NP-DRAW: A Non-Parametric Structured Latent Variable Model for Image Generation

Xiaohui Zeng[1,2]   **Raquel Urtasun**[1,2,4]   **Richard Zemel**[1,2,4]   **Sanja Fidler**[1,2,3]   **Renjie Liao**[1,2]

[1] University of Toronto
[2] Vector Institute
[3] NVIDIA
[4] Canadian Institute for Advanced Research

Abstract

In this paper, we present a non-parametric structured latent variable model for image generation, called NP-DRAW, which sequentially draws on a latent canvas in a part-by-part fashion and then decodes the image from the canvas. Our key contributions are as follows. 1) We propose a non-parametric prior distribution over the appearance of image parts so that the latent variable "what-to-draw" per step becomes a categorical random variable. This improves the expressiveness and greatly eases the learning compared to Gaussians used in the literature. 2) We model the sequential dependency structure of parts via a Transformer, which is more powerful and easier to train compared to RNNs used in the literature. 3) We propose an effective heuristic parsing algorithm to pre-train the prior. Experiments on MNIST, Omniglot, CIFAR-10, and CelebA show that our method significantly outperforms previous structured image models like DRAW and AIR and is competitive to other generic generative models. Moreover, we show that our model's inherent compositionality and interpretability bring significant benefits in the low-data learning regime and latent space editing. Code is available at `https://github.com/ZENGXH/NPDRAW`.

1 INTRODUCTION

Humans understand and create images in a structural way. Given a scene, one can quickly recognize different objects and their functional relationships. While drawing a picture, humans naturally draw one part at a time and iteratively improve the drawing. However, many existing generative models like variational autoencoders (VAEs) [Kingma and Welling, 2013] and generative adversarial networks (GANs)

[Goodfellow et al., 2014] generate all pixels of an image simultaneously, which is unnatural and hardly interpretable. For example, the latent space of VAEs is a high-dimensional vector space which is hard to visualize and meaningfully manipulate. Therefore, building more human-like and interpretable models for image understanding and generation has been an important quest [Lake et al., 2015]. Note that good generative models could often facilitate both image understanding and generation (*e.g.*, a good encoder in VAEs could recognize specific patterns in the image, whereas a good decoder could generate images with high quality).

In this paper, we propose a structured probabilistic model for image generation, which follows the line of Deep Recurrent Attentive Writer (DRAW) [Gregor et al., 2015] and Attend-Infer-Repeat (AIR) networks [Eslami et al., 2016] that mimic how humans draw in a part-by-part fashion. In particular, we sequentially decide whether-to-draw, generate parts (what-to-draw) and their locations (where-to-draw), and draw them on a latent canvas from which the final image is decoded. Our main contributions are as follows.

First, instead of using Gaussian latent variables as in DRAW and AIR which are too limited to capture the large variation of the appearance of the individual image parts, we resort to a non-parametric categorical distribution, which is more expressive, *e.g.*, capturing multi-modalities in the appearance distribution. We build a dictionary of exemplar parts by performing k-medoids clustering on random local patches collected from images. This categorical distribution is then defined over the discrete choice of the exemplar parts. By doing so, we convert the hard what-to-draw problem to an easier which-to-choose problem.

Second, instead of using recurrent neural networks (RNNs) to construct the encoder, prior, and decoder like what DRAW and AIR do, we simplify the overall model by using a Transformer-based prior and plain convolutional neural networks (CNNs) based encoder and decoder. Transformer excels at capturing dependencies among parts and is easier to train compared to RNNs since it does not carry hidden

states over generation steps. This design choice significantly eases the learning and makes our model more scalable.

At last, we propose a heuristic algorithm to parse the latent variables given an image so that we can effectively pre-train our prior network. Moreover, at each generation step, we allow the model to decide whether-to-draw by sampling from a learned Bernoulli distribution. If yes, then we draw the generated part in the generated location of the canvas. Otherwise, we skip this step and leave the canvas unchanged. Therefore, although fixing the total number of generation steps, the effective number of steps varies from image to image, thus being more flexible.

We extensively benchmark our model on four datasets, *i.e.*, MNIST, Omniglot, CIFAR-10, and CelebA. In terms of the sample quality, our model is significantly superior to other structured generative models like DRAW [Gregor et al., 2015] and AIR [Eslami et al., 2016] and is comparable to state-of-the-art generic generative models like WGAN [Arjovsky et al., 2017]. Furthermore, under small portions of training data, our model consistently outperforms other structured generative models and is comparable to the best generic generative model. The strong performances under the low-data regime support that our model better exploits the compositionality in the data, since otherwise it would require much more training samples to reach satisfying performances. We also demonstrate the interpretability of our model by manipulating sampled images in a compositional way through latent space editing.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. Then we introduce the details of our model in Section 3. Section 4 provides experimental results. Section 5 concludes the paper.

2 RELATED WORK

Generative models for images have been studied for several decades [Dayan et al., 1995, Zhu et al., 1997, Freeman et al., 2000, Crandall et al., 2005, Jin and Geman, 2006]. In the context of deep learning, generative models for images have been extensively studied from many aspects or principles, *e.g.*, deep belief nets [Hinton et al., 2006], deep Boltzmann machines [Salakhutdinov and Hinton, 2009], variational auto-encoders (VAEs) [Kingma and Welling, 2013, Rezende et al., 2014, Vahdat and Kautz, 2020, Razavi et al., 2019], generative adversarial networks (GANs) [Goodfellow et al., 2014, Arjovsky et al., 2017, Gulrajani et al., 2017, Brock et al., 2018], generative moment matching networks [Li et al., 2015, Dziugaite et al., 2015], normalizing flows [Rezende and Mohamed, 2015, Dinh et al., 2016, Kingma and Dhariwal, 2018, Papamakarios et al., 2019], energy based models (EBMs) [Du and Mordatch, 2019, Song and Ermon, 2019], and deep diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020]. Our work also relates to exam-

plar VAE [Norouzi et al., 2020] which use a non-parametric prior with exemplars sampled from the training set. However, most of the above generative models generate all pixel values of an image at once, which contrasts with how humans draw pictures.

In this paper, we focus on a class of deep generative models that sequentially generate an image in a part-by-part fashion. One of the first such models is the Deep Recurrent Attention Writer (DRAW) [Gregor et al., 2015]. It is a sequential variational auto-encoding framework with the encoder and the decoder parameterized by two recurrent neural networks (RNNs) respectively. During inference, the encoder uses the spatial attention mechanism to decide where-to-read and then constructs the approximated posterior. During generation, the decoder also leverages spatial attention to decide where-to-write and creates what-to-write. Convolutional DRAW [Gregor et al., 2016] further improves DRAW by leveraging convolutions to construct the encoder/decoder and learning the prior. Attend-Infer-Repeat (AIR) networks [Eslami et al., 2016] enriches this model class by additionally learning to choose the appropriate number of inference/generation steps. AIR was later extended to a sequential version (SeqAIR) [Kosiorek et al., 2018], which models videos and can handle dynamically changing objects. More recently, VQ-DRAW [Nichol, 2020] introduces vector quantization in the latent space and decides what-to-draw by choosing one of the proposals output by a network instead of learning a Gaussian. Arguably, PixelCNN [Van den Oord et al., 2016] can be viewed as an extreme case of this model class that generates one pixel at a time conditioned on previously generated ones without considering a latent space. There are also stroke based generative models like SPIRAL [Ganin et al., 2018], Cose [Aksan et al., 2020], and SketchEmbedNet [Wang et al., 2020]. SPIRAL generates images through a sequence of strokes while Cose and SketchEmbedNet focus on generating sketch images.

3 MODEL

In this section, we introduce our model NP-DRAW, which follows the general framework of VAEs [Kingma and Welling, 2013]. Our generative model $p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, parameterized by θ, first generates some latent variables $\mathbf{z}$ from the prior $p_\theta(\mathbf{z})$ and then generates an image $\mathbf{x}$ conditioned on $\mathbf{z}$ via the decoder $p_\theta(\mathbf{x}|\mathbf{z})$. Given an image $\mathbf{x}$, the inference of latent variables $\mathbf{z}$ is implemented by an amortized encoder network $q_\phi(\mathbf{z}|\mathbf{x})$ parameterized by ϕ. Ideally, $\mathbf{z}$ should describe attributes of the image, *e.g.*, appearances and structural relationships of objects and parts. Many existing latent variable models, *e.g.*, VAEs, use a single vector $\mathbf{z}$, which hardly captures such structural information.

Part-based Image Representations   Inspired by part-based models in image/object modeling [Fischler and
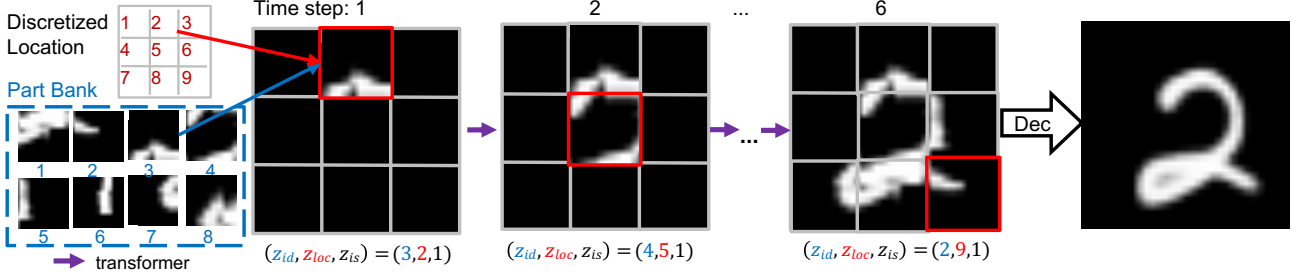
Figure 1: The generative process of our model. At each time step, a Transformer-based auto-regressive prior generates latent variables including what-to-draw $\mathbf{z}_{\text{id}}$ (choosing a part from the part bank), where-to-draw $\mathbf{z}_{\text{loc}}$ (choosing one of the discretized locations), and whether-to-draw $\mathbf{z}_{\text{is}}$ (whether skipping the current step). The sampled image is generated from the final canvas via a decoder.

Elschlager, 1973, Hinton, 1979, Felzenszwalb and Huttenlocher, 2005, Zhu et al., 2007], we introduce multiple groups of latent variables $\mathbf{z} = \{\mathbf{z}^1, \ldots, \mathbf{z}^T\}$ where each group $\mathbf{z}^t$ corresponds to the drawing of a part of the image and T is the total number of generation steps. Here we use parts to refer to local spatial regions of images. They could be semantically meaningful parts of objects like wheels of cars or some recurring regions of the scene. In particular, at the t-th generation step, the group $\mathbf{z}^t$ describes an image part in terms of its location $\mathbf{z}_{\text{loc}}^t$, its appearance $\mathbf{z}_{\text{id}}^t$, and whether we draw it $\mathbf{z}_{\text{is}}^t$ on the latent canvas $\mathbf{c}^t$, *i.e.*, $\mathbf{z}^t = [\mathbf{z}_{\text{loc}}^t, \mathbf{z}_{\text{id}}^t, \mathbf{z}_{\text{is}}^t]$. Although we fixed T as a hyperparameter following DRAW, we allow the model to learn to skip a step (*i.e.*, $\mathbf{z}_{\text{is}}^t = 0$) during the generation. Therefore, the effective number of steps could vary from image to image. This modeling choice is not restrictive if one pre-specifies a reasonably large T. Also, we found that such a sequence of Bernoulli random variables $\{\mathbf{z}_{\text{is}}^t\}$ is easier to learn compared to the geometric distribution used by AIR. For example, AIR tends to learn very small values of T (*e.g.*, $T = 2$ or 3) in practice. We will explain how to construct the probability distributions of these latent variables in Section 3.1.

Overall Generative Process The overview of our generative process is shown in Fig. 1. The sampling from the prior $p_\theta(\mathbf{z}^1, \ldots, \mathbf{z}^T)$ is like drawing one part at a step on a blank canvas for T steps (with the possibility to skip some generation steps), which should produce the rough objects/scenes of the image. The conditional sampling from the decoder $p_\theta(\mathbf{x}|\mathbf{z}^1, \ldots, \mathbf{z}^T)$ then generates the image by completing the fine details on the canvas. In the following, we will explain the design choices of the individual components in detail.

3.1 NON-PARAMETRIC STRUCTURED PRIOR

As discussed above, our prior model $p_\theta(\mathbf{z}^1, \ldots, \mathbf{z}^T)$ aims at sequentially drawing the main objects/scenes of an image one part at a time for T steps in total. Apparently, parts within an image have dependencies. For example, some background parts may have strong correlation in terms of

their appearance, *e.g.*, two parts cropped from an image of sea may highly likely be blue. Some foreground parts may often appear together like wheels and windows of a car, sometimes even under certain relative poses. To capture such dependencies, we employ a auto-regressive model in the latent space,

$$p_\theta(\mathbf{z}^1, \ldots, \mathbf{z}^T) = \prod_{t=1}^{T} p_\theta(\mathbf{z}^t | \mathbf{z}^{<t}) \qquad (1)$$

where $p_\theta(\mathbf{z}^t | \mathbf{z}^{<t})$ represents the distribution of $\mathbf{z}^t$ conditioned on $\{\mathbf{z}^1, \cdots, \mathbf{z}^{t-1}\}$. Recall that $\mathbf{z}^t = [\mathbf{z}_{\text{loc}}^t, \mathbf{z}_{\text{id}}^t, \mathbf{z}_{\text{is}}^t]$. We decompose the conditional distribution in Eq. (1) as below,

$$p_\theta(\mathbf{z}^t | \mathbf{z}^{<t}) = \underbrace{p_\theta(\mathbf{z}_{\text{id}}^t | \mathbf{c}^{<t}, \mathbf{z}_{\text{loc}}^{<t})}_{\text{Appearance}} \times \underbrace{p_\theta(\mathbf{z}_{\text{loc}}^t | \mathbf{c}^{<t}, \mathbf{z}_{\text{loc}}^{<t})}_{\text{Location}}$$
$$\times \underbrace{p_\theta(\mathbf{z}_{\text{is}}^t | \mathbf{c}^{<t}, \mathbf{z}_{\text{loc}}^{<t})}_{\text{Skip}}, \qquad (2)$$

where $\mathbf{c}^{<t}$ is the collection of past canvases. We introduce the canvas in the conditioning to better explain the model. The canvas at time step t is a deterministic function of all latent variables generated so far, *i.e.*, $\mathbf{c}^t = g(\mathbf{z}_{\text{loc}}^t, \mathbf{z}_{\text{id}}^t, \mathbf{z}_{\text{is}}^t, \mathbf{c}^{t-1})$. We will explain the canvas update function g in Section 3.1.3. $\mathbf{c}^0$ is initialized to be empty. The above three probability terms correspond to modeling the appearance of the part, the location of the part, and whether to skip the current generation step respectively.

Transformer Backbone To model the dependencies of all previous canvases and locations which are of variable lengths, we leverage Transformer [Vaswani et al., 2017] as the backbone. We use the same architecture as Vision Transformer (ViT) [Dosovitskiy et al., 2020]. Specifically, at time step t, we feed a length-$(t-1)$ sequence of 2-channel images (each image corresponds to a time step in the past) to Transformer. The first channel of an image contains the latent canvas at that time step whereas the second one is a binary mask which encodes the location of the part (pixels belong to the part are set to 1 otherwise 0) generated at that

time step. The canvas contains the appearance information of all parts drawn so far. The binary masks help the model aware of the locations of the recently generated parts which often provide strong clues about where to draw the next part. We found conditioning on the previously generated locations empirically improves the performance. Our design choice facilitates more efficient learning compared to RNNs used by DRAW and AIR, since Transformer enjoys the parallel training and is easier to optimize. The process is illustrated in Fig. 2. In the following, we explain how to construct the probabilities from the embeddings of Transformer in detail.

3.1.1 What-to-Draw: Non-parametric Appearance Modeling for Image Parts

To model the appearance of individual parts, a straightforward idea is to model $p_\theta(\mathbf{z}_{id}^t|\mathbf{c}^{<t}, \mathbf{z}_{loc}^{<t})$ using a parametric family of distributions, $e.g.$, Gaussians as used in DRAW and AIR. However, such a parametric form has the limited expressiveness and hardly captures the large variation of the appearance in natural images. We instead take a non-parametric approach by first constructing a bank of exemplar parts through clustering and then representing $\mathbf{z}_{id}^t$ as a categorical random variable over the choice of exemplars.

In particular, we first collect raw patches with size $K \times K$ from the training dataset through random sampling. Then we vectorize the patches and perform K-medoids clustering to obtain the bank of exemplar parts which are the M cluster centroids $B = \{\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_M\}$. Therefore, $\mathbf{z}_{id}^t$ takes value from the possible indices of parts, $i.e.$, 1 to M.

To construct the distribution $p_\theta(\mathbf{z}_{id}^t|\mathbf{c}^{<t}, \mathbf{z}_{loc}^{<t})$, we first pool the embeddings of each 2-channel image returned by a projection layer and add to the positional embeddings which encode the information of time steps. We then feed the feature to the transformer encoder which uses the self-attention to improve the embedding. The categorical distribution $\mathbf{z}_{id}^t$ is output by a MLP head (a size-64 hidden layer and a ReLU activation function [Nair and Hinton, 2010]) which takes the embedding as input.

3.1.2 Where-to-Draw: Location Modeling

Previous methods model the distribution of locations of parts via Gaussians which requires truncating the out-of-image ones. For simplicity, we model the locations of parts by discretizing the images into a 2D grid so that a part can only center on a grid. Therefore, $\mathbf{z}_{loc}^t$ is a categorical random variable taking values from the possible indices of the 2D grid. We discretize the image so that each grid is of size $K \times K$ ($i.e.$, same size as a part), thus disallowing parts to have overlap and greatly simplifying the pipeline. In practice, we found learning with such a categorical distribution is easier compared to Gaussians. We use the same embedding returned by the transformer encoder and pre-
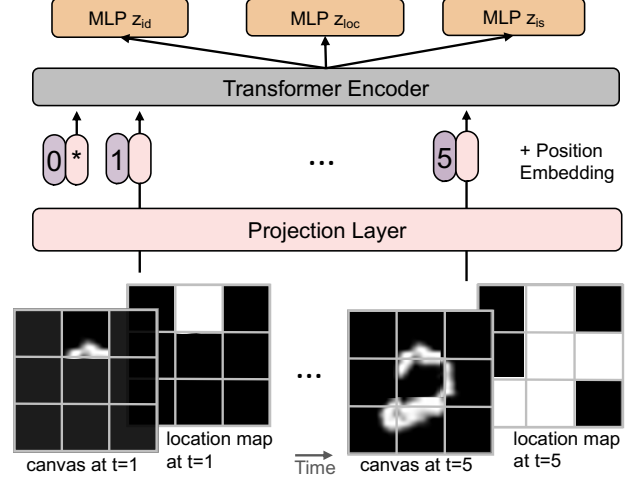

Figure 2: The Transformer-based conditional probability in Eq. (2). The input is a sequence of 2-channel images containing the canvas and the location map.

dict the categorical distribution using a separate MLP head, which has one size-64 hidden layer and a ReLU activation function [Nair and Hinton, 2010].

3.1.3 Whether-to-Draw and Canvas Update

While generating both "what-to-draw" $\mathbf{z}_{id}^t$ and "where-to-draw" $\mathbf{z}_{loc}^t$ at time step t, we allow the model to choose whether to draw it on the canvas by sampling a per-step Bernoulli random variable $\mathbf{z}_{is}^t$. The distribution is again constructed based on a separate MLP head that takes the embedding from the transformer encoder as input. This mechanism allows the model to have varying numbers of generation steps from image to image. We update the canvas as below,

$$\mathbf{c}^t = g(\mathbf{z}_{loc}^t, \mathbf{z}_{id}^t, \mathbf{z}_{is}^t, \mathbf{c}^{t-1})$$
$$= \left(\mathbf{c}^{t-1}\right)^{1-\mathbf{z}_{is}^t} \left(\max\left(\mathbf{c}^{t-1}, \text{Draw}(\tilde{\mathbf{x}}_{\mathbf{z}_{id}^t}, \mathbf{z}_{loc}^t)\right)\right)^{\mathbf{z}_{is}^t}. \tag{3}$$

If $\mathbf{z}_{is}^t = 1$, the "Draw" function generates a mask (same size as the canvas) where the region specified by "where-to-draw" $\mathbf{z}_{loc}^t$ is filled by values from an exemplar part corresponding to "what-to-draw" $\mathbf{z}_{id}^t$. The values outside this designated region are zero. Then we obtain the new canvas via a element-wise maximum between the previous canvas and the generated mask. If $\mathbf{z}_{is}^t = 0$, then we keep the canvas unchanged. The whole update process is deterministic.

3.1.4 Heuristic Parsing for Pre-Training Prior

Learning with a structured prior in VAEs is not easy as demonstrated by vector quantized VAEs (VQ-VAEs) [Oord et al., 2017]. To improve the learning, we also adopt the pre-training strategy as what VQ-VAEs do. Before we explain

---

Algorithm 1 Heuristic Parsing of Latent Variables

1: **Input**: image $\mathbf{x}$, patch bank $B = \{\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_M\}$ containing $K \times K$ exemplar patches, threshold $\epsilon = 0.01$
2: Discretize image (padding if needed) into grids so that each grid is $K \times K$ (assuming there are T such grids)
3: **for** $t = 1, 2, \ldots, T$ **do**   ▷ Parsing latent variables at t-th step
4:     $\mathbf{z}_{\text{loc}}^t = \text{t}$
5:     loc^t = Center location of the t-th grid
6:     $\mathbf{x}_c = \text{CropImg}(\mathbf{x}, \text{loc}^t)$ ▷ Crop a $K \times K$ patch centered at loc^t
7:     $\mathbf{z}_{\text{id}}^t = \text{NearestNeighbor}(B, \mathbf{x}_c)$ ▷ Find the index of the nearest patch to $\mathbf{x}_c$ from B
8:     $\text{Cost} = \|\mathbf{x}_c - \tilde{\mathbf{x}}_{\mathbf{z}_{\text{id}}^t}\| - \|\mathbf{x}_c - \mathbf{0}\|$ ▷ Cost of pasting this patch vs. an empty patch
9:     $\mathbf{z}_{\text{is}}^t = 1$ **if** $\text{Cost} \leq \epsilon$ **else** 0
10: **end for**
11: **Output**: $\mathbf{z} = \left((\mathbf{z}_{\text{loc}}^1, \mathbf{z}_{\text{id}}^1, \mathbf{z}_{\text{is}}^1), \ldots, (\mathbf{z}_{\text{loc}}^T, \mathbf{z}_{\text{id}}^T, \mathbf{z}_{\text{is}}^T)\right)$

---

details of the heuristic parsing, let us denote the general probabilistic parsing of latent variables $\mathbf{z}$ given an image $\mathbf{x}$ as $p_h(\mathbf{z}|\mathbf{x})$. To pre-train the prior, we ideally need to,

$$\min_\theta \quad D_{\text{KL}}(\int p_h(\mathbf{z}|\mathbf{x})p_{\text{data}}(\mathbf{x})\mathrm{d}\mathbf{x}\|p_\theta(\mathbf{z})). \quad (4)$$

However, this is impossible since we do not know the data distribution $p_{\text{data}}(\mathbf{x})$. In practice, we observe samples from $p_{\text{data}}(\mathbf{x})$ and then parse $\mathbf{x}$ to get samples of latent variables $\mathbf{z}$. Therefore, the sampled version of the objective becomes,

$$\max_\theta \sum_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} \sum_{\mathbf{z} \sim p_h(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{z}). \quad (5)$$

Note that if the parsing is deterministic, then the objective becomes the maximum log likelihood of the observed pairs $(\mathbf{x}, \mathbf{z})$. In our case, we propose an effective heuristic parsing algorithm to deterministically infer the latent variables $\mathbf{z}$ given an image $\mathbf{x}$ as shown in Alg. 1. In other words, $p_h(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} = \text{Parsing}(\mathbf{x}))$ is a delta distribution. The idea is as follows. We first discretize the observed image into grids. Then for each grid, we search the nearest patch in the part bank and check whether it is closer to the observed patch than an empty patch. If so, we use its index as the parsed value for the latent variable of the appearance. Otherwise, we choose to skip this grid. Using Alg. 1, we can collect $(\mathbf{x}, \mathbf{z})$ pairs and pre-train the prior using Eq. (5).

3.2 ENCODER & DECODER

Similar to VAEs, we approximate the posterior by learning an encoder parameterized by ϕ. For simplicity, we assume the following conditionally independent encoder,

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^T q_\phi(\mathbf{z}_{\text{id}}^t|\mathbf{x})q_\phi(\mathbf{z}_{\text{loc}}^t|\mathbf{x})q_\phi(\mathbf{z}_{\text{is}}^t|\mathbf{x}). \quad (6)$$

We implement the encoder as a CNN which takes the image $\mathbf{x}$ as input. Distributions of latent variables are constructed using separate MLP heads which take the flatten feature map of the CNN as input. Our encoder CNN consists of 5 convolutional layers with hidden size 128, followed by Batch Normalization [Ioffe and Szegedy, 2015] and a ReLU activation function [Nair and Hinton, 2010]. The first 3 convolutional layers use kernel size 3, stride size 2, and padding size 1. The last two convolutional layers use stride 1. Note that one could further improve the encoder by using a Transformer-based auto-regressive posterior which captures the dependencies across time steps. We leave it for future exploration.

To decode an image, suppose we finish the sampling from the prior and get a canvas $\mathbf{c}^T$ at the last time step T. The canvas should already roughly form the objects/scenes of the image. Then we feed the canvas to a CNN decoder to generate the final image with refined details. Our decoder CNN consists of 2 convolutional layers with stride 2, and two residual blocks, followed by two transposed convolutional layers. All layers have hidden size 128 followed by Batch Normalization and ReLU activations. In particular, we model the output image as,

$$p_\theta(\mathbf{x}|\mathbf{z}) = \text{OutputDistribution}(\mathbf{x}; \text{CNN}(\mathbf{c}^T)), \quad (7)$$

where we choose the output distribution for pixel intensities as Bernoulli for MNIST and Omniglot datasets and as the discretized logistic mixture for CIFAR10 and CelebA following [Salimans et al., 2017].

3.3 LOSS FUNCTIONS

The standard objective for training a VAE model is to maximize the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) \quad (8)$$

However, we found that if we train the model with ELBO alone, the behavior of the learned posterior deviates from our expectation, *i.e.*, the canvas should roughly form the objects/scenes of the image. Therefore, we add an extra regularization term defined as

$$\mathcal{L}_{\text{reg}} = -D_{\text{KL}}(p_h(\mathbf{z}|\mathbf{x})\|q(\mathbf{z}|\mathbf{x})). \quad (9)$$

This term could guide the posterior to behave like our heuristic parser. Therefore, we train our model by maximizing the following full objective

$$\mathcal{L} = \mathcal{L}_{\text{ELBO}} + \lambda \mathcal{L}_{\text{reg}}, \qquad (10)$$

where λ is the weight of the regularization term. We set λ as 50 for experiments on MNIST and Omniglot, and 500 for experiments on CIFAR-10 and CelebA. The value of λ is chosen so that the scaled regularization loss is in the same order of magnitude with ELBO. During training, we fix the prior for simplicity. Exploring learning the prior together with the encoder and the decoder is left to future work.

4 EXPERIMENTS

We test our model on four datasets, *i.e.*, MNIST [LeCun et al., 1998], Omniglot [Lake et al., 2015], CIFAR-10 [Krizhevsky et al., 2009], and CelebA [Liu et al., 2015], from three different perspectives, *i.e.*, image generation under full training data, image generation under low-data (*i.e.*, less data than the original training set), and the latent space editing. More details on datasets can be found in Appendix.

Baselines We compare our method with two classes of models, *i.e.*, 1) structured image models including DRAW [Gregor et al., 2015], AIR [Eslami et al., 2016], PixelCNN++ [Salimans et al., 2017], and VQ-DRAW [Nichol, 2020]; 2) generic generative models including VAE, 2sVAE [Dai and Wipf, 2019], NVAE [Vahdat and Kautz, 2020], WGAN [Arjovsky et al., 2017], snGAN [Miyato et al., 2018] and WGAN-GP [Gulrajani et al., 2017]. For all baselines, we use the publicly released code. We control the number of parameters of all methods roughly the same. More details on hyperparameters can be found in Appendix.

Evaluation Metric For all experiments, we compute the FID score [Heusel et al., 2017] and the negative log likelihood (NLL) (if applicable). We draw 10K samples from each model and compute the FID score w.r.t. 10K images in the test set. For evaluating NLL of VAEs, we use 50 importance weighted samples to get a tighter bound as in [Burda et al., 2016]. We train all models for 500 epochs and test the checkpoint with the best validation FID score.

4.1 IMAGE GENERATION

We first compare our model with all competitors on image generation under the full training data. As shown in Table 1, our model significantly outperforms other structured image models in terms of the FID score. We also report NLLs of all models (if applicable) in Appendix. Our model is comparable to the state-of-the-art generic generative models like NVAE and WGAN-GP. Moreover, our model is easier to train compared to other structured image models. For example, while training AIR, we found it requires careful tuning

Method	MNIST 28×28	Omniglot 28×28	CIFAR-10 32×32	CelebA 64×64
VAE	16.13	31.97	106.7	70
2sVAE*	12.6	-	72.9	44.4
NVAE	**7.93**	**6.84**	55.97	**14.74**
snGAN	-	-	**14.2**	-
WGAN	10.28	13.99	54.82	40.29
WGAN GP*	-	-	42.18	30.3
PixelCNN++	11.38	9.18	68.00	72.46
AIR	482.69	373.93	673.93	399.41
DRAW	27.07	47.22	162.00	157.00
VQ-DRAW	19.64	56.03	80.19	41.87
Ours	**2.48**	**5.53**	62.72	**41.10**

Table 1: Comparison of sample qualities (lower FID score is better). * entries are from 2sVAE paper. - means unavailable.

K	M	λ					
5	50	50	**17.71**	**5.53**	129.73	**89.95**	54.80
6	50	50	16.38	12.86	**118.37**	108.81	25.62
8	50	50	15.60	22.00	132.98	130.39	**21.34**
5	50	50	17.71	**5.53**	129.73	89.95	54.80
5	10	50	15.77	15.02	127.22	115.53	**27.37**
5	200	50	**19.50**	6.08	**115.53**	89.94	38.41
5	50	50	17.71	**5.53**	129.73	**89.95**	54.80
5	50	0	17.71	35.27	110.64	112.07	**25.67**
5	50	1	17.71	5.58	**107.72**	92.27	34.40
5	50	100	17.71	5.74	134.54	92.23	57.96

Table 2: Ablation study of patch size K (top), part bank size M (middle), and regularization λ (bottom) on Omniglot.

to avoid a diverging loss. Note that AIR does not perform well on these datasets, which is likely due to their independent priors (*e.g.*, latent locations drawn from independent Gaussians) over different time steps. Therefore, parts drawn at different steps tend to be randomly placed on the canvas, leading to worse FID scores. We provide more visualization of our generated canvases and images in Fig. 5. It is clear that our canvases do capture the rough objects/scenes in the images. We show more visual comparisons with other models in Fig. 6 and the Appendix.

4.2 IMAGE GENERATION WITH LOW-DATA

In this part, we investigate our model and other baselines under low-data regimes. The reasoning is similar to [Rezende et al., 2016, Lake et al., 2015], *i.e.*, if our model could generate high-fidelity images with less training data, it is likely that our model better exploits the compositionality within the data. Under each low-data setting, our model constructs the part bank only using the given limited training data, which still generalizes well to unseen images empirically. We group the experiments based on how we split the original dataset to create the low-data regime, *i.e.*, split by digits in MNIST, split by characters in Omniglot, split by alphabets in Omniglot, and split by samples in Omniglot and CelebA. We design such splits since only MNIST has the information of digits and only Omniglot has the information of alphabets
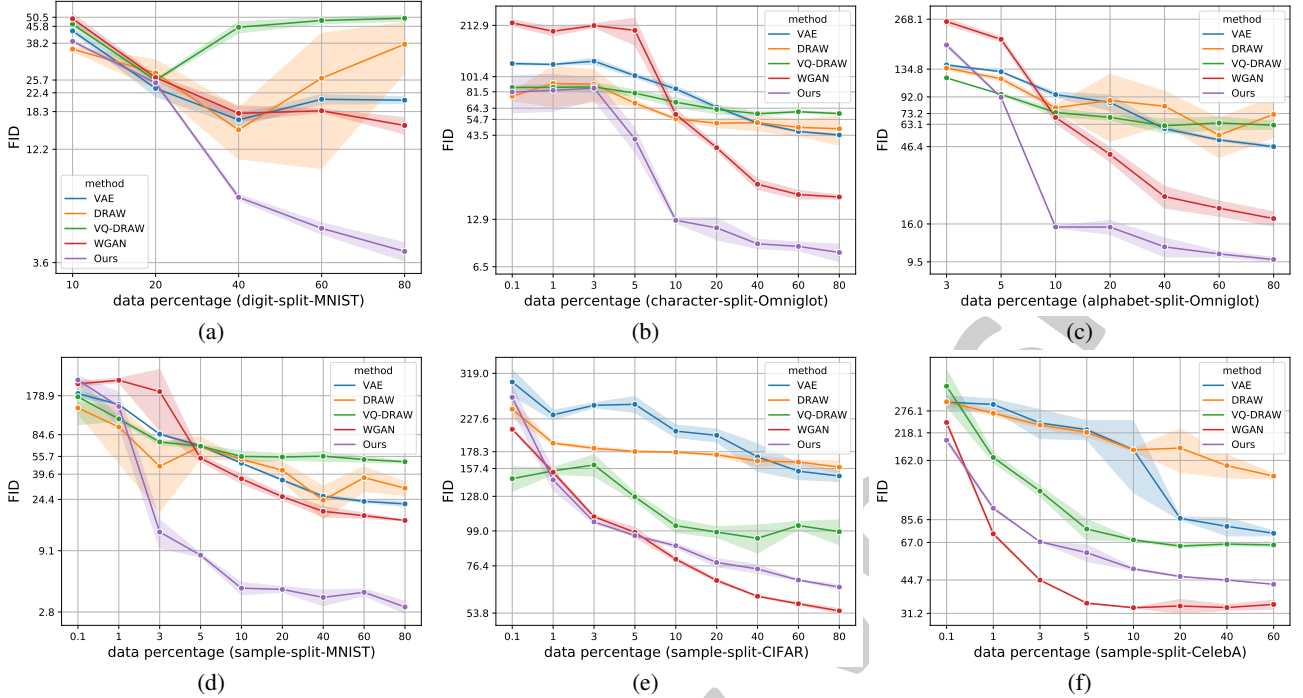
Figure 3: FID score vs. percentage of training data for low-data regimes. Different splits are indicated in the subcaptions. Our method consistently outperforms other structured image models. All experiments are run with three random seeds.

and characters. All results are shown in Fig 3.

Split by Digits We split the MNIST dataset based on digits, *i.e.*, class labels. In particular, there are 10 digits in total each of which occupies 10% of the training data. We train all models with different proportions of training data, *e.g.*, $10\%, 20\%, \cdots, 80\%$ which contain digits $\{0\}, \{0, 1\}, \cdots, \{0, 1, 2, \cdots, 7\}$ respectively. We use the original hold-out test set to evaluate different methods which contains all 10 digits. As shown in Fig. 3a, we can see that our model performs similarly with other models in the extreme cases (*i.e.*, 10% and 20%) and outperforms others with other percentages. Note that the curves of DRAW and VQ-DRAW increase a bit after certain proportions which are likely due to their instable training.

Split by Characters In this setting, we split Omniglot (containing 2089 characters in total and 11 images per character on average) based on characters. We randomly sample certain proportions (shown in the X-axis of Fig. 3b) of characters to form the new training set and validate on the rest. We report the final test performance on the original hold-out test set. Since the test set contains different characters, the task is more difficult than a plain split by samples. From Fig. 3b, we can see that our method consistently outperforms others under all proportions of training data.

Split by Alphabets In this setting, we split Omniglot (containing 50 alphabets in total) based on alphabets. We randomly sample certain proportions (shown in the X-axis of Fig. 3c) of alphabets to form the new training set and val-

idate on the rest. Performances are reported on the hold-out test set. Since the test set contains different alphabets and different alphabets in Omniglot have drastically different appearances, the task is extremely challenging than other splits. Our method achieves the best FID scores on all proportions except 3% as shown in Fig. 3c. Under the 3% case, we only use 1 alphabet for training and constructing the part bank. The slight inferior performance of our model is likely caused by that the part bank may not be expressive enough.

Split by Samples In this setting, we randomly sample certain proportions (shown in the X-axis of bottom subfigures in Fig. 3) of samples to form the new training data set. We test this setting on MNIST, CIFAR-10, and CelebA. It is clear that our method outperforms others under most of the proportions on MNIST. On CIFAR-10 and CelebA, ours is superior to others except for WGAN under almost all proportions. The gap between ours and WGAN may be attributed to the different learning principles, *i.e.*, maximum likelihood vs. adversarial training.

4.3 INTERPRETABILITY

In this section, we demonstrate the advantage of our interpretable latent space via interactively editing/composing the latent canvas. In particular, given two images A and B, we first obtain their latent canvases via the encoder. Then we manually compose a new canvas by placing a portion of the canvas B on top of the canvas A. At last, we generate a new image by feeding this composed canvas to the decoder. Since each part in the canvas corresponds to a local region
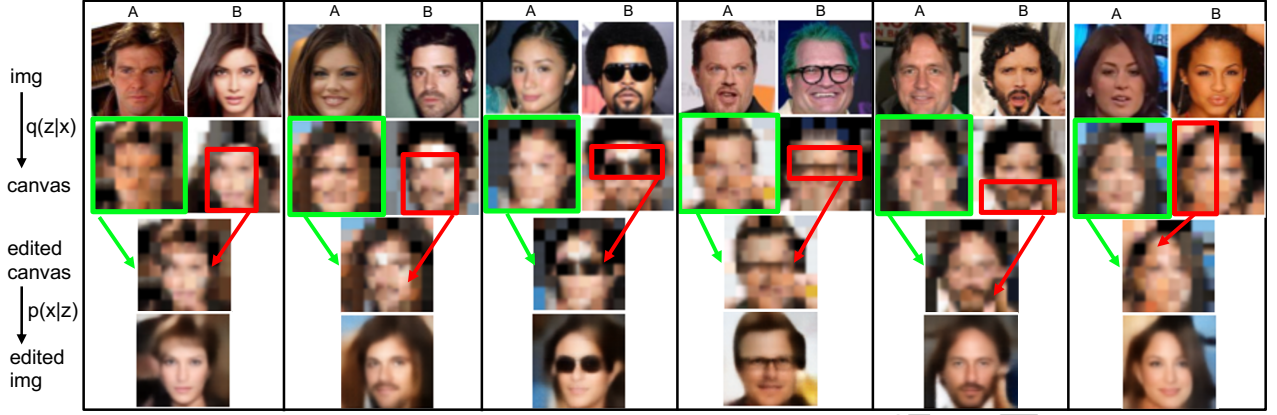
Figure 4: Latent space editing. Given images A and B, we encode them to obtain the latent canvases. Then we compose a new canvas by placing a portion of canvas B on top of canvas A. Finally, we decode an image using the composed canvas.

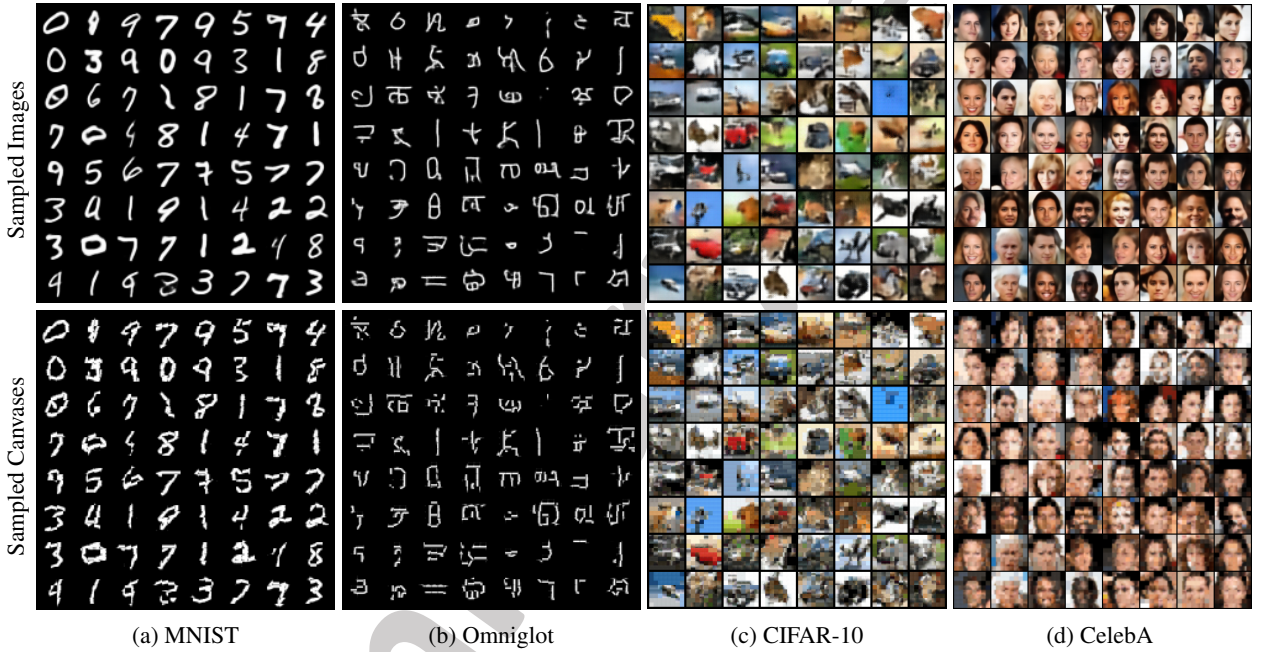| (a) MNIST | (b) Omniglot | (c) CIFAR-10 | (d) CelebA |

Figure 5: Visualization of sampled canvases ($\mathbf{z} \sim p(\mathbf{z})$) and images ($\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$) generated by our model. Same positions within the top and bottom rows correspond to one $(\mathbf{x}, \mathbf{z})$ pair.

in the sampled image, this replacement would only bring local changes, thus ensuring the spatial compositionality. Note that this can not be achieved by common VAEs since even a slight modification to one dimension of the latent space would typically cause global changes w.r.t. appearance, color, *etc*. As shown in Fig 4, we can synthesize new images by placing certain semantically meaningful parts (*e.g.*, eyeglasses, hair, beard, face) from B to A.

4.4 ABLATION STUDY

We show the ablation study of important hyperparameters on Omniglot, *i.e.*, the patch size K, the size of the part bank M, and the regularization weight λ. We use the peak signal-to-noise ratio (PSNR) to measure the quality of our heuristic parsing algorithm (see more details in Appendix). We also

report the FID score, NLL, binary cross-entropy, and the KL divergence. Results are listed in Table 2. From the table, we find that the quality of our heuristic parsing improves as the part bank size M increases and the patch size K decreases. There are some inherent trade-offs on choosing the values of M and K. Intuitively, a large M would make the learning harder since we need to deal with a higher dimensional categorical distribution, whereas a small M would have inferior performance due to the lack of expressiveness. A large K is hard to learn since it requires a large M to ensure the expressiveness, while a small K requires more generation steps. We hence identify a range of K and M based on these trade-offs and then choose the values with the best FID scores. Moreover, adding regularization weight $\lambda > 0$ improves the performance compared to disabling it ($\lambda = 0$). $\lambda = 50$ empirically gives the best result.

Figure 6: Visualization of generated images from different models on different dataset.

5 CONCLUSION

In this paper, we propose a structured latent variable model which generates images in a part-by-part fashion. We construct a non-parametric distribution over the latent variables that describe the appearance of image parts. We then propose a Transformer based auto-regressive prior to capture sequential dependencies among parts. At last, we develop a heuristic parsing algorithm to effectively pre-train the prior. Experiments on several challenging datasets show that our model significantly outperforms previous structured image models and is on par with state-of-the-art generic generative models. Future work include exploring more expressive encoder, multi-scale generation, and other structured priors beyond auto-regressive models.

References

Emre Aksan, Thomas Deselaers, Andrea Tagliasacchi, and Otmar Hilliges. Cose: Compositional stroke embeddings. *arXiv preprint arXiv:2006.09930*, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, 2005.

Bin Dai and David P. Wipf. Diagnosing and enhancing VAE models. In *ICLR*, 2019.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural Computation*, 7(5):889–904, 1995.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

William T Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000.

Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *International Conference on Machine Learning*, pages 1666–1675. PMLR, 2018.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *NIPS*, 2016.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Geoffrey Hinton. Some demonstrations of the effects of structural descriptions in mental imagery. *Cognitive Science*, 3(3):231–250, 1979.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Ya Jin and Stuart Geman. Context and hierarchy in a probabilistic image model. In *CVPR*, 2006.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

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

Adam R Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential attend, infer, repeat: Generative modelling of moving objects. *arXiv preprint arXiv:1806.01794*, 2018.

Alex Krizhevsky, V Nair, and Geoff Hinton. CIFAR10. Technical report, University of Toronto, 2009.

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, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *ICML*, 2015.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Alex Nichol. Vq-draw: A sequential discrete vae. *arXiv preprint arXiv:2003.01599*, 2020.

Sajad Norouzi, David J Fleet, and Mohammad Norouzi. Exemplar vae: Linking generative models, nearest neighbor retrieval, and data augmentation. *Advances in Neural Information Processing Systems*, 33, 2020.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

Danilo Rezende, Ivo Danihelka, Karol Gregor, Daan Wierstra, et al. One-shot generalization in deep generative models. In *ICML*, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *AISTATS*, 2009.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Alexander Wang, Mengye Ren, and Richard Zemel. Sketchembednet: Learning novel concepts by imitating drawings. *arXiv preprint arXiv:2009.04806*, 2020.

Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.

Song-Chun Zhu, David Mumford, et al. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362, 2007.