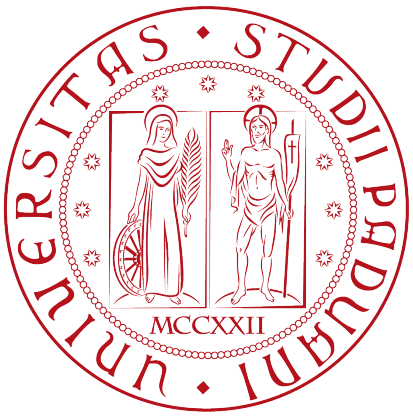

UNIVERSITY OF PADUA

DEPARTMENT OF PHYSICS AND ASTRONOMY "GALILEO GALILEI"

MASTER DEGREE IN PHYSICS OF DATA

Unsupervised generative modeling based on Markov chains out of equilibrium

Supervisor:
Prof. Marco Baiesi

Co-Supervisor:
Prof. Alberto Rosso

Candidate:
Marco Boscolo

Student ID:
2157559

ACADEMIC YEAR 2025/2026

“ Reality exceeds imagination. ”

— Anonymous

Acknowledgements

First of all, I would like to thank my academic supervisor, Prof. Baiesi, who has always been kind and available. Our conversations were genuinely helpful and valuable throughout this work.

I am deeply grateful to my family, especially my mother and father. A simple "thank you" is nowhere near enough; I owe them everything.

Finally, I want to acknowledge luck; something most of us should think about. None of us chooses when or where to be born. Considering the huge span of human history and all the possible places one could end up, being able to live in 21st-century Italy can be seen as a pretty rare and fortunate outcome. This is not a choice; it is simply luck.

As the opening of the movie *Match Point* reminds us: "The man who said -I would rather be lucky than good- saw deeply into life. People are afraid to face how great a part of life is dependent on luck. It is scary to think so much is out of one's control."

During the writing of this thesis, AI tools were used to improve the fluency of the English language. The responsibility for all claims and references lies with the author.

Padua, August 30, 2026

Marco Boscolo

Abstract

This thesis investigates steady-state Markov Machines (ssMMs), a class of unsupervised generative models where visible and hidden variables interact through independently parameterized transition matrices. Unlike Restricted Boltzmann Machines, ssMMs enable the system to operate intrinsically out of equilibrium and develop persistent latent-state cycles. The study, using MNIST as the dataset, provides the derivation of the model's log-likelihood gradient and analyzes its non-equilibrium dynamics, characterized by entropy production and probability currents. A central focus of this work is the one-hot encoded latent space, where we introduce specific strategies to prevent latent-state collapse and ensure systematic utilization of all hidden units. Our results show that operating far from equilibrium, together with other regularization techniques, allows the model to reproduce the empirical class distribution with excellent performance that stays consistent across independent training runs.

Contents

Introduction

This thesis focuses on a specific type of generative model that we refer to as steady-state Markov Machines (ssMMs). A generative model is an unsupervised model that aims to learn the underlying probability distribution of a dataset [1]. A generative model can be conceptually viewed as a "black box", internally composed of nodes called hidden units, which are connected to each other through parameters known as weights and biases, using linear or nonlinear functions. Training such a model typically involves two phases: the learning phase, where real inputs are provided and the model is trained to learn the best weights and biases; and the generative phase, in which the trained model runs on its own and produces new samples that should belong to the same distribution as the data, while still displaying some degree of generalization rather than simply reproducing what it has already seen.

The architecture closest to ssMMs, and one of the most widely used in machine learning models grounded in statistical mechanics, is the Restricted Boltzmann Machine (RBM) [2–4]. An RBM is a two-layer network made of visible and hidden units only, connected through a single weight matrix $\mathcal{W}$ that is shared between the two directions: the same $\mathcal{W}$ is used, transposed, both to infer the hidden units from the visible ones and to generate the visible units back from the hidden ones. This weight sharing is what drives the network toward a thermal equilibrium, described by a Boltzmann distribution. Moreover in RBM a detailed balance on the dynamics of the model is imposed, meaning that the probability flux from a visible state to a hidden state is equal to the reverse flux, and this constraint, in principle, restricts the latent space exploration capabilities of the network.

In practice, this equilibrium can lead to the situation where, once sampling is initialized near a given data point, the absence of persistent probability currents combined with a multimodal energy landscape tends to keep the chain trapped within the corresponding mode for many steps. This slow-mixing behavior, driven by free-energy barriers separating the learned configurations, has been well documented in the RBM literature [5, 6].

A first step away from the equilibrium restriction was the Helmholtz Machine [7], whose generative model samples from a free, factorized prior $p(\mathbf{z})$ at the top layer and propagates downward to the visible units, complemented by a separate set of bottom-up recognition weights used to approximate the posterior over the latent variables. With the decoupling of the forward and backward paths, the model can now break down the detailed balance condition. However, the generative process in Helmholtz Machines follows a one-way path;

starting from a prior distribution, the model produces data in a single pass, with no feedback loop running back and forth between the visible and latent units.

Baiesi and Rosso, inheriting the Helmholtz architecture, made a step further by introducing ssMM [8]: here the model keeps jumping back and forth between the visible and hidden layers through a Markov chain, and because the two transition matrices governing this chain are independent, the system settles into a steady state that is out of equilibrium.

At first glance, equilibrium dynamics seems the safer and more natural choice for a generative model, yet the ssMM architecture intentionally departs from this setting. Taking this concept one step further, one might even ask what happens if the model is explicitly encouraged to move even further away from equilibrium: this is a less conventional direction to pursue, and the empirical results are not something one could easily predict in advance.

While previous work has already compared RBMs and ssMMs [9], in this thesis we focus on studying and controlling the latent dynamics of the ssMM. In particular, we show that acting directly on the non-equilibrium cycles in the latent space improves generative performance: penalizing 2-step return cycles for individual latent states, alongside a regularization to ensure that all hidden units are utilized, consistently enhances the final log-likelihood $\mathcal{L}$. Accordingly, the dissertation is structured as follows:

- **Chapter 1** reviews how RBMs operate and why their dynamics remain confined to equilibrium.
- **Chapter 2** introduces the theoretical framework of the ssMM architecture alongside the derivation of its log-likelihood gradient.
- **Chapter 3** presents the regularization strategies adopted and discusses their motivation.
- **Chapter 4** reports the main results and the analysis carried out.

Chapter 1

Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) are among the simplest and most widely studied unsupervised generative models [10].

An RBM is a bipartite model with two sets of variables:

- **Visible layer:** Representing the observed data that we want to study, we usually refer to it as a D units vector $\mathbf{v} = (v_1, v_2, \ldots, v_D)$ (e.g., image pixels).
- **Hidden layer:** Where the latent structure and features of the raw inputs are camptured, conventionally to represent it, we use L units vector $\mathbf{h} = (h_1, h_2, \ldots, h_L)$, with $L \ll D$.

The main characteristic of a RBM is the absence of intra-layer connections: visible units do not interact with each other, and hidden units do not interact among themselves. All connections run strictly between the two layers, a restriction that distinguishes RBMs from general Boltzmann Machines [2]. This bipartite structure makes units in one layer conditionally

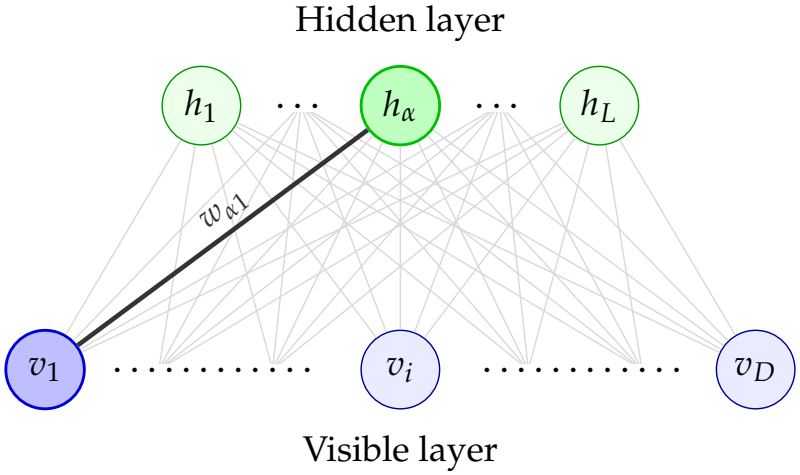

Figure 1.1: *Bipartite architecture of a RBM, with no intra-layer connections, consisting of D visible units and L hidden units. The highlighted link between h_α and v_1 has the associated weight $w_{\alpha 1}$.*

independent given the other, enabling efficient sampling procedures, such as Gibbs sampling, and this is the core reason RBMs have played a central role in the development of generative models.

From now on, we will focus on the case where the units of both layers are discrete and have only binary values, namely $v_i \in \{0, 1\}$ and $h_\alpha \in \{0, 1\}$.

1.1 The energy function and conditional probability

In an RBM, a scalar energy function is defined for every joint state $(\mathbf{v}, \mathbf{h})$.
The joint energy function $E(\mathbf{v}, \mathbf{h})$ is defined by the following expression:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{D} a_i v_i - \sum_{\alpha=1}^{L} b_\alpha h_\alpha - \sum_{i=1}^{D}\sum_{\alpha=1}^{L} v_i w_{i\alpha} h_\alpha \tag{1.1}$$

The choice of the energy function in Eq. 1.1 is inspired by the discrete Ising model in statistical mechanics [11]. Note that by adopting an interaction term between visible and hidden units, analogous to spin–spin couplings, we get a probabilistic model whose structure mirrors the Ising model while remaining computationally tractable thanks to bipartite restriction and the choice of a small hidden space $L \ll D$.

The model parameters are the two biases vectors $\mathbf{a}$, $\mathbf{b}$ and the interacting matrix $\mathcal{W}$:

- $a_i \in \mathbb{R}$, representing the local bias acting on the visible unit i.
- $b_\alpha \in \mathbb{R}$, representing the local bias acting on the hidden unit α.
- $w_{i\alpha} \in \mathbb{R}$, representing the symmetric interaction weight connecting visible unit i and hidden unit α.

The joint probability distribution of finding the network in a specific $(\mathbf{v}, \mathbf{h})$ state is given by the classic Boltzmann-Gibbs distribution:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})} \tag{1.2}$$

where Z is the partition function, obtained by summing the unnormalized Boltzmann weights over the whole state space:

$$Z = \sum_{\mathbf{v}'}\sum_{\mathbf{h}'} e^{-E(\mathbf{v}', \mathbf{h}')} \tag{1.3}$$

The probability of obtaining a visible vector $\mathbf{v}$, is obtained by marginalizing the joint distribution:

$$P(\mathbf{v}) = \sum_{\mathbf{h}'} P(\mathbf{v}, \mathbf{h}') = \frac{1}{Z}\sum_{\mathbf{h}'} e^{-E(\mathbf{v}, \mathbf{h}')} \tag{1.4}$$

Thanks to the bipartite structure, we can further massage Eq. 1.4. By substituting the explicit form of the energy function (1.1), recalling that there are no cross-terms between different

hidden units, we get:

$$
\begin{aligned}
P(\mathbf{v}) &= \frac{1}{Z}\sum_{\mathbf{h}'} e^{-E(\mathbf{v},\mathbf{h}')} \\
&= \frac{1}{Z}\sum_{h_1\in\{0,1\}}\cdots\sum_{h_L\in\{0,1\}}\exp\left(\sum_{i=1}^{D}a_i v_i + \sum_{\alpha=1}^{L} b_\alpha h_\alpha + \sum_{i=1}^{D}\sum_{\alpha=1}^{L} v_i w_{i\alpha} h_\alpha\right) \\
&= \frac{1}{Z}\, e^{\sum_{i=1}^{D} a_i v_i}\sum_{h_1\in\{0,1\}}\cdots\sum_{h_L\in\{0,1\}}\prod_{\alpha=1}^{L}\exp\left(b_\alpha h_\alpha + h_\alpha\sum_{i=1}^{D} v_i w_{i\alpha}\right) \\
&= \frac{1}{Z}\, e^{\sum_{i=1}^{D} a_i v_i}\prod_{\alpha=1}^{L}\left(\sum_{h_\alpha\in\{0,1\}} e^{h_\alpha(b_\alpha+\sum_{i=1}^{D} v_i w_{i\alpha})}\right) \\
&= \frac{1}{Z}\, e^{\sum_{i=1}^{D} a_i v_i}\prod_{\alpha=1}^{L}\left(1 + e^{b_\alpha+\sum_{i=1}^{D} v_i w_{i\alpha}}\right)
\end{aligned}
\tag{1.5}
$$

Let us now derive the conditional probability $P(\mathbf{h}|\mathbf{v})$. By definition of conditional probability:

$$
P(\mathbf{h}|\mathbf{v}) = \frac{P(\mathbf{v},\mathbf{h})}{P(\mathbf{v})} = \frac{\frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})}}{\frac{1}{Z}\sum_{\mathbf{h}'} e^{-E(\mathbf{v},\mathbf{h}')}} = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}'} e^{-E(\mathbf{v},\mathbf{h}')}} \tag{1.6}
$$

Finally, using (1.5) and simplifying the factor $e^{\sum_i a_i v_i}$, we obtain:

$$
P(\mathbf{h}|\mathbf{v}) = \prod_{\alpha=1}^{L}\frac{e^{h_\alpha(b_\alpha+\sum_i v_i w_{i\alpha})}}{1+e^{b_\alpha+\sum_i v_i w_{i\alpha}}} = P(\mathbf{h}|\mathbf{v}) = \prod_{\alpha=1}^{L} P(h_\alpha|\mathbf{v}) \tag{1.7}
$$

Where in that passage we highlighted that the states of the hidden units are mutually independent when conditioned on the visible layer.

Therefore the probability of being active for a hidden unit is:

$$
P(h_\alpha = 1|\mathbf{v}) = \frac{e^{1\cdot(b_\alpha+\sum_i v_i w_{i\alpha})}}{1+e^{b_\alpha+\sum_i v_i w_{i\alpha}}} = \frac{1}{1+e^{-(b_\alpha+\sum_{i=1}^{D} v_i w_{i\alpha})}} = \sigma\left(b_\alpha + \sum_{i=1}^{D} v_i w_{i\alpha}\right) \tag{1.8}
$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the sigmoid function and we usually refer to the argument of the sigmoid function $b_\alpha + \sum_{i=1}^{D} v_i w_{i\alpha}$ as the local field, exploiting the analogy with statistical mechanics. By repeating the same derivation for the visible layer, we find that the visible

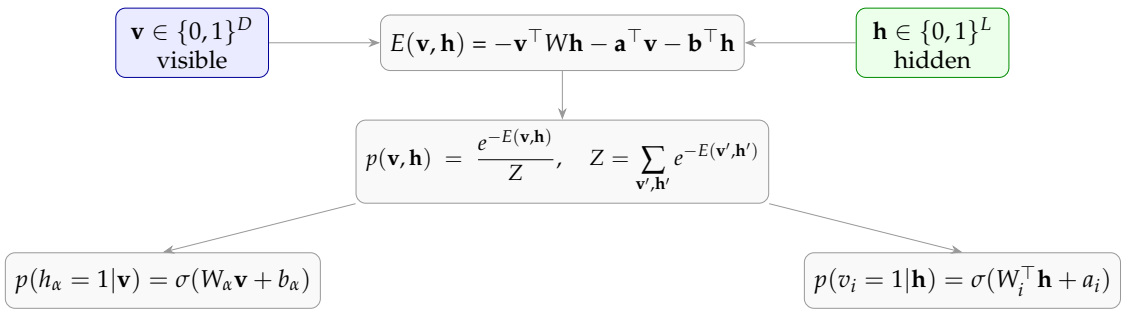

Figure 1.2: *Summary diagram of the formulas governing a discrete (0/1) RBM and their relationships.*

units are also mutually independent:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P(v_i|\mathbf{h}) \tag{1.9}$$

Where the activation probability for an individual visible unit i is again determined by the sigmoid function acting on its local field:

$$P(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_{\alpha=1}^{L} w_{i\alpha} h_\alpha\right) \tag{1.10}$$

These conditional probabilities form the mathematical foundation of RBMs, enabling effective sampling of both the visible and hidden layers during training.

1.2 Detailed balance condition in RBMs

A key feature of RBMs is that, given their construction, the dynamics takes place at thermodynamic equilibrium. This equilibrium behavior is guaranteed by the fact that the transition dynamics satisfy the detailed balance condition.

In a stochastic process, a transition matrix $T(\mathbf{s} \to \mathbf{s}')$ satisfies detailed balance condition with respect to a stationary distribution $P(\mathbf{s})$ if the probability flux between any pair of states is exactly balanced by the reverse probability flux:

$$P(\mathbf{s})T(\mathbf{s} \to \mathbf{s}') = P(\mathbf{s}')T(\mathbf{s}' \to \mathbf{s}) \tag{1.11}$$

In an RBM, sampling is typically performed using a Gibbs sampling procedure. One full step in the visible space (namely from $\mathbf{v}$ to $\mathbf{v}'$) is obtained by passing through an intermediate hidden state $\mathbf{h}$. Hence the transition probability $T(\mathbf{v} \to \mathbf{v}')$ is defined by summing over all possible intermediate $\mathbf{h}$:

$$T(\mathbf{v} \to \mathbf{v}') = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})P(\mathbf{v}'|\mathbf{h}) \tag{1.12}$$

Let us explicitly verify whether the equilibrium marginal distribution $P(\mathbf{v})$ satisfies the detailed balance condition with respect to this Gibbs transition matrix.

$$\begin{aligned} &\text{Flux}(\mathbf{v} \to \mathbf{v}') = P(\mathbf{v})T(\mathbf{v} \to \mathbf{v}') = \\ &P(\mathbf{v}) \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})P(\mathbf{v}'|\mathbf{h}) = \sum_{\mathbf{h}} P(\mathbf{v})P(\mathbf{h}|\mathbf{v})P(\mathbf{v}'|\mathbf{h}) \end{aligned} \tag{1.13}$$

Using the definition of conditional probability we can rewrite the previous one as:

$$\text{Flux}(\mathbf{v} \to \mathbf{v}') = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}) \frac{P(\mathbf{v}', \mathbf{h})}{P(\mathbf{h})} \tag{1.14}$$

Now recall that the joint probabilities $P(\mathbf{v}, \mathbf{h})$ and $P(\mathbf{v}', \mathbf{h})$ are based on the Boltzmann

distribution as seen in Eq 1.2:

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})}, \quad P(\mathbf{v}',\mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v}',\mathbf{h})} \tag{1.15}$$

We can explicitly expand the fraction inside the summation in Eq. 1.14:

$$\frac{P(\mathbf{v},\mathbf{h})P(\mathbf{v}',\mathbf{h})}{P(\mathbf{h})} = \frac{1}{Z}\frac{1}{Z}\frac{e^{-E(\mathbf{v},\mathbf{h})}e^{-E(\mathbf{v}',\mathbf{h})}}{\sum_{\mathbf{v}''}P(\mathbf{v}'',\mathbf{h})} \tag{1.16}$$

Notice that the terms in the numerator are completely symmetric under the transposition of the visible states. Because the weight matrix $w_{i\alpha}$ is shared and identical for both the forward and backward paths, the joint distribution can be refactored symmetrically:

$$P(\mathbf{v},\mathbf{h})\frac{P(\mathbf{v}',\mathbf{h})}{P(\mathbf{h})} = P(\mathbf{v}',\mathbf{h})\frac{P(\mathbf{v},\mathbf{h})}{P(\mathbf{h})} = P(\mathbf{v}',\mathbf{h})P(\mathbf{v}|\mathbf{h}) \tag{1.17}$$

Therefore, we can rewrite (1.14) by switching the roles of $\mathbf{v}$ and $\mathbf{v}'$:

$$\begin{aligned}\text{Flux}(\mathbf{v}\to\mathbf{v}') &= \sum_{\mathbf{h}}P(\mathbf{v}',\mathbf{h})P(\mathbf{v}|\mathbf{h}) = \sum_{\mathbf{h}}P(\mathbf{v}')P(\mathbf{h}|\mathbf{v}')P(\mathbf{v}|\mathbf{h}) \\ &= P(\mathbf{v}')T(\mathbf{v}'\to\mathbf{v}) = \text{Flux}(\mathbf{v}'\to\mathbf{v})\end{aligned} \tag{1.18}$$

And this concludes the verification, where the critical point is the sharing of the same weights between the forward and backward conditional steps, which implies that the system obeys time-reversal symmetry.

1.3 Log-likelihood and training

To train a RBM, we optimize the parameters to maximize the log-likelihood of a training dataset composed of N samples. The objective function is the average log-likelihood $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\ln P(\mathbf{v}^{(n)}) \tag{1.19}$$

where $\mathbf{v}^{(n)}$ is the n-th data sample. Let us derive the gradient of the log-likelihood for a single $\mathbf{v}$ with respect to the weight $w_{i\alpha}$.

Exploiting the chain rule of the natural logarithm:

$$\frac{\partial \ln P(\mathbf{v})}{\partial w_{i\alpha}} = \frac{1}{P(\mathbf{v})}\frac{\partial P(\mathbf{v})}{\partial w_{i\alpha}} = \frac{1}{P(\mathbf{v})}\frac{\partial}{\partial w_{i\alpha}}\left(\frac{\sum_{\mathbf{h}}e^{-E(\mathbf{v},\mathbf{h})}}{Z}\right) \tag{1.20}$$

Applying the quotient differentiation:

$$\frac{\partial \ln P(\mathbf{v})}{\partial w_{i\alpha}} = \frac{1}{P(\mathbf{v})} \left[\frac{1}{Z} \frac{\partial \left(\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right)}{\partial w_{i\alpha}} - \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{Z^2} \frac{\partial Z}{\partial w_{i\alpha}} \right] \tag{1.21}$$

We recognize a $P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ term to simplify further:

$$\frac{\partial \ln P(\mathbf{v})}{\partial w_{i\alpha}} = \frac{1}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \frac{\partial \left(\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right)}{\partial w_{i\alpha}} - \frac{1}{Z} \frac{\partial Z}{\partial w_{i\alpha}} \tag{1.22}$$

Let us consider the derivative in the first term of the above expression. Recalling that from (1.1) we get $\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i\alpha}} = -v_i h_\alpha$.

$$\frac{\partial \left(\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \right)}{\partial w_{i\alpha}} = \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{i\alpha}} \right) = \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} v_i h_\alpha \tag{1.23}$$

We substitute this back into 1.22:

$$\frac{1}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} v_i h_\alpha = \sum_{\mathbf{h}} \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}'} e^{-E(\mathbf{v},\mathbf{h}')}} v_i h_\alpha = \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v}) v_i h_\alpha \tag{1.24}$$

This term represents the conditional expectation value of the product $v_i h_\alpha$ given the data vector $\mathbf{v}$, which is often written as $\langle v_i h_\alpha \rangle_{\text{data}}$.

Now, let us calculate the second term in (1.22). We expand the definition of Z from (1.3):

$$\frac{\partial Z}{\partial w_{i\alpha}} = \frac{\partial}{\partial w_{i\alpha}} \sum_{\mathbf{v}'} \sum_{\mathbf{h}'} e^{-E(\mathbf{v}',\mathbf{h}')} = \sum_{\mathbf{v}'} \sum_{\mathbf{h}'} e^{-E(\mathbf{v}',\mathbf{h}')} \left(-\frac{\partial E(\mathbf{v}',\mathbf{h}')}{\partial w_{i\alpha}} \right) = \sum_{\mathbf{v}'} \sum_{\mathbf{h}'} e^{-E(\mathbf{v}',\mathbf{h}')} v'_i h'_\alpha \tag{1.25}$$

Also considering $\frac{1}{Z}$ multiplying factor:

$$\frac{1}{Z} \frac{\partial Z}{\partial w_{i\alpha}} = \sum_{\mathbf{v}'} \sum_{\mathbf{h}'} \frac{1}{Z} e^{-E(\mathbf{v}',\mathbf{h}')} v'_i h'_\alpha = \sum_{\mathbf{v}'} \sum_{\mathbf{h}'} P(\mathbf{v}',\mathbf{h}') v'_i h'_\alpha \tag{1.26}$$

This term represents the expected value of the product $v_i h_\alpha$ over the full joint distribution of the model, which is often written as $\langle v_i h_\alpha \rangle_{\text{model}}$.

Combining both parts, we arrive at the final expression for the log-likelihood gradient with respect to the weights:

$$\frac{\partial \ln P(\mathbf{v})}{\partial w_{i\alpha}} = \langle v_i h_\alpha \rangle_{\text{data}} - \langle v_i h_\alpha \rangle_{\text{model}} \tag{1.27}$$

By following an identical mathematical procedure for the visible biases a_i and hidden biases b_α, we get:

$$\frac{\partial \ln P(\mathbf{v})}{\partial a_i} = \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \tag{1.28}$$

$$\frac{\partial \ln P(\mathbf{v})}{\partial b_\alpha} = \langle h_\alpha \rangle_{\text{data}} - \langle h_\alpha \rangle_{\text{model}} \tag{1.29}$$

Equations (1.27), (1.28), and (1.29) all share the same structure:

1. **The Positive Phase ($\langle \cdot \rangle_{\text{data}}$):** Here, the visible units are clamped to real data, and the hidden units are sampled accordingly. This phase lowers the energy of configurations that resemble the dataset, making those states more likely.

2. **Negative Phase ($\langle \cdot \rangle_{\text{model}}$):** In this phase, the model runs freely without any data constraints. It explores the states it can generate on its own, the so-called fantasy particles. These unobserved states get their energy increased, which makes them less likely.

While the positive phase is straightforward to compute, the negative phase requires averaging over the full joint distribution $P(\mathbf{v}, \mathbf{h})$, which is computationally intractable. In practice, this term is approximated using stochastic sampling algorithms such as Contrastive Divergence (CD-k) [12, 13] or Persistent Contrastive Divergence (PCD) [14], which track the equilibrium distribution through finite Markov Chain steps.

Chapter 2

Steady-State Markov Machines

2.1 Model architecture

Here, differently from the RBMs notation, we denote the visible vectors as $\mathbf{x} = (x_1, \ldots, x_D)$ with $x_i \in \{0, 1\}$ (they will be an image made up of D pixels in our dataset), and the hidden vectors as $\mathbf{z} = (z_1, \ldots, z_L)$ with $z_\alpha \in \{0, 1\}$.

In a steady-state Markov Machine (ssMM) instead of relying on an unique energy function, the relation between the two layers is defined through two independent Markov transition matrices:

$$A_{\mathbf{zx}} \equiv P(\mathbf{x}|\mathbf{z}), \qquad \overline{A}_{\mathbf{xz}} \equiv P(\mathbf{z}|\mathbf{x}) \tag{2.1}$$

These matrices refer respectively to the forward path from $\mathbf{z}$ to $\mathbf{x}$ (the sleep or negative phase in an RBM) and the backward path from $\mathbf{x}$ to $\mathbf{z}$ (the awake or positive phase in an RBM).

To be valid probabilities, they must satisfy the normalization conditions $\sum_{\mathbf{x}} A_{\mathbf{zx}} = 1$ and $\sum_{\mathbf{z}} \overline{A}_{\mathbf{xz}} = 1$.

We parameterize these matrices using an exponential way that resembles the RBM form:

$$A_{\mathbf{zx}} = \frac{e^{\mathbf{x}\cdot(w\cdot\mathbf{z}+a)}}{\sum_{\mathbf{x}'} e^{\mathbf{x}'\cdot(w\cdot\mathbf{z}+a)}} \tag{2.2a}$$

$$\overline{A}_{\mathbf{xz}} = \frac{e^{\mathbf{z}\cdot(\overline{w}\cdot\mathbf{x}+\overline{a})}}{\sum_{\mathbf{z}'} e^{\mathbf{z}'\cdot(\overline{w}\cdot\mathbf{x}+\overline{a})}} \tag{2.2b}$$

where a and $\overline{a}$ are the bias vectors for the visible and hidden spaces.

This is where the model introduces a fundamental structural shift. The weights connecting the layers are governed by two entirely separate matrices: w drives the $\mathbf{z} \to \mathbf{x}$ transitions, while $\overline{w}$ drives the $\mathbf{x} \to \mathbf{z}$ transitions. Unlike RBMs, w and $\overline{w}$ are not transposes of each other. This decoupling of the forward and backward paths is a crucial aspect of the architecture, as it leads to an explicit break in the detailed balance condition and allows the system to gain non-equilibrium dynamics. Note that throughout the rest of this work, we will often refer simply to the forward local field, $h_i = \sum_{\alpha=1}^{L} w_{i\alpha} z_\alpha + a_i$, and the backward local field, $\overline{h}_\alpha = \sum_{i=1}^{D} \overline{w}_{\alpha i} x_i + \overline{a}_\alpha$.

Thanks to this exponential parameterization, if we are dealing with independent individual units/coordinates in a vector layer the transition probability factorizes down to the level of each unit, making the sampling extremely simple. In the case of the forward pass, considering each pixel as independent, the conditional probability becomes:

$$A_{\mathbf{zx}} \equiv P(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{D} P(x_i|\mathbf{z}) = \prod_{i=1}^{D} \frac{e^{x_i h_i}}{1+e^{h_i}} = \prod_{i=1}^{D} \sigma(h_i)^{x_i} \big(1-\sigma(h_i)\big)^{1-x_i} \tag{2.3a}$$

The last step simply exploits the fact that $x_i \in \{0,1\}$, where the exponents x_i and $1-x_i$ act as selectors between the two states, yielding a factorization of terms involving the sigmoid function evaluated at the corresponding local field coordinates.

In exactly the same way, if the L units in a hidden vector are independent, the backward conditional probability is:

$$\overline{A}_{\mathbf{xz}} \equiv P(\mathbf{z}|\mathbf{x}) = \prod_{\alpha=1}^{L} P(z_\alpha|\mathbf{x}) = \prod_{\alpha=1}^{L} \frac{e^{z_\alpha \overline{h}_\alpha}}{1+e^{\overline{h}_\alpha}} = \prod_{\alpha=1}^{L} \sigma(\overline{h}_\alpha)^{z_\alpha} \big(1-\sigma(\overline{h}_\alpha)\big)^{1-z_\alpha} \tag{2.3b}$$

By construction of an ssMM we alternate between $\overline{A}$ and A and set up a discrete-time

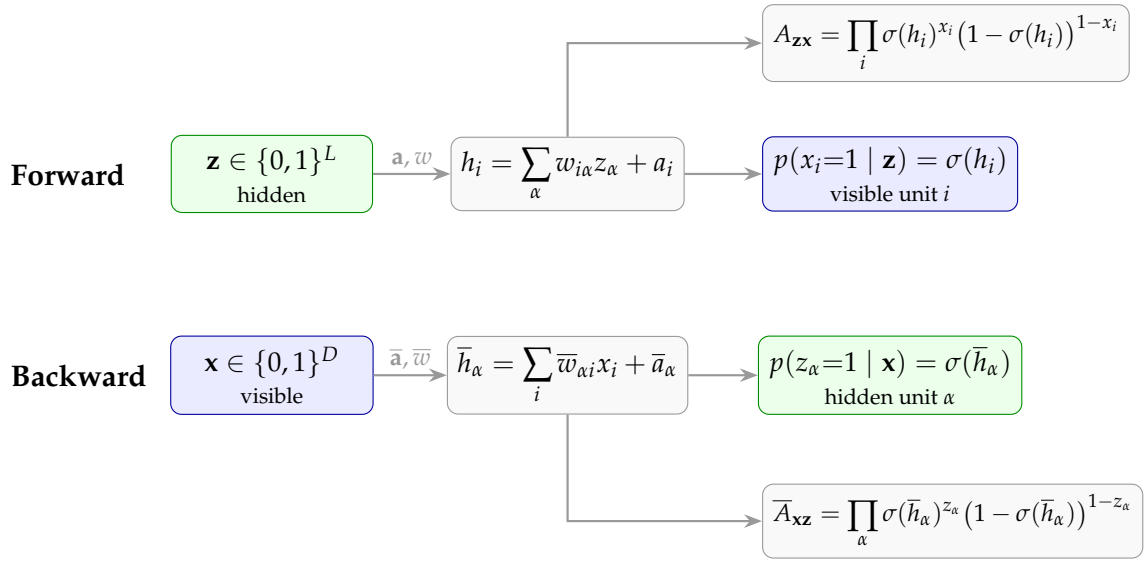

Figure 2.1: *Schematic representation of the forward $A_{\mathbf{zx}} = P(\mathbf{x}|\mathbf{z})$ and backward $\overline{A}_{\mathbf{xz}} = P(\mathbf{z}|\mathbf{x})$ steps, leading to factorization across conditionally independent units via sigmoid activations.*

Markov chain across the two layers [8]. A full step in the latent space $\mathbf{z} \to \mathbf{z}'$ is defined by first generating a visible $\mathbf{x}$ and then proceeding with a backward pass into the hidden space (Fig. 2.2). We can express this latent transition matrix M as:

$$M_{\mathbf{zz}'} = \sum_{\mathbf{x}} A_{\mathbf{zx}} \overline{A}_{\mathbf{xz}'} \tag{2.4}$$

If we let this Markov chain run, the hidden states eventually reach a stationary probability

distribution $p(\mathbf{z})$, which is simply the solution to the eigenvalue equation:

$$p(\mathbf{z}) = \sum_{\mathbf{z}'} M_{\mathbf{z}\mathbf{z}'} p(\mathbf{z}') \tag{2.5}$$

We will look at the dynamics of this Markov chain in the latent space, see how

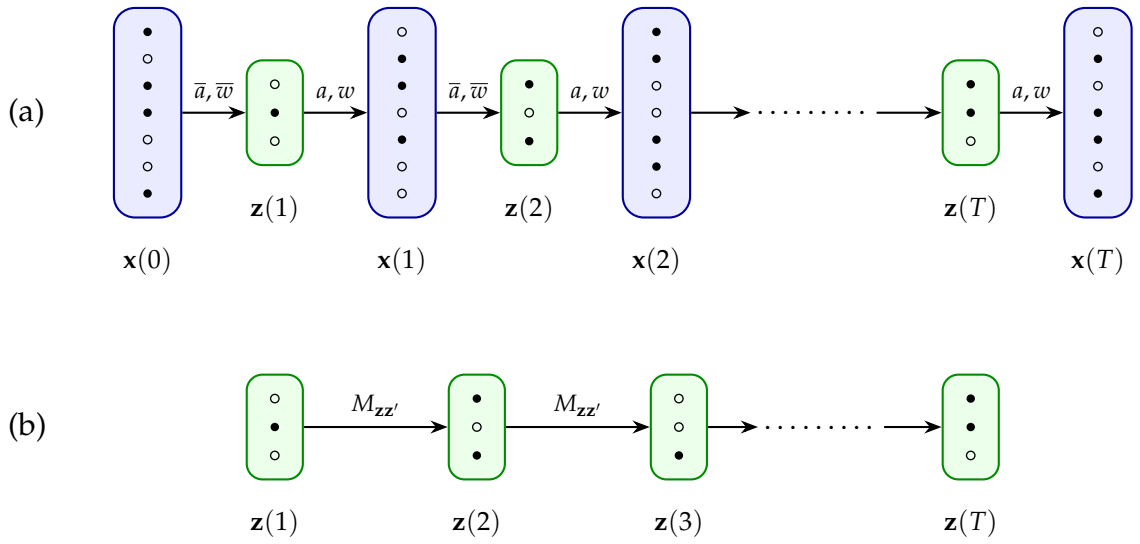

Figure 2.2: *Illustration of the generative process in a ssMM. Inside each box, Black and white dots represent the values of the binary units,* 1*s and* 0*s. (a) Evolution from an initial state* $\mathbf{x}(0)$ *to* $\mathbf{x}(T)$ *through repeated forward-backward sampling. After* T *such alternations, the chain outputs* $\mathbf{x}(T)$*, the final fantasy particle produced by the model. (b) Markov chain on the latent units, where* $M_{\mathbf{z}\mathbf{z}'} = \sum_{\mathbf{x}} A_{\mathbf{z}\mathbf{x}} \overline{A}_{\mathbf{x}\mathbf{z}'}$ *condenses each round-trip into a direct transition* $\mathbf{z}(t) \to \mathbf{z}(t+1)$*.*

non-equilibrium cycles emerge, and ultimately conclude that not having an equilibrium constraint gives the model much more freedom to explore the latent space. Moreover, by taking advantage of this property, we can obtain models with very good performance in a more consistent way.

2.2 Log-likelihood gradient

The goal of the training process is to best match the dataset distribution. The marginal probability of observing a specific input $\mathbf{x}$ depends on the steady-state distribution of the latent variables and the forward transition matrix. We write this as:

$$p(\mathbf{x}) = \sum_{\mathbf{z}} A_{\mathbf{z}\mathbf{x}} p(\mathbf{z}) \tag{2.6}$$

So we define the log-likelihood as:

$$\mathcal{L} = \frac{1}{N} \sum_{m=1}^{N} \ln p(\mathbf{x}^{(m)}) \tag{2.7}$$

where N is the total number of elements in the dataset $\mathcal{D} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})$.

It is important to clarify that all values of $\mathcal{L}$ reported in plots and tables in this thesis refer to Eq. (2.7) evaluated at the end of training. The regularization terms introduced in Chapter 3 only act as gradient modifications during training.

To optimize the model, following standard practice in unsupervised learning, we perform gradient ascent on this log-likelihood.

We will now compute the derivative of $\ln p(\mathbf{x})$ (log-likelihood for a single data point) with respect to a generic forward weight $w_{i\alpha}$.

As for RBM we apply the chain rule of the logarithm:

$$\frac{\partial}{\partial w_{i\alpha}} \ln p(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \frac{\partial p(\mathbf{x})}{\partial w_{i\alpha}} = \frac{1}{p(\mathbf{x})} \frac{\partial}{\partial w_{i\alpha}} \left(\sum_{\mathbf{z}} A_{\mathbf{zx}} p(\mathbf{z}) \right) \tag{2.8}$$

By linearity, we move the derivative inside the summation:

$$\frac{\partial}{\partial w_{i\alpha}} \ln p(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{\mathbf{z}} \frac{\partial}{\partial w_{i\alpha}} \left(A_{\mathbf{zx}} p(\mathbf{z}) \right) \tag{2.9}$$

Now we introduce an approximation. We truncate the dependence of the Markov chain to the last two steps. To see why this is necessary, notice that the steady-state latent distribution $p(\mathbf{z})$ satisfies the stationarity condition over two-step transitions ($\mathbf{z}^{(t-1)} \to \mathbf{x}^{(t-1)} \to \mathbf{z}^{(t)}$). Writing the chain back in time, we have:

$$\begin{aligned} p(\mathbf{z}) &= \sum_{\mathbf{x}} p(\mathbf{x}) \overline{A}_{\mathbf{xz}} = \sum_{\mathbf{x},\mathbf{z}'} p(\mathbf{z}') A_{\mathbf{z}'\mathbf{x}} \overline{A}_{\mathbf{xz}} \\ &= \sum_{\mathbf{x},\mathbf{z}',\mathbf{x}',\mathbf{z}''} p(\mathbf{z}'') A_{\mathbf{z}''\mathbf{x}'} \overline{A}_{\mathbf{x}'\mathbf{z}'} A_{\mathbf{z}'\mathbf{x}} \overline{A}_{\mathbf{xz}} = \dots \end{aligned} \tag{2.10}$$

Differentiating the marginal probability $p(\mathbf{x}) = \sum_{\mathbf{z}} A_{\mathbf{zx}} p(\mathbf{z})$ with respect to the forward parameters $w_{i\alpha}$ yields:

$$\frac{\partial p(\mathbf{x})}{\partial w_{i\alpha}} = \sum_{\mathbf{z}} \left(\frac{\partial A_{\mathbf{zx}}}{\partial w_{i\alpha}} p(\mathbf{z}) + A_{\mathbf{zx}} \frac{\partial p(\mathbf{z})}{\partial w_{i\alpha}} \right). \tag{2.11}$$

As shown by Eq. (2.10), the steady-state distribution $p(\mathbf{z})$ contains forward matrices A across all earlier time steps, meaning that an exact calculation of $\frac{\partial p(\mathbf{z})}{\partial w_{i\alpha}}$ would be computationally intractable. We neglect these long-range temporal dependencies, and since the parameters of the backward transition $\overline{A}_{\mathbf{xz}}$ are decoupled from the forward transition $A_{\mathbf{zx}}$, we treat the steady-state distribution $p(\mathbf{z})$ as constant when differentiating with respect to the forward parameters, so that Eq. (2.9) becomes:

$$\frac{\partial}{\partial w_{i\alpha}} \ln p(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{\mathbf{z}} p(\mathbf{z}) \left(\frac{\partial A_{\mathbf{zx}}}{\partial w_{i\alpha}} \right). \tag{2.12}$$

We focus on the term in parentheses, using again the calculus identity $\frac{\partial \ln f(\theta)}{\partial \theta} = \frac{1}{f(\theta)} \frac{\partial f(\theta)}{\partial \theta}$ we

obtain:

$$\frac{\partial A_{\mathbf{zx}}}{\partial w_{i\alpha}} = A_{\mathbf{zx}} \frac{\partial \ln A_{\mathbf{zx}}}{\partial w_{i\alpha}} \tag{2.13}$$

Inserting (2.13) into Equation (2.12):

$$\frac{\partial}{\partial w_{i\alpha}} \ln p(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{\mathbf{z}} p(\mathbf{z}) A_{\mathbf{zx}} \left(\frac{\partial \ln A_{\mathbf{zx}}}{\partial w_{i\alpha}} \right) \tag{2.14}$$

Looking at $\frac{\partial \ln A_{\mathbf{zx}}}{\partial w_{i\alpha}}$, from (2.3a):

$$\ln A_{\mathbf{zx}} = \sum_{j=1}^{D} \left[x_j h_j - \ln\left(1 + e^{h_j}\right) \right] \tag{2.15}$$

Since the field h_j depends on $w_{i\alpha}$ only when $j = i$, in the partial derivative the sum collapses to a single term:

$$\frac{\partial \ln A_{\mathbf{zx}}}{\partial w_{i\alpha}} = \frac{\partial}{\partial w_{i\alpha}} \left[x_i h_i - \ln\left(1 + e^{h_i}\right) \right] \tag{2.16}$$

So:

$$\frac{\partial \ln A_{\mathbf{zx}}}{\partial w_{i\alpha}} = x_i \frac{\partial h_i}{\partial w_{i\alpha}} - \frac{1}{1 + e^{h_i}} \left(e^{h_i} \frac{\partial h_i}{\partial w_{i\alpha}} \right) = \left(x_i - \frac{e^{h_i}}{1 + e^{h_i}} \right) \frac{\partial h_i}{\partial w_{i\alpha}} \tag{2.17}$$

We recognize the sigmoid function, which represents the mean activation of the visible unit given the hidden state. We define this as $\mu_{x_i|\mathbf{z}} \equiv \sigma(h_i)$. Then from the definition of h_i, we have $\frac{\partial h_i}{\partial w_{i\alpha}} = z_\alpha$. Substituting these yields the final result:

$$\frac{\partial \ln A_{\mathbf{zx}}}{\partial w_{i\alpha}} = (x_i - \mu_{x_i|\mathbf{z}}) z_\alpha \tag{2.18}$$

Hence substituting back into (2.14), using the definition of $p(\mathbf{x})$:

$$\frac{\partial}{\partial w_{i\alpha}} \ln p(\mathbf{x}) = \frac{\sum_{\mathbf{z}} p(\mathbf{z}) A_{\mathbf{zx}} z_\alpha (x_i - \mu_{x_i|\mathbf{z}})}{\sum_{\mathbf{z}} p(\mathbf{z}) A_{\mathbf{zx}}} \tag{2.19}$$

We can write this expression as an expectation value over the generated trajectory:

$$\frac{\partial}{\partial w_{i\alpha}} \ln p(\mathbf{x}) = \frac{\langle A_{\mathbf{zx}} z_\alpha (x_i - \mu_{x_i|\mathbf{z}}) \rangle_{p(\mathbf{z})}}{\langle A_{\mathbf{zx}} \rangle_{p(\mathbf{z})}} \tag{2.20}$$

The derivation for the gradient with respect to the visible bias a_i follows the same steps. The only difference is $\frac{\partial h_i}{\partial a_i} = 1$. So:

$$\frac{\partial}{\partial a_i} \ln p(\mathbf{x}) = \frac{\langle A_{\mathbf{zx}} (x_i - \mu_{x_i|\mathbf{z}}) \rangle_{p(\mathbf{z})}}{\langle A_{\mathbf{zx}} \rangle_{p(\mathbf{z})}} \tag{2.21}$$

For the backward parameters ($\overline{w}_{\alpha i}$ and $\overline{a}_\alpha$) the precedure is the same. The logarithm of (2.3b)

is:

$$\ln \overline{A}_{\mathbf{xz}} = \sum_{\beta=1}^{L} \left[z_\beta \overline{h}_\beta - \ln\left(1 + e^{\overline{h}_\beta}\right) \right] \tag{2.22}$$

Computing the derivative with respect to the backward weight $\overline{w}_{\alpha i}$:

$$\frac{\partial \ln \overline{A}_{\mathbf{xz}}}{\partial \overline{w}_{\alpha i}} = \frac{\partial}{\partial \overline{w}_{\alpha i}} \left[z_\alpha \overline{h}_\alpha - \ln\left(1 + e^{\overline{h}_\alpha}\right) \right] = \left(z_\alpha - \frac{e^{\overline{h}_\alpha}}{1 + e^{\overline{h}_\alpha}} \right) \frac{\partial \overline{h}_\alpha}{\partial \overline{w}_{\alpha i}} \tag{2.23}$$

We define the mean hidden activation as $\mu_{z_\alpha|\mathbf{x}} \equiv \sigma(\overline{h}_\alpha)$. Since $\frac{\partial \overline{h}_\alpha}{\partial \overline{w}_{\alpha i}} = x_i$ the result is:

$$\frac{\partial \ln \overline{A}_{\mathbf{xz}}}{\partial \overline{w}_{\alpha i}} = (z_\alpha - \mu_{z_\alpha|\mathbf{x}}) x_i \tag{2.24}$$

Similarly, for the hidden bias $\frac{\partial \overline{h}_\alpha}{\partial \overline{a}_\alpha} = 1$, hence:

$$\frac{\partial \ln \overline{A}_{\mathbf{xz}}}{\partial \overline{a}_\alpha} = (z_\alpha - \mu_{z_\alpha|\mathbf{x}}) \tag{2.25}$$

To sum up the gradients for the forward pass are given by:

$$\frac{\partial \ln p(\mathbf{x})}{\partial w_{j\beta}} = \frac{\left\langle A_{\mathbf{zx}}\, z_\beta \left(x_j - \mu_{x_j|\mathbf{z}}\right) \right\rangle_{p(\mathbf{z})}}{\left\langle A_{\mathbf{zx}} \right\rangle_{p(\mathbf{z})}}, \tag{2.26a}$$

$$\frac{\partial \ln p(\mathbf{x})}{\partial a_j} = \frac{\left\langle A_{\mathbf{zx}} \left(x_j - \mu_{x_j|\mathbf{z}}\right) \right\rangle_{p(\mathbf{z})}}{\left\langle A_{\mathbf{zx}} \right\rangle_{p(\mathbf{z})}}. \tag{2.26b}$$

For the backward pass the gradients follow an analogous expression. Here the average is taken over the joint distribution $p(\mathbf{x}', \mathbf{z}) = p(\mathbf{x}')\overline{A}_{\mathbf{x}'\mathbf{z}}$, i.e. over a fantasy sample $\mathbf{x}'$ drawn from the model together with the hidden state $\mathbf{z}$ it subsequently generates, rather than over $\mathbf{z}$ alone as in the forward case:

$$\frac{\partial \ln p(\mathbf{x})}{\partial \overline{w}_{\beta j}} = \frac{\left\langle A_{\mathbf{zx}}\, x'_j \left(z_\beta - \mu_{z_\beta|\mathbf{x}'}\right) \right\rangle_{p(\mathbf{x}', \mathbf{z})}}{\left\langle A_{\mathbf{zx}} \right\rangle_{p(\mathbf{z})}}, \tag{2.27a}$$

$$\frac{\partial \ln p(\mathbf{x})}{\partial \overline{a}_\beta} = \frac{\left\langle A_{\mathbf{zx}} \left(z_\beta - \mu_{z_\beta|\mathbf{x}'}\right) \right\rangle_{p(\mathbf{x}', \mathbf{z})}}{\left\langle A_{\mathbf{zx}} \right\rangle_{p(\mathbf{z})}}. \tag{2.27b}$$

These simple and elegant expressions rule the entire learning process of the ssMM. Notice how the model parameters are updated by minimizing the discrepancy between the statistics observed in the dataset and those generated by the model. Eqs. 2.26a, 2.26b and 2.27a, 2.27b also avoid the computational bottlenecks of calculating a global partition function Z as it was the case in RBM, making the network highly efficient while allowing it to explore non-equilibrium phase spaces.

2.3 One-hot encoding

In this thesis we will focus mainly on the one-hot case for the latent space; we will have exactly one of the L hidden units active at each step of the Markov chain, i.e. $z_\alpha = 1$ for a single α and $z_\beta = 0$ for all $\beta \neq \alpha$.
We will still use the Bernoulli factorization in Eq. 2.3b which treats each hidden units as an independent Bernoulli variable with mean activation probability $\mu_{z_\alpha|\mathbf{x}} = \sigma(\overline{h}_\alpha)$. The one-hot constraint is imposed by normalizing these marginal probabilities into a categorical distribution,

$$p_{\text{one}}(\alpha \mid \mathbf{x}) = \frac{p(z_\alpha = 1 \mid \mathbf{x})}{\sum_{\beta=1}^{L} p(z_\beta = 1 \mid \mathbf{x})} = \frac{\sigma(\overline{h}_\alpha)}{\sum_{\beta=1}^{L} \sigma(\overline{h}_\beta)}, \tag{2.28}$$

and drawing the single active unit α from this distribution.

The renormalization in Eq. (2.28) plays a crucial role and it is worth being precise what it actually means. This type of one-hot encoding consists of a normalization of L independent sigmoid marginals, computed as if each unit α were still a free coin, which is only forced to sum to one "after the fact".

While starting from the most general expression in 2.2b we would obtain a softmax over the local fields,

$$p_{\text{softmax}}(\alpha \mid \mathbf{x}) = \frac{e^{\overline{h}_\alpha}}{\sum_{\beta=1}^{L} e^{\overline{h}_\beta}}, \tag{2.29}$$

where $\mathbf{z}$ is treated as L-sided die, with the L states competing directly for the same probability mass.

More formally, we can write Eq. (2.29) as the conditional probability of activating the α-th hidden unit given the global one-hot constraint from the beginning of the computation:

$$p_{\text{softmax}}(\alpha \mid \mathbf{x}) \equiv P\left(z_\alpha = 1 \;\middle|\; \sum_{\beta=1}^{L} z_\beta = 1, \mathbf{x}\right), \tag{2.30}$$

where, to be explicit, when $z_1 = 1$ we are considering the latent vector $\mathbf{z} = (1, 0, \ldots, 0)^\top$, whereas when $z_2 = 1$ we have $\mathbf{z} = (0, 1, \ldots, 0)^\top$, and so on for all L states. Under this constraint $\sum_{\beta=1}^{L} z_\beta = 1$, exactly one latent coordinate is active at any time, forcing the L units into direct competition. In contrast, Eq. (2.28) models each latent unit as an independent variable:

$$p(z_\beta = 1 \mid \mathbf{x}) = \sigma(\overline{h}_\beta), \tag{2.31}$$

which evaluates the activation of coordinate β independently of all other units, thus allowing multiple hidden variables to be simultaneously active in principle before probability normalization is applied.

A toy example with two units, having local fields $\overline{h}_1 = 2.0$ and $\overline{h}_2 = 1.0$, makes the

difference clearer. The independent sigmoids give $\mu_1 \simeq 88.1\%$ and $\mu_2 \simeq 73.1\%$, so normalizing them as in Eq. (2.28) yields $p_{\text{one}}(1|\mathbf{x}) \simeq 54.6\%$ and $p_{\text{one}}(2|\mathbf{x}) \simeq 45.4\%$, a nearly maximum-entropy split with a bias determined by the independent activation probabilities. Meanwhile, the theoretical outcome of Eq. (2.29) gives $p_{\text{softmax}}(1|\mathbf{x}) \simeq 73.1\%$ and $p_{\text{softmax}}(2|\mathbf{x}) \simeq 26.9\%$, showing a considerably stronger displacement that can easily lead to the dramatic effect of dead hidden units when the imbalance between the initial local fields h_α is larger.

Formally, Eq. (2.28) is not the correct expression, and it was originally coded as an unnoticed implementation slip. The realization came to light well after, with the project already at an advanced stage. But what we found, somewhat to our surprise, is that training with the "wrong" rule of Eq. (2.28) consistently and outrageously outperformed training with the formally correct softmax of Eq. (2.29), mainly due to how many of the L hidden units ended up actually being used by the model.

For example, even introducing a temperature parameter τ in Eq. (2.29) and applying well-known temperature annealing techniques [15, 16] could not achieve results as good as those obtained with the $p_{\text{one}}(\alpha|\mathbf{x})$ expression. Moreover, the chosen one-hot encoding brings stability to the whole model architecture, whereas a temperature annealing procedure provides some relief but it was evident that, at the end of training when $\tau \to 1$, the instability of $p_{\text{softmax}}(\alpha|\mathbf{x})$ reappeared, leading to a deterioration of the final outcome.

Typically in this kind of architecture, where the $\mathbf{x} \to \mathbf{z}$ pass learns faster than the generative step $\mathbf{z} \to \mathbf{x}$, the model is susceptible to a latent-state collapse: the sigmoid activations $\mu_{z_\alpha|\mathbf{x}}$ saturate for some dominant units α, driving $p(z_\beta = 1|\mathbf{x}) \to 0$ for all other units. This asymmetric learning dynamic is reminiscent of the generator-discriminator imbalance in Generative Adversarial Networks [17], where an overpowered discriminator deprives the generator of useful gradient signal, causing mode collapse [18]. In ssMM, the backward pass acts as a fast-converging discriminator that assigns all probability to a subset of latent units, while the forward generative pass, analogous to the GAN generator, cannot learn fast enough to counteract this tendency.

The dead-unit problem arising from one-hot (Potts-type) hidden variables has been documented also in RBM [19, 20]. RBMs with q-state Potts hidden units (the GM-RBM) collapse onto a single latent state during training unless explicit countermeasures are applied; temperature annealing and intra-slot diversity constraints as remedies were proposed [15, 16, 19].

The one-hot expression of Eq. (2.28) provides an analogous built-in regularization: by making the L marginal probabilities compete on the same level before sampling, we avoid the situation where some units take over all the activation probability. Units that have not yet specialized to a specific data pattern still get sampled often enough, which keeps their gradients alive and allows them to gradually develop their own role during training.

Given how it happened, Eq. (2.28) was not meant to be a design choice; it started out as a small implementation mistake, and only later we realize that it was actually helping the model. We do not think this kind of serendipity is particularly unusual in machine learning,

though.

A number of now-standard techniques (e.g., label smoothing [18] or the non-saturating generator loss used in the already mentioned problematic GAN training [17]) were adopted not because they follow cleanly from the underlying probabilistic model, but simply because they were seen to work, and only later were they explained in terms of the implicit regularization they provide. We regard the normalized-sigmoid rule of Eq. (2.28) in the same spirit: a practical trick that avoids the collapse that one-hot (Potts-like) hidden layers often experience in related architectures such as the RBM, while still offering a clear and straightforward interpretation of how the chosen one-hot sampling behaves, as discussed above.

2.4 Model application

We now move on to applying the model theory to a specific case. In our case, the analysis was carried out on the binarized version of the MNIST dataset [21]. In particular, we focus on a balanced subset of 4000 samples containing digits 0, 1, 2, and 3, chosen for computational reasons. Consequently, the input data $\mathbf{x}$ consist of vectors of $D = 784$ binary values (0 or 1), corresponding to the handwritten digits made up by 28×28 pixels (Fig. 2.3). The one-hot latent space is fixed at $L = 12$ hidden units, which represents a good compromise that allows our model to capture the main generative features of the original dataset distribution. We

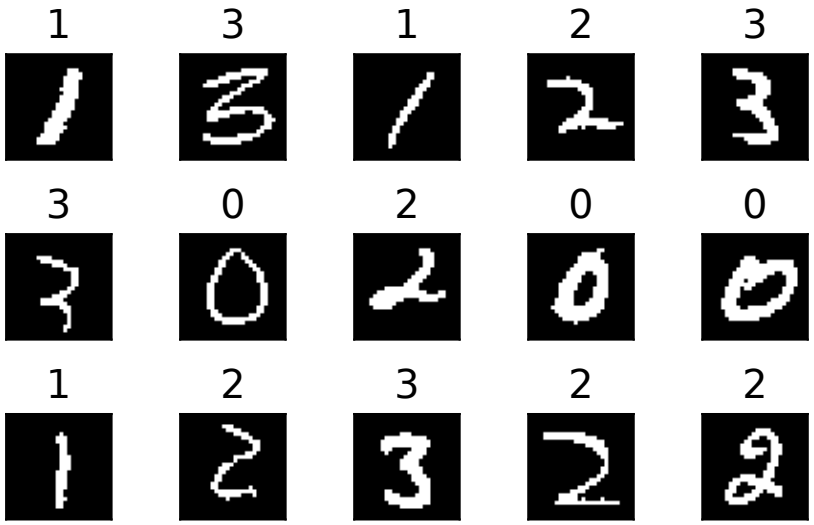

Figure 2.3: *Examples of binarized handwritten digits 0, 1, 2 and 3 from the MNIST dataset. Black pixels represent bit 0, whereas white pixels represent bit 1.*

now present an example of a ssMM model with $L = 12$ (without any regularization, as will be discussed later) trained with the hyperparameters listed in Sec. 4.1. To obtain a clearer visual comparison between the learned patterns and the MNIST dataset, we consider the case where the bias a is absent; otherwise, one would simply need to add the offset of the

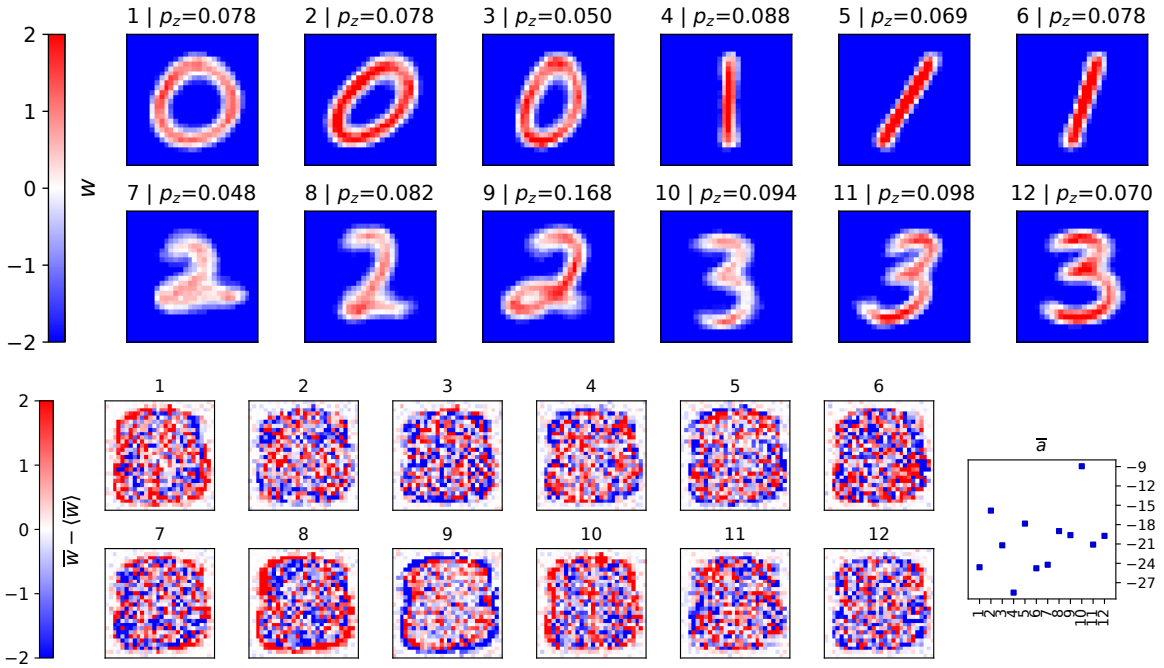

Figure 2.4: *Example of learned weights from an ssMM model with $L = 12$, namely $\alpha = 1, \ldots, 12$. The weights have been reordered for ease of presentation. For each $w_{j\alpha}$, the marginal activation probability p_z is shown at the top. The lower panel reports the deviation from the mean of the backward weights $\overline{w}_{\alpha j}$, while the right panel displays the values of the bias vector $\overline{a}$.*

bias to the learned weights[1]. The result is shown in Fig. 2.4. Not all digits are learned with the same level of accuracy. In Fig. 2.5(b), one can clearly see that simpler digits, such as the "1", are captured much more effectively by the model. Although in the following sections we will mostly refer to the average log-likelihood computed over the considered dataset, the actual performance varies considerably from image to image within the training set. This is mainly due to the fact that the generative model attempts to generalize a broad variety of handwritten styles into a small number of macrocategories.

Finally, Fig. 2.5(a) illustrates the latent space dynamics through the conditional transition matrix $M_{\mathbf{z}\mathbf{z}'}$. Even in this simple example, one can already appreciate the presence of strongly preferred transitions and the asymmetric structure of the matrix with respect to its diagonal. This behavior is directly linked to the ability of our architecture to operate out of equilibrium, a feature that plays a central role in the remainder of this work. Building on this principle, Chapter 3 details the novel regularization strategies designed to control the latent cyclic dynamics, while Chapter 4 presents the empirical results, showing significant generative enhancements over the standard ssMM.

[1]From the forward local field $h_i = w_{i\alpha} + a_i$ (valid for a one-hot encoding where the active hidden unit is the one with $z_\alpha = 1$), note that the visible bias a_i is always present regardless of which hidden unit is active. Removing it makes each matrix $w_{i\alpha}$ directly readable as the digit pattern, whereas keeping it simply requires accounting for this common offset a_i to recover a human-visible handwritten digit.

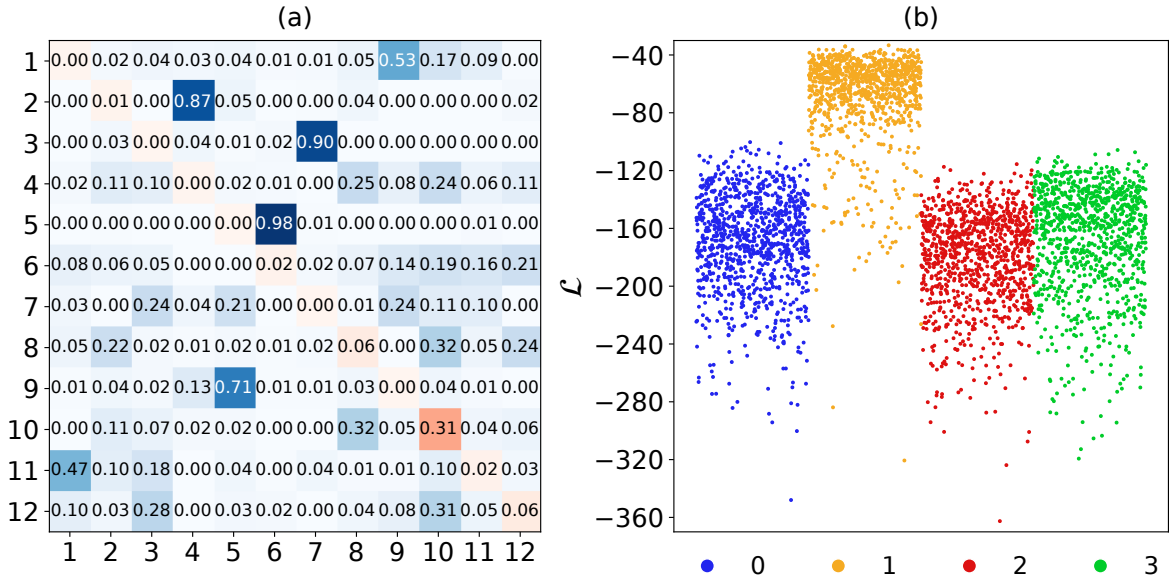

Figure 2.5: *For the model shown in Fig. 2.4: (a) conditional transition matrix $M_{zz'}$, with the diagonal highlighted in red to indicate self transitions (i.e., steps in which the latent state remains unchanged); (b) log-likelihood values for each of the 4000 images generated by the model, colored according to the corresponding digit class. The final average log-likelihood is $\mathcal{L} = -146.9 \pm 0.9$.*

Chapter 3

Regularizations

In this chapter, we discuss some regularizations for the ssMM, explaining their motivation, the effects they produce, and the conclusions we can draw from them. Rather than acting as standard regularizers to prevent overfitting, these terms are explicitly added to the log-likelihood to stimulate specific behaviors in the latent space. Through this analysis, we also gain a clearer understanding of the model's actual dynamics and why these terms are crucial for enhancing its generative performance.

3.1 Keeping alive all hidden units

We introduce a simple "spring-like" regularization term that tries to keep the mean activation probabilities of the latent units close to each other. We append to the log-likelihood a penalty that pulls each activation bias $\overline{a}$ toward the average value across all units. In practice, this tries to keep the activation probabilities of (2.2b) far from diverging too much, while still allowing the model to differentiate the latent states when needed. Hence we get:

$$\mathcal{R}_{\overline{a}} = -\frac{\lambda_{\overline{a}}}{2} \sum_{\alpha=1}^{L} \left(\overline{a}_\alpha - \langle \overline{a} \rangle\right)^2 \quad \Rightarrow \quad \nabla_{\overline{a}} \mathcal{R}_{\overline{a}} = \lambda_{\overline{a}} \left(\overline{a} - \langle \overline{a} \rangle\right) \tag{3.1}$$

Notice that, in principle, if we want to act on the activation probabilities of $\mathbf{z}$, we should modify $\overline{h}$, which also contains the parameter w. However, this would lead to much more complicated and difficult to handle mathematical expressions. On the contrary, acting only on $\overline{a}$ is more tractable, and we will show empirically that it is perfectly sufficient to achieve our goal and all L units will remain alive during training. Moreover, acting on $\overline{w}$ would mean affecting the whole structure of the interactions between the input and the latent states, while $\overline{a}$ is just an additive bias, which allows perfectly to adjust the activation level of each unit.

3.2 Avoiding return cycles after n steps

In this section, we introduce the general idea behind this type of regularization, and then we will see two specific cases where this approach turns out to be quite effective. Since our

model can operate out of equilibrium and indeed tends to do so spontaneously by developing probability flow between different latent states, our goal is to take advantage of this behaviour and encourage it even more.

The simplest way to push the model in this direction is to promote out-of-equilibrium behaviour in the latent Markov chain. To do that, we came up with the idea of penalizing self-transitions after n steps (where each step corresponds to a single latent transition $\mathbf{z}(t) \to \mathbf{z}(t+1)$ governed by $M_{\mathbf{z}\mathbf{z}'}$, as shown in Fig. 2.2(b)). In practice, this means forcing the system not to stay stuck in the same latent state, but instead to move around and explore the others.

To implement this idea, we define a term that measures how on average the chain returns to the same state after n steps along a trajectory of length T. This term is then used as a regularization penalty to discourage self-return cycles and promote more dynamic latent-space exploration. We can express the amount of self-returns after n steps as:

$$\frac{1}{T-n}\sum_{t=1}^{T-n}\sum_{\alpha=1}^{L} z_\alpha(t)\mu_\alpha(t+n) \tag{3.2}$$

where $z_\alpha(t)$ is the hidden state sampled at time t and $\mu_\alpha(t+n) = P(z_\alpha(t+n) = 1 \mid x(t+n))$ is the mean activation probability for the hidden state α after n steps, calculated by the sigmoid function $\mu_\alpha(t+n) = \sigma(\sum_i \overline{w}_{\alpha i} x_i(t+n) + \overline{a}_\alpha)$.

Hence we define the following regularization term to be added to the log-likelihood:

$$\mathcal{R}_n = -\frac{\lambda_n}{T-n}\sum_{t=1}^{T-n}\sum_{\alpha=1}^{L} z_\alpha(t)\mu_\alpha(t+n) \tag{3.3}$$

Notice that as we wrote it, when $\lambda_n > 0$ self-return loops are penalized. Conversely, when $\lambda_n < 0$, the regularization favors self-return loops after n steps. The latter will be useful in tests, to check that increasing self-returns hinders learning.

In the computation of the gradient we employ the fact that, once the initial state $z_\alpha(t)$ is sampled, it is treated as a constant scalar and we do not need to compute its derivative. We solely differentiate the expectation of the arrival state, $\mu_\alpha(t+n)$, with respect to the backward parameters ($\overline{w}_{i\alpha}$ and $\overline{a}_\alpha$).

First, we define the auxiliary term $\Delta_\alpha(t)$ (which is essentially almost the derivative of the sigmoid for the mean activation probability):

$$\Delta_\alpha(t) \equiv z_\alpha(t)\mu_\alpha(t+n)\left(1-\mu_\alpha(t+n)\right) \tag{3.4}$$

Hence the regularization gradients become:[1]

$$\left(\nabla_{\overline{w}}\mathcal{R}_n\right)_{\alpha j} = -\frac{\lambda_n}{T-n}\sum_{t=1}^{T-n}\Delta_\alpha(t)\otimes x_j(t+n) \tag{3.5a}$$

$$\left(\nabla_{\overline{a}}\mathcal{R}_n\right)_{\alpha} = -\frac{\lambda_n}{T-n}\sum_{t=1}^{T-n}\Delta_\alpha(t) \tag{3.5b}$$

We will apply the general idea introduced in this Section 3.2 to two specific cases. First, the $n = 1$ case, which corresponds to avoiding the self-transition, where a latent state simply moves to itself in the next step. Then we examine the case $n = 2$, which means avoiding the so-called "ping-pong" effect, where a state keeps bouncing back and forth between itself and another state. This second restriction leads to even better results.

3.2.1 Generalization to Bernoulli

The regularization term seeks to penalize the expected number of times the state $\mathbf{z}$ repeats exactly from step t to $t+n$. In the general case, this occurs when all L components of the vector $\mathbf{z}$ match perfectly. In a non–one-hot setting, a scenario we refer to as Bernoulli, the L coordinates are genuinely independent (the units previously referred to as hidden units), and they can be active simultaneously. Hence, the joint probability that the entire state repeats is simply the product of the individual bit-matching probabilities.

For a single coordinate/bit $\alpha \in \{1,\dots,L\}$, we know the sampled previous state $z_\alpha(t) \in \{0,1\}$ and the expected arrival probability $\mu_\alpha(t+n) \in [0,1]$.

The probability $q_\alpha(t)$ that the bit at $t+n$ exactly matches $z_\alpha(t)$ is:

$$q_\alpha(t) = P[z_\alpha(t+n) = z_\alpha(t)] = \begin{cases} \mu_\alpha(t+n) & \text{if } z_\alpha(t) = 1 \\ 1-\mu_\alpha(t+n) & \text{if } z_\alpha(t) = 0 \end{cases}$$

This can be compactly written as:

$$q_\alpha(t) = z_\alpha(t)\mu_\alpha(t+n) + (1-z_\alpha(t))(1-\mu_\alpha(t+n)) \tag{3.6}$$

Because the hidden units are conditionally independent given $x(t+n)$, the joint probability $P_{\text{match}}(t)$ that all L bits match is the product of their individual probabilities:

$$P_{\text{match}}(t) = \prod_{\alpha=1}^{L} q_\alpha(t) \tag{3.7}$$

To update the parameters of the model, we show how to calculate the derivative of $P_{\text{match}}(t)$

[1] Here, $\otimes$ denotes the outer product used to construct the gradient matrix with respect to the backward weights, obtained by multiplying the vector $\Delta_\alpha(t)$ by the delayed input $x_j(t+n)$ (treated as a row vector).

with respect to the backward weights $\overline{w}_{\alpha j}$. By the chain rule:

$$\frac{\partial P_{\text{match}}(t)}{\partial \overline{w}_{\alpha j}} = \frac{\partial P_{\text{match}}(t)}{\partial \mu_\alpha(t+n)} \cdot \frac{\partial \mu_\alpha(t+n)}{\partial h_\alpha(t+n)} \cdot \frac{\partial h_\alpha(t+n)}{\partial \overline{w}_{\alpha j}} \tag{3.8}$$

Let us now focus on the first factor, namely $\frac{\partial P_{\text{match}}}{\partial \mu_\alpha}$. By applying the logarithmic derivative property and recalling the definition in Eq. 3.7, we obtain:

$$\frac{\partial P_{\text{match}}}{\partial \mu_\alpha} = P_{\text{match}} \frac{\partial \ln P_{\text{match}}}{\partial \mu_\alpha} = P_{\text{match}} \frac{\partial}{\partial \mu_\alpha} \sum_{\gamma=1}^{L} \ln q_\gamma \tag{3.9}$$

Since only q_α depends on μ_α

$$\frac{\partial P_{\text{match}}}{\partial \mu_\alpha} = P_{\text{match}} \frac{1}{q_\alpha} \frac{\partial q_\alpha}{\partial \mu_\alpha} = P_{\text{match}} \frac{z_\alpha - (1 - z_\alpha)}{q_\alpha} = P_{\text{match}} \frac{2z_\alpha - 1}{q_\alpha} \tag{3.10}$$

Considering that $z_\alpha \in \{0, 1\}$, we can rewrite the algebraic fraction $\frac{2z_\alpha - 1}{q_\alpha}$ into an equivalent form:

$$\frac{2z_\alpha - 1}{q_\alpha} = \frac{z_\alpha - \mu_\alpha}{\mu_\alpha(1 - \mu_\alpha)} \tag{3.11}$$

We can prove the equivalence in Eq. 3.11 by evaluating its Left Hand Side (LHS) and the Right Hand Side (RHS) in the two exact discrete cases.

Case 2: $z_\alpha = 0$
Substituting $z_\alpha = 0$ into expression 3.6 yields $q_\alpha = (0)\mu_\alpha + (1)(1 - \mu_\alpha) = 1 - \mu_\alpha$, hence

$$\text{LHS} = \frac{2(0) - 1}{q_\alpha} = \frac{-1}{1 - \mu_\alpha}$$

$$\text{RHS} = \frac{0 - \mu_\alpha}{\mu_\alpha(1 - \mu_\alpha)} = \frac{-\mu_\alpha}{\mu_\alpha(1 - \mu_\alpha)} = \frac{-1}{1 - \mu_\alpha}$$

Case 1: $z_\alpha = 1$
Substituting $z_\alpha = 1$ into expression 3.6 yields $q_\alpha = (1)\mu_\alpha + (0)(1 - \mu_\alpha) = \mu_\alpha$, therefore

$$\text{LHS} = \frac{2(1) - 1}{q_\alpha} = \frac{1}{\mu_\alpha}$$

$$\text{RHS} = \frac{1 - \mu_\alpha}{\mu_\alpha(1 - \mu_\alpha)} = \frac{1}{\mu_\alpha}$$

This shows that the equality holds.

Thus:

$$\frac{\partial P_{\text{match}}(t)}{\partial \mu_\alpha(t+n)} = P_{\text{match}}(t) \frac{z_\alpha(t) - \mu_\alpha(t+n)}{\mu_\alpha(t+n)(1 - \mu_\alpha(t+n))} \tag{3.12}$$

Moreover, for $\mu_\alpha = \sigma(h_\alpha)$ we have $\frac{\partial \mu_\alpha(t+n)}{\partial h_\alpha(t+n)} = \mu_\alpha(t+n)(1 - \mu_\alpha(t+n))$.

Now, combining these results together, the denominator from the joint probability exactly cancels out with the derivative of the sigmoid function:

$$\begin{aligned}\frac{\partial P_{\text{match}}(t)}{\partial h_\alpha(t+n)} &= \left[P_{\text{match}}(t)\frac{z_\alpha(t)-\mu_\alpha(t+n)}{\mu_\alpha(t+n)(1-\mu_\alpha(t+n))}\right]\cdot\Big[\mu_\alpha(t+n)(1-\mu_\alpha(t+n))\Big] \\ &= P_{\text{match}}(t)\big(z_\alpha(t)-\mu_\alpha(t+n)\big)\end{aligned} \tag{3.13}$$

We define this specific quantity as our new generalized $\Delta_\alpha(t)$:[2]

$$\Delta_\alpha(t) \equiv P_{\text{match}}(t)\big(z_\alpha(t)-\mu_\alpha(t+n)\big) \tag{3.14}$$

Finally, multiplying by the derivative of the field $\frac{\partial h_\alpha}{\partial \overline{w}_{\alpha j}} = x_j(t+n)$, the original Eq. 3.8 can be written simply as[3]

$$\frac{\partial P_{\text{match}}(t)}{\partial \overline{w}_{\alpha j}} = \Delta_\alpha(t)\otimes x_j(t+n) \tag{3.15}$$

In the Bernoulli case, the total regularization term is defined as proportional to the empirical average of the joint matching probabilities over the entire trajectory:

$$\mathcal{R}_n = -\frac{\lambda_n}{T-n}\sum_{t=1}^{T-n} P_{\text{match}}(t) \tag{3.16}$$

Recalling that, for the derivative with respect to the backward bias $\overline{a}_\alpha$, the last factor in Eq. 3.8 would simply become $\frac{\partial h_\alpha}{\partial \overline{a}_\alpha} = 1$, the final regularization gradients for both backward parameters $\overline{w}$ and $\overline{a}$ are:[3]

$$\big(\nabla_{\overline{w}}\mathcal{R}_n\big)_{\alpha j} = -\frac{\lambda_n}{T-n}\sum_{t=1}^{T-n}\Delta_\alpha(t)\otimes x_j(t+n) \tag{3.17a}$$

$$\big(\nabla_{\overline{a}}\mathcal{R}_n\big)_{\alpha} = -\frac{\lambda_n}{T-n}\sum_{t=1}^{T-n}\Delta_\alpha(t) \tag{3.17b}$$

[2]Note that the multiplicative factor $P_{\text{match}}(t)$ follows directly from the log-derivative identity used in Eq. 3.9: it weights the gradient by how likely a full return of the hidden vector actually is, so small values correctly indicate an already unlikely event. Since P_{match} is the product of L such probabilities, it decreases geometrically with L and can damp the gradient for large hidden layers, which is why the Bernoulli generalization is restricted to $L = 5$ in Sec. 4.4.

[3]As in Eq. 3.5a, the operator $\otimes$ denotes the outer product.

Chapter 4

Results

We now proceed to show the results obtained for the Markov Machine model, focusing specifically on the effects of the regularizations introduced in Chapter 3.

4.1 Methodology

We train the models for 150 epochs, using 25 mini-batches per epoch. During training, the mini-batch size increases progressively: we start with 20 images per mini-batch and go up to 100 images. This balances initial exploratory gradient noise with subsequent stability in parameter updates as the model approaches convergence. The length of the Markov chain trajectories is set to $T = 100$ steps during training, and we compute the final statistics and quantities using a long run of $T = 10^5$. Finally, we use RMSprop [22] for gradient optimization with a learning rate of 0.1.
For the regularization constants $\lambda_{\overline{a}}$ of Section 3.1 and λ_1, λ_2 for Section 3.2 (respectively concerning the return cycles after $n = 1$ step and after $n = 2$ steps), we show the results obtained using values that produced a modification of the gradients of the parameters $\overline{w}$ and $\overline{a}$ that yields the best outcome while not exhibiting problematic behavior (more about this in Sec. 4.2.3).
In the end, for each specific hyperparameter configuration, 100 models were trained for statistical significance.

4.2 Effects of regularization

First, we highlight the most significant effect obtained for each type of regularization, and then we proceed with the overall comparison from which we draw the final conclusions. Throughout the subsequent analysis and in all corresponding plots, the regularizations explicitly indicated in the legend specify the active terms employed during training, with their associated parameters λ. Any regularization term omitted from the legend is implicitly inactive (i.e., its corresponding λ is set to zero). In particular, the configuration labeled as "normal" denotes the baseline model introduced in Chapter 2.

4.2.1 Use of hidden units

What emerged as fundamental is the ability to keep all the available latent units alive, meaning with a marginal activation probability strictly greater than zero. This is a rather intuitive result: the performance is better when the system makes use of all the resources it has been given. However, the ssMM alone does not guarantee this kind of behaviour, since in roughly one third of the cases not all hidden units were actually used (Fig. 4.1), bringing us back to the issue already known in the literature and mentioned in Section 2.3. All the regularizations used individually improve this aspect, as shown in Fig. 4.1, but the one that proved to have the most substantial impact is the "spring" regularization (associated to $\lambda_{\overline{a}}$ in the plots). It performs remarkably well for the purpose of imposing a bounded constraint on $\overline{h}$ and therefore on $p(\mathbf{z})$; this leads to a 100% success rate in avoiding the presence of dead units, which typically appear during the early stages of training.

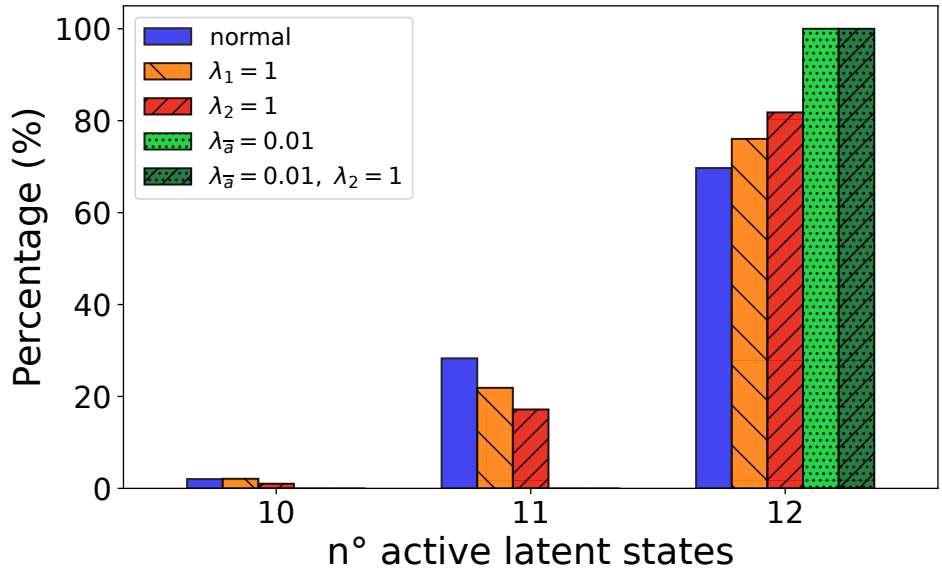

Figure 4.1: *Percentages of usage of the $L = 12$ hidden states for different regularizations. A hidden state is considered active when its marginal probability exceeds the arbitrary threshold of* 0.3%. *Since there are 12 states, if they were visited uniformly the marginal probability would tend to* $1/12 \simeq 8.3\%$.

Moreover, what is crucial beyond simply using all the latent space is to use it in a balanced way, where all the hidden units tend to be employed with similar probability. This is clearly shown by looking at how the Shannon entropy of the $p(z)$ at the end of training is directly correlated with better performances (Fig. 4.2(a)). In particular, the best models are those where $\mathrm{H}[p_z]$ stabilizes around a certain threshold during training; this behaviour is achieved thanks to the regularization associated with $\lambda_{\overline{a}}$, and partly also through the regularization that prevents the "ping-pong" effect linked to λ_2 (Fig. 4.2(b)).

4.2.2 Return cycles regularization

Now we examine the specific effect of the regularizations introduced in Section 3.2, which act directly on the Markov chain in the latent space.

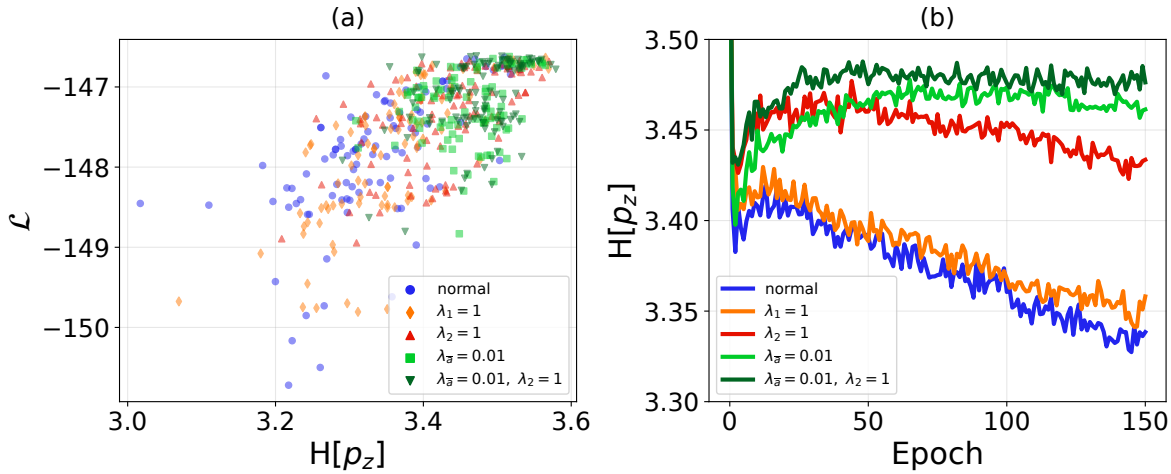

Figure 4.2: *(a) Plot showing the direct correlation between the log-likelihood and the Shannon entropy of the $p(z)$ associated with the $L = 12$ units. (b) Evolution dynamics of $\mathrm{H}[p_z]$ over 100 independent training runs for each different regularizations. Notice how all the hidden units start with equal probability at the beginning (namely $\mathrm{H}[p_z] = \log_2(12) \simeq 3.58$), and how they then evolve with significantly different trends. In particular, the normal ssMM model tends to favour the development of a lower entropy* H*, a behaviour that is also connected to the dead unit issue shown in Fig. 4.1.*

Penalizing 1-step self-transitions (λ_1). The greater flexibility of the architecture provided by the ssMM allows these models to naturally operate out of equilibrium, with an entropy production σ (used as a measure of the breaking of the detailed balance condition) that is strictly greater than zero. Before proceeding, let us recall how entropy production is defined in this framework (with $M_{\mathbf{z}\mathbf{z}'}$ defined in Eq. (2.4)):

$$\sigma = \sum_{z'<z} \left[p(z')M_{z'z} - p(z)M_{zz'}\right] \ln \frac{p(z')M_{z'z}}{p(z)M_{zz'}}, \tag{4.1}$$

Trying to take advantage of this feature, and with the idea that a higher ergodicity of the system should help the learning process, we observe that depending on the sign of λ_1 we obtain a peculiar relation between $\mathcal{L}$ and σ, occupying predominantly the upper triangular region in Fig. 4.3. What emerges is a curved-shaped relationship in which the main statement we can assess is the following: when the model works strongly out of equilibrium, this guarantees that it does not fall into the case of giving a bad model; on the contrary, if the model is in a near-equilibrium condition, we are not guaranteed to avoid the bad case of a poor model. In fact, among the bad models, almost all of them have low σ, while none of them show high σ. Enforcing this penalty with $\lambda_1 = 2$ effectively shifts the training dynamics away from this low-σ, low-$\mathcal{L}$ failure mode, ensuring consistently higher likelihoods. Conversely, setting $\lambda_1 = -2$ significantly increases the risk of suboptimal performance, as the model is incentivized to remain trapped in the same latent state across consecutive time steps; an illustrative example of the resulting diagonal dominated transition matrix $M_{\mathbf{z}\mathbf{z}'}$ for this scenario is shown in Fig. 4.4.

Finally, notice how $\mathcal{L}$ tends to saturate around a certain threshold that seems impossible

to go beyond (this can be appreciated by looking at the projection distribution in Fig. 4.5(a)), we will return to this matter in the following sections.

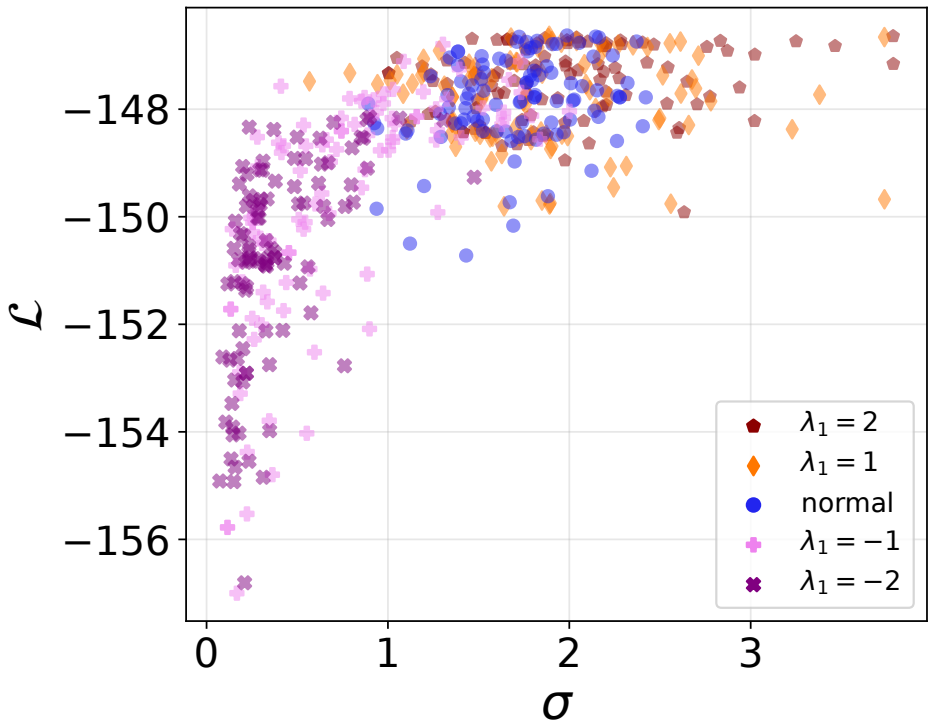

Figure 4.3: *Log-likelihood versus entropy production of the Markov chain. The models distribute themselves along a characteristic curved pattern, where those with negative λ_1 (favoured self-transition) show a σ close to zero and a clearly worse performance, while the models with positive λ_1 (discouraged self-transition) display a larger σ compared to the baseline models, together with a slightly better $\mathcal{L}$ (see Fig. 4.5).*

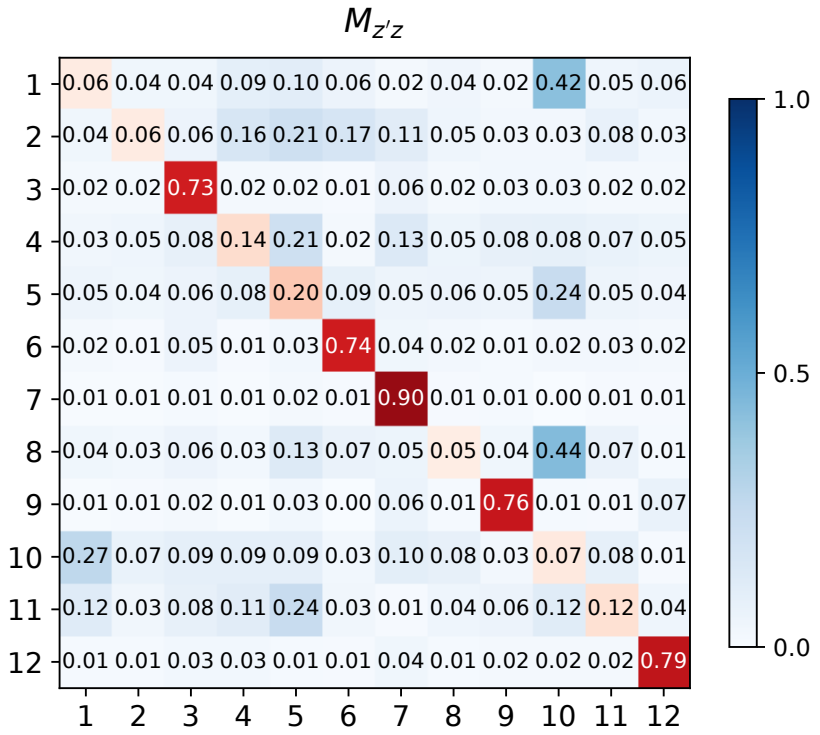

Figure 4.4: *Example of a conditional transition matrix from a model with $\lambda_1 = -2$, showing both low σ and low $\mathcal{L}$. Emerges how favouring self-transition leads to a diagonal (highlighted in red) where, for 5 latent states out of 12, the probability of remaining in the same state at the next step exceeds 50%.*

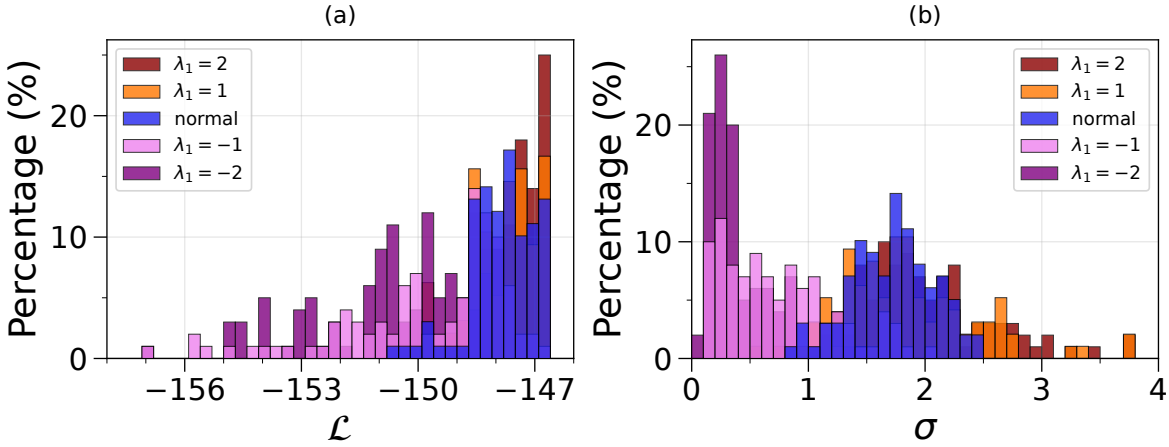

Figure 4.5: *(a) Comparative histogram of the log-likelihood for each value of λ_1 (the normal case is the one with $\lambda_1 = 0$). Negative λ_1 values lead to a noticeable worsening, while positive λ_1 values produce a slight improvement with a small predominance of better performing models. (b) Comparative histogram of the entropy production for each value of λ_1. An asymmetric effect arises with positive λ_1, which spreads the distribution by extending the right tail toward higher values, while for negative λ_1 the distribution is shifted and clearly concentrated around low values, essentially towards zero.*

Penalizing 2-step return cycles (λ_2). Regarding the two–step regularization, in the case $\lambda_2 < 0$ (i.e., favouring the creation of closed two–state cycles $z \leftrightarrow z'$), it is easy to understand how this may lead, in the long run, to an equilibrium in the exploration flows of the latent space. This is not the intended behaviour, and it also creates an additional issue: the regularizations introduced in Section 3.2 were designed for positive values of λ_n, since they are based on the amount of return cycles after n steps. If these returns are encouraged (by setting $\lambda_n < 0$), their frequency naturally grows throughout training; as a result, the regularization term keeps increasing and eventually compromises the learning process.

Focusing on the case $\lambda_2 > 0$, its positive impact on non-equilibrium dynamics is not immediately obvious. Although suppressing 2-step cycles discourages "ping-pong" oscillations, it does not explicitly penalize 1-step self-transitions ($\mathbf{z} \to \mathbf{z}$). In principle, the chain could avoid 2-step returns simply by remaining stuck in the same state, which, as seen before, represents an equally undesirable outcome.

However, empirical results in Fig. 4.6 reveal that this regularization simultaneously suppresses both effects. While the reduction of 2-step loops $\langle M^2_{\mathbf{zz}} \rangle$ in Fig. 4.6(b) is expected by construction, the decrease in 1-step self-transitions $\langle M_{\mathbf{zz}} \rangle$ in Fig. 4.6(a) is a notable emergent property. Intuitively, when 2-step returns are penalized, the system finds it more favorable to use longer exploratory cycles of length $n \geq 3$ rather than remaining trapped in a self-loop. In this way, setting $\lambda_2 > 0$ effectively combines the benefits of two penalties, strengthening the out-of-equilibrium dynamics and improving generative performance.

Conversely, when favoring two-state cycles ($\lambda_2 = -1$), trapping the trajectory into 2-step loops naturally yields lower diagonal values $\langle M_{\mathbf{zz}} \rangle$ compared to the baseline ssMM.

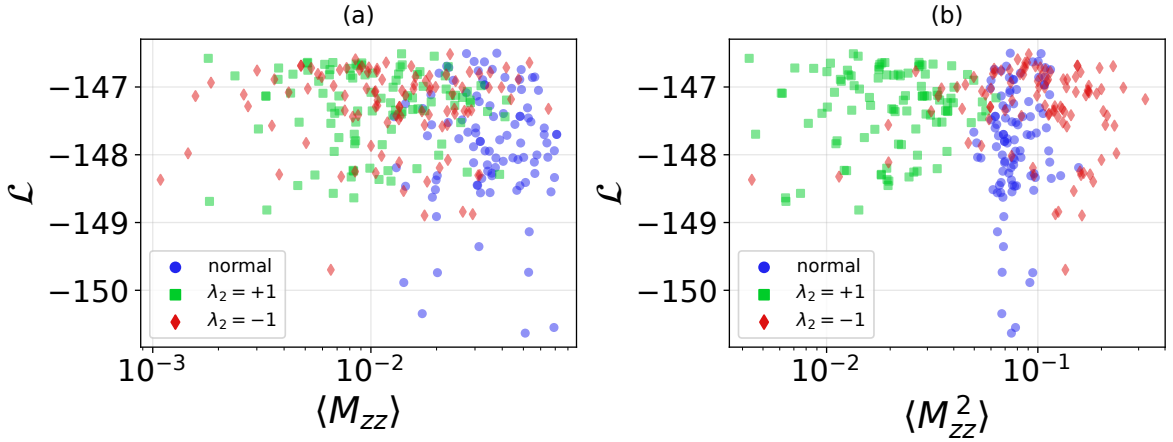

Figure 4.6: *Effects of the 2–step return cycle regularization. When $\lambda_2 = +1$, the model is discouraged from falling into "ping–pong" transitions, whereas $\lambda_2 = -1$ tends to encourage it. (a) shows $\mathcal{L}$ with respect to the average conditional probability of remaining in the same initial state after one step, (b) reports $\mathcal{L}$ with respect to the average conditional probability of staying in the same state after two steps.*

4.2.3 Regularization strength selection

In order to choose appropriate values for the λs, besides looking at the final value of $\mathcal{L}$, we carried out an analysis to understand how much the considered regularizations actually influence the learning process. Here we focus on $\lambda_{\overline{a}}$ and λ_2, which are the two regularizations that lead to the best overall performance (see Tab. 4.1). To do this, we monitored the percentage variation that these regularizations introduce in the L_1 norms of the gradient vectors of the parameters, the outcome is shown in Fig. 4.7.

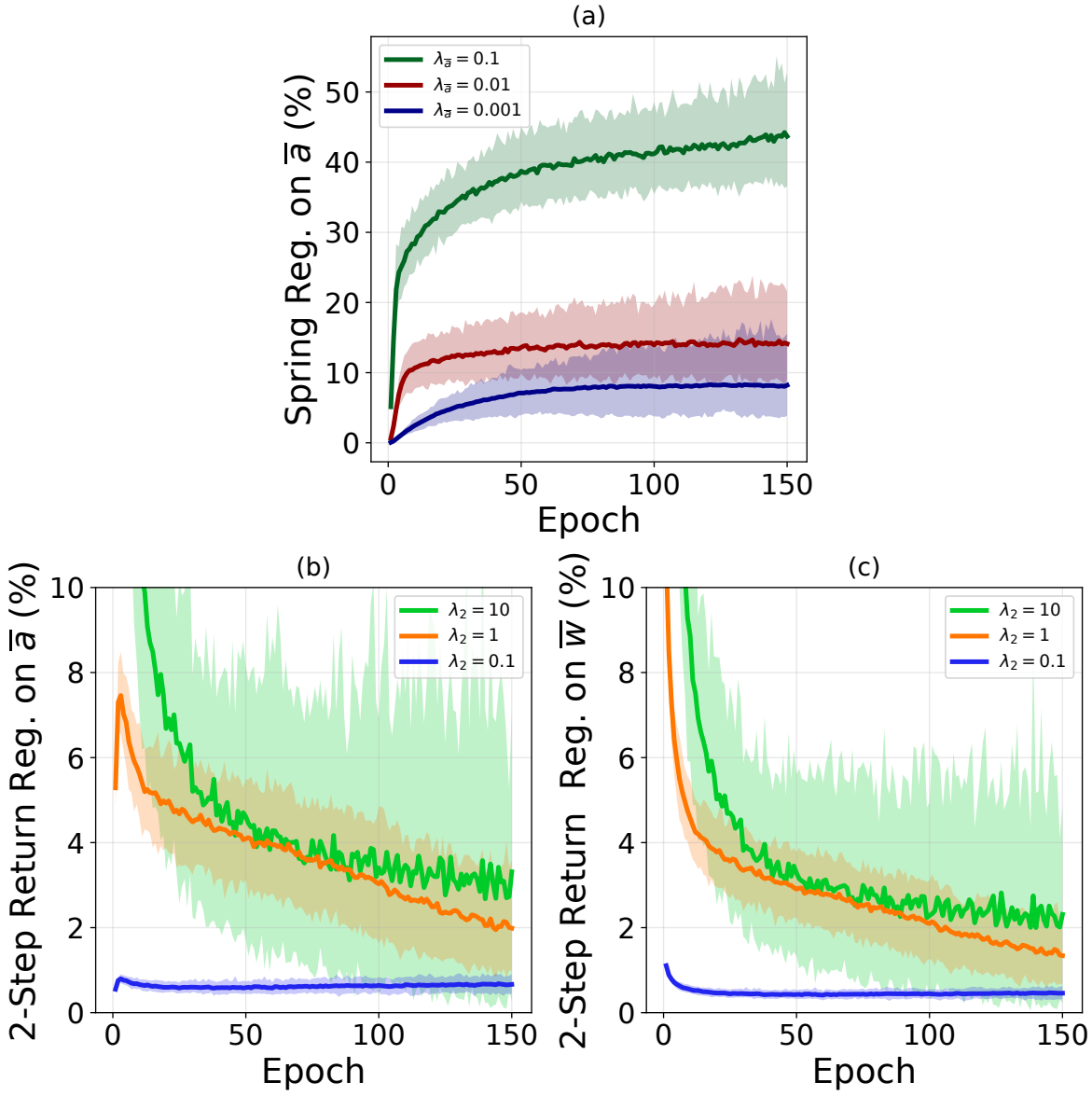

Figure 4.7: *Percentage impact of the regularization terms on the L_1 norm of the parameter gradients during training. The dark solid line shows the average trend, while the colored bands indicate the 90% dispersion band (5th to 95th percentiles) across the 100 models trained for each regularization level. In panel (a), the effect of the regularization with $\lambda_{\overline{a}}$ on the backward bias is shown. Below, the influence of the regularization with λ_2 is reported for the backward bias (b) and for the backward weight (c).*

In terms of log-likelihood, for $\lambda_{\bar{a}}$ we did not observe a large difference across the tested values. This is probably because even for small values of $\lambda_{\bar{a}}$ the correction already reaches relatively high levels, around 10% of the L_1 norm of the parameter gradients. Moreover, for larger values such as $\lambda_{\bar{a}} = 0.1$, the trend does not settle at a stable level but instead shows an undesired divergent behaviour. For this reason, we decided to keep the largest regularization strength that did not lead to divergence.

It is interesting to note, as shown in Fig. 4.7(a), that the curve has a concave shape and tends to stabilise around a certain value. This suggests that this type of regularization needs to remain active throughout training, acting like a spring that keeps the activation probabilities p_z close to each other, helping to contain the dead-unit collapse that can leave some latent states essentially unused (Sec. 2.3).

On the other hand, λ_2 affects both the backward bias (Fig. 4.7(b)) and the backward weight (Fig. 4.7(c)). The correction trend for the two parameters is similar, and in this case the curve is convex and gradually decreases as the epochs progress. This happens because, as the correction takes effect, the amount of self-returns after 2 steps computed with Eq. 3.2 tends to vanish, leading to a progressive disappearance of the regularization once the model has stabilised its probability flows in the latent space.

In this case we chose $\lambda_2 = 1$, since it unexpectedly produced better results in terms of $\mathcal{L}$ (mean -147.35, standard deviation 0.58) compared to the values obtained with $\lambda_2 = 10$ (mean -147.96, standard deviation 0.97), where one can also see in the figure a noticeable increase in the variability of the correction level on the L_1 norms.

4.2.4 Comprehensive comparison

We now present a unified comparative analysis of the various regularized models, focusing on positive values of λ aimed at optimizing generative performance.

Once again, in Fig. 4.8 the relation between $\mathcal{L}$ and σ shows a curve-shaped trend, where only the upper triangle is actually populated, and with no model that is strongly out of equilibrium ending up with a poor performance. Note that the σ shown in Fig. 4.8 is the final value obtained at the end of the training procedure. Furthermore, it is interesting to observe how, in the best models (those using $\lambda_{\bar{a}}$ and λ_2), the emergence of preferred flows in the latent space follows a steadily increasing trend during training compared to the normal model (Fig. 4.9). Over time, this helps create preferred transition paths between latent states, making the dynamics more directional and helping the model explore the latent space more effectively.

An important aspect to underline, as shown in Fig. 4.10(a), is which models actually emerge as the best ones. Without any regularization, the ssMM display a broad spread of final outcomes, with $\mathcal{L}$ ranging roughly from -151 to -146.5, and a distribution that remains quite diffuse, without any clear concentration. As we progressively introduce the different regularizations, we start observing the appearance of maxima in the log-likelihood landscape. For the top models, the entire region corresponding to the long left tail disappears, and the

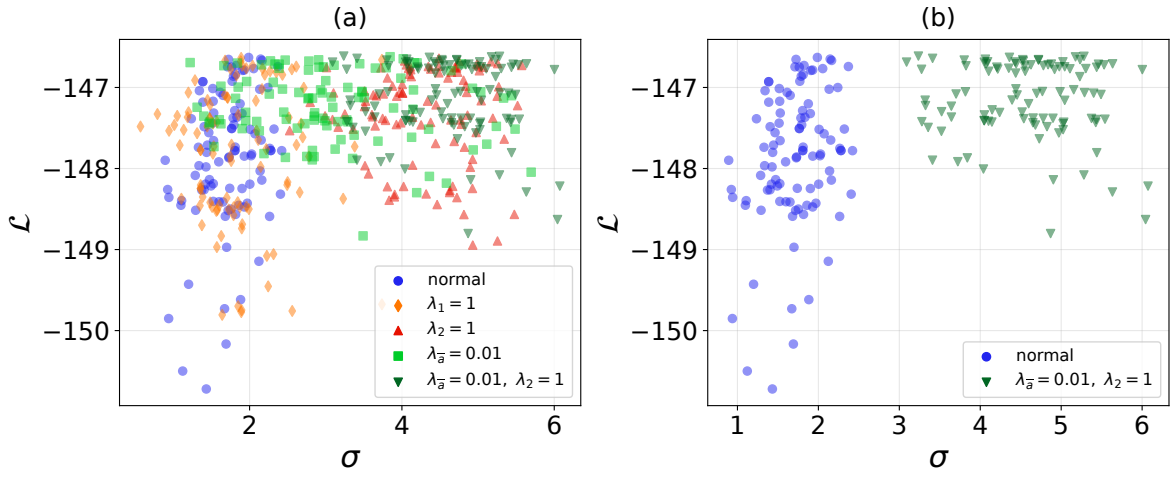

Figure 4.8: *(a) Positioning of models with different types of regularization in the log-likelihood versus entropy production plane. (b) Highlighting the different regions occupied by the normal ssMM and by the best-performing model, namely the one using both $\lambda_{\bar{a}}$ and λ_2, which leads to a strongly out-of-equilibrium behaviour.*

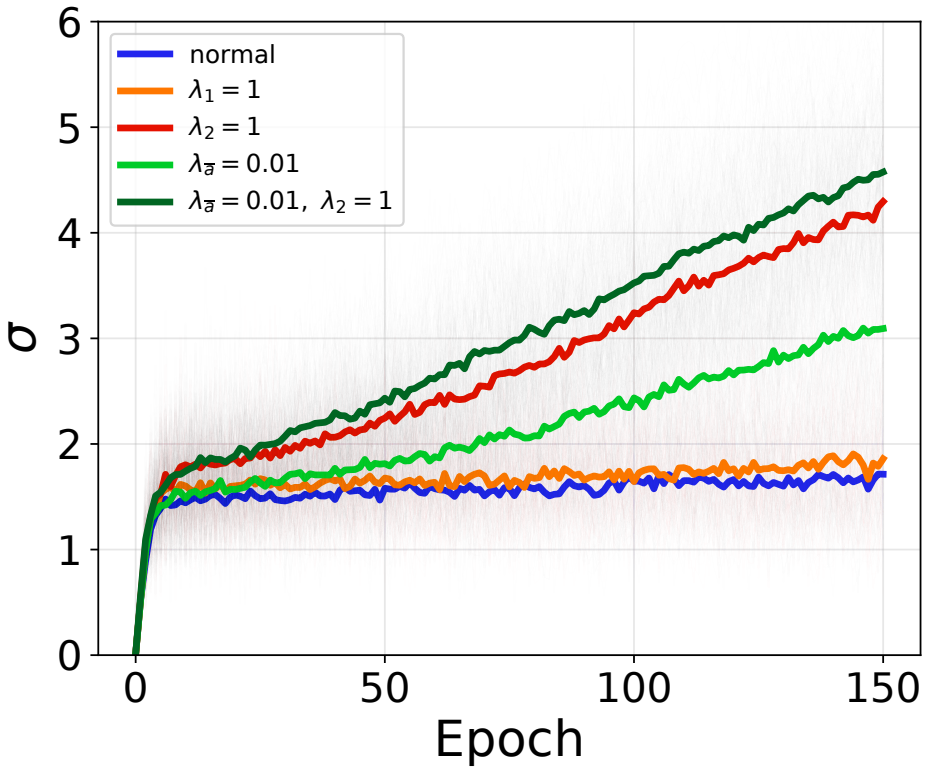

Figure 4.9: *Average trend of σ for the different regularizations, shown with thick solid lines in various colours, while the paths of each individual model are plotted with lower opacity.*

solutions cluster into three distinct peaks in the higher part of the spectrum. Still, none of these models manage to surpass the -146.5 threshold , which seems to act as a structural limit imposed by the ssMM architecture itself or maybe it is simply the best achievable performance when only $L = 12$ hidden units are available. The average value of $\mathcal{L}$ improves significantly, and the variability with which the model reaches a better result also decreases, since the

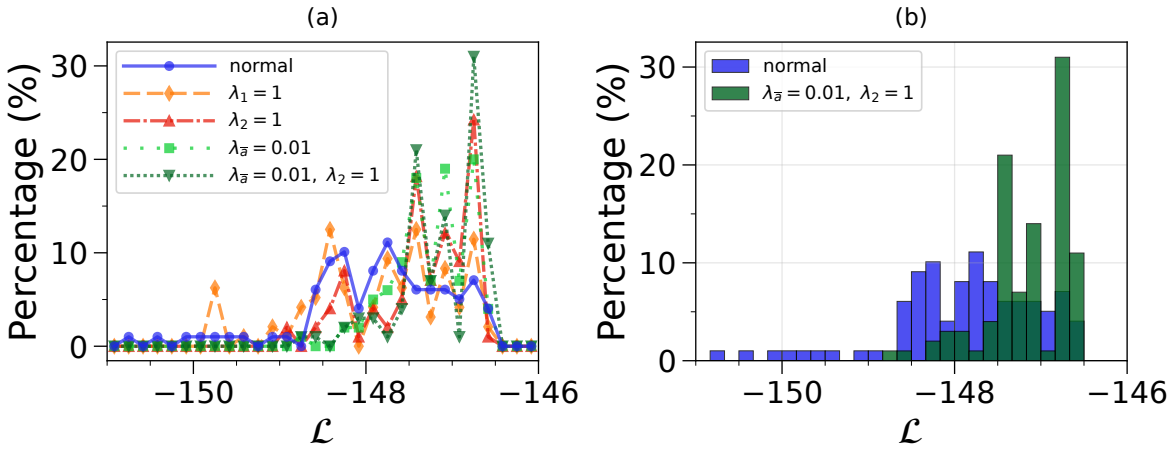

Figure 4.10: *Comparative histograms of the log-likelihood. The regularizations induce the models to cluster around specific peaks, in contrast with the normal ssMM, which instead shows a more spread out distribution with an undesired long tail on the left.*

standard deviation over the 100 runs becomes smaller (Tab. 4.1). This behaviour is mainly due to a more uniform activation of the hidden units. In particular, the regularizations help avoid those unlucky situations in which a few hidden units dominate the dynamics while others remain essentially unused, as suggested by the minimum values of $\mathrm{H}[p_z]$ observed across the different cases (Tab. 4.2). Finally, the out-of-equilibrium measure of the model provides a useful guarantee for assessing whether poor solutions are effectively avoided. In particular, the baseline ssMM and the optimized variants operate in clearly distinct σ regimes, as illustrated in Fig. 4.10(b) and supported by the values reported in Tab. 4.3.

	mean	**sd**	**min**	**max**
normal	-147.86 ± 0.09	0.9	-150.72	-146.63
$\lambda_1 = 1$	-147.84 ± 0.08	0.8	-149.81	-146.64
$\lambda_2 = 1$	-147.35 ± 0.06	0.6	-148.94	-146.64
$\lambda_{\bar{a}} = 0.01$	-147.24 ± 0.04	0.4	-148.83	-146.62
$\lambda_{\bar{a}} = 0.01,\ \lambda_2 = 1$	-147.14 ± 0.05	0.5	-148.80	-146.61

Table 4.1: *Descriptive statistics for the LogLikelihood $\mathcal{L}$ (one-hot encoding).*

	mean	**sd**	**min**	**max**
normal	3.338 ± 0.009	0.09	3.017	3.530
$\lambda_1 = 1$	3.358 ± 0.009	0.09	3.069	3.569
$\lambda_2 = 1$	3.434 ± 0.008	0.08	3.208	3.570
$\lambda_{\bar{a}} = 0.01$	3.461 ± 0.005	0.05	3.309	3.565
$\lambda_{\bar{a}} = 0.01,\ \lambda_2 = 1$	3.476 ± 0.006	0.06	3.291	3.579

Table 4.2: *Descriptive statistics for the latent space Shannon entropy $\mathrm{H}[p_z]$.*

	mean	sd	min	max
normal	1.71 ± 0.04	0.4	0.89	2.42
$\lambda_1 = 1$	1.86 ± 0.06	0.6	0.57	3.74
$\lambda_2 = 1$	4.30 ± 0.07	0.7	2.53	5.57
$\lambda_{\bar{a}} = 0.01$	3.09 ± 0.11	1.1	1.22	5.70
$\lambda_{\bar{a}} = 0.01$, $\lambda_2 = 1$	4.58 ± 0.07	0.7	3.09	6.07

Table 4.3: *Descriptive statistics for the entropy production σ.*

4.3 Learned patterns

We now move on to examining the images that our generative model has learned in order to categorize the dataset. To do this, we report below the learned weights w that appear in the three distinct peaks shown in Fig. 4.10. For each of these regions, we display the five best models, where the weights of different models have been aligned using the Hungarian algorithm [23].

More precisely, the 28×28 matrices of $w_{i\alpha}$ from all models were first normalised. Then, for each region, the best model was taken as a reference. For every other model, considered in pair with the reference one, we computed the dot product between all possible combinations of the weights, obtaining a matrix C of size $L \times L$. Since the weights were normalised beforehand, C corresponds to the cosine similarity metric. Because the Hungarian algorithm is natively formulated to solve a cost minimization problem, we applied it to $-C$ to find the optimal one-to-one matching that maximizes the total pairwise similarity between the learned features. In fact, more negative values of $-C$ indicate that two weight matrices overlap more strongly (i.e., they have their active bits in the same positions). We chose the Hungarian algorithm as it provides the most efficient exact method for reordering a set of L elements with respect to another set of L elements once a comparison metric is established.

Fig. 4.11 shows our best case, occurring when the model distributes its available resources evenly across the digits, which is exactly what we expected. Suboptimal alternatives appear in Fig. 4.12 and Fig. 4.13, where one hidden unit is taken away from learning the digit "1" in favour of digits where the model recognises more variability, such as "0" and "2", respectively. This behaviour could already be anticipated from Fig. 2.5(b), where it is clear that the digit "1" is considerably easier to learn compared to the others.

Finally, the remarkable outcome highlighted in this section is that there are some clearly defined configurations that work noticeably better, and that our model, once the regularizations are applied, is able to reach them with consistent behaviour.

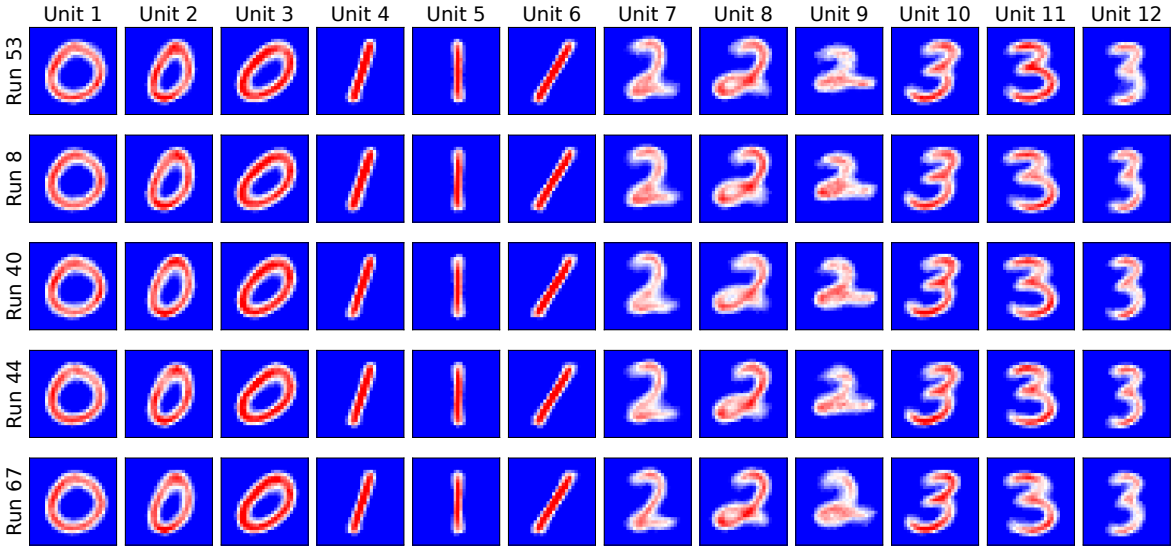

Figure 4.11: *Learned $w_{i\alpha}$ (where unit index α corresponds to the top-row units) associated with the right-hand peak in Fig. 4.10(b) for the models trained with $\lambda_{\vec{a}} = 0.01$ and $\lambda_2 = 1$. The top row displays the best performing model in this area, with $\mathcal{L} = -146.61$. The colorbar ranges from -2 (blue) to $+2$ (red).*

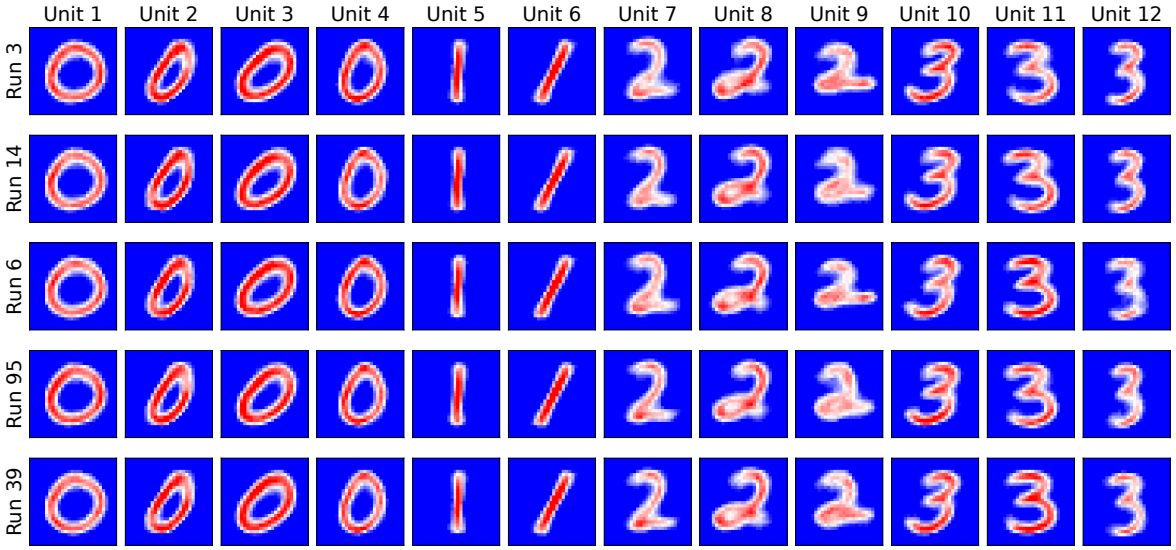

Figure 4.12: *Learned $w_{i\alpha}$ (where unit index α corresponds to the top-row units) associated with the middle peak in Fig. 4.10(b) for the models trained with $\lambda_{\vec{a}} = 0.01$ and $\lambda_2 = 1$. The top row displays the best performing model in this area, with $\mathcal{L} = -147.02$. The colorbar ranges from -2 (blue) to $+2$ (red).*

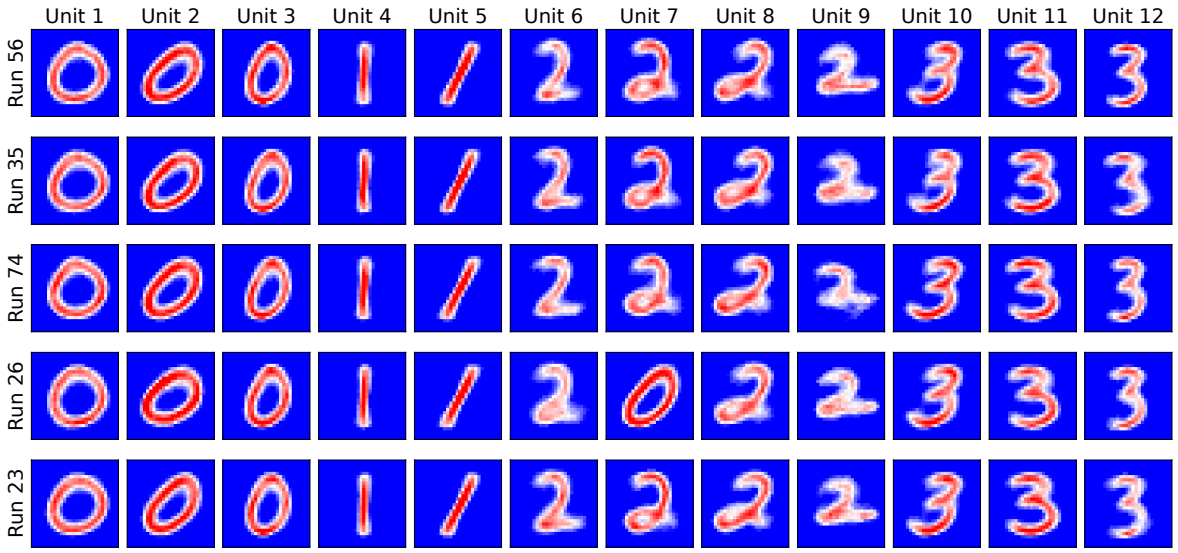

Figure 4.13: *Learned $w_{i\alpha}$ (where unit index α corresponds to the top-row units) associated with the left-hand peak in Fig. 4.10(b) for the models trained with $\lambda_{\bar{a}} = 0.01$ and $\lambda_2 = 1$. The top row displays the best performing model in this area, with $\mathcal{L} = -147.34$. The colorbar ranges from -2 (blue) to $+2$ (red).*

4.4 A Bernoulli insight

We now focus on the Bernoulli case, where the hidden units are independent and can be active at the same time. In this setting, the number of possible states is 2^L. To keep the problem computationally manageable, and to ensure that all the relevant metrics (including σ) can be computed in a meaningful way, we choose $L = 5$, which corresponds to 32 possible states.

In this case we move directly to the comparison between the ssMM and the ssMM with regularizations; note that to apply $\lambda_{\bar{a}}$ we always use the expression in Eq. 3.1, while for λ_2 we must use Eqs. 3.17a and 3.17b presented in Sec. 3.2.1. In this case the result again confirms the hierarchy already observed in the one-hot encoded setting (Fig. 4.14(a) and Tab. 4.4), with the use of both regularizations yielding the best overall performance. In particular, this combination avoids exceptionally unfortunate cases, as indicated by the minimum reached by the worst ssMM baseline model in Tab. 4.4, and it reduces the dispersion of $\mathcal{L}$. It is crucial to note that the improvement is tightly linked to a global upward shift in the number of states visited throughout the Markov chain (Fig. 4.14(b)).

	mean	**sd**	**min**	**max**
normal	-156.8 ± 0.3	3.2	-179.2	-153.8
$\lambda_{\bar{a}} = 0.01$	-155.8 ± 0.1	1.3	-162.2	-153.7
$\lambda_{\bar{a}} = 0.01$, $\lambda_2 = 1$	-155.7 ± 0.1	1.3	-161.5	-153.4

Table 4.4: *Descriptive statistics for the log-likelihood $\mathcal{L}$ of Fig. 4.14(a) (Bernoulli case).*

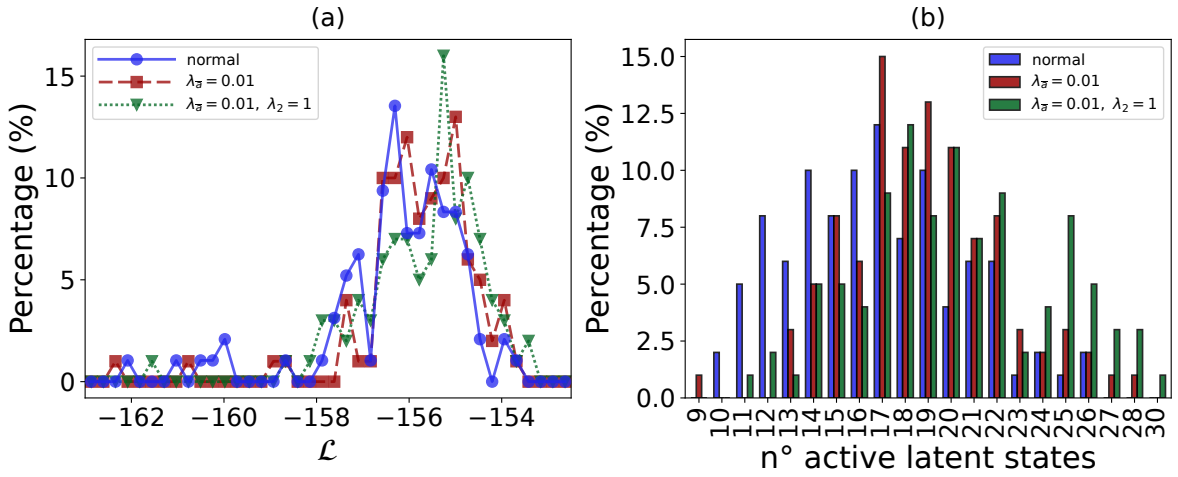

Figure 4.14: *(a) Log-likelihood in the Bernoulli case with the effect of the regularization terms $\lambda_{\bar{a}}$ and λ_2. Note that more hidden units would be required to reach the same performance obtained with the one-hot model. For the normal model, there are three additional runs beyond the minimum value shown, which was selected for aesthetic reasons (see Tab. 4.4). (b) Number of activated 2^L states (states visited at least 0.3% of the time in the trajectories after training). Once again, the applied regularizations lead to an upward shift in the number of visited states, resulting in a more effective exploration of the latent space.*

Finally, in Fig. 4.15 we show the fantasy particles that the Bernoulli model is capable of generating. Note that the non–one-hot case naturally exhibits greater generative capacity, since the matrices $w_{i\alpha}$ are now composed of small digit fragments that, when activated jointly, combine to form the handwritten image.

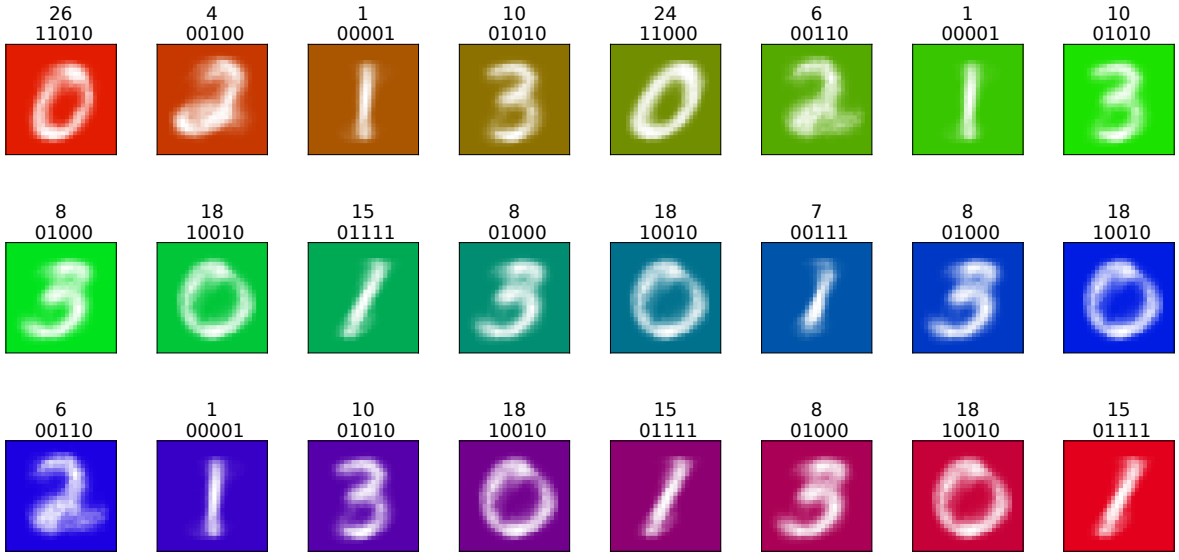

Figure 4.15: *Trajectory of mean fantasy particles generated by the best trained model ($L = 5, \lambda_{\bar{a}} = 0.01, \lambda_2 = 1$ with $\mathcal{L}$ -153.3). For each entry, the state index is shown at the top, immediately followed by its binary representation, which specifies the hidden units active for that particular state $\mathbf{z}$. The corresponding generated image is displayed below. The variety of patterns is larger, and the system exhibits good ergodic behaviour, although some outputs still fail to resemble digits perfectly. Colours are used solely for visual clarity and have no intrinsic meaning.*

Conclusions

Conclusions

This thesis examined the steady-state Markov Machine (ssMM), an architecture closely related to restricted Boltzmann machines. A central part of the study focused on the behavior of hidden units under one-hot encoding, where RBMs typically encounter dead-unit problems. As discussed in Sec. 2.3, a serendipitous discovery showed that treating the L hidden units as independent during the activation step, and renormalizing their probabilities only afterwards, greatly reduces this issue. Although this procedure is formally inconsistent—activating one component necessarily suppresses the others—it nonetheless performs better than the theoretically correct approach. A full theoretical explanation for this discrepancy remains open.

Further improvements were obtained through a simple "spring" regularization applied to the backward bias vector $\overline{a}$. This mechanism prevents the activation probabilities of the latent states from drifting excessively, keeping them confined within a balanced range, therefore ensures that all latent states remain effectively accessible during training.

The analysis also showed that ssMMs naturally operate out-of-equilibrium. The latent dynamics consistently exhibit positive entropy production and non-symmetric transition matrices $M_{\mathbf{z}\mathbf{z}'}$. Encouraging this nonequilibrium behavior leads to more reliable generative performance: models with stronger directional flows in latent space tend to produce more consistent and higher-quality results.

Furthermore, the analysis revealed the presence of a performance threshold. Once the model is provided with fixed resources, the log-likelihood landscape develops well-defined peaks and valleys, which become even more pronounced under regularization, and the training trajectories tend to converge toward characteristic patterns. A notable result was the demonstration that reducing the variance across trained models, and consistently obtaining the best-performing ones, becomes feasible. The methods used to achieve these improvements are lightweight and interpretable, in contrast with the black-box behavior often encountered in neural networks, where identifying effective regularizations is considerably more challenging.

Finally, an attempt was made to extend the one-hot results to the more general Bernoulli case. In this setting, the rapid growth of the latent space size, 2^L, complicates both the computation and the definition of nonequilibrium quantities, especially for large transition

matrices $M_{\mathbf{z}\mathbf{z}'}$. For a limited case with $L = 5$ (32 latent states), the previously introduced regularizations provided partial improvement, although further investigation is required to assess their effectiveness in higher-dimensional settings.

Bibliography

[1] Pankaj Mehta et al. "A High-Bias, Low-Variance Introduction to Machine Learning for Physicists". In: *Physics Reports* 810 (2019), pp. 1–124.

[2] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. "A learning algorithm for Boltzmann machines". In: *Cognitive Science* 9.1 (1985), pp. 147–169.

[3] Paul Smolensky. "Information Processing in Dynamical Systems: Foundations of Harmony Theory". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Ed. by David E. Rumelhart and James L. McClelland. Cambridge, MA: MIT Press, 1986, pp. 194–281.

[4] Geoffrey E. Hinton. "A Practical Guide to Training Restricted Boltzmann Machines". In: *Neural Networks: Tricks of the Trade*. 2nd. Berlin: Springer, 2012, pp. 599–619.

[5] Cédric Roussel, Simona Cocco, and Rémi Monasson. "Barriers and Dynamical Paths in Alternating Gibbs Sampling of Restricted Boltzmann Machines". In: *Physical Review E* 104 (2021), p. 034109.

[6] Elisabeth Agoritsas et al. "Explaining the Effects of Non-Convergent MCMC in the Training of Energy-Based Models". In: *International Conference on Machine Learning*. Honolulu, HI: PMLR, 2023, pp. 322–336.

[7] Peter Dayan et al. "The Helmholtz machine". In: *Neural Computation* 7.5 (1995), pp. 889–904.

[8] Marco Baiesi and Alberto Rosso. "Emergence of nonequilibrium latent cycles in unsupervised generative modeling". In: *Physical Review E* 114.1 (2026), p. 015301.

[9] Manjodh Singh. "Exploiting the conservation of probability in simple machine learning models". MA thesis. University of Padova, Department of Physics and Astronomy "Galileo Galilei", 2025.

[10] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural Computation* 18.7 (2006), pp. 1527–1554.

[11] Barry M. McCoy and Tai Tsun Wu. *The Two-Dimensional Ising Model*. Harvard University Press, 1973.

[12] Geoffrey E. Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural Computation* 14.8 (2002), pp. 1771–1800.

[13] Miguel Á. Carreira-Perpiñán and Geoffrey E. Hinton. "On Contrastive Divergence Learning". In: *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2005, pp. 33–40.

[14] Tijmen Tieleman. "Training restricted Boltzmann machines using approximations to the likelihood gradient". In: *Proceedings of the 25th International Conference on Machine Learning*. 2008, pp. 1064–1071.

[15] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *International Conference on Learning Representations (ICLR)*. 2017.

[16] 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 (ICLR)*. 2017.

[17] Ian J. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014.

[18] Tim Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016, pp. 2234–2242.

[19] Nikhil Kapasi et al. "The Gaussian-Multinoulli Restricted Boltzmann Machine: A Potts Model Extension of the GRBM". In: *Transactions on Machine Learning Research* (2026).

[20] Kevin Swersky et al. "Cardinality Restricted Boltzmann Machines". In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012, pp. 3302–3310.

[21] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[22] Tijmen Tieleman and Geoffrey Hinton. *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*. Tech. rep. 2. COURSERA: Neural Networks for Machine Learning, 2012, pp. 26–31.

[23] Harold W. Kuhn. "The Hungarian Method for the Assignment Problem". In: *Naval Research Logistics Quarterly* 2.1–2 (1955), pp. 83–97.