper build upon previous research around fully-differentiable cellular automata called Neural Cellular Automata (NCA) and Variational Auto-Encoders (VAE). They propose a novel generative model, a VAE whose decoder is implemented using a NCA, which they name Variational Neural Cellular Automata (VNCA).

2 Scope of reproducibility

In this study, we focus on the results in the original paper which treat the binarized MNIST dataset [2]. This includes all the proposed models which are: a baseline consisting of a convolutional VAE, a doubling version of the VNCA, and a non-doubling version of the VNCA that is trained to be resilient to damage during generation.
We identified the following claims in the original paper:

Claim 1 The doubling VNCA architecture, using a self-organizing process, has similar generative performance as a convolutional baseline.

Claim 2 The latent space of the Doubling VNCA architecture has a simpler t-SNE structure than that of the convolutional baseline.

Claim 3 The non-doubling VNCA architecture performs well on damage recovery tasks.

3 Methodology

To reproduce the results, all models and methods are re-implemented from the indications present in the original paper. The original implementation was developed using PyTorch and is available on GitHub [3]. Our re-implementation uses JAX [4] and Equinox [5], and we adhere to the same experimental setup and hyperparameters as the original authors. The training and evaluation data was the binarized MNIST dataset [2], as used in the original paper. A TPU v4-8 from Kaggle was used with a 4 TPU hour budget.

3.1 Model descriptions

The VNCA architecture is composed of a convolutional encoder and a NCA-based decoder. The encoder $q_\phi(z|x)$ is the same for all the VNCA models and the fully-convolutional baseline. The decoder $p_\theta(x|z)$ is NCA-based, meaning that the only parameters contained in the decoder are those of an iterative update rule that does local-only communication between neighboring cells. The cells of the NCA are vectors of size $|Z|$. It should also be noted that the aggregate of all NCA steps can be efficiently implemented as a sequence of CNN layers using 3×3 or 1×1 kernels that for a single cell represent local communication and linear layers respectively.

The first variant of the VNCA model is the doubling VNCA variant. This model is loosely inspired by the process of cellular growth. Presented in figure 1 is a corrected version of figure 2 from the original paper [1], that is slightly misleading as it appears as if the NCA decoder ends with a doubling operation, severely limiting the decoders expressive power. The model would then in practice be constrained to generate images of resolution 16×16 which would be doubled to 32×32 without any additional NCA steps. However, as this was not the case in the figures showing the growing process, the mistake could be detected without inspecting the code.

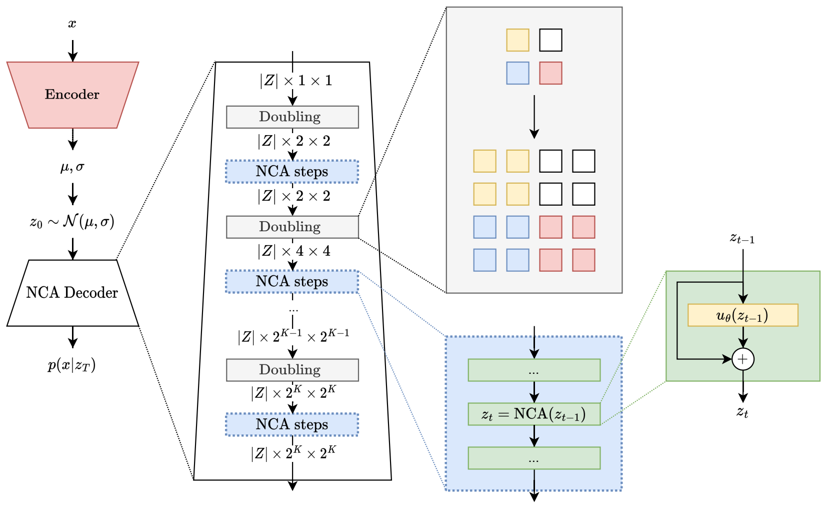


Figure 1. Overview of the doubling VNCA model, (inspired by figure 2 from [1]). **Left**: Whole reconstruction process, input image x is encoded using the convolutional encoder and z_0 is sampled from the distribution $\mathcal{N}(\mu, \sigma)$ and then fed into the NCA decoder. **Middle left**: The decoder consists of K doubling steps, each followed by a number of NCA steps, the shape of the multidimensional array between each step is shown on the arrows. **Top middle right**: The doubling operation repeats the grid as depicted, each cell is repeated four times. **Bottom middle right**: The NCA steps consist as the name indicates of multiple steps using the NCA. It is to be noted that the NCA in all the steps contains the same parameters θ. **Right**: The NCA step is defined by a local-only communication function u_θ that is added to the input.

The second variant of the VNCA model is the non-doubling model. The decoder of this model repeats the initial sample from the latent distribution over the entire grid and then runs T NCA steps. As the non-doubling VNCA variant does not change the resolution, it can be run for any number of steps T and can be optimized for damage recovery and stability over many steps. To optimize for these properties, a pool of previous samples z_T is stored and updated after each training batch. Half of every training batch is sampled from this pool, half of which are subjected to damage by setting a $|Z| \times H/2 \times W/2$ slice to zero. The number of NCA steps T for each batch is sampled between T_{min} and T_{max}. This training procedure will make a variable number of NCA steps T and apply damage to some of the samples, which will encourage stability and damage recovery. Due to this training procedure, this model does not exactly maximize the ELBO but as shown by our results, it still produces good samples.

In summary, these models were trained from scratch:

- Convolutional Variational Autoencoder baseline with $10,282,497$ parameters.
- Doubling Variational Neural Cellular Automata with $6,585,088$ parameters.
- Non-Doubling Variational Neural Cellular Automata with $5,042,432$ parameters.

3.2 Datasets

The dataset used by the original authors [1] is the publicly available statically binarized version of the MNIST dataset [2], which contains binary images of size 28×28. However, to account for the fact that the doubling VNCA only can produce outputs of size with powers of 2, the dataset is padded with zeros to become 32×32. Similarly, as the original

MNIST dataset [6], the binarized MNIST dataset contains $60,000$ training samples and $10,000$ testing samples. As this version of the dataset is usually used for generative tasks, labels are not provided.

3.3 Hyperparameters

The hyperparameters are described in section 3 of [1] where it was stated that unless otherwise specified, all models are trained using a batch size of 32, the Adam optimizer [7] was used, a learning rate of 10^{-4} was applied, gradient clipping was used with a norm of 10 [8], a latent vector of size $|Z| = 256$ was used and the models were trained for a total of $100,000$ gradient updates. These parameters were therefore used for training the convolutional baseline and the doubling VNCA. For the non-doubling VNCA, it is indicated that a larger batch size of 128 and a smaller latent vector size of $|Z| = 128$ was used. For the non-doubling VNCA $T_{\min} = 32$ and $T_{\max} = 64$ were set. When calculating the test ELBO, T was not sampled but instead $T = 36$ was used. Since all hyperparameters were provided no searching was needed.

3.4 Experimental setup and code

The first experiment consisted of training each model with the stated hyperparameters and then evaluating the ELBO with 128 importance weights on the test set.

In the second experiment, multiple samples were created from each model by sampling from a standard normal distribution of size $|Z|$ and then passing it through the doubling VNCA decoder and sampling from the output Bernoulli distribution. Test reconstruction was done in a similar way by passing test images into the model and sampling from the output Bernoulli distribution. The growth process was visualized by recording the first channel of each latent vector of decoder after each NCA step. The parameters of the final Bernoulli distribution is shown (meaning that the average of the samples).

In the third experiment, linear interpolation was conducted by sampling two random points from a standard normal distribution of size $|Z|$ and then creating 8 uniformly distant points linearly between them. All of these points were then fed through the doubling VNCA decoder and the results were plotted. A t-SNE reduction [9] is also done on $5,000$ randomly chosen images from the test set, encoded into the latent space of the doubling VNCA and convolutional baseline using their respective encoders.

In the fourth experiment, a standard normal distribution of size $|Z|$ was sampled 11 times and was then fed through the non-doubling decoder for 40 steps. Each produced image was then subjected to damage by setting a random $|Z| \times H/2 \times W/2$ slice to zero and then fed through the non-doubling decoder for 40 steps again.

Beyond the experiments of the original paper, we also investigated the representations of the individual cells' latent space using a linear probe [10]. This was done to learn how much the model relies on the individual cell latent vector for generating the image, as the cell latent vector can potentially encode the content of the entire image.
To train the linear probe we encoded the test set using the non-doubling vnca and got a dataset of cell latent vectors of shape $10,240,000 \times |Z|$ where the width and height dimensions have been merged ($10,240,000 = 32 \cdot 32 \cdot 10,000$). We then trained a one layer linear model on 80% of this dataset to predict the label of the whole original image from which the individual cell latent vectors came from. The linear probe classification performance was then evaluated on the rest 20% of the dataset.
The code can be found at github.com/albertaillet/vnca.

3.5 Computational requirements

The original paper's code implementation was run on some type of GPU, for which the exact specifications are not presented in [1]. However, these details are not crucial for the reproduction of the results. Our re-implementation was run in a single program, multiple data (SPMD) configuration over v3-8 TPU cores provided by Kaggle for a total of 4h using the stated hyperparameters. For the baseline model, the training time was 10 minutes and the inference of a single image was 14.6 ms. Calculating the ELBO with 128 importance-weighted samples took 1.45 seconds in total for the whole test dataset of $10,000$ images. The training time of the doubling VNCA was 40 minutes, inference time 18.7 ms, and test time 162 seconds. Finally, for the non-doubling VNCA it took 2.5 hours to train, inference time 12.3 ms, and test time 75 seconds.

4 Results

The results was overall very similar to the experimental results in the original paper and therefore support its claims. For instance, the replicated ELBO was within 1.8% of the paper's stated value. This was however not the case for the t-SNE experiment where our results differ from the original study.

4.1 Results reproducing original paper

Result 1 — The convolutional baseline achieved $\log p(x) \geq -84.64$ nats evaluated with 128 importance-weighted samples on the entire test set, and cannot be compared with the original experiment since it is not presented in the paper. For the doubling VNCA, it achieved $\log p(x) \geq -84.15$ nats compared with the $\log p(x) \geq -84.23$ from the original paper. The non-doubling VNCA achieved $\log p(x) \geq -89.3$ nats compared with the $\log p(x) \geq -90.97$. These results are very similar to those from the original paper and therefore support the first claim.

Result 2 — Figure 2 presents test set reconstructions and unconditional samples from the prior $\mathcal{N}(0, I)$. In figure 3 a visualization of the growing process in the NCA decoder of the doubling VNCA is presented.

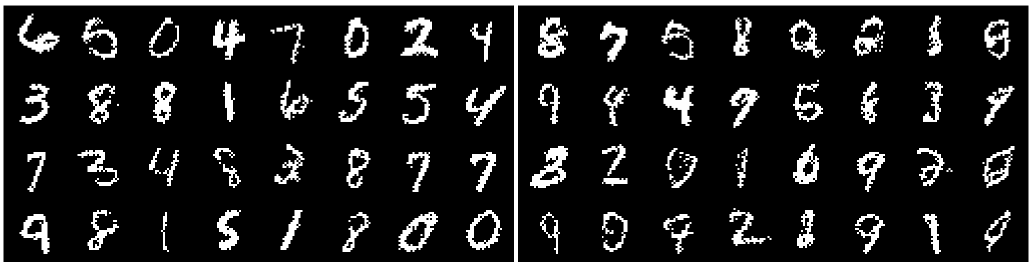


Figure 2. **Left:** Test set reconstructions. **Right:** Unconditional samples from the prior $\mathcal{N}(0, I)$. **Note:** This figure shows samples and not averages.

These results are visually similar to the results of the original paper and therefore support its claims.

Result 3 — In Figure 4 an exploration of the latent space of the doubling VNCA is presented. This exploration includes linear interpolations between samples from the prior and the t-SNE reduction of $5,000$ encoded test set digits.

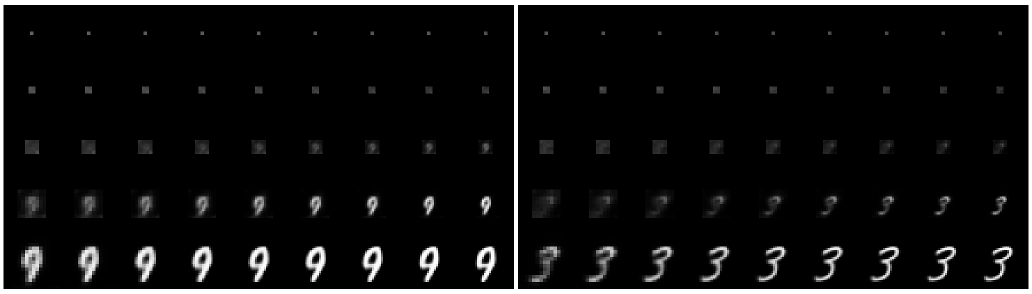


Figure 3. Visualization of the states in the NCA decoder. The first channel of z_t is shown after each doubling or NCA step. Doubling is performed on the first column. Both figures show unconditional samples from the prior $\mathcal{N}(0, I)$. **Note:** This figure shows averages.

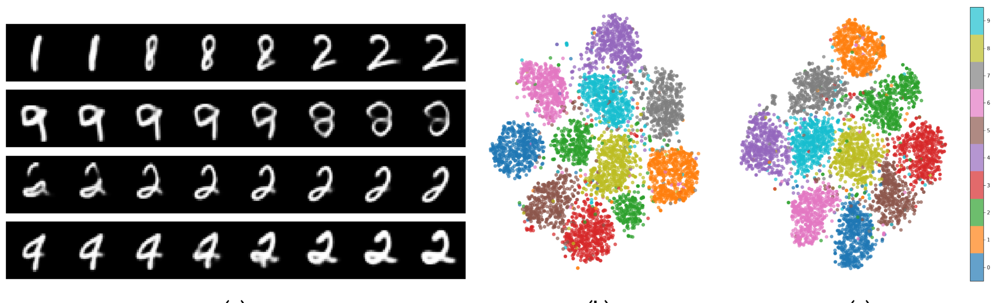


Figure 4. Latent space exploration of doubling VNCA. Figure **4a** shows decoded versions of linear interpolations between samples from the prior $\mathcal{N}(0, I)$. Figure **4b** and **4c** show the t-SNE reduction of $5,000$ encoded images from the test. Figure **4b** uses the doubling VNCA encoder and **4c** uses the encoder of the convolutional baseline. **Note:** This figure shows averages.

The obtained linear interpolation is visually similar to that of the original paper. The fully convolution baseline t-SNE structure does however look different. Both models appear to have a similar t-SNE structure, none of them appearing fragmented as in the original paper, meaning that the second identified claim is not supported by our results.

Result 4 — In Figure 5 the damage recovery properties of the non-doubling VNCA are presented.
These results are visually similar to the original paper's results and therefore support the third identified claim that damage recovery arises from the training process of the non-doubling VNCA.

4.2 Results beyond original paper

The final accuracy of the linear probe was 80.37% on the test set when predicting the whole image label from the latent vector of each cell. In figure 6 the results of classifying individual cell latent vectors decoded from samples of the prior are shown. For the digits 4, 5 and 1 almost all the cell vectors are classified correctly, with high certainty. For the digits 6 and 0, the cell vectors in the part of the image where the digit appears are correctly classified with high certainty while the parts of the image without any digit are often incorrectly classified. Finally, the digit 9 and the fifth digit from the right are given very different classes throughout the image. This could be explained by the fact that these unconditional samples from the prior seem out of the distribution of those present in the generated dataset.

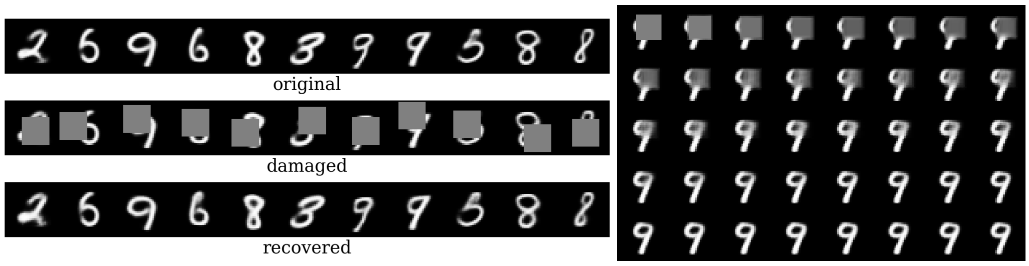


Figure 5. Damage recovery process for non-doubling VNCA. **Left**: **original**: Samples after $T = 40$ NCA steps. **damaged**: Damage applied to the samples. **recovered**: Recovered samples after 40 additional NCA steps. **Right**: Damage recovery process. **Note:** This figure shows averages.

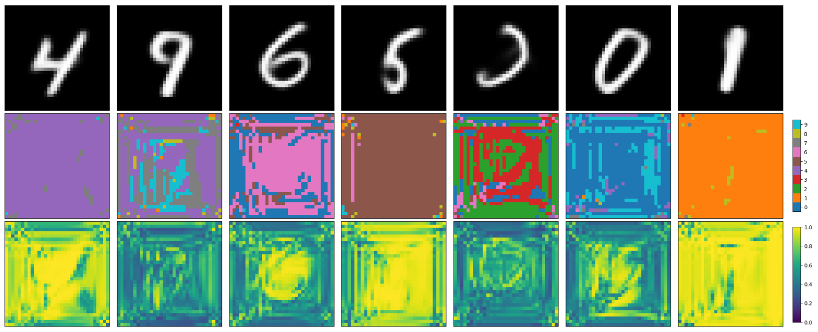


Figure 6. Classification using linear probe trained on individual cell latent vectors. **Top**: First channel of cell latent vector decoded from an unconditional sample of the prior. **Middle**: Top-1 predicted class of each of these cell latent vectors. **Bottom**: Softmax value of top-1 predicted class. **Note:** This figure shows averages.

5 Discussion

From our first two experiments, we find that the paper's results can successfully be replicated. It however has to be noted that these results concern the binarized MNIST dataset, which is relatively simple and these results might not hold for more complex datasets. The study also utilized a substantial latent space size, leaving room for further exploration of the impact on performance when varying the latent space and the number of parameters in the update model.

It was not possible to replicate the experimental results supporting the claim of the doubling VNCA model having a simpler t-SNE structure than the convolutional baseline. As the parameters used for the t-SNE are not present in the original paper, we tried multiple different parameters and random subsets of the test data. We were however not able to reproduce a t-SNE structure similar to the one presented in the original paper. We obtained a very similar t-SNE structure for all three models. The fact that we found a t-SNE reduction of the baseline latent space with the same clean separation shows that the second identified claim of the original paper was not supported.

From the third result, it is apparent the model is capable of recovering from damage during the generation process. This result is visually similar to the one in the original paper and supports the identified third claim.
Our additional experiment was conducted to investigate the representational power of the individual cell latent vectors. From the high accuracy of the linear classifying probe we can conclude that the cell latent vectors encode a lot of information about the whole

image which could be an explanation for the good reconstruction.

5.1 What was easy

Re-implementing the proposed models was easy since the original paper had a detailed description and the exact model specifications were provided as a PyTorch module in the appendix of the paper. The paper also had multiple diagrams, explanations and pseudocode for the used methods to facilitate understanding.

5.2 What was difficult

As mentioned in Section 3.2, the binarized MNIST dataset is provided without labels and is in a different ordering than the original MNIST dataset. It was therefore unclear how to get the dataset labels for the t-SNE latent space visualization in Figure 4 as it is not mentioned in the paper [1]. It is however known that each image in the binarized MNIST is derived from the original MNIST by stochastically setting each pixel value to 1 in proportion to its intensity [2]. We could therefore estimate the source image by calculating the probability of each binarized image given the probability distribution of each original image and choose the original image j to maximize $p(x_i^{\text{binary}}|x_j^{\text{original}})$. As the original MNIST has labels, each image in the binarized MNIST could then be associated with the label of the estimated source image.

A challenge when implementing the non-doubling version was making it work with a distributed training setup. This was due to the setup using a pool that had to be shared on each TPU core. In order to facilitate random shuffling of this shared buffer, all the TPU's local pools had to be combined at each step (gather-all operation), shuffled, and then returned. Using only a local permutation of the pool on each TPU was also attempted and yielded similar results with faster training time. However, due to this being a replication study, the method with global pooling was used as it was equivalent to the one proposed in the original paper.

5.3 Communication with original authors

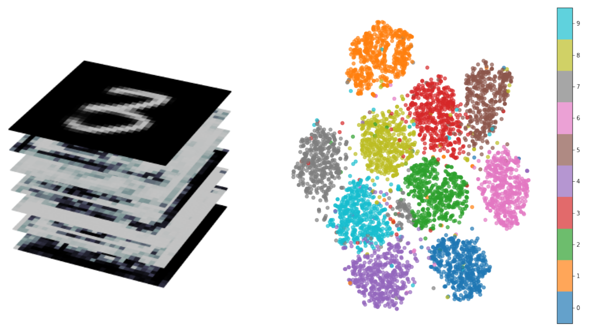
There was no communication with the original authors.

References

1. R. B. Palm, M. G. Duque, S. Sudhakaran, and S. Risi. "Variational Neural Cellular Automata." In: **International Conference on Learning Representations**. https://openreview.net/forum?id=7fFO4cMBx_9. 2022.
2. R. Salakhutdinov and I. Murray. "On the Quantitative Analysis of Deep Belief Networks." In: **Proceedings of the 25th International Conference on Machine Learning**. ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 872–879. DOI: 10.1145/1390156.1390266. URL: https://doi.org/10.1145/1390156.1390266.
3. R. B. Palm. https://github.com/rasmusbergpalm/vnca. 2022.
4. J. Bradbury et al. **JAX: composable transformations of Python+NumPy programs**. https://github.com/google/jax. Version 0.3.13. 2018.
5. P. Kidger and C. Garcia. "Equinox: neural networks in JAX via callable PyTrees and filtered transformations." In: **Differentiable Programming workshop at Neural Information Processing Systems 2021** (2021). https://github.com/patrick-kidger/equinox.
6. L. Deng. "The mnist database of handwritten digit images for machine learning research." In: **IEEE Signal Processing Magazine** 29.6 (2012), pp. 141–142.
7. D. P. Kingma and J. Ba. **Adam: A Method for Stochastic Optimization**. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

8. R. Pascanu, T. Mikolov, and Y. Bengio. "On the difficulty of training recurrent neural networks." In: **Proceedings of the 30th International Conference on Machine Learning**. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1310–1318. URL: https://proceedings.mlr.press/v28/pascanu13.html.
9. L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE." In: **Journal of Machine Learning Research** 9 (2008), pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.
10. G. Alain and Y. Bengio. **Understanding intermediate layers using linear classifier probes**. 2017. URL: https://openreview.net/forum?id=ryF7rTqgl.

Appendix



(a) (b)

Figure 7. Figure **7a** Visualization of the latent vectors of the VNCA. The figure shows slices with the values of the seven first channels of the latent vectors. The first channel of the latent vector is trained to be the logits for a Bernoulli distribution. Figure **7b** t-SNE reduction of 5000 encoded images from the test set using the encoder from the non-doubling VNCA.

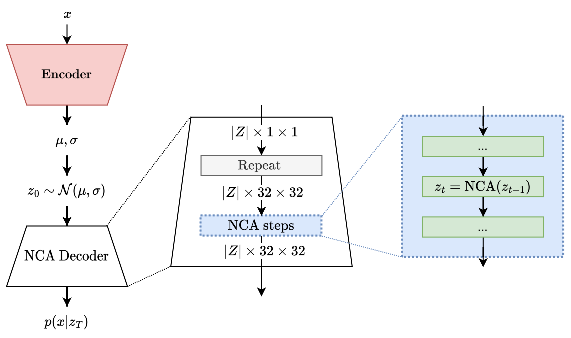

x
Encoder
μ, σ
$z_0 \sim \mathcal{N}(\mu, \sigma)$
NCA Decoder
$p(x|z_T)$
$|Z| \times 1 \times 1$
Repeat
$|Z| \times 32 \times 32$
NCA steps
$|Z| \times 32 \times 32$
...
$z_t = \text{NCA}(z_{t-1})$
...

Figure 8. Overview of the non-doubling VNCA model. **Left**: Whole reconstruction process, input image x is encoded using the convolutional encoder and z_0 is sampled from the distribution $\mathcal{N}(\mu, \sigma)$ and then fed into the NCA decoder. **Middle**: The decoder consists of repeating the initial sample over the grid followed by T NCA steps.