STRUCTURED IMAGE REPRESENTATION LEARNING FOR FLOW-MATCHING MODELS

Alexandros Graikos[1], **Kostas Triaridis**[1], **Nikolay Malkin**[2], **Dimitris Samaras**[1]
[1]Stony Brook University, [2]University of Edinburgh
{agraikos, kostas, samaras}@cs.stonybrook.edu,
nmalkin@ed.ac.uk

ABSTRACT

We examine representation learning in the context of continuous-time generative models. When tasked with learning to sample from a distribution of images, flow-matching (and denoising diffusion) has been the standard approach, due to the simplicity and stability of training as well as its diverse, high-quality results. However, unlike previous generative model iterations, such as VAEs, vanilla flow-matching models do not learn reusable representations of the data, i.e., latents that can be easily manipulated, combined, or generally utilized in other tasks. In this work, we propose an algorithm to train both an encoder that embeds images into latents and a flow-matching "decoder" model that synthesizes images conditioned on these latents. We find that we can train the encoder with a reinforcement learning objective, utilizing the flow-matching regression loss as a stochastic reward. We modify the RL objective to make the expected reward conditioned on the noise level, allowing the encoder to effectively learn from the intermediate signals obtained by comparing the flow-matching model outputs to a noisy target. Our approach enables unsupervised representation learning of unstructured and **structured latents**, while also retaining the unmatched sample quality of flow-matching models.

1 INTRODUCTION

Generative modeling has made substantial strides in learning to synthesize high-quality, diverse samples with continuous-time generative models, most prominently diffusion (Ho et al., 2020) and flow-matching (Liu et al., 2023). These approaches have focused on training a high-quality generator that transforms noise to samples, and any additional control is provided through human (or automatically-generated) annotations, for instance, text (Nichol et al., 2022; Rombach et al., 2022). This is in stark contrast to previous iterations of deep generative models, which were closely tied to representation learning, and simultaneously learned both how to decompose data into compact, reusable, and often explainable latent representations and how these latents map to samples.

For example, Variational autoencoders (VAEs) (Kingma & Welling, 2013), jointly train an encoder to embed samples to a compact (Gaussian) latent, and while early GANs (Goodfellow et al., 2020) prioritized sample quality, subsequent variants sought to learn interpretable or disentangled latent structures in the data, e.g., style-based factorizations (Karras et al., 2019) or implicit latent structures (Chen et al., 2016).

We attribute this oversight of flow-matching to the fact that jointly learning an image encoder and a generator relies on a reconstruction signal, obtained by evaluating a similarity metric (e.g., L_2 or semantic) between the original and generated samples. For flow-matching, however, decoding is an expensive multi-step process that cannot be easily incorporated into the training, and we thus only have access to a proxy signal that compares the model output to the noisy velocity/noise target.

In this work, we consider representation learning for flow-matching models and propose an algorithm to train both an encoder that embeds images into compact, latent representations and a flow-matching "decoder" that generates high-quality samples from these representations.

Specifically, we employ reinforcement learning to train an encoder using the flow-matching loss as a direct reward, eliminating the need for multi-step sampling. Since the flow-matching model regresses the velocity of a noisy sample, the reward is stochastic and inherently noisy. We thus modify the RL training by conditioning the expectation of the reward at each state (i.e., the Q or $\log F$ function in our formulation) on the noise level, or equivalently, the timestep t at which the flow-matching is evaluated. To train the models, we alternate between training the encoder and a flow-matching decoder that maps tuples of noise and sampled latents into images.

We show that this approach can effectively learn to extract unstructured and structured representations from images, where in the latter case we enforce structure by directly encoding biases in the encoder and decoder models. The proposed algorithm is showcased on CelebA for unstructured representation learning, and on a compositional object dataset and MNIST for structured latents.

2 BACKGROUND

2.1 FLOW-MATCHING GENERATIVE MODELS

Score-based generative models (Song et al., 2021), draw samples from a target distribution by learning the gradient of noise-perturbed versions of the log-probability distribution of the data. This idea, initially established with diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), has more recently been generalized with flow-matching (Lipman et al., 2023) as learning how to dynamically transform random noise samples to data using an ODE parameterized by a learned vector field.

These models have monopolized generative image modeling due to their unmatched sample quality and sample diversity (Nichol & Dhariwal, 2021). A key component in all such models is conditioning, usually applied by cross-attention mechanisms (Rombach et al., 2022) to incorporate additional control signals to the generation steps.

2.2 LEARNING REPRESENTATIONS WITH CONTINUOUS-TIME GENERATIVE MODELS

Although diffusion and flow-matching have dominated the generative space, the efforts in utilizing them for representation learning have been limited. DiffAE (Preechakul et al., 2022) initially proposed jointly training an image encoder with a diffusion model, utilizing the gradients from the denoising loss to learn image representations. This approach is limited in learning unstructed representations, in the form of a continuous vector that are difficult to utilize in downstream tasks. Hudson et al. (2024) expands upon this idea and enforces structure in the latents by enforcing the conditioning to apply to different layers of the denoiser architecture, effectively learning a coarse-to-fine representation. Lyo et al. (2025) focused on learning disentangled representations, but their algorithm requires extra gradient steps to apply the encoder outputs directly onto the denoiser-learned score function.

2.3 REINFORCEMENT LEARNING FOR SAMPLING FROM DISTRIBUTIONS

Reinforcement learning (RL) is used to learn policies that make a sequence of reward-maximizing decisions, naturally fitting problems where an object is to be constructed step by step. In the context of learning to sample from a probability distribution, RL treats drawing a sample as a sequential decision process: starting from an initial state, a policy repeatedly updates the object such that the outcome resembles objects from the target distribution. The reward to be maximized should weigh objects that belong to the empirical distribution higher than those that do not.

Generative Flow Networks (GFNs) (Bengio et al., 2021) build on this idea of RL for sampling, by training a stochastic policy so that the probability of producing a particular object is proportional to the unnormalized target density of that object. A trained GFlowNet can be used as a sampler for the corresponding target distribution without requiring access to the real density–just the unnormalized energy. We refer to Bengio et al. (2023) for a broader overview of the framework. GFlowNet training objectives have been shown to be equivalent to well-known maximum entropy RL (Deleu et al., 2024) algorithms, making the usage of the two terms interchangeable.

GFlowNet-EM (Hu et al., 2023) uses GFNs for discrete latent-variable models with compositional latent spaces. That work highlights how GFNs can be used to learn expressive discrete latents,

without needing tractable approximations (e.g., soft approximations), but is overall limited to using a likelihood-based decoder model, which limits the ability to apply it in modern generative model settings.

3 METHOD

We want to train two models: an encoder $p_\theta(z \mid x)$ that embeds an image x into a latent representation z, and a flow-matching decoder v_ϕ that maps noise $\epsilon \sim \mathcal{N}(0, I)$ and latents z back to images x. We describe the training process for each below.

Decoder For the latent-to-image decoder, we employ a flow-matching model (Lipman et al., 2023), parametrized by a learned conditional velocity field $v_\phi(x_t, t, z)$. Training this model is straightforward, assuming that we already have an encoder from which we can sample latents z for all x, by matching the model's output to the target velocity

$$\mathcal{L}_{\mathrm{FM}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\ x_t\sim p(x_t|x_0),\ z\sim p_\theta(z|x)}\Big[\big\| v_\phi(x_t, t, z) - v(x_t, t) \big\|_2^2\Big]. \tag{1}$$

In our experiments, we use the vanilla flow-matching formulation, where $p(x_t \mid x) = \mathcal{N}((1 - t)x, t^2 I)$ and $v(x_t, t) = \epsilon - x$.

We can select different ways to incorporate the latent z into the flow-matching network, and this choice allows us to introduce inductive biases that control what the learned latent represents, which we discuss in detail in our experiments.

Encoder For the encoder, we need to train a model that draws latents z from images x, such that using z to condition the velocity prediction v_ϕ minimizes $\mathcal{L}_{\mathrm{FM}}$. This translates to training an encoder that samples the latents that best recover the image from added noise. We propose training this encoder with a reinforcement learning objective, where sampling 'useful' latents is equivalent to maximizing the log reward

$$\log R(x, z) = -\mathbb{E}_{t\sim\mathcal{U}[0,1],\ x_t\sim p(x_t|x_0)}\Big[\big\| v_\phi(x_t, t, z) - v(x_t, t) \big\|_2^2\Big]. \tag{2}$$

Under this RL formulation, we define the latent to be a compositional object $z = \{z_1, z_2, \ldots, z_M\}$ and the encoder a model that learns the policy $p_\theta(z_i \mid z_{<i}, x)$, of adding a new object to the latent. Therefore, sampling a latent z requires drawing M samples from the learned policy conditioned on the image x.

To learn this policy, we resort to Generative Flow Networks (GFlowNets) (Bengio et al., 2021; 2023). GFlowNets pose the task of learning to draw samples from a distribution as a sequential decision problem, where the learned policy samples objects proportionally to a reward. Specifically, we use the forward-looking loss (Pan et al., 2023):

$$\begin{aligned}\mathcal{L}_{\mathrm{FL}}(\theta) = \sum_i \Bigg(& \underbrace{\log p_\theta(z_i \mid x, z_{<i})}_{\text{log-prob of } z_i} - \underbrace{\Big(\log F_\theta(z_{<i+1}) - \log F_\theta(z_{<i})\Big)}_{\text{change in log-reward to-go}} \\ & - \underbrace{\Big(\log R(x, z_{<i+1}) - \log R(x, z_{<i})\Big)}_{\text{change in current log-reward}} \Bigg)^2\end{aligned} \tag{3}$$

where the log-reward $\log R(x, z)$ is computed for $z_{<i}$ by conditioning the flow-matching network on incomplete latents, and $\log F_\theta(z_{<i})$ expresses the log of the total remaining reward over all completions from $z_{<i}$, or how "promising" the partial latent $z_{<i}$, aggregated over all ways to finish it.

Training with a flow-matching decoder We make a few modifications to the GFN objective of Eq 3 to enable stable training, which we further discuss and ablate in the Experiments. First, instead of using the velocity prediction of Eq. 2, we use the (partial) reward

$$\log R(x, z) = -\mathbb{E}_{t\sim\mathcal{U}[0,1],\ x_t\sim p(x_t|x_0)}\Big[-\|\hat{x}_0(x_t, t, z_{<k}) - x\|_2^2\Big]. \tag{4}$$

where $\hat{x}_0(x_t, t, z)$ is the final image prediction given by the decoder, obtained by

$$\hat{x}_0(x_t) = x_t - t\, v_\theta(x_t, t, z_i). \tag{5}$$

In our experiments, we find that using the $\hat{x}_0$ for the reward performs better than the velocity prediction. We attribute this to the weighing based on the timestep t. For $t \to 0$, the sample x_t has little noise and the influence of the learned velocity v_θ is diminished, since the prediction does not affect the reconstruction. When $t \to 1$, where x_t is mostly noise, sampling a latent z_i that provides useful information to the denoiser on how to correctly predict the final image is weighted more heavily in the reward.

The second modification we make is to the forward-looking loss. Training also requires learning $\log F(z_{<k})$, which computes the log rewards left for future latents after k. For the stochastic reward of Equation 4, future rewards depend on the timestep t. Thus, we also condition the $\log F$ estimation on the timestep, i.e., $\log F(z, t)$. We find this modification necessary to even learn a non-random policy. Since the reward is partially computed over different incomplete latents $z_{<i}$, we reduce the variance of the forward-looking loss by fixing x_t

$$\mathbb{E}_{t\sim\mathcal{U}[0,1],\ x_t\sim p(x_t|x_0)}\Big[\mathcal{L}_{\mathrm{FL}}(\theta)\Big]. \tag{6}$$

and setting the log-reward to be dependent on the sampled x_t

$$\log R(x_t, t, z) = -\left\|\hat{x}_0(x_t, t, z_{<k}) - x\right\|_2^2 \tag{7}$$

To train the two models, we alternate between training the encoder and decoder. The training schedule, i.e., the number of encoder and flow-matching updates, is a hyperparameter that we fix throughout the training process.

Exploration As in all policy learning problems, we need to introduce exploration to avoid the collapse of the encoder in our case. We introduce additional steps that explore the latent space, by sampling latents from a prior distribution $p(z)$, generating samples x, and maximizing the log-likelihood of these samples

$$\mathcal{L}_{\text{explore}}(\theta) = -\mathbb{E}_{z\sim p(z),\ x\sim p(x|z)}\left[\sum_i \log p_\theta(z_i \mid x, z_{<i})\right] \tag{8}$$

This step is critical in learning an expressive posterior distribution of latents. In our experiments, we use fixed priors $p(z)$, but one could also jointly learn the prior and sample from it. In all our experiments, we sample the image x by running the flow-matching model with a single step of inference. This may produce blurry images, but we find it sufficient to prompt the necessary exploration in the encoder.

4 Experiments

In Section 4.1, we apply our method to learn unstructured representations of CelebA (Liu et al., 2015) images. In Section 4.2 we examine a structured latent space of discrete objects, using the Tetrominoes dataset (Burgess et al., 2019; Emami et al., 2021). Finally, in Appendix A.1, we apply our model to a setting where no apparent structure exists, in contrast to this object-centric dataset. There, we discuss how the encoder mines for the imposed structure in the dataset to come up with a latent representation that best describes the images for the denoising task of the decoder.

4.1 Unstructured representation learning

We first test our method by learning unstructured representations, in the form of a single continuous embedding z on 32×32 CelebA images (Liu et al., 2015). We use this formulation to compare with DiffAE (Preechakul et al., 2022), which also learns unstructured latent representations of images jointly with the generative flow-matching model. We train both image encoders using a simple convolutional network, and the same 32-dim latent parametrized by a Beta distribution to restrict exploration in $[0, 1]$. To predict $\log F$, we add a second prediction head to the encoder, and use a sinusoidal embedding for timestep conditioning (Ho et al., 2020). For the flow-matching model, we

use a standard denoiser UNet (Ho et al., 2020; Rombach et al., 2022), and apply the condition using cross-attention.

In Figure 1, we qualitatively and quantitatively compare the two approaches by providing examples and measuring the reconstruction MSE and FID on the CelebA test dataset. Both models achieve similar reconstruction results, and interestingly, when starting from the same initial noise, the synthesized images for a given reference sometimes portray identical visual features.

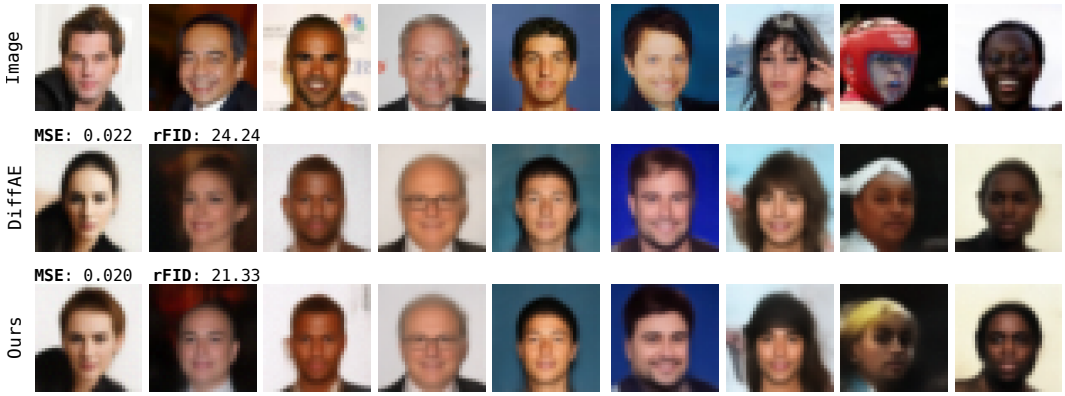

Figure 1: Comparison between DiffAE and the proposed training algorithm on CelebA-32. For both models, the encoder produces a 32-dimensional latent parametrized by a Beta distribution.

With this comparison, we show that DiffAE, which propagates gradients directly from the flow-matching model to the image encoder, ends up learning similar representations to our approach, which does not involve gradient flow from decoder to encoder. In the proposed method, the image encoder is optimized to sample latents that minimize the flow-matching loss using the reward. The gradient-free training is advantageous when using a massive flow-matching decoder for which backpropagation may be time-consuming, and can also be exploited to train encoders that would otherwise require soft gradient approximations, e.g., encoders that embed images into discrete latents. We discuss the discrete encoder case in the following experiment.

4.2 Structured representation learning

We apply our algorithm to the Tetrominoes dataset (Burgess et al., 2019; Emami et al., 2021), where the latent now is defined as a set of three objects $\{(x_1, y_1, z_1),\ (x_2, y_2, z_2),\ (x_3, y_3, z_3)\}$. Each object is described by its location (x, y) and appearance z, with both being discrete **categorical variables**. Location is one of 35 rows and columns, and appearance takes a value out of the 108 possible combinations of shapes, orientations, and colors. An overview of the model is shown in Figure 2 (a).

To implement the encoder, we use a UNet (Ronneberger et al., 2015) that outputs a location map, normalized over rows and columns, and an object map which predicts the presence of objects at every location in the image. We regress $\log F$ using a prediction head on the UNet bottleneck features. To enable the location-dependent conditioning of the denoiser, we encode the sampled position (x, y) with sinusoidal embeddings (Vaswani et al., 2017). We encoded the object appearance with a learned dictionary that maps discrete objects to continuous vectors.

In Figure 2 (b), we present results of the trained encoder and decoder models. The encoder samples latents at the locations of objects in the image, and chooses the appropriate object for each location. The flow-matching model generates images that closely match the reference by utilizing the location and object information provided in the conditioning. This is learned during the alternating training process, where the encoder is encouraged to sample latents that help denoise images, while the decoder learns to utilize the provided latents to denoise images. Since the image generation is stochastic, depending on the initial noise, it is not guaranteed to always be perfect, as observed in the last two columns of Figure 2 (b).

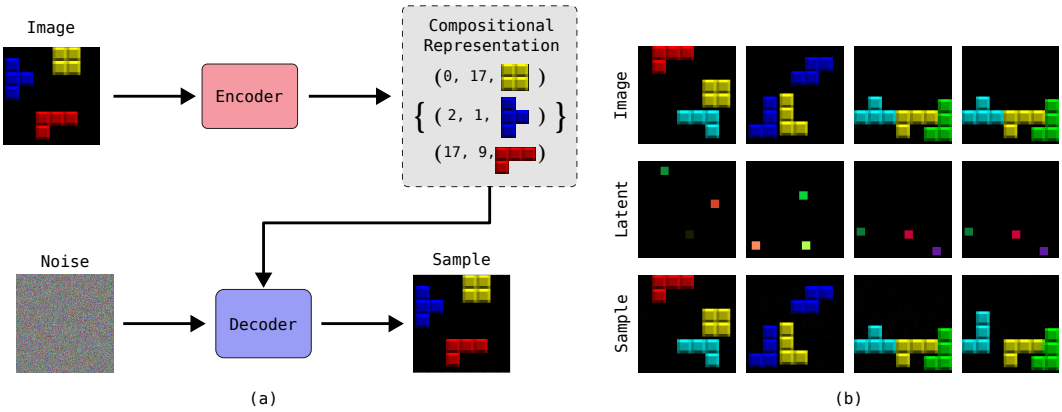

Figure 2: (a) Overview of the proposed method. We train an encoder and a decoder to learn how to decompose images into compositional latents (locations + objects) and how to reconstruct from these compositional representations. (b) The encoder learns to embed tetromino images into distinct objects and locations. We visualize this latent as a map of color-coded object locations. The flow matching decoder maps these latents back to images that closely resemble the original samples. Since this mapping is stochastic, the decoder captures the uncertainty in the encoder's latents, as shown in the last column.

Comparison to slot-attention To obtain a valid baseline, we train a slot-attention-conditioned flow-matching model (Jiang et al., 2023). We use a convolutional encoder network to first downsample the image into an 8×8 grid of features, and then apply the slot-attention mechanism with $K = 4$ slots, one for each object and one for the background. The slot-attention encoder is jointly trained with the flow-matching model, similarly to the DiffAE approach (Preechakul et al., 2022).

The encoder trained with our method can sample latents in multiple ways. We can follow the "maximum probability path'', by choosing at each step $z_i = \text{argmax}_l \; p_\theta(z_i = l \mid z_{<i}, x)$ (argmax). Alternatively, we can sample multiple latents from the learned distribution and only keep the best reconstructions (Top-k). We report both strategies in Table 1, and also measure the Top-K reconstructions for the slot-attention for fair comparison. Here, we note that one could also perform a tree search to find latents that maximize a reward over both latents and the image, which we leave to future work.

In Table 1, we compare the mean squared error of the reconstructions between the proposed approach and the slot-attention-based model. The results show that both methods learn to effectively extract an informative conditioning signal for the flow-matching model, representing the different objects present in a given image. However, we highlight that our RL-trained encoder produces an interpretable (x, y, z) tuple for each object in the image, which we can easily manipulate as we discuss below. This is a significant advantage over slot-attention, which only produces uninterpretable cluster centers from deep features. Within these cluster centers, position is tied to appearance, making image editing difficult.

Model	MSE
Slot-Attention	0.0084 ± 0.00012
Slot-Attention (Top-10)	0.0016 ± 0.00014
Ours (argmax)	0.0083 ± 0.00032
Ours (Top-1)	0.0093 ± 0.00055
Ours (Top-10)	0.0018 ± 0.00006

Table 1: Reconstruction MSE for the Tetrominoes dataset. We repeat the sampling five times to report the standard deviation.

Manipulating latents In Figure 3, we showcase how we can manipulate the latents by hand to control the generated image. For the given image, we extracted the latents using the trained encoder and performed two edits, denoted by the white arrows: moving a single object from left to right, and swapping a single object's appearance. For each edit we make, the decoder correctly interprets the changes in latents to changes in the generated image. This is learned in an unsupervised manner

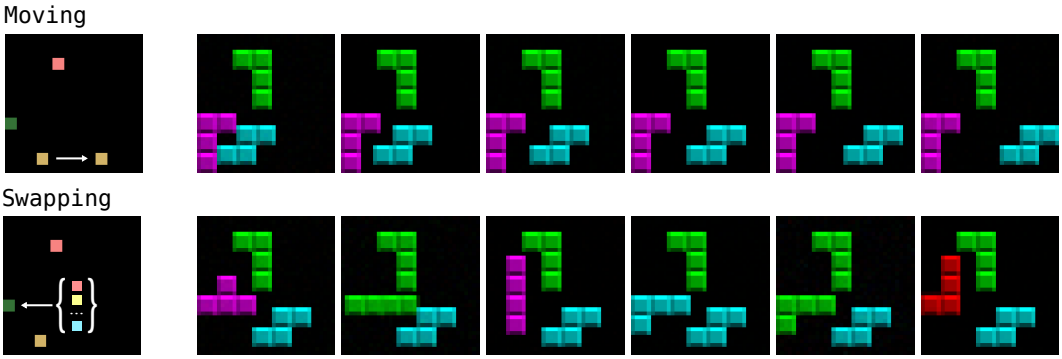

Figure 3: Manipulating latents by hand in the learned latent space. We select a single object latent and modify its location by hand, moving it left-to-right (row 1). We change an object's appearance to random appearances, leading to the object switching between different colors, shapes, and orientations but retaining the location of the original object (row 2).

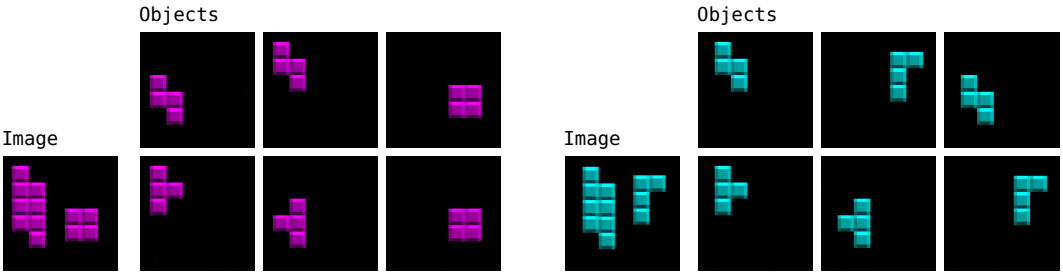

Figure 4: The learned encoder stochastic policy allows us to sample multiple latent 'interpretations' for the same image. This results in different object decompositions in ambiguous cases such as the ones shown in the example above.

by enforcing the desired structure directly on the latent and by introducing the appropriate biases (positional embeddings) in the conditioning mechanism of the flow-matching generator.

Multimodality Another advantage is that the encoder trained with the proposed RL-objective learns a stochastic policy from which we can sample multiple 'explanations' for a single image. In Figure 4, we show how the policy's learned stochasticity yields different latent interpretations of a single image. We provide an image with an ambiguous object decomposition and simply draw multiple latents from the encoder. The encoder comes up with different ways to decompose the image, arising naturally from the RL-based training that encourages the model to explore all possible ways of representing an image in latent space. Here, to visualize a set of latents, we change the appearance of all but one object and generate a new image. In this new image, the unchanged object retains the same position and appearance, allowing us to mask the other objects out by comparing the original image to the newly-generated one.

Ablations To understand the training dynamics of the proposed algorithm, we ablate the effect of the different choices we make for training the encoder. The results are presented in Figure 5, where we compare the forward-looking loss $\mathcal{L}_{FL}$, the exploration loss $\mathcal{L}_{\text{explore}}$ and the flow-matching (decoder) loss $\mathcal{L}_{FM}$.

When training the baseline model (column 1), we observe that the forward-looking loss initially increases and then stabilizes, while the exploration loss consistently decreases. The exploration is performed by drawing latents z from a (uniform) prior, generating images x from z, and then trying to predict z from x using the encoder's learned policy. As the training progresses, latents are easier to predict from the synthesized images as the flow-matching model better incorporates information from z to the generated x.

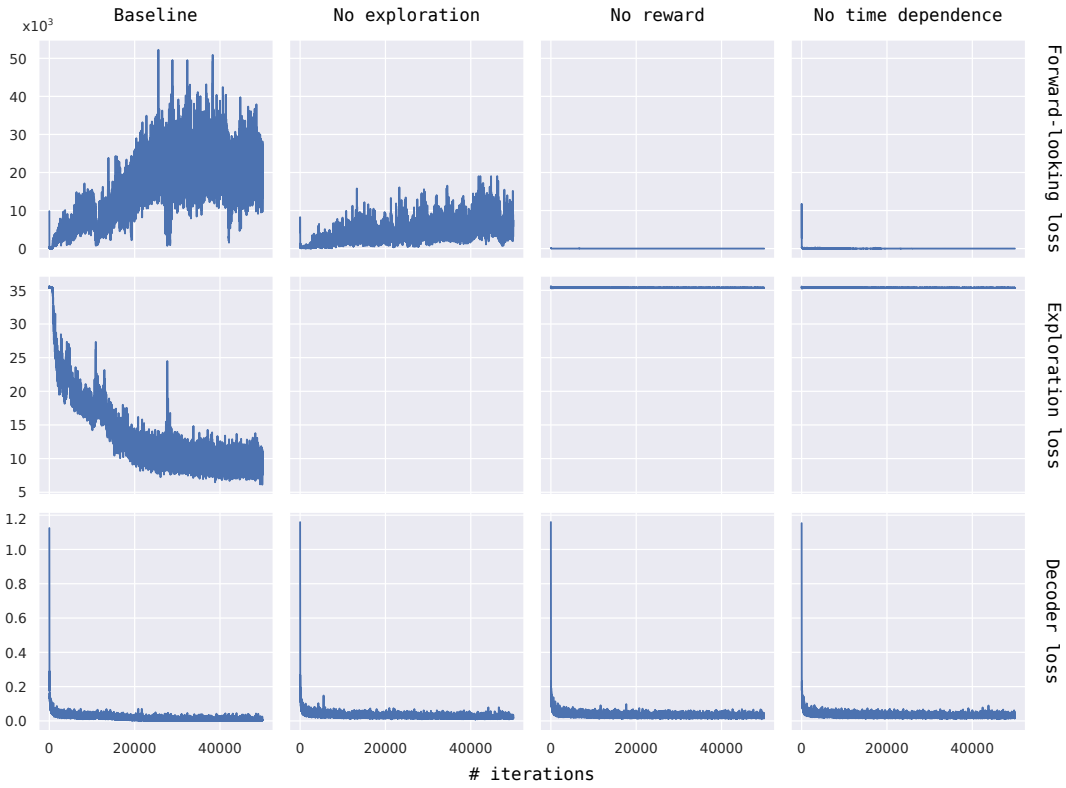

Figure 5: We plot the training dynamics of the proposed algorithm and ablate different design choices. We show how the different losses (forward-looking, exploration, and decoder) behave during training on the Tetrominoes dataset. Convergence is slower without exploration quickly gets stuck at a local minimum (column 1). When we disable the stochastic log-reward or remove the dependency of $\log F$ on the timestep, the training collapses (columns 3-4). The decoder always learns to sample images, even unconditionally, in case the encoder training collapses.

We perform three ablations. When we do not run any exploration steps (column 2), training becomes much slowe,r and the encoder appears to get stuck at a suboptimal policy. When we set the log-reward to zero (column 3), or do not make $\log F$ time-dependent, the encoder fails to learn anything, emphasizing the necessity of the proposed modifications we make to the training. The decoder loss is reduced and stable in all three settings; if the encoder does not learn meaningful representations, then the decoder simply ignores the latent z and is trained as an unconditional model.

5 Conclusion

In this work, we discuss representation learning for flow-matching models. We propose an algorithm to simultaneously train a (structured) image encoder and a flow-matching generative model that maps the learned representations to images. We utilize a reinforcement-learning objective and make the necessary modifications to train the encoder with a stochastic, noisy reward obtained from the flow-matching decoder. We apply the proposed approach to two datasets, one with unstructured latents and a compositional object dataset, showing that the trained models effectively learn to map images into expressive latent representations that can also be explainable and manipulable when enforcing the appropriate structure biases.

Acknowledgments

This research was partially supported by NSF grants IIS-2123920 and IIS-2212046.

REFERENCES

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. In *UAI*, volume 244 of *Proceedings of Machine Learning Research*, pp. 997–1021. PMLR, 2024.

Patrick Emami, Pan He, Sanjay Ranka, and Anand Rangarajan. Efficient iterative amortized inference for learning symmetric and disentangled multi-object representations. In *International conference on machine learning*, pp. 2970–2981. PMLR, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Edward J Hu, Nikolay Malkin, Moksh Jain, Katie E Everett, Alexandros Graikos, and Yoshua Bengio. Gflownet-em for learning compositional latent variable models. In *International Conference on Machine Learning*, pp. 13528–13549. PMLR, 2023.

Drew A Hudson, Daniel Zoran, Mateusz Malinowski, Andrew K Lampinen, Andrew Jaegle, James L McClelland, Loic Matthey, Felix Hill, and Alexander Lerchner. Soda: Bottleneck diffusion models for representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23115–23127, 2024.

Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. *Advances in Neural Information Processing Systems*, 36:8563–8601, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*. OpenReview.net, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Benjamin SH Lyo, Eero P Simoncelli, and Cristina Savin. Disentangled representations via score-based variational autoencoders. *arXiv preprint arXiv:2512.17127*, 2025.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pp. 16784–16804. PMLR, 2022.

Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of gflownets with local credit and incomplete trajectories. In *International Conference on Machine Learning*, pp. 26878–26890. PMLR, 2023.

Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10619–10629, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

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

A Appendix

A.1 Strokes to MNIST digits

We apply our model in an additional setting, where we learn to encode MNIST images (LeCun et al., 2010) as a set of fixed-width strokes. The encoder samples these strokes by choosing a start and end position

$$z = \{(x^1_{\text{start}}, y^1_{\text{start}}, x^1_{\text{end}}, y^1_{\text{end}}), \ldots, (x^K_{\text{start}}, y^K_{\text{start}}, x^K_{\text{end}}, y^K_{\text{end}})\}. \tag{9}$$

The encoder is implemented as a UNet with two outputs, one for sampling the starting point and one for the end point. We transpose those two output maps to avoid early collapse of the model (predicting the same start and end). The decoder is the same UNet as in the other experiments, but the conditioning is provided as an additional image appended to the input of the model. We choose to sample $K = 4$ strokes for our experiments.

In Figure 6 (a), we present results of the trained encoder and decoder models. The different colored strokes correspond to the order in which the strokes are sampled (red →green →blue →yellow). This order is just for visualization; the decoder only looks at the strokes and not the order (grayscale).

The encoder and decoder adapt to the dataset but learning both positive associations between strokes and outputs, where the presence of a stroke leads to 'on' pixels, and negative associations, where strokes mean lack of activated pixels. Since we enforce no other constraint on the models and rely completely on their inductive biases, this is an acceptable solution.

In Figure 6 (b), we probe the generalization capabilities of the models by using out-of-distribution images from the Omniglot dataset (Lake et al., 2015) as inputs. We observe that the encoder adapts to the input, while the decoder is more constrained to synthesizing an image from the MNIST digits.

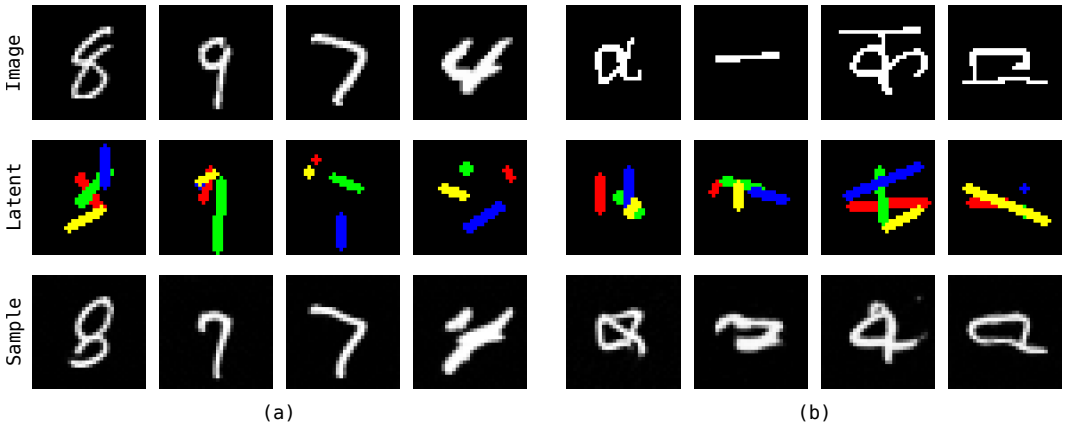

Figure 6: (a) The encoder maps MNIST digits to fixed-width strokes. The decoder translates these stroke images back to MNIST digits. The color represents the order of the sampled strokes. (b) We attempt to encode-decode out-of-distribution samples from the Omniglot dataset. The sampled images resemble edge cases of MNIST digits that best match the given inputs.

A.2 Visualizing sampling steps

In Figure 7, we present the intermediate sampling steps of the flow-matching decoder for each of the datasets that we train on. By visualizing the intermediate steps, we can see how the flow-matching model captures the uncertainty over images for a given set of latents. For instance, in CelebA the exact facial features are decided over all steps of generation, while the latent controls the overall pose and background. For Tetrominoes, some of the learned objects seem to have multiple colors, and initially, the prediction is between cyan and yellow in row 3. Finally, for MNIST, a given set of strokes can correspond to multiple digits or different digit thicknesses, which is not captured by the fixed-width strokes.

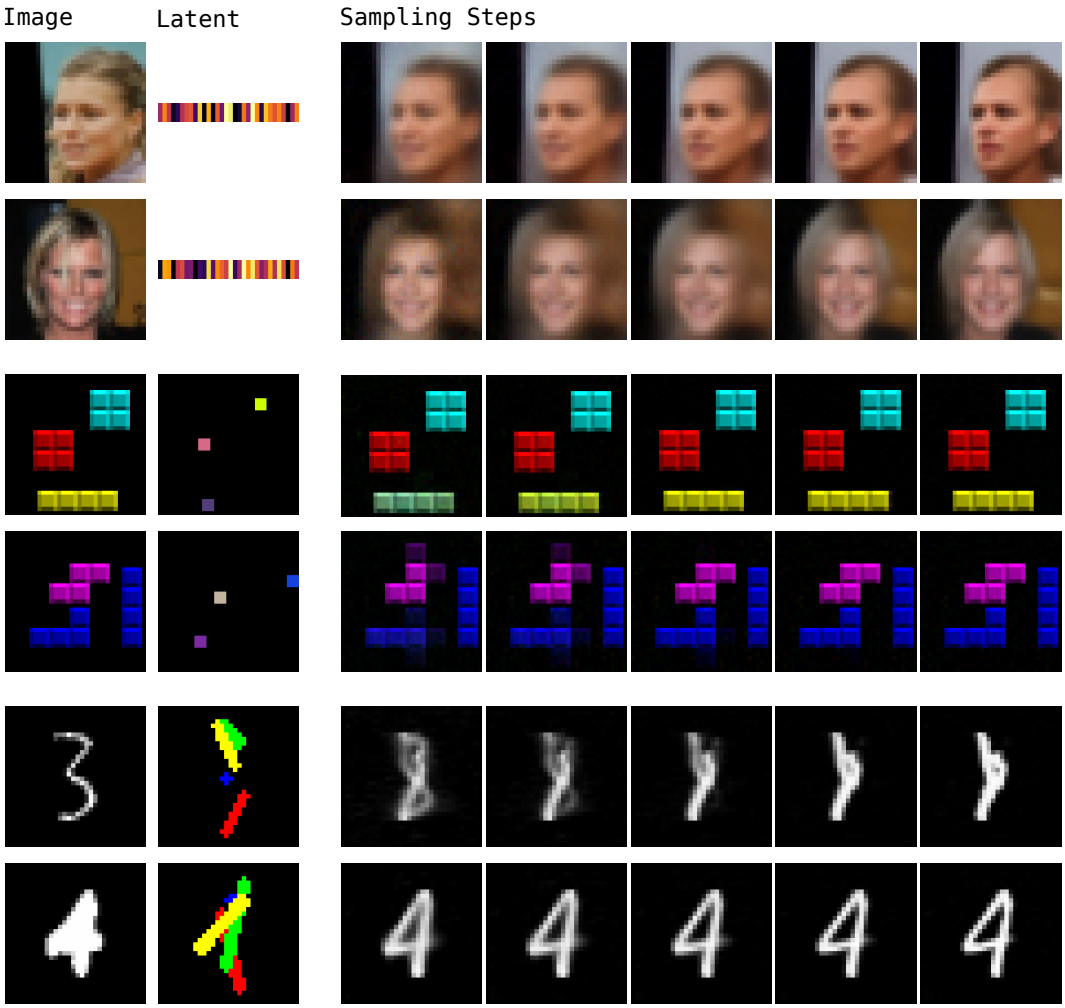

Figure 7: We visualize the sampling steps of the decoder for a given image and latent. We show the final image predictions at every step $\hat{x}_0$ instead of the intermediate noisy image x_t for clarity. These intermediate steps are used to compute the reward with which the encoder is trained.

Algorithm 1 The proposed algorithm.

Input: Dataset $\mathcal{D}$, minibatch size B, number of encoder/decoder training iterations L, total training iterations K, encoder/decoder learning rates $\lambda_{\text{enc}}, \lambda_{\text{dec}}$, exploration loss weight w_{explore}
for $k = 1, 2, ..., K$ **do**
  Sample minibatch $x^j \sim \mathcal{D}$, noise $\epsilon^j \sim \mathcal{N}(0, I)$ timesteps $t^j \sim U[0, 1]$, for $j = 1, 2, \ldots, B$
  Add noise to images $x_t^j = (1 - t^j)x_i + t^j \epsilon^j$
  if $k \mod 2L < L$ **then**
    # Encoder training
    Sample latents from the encoder $z^j = \{z_1^j, z_2^j, \ldots z_M^j\} \sim p_\theta(z \mid x^j)$
    Predict intermediate velocities using decoder $v_\phi^j(x_t^j, t^j, z_{<i}^j)$
    Compute log-rewards $\log R(z_{<i}^j, x^j) = -\|\hat{x}_0^j(x_t^j, t^j, z_{<i}^j) - x^j\|_2^2$
    Compute the forward-looking loss $\mathcal{L}_{FL}$ (3)
    Sample $z^p \sim p(z)$, $x^p \sim p(x \mid z^p)$
    Compute the exploration loss $\mathcal{L}_{\text{explore}}$ (8)
    Update encoder $\theta \leftarrow \theta - \lambda_{\text{enc}} \nabla_\theta (\mathcal{L}_{FL} + w_{\text{explore}} \mathcal{L}_{\text{explore}})$
  else
    # Decoder training
    Sample latents from the encoder $z^j \sim p(z \mid x^j)$
    Predict velocities using decoder $v_\phi^j(x_t^j, t^j, z^j)$
    Compute the flow-matching loss $\mathcal{L}_{FM}$ (1)
    Update decoder $\phi \leftarrow \phi - \lambda_{\text{dec}} \nabla_\phi \mathcal{L}_{FM}$
  end if
end for
Return: Trained encoder p_θ and decoder v_ϕ models.

A.3 Training algorithm

Algorithm 1 describes the training process of the proposed method in pseudocode. We alternate between training the encoder model and decoder model since no gradient flows from decoder to encoder. While training the encoder, the decoder is frozen, providing only rewards to the encoder training objective. Correspondingly, when training the decoder, the encoder acts as a fixed image representation extraction network, that provides the conditioning to the velocity predictions.

We find that including an exploration step with appropriate weighting is necessary to avoid collapse of the encoder. Since the scale of the forward-looking loss varies a lot during training, in all our experiments, we dynamically select w_{explore} such that the two losses (forward-looking and exploration) have similar scales. Analytically,

$$w_{\text{explore}} = \text{pow}(10,\ \log_{10}(\lfloor\, |\mathcal{L}_{FL}| \,\rfloor) - \log_{10}(\lfloor\, |\mathcal{L}_{\text{explore}}| \,\rfloor) \tag{10}$$

where pow(x,y) raises x to the power of y.