

PixelTransformer: Sample Conditioned Signal Generation

Shubham Tulsiani¹ Abhinav Gupta^{1,2}

<https://shubhtuls.github.io/PixelTransformer/>

Abstract

We propose a generative model that can infer a distribution for the underlying spatial signal conditioned on sparse samples *e.g.* plausible images given a few observed pixels. In contrast to sequential autoregressive generative models, our model allows conditioning on arbitrary samples and can answer distributional queries for any location. We empirically validate our approach across three image datasets and show that we learn to generate diverse and meaningful samples, with the distribution variance reducing given more observed pixels. We also show that our approach is applicable beyond images and can allow generating other types of spatial outputs *e.g.* polynomials, 3D shapes, and videos.

1. Introduction

Imagine an artist with an empty canvas. She starts with a dab of sky blue paint at the top, and a splash of fresh green at the bottom. What is the painting going to depict? Perhaps an idyllic meadow, or trees in garden under a clear sky? But probably not a living room. It is quite remarkable that given only such sparse information about arbitrary locations, we can make guesses about the image in the artist’s mind.

The field of generative modeling of images, with the goal of learning the distribution of possible images, focuses on developing similar capabilities in machines. Most recent approaches can be classified as belonging to one of the two modeling frameworks. First, and more commonly used, is the latent variable modeling framework (Kingma & Welling, 2013; Goodfellow et al., 2014). Here, the goal is to represent the possible images using a distribution over a bottleneck latent variable, samples from which can be decoded to obtain images. However, computing the exact probabilities for images is often intractable and it is not straightforward to condition inference on sparse observations *e.g.* pixel values. As an alternative, a second class of autoregres-

sive approaches directly model the joint distribution over pixels. This can be easily cast as product of conditional distribution (van den Oord et al., 2016b;c) which makes it tractable to compute. Conditional distributions are estimated by learning to predict new pixels from previously sampled/generated pixels. However, these approaches use fixed sequencing (mostly predicting pixels from top-left to bottom-right) and therefore the learned model can only take a fixed ordering between query and sampled pixels. This implies that these models cannot predict whole images from a few random splashes – similar to what we humans can do given a description of the artist’s painting above.

In this work, our goal is to build computational generative models that can achieve this – given information about some *random* pixels and their associated color values, we aim to predict a *distribution* over images consistent with the evidence. To this end, we show that it suffices to learn a function that estimates the distribution of possible values at any query location conditioned on an arbitrary set of observed samples. We present an approach to learn this function in a self-supervised manner, and show that it can allow answering queries that previous sequential autoregressive models cannot *e.g.* mean image given observed pixels, or computing image distribution given random observations. We also show that our proposed framework is generally applicable beyond images and can be learned to generate generic dense spatial signals given corresponding samples.

2. Formulation

Given the values of some (arbitrary) pixels, we aim to infer what images are likely conditioned on this observation. More formally, for any pixel denoted by random variable \mathbf{x} , let $\mathbf{v}_\mathbf{x}$ denote the value for that pixel and let $S_0 \equiv \{\mathbf{v}_{\mathbf{x}_k}\}_{k=1}^K$ correspond to a set of such sampled values. We are then interested in modeling $p(I|S_0)$ *i.e.* the conditional distribution over images I given a set of sample pixel values S_0 .

From Image to Pixel Value Distribution. We first note that an image is simply a collection of values of pixels in a discrete grid. Assuming an image has N pixels with locations denoted as $\{g_n\}_{n=1}^N$, our goal is therefore to model $p(I|S_0) \equiv p(\mathbf{v}_{g_1}, \mathbf{v}_{g_2}, \dots, \mathbf{v}_{g_N}|S_0)$. Instead of modeling

¹Facebook AI Research ²Carnegie Mellon University. Correspondence to: Shubham Tulsiani <shubhtuls@fb.com>.

this joint distribution directly, we observe that it can be further factorized as a product of conditional distributions using the chain rule:

$$p(\mathbf{v}_{g_1}, \mathbf{v}_{g_2}, \dots, \mathbf{v}_{g_N} | S_0) = \prod_n p(\mathbf{v}_{g_n} | S_0, \mathbf{v}_{g_1}, \dots, \mathbf{v}_{g_{n-1}})$$

Denoting by $S_n \equiv S_0 \cup \{\mathbf{v}_{g_j}\}_{j=1}^n$, we obtain:

$$p(I|S_0) = \prod_n p(\mathbf{v}_{g_n} | S_{n-1}) \quad (1)$$

Sample Conditioned Value Prediction. The key observation from Eq. 1 is that all the factors are in the form of $p(\mathbf{v}_x|S)$. That is, the only queries we need to answer are: ‘given some observed samples S , what is the distribution of possible values at location x ?’ To learn a sample conditioned generative model for images, we therefore propose to learn a function f_θ to infer $p(\mathbf{v}_x|S)$ for arbitrary inputs x and S . Concretely, we formulate our task as that of learning a function $f_\theta(x, \{(x_k, v_k)\})$ that can predict the value distribution at an arbitrary query location x given a set of arbitrary sample (position, value) pairs $\{(x_k, v_k)\}$.

In summary:

- The task of inferring $p(I|S_0)$ can be reduced to queries of the form $p(\mathbf{v}_x|S)$.
- We propose to learn a function $f_\theta(x, \{(x_k, v_k)\})$ that can predict $p(\mathbf{v}_x|\{v_{x_k}\})$ for arbitrary inputs.

While we used images as a motivating example, our formulation is also applicable for modeling distributions of other dense spatially varying signals. For RGB images, $x \in \mathbb{R}^2, v \in \mathbb{R}^3$, but other spatial signals e.g. polynomials ($x \in \mathbb{R}^1, v \in \mathbb{R}^1$), 3D shapes represented as Signed Distance Fields, ($x \in \mathbb{R}^3, v \in \mathbb{R}^1$) or videos ($x \in \mathbb{R}^3, v \in \mathbb{R}^3$) can also be handled by learning $f_\theta(x, \{(x_k, v_k)\})$ of the corresponding form (see Section 6).

3. Related Work

Autoregressive Generative Models. Closely related to our work, autoregressive generative modeling approaches also factorize the joint distribution into per-location conditional distributions. Seminal works such as Wavenet (van den Oord et al., 2016a), PixelRNN (van den Oord et al., 2016c) and PixelCNN (van den Oord et al., 2016b) showed that we can learn the distribution over the values of the ‘next’ timestep/pixel given the values of the previous ones, and thereby learn a generative model for the corresponding domain (speech/images). Subsequent approaches have further improved over these works by modifying the parametrization (Salimans et al., 2017), incorporating hierarchy (van den Oord et al., 2017; Razavi et al., 2019), or (similar to ours) foregoing convolutions in favor of alternate

base architectures (Chen et al., 2020; Parmar et al., 2018) such as Transformers (Vaswani et al., 2017).

While this line of work has led to impressive results, the core distribution modeled is that of the ‘next’ value given ‘previous’ values. More formally, while we aim to predict $p(\mathbf{v}_x|S)$ for arbitrary x, S , the prior autoregressive generative models only infer this for cases where S contains pixels in some sequential (e.g. raster) order and x is the immediate ‘next’ position. Although using masked convolutions can allow handling many possible inference orders (Jain et al., 2020), the limited receptive field of convolutions still limits such orders to locally continuous sequences. Our work can therefore be viewed as a generalization of previous ‘sequential’ autoregressive models in two ways: a) allowing *any* query position x , and b) handling arbitrary samples S for conditioning. This allows us to answer questions that prior autoregressive models cannot e.g. ‘if the top-left pixel is blue, how likely is the bottom-right one to be green?’, ‘what is the mean image given some observations?’, or ‘given values of 10 specific pixels, sample likely images’.

Implicit Neural Representations. There has been a growing interest in learning neural networks to represent 3D textured scenes (Sitzmann et al., 2019), radiance fields (Mildenhall et al., 2020; Martin-Brualla et al., 2021; Zhang et al., 2020) or more generic spatial signals (Sitzmann et al., 2020; Tancik et al., 2020). The overall approach across these methods is to represent the underlying signal by learning a function g_ϕ that maps query positions x to corresponding values v (e.g. pixel location to intensity). Our learned $f_\theta(\cdot, \{(x_k, v_k)\})$ can similarly be thought of as mapping query positions to a corresponding value (distribution), while being conditioned on some sample values. A key difference however, is the ability to generalize – the above mentioned approaches learn an independent network per instance e.g. a separate g_ϕ is used to model each scene, therefore requiring from thousands to millions of samples to fit g_ϕ for a specific scene. In contrast, our approach uses a common f_θ across all instances and can therefore generalize to unseen ones given only a sparse set of samples. Although some recent approaches (Xu et al., 2019; Park et al., 2019; Mescheder et al., 2019) have shown similar ability to generalize and infer novel 3D shapes/scenes given input image(s), these cannot handle sparse input samples and do not allow inferring a distribution over the output space.

Latent Variable based Generative Models. Our approach, similar to sequential autoregressive models, factorizes the image distribution as products of per-pixel distributions. An alternate approach to generative modeling, however, is to transform a prior distribution over latent variables to the output distribution via a learned decoder. Several approaches allow learning such a decoder by leveraging diverse objectives e.g. adversarial loss (Goodfellow et al.,

2014), variational bound on the log-likelihood (Kingma & Welling, 2013), nearest neighbor matching (Bojanowski et al., 2018; Li & Malik, 2018), or the log-likelihood with a restricted decoder (Rezende & Mohamed, 2015). While all of these methods allow efficiently generating new samples from scratch (by randomly sampling in the latent space), it is not straightforward to condition this sampling given partial observations – which is the goal of our work.

Bayesian Optimization and Gaussian Processes. As alluded to earlier, any spatial signal can be considered a function from positions to values. Our goal is then to infer a distribution over possible functions given a set of samples. This is in fact also a central problem tackled in bayesian optimization (Brochu et al., 2010), using techniques such as gaussian processes (Rasmussen, 2003) to model the distribution over functions. While the goal of these approaches is similar to ours, the technique differs significantly. These classical methods assume a known prior over the space of functions and leverage it to obtain the posterior given some samples (we refer the reader to (Murphy, 2012) for an excellent overview). Such a prior over functions (that also supports tractable inference), however, is not easily available for complex signals such as images or 3D shapes – although some weak priors (Ulyanov et al., 2018; Osher et al., 2017) do allow impressive image restoration, they do not enable generation given sparse samples. In contrast, our approach allows learning from data, and can be thought of as learning this prior as well as performing efficient inference via the learned model f_θ .

4. Learning and Inference

Towards inferring the distribution of images given a set of observed samples, we presented a formulation in Section 2 that reduced this task to that of learning a function to model $p(\mathbf{v}_x | \{\mathbf{v}_{x_k}\})$. We first describe in Section 4.1 how we parametrize this function and how one can learn it from raw data. We then show in Section 4.2 and Section 4.3 how this learned function can be used to query and draw samples from the conditional distribution over images $p(I|S_0)$. While we use images as the running example, we reiterate that the approach is more generally applicable (as we also empirically show in Section 6).

4.1. Learning to Predict Value Distributions

We want to learn a function f_θ that can predict the probability distribution of possible values at any query location \mathbf{x} conditioned on a (arbitrary) set of positions with known values. More formally, we want $f_\theta(\mathbf{x}, \{(\mathbf{x}_k, \mathbf{v}_k)\})$ to approximate $p(\mathbf{v}_x | \{\mathbf{v}_{x_k}\})$.

Distribution Parametrization. The output of f_θ is supposed to be a *distribution* over possible values at location \mathbf{x} and not a single value estimate. How should we parametrize

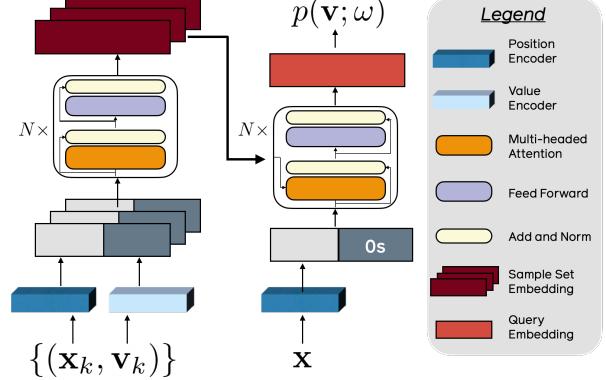


Figure 1. Prediction Model. Given a set of (position, value) pairs $\{(\mathbf{x}_k, \mathbf{v}_k)\}$, our model encodes them using a Transformer (Vaswani et al., 2017) encoder. A query position \mathbf{x} is then processed in context of this encoding and a value distribution is predicted (parametrized by ω).

this distribution? Popular choices like gaussian parametrization may not capture the multimodal nature of the distribution e.g. a pixel maybe black or white, but not gray. An alternate is to discretize the output space but this may require a large number of bins e.g. 256^3 for possible RGB values. Following PixelCNN++ (Salimans et al., 2017), we opt for a hybrid approach – we predict probabilities for the value belonging to one of B discrete bins, while also predicting a continuous gaussian parametrization within each bin. This allows predicting multimodal distributions while enabling continuous outputs.

Concretely, we instantiate B bins (roughly) uniformly spaced across the output space where for any bin b , its center corresponds to c^b . The output distribution is then parametrized as $\omega \equiv \{(q^b, \mu^b, \sigma^b)\}_{b=1}^B$. Here $q^b \in \mathbb{R}^1$ is the probability of assignment to bin b , $c^b + \mu^b$ is the mean of the corresponding gaussian distribution with uniform variance $\sigma^b \in \mathbb{R}^1$. Assuming the values $\mathbf{v} \in \mathbb{R}^d$, our network therefore outputs $\omega \in \mathbb{R}^{B \times (d+2)}$. We note that this distribution is akin to a mixture-of-gaussians, and given a value \mathbf{v} , we can efficiently compute its likelihood $p(\mathbf{v}; \omega)$ under it (see appendix for details). We can also efficiently compute the expected value $\bar{\mathbf{v}}$ as:

$$\bar{\mathbf{v}} = \int p(\mathbf{v}; \omega) \mathbf{v} d\mathbf{v} = \sum_{b=1}^B q^b (\mu^b + c^b) \quad (2)$$

Model Architecture. Given a query position \mathbf{x} , we want $f_\theta(\mathbf{x}, \{(\mathbf{x}_k, \mathbf{v}_k)\})$ to output a value distribution as parametrized above. There are two design considerations that such a predictor should respect: a) allow a variable number of input samples $\{(\mathbf{x}_k, \mathbf{v}_k)\}$, and b) be permutation-invariant w.r.t. the samples. We leverage the Transformer (Vaswani et al., 2017) architecture as our backbone as it satisfies both these requirements. As depicted in Figure 1, our model can be considered as having two stages:

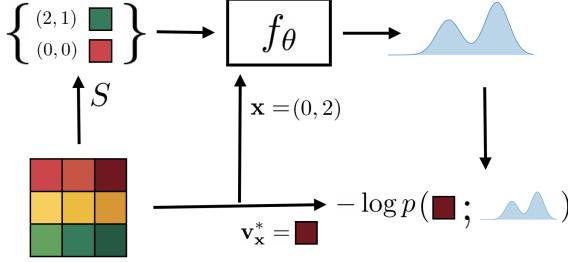


Figure 2. Training Overview. Given an image, we randomly sample pixels to obtain the conditioning set S as well as a query pixel \mathbf{x} with value \mathbf{v}_x^* . Our model predicts the conditional value distribution for this arbitrary query location and we use the negative log-likelihood for the true value as our learning objective.

a) an encoder that, independent of the query \mathbf{x} , processes the input samples $\{(\mathbf{x}_k, \mathbf{v}_k)\}$ and computes a per-sample embedding, and b) a decoder that predicts the output distribution by processing the query \mathbf{x} in context of the encodings.

As shown in Figure 1, we first independently embed each input sample $(\mathbf{x}_k, \mathbf{v}_k)$ using position and value encoding modules respectively, while following the insight from (Tancik et al., 2020) to use fourier features when embedding positions. These per-sample encodings are then processed by a sequence of multi-headed self-attention modules (Vaswani et al., 2017) to yield the encoded representations for the input samples. The query position \mathbf{x} is similarly embedded, and processed via multi-headed attention modules in context of the sample embeddings. A linear decoder finally predicts $\omega \in \mathbb{R}^{B \times (d+2)}$ to parametrize the output distribution.

Training Objective. Recall that our model $f_\theta(\mathbf{x}, \{(\mathbf{x}_k, \mathbf{v}_k)\})$ aims to approximate $p(\mathbf{v}_x | \{\mathbf{v}_{x_k}\})$ for arbitrary query positions \mathbf{x} and sample sets $S \equiv \{\mathbf{v}_{x_k}\}$. Given a collection of training images, we can in fact generate training data for this model in a self-supervised manner. As illustrated in Figure 2, we can simply sample arbitrary \mathbf{x}, S from any image, and maximize the log-likelihood of the true value \mathbf{v}_x^* under the predicted distribution $p(\mathbf{v}_x | \{\mathbf{v}_{x_k}\})$.

While we described the processing for a single query position \mathbf{x} , it is easy to parallelize inference and process a batch of queries Q conditioned on the same input sample set S . In this case, we can consider the model as independently predicting $p(\mathbf{v}_x | \{\mathbf{v}_{x_k}\})$ for each $\mathbf{x} \in Q$. Instead of using a single query \mathbf{x} , we therefore use a batch of queries Q and minimize the negative log-likelihood across them. More formally, given a dataset D of images, we randomly sample an image I , and then choose arbitrary sample and query sets S, Q , and minimize the expected negative log-likelihood of the true values as our training objective:

$$L = \mathbb{E}_{I \sim D} \mathbb{E}_{S, Q \sim I} \mathbb{E}_{\mathbf{x} \sim Q} -\log p(\mathbf{v}_x^*; \omega) \quad (3)$$

where, $\omega = f_\theta(\mathbf{x}; \{(\mathbf{x}_k, \mathbf{v}_k)\})$

4.2. Inferring Marginals and Mean

Section 4.1 introduced our approach to enable learning f_θ that can approximate $p(\mathbf{v}|S)$. But given such a learned function, what can it enable us to do? One operation that we focus on later in Section 4.3 is that of sampling images $I \sim p(I|S)$. However, there is another question of interest which is not possible to answer with the previous sequential autoregressive models (van den Oord et al., 2016b;a), but is efficiently computable using our model: ‘what is the expected image \bar{I} given the samples S ?’.

We reiterate that an image can be considered as a collection of values of pixels located in a discrete grid $\{g_n\}_{n=1}^N$. Instead of asking what the expected image \bar{I} is, we can first consider a simpler question – what is the expected value $\bar{\mathbf{v}}_{g_n}$ for the pixel g_n given S ? By definition:

$$\bar{\mathbf{v}}_{g_n} = \int p(\mathbf{v}_{g_n} | S) \mathbf{v}_{g_n} d\mathbf{v}_{g_n}$$

As our learned model f_θ allows us to directly estimate the marginal distribution $p(\mathbf{v}_{g_n} | S)$, the above computation is extremely efficient to perform and can be done independently across all locations in the image grid $\{g_n\}_{n=1}^N$.

$$\bar{\mathbf{v}}_{g_n} = \int p(\mathbf{v}; \omega_n) \mathbf{v} d\mathbf{v}; \quad \omega_n = f_\theta(g_n, \{(\mathbf{x}_k, \mathbf{v}_k)\}) \quad (4)$$

Given the estimate of $\bar{\mathbf{v}}_{g_n}$, the mean image \bar{I} is then just the image with each pixel assigned its mean value $\bar{\mathbf{v}}_{g_n}$ i.e. $\bar{I} \equiv \{\bar{\mathbf{v}}_{g_n}\}_{n=1}^N$. The key difference compared to sequential autoregressive models (van den Oord et al., 2016b;a) that enables our model to compute this mean image is that our model allows computing $p(\mathbf{v}_{g_n} | S)$ for *any* location g_n , whereas approaches like (van den Oord et al., 2016b;a) can only do so for the ‘next’ pixel.

4.3. Autoregressive Conditional Sampling

One of the driving motivations for our work was to be able to sample the various likely images conditioned on a sparse set of pixels with known values. That is, we want to be able to draw samples from $p(I|S_0)$. Equivalently, to sample an image from $p(I|S_0)$, we need to sample the values at each pixel $\{\mathbf{v}_{g_n}\}$ from $p(\mathbf{v}_{g_1}, \mathbf{v}_{g_2}, \dots, \mathbf{v}_{g_N} | S_0)$.

As we derived in Eq. 1, this distribution can be factored as a product of per-pixel conditional distributions. We can therefore sample from this distribution autoregressively – sampling one pixel at a time, with subsequent pixels being informed by ones sampled prior. Concretely, we iteratively perform the following computation:

$$\omega_n = f_\theta(g_n, \{\mathbf{x}_k, \mathbf{v}_k\} \cup \{g_j, \mathbf{v}'_j\}_{j=1}^{n-1}) \quad (5)$$

$$\mathbf{v}'_n \sim p(\mathbf{v}; \omega_n) \quad (6)$$

Here, ω_n denotes the parameters for the predicted distribution for the pixel g_n . Note that this prediction takes into



Figure 3. Inferred Mean Images. We visualize the mean image predicted by our learned model on random instances of the Cat Faces dataset. Top row: ground-truth image. Rows 2-8: Predictions using increasing number of observed pixels $|S|$.

account not just the initial samples S_0 , but also the subsequent $n - 1$ samples (hence the difference from ω_n in Eq. 4). v'_n represents a value then sampled for the pixel g_n from the distribution parametrized by ω_n .

Randomized Sampling Order. While we sample the values one pixel at a time, the ordering of pixels g_1, \dots, g_N need not correspond to anything specific *e.g.* it is not necessary that g_1 should be the top-left pixel and g_N be the bottom-right one. In fact, as our model f_θ is trained using arbitrary sets of samples S , using a structured sampling ordering *e.g.* raster order would make the testing setup differ from training. Instead, for every sample $I \sim p(I|S)$ that we draw, we use a new *random* order in which the pixels of the image grid are sampled.

Sidestepping Memory Bottlenecks. As Eq. 5 indicates, the input to f_θ when sampling the $(n + 1)^{th}$ pixel is a set of size $K + n$ – the initial K observations and the subsequent n samples. Unfortunately, our model’s memory requirement, due to the self-attention modules, grows cubically with this input size. This makes it infeasible to autoregressively sample a very large number of pixels. However, we empirically observe that given a sufficient number of (random) samples, subsequent pixel value distributions do not exhibit a high variance. We leverage this observation to design a hybrid sampling strategy. When generating an image with N pixels, we sample the first N' (typically 2048) autoregressively *i.e.* following Eq. 5 and Eq. 6. For the remaining $N - N'$ pixels, we simply use their mean value estimate conditioned on the initial and generated $K + N'$ samples (using Eq. 4). While this may lead to some loss in detail, we qualitatively show that the effects are not prohibitive and that the sample diversity is preserved.

5. Experiments

To qualitatively and quantitatively demonstrate the efficacy of our approach, we consider the task of generating images given a set of pixels with known values. The goal of our experiments is twofold – a) to validate that our predictions account for the observed pixels, and b) to show that the generated samples are diverse and plausible.

Datasets. We examine our approach on three different image datasets – CIFAR10 (Krizhevsky, 2009), MNIST (Le-Cun et al., 1998), and the Cat Faces (Wu et al., 2020) dataset while using the standard image splits. Note that we only require the images for training – class or attribute labels are not leveraged for learning our models *i.e.* even on CIFAR10, we learn a class-agnostic generative model.

Training Setup. We vary the number of observed pixels S randomly between 4 and 2048 (with uniform sampling in log-scale), while the number of query samples Q is set to 2048. During training, the locations x are treated as varying over a continuous domain, using bilinear sampling to obtain the corresponding value – this helps our implementation be agnostic to the image resolution in the dataset. While we train a separate network f_θ for each dataset, we use the exact *same* model, hyper-parameters *etc.* across them.

Qualitative Results: Mean Image Prediction. We first examine the expected image \bar{I} inferred by our model given some samples S . We visualize in Figure 3 our predictions on the Cat Faces dataset using varying number of input samples. We observe that even when using as few as 4 pixels in S , our model predicts a cat-like mean image that, with some exceptions, captures the coarse color accurately.



Figure 4. Image Samples. Sample images generated by our learned model on three datasets (left: MNIST, middle: Cat Faces, right: CIFAR10) given $|S| = 32$ observed pixels. Top row: ground-truth image from which S is drawn. Row 2: A nearest neighbor visualization of S – for each image pixel we assign it the color of the closest observed sample in S . Rows 3-5: Randomly sampled images from $p(I|S)$.

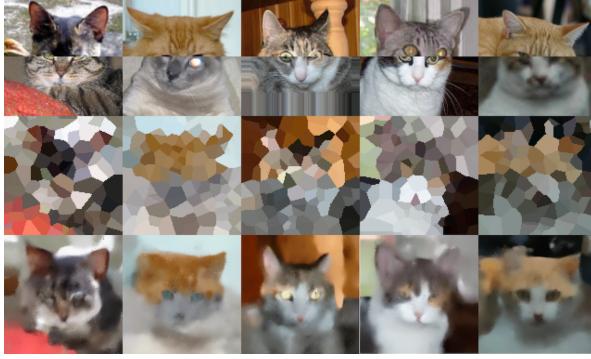


Figure 5. Image Composition. Generation results when drawing pixels from two different images. Top row: the composed image from which S is drawn. Row 2: A nearest neighbor visualization of S . Row 3: Randomly sampled image from $p(I|S)$.

A very small number of pixels, however, is not sufficiently informative of the pose/shape of the head, which become more accurate given around 100 samples. As expected, the mean image becomes closer to the true image given additional samples, with the later ones even matching finer details *e.g.* eye color, indicating that the distribution $p(I|S)$ reduces in variance as $|S|$ increases.

Qualitative Results: Sampling Images. While examining the mean image assures us that our average prediction is meaningful, it does not inform us about samples drawn from $p(I|S)$. In Figure 4, we show results on images from each of the three datasets considered using $|S|=32$ randomly observed pixel values in each case. We see that the sampled images vary meaningfully (*e.g.* face textures) while preserving the coarse structure, though we do observe some artefacts *e.g.* missing horse legs.

As an additional application, we can generate images by mixing pixel samples from different images. We showcase some results in Figure 5 where we show one generated im-

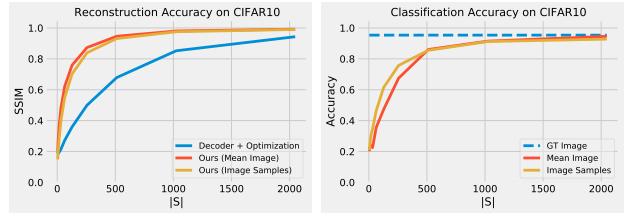


Figure 6. Reconstruction Accu-Figure 7. Classification Accuracy of generated images.

age given some pixels from top/bottom of two different images. We see that, despite some mismatch in the alignment/texture of the underlying faces, our model is able to compose them to generate a plausible new image.

Reconstruction and Classification Accuracy. In addition to visually inspecting the mean and sampled images, we also quantitatively evaluate them using reconstruction and classification based metrics on the CIFAR10 dataset. First, we measure how similar our obtained images are to the underlying ground-truth image. Figure 6 plots this accuracy for varying size of S – we compute this plot using 128 test images, varying $|S|$ from 4 to 2048 for each. When reporting the accuracy for sampled images, we draw 3 samples per instance and use the average performance. We also report a baseline that uses a pretrained decoder(from a VAE) and optimizes the latent variable to best match the pixels in S (see appendix for details). We observe that our predicted images, more so than the baseline, match the true image. Additionally, the mean image is slightly more ‘accurate’ in terms of reconstruction than the sampled ones – perhaps because the diversity of samples makes them more different.

We also plot the classification accuracy of the generated images in Figure 7. To do so, we use a pretrained ResNet-18 (He et al., 2016) based classifier and measure whether the correct class label is inferred from our generated images.

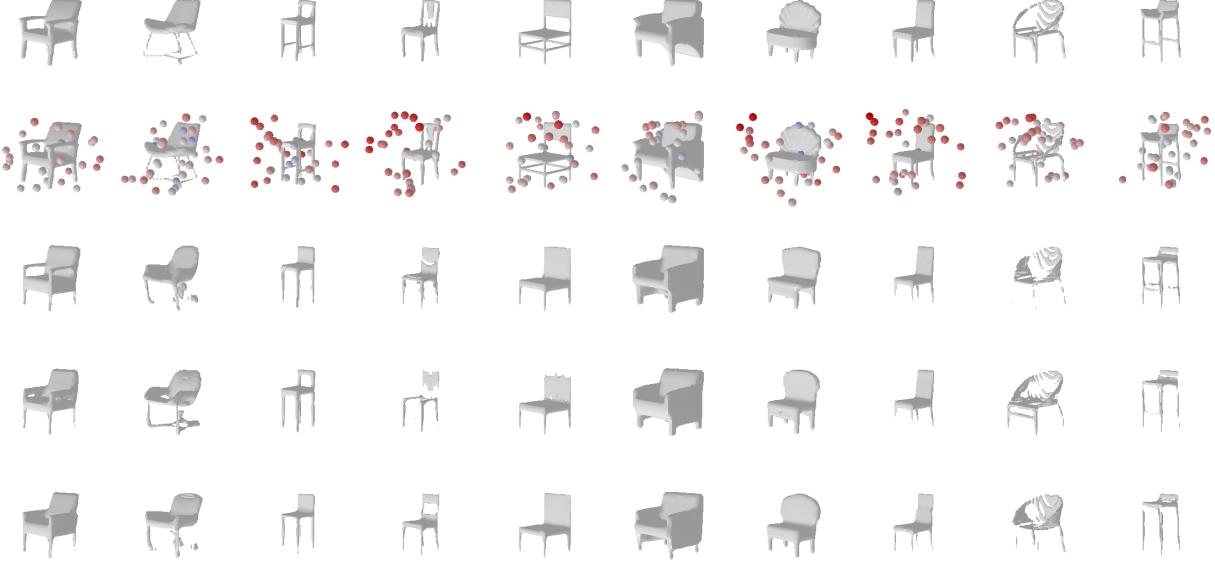


Figure 8. Shape Generation. Sample 3D shapes generated given $|S| = 32$ observed SDF values at random locations. Top row: ground-truth 3D shape. Row 2: A visualization of S – a sphere is centred at each position with color indicating value (red implies higher SDF). Rows 3-5: Randomly sampled 3D shapes from our predicted conditional distribution.

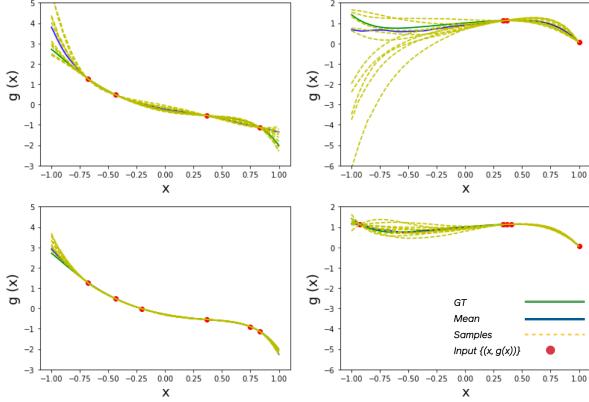


Figure 9. Polynomial Prediction. Mean and sampled polynomials generated by our learned model. Row 1: Predictions using $|S| = 4$ samples (red dots). Row 2: Predictions using $|S| = 6$.

Interestingly, we see that even if using images generated from as few as 16 pixels, we obtain about a 30% classification accuracy (or over 60% with 128 pixels). As we observe more pixels, the accuracy matches that of using the ground-truth images. Finally, we see that using the sampled images yields better results compared to the mean image, as the sampled ones look more ‘real’.

6. Beyond Images: 1D and 3D Signals

While we leveraged our proposed framework for generating images given some pixel observations, our formulation



Figure 10. Video Synthesis. Sample videos generated by our model given $|S|=1024$ observed pixels across 34 frames. Top row: 4 uniformly sampled frames of the ground-truth video. Row 2: A nearest neighbor visualization of S . Rows 3-5: Randomly sampled videos from the predicted conditional distribution.

is applicable beyond images. In particular, assuming the availability of (unlabeled) examples, our approach can learn to generate any dense spatial signal given some (position, value) samples. In this section, we empirically demonstrate this by learning to generate 1D (polynomial) and 3D (shapes and videos) signals using our framework.

We would like to emphasize that across these settings, where we are learning to generate rather different spatial signals, we use the *same* training objective and model design. That is, except for the dimensionality of input/output layers and distribution parametrization to handle the corresponding

inputs/outputs $\mathbf{x} \in \mathbb{R}^x$, $\mathbf{v} \in \mathbb{R}^v$, our model or learning objective is not modified in any way specific to the domain.

6.1. Polynomial Prediction

As an illustrative example to study our method, we consider a classical task – given a sparse set of $(\mathbf{x}, g(\mathbf{x}))$ pairs, where $\mathbf{x}, g(\mathbf{x}) \in \mathbb{R}^1$, we want to predict the value of g over its domain. We randomly generate 6-degree polynomials, draw from 4 to 20 samples to obtain S , and learn f_θ to predict distribution of values at $|Q|=20$ query locations. One simplification compared to the model used for images is we use $B = 1$ instead of $B = 256$ (*i.e.* a simple gaussian distribution) to parametrize the output distribution.

We visualize our predictions in Figure 9, where the columns correspond to different polynomials, and the rows depict our results with varying number of inputs in S . We see that the various sample functions we predict are diverse and meaningful, while being constrained by the observed position, value pairs. Additionally, as the number of observations in S increase, the variance of the function distribution reduces and matches the true signal more closely.

6.2. Generating 3D Shapes

We next address the task of generating 3D shapes represented as signed distance fields (SDFs). We consider the category of chairs using models from 3D Warehouse (3DW), leveraging the subset recommended by Chang *et al.* (Chang *et al.*, 2015). We use the train/test splits provided by (Xu *et al.*, 2019), with 5268 shapes used for training, and 1311 for testing. We extract a SDF representation for each shape as a grid of size 64^3 , with each location recording a continuous signed distance value – this dense representation is better suited for our approach compared to sparse occupancies. Our training procedure is exactly the same as the one used for 2D images – we sample the SDF grid at random locations to generate S, Q , with the number of samples in S varying from 4 to 2048, and $|Q|$ being 2048.

We present some *randomly chosen* 3D shapes generated by our model when using $|S| = 32$ in Figure 8. While we actually generate a per-location signed distance value, we extract a mesh using marching cubes to visualize this prediction. As the results indicate, even when using only 32 samples from such a high-dimensional 3D spatial signal, our model is able to generate diverse and plausible 3D shapes. In particular, even though this is not explicitly enforced, our model generates symmetric shapes and the variations are semantically meaningful as well as globally coherent *e.g.* slope of chair back, handles with or without holes. However, as our model generates the SDF representation, and does not directly produce a mesh, we often see some artefacts in the resulting mesh *e.g.* disconnected components, which can occur when thresholding a slightly inconsistent SDF.

6.3. Synthesizing Videos

Lastly, we examine the domain of ‘higher-dimensional’ images (*e.g.* videos). In particular, we use the subset of ‘beach’ videos in the TinyVideos dataset (Vondrick *et al.*, 2016; Thomee *et al.*, 2016) (with a random 80% – 20% train-test split) and train our model to generate video clips with 34 frames. Note that these naturally correspond to 3D spatial signals, as the position \mathbf{x} includes timeframe $\in \mathbb{R}^1$ in addition to a pixel coordinate.

We train our model f_θ to generate the underlying signal distribution given sparse pixel samples where we randomly choose a frame and pixel coordinate for each sample. We empirically observe that due to the high complexity of the output space, using only a small number of samples does not provide significant information for learning generation. We therefore train our model using more samples than the image generation task – varying $|S|$ between 512 to 2048 (this corresponds to 30 pixels per frame).

We present representative results in Figure 10 but also encourage the reader to see the videos in the project page. Our model generates plausible videos with some variation *e.g.* flow of waves and captures the coarse structure of the output well. However, the predictions lack precise detail. We attribute this to the limited number of pixels we can generate autoregressively (see discussion in Section 4.3 on memory bottlenecks) and hypothesize that a higher number maybe needed for modeling these richer signals.

7. Discussion

We proposed a probabilistic generative model capable of generating images conditioned on a set of random observed pixels, or more generally, synthesizing spatial signals given sparse samples. At the core of our approach is a learned function that predicts value distributions at any query location given an arbitrary set of observed samples. While we obtain encouraging results across some domains, there are several aspects which could be improved *e.g.* scalability, perceptual quality, and handling sparse signals. To allow better scaling, it could be possible to generalize the outputs from distributions over individual pixels to those over a vocabulary of tokens encoding local patches or investigate strategies to better select conditioning subsets (*e.g.* nearest samples). The perceptual quality of our results could be further improved and incorporating adversarial objectives maybe a promising direction. Finally, while our framework allowed generating pixel values, we envision that a similar approach could predict other dense properties of interest *e.g.* semantic labels, depth, generic features.

Acknowledgements. We would like to thank Deepak Pathak and the members of the CMU Visual Robot Learning lab for helpful discussions and feedback.

References

- 3D Warehouse. <https://3dwarehouse.sketchup.com/>.
- Bojanowski, P., Joulin, A., Lopez-Pas, D., and Szlam, A. Optimizing the latent space of generative networks. In *ICML*, 2018.
- Brochu, E., Cora, V. M., and De Freitas, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.
- Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Dhariwal, P., Luan, D., and Sutskever, I. Generative pretraining from pixels. In *ICML*, 2020.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NeurIPS*, 2014.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.
- Jain, A., Abbeel, P., and Pathak, D. Locally masked convolution for autoregressive models. In *UAI*, 2020.
- Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Li, K. and Malik, J. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.
- Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A., and Duckworth, D. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.
- Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.
- Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- Murphy, K. P. *Machine learning: a probabilistic perspective*. 2012.
- Osher, S., Shi, Z., and Zhu, W. Low dimensional manifold model for image processing. *SIAM J. Imaging Sci.*, 2017.
- Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.
- Parmar, N. J., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., and Ku, A. Image transformer. 2018. URL <https://arxiv.org/abs/1802.05751>.
- Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 2003.
- Razavi, A., van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*, 2019.
- Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.
- Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *ICRL*, 2017.
- Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., and Zollhofer, M. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019.
- Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020.
- Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.
- Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.
- Ulyanov, D., Vedaldi, A., and Lempitsky, V. Deep image prior. In *CVPR*, 2018.
- van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

- van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *NeurIPS*, 2016b.
- van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016c.
- van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *NeurIPS*, 2017.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.
- Vondrick, C., Pirsiavash, H., and Torralba, A. Generating videos with scene dynamics. In *NeurIPS*, 2016.
- Wu, S., Rupprecht, C., and Vedaldi, A. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, 2020.
- Xu, Q., Wang, W., Ceylan, D., Mech, R., and Neumann, U. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *NeurIPS*, 2019.
- Zhang, K., Riegler, G., Snavely, N., and Koltun, V. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

Appendix

Log-likelihood under Value Distribution. The predicted value distribution for a query position \mathbf{x} is of the form $p(\mathbf{v}; \omega)$, where $\omega \equiv \{(q^b, \mu^b, \sigma^b)\}_{b=1}^B$. We reiterate $q^b \in \mathbb{R}^1$ is the probability of assignment to bin b , $c^b + \mu^b$ is the mean of the corresponding gaussian distribution with uniform variance $\sigma^b \in \mathbb{R}^1$.

Under this parametrization, we compute the log-likelihood of a value \mathbf{v}^* by finding the closest bin b^* , and computing the log-likelihood of assignment to this bin as well as the log-probability of the value under the corresponding gaussian. We additionally use a weight $\alpha = 0.1$ to balance the classification and gaussian log-likelihood terms.

$$b^* = \operatorname{argmin}_b \|v^* - c^b\|$$

$$\log p(v^*; \omega) \equiv \log q^{b^*} - \alpha(\log \sigma^{b^*} + (\frac{\mathbf{v}^* - c^{b^*} - \mu^{b^*}}{\sigma^{b^*}})^2)$$

VAE Training and Inference. We train a variational autoencoder (Kingma & Welling, 2013) on the CIFAR10 dataset with a bottleneck layer of dimension $4 \times 4 \times 64$ i.e. spatial size 4 and feature size 64. We consequently obtain a decoder D which we use for inference given some observed samples S . Specifically, we optimize for an optimal latent variable the minimizes the reconstruction loss for the observed samples (with an additional prior biasing towards the zero vector). Denoting by $I(\mathbf{x})$ the value of image I (bilinearly sampled) at position \mathbf{x} , the image I^* inferred using a decoder D by optimizing over S can be computed as:

$$z^* = \operatorname{argmin}_z L(D(z), S) + 0.001 * \|z\|^2; \quad I^* = D(z^*)$$

$$L(I, \{(\mathbf{x}_k, \mathbf{v}_k)\}) = \mathbb{E}_k \|I(\mathbf{x}_k) - \mathbf{v}_k\|_1$$