Exploring Test Time Optimization for Discrete Neural Representation Learners

Ishaan P. Chandratreya
Columbia University
New York, NY 10027
ipc2107@columbia.edu

Max H. Helman
Columbia University
New York, NY 10027
mhh2148@columbia.edu

Raghav Mecheri
Columbia University
New York, NY 10027
rm3614@columbia.edu

Abstract

Many methods seek to better understand the latent space of image synthesis models (eg. VAEs, GANs) in order to dissect the generative process and to enable easy *latent space search*: the task of navigating the latent space of the models such that the image produced by it follows some pre-conceived notion. The success and speed of such methods has led to the widespread adoption of generative models as self-supervised representation learners with *test-time optimization*: optimizing on the latent space subject to constraints on the decoded image space can allow constraint-conditioned generation. It is not obvious how to carry search methods for continuous latent spaces over to autoencoders that have discrete latent spaces. In this paper, we provide empirical analysis of the latent space of Vector Quantized Variational Autoencoders (VQVAE), and propose a framework to extend continuous distribution search to the latent space of discrete models. We build and verify the different components of our framework and propose future experiments that we believe would lead to their successful combination. Our code is available at Github.

1 Introduction

The past decade of research in image synthesis has been dominated by generative models, each of which has come with its respective set of advantages and training tricks to help elucidate their best behaviour. While variational methods trained with evidence lower bound ([1] [2] [3]) typically have better ability to capture a range of modes for complicated data distributions (and have been applied to a range of downstream tasks from model-based reinforcement learning ([4] to 3D representations [**?**]), training generative models with discriminative signals, as popularized by the success of Generative Adversarial Networks ([5]) has paved the way for generation at scales and resolution that was not available before. It was shown that such models could also be scaled [6] and trained with controlled mode collapse.

Independently, papers like [7] [8] [9] [10] [11] have investigated the role of such generative models in representation learning. They show that the distributional (latent) space from which these generative models sample can have interesting semantic meanings which may be useful for other tasks. Methods in **test time optimization** and **inversion** both seek to provide quick ways of probing this representational space given some constraints on the co-domain image space. However, they are limited by the properties of the generative model itself: how many modes of the data does it capture and whether it is able to decode to high resolutions are both questions of concern for supervision.

A big leap in high fidelity conditional image synthesis space has been achieved through the recent progress in discrete neural representation learning [12] [13]. Vector-Quantized Variational Autoencoders, in particular, showed that using a small set ($\approx 32^2$) of latent codes, each of which can take on the same discrete set of values (≈ 512), it is still possible to achieve high quality reconstruction and generalization. Note that despite being called **variational**, there is nothing fundamentally variational

COMS 6998 Self Supervised Learning (Spring 2022), Prof. Richard Zemel.

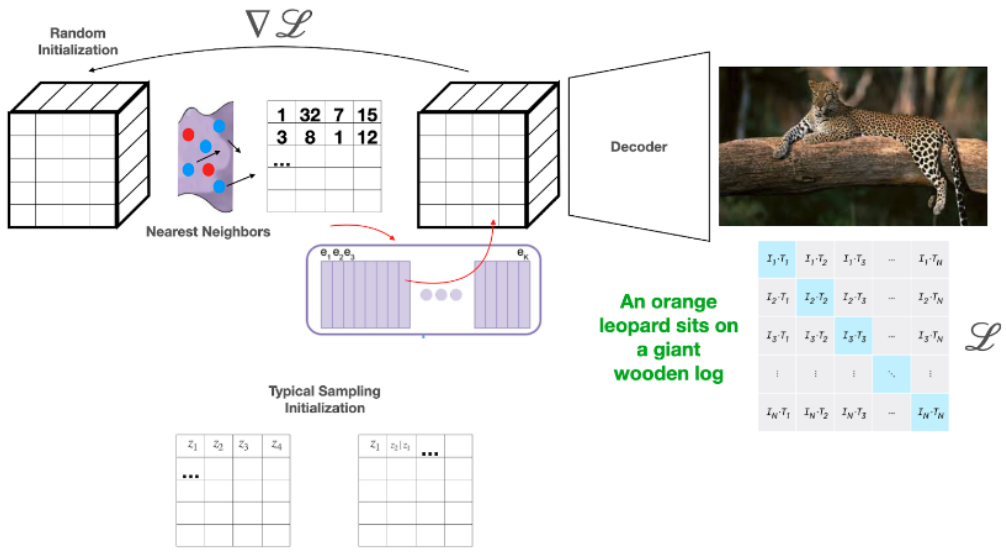

Figure 1: Searching the Latent Space of a Discrete 'Variational' Autoencoder is a challenging task

about this style of auto-encoder as there is no sampling happening in the latent space; instead, the latent codes and the model are jointly optimized for reconstruction, commitment and alignment. It is the combinatorial blowup of all possible discrete combinations, and not variational inference from distributional assumptions on the latent space, that allows for various modes of the image space to be represented. Hence, further training on the latent space is required to sample new images from them. Since VQVAEs have arrived, variations on trained discrete autoencoders have achieved state of the art results in the image synthesis fields, and the compact categorical nature of the latent space has allowed for all sorts of conditioning. The hugely popular model DALL.E [12] is essentially a VQVAE with a text conditioned sampler for its latent space; the variance and novelty in its results show that discrete generative models might be capturing rich representations which we would like to take advantage of via test-time optimization. However, there has been little progress in search the latent space on these discrete models. In fact, it is unclear what searching the latent space of the model would even look like, given that their exist no distributional structure over them.

In this work, our goal is to make progress toward resolving this problem given the huge representational potential of discrete VAEs. Our contributions are three-fold: (1) First through we provide a detailed review of the most powerful latent-space based generative models in literature today that are amenable to conditioning (on class, text, etc) with a *special emphasis* on the ease with which we can search their latent space, contextualizing VQVAEs in this landscape (2) we provide an analysis of discrete variational autoencoders for simple image-based datasets, compare their performance on ablations to their latent structure and offer insights from the failures of simple gradient based methods to navigate their latent space, and (3) we provide a flow based framework with preliminary synthetic experiments, which we believe if scaled correctly has the potential to close the gap between test time optimization for discrete and continuous generative models.

2 Related Work

Image Synthesis and Latent Spaces A wide class of algorithms have been applied to the image synthesis task. Perhaps the most popular of them are Generative Adversarial Networks (GANS). GANs consist of a generator network G and a discriminator network D that are trained adversarially: for the generator to generate images that fool the discriminator and the discriminator to in turn become better at telling reals and fake images apart. They are also amenable to a wide variety of conditioning:

for example, in text-conditioned GANS,a text classifier learns a correspondence function with images to capture whether a generated image matches the description [14]. However, GANs are difficult to train, require significant compute, and are susceptible to mode collapse in the sense that in their default setup, there is nothing preventing them from allocating all distributional mass on a single mode of the data distribution.

Variational Autoencoders (VAEs) [1] are an older but equally popular method for synthesis that face this issue to a lower extent. They consist of an encoder that produces latents conditioned on images, a prior distribution on the latents and a decoder which goes back from the latents to image space. Typically, an identity-variance Gaussian distribution parametrizes the prior on the latent, and the model is trained for reconstruction and a posterior that resembles the prior while still producing the right reconstructions. Despite the sampling step being non-differentiable, the reparametrization trick allows us to train them effectively. While being less susceptible to mode collapse, these models carry over the baggage from variational inference, such as posterior collapse, and tend to produce lower quality reconstructions.

Vector Quantized Variational Autoencoders, [13] which we review in greater detail in section 3, belong to a new wave of models that use the overarching philosophy of a VAE but have the latent vectors lie inside a discrete set of values. They leads to great compression in the set of values the latent space can take on, providing better possibility to scale. They form the basis of many large 'foundation' generative models at the frontier of AI research today. We provide a detailed review of how they are trained in Section 3.

Note that while diffusion models represent the state-of-the-art in synthesis today [15], we omit them from our work since they do not have an explicit latent space that can be searched over. Instead, they learn how to reverse a Markovian diffusion process and construct relevant examples from the noise. We note that despite their success in synthesis, the lack of ability to search a latent space (given the lack of existence of one) should be counted as one of the drawbacks of using these models.

Test Time Optimization Test Time Optimization (TTO) has many meanings in representation learning literature, with nuances coming from particular domain and application. Our primary reference for defining the test-time optimization is PULSE [16]. PULSE works on the assumption that the high dimensional image manifold is approximated by a generative model that has been trained to produce samples from this manifold. Since the generative model is a differentiable function from latents to the images, we can traverse the latent space using gradients from a constraint placed on the output of the generative model corresponding to these latents. The constraint PULSE uses is that the produced image must downscale correctly to a reference low resolution image, and hence it uses test time optimization to solve a super-resolution task. PULSE relies on the fact that GANs sample from a Gaussian prior since it uses properties of high dimensional gaussians to accelerate search and constrain navigation in high density regions of the latent space. Such an assumption is not always available, and constraining the search to high density regions of the latent space is typically very challenging. Building on PULSE we define the test time optimization problem in Section 4.

Modeling and Matching Distributions Our work also builds on progresses in modeling and matching distributions in many senses. Given a set of samples from a compressed discrete latent space of the VQVAE corresponding to actual images, we seek to learn a discrete categorical distribution on them. for this, we rely on literature in self-supervised autoregressive models [17] that define joint categorical distributions, which we review in section 5.3. For taking a continuous approximation of discrete distributions, we rely on the Gumbel-Max trick and the Gumbel softmax distribution, which we review in the appendix . Finally, to map from a gaussian-like distribution to a set of Gumbel-Softmaxs we rely on literature in normalizing flows, which learn a bijective mapping between distributions sharing the same support. We review these in Section 5.4

3 Preliminary: Discrete Neural Representation Learning

As opposed to traditional VAEs, Vector Quantized VAEs have categorical posterior and prior distributions. Sampling from such distributions yields a sequence of indices in a fixed range, and these are used to retrieve entries from a *codebook* to construct a representational tensor which is then decoded. Precisely, we define the codebook as $c \in \mathbb{R}^{k \times d}$, containing k different entries of d dimensions each. We further define an encoder function e (which acts as our posterior) and a function **NN** which takes the output of the encoder, and, at every spatial location, replaces the vector with its nearest neighbour

present inside the codebook. We denote this output tensor $q(x)$. A decoder then reconstructs back to the original image dimensions from $q(x)$. Note that the **NN** operation is non differentiable. Hence, to past gradients to the encoder, in analogy to how we copy over the nearest embedding in the forward pass, we simpy copy over the gradients present on the $q(x)$ tensor to the $e(x)$ tensor during the backward pass. We then train the model jointly for:

$$L = \log p(x|q(x)) + ||sg(e(x)) - c||_2^2 + \beta||e(x) - sg(c)||_2^2$$

where sg is the stop gradient operator equivalent to the **.detach()** call in PyTorch. The first term optimizes for reconstruction, but this term offers no gradients to the codebook as the gradient goes straight through the $q(x)$ to $e(x)$ due to our trick. The other two losses come from vector quantization family of work, where the encoder inputs and codebook are gradually moved toward each other.

4 Problem Definition

4.1 General

The notation used in this section is re-used in all algorithms defined hereafter. Given a generative model $g \in C^\infty : \mathbb{R}^d \mapsto \mathcal{X}$ where $\mathcal{X}$ represents the image space, our goal is to search for a $z \in \mathbb{R}^d$ such that z produces the optimal image for a given constraint. In particular, we parametrize success on the constraint using a loss function $\mathcal{L}$ and a target t such that we want $\mathcal{L}(g(z), t)$ to be very low. Note that the target t should be sampled from a space that has some semantic relationship to the image space $\mathcal{X}$, and that the chosen loss should be adjusted according to the choice of the target. For example, the target can be another image $t \in \mathcal{X}$ with $\mathcal{L}$ then being used to represent some notion of image-image similarity (Eg. MSE loss in pixel space), or it could be some other modality, such as text that represents the description of the intended image to be produced, with $\mathcal{L}$ measuring cross-modal similarity (Eg. CLIP loss). While our figure suggests the latter to motivate the high level flexibility of this framework, all our experiments for the scope of this class are done with the target being a random test image, and the loss measured being a simple mean squared error loss between the generated image and the target image.

4.2 Specific to VQVAEs

We are interested in investigating the problem definition from the previous section where g is a discrete variational autoencoder. While the problem definition is widely applicable for many different kinds of g, including VQVAE, it is worth noting that an interesting property emerges while considering the problem in light of VQVAES. Essentially, due to the codebook constraint, we are searching for a set of *indices* of the codeboook that produce the required constrained image. This makes the problem combinatorial in nature and we show in the appendix that the problem is essentially atleast as difficult as integer programming. Figure 1 nicely motivates the nature of the problem: we must look for the optimal index at every spatial location in the latent space, such that replacing the encoded output with the nearest neighbour from the codebook at that place and decoding produces an image that obeys our test time constraint.

5 Methods and Experiments

5.1 Training Discrete Variational Autoencoders

We begin by training VQVAEs on both simple image datasets (FashionMNIST) and more complicated dataset with diverse classes (CIFAR10, MiniIMageNet). Some of the key parameters we have to tune include the β for the alignment term and the learning rate. Even though the default number of discrete categories to maintain at each index is 512, we experiment with lower number of categories to reduce the search space of our effectively combinatorial optimization problem. We observe that with precise hyperameter tuning it is possible to reduce the search space of our model by large margins. For FashionMNIST, we are able to reduce from $k = 512$ to $k = 40$ with only a 0.002 difference in test reconstruction loss and for CIFAR-10, we are able to do so with a 0.003 difference. The full results of our best training runs are seen in Figure 2, while sample **test** reconstructions are seen in Figure

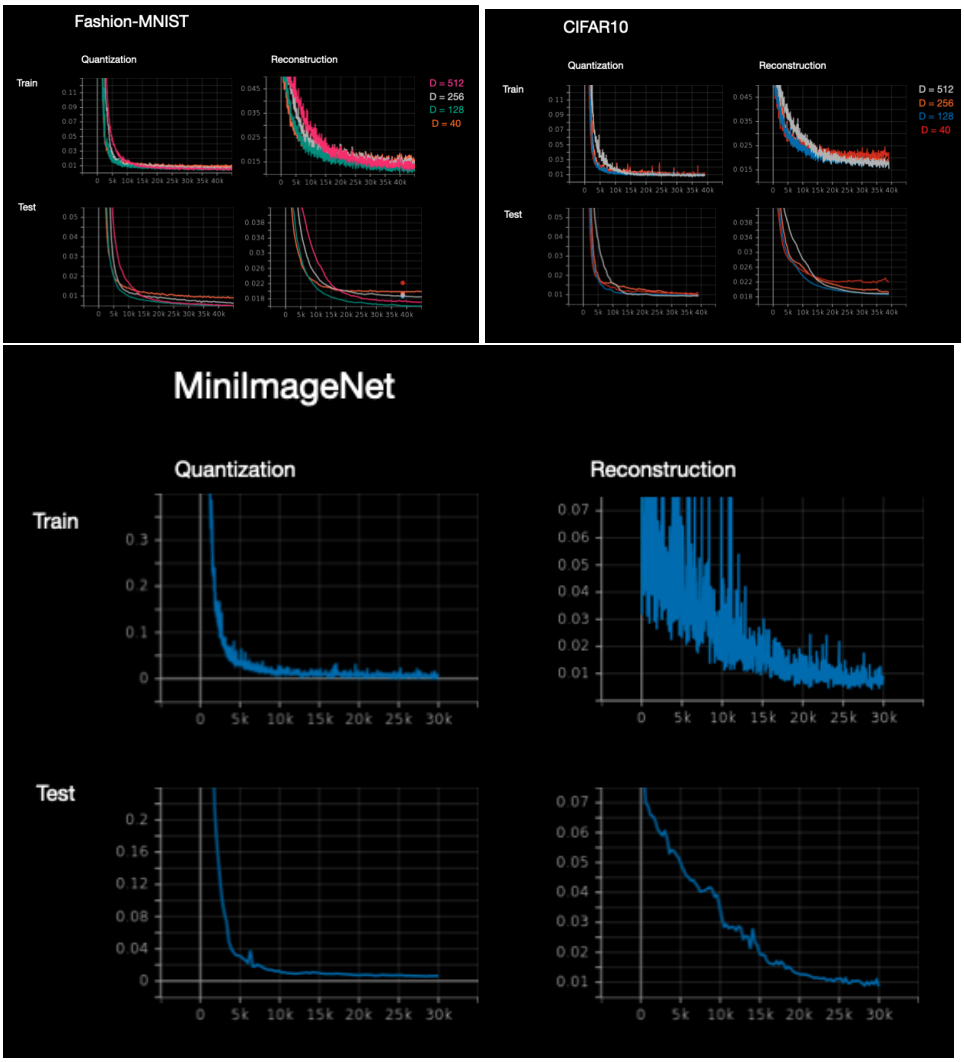

Figure 2: Train and Test Quantization and Reconstruction Losses on (a) Fashion-MNIST (b) CIFAR 10 (c) MiniImageNet. After tuning hyperparameters such as β for the alignment/committment term and the learning rates, we were able to get the model to converge on a wide variety of values for k, the number of codebook vectors and many different datasets.

3. Note that the quality of reconstruction does not severely degrade between $k = 512$ and $k = 40$, atleast for the simple dataset.

5.2 Gradient Based Optimization

Given the trained VQVAEs, the next step is to characterize the behavior of performing latent space search using simple gradient based algorithms. While it is unlikely these will work well, these are critical experiments to explore the latent space organization of VQVAEs, and the challenges with test-time optimization on them.

In Algorithm 1, we show how test-time optimization is typically performed on a generative model that has a continuous latent space. Notice how the stopping criterion is natural and intuitive: you stop exploring once subsequent decreases in the loss get too small. Notice also how gradient taken with respect to z_i is being used to update z_i due to the end-to-end differentiable nature of the continuous space model. Algorithm 1 is similar to, but not PULSE, since Algorithm 1 does plain gradient descent and does not constraint the search to high probability regions using properties of z_i like PULSE does.

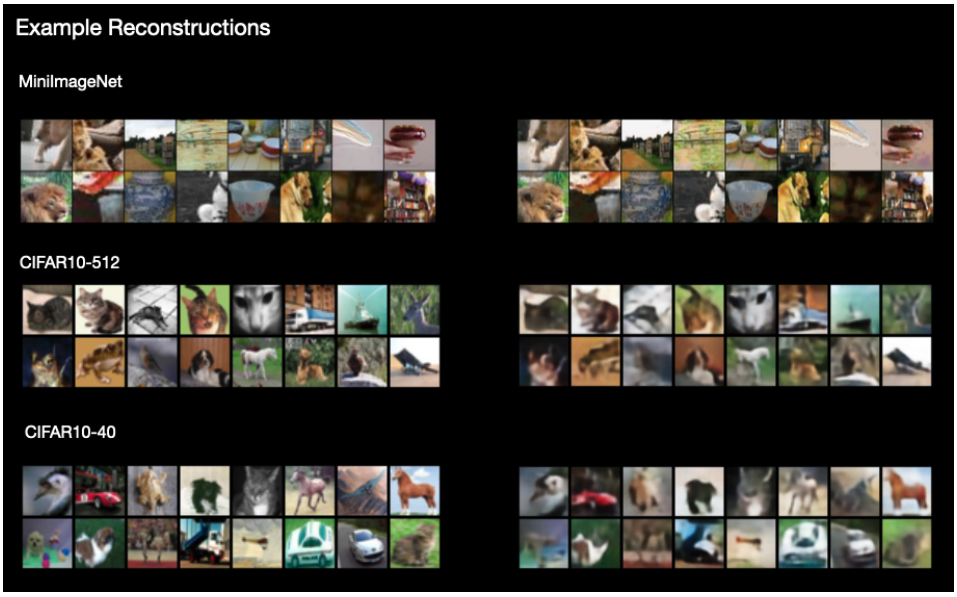

Figure 3: Exemplar Reconstructions

In Algorithm 2, we provide the natural way to "convert" this algorithm so that it can search the latent space of a discrete generative model. There are 2 key differences (1) We use the straight-through trick used at train time once again for test time optimization, where we copy over the gradients from the tensor at the decoder to the latents before the nearest neighbours. We illustrate this 'copy over' trick in Figure 1. and (2) We change the stopping criterion: due to the nearest neighbours operation, a shift in the latents before the nearest neighbours does not actually imply a change in the image generated, as the updated latents might still have the same nearest neighbours in the codebook, and hence produce the same gradient on the next step. This means that consecutive steps can have no difference in the loss despite an update being made! (In this case, we want to keep taking the same step until the difference in latents is significant enough to induce a different in the nearest neighbours selected!). As such, we introduce a hashset where we keep track of every possible combination of latent indices seen so far, and stop if we see any set of latent indices twice. This is because if we have arrived at a previously used latent index, then the gradient will inevitably guide us there again, since you would decode to the same image you had in the past. Note that this situation is next to impossible in the continuous case!

In Algorithm 3, we provide a simple modification on Algorithm 2 which limits the scope of changes in any single gradient step. If the learning rate is too high in Algorithm 2, it may be that a single gradient step places the latent completely out of distribution, and hence hinders subsequent latent space search. To control the size/direction of the step, Algorithm 3 only allows a single spatial position in the latents to change value at any given time step, following Algorithm2 exactly otherwise. The spatial position which has the highest norm gradient is the one chosen to be updated, and the rest have gradient masked out.

A full pseudocode style description of these algorithms is found in the appendix.

We implement Algorithm 2 and Algorithm 3 on $k = 512$ trained versions of the models on MiniImageNet and CIFAR 10. Based on our analysis, we make the following observations.

Consider Figure 4. The two rows show test-time optimization with Algorithm 2 on CIFAR10 and MiniImageNet respectively, with each image corresponding to a consecutive unique reconstruction achieved in the traversal. The far left is the source image from which the latent space is initialized, whereas the far right is the target for reconstruction- our hope is that the method finds a sequence of edits in the latent space such that the target image is produced. However, this is not what is observed. Instead, we have

Dataset	Mean	Std. Deviation	Min.	Max
Mini-ImageNet	2A.47	23.12	2	103
CIFAR-10	5.97	5.00	2	34

Table 1: Statistics on unique traversal steps before first repeat index set occurence

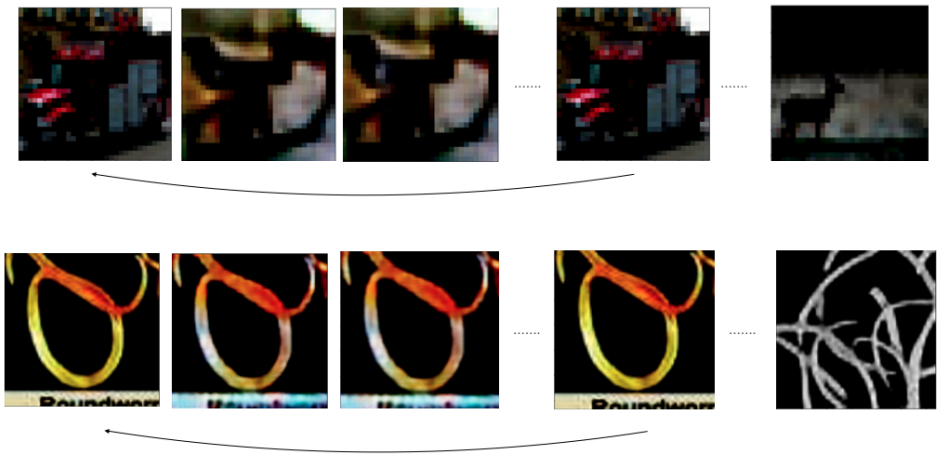

Figure 4: Examples for optimization process. Most of the navigations inevitably run into a repeated set of latents

1. Each incremental change between consecutive images in a row takes several steps at our pre-set $\alpha = 0.1$. This is because we must take multiple steps with the same gradients before the updated weights correspond to a different nearest neighbour. The issue with this is that there is a high variance in the number of steps required to enter a different nearest neighbour, and this makes hyperparameter tuning difficult. Contrast this to continuous representation space, where the gradient is guaranteed to be different each time.

2. A greater cause for concern is the existence of **short navigational loops** inside the test-time optimization. These are indicated in the form of reverse arrows on Figure 4. After a sequence of edits, a latent code that was previously encountered is seen again, which means that following the gradients would keep us in an infinite local loop of perturbations, never making any progress toward the final goal. In Table 1, we provide statistics for the earliest occurence of such loops (in units of number of unique reconnstructions). We run the experiment for 200 unique test source target pairs and measure the earliest occurence of such a loop. We find one in every single instantiation of the experimet and find that they have a very high variance in the range in which they occur.

Algorithm 3 can be considered as a noisy version of Algorithm 2 as it only allows for editing the latent code at one location at a time. While this algorithm experimentally alleviates the issue of finding the **short navigational loops**, it unfortunately amplifies the first of the two problems, with long periods of non-changing gradients and movement in a single direction. If the learning rate is scaled to counter this problem, then this leads to instability in the test-time optimization, and the creation of a highly out of distribution image. The existence of these issues with simple heuristics motivates our next steps to learn a categorical distribution on the latent space of the model.

A positive takeaway from our experiment is that the results of Figure 4 suggest that the local neighbourhood of latents corresponds to a semantically close images. This is encouraging as going forward it allows us to design our task to take advantage of this property.

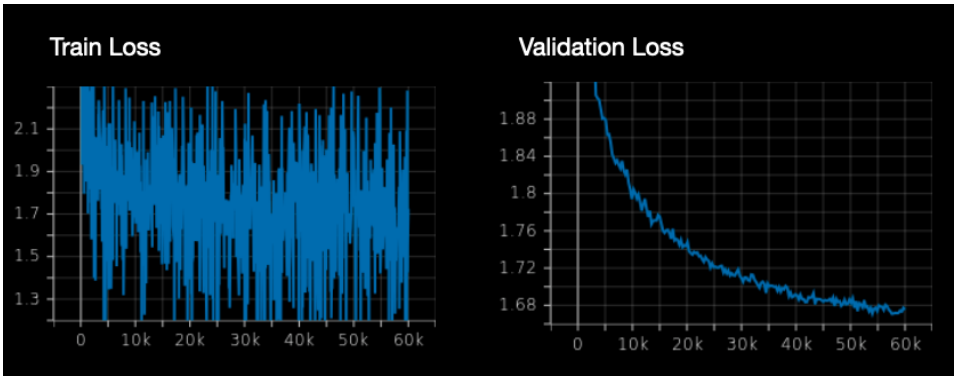

Figure 5: Train and Validation Loss for PixelCNN

5.3 Sampling: Modelling the distribution on the latents categorically

The first step toward resolving the countintertuitive solutions in the search is to constrain the search to high probability regions of the latent space. But unlike a VAE or a GAN, there is no way to measure this (much less the fact that it would have the nice structure of a gaussian) as at train time the prior is set to a uniform distribution between $\frac{-1}{k}$ and $\frac{1}{k}$. Hence to get a distribution over our combinatorial latents, we must first encode all of the images we have and perform NN on them to get a categorical dataset of spatially organized indices which are representative of the data distribution. Then we must learn to model these conditioned on the indices sampled at other spatial positions. One way to interpret this is to think of the post nearest neighbour space as a smaller resolution image with a fewer number of possibilties at each timestep. Hence we can use a similar autoregressive model as that used on images to get a discrete categorical joint representation on this space. We choose to use a PixelCNN following previous work, which models the distribution autoregressively in a raster order scan fashion:

$$p(\mathbf{z}) = p(z_1, z_2, \ldots, z_{n^2}) = \prod_{i=1}^{n^2} p(z_i | z_{<i})$$

The method is then trained step-wise with log-likelihood. We are able to get the method to converge on all of our datasets. The loss curve for the MiniImageNet $d = 512$ model has been shown below, and samples are visualized in the appendix. This gives us a discrete distribution which we can sample from. We propose that if we are able to make a continuous approximation of this joint probability distribution and learn a bijective function from a gaussian to it, we would be able to use the merits of PULSE to solve the problem, as the bijective mapping plus the approximation would be bundled with the decoder and the problem would essetially reduce to PULSE. In the subsequent sections, we study the feasibility of this with some toy experiments.

5.4 Reducing to PULSE with Normalizing Flows

We use the gumbel softmax trick to get a continuous relaxation of a discrete distribution. More details can be found in the appendix. Given that there is an underlying categorical distribution to our code vectors, we seek to approximate it with a continuous distribution so we can run PULSE, which is where Gumbel-Softmax comes into play. We can learn a mass-preserving bijection from our distribution to a Gumbel-Softmax distribution using normalizing flows [18]. Our model architecture consists of sixteen Affine Coupling Blocks, each of which takes in a Multilayer Perceptron p with output dimension D. An Affine Coupling Block which takes in input $p(x)$ produces output y as follows:

$$y_{1:d} = p(x)_{1:d}$$

$$y_{d+1:D} = p(x)_{d+1:D} \odot \exp\left(s(p(x)_{1:d})\right) + t(p(x)_{1:d})$$

Note that $d < D$, and that s and t denote scale and translation.

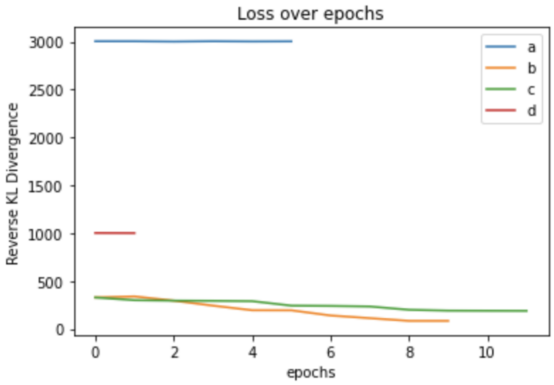

Figure 6: Revrerse KL Divergence vs. Epochs, demonstrating characteristic instability of flow models

We tested four configurations of this experiment. Our loss function was Reverse KL-Divergence [19]. For practical purposes, our source distribution was set to a multivariate Gaussian with a diagonal covariance matrix, and our target was set as a Gumbel-Softmax:

1. The standard Gaussian with $\tau = 0.1$ and Adam as its optimizer.
2. A shifted Gaussian with mean 5, $\tau = 0.1$, and Stochastic Gradient Descent with Nesterov's Acceleration.
3. A shifted Gaussian with mean 5, $\tau = 1.0$, and Stochastic Gradient Descent with Nesterov's Acceleration.
4. The standard Gaussian with $\tau = 0.1$ and Stochastic Gradient Descent with Nesterov's Acceleration.

Unfortunately, our tests did not yield fruitful results. All configurations were highly unstable, with experiments requiring repeated multiple runs with different initializations to produce the following values. Truncations in the plot denote exploding or vanishing gradients.

We believe these results are due to two distinct phenomena:

1. As stated in [18], flow models with Affine Coupling Blocks are highly unstable unless extremely deep networks are used.
2. In flow models, the support of the source distribution must match the source of the target distribution [20]. The support of the Gumbel-Softmax distribution is a simplex; we struggled to derive a natural Gaussian-like distribution over a simplex. However, we are hopeful that learning code vectors over a simplex is more feasible than this task.

6 Limitations, Conclusions and Future Work

Our experimentation with the VQVAEs reveals many ingredients to the ideal recipe of how best we can search them. First, we verify that they can be trained at scaled-down sizes where searching them is feasible without compromising too much on image quality. Then, we propose, validate and critique gradient-based methods for test-time optimization that reveal interesting properties (and problems) with working in latent space. One particular takeaway is that it is *imperative* to ensure that the search remains in high probability regions of the prior distribution on the latents, even in the discrete case. Towards this end, we propose a flow-based method. Here, our results were limited by the instability of the flow method we choose, and the large support on our source distribution. Overall, we identify three important verticals for progress:

Redefinition of task to editing and not search: In section 5.2, we briefly noted that subsequent unique reconstructions had highly fine-grained, incremental changes corresponding to changes in latent space. This might mean that test-time optimization on VQVAEs might lend itself to more fine-grained tasks. Given an initialization in the latent space from a source image, our goal could be

rephrased as searching only in the ϵ ball around the source image, with the target sampled strictly from this region. Experimenting with this might help pave way for more advanced work.

Better Heuristics on Gradient Based Learning: We believe that with the newfound awareness of **short navigational loops** and the problem of a lack of **continuous gradient**, we can design better heuristics for gradient based learning. Dynamically adjusted learning rates, modular masking depending on previously seen combinations to promote exploration in the latent space, and framing search in the latent space as contextual bandits problem are all possible options here.

Rewriting the source distribution class for Flows: As of now, our flow algorithm is unstable because our source distribution does not share the exact support as for our target distribution. While our source distribution has infinite support our target distribution is defined on top of the probability simplex. We are currently writing a source distribution class on top of the Normalizing Flows library we used for a Gaussian defined on top of a probability simplex. We anticipate this would improve results.

7 A Note on Novelty

We wish to note that despite not providing complete results for a working end-to-end method, the novelty in this paper is three-fold: (1) While previous methods have considered inversion into VQVAE space, we believe that we are the first to propose searching the latent space of the VQVAE model with a test time constraint: in particular, we are the first to phrase it in the language of a combinatorial problem, recognize it is equivalent to integer programming and motivate the wide-scale benefits in solving it, (2) We are the first to try and establish results on the simple "straight-through" gradient based baseline and the one-step-extension in context of test time optimization for VQVAEs; in particular, we show through qualitative examples that the failure modes of these methods are well characterized, which motivates the necessity of distribution-grounded search, and (3) we are the first to propose mapping between a simple distribution with high dimensional properties favorable to search (aka Gaussian) to a continuous approximation of a discrete categorical distribution. From our toy experiments, it is clear that (3) is a complicated task, but we believe that with further work on choice of source and target, we can make it work.

8 Contributions

IPC proposed the experiment, proposed and coded the gradient based algorithms, managed the VQ-VAE repository, trained the autoencoders and priors and made the figures. MHH provided reduction from Integer Programming, reviewed heuristics for search and literature review of conditioned generative models. RM and MHH structured, wrote the code for and completed the preliminary normalizing flow experiments. All authors contributed to writing the paper, to discussions and brainstorming.

References

A Appendix

A.1 Relation to original project proposal

We wish to take note of the difference between our initial project proposal and final project report in this section. Our original proposal was for Self-Supervised Generative Machine Translation with Latent Space Search. We planned to replicate the work of [21] using generative models. Our intention was to use a GAN to generate an image from text in language A, perform latent space search to find a similar image that generated a good caption in language B, and then take this as our translation. We believe that this proposal warrants future investigation, but it was not well-suited for our class project due to the computational resources needed to train GANs, the lack of suitable models to compare it to, and the difficulty of setting up a legacy environment for replicating the original experiment.

The direction that we ultimately chose does draw inspiration from the original proposal. We took the problem of latent space search and applied it to another class of models that can be used for the same purpose. By focusing on the latent space search algorithms, we were able to focus specifically on

the most challenging part of the original problem. Finally, these discrete generative models can be used for text-to-image synthesis, so it is not hard to imagine a self-supervised generative machine translation model built on top of them.

A.2 Proof: Discrete Latent Space Search is NP-Hard

Proof. Finding the optimal loss $\mathcal{L}$ as a function of the code vectors is NP-Hard.

We can rewrite the optimization problem as an Integer Program, which is known to be a NP-Hard problem [22]. The latent space of a VQ-VAE can be viewed as a discrete $n \times n$ codebook $X \in [K]^{n \times n}$ with the value of $X_{i,j}$ being a fixed and finite index from $[K] = [1, \ldots, K] \subseteq \mathbb{Z}$. From the codebook, we have a mapping $f : [K] \to \mathbb{R}^d$ from indices to d-dimensional vectors. Our loss function $\mathcal{L}$ is a function of the code vectors themselves, and we define $\mathcal{L} : [f([K])] \to \mathbb{R}$ such that it is differentiable over $\mathbb{R}^d$. Thus, for known $\mathcal{L}$ (which is given as INSERT) and f (which is learned as a Deconvolutional Neural Network), we can pose the optimization problem as a constrained integer program:

$$\inf_X \mathcal{L}(f(X)) \text{ such that } \forall i, j,\ X_{i,j} \in \mathbb{Z}, X_{i,j} - K \leq 0, 1 - X_{i,j} \leq 0$$

This problem is an Integer Program with linear constraints. If $\mathcal{L}(f(X))$ is restricted to the class of linear functions (it almost certainly is not), the reduction is trivial. If $\mathcal{L}(f(X))$ is not restricted to the class of linear functions, it is still valid for it to have a linear component, so $\mathcal{L}(f(X))$ is at least as expressive as any linear function, and thus, solving $\inf_X \mathcal{L}(f(X))$ will be at least as challenging of a problem as the linear case. So, being able to minimize this specific loss function implies a solution to general constrained Integer Linear Programming, and it is thus NP-Hard. □

A.3 On the Limitations of Traditional Heuristics for Integer Programming

While traditional search heuristics, notably branch-and-bound, have seen success with Integer Programming [23], we have several reasons to believe they would not be suitable for searching a Discrete Latent Space. First, many of these heuristics have been designed for linear objectives, whereas our objective $\mathcal{L}(f(X))$ is nonlinear. These heuristics also assume an order to the feasible set, whereas we saw no natural way to impose an ordering on the indices of the learned code vectors. Finally, interpolation between fixed indices in the codebook becomes a difficult challenge with no natural order in place; such interpolation would be necessary if we were to take $\mathbb{R}$ instead of $[K]$ as the domain of f.

A.4 Precise descriptions of Algorithms

Algorithm 1 Generic Test Time Optimization on Continuous Latent Space Generative Model

Require: $\mathcal{L}, t \in f(\mathcal{X}), g \in C^\infty : \mathbb{R}^d \mapsto \mathcal{X}, \lambda, \alpha$
  $\mathcal{L}_0 =$ **INTMAX**
  $z_1 =$ **init**()
  $x_1 = g(z_1)$
  $\mathcal{L}_1 = \mathcal{L}(x_1, t)$
  $i = 1$
  while $|\mathcal{L}_i - \mathcal{L}_{i-1}| > \lambda$ **do**
    $z_{i+1} = z_i - \alpha \cdot \nabla_{z_i} \mathcal{L}_i$
    $x_{i+1} = g(z_{i+1})$
    $\mathcal{L}_{i+1} = \mathcal{L}(x_{i+1}, t)$
    $i = i + 1$
  end while

A.5 Continuous Approximations of Discrete Distributions

We want to approximate the underlying discrete distribution of the code vectors with a continuous distribution for two main purposes: to allow for differentiability and to learn a flow-based model

Algorithm 2 Generic Test Time Optimization on Discrete Latent Space Generative Model

Require: $\mathcal{L}, t \in f(\mathcal{X}), g \in C^{\infty} : \mathbb{R}^d \mapsto \mathcal{X}, c \in \mathbb{R}^{k \times d}, \lambda, \alpha$
  $z_{hash} =$ **set()**
  $e_1 =$ **init**()
  $z_1 =$ **NN**(e_1, c) ▷ This step is non-differentiable
  $x_1 = g(z_1)$
  $i = 1$
  while $z_i \notin z_{hash}$ **do**
    z_{hash}.**add**(z_i)
    $\mathcal{L}_i = \mathcal{L}(x_i, t)$
    $\mathcal{L}_{i+1} = \mathcal{L}_i$
    count = 0
    while $\mathcal{L}_{i+1} = \mathcal{L}_i$ **do**
      if count $> \lambda$ **then**
        break
      end if
      $e_{i+1} = e_i - \alpha \cdot \nabla_{z_i} \mathcal{L}_i$
      $z_{i+1} =$ **NN**(e_{i+1}, c) ▷ This step is non-differentiable
      $x_{i+1} = g(z_{i+1})$
      $\mathcal{L}_{i+1} = \mathcal{L}(x_{i+1}, t)$
      count = count + 1
    end while
    $i = i + 1$
  end while

Algorithm 3 Single Step Test Time Optimization on Discrete Latent Space Generative Model

Require: $\mathcal{L}, t \in f(\mathcal{X}), g \in C^{\infty} : \mathbb{R}^d \mapsto \mathcal{X}, c \in \mathbb{R}^{k \times d}, \lambda, \alpha$
  $z_{hash} =$ **set()**
  $e_1 =$ **init**()
  $z_1 =$ **NN**(e_1, c) ▷ This step is non-differentiable
  $x_1 = g(z_1)$
  $i = 1$
  while $z_i \notin z_{hash}$ **do**
    z_{hash}.**add**(z_i)
    $\mathcal{L}_i = \mathcal{L}(x_i, t)$
    $\mathcal{L}_{i+1} = \mathcal{L}_i$
    count = 0
    while $\mathcal{L}_{i+1} = \mathcal{L}_i$ **do**
      if count $> \lambda$ **then**
        break
      end if
      grad $= \nabla_{z_i} \mathcal{L}_i$
      $m_x, m_y = \text{argmax} \, ||\text{grad}||_2$
      mask $=$ **zeroslike**(grad)
      mask$[:, :, m_x, m_y] = 1.0$
      grad $=$ grad $*$ mask
      $e_{i+1} = e_i - \alpha \cdot$ grad
      $z_{i+1} =$ **NN**(e_{i+1}, c) ▷ This step is non-differentiable
      $x_{i+1} = g(z_{i+1})$
      $\mathcal{L}_{i+1} = \mathcal{L}(x_{i+1}, t)$
      count = count + 1
    end while
    $i = i + 1$
  end while

Figure 7: Original Problem: Machine Translation with Visual Constraints using latent space search

to it. Since the underlying distribution of the code vectors is a categorical distribution Z with the probability of the kth vector denoted π_k, we typically sample z from Z as:

$$z = \text{onehot}\left(\max\{i \mid \pi_1 + \cdots + \pi_{i-1} \leq U\}\right)$$

Where $U \sim \text{Uniform}[0, 1]$. However, using the Gumbel-Max trick [24], we can sample z as:

$$z = \text{onehot}\left(\underset{i}{\text{argmax}}\left(g_i + \log(\pi_i)\right)\right)$$

Where $g_i \sim \text{Gumbel}(0, 1)$. Then, we can approximate argmax with softmax, which is differentiable. The resultant samples are given by:

$$z_i = \frac{\exp\left(\left(\log(\pi_i) + g_i\right)/\tau\right)}{\sum_{j=1}^{k} \exp\left(\left(\log(\pi_j) + g_j\right)/\tau\right)}$$

Where τ denotes the softmax temperature. The probability density function of the Gumbel-Softmax distribution is given as:

$$p_{\pi,\tau}(z_1, \ldots, z_k) = \Gamma(k)\tau^{k-1}\left(\sum_{i=1}^{k} \pi_i / z_i^T\right)^{-k} \prod_{i=1}^{k}\left(\pi_i / z_i^{\tau+1}\right)$$

Note that as $\tau \to 0$, the Gumbel-Softmax distribution approaches the original categorical distribution [24].

A.6 Visualization of Novel Samples from MiniImageNet PixelCNN prior

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[2] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

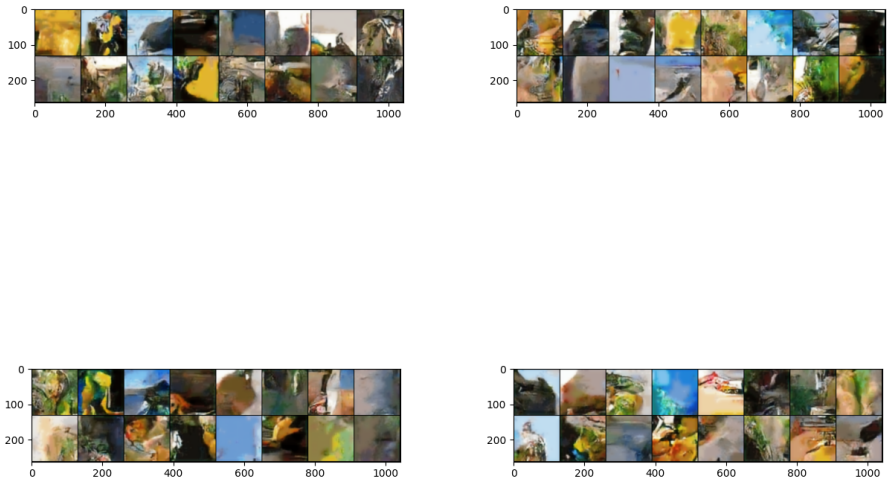

Figure 8: Sampled images from trained PixelCNN prior. These show a lot better structural understanding of object and color as compared to random sampling, which will simply produce fuzzy noise

[3] Otto Fabius and Joost R. van Amersfoort. Variational recurrent auto-encoders, 2014.

[4] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.

[7] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, Hossein Sadeghi, and Mohammad H Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1):014001, sep 2018.

[8] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jun Wang. Causalvae: Structured causal disentanglement in variational autoencoder. *CoRR*, abs/2004.08697, 2020.

[9] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? *CoRR*, abs/1904.03189, 2019.

[10] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[11] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021.

[13] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017.

[14] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis, 2016.

[15] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[16] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2437–2445, 2020.

[17] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.

[18] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

[19] Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness, 2019.

[20] Gonzalo Mena. The gumbel-softmax trick for inference of discrete variables. `https://casmls.github.io/general/2017/02/01/GumbelSoftmax.html`, 2017.

[21] Dídac Surís, Dave Epstein, and Carl Vondrick. Globetrotter: Unsupervised multilingual translation from visual alignment. *arXiv preprint arXiv:2012.04631*, 2020.

[22] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, oct 1981.

[23] E.M.L. Beale. Branch and bound methods for mathematical programming systems. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 201–219. Elsevier, 1979.

[24] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016.