# Parallelizing Autoregressive Generation with Variational State Space Models

**Gaspard Lambrechts*** GASPARD.LAMBRECHTS@ULIEGE.BE
**Yann Claes*** Y.CLAES@ULIEGE.BE
**Pierre Geurts** P.GEURTS@ULIEGE.BE
**Damien Ernst** DERNST@ULIEGE.BE
*Montefiore Institute, University of Liège*

## Abstract

Attention-based models such as Transformers and recurrent models like state space models (SSMs) have emerged as successful methods for autoregressive sequence modeling. Although both enable parallel training, none enable parallel generation due to their autoregressiveness. We propose the variational SSM (VSSM), a variational autoencoder (VAE) where both the encoder and decoder are SSMs. Since sampling the latent variables and decoding them with the SSM can be parallelized, both training and generation can be conducted in parallel. Moreover, the decoder recurrence allows generation to be resumed without reprocessing the whole sequence. Finally, we propose the autoregressive VSSM that can be conditioned on a partial realization of the sequence, as is common in language generation tasks. Interestingly, the autoregressive VSSM still enables parallel generation. We highlight on toy problems (MNIST, CIFAR) the empirical gains in speed-up and show that it competes with traditional models in terms of generation quality (Transformer, Mamba SSM).

**Keywords:** Parallel, Autoregressive, Generation, VAE, SSM, VSSM

## 1. Introduction

Sequence modeling tasks, namely time-series forecasting and text generation, have gained in popularity and various types of architectures were designed to tackle such problems. Transformers were proven effective [17, 19], yet they nonetheless reprocess the complete sequence at each timestep, making generation less efficient. Recurrent neural networks (RNNs) [3, 8] update a hidden state based on new inputs at each timestep, enabling efficient generation. SSMs [9–11, 18], a recently introduced class of RNNs, enable parallel training thanks to their linear recurrence. Alternatively, several works adapt VAEs for sequential modeling. Some architectures integrate Transformers [13, 14] and enable parallel training, although little work [5] proposes models that can be conditioned on partial realizations (e.g., prompts). Conversely, variational RNNs (VRNNs) [4] loose parallelizability by making the model both autoregressive and recurrent, allowing it to be conditioned on partial realizations and to resume generation. However, all introduced autoregressive models perform generation sequentially, as they are explicitly conditioned on previously generated data.

Therefore, we propose the VSSM, a VAE whose encoder and decoder are SSMs. Thanks to key architectural choices, both training and inference can be performed in parallel and linear time with

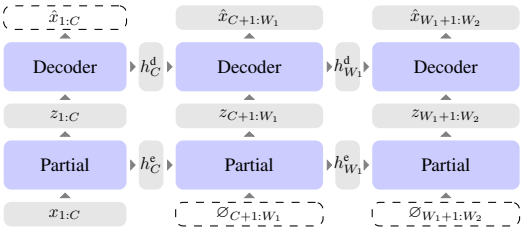
*Equal contributions.

© G. Lambrechts, Y. Claes, P. Geurts & D. Ernst.

respect to the sequence length, while still allowing generation to be to resumed without reprocessing the entire sequence. In contrast, a VAE with Transformer encoder and decoder, which we call Transformer VAE (TVAE), would preserve parallel training and generation, but would not be resumable. We then propose the autoregressive VSSM, that can be conditioned on partial realizations of the sequence and still generates in parallel. The VSSM combines all advantages of previous models, as observed in Figure 1a, while producing results comparable to Transformers and SSMs on simple tasks (MNIST, CIFAR). We highlight a recent work [20] that proposes a similar architecture, yet their prior and generative models are explicitly autoregressive and do not exploit the parallelizability of SSMs. Moreover, they only consider generation from sampled latents, while we also propose an approach to condition the model on partial realizations. We do not consider diffusion models for sequences (e.g., [7]), but note that they would not allow recurrent (i.e., resuming) generation.

| Model | Training | // | Sampling | // | Prompt | Resume |
|---|---|---|---|---|---|---|
| Transformer | $O(T^2)$ | ✓ | $O(T^2)$ | ✗ | ✓ | ✗ |
| RNN | $O(T)$ | ✗ | $O(T)$ | ✗ | ✓ | ✓ |
| SSM | $O(T)$ | ✓ | $O(T)$ | ✗ | ✓ | ✓ |
| TVAE | $O(T^2)$ | ✓ | $O(T^2)$ | ✓ | ✗/✓ | ✗ |
| VRNN | $O(T)$ | ✗ | $O(T)$ | ✗ | ✓ | ✓ |
| VSSM | $O(T)$ | ✓ | $O(T)$ | ✓ | ✓ | ✓ |

(*a*) Time complexities and parallelizability at training and sampling, and generation properties.




(*b*) Parallel and recurrent sampling algorithm, given a contextual prompt $x_{1:C}$.

Figure 1: Sequence models properties and VSSM sampling algorithm.

## 2. Background

### 2.1. Variational Autoencoders for Time Series

We consider dynamical VAEs [6], that model sequential data $x_{1:T}$ of length $T$ through $T$ latent variables $z_{1:T}$. Given a target space $\mathcal{X}$, they define the joint distribution $p_\phi(x_{1:T}, z_{1:T})$ with,

- A latent space $\mathcal{Z}$,
- A prior distribution $p_\phi(z_{1:T}) = \prod_{t=1}^{T} p_\phi(z_t|z_{1:t-1})$,
- A generative distribution $p_\phi(x_{1:T}|z_{1:T}) = \prod_{t=1}^{T} p_\phi(x_t|x_{1:t-1}, z_{1:T})$,

where $\phi$ denotes the parameters of these probability distributions. Unfortunately, the likelihood of the data $p_\phi(x_{1:T}) = \mathbb{E}_{p_\phi(z_{1:T})} p_\phi(x_{1:T}|z_{1:T})$ under this model cannot be evaluated in practice. Nevertheless, we can show that the log-likelihood is lower bounded by the evidence lower bound (ELBO), for any conditional probability distribution $q(z_{1:T}|x_{1:T})$,

$$\log p_\phi(x_{1:T}) \geq \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log p_\phi(x_{1:T}|z_{1:T}) - \mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T})) = \mathrm{ELBO}_\phi(x_{1:T}) \quad (1)$$

Moreover, the ELBO becomes tight when $q(z_{1:T}|x_{1:T})$ corresponds to the true posterior distribution $p_\phi(z_{1:T}|x_{1:T})$. Thus, the generative model $p_\phi$ is usually jointly optimized with,

- A posterior distribution $q_\psi(z_{1:T}|x_{1:T}) = \prod_{t=1}^{T} q_\psi(z_t|z_{1:t-1}, x_{1:T})$,

where $\psi$ denotes the parameters of this distribution. These four components compose the dynamical VAE. More details are provided in Section A.

### 2.2. State Space Models

SSMs are linear and time-invariant dynamical systems that can be discretized into $h_t = Ah_{t-1} + Bu_t$, where $\zeta = (A, B)$ are learnable parameters. Using the prefix-sum algorithm [1], we can parallelize the computation of the state sequence $h_t = \text{SSM}_\zeta(u_{1:t})$ along all timesteps $t \in [1, T]$. Furthermore, we can obtain effective sequence models of the form $y_t = f_\theta(u_{1:t})$ by stacking $L$ layers $i = \{1, \dots, L\}$ of interleaved SSMs and timestep-wise feedforward neural networks (FNNs),

$$h_t^i = \text{SSM}_{\zeta_i}(u_{1:t}^{i-1}), \qquad y_t^i = \text{FNN}_{\xi_i}(h_t^i), \tag{2}$$

where $u_t^i = y_t^{i-1}$, $u_t^0 = u_t$, $y_t = y_t^L$, and $\theta = \cup_{i=1}^L(\zeta_i, \xi_i)$ includes all SSMs and FNNs parameters. Indeed, it is believed that such stacking of SSMs and timestep-wise FNNs is a universal approximator of sufficiently regular non-linear sequence-to-sequence maps [16].

## 3. Method

### 3.1. Variational State Space Model

We introduce the VSSM as an instance of dynamical VAE, where we select, given a target space $\mathcal{X}$,

- A discrete latent space $\mathcal{Z} = \{1, \dots, N\}^Z$ of $Z$ components of cardinality $N$ each,
- A uniform prior distribution $p_\phi(z_{1:T}) = \prod_{t=1}^T p_\phi(z_t|z_{1:t-1}) = \prod_{t=1}^T p_\phi(z_t) = \prod_{t=1}^T \frac{1}{N^Z}$,
- A generative distribution $p_\phi(x_{1:T}|z_{1:T}) = \prod_{t=1}^T p_\phi(x_t|z_{1:t}) = \prod_{t=1}^T \mathcal{P}(x_t|f_\phi^{\text{dec}}(z_{1:t}))$, where $\mathcal{P}(x_t|w_t)$[1] is a distribution of parameters $w_t = f_\phi^{\text{dec}}(z_{1:t})$ outputted by a stacked SSM,
- A posterior distribution $q_\psi(z_{1:T}|x_{1:T}) = \prod_{t=1}^T q_\psi(z_t|x_{1:t}) = \prod_{t=1}^T \mathcal{D}(z_t|f_\psi^{\text{enc}}(x_{1:t}))$, where $\mathcal{D}(z_t|v_t)$ is a discrete distribution of probabilities $v_t = f_\psi^{\text{enc}}(x_{1:t})$ outputted by a stacked SSM.

The independence of the prior over all timesteps $z_t$, along with the conditional independence between $z_{\neq t}$ and $z_t$ given $x_{1:t}$ in $q_\psi$, and between $x_{\neq t}$ and $x_t$ given $z_{1:T}$ in $p_\phi$ enables the prior, posterior and generative models to be sampled in parallel. Note that the discrete latent space requires the Gumbel reparametrization trick for computing $\nabla_\psi z_{1:T}$ when maximizing the ELBO [12, 15].

### 3.2. Autoregressive Variational State Space Model

In some applications, (e.g., language modeling) it is useful to learn a generative model of the distribution $p(x_{1:T}|x_{1:C})$ conditioned on a partial realization $x_{1:C}$. Under the modeling assumptions of a trained dynamical VAE like the VSSM prior and generative models of Section 3.1, we have,

$$p_\phi(x_{1:T}|x_{1:C}) = \int_{\mathcal{Z}^T} p_\phi(x_{1:T}|z_{1:T}) p_\phi(z_{1:T}|x_{1:C}) \, \mathrm{d}z_{1:T}, \tag{3}$$

where $p_\phi(x_{1:T}|z_{1:T})$ is our generative model, while $p_\phi(z_{1:T}|x_{1:C})$ is the true partial posterior, from which we cannot sample for a given $x_{1:C}$. We thus propose to learn an approximate partial posterior $q_\omega(z_{1:T}|x_{1:C})$ of the true partial posterior $p_\phi(z_{1:T}|x_{1:C})$, by exploiting samples $p(x_{1:T}|x_{1:C})$ from the dataset to construct samples of $p_\phi(z_{1:T}|x_{1:C})$ (see details Section A.2).

The partial posterior $q_\omega(z_{1:T}|x_{1:C})$ is implemented with a stacked SSM, where the input $x_{1:C}$ is padded with empty tokens: $\bar{x}_{1:T} = (x_{1:C}, \varnothing, \dots, \varnothing)$. The autoregressive VSSM is a VSSM with,

[1] Gaussian of mean $w_t$ and fixed variance for continuous $\mathcal{X}$ or discrete distribution of probabilities $w_t$ for discrete $\mathcal{X}$.

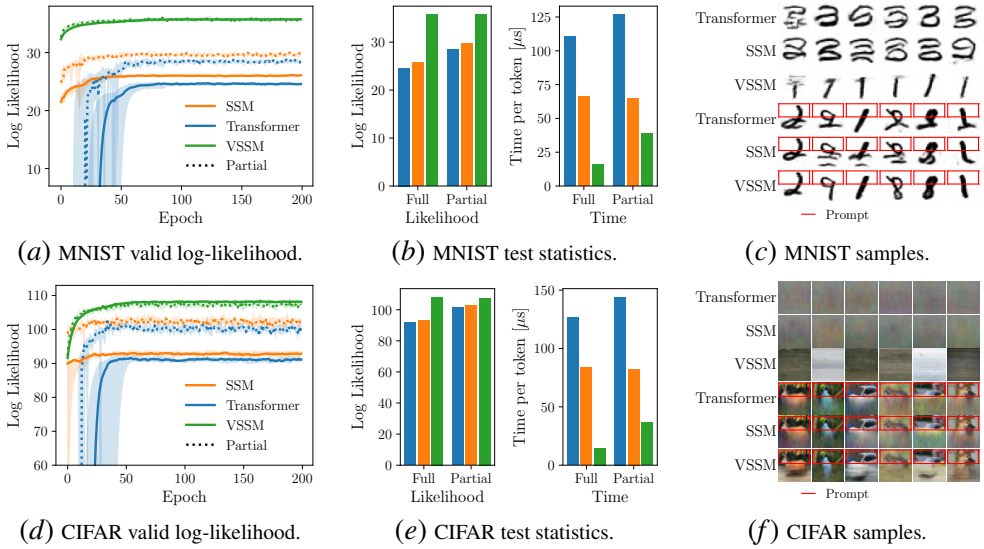
- A partial posterior distribution $\mathcal{D}(z_t|\bar{v}_t)$, where probabilities $\bar{v}_t = f_\omega^{\text{par}}(\bar{x}_{1:T})$ are the output of a stacked SSM, such that $q_\omega(z_{1:T}|x_{1:C}) = \prod_{t=1}^{T} q_\omega(z_t|x_{1:\min(C,t)}) = \prod_{t=1}^{T} \mathcal{D}(z_t|f_\omega^{\text{par}}(\bar{x}_{1:T}))$.

Note that the partial posterior distribution $q_\omega$ should ideally correspond to the prior when $\bar{x}_{1:T} = (\varnothing, \dots, \varnothing)$, and it will be used in practice for unconditional generation.

The autoregressive VSSM generates in parallel, possibly conditioned on a partial realization, and can resume generation, as illustrated on Figure 1 (see detailed algorithms comparison in Section B).

## 4. Experiments

In the following, we compare Transformer, SSM and VSSM on two toy sequence modeling tasks: MNIST, for which we consider 28-dimensional sequences of length 28, and CIFAR, for which we consider $(32 \times 3)$-dimensional sequences of length 32. Transformer and SSM both output the mean of a Gaussian distribution of fixed variance. For more details about model architectures, see Section C.1. We report samples, generation times and likelihoods in Figure 2, estimated by importance-sampling for the VSSM, see Section C.2. We report additional results in Section C.3.




(a) MNIST valid log-likelihood. (b) MNIST test statistics. (c) MNIST samples.

(d) CIFAR valid log-likelihood. (e) CIFAR test statistics. (f) CIFAR samples.

Figure 2: We report results over 5 runs of each model. Confidence intervals correspond to the minimum and maximum values observed. In 2a, 2d, we plot the median full and partial log-likelihood $\log p_\phi(x_{1:T})$ and $\log p_\phi(x_{C+1:T} \mid x_{1:C})$ on the validation set throughout training. In 2b, 2e, we report the average full and partial log-likelihood on the test set, along with mean execution times at generation, in both cases. In 2c, 2f, we report random qualitative examples for all models, for unconditioned sampling (first three rows) and conditioned on partial realizations (last three rows).

## 5. Conclusion

We introduce the VSSM, a dynamical VAE using SSMs as encoder and decoder. Compared to other architectures, our model is the first one that can generate in parallel while being recurrent, which allows generation to be resumed. Although tested on simple tasks, we show that it produces decent results in only a fraction of the time. The advantages of this architecture motivate further work to scale and improve performance on more challenging tasks such as language generation.

## Acknowledgments

Gaspard Lambrechts gratefully acknowledges the financial support of the *Wallonia-Brussels Federation* for his FRIA grant. Yann Claes gratefully acknowledges the financial support of the *Walloon Region* under Grant No. 2010235 (ARIAC by Digital Wallonia 4.AI). Computational resources have been provided by the *Consortium des Équipements de Calcul Intensif* (CÉCI), funded by the *National Fund for Scientific Research* (F.R.S.-FNRS) under Grant No. 2502011 and by the *Walloon Region*, including the Tier-1 supercomputer of the *Wallonia-Brussels Federation*, infrastructure funded by the *Walloon Region* under Grant No. 1117545.

## References

[1] Guy E Blelloch. Prefix Sums and Their Applications. *School of Computer Science, Carnegie Mellon University Pittsburgh, PA, USA*, 1990.

[2] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[3] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[4] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. In *Advances in Neural Information Processing Systems*, 2015.

[5] Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. Transformer-Based Conditional Variational Autoencoder for Controllable Story Generation. *arXiv preprint arXiv:2101.00828*, 2021.

[6] Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical Variational Autoencoders: A Comprehensive Review. *Foundations and Trends in Machine Learning*, 2021.

[7] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. In *International Conference on Learning Representations*, 2023.

[8] Alex Graves. Generating Sequences with Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*, 2013.

[9] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[10] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the Parameterization and Initialization of Diagonal State Space Models. In *Advances in Neural Information Processing Systems*, 2022.

[11] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal State Spaces are as Effective as Structured State Spaces. In *Advances in Neural Information Processing Systems*, 2022.

[12] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017.

[13] Junyan Jiang, Gus G Xia, Dave B Carlton, Chris N Anderson, and Ryan H Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[14] Danyang Liu and Gongshen Liu. A Transformer-Based Variational Autoencoder for Sentence Generation. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.

[15] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2016.

[16] Antonio Orvieto, Soham De, Caglar Gulcehre, Razvan Pascanu, and Samuel L Smith. On the Universality of Linear Recurrences Followed by Nonlinear Projections. *arXiv e-prints*, 2023.

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 2019.

[18] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified State Space Layers for Sequence Modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

[20] Linqi Zhou, Michael Poli, Winnie Xu, Stefano Massaroli, and Stefano Ermon. Deep Latent State Space Models for Time-Series Generation. In *International Conference on Machine Learning*, 2023.

## Appendix A. Mathematical derivations

### A.1. Learning objective

Thanks to Jensen's inequality, we can show for dynamical VAEs of Section 2.1 that,

$$\log p_\phi(x_{1:T}) = \log \mathop{\mathbb{E}}_{p_\phi(z_{1:T})} p_\phi(x_{1:T}|z_{1:T}) \frac{q(z_{1:T}|x_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{4}$$

$$= \log \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \frac{p_\phi(x_{1:T}|z_{1:T})p_\phi(z_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{5}$$

$$\geq \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log \frac{p_\phi(x_{1:T}|z_{1:T})p_\phi(z_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{6}$$

$$\geq \underbrace{\mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log p_\phi(x_{1:T}|z_{1:T}) - \mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T}))}_{\mathrm{ELBO}_\phi(x_{1:T})}, \tag{7}$$

Note that the ELBO becomes tight when the inference model $q(z_{1:T}|x_{1:T})$ corresponds to the true posterior distribution $p_\phi(z_{1:T}|x_{1:T})$. Indeed,

$$\mathrm{ELBO}_\phi(x_{1:T}) = \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log \frac{p_\phi(z_{1:T}|x_{1:T})p(x_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{8}$$

$$= \log p(x_{1:T}) - \mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T}|x_{1:T})), \tag{9}$$

and $\mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T}|x_{1:T})) = 0$ if and only if $q(z_{1:T}|x_{1:T}) = p_\phi(z_{1:T}|x_{1:T})$ almost everywhere. Hence, the dynamical VAE, composed of the prior $p_\phi(z_{1:T})$, generative model $p_\phi(x_{1:T}|z_{1:T})$ and inference model $q_\psi(z_{1:T}|x_{1:T})$ can be trained according to the objective function,

$$\max_{\phi,\psi} \mathop{\mathbb{E}}_{p(x_{1:T})} \left[ \mathop{\mathbb{E}}_{q_\psi(z_{1:T}|x_{1:T})} [\log p_\phi(x_{1:T}|z_{1:T})] - \mathrm{KL}(q_\psi(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T})) \right]. \tag{10}$$

### A.2. Approximate partial posterior

To learn the approximate partial posterior $q_\omega(z_{1:T}|x_{1:C})$ of the true partial posterior $p_\phi(z_{1:T}|x_{1:C})$ introduced in Section 3.2, we propose to exploit samples of the true partial posterior. Such samples are derived from samples $(x_{1:C}, x_{1:T})$, constructed from the dataset of sequences $x_{1:T}$ by taking random cuts $C \sim \mathcal{U}([0, T])$. Indeed, these allow us to draw samples $(x_{1:C}, z_{1:T})$ such that $z_{1:T} \sim p_\phi(z_{1:T}|x_{1:C})$, as suggested by the decomposition,

$$p_\phi(z_{1:T}|x_{1:C}) = \int_{x_{1:T}} p_\phi(z_{1:T}|x_{1:T}) p(x_{1:T}|x_{1:C}) \, \mathrm{d}x_{1:T}, \tag{11}$$

where $p_\phi(z_{1:T}|x_{1:T})$ is the true posterior of this VAE, which we approximate by the variational posterior $q_\psi(z_{1:T}|x_{1:T})$ during the VSSM training. The training objective for the approximate partial posterior $q_w(z_{1:T}|x_{1:C})$ is,

$$\arg\min_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathrm{KL}(p_\phi(z_{1:T}|x_{1:C}) \parallel q_\omega(z_{1:T}|x_{1:C})) \tag{12}$$

$$= \arg\min_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p_\phi(z_{1:T}|x_{1:C})} [\log p_\phi(z_{1:T}|x_{1:C}) - \log q_\omega(z_{1:T}|x_{1:C})] \tag{13}$$

$$= \arg\max_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p_\phi(z_{1:T}|x_{1:C})} [\log q_\omega(z_{1:T}|x_{1:C})] \tag{14}$$

$$= \arg\max_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p(x_{1:T}|x_{1:C})} [\mathop{\mathbb{E}}_{p_\phi(z_{1:T}|x_{1:T})} [\log q_\omega(z_{1:T}|x_{1:C})]] \tag{15}$$

$$\approx \arg\max_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p(x_{1:T}|x_{1:C})} [\mathop{\mathbb{E}}_{q_\psi(z_{1:T}|x_{1:T})} [\log q_\omega(z_{1:T}|x_{1:C})]] \tag{16}$$

## Appendix B. Comparison of Autoregressive Generation

The VSSM sampling algorithm (Algorithm 1) can be compared to the RNN (Algorithm 2), SSM (Algorithm 3), and Transformer (Algorithm 4) algorithms. We also provide an algorithm for the chunk sampling method proposed in Section 3.2 in Algorithm 5.

**Algorithm 1:** VSSM sampling algorithm.

**input:** *Prompt* $x_{1:C}$*, length* $T$.
Let $\bar{v}_{1:T} = f_\omega^{\text{par}}(k)$.
Sample $z_t \sim \mathcal{D}(z_t|\bar{v}_t),\ t = 1, \dots, T$.
Let $w_{1:T} = f_\phi^{\text{dec}}(z_{1:T})$.
Sample $x_t \sim \mathcal{P}(x_t|w_t),\ t = C+1, \dots, T$.
**return** $x_{1:T}$

**Algorithm 2:** RNN sampling algorithm.

**input:** *Prompt* $x_{1:C}$*, length* $T$.
Let $h_0 \leftarrow 0$.
**for** $t \leftarrow 1, \dots, C$ **do**
  Let $h_t = f_\phi(x_t, h_{t-1})$.
**end**
**for** $t \leftarrow C+1, \dots, T$ **do**
  Let $w_t = g_\phi(h_{t-1})$.
  Sample $x_t \sim \mathcal{D}(x_t|w_t)$.
  Let $h_t = f_\phi(x_t, h_{t-1})$.
**end**
**return** $x_{1:T}$

## Appendix C. Additional details on experiments

### C.1. Training details

We train all three architectures (Transformer, SSM, VSSM) with comparable sizes for 200 epochs on the classical train set of the considered benchmarks (MNIST, CIFAR). To prevent overfitting, we use $10\%$ of the train set for computing validation losses and likelihoods, as well as to select the final set

**Algorithm 3:** SSM sampling algorithm.

**input:** *Prompt* $x_{1:C}$*, length* $T$.
Let $h_0 \leftarrow 0$.
Let $h_C = f_\phi(x_{1:C}, h_0)$.
**for** $t \leftarrow C+1, \ldots, T$ **do**
  Let $w_t = g_\phi(h_{t-1})$.
  Sample $x_t \sim \mathcal{D}(x_t|w_t)$.
  Let $h_t = f_\phi(x_t, h_{t-1})$.
**end**
**return** $x_{1:T}$

**Algorithm 4:** Transformer sampling algorithm.

**input:** *Prompt* $x_{1:C}$*, length* $T$.
**for** $t \leftarrow C+1, \ldots, T$ **do**
  Let $w_t = f_\phi(x_{1:t-1})$.
  Sample $x_t \sim \mathcal{D}(x_t|w_t)$.
**end**
**return** $x_{1:T}$

**Algorithm 5:** VSSM chunk sampling algorithm.

**input:** *Prompt* $x_{1:C}$*, length* $T$*, chunk size* $W \in [1, T-C]$.
Let $(\bar{v}_{1:C}, h_C^{\text{par}}) = f_\omega^{\text{par}}(x_{1:C})$.
Sample $z_t \sim \mathcal{D}(z_t|\bar{v}_t),\ t = 1, \ldots, C$.
Let $(w_{1:C}, h_C^{\text{dec}}) = f_\phi^{\text{dec}}(z_{1:C})$.
**while** $C \leq T$ **do**
  Let $(\bar{v}_{C+1:C+W}, h_{C+W}^{\text{par}}) = f_\omega^{\text{par}}(\varnothing_{C+1:C+W}, h_C^{\text{par}})$.
  Sample $z_t \sim \mathcal{D}(z_t|\bar{v}_t),\ t = C+1, \ldots, C+W$.
  Let $(w_{C+1:C+W}, h_{C+W}^{\text{dec}}) = f_\phi^{\text{dec}}(z_{C+1:C+W}, h_C^{\text{dec}})$.
  Sample $x_t \sim \mathcal{P}(x_t|w_t),\ t = C+1, \ldots, C+W$.
  Update $C \leftarrow C+W$.
**end**
**return** $x_{1:T}$

of weights for evaluation on the test set and generating samples. All three architectures use $4$ layers of dimension $1024$ (with a state size of $16$ for the SSM and VSSM, and with $8$ heads of dimension $1024/8 = 128$ for the Transformer). We follow the attention block of GPT-2 for the Transformer, and the SSM block of Mamba for the SSM and VSSM architectures. The SSM and Transformer architectures output the mean of a Gaussian distribution with fixed variance ($\sigma = 0.1$), and are trained to maximize the log-likelihood. The VSSM generative model $p_\phi$ also outputs the mean of a Gaussian distribution with fixed variance ($\sigma = 0.1$), and is trained along with the posterior $q_\psi$ to maximize the ELBO (7). Note that the temperature of the Gumbel softmax for computing $\nabla_\psi z_{1:T}$ was fixed to 1. The partial posterior $q_\omega$ is trained jointly with the encoder and decoder according to objective (16) and does not require to perform a subsequent training. All learning rates have been selected using a grid search in $(1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4})$.

### C.2. Evaluation

We evaluate the likelihood by sampling $K = 100$ latent variables [2] from the posterior, and reweighting by the prior (resp. partial posterior), in order to measure the likelihood (resp. the partial likelihood),

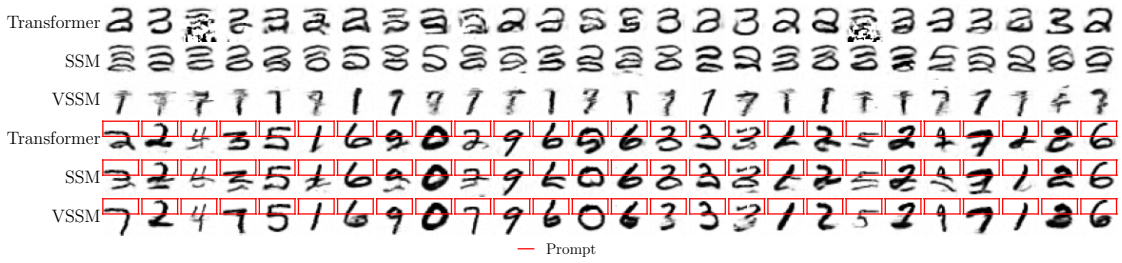
$$p_\phi(x_{1:T}) \approx \frac{1}{K} \sum_{k=1}^{K} p_\phi(x_{1:T}|z^k_{1:T}) \frac{p_\phi(z^k_{1:T})}{q_\psi(z^k_{1:T}|x_{1:T})}, \; z^k_{1:T} \sim q_\psi(z^k_{1:T}|x_{1:T}), \tag{17}$$

$$p_\phi(x_{C+1:T}|x_{1:C}) \approx \frac{1}{K} \sum_{k=1}^{K} p_\phi(x_{C+1:T}|z^k_{1:T}) \frac{q_\omega(z^k_{1:T}|x_{1:C})}{q_\psi(z^k_{1:T}|x_{1:T})}, \; z^k_{1:T} \sim q_\psi(z^k_{1:T}|x_{1:T}). \tag{18}$$

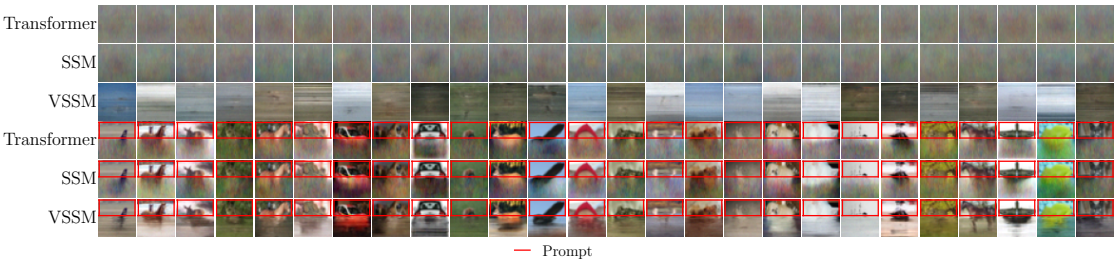
This expression is known to be a lower bound on the likelihood in expectation, and it tends towards the true likelihood as $K$ grows to infinity.

### C.3. Additional results

We report additional samples from all models in Figure 3a and Figure 3b, sampled randomly and using random prompts from the test set.




($a$) MNIST samples.




($b$) CIFAR samples.

Figure 3: We report random qualitative examples for all models, for unconditioned sampling (first three rows) and conditioned on partial realizations (last three rows).