PATCHVAE: LEARNING LOCAL LATENT CODES FOR RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised representation learning holds the promise of exploiting large amount of available unlabeled data to learn general representations. A promising technique for unsupervised learning is the framework of Variational Auto-encoders (VAEs). However, unsupervised representations learned by VAEs are significantly outperformed by those learned by supervising for recognition. Our hypothesis is that to learn useful representations for recognition the model needs to be encouraged to learn about repeating and consistent patterns in data. Drawing inspiration from the mid-level representation discovery work, we propose PatchVAE, that reasons about images at patch level. Our key contribution is a bottleneck formulation in a VAE framework that encourages mid-level style representations. Our experiments demonstrate that representations learned by our method perform much better on the recognition tasks compared to those learned by vanilla VAEs.

1 INTRODUCTION

Due to the availability of large labeled visual datasets, supervised learning has become the dominant paradigm for visual recognition. That is, to learn about any new concept, the modus operandi is to collect thousands of labeled examples for that concept and train a powerful classifier, such as a deep neural network. This is necessary because the current generation of models based on deep neural networks require large amounts of labeled data (Sun et al., 2017). This is in stark contrast to the insights that we have from developmental psychology on how infants develop perception and cognition without any explicit supervision (Smith & Gasser, 2005). Moreover, the supervised learning paradigm is ill-suited for applications, such as health care and robotics, where annotated data is hard to obtain either due to privacy concerns or high cost of expert human annotators. In such cases, learning from very few labeled images or discovering underlying natural patterns in large amounts of unlabeled data can have a large number of potential applications. Discovering such patterns from unlabeled data is the standard setup of unsupervised learning.

Over the past few years, the field of unsupervised learning in computer vision has followed two seemingly different tracks with different goals: generative modeling and self-supervised learning. The goal of generative modeling is to learn the probability distribution from which data was generated, given some training data. A learned model can draw samples from the same distribution or evaluate the likelihoods of new data. Generative models are also useful for learning compact representation of images. However, we argue that these representations are not as useful for visual recognition. This is not surprising since the task of reconstructing images does not require the bottleneck representation to sort out meaningful data useful for recognition and discard the rest; on the contrary, it encourages preserving as much information as possible for reconstruction. In comparison, the goal in self-supervised learning is to learn representations that are useful for recognition. The standard paradigm is to establish proxy tasks that don't require human-supervision but can provide signals useful for recognition. Due to the mismatch in goals of unsupervised learning for visual recognition and the representations learned from generative modeling, self-supervised learning is a more popular way of learning representations from unlabeled data. However, fundamental limitation of this self-supervised paradigm is that we need to define a proxy-task that can mimic the desired recognition. It is not always possible to establish such a task, nor are these tasks generalizable across recognition tasks.

In this paper, we take the first steps towards enabling the unsupervised generative modeling approach of VAEs to learn representations useful for recognition. Our key hypothesis is that for a representation to be useful, it should capture just the *interesting* parts of the images, as opposed to *everything* in the images. What constitutes an interesting image part has been defined and studied in earlier works that pre-date the end-to-end trained deep network methods (Singh et al., 2012; Doersch et al., 2012; Juneja et al., 2013). Taking inspiration from these works, we propose a novel representation that only encodes such few parts of an image that are repetitive across the dataset, i.e., the patches that occur often in images. By avoiding reconstruction of the entire image our method can focus on regions that are repeating and consistent across many images. In an encoder-decoder based generative model, we constrain the encoder architecture to learn such repetitive parts – both in terms of representations for appearance of these parts (or patches in an image) and where these parts occur. We formulate this using variational auto-encoder (β-VAEs) (Kingma & Welling, 2013; Matthey et al., 2017), where we impose novel structure on the latent representations. We use discrete latents to model part presence or absence and continuous latents to model their appearance. We present this approach, PatchVAE, in Section 3 and demonstrate that it learns representations that are much better for recognition as compared to those learned by the standard β-VAEs (Kingma & Welling, 2013; Matthey et al., 2017).

In addition, we propose in Section 3.4 that losses that favor foreground, which is more likely to contain repetitive patterns, result in representations that are much better at recognition. In Section 4, we present results on CIFAR100 (Krizhevsky et al., 2009), MIT Indoor Scene Recognition (Quattoni & Torralba, 2009), Places (Zhou et al., 2017), and ImageNet (Deng et al., 2009) datasets. Our contributions are as follows:

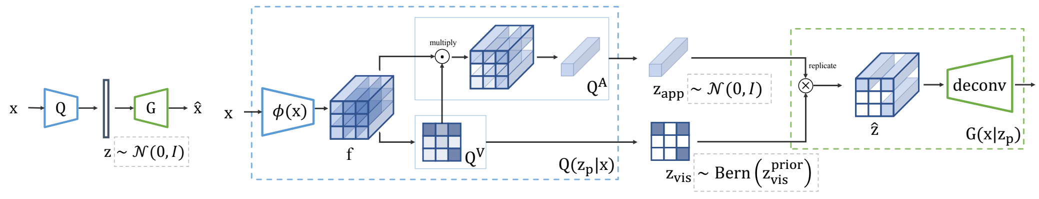
- We propose a novel patch-based bottleneck in the VAE framework that learns representations that can encode repetitive parts across images.
- We demonstrate that our method, **PatchVAE**, learns unsupervised representations that are better suited for recognition in comparison to traditional VAEs.
- We show that losses that favor foreground are better for unsupervised learning of representations for recognition.
- We perform extensive ablation analysis to understand the importance of different aspects of the proposed PatchVAE architecture.

2 Related Work

Due to its potential impact, unsupervised learning (particularly for deep networks) is one of the most researched topics in visual recognition over the past few years. Generative models such as VAEs (Kingma & Welling, 2013; Matthey et al., 2017; Kingma et al., 2016; Gregor et al., 2015), PixelRNN (van den Oord et al., 2016), PixelCNN (Gulrajani et al., 2016; Salimans et al., 2017), and their variants have proven effective when it comes to learning compressed representation of images while being able to faithfully reconstruct them as well as draw samples from the data distribution. GANs (Goodfellow et al., 2014; Radford et al., 2015; Zhu et al., 2017; Arjovsky et al., 2017) on the other hand, while don't model the probability density explicitly, can still produce high quality image samples from noise. There has been work combining VAEs and GANs to be able to simultaneously learn image data distribution while being able to generate high quality samples from it (Khan et al., 2018; Donahue et al., 2016; Larsen et al., 2015). Convolution sparse coding (Affara et al., 2018) is an alternative approach for reconstruction or image in-painting problems. Our work complements existing generative frameworks in that we provide a structured approach for VAEs that can learn beyond low-level representations. We show the effectiveness of the representations learned by our model by using them for standard visual recognition tasks.

There has been a lot of work in interpreting or disentangling representations learned using generative models such as VAEs (Matthey et al., 2017; Fraccaro et al., 2017; Kim & Mnih, 2018). However, there is little evidence of effectiveness of disentangled representations in visual recognition tasks. In our work, we focus on incorporating inductive biases in these generative models (e.g., VAEs) such that they can learn representations better suited for visual recognition tasks.

A related, but orthogonal, line of work is self-supervised learning where a proxy task is designed to learn representation useful for recognition. These proxy tasks vary from simple tasks like arranging patches in an image in the correct spatial order (Doersch et al., 2014; 2015) and arranging frames from a video in correct temporal order (Wang & Gupta, 2015; Pathak et al., 2017), to more involved tasks like in-painting (Pathak et al., 2016) and context prediction (Noroozi & Favaro, 2016; Wang et al., 2017). We follow the best practices from this line of work for evaluating the learned representations.



(a) VAE Architecture (b) PatchVAE Architecture

Figure 1: (a) **VAE Architecture**. (b) **Proposed PatchVAE Architecture**: Our encoder network computes a set of feature maps $\mathbf{f}$ using $\phi(\mathbf{x})$. This is followed by 2 independent single layer networks - bottom network generates part visibility parameters Q^{V}. We combine Q^{V} with output of top network to generate part appearance parameters Q^{A}. We sample z_{vis} and z_{app} to construct $\hat{\mathbf{z}}$ as described in Section 3.2 which is input to the decoder network. We also visualize the corresponding priors for latents $\mathbf{z}_{\text{app}}$ and $\mathbf{z}_{\text{vis}}$ in the dashed gray boxes.

3 Our Approach

Our work builds upon VAE framework proposed by Kingma & Welling (2013). We briefly review relevant aspects of the VAE framework and then present our approach.

3.1 VAE Review

Standard VAE framework assumes a generative model for data where first a latent $\mathbf{z}$ is sampled from a prior $p(\mathbf{z})$ and then the data is generated from a conditional distribution $G(\mathbf{x}|\mathbf{z})$. A variational approximation $Q(\mathbf{z}|\mathbf{x})$ to the true intractable posterior is introduced and the model is learned by minimizing the following negative variational lower bound (ELBO).

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = -\,\mathbb{E}_{z\sim Q(\mathbf{z}|\mathbf{x})}\left[\log G(\mathbf{x}|\mathbf{z})\right] + D_{\text{KL}}\left[Q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right] \tag{1}$$

$Q(\mathbf{z}|\mathbf{x})$ is often referred to as an encoder as it can be viewed as mapping data to the the latent space, while $G(\mathbf{x}|\mathbf{z})$ is referred to as a decoder (or generator) that can be viewed as mapping latents to the data space. Both Q and G are commonly paramterized as neural networks. Fig. 1a shows the commonly used VAE architecture. If the conditional $G(\mathbf{x}|\mathbf{z})$ takes a gaussian form, negative log likelihood in the first term of RHS of Eq. 1 becomes mean squared error between generator output $\hat{\mathbf{x}} = G(\mathbf{x}|\mathbf{z})$ and input data $\mathbf{x}$. In the second term, prior $p(\mathbf{z})$ is assumed to be a multi-variate normal distribution with zero-mean and diagonal covariance $\mathcal{N}(0,\, \mathcal{I})$ and the loss simplifies to

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + D_{\text{KL}}\left[Q(\mathbf{z}|\mathbf{x}) \parallel \mathcal{N}(0,\, \mathcal{I})\right] \tag{2}$$

When G and Q are differentiable, entire model can be trained with SGD using reparameterization trick (Kingma & Welling, 2013). Matthey et al. (2017) propose an extension for learning disentangled representation by incorporating a weight factor β for the KL Divergence term yielding

$$\mathcal{L}_{\beta\text{VAE}}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 + \beta D_{\text{KL}}\left[Q(\mathbf{z}|\mathbf{x}) \parallel \mathcal{N}(0,\, \mathcal{I})\right] \tag{3}$$

VAE framework aims to learn a generative model for the images where the latents $\mathbf{z}$ represent the corresponding low dimensional generating factors. The latents $\mathbf{z}$ can therefore be treated as image representations that capture the necessary details about images. However, we postulate that representations produced by the standard VAE framework are not ideal for recognition as they are learned to capture *all* details, rather than capturing 'interesting' aspects of the data and dropping the rest. This is not surprising since there formulation does not encourage learning semantic information. For learning semantic representations, in the absence of any relevant supervision (as is available in self-supervised approaches), inductive biases have to be introduced. Therefore, taking inspiration from works on unsupervised mid-level pattern discovery (Singh et al., 2012; Doersch et al., 2012; Juneja et al., 2013), we propose a formulation that encourages the encoder to only encode such few parts of an image that are repetitive across the dataset, i.e., the patches that occur often in images.

Since the VAE framework provides a principled way of learning a mapping from image to latent space, we consider it ideal for our proposed extension. We chose β-VAEs for their simplicity and widespread use. In Section 3.2, we describe our approach in detail and in Section 3.4 propose a modification in the reconstruction error computation to bias the error term towards foreground high-energy regions (similar to the biased initial sampling of patterns in (Singh et al., 2012)).

3.2 PatchVAE

Given an image $\mathbf{x}$, let $\mathbf{f} = \phi(\mathbf{x})$ be a deterministic mapping that produces a 3D representation $\mathbf{f}$ of size $h \times w \times d_e$, with a total of $L = h \times w$ locations (grid-cells). We aim to encourage the encoder network to only encode parts of an image that correspond to highly repetitive patches. For example, a random patch of noise is unlikely to occur frequently, whereas patterns like faces, wheels, windows, etc. repeat across multiple images. In order capture this intuition, we force the representation $\mathbf{f}$ to be useful for predicting frequently occurring parts in an image, and use *just* these predicted parts to reconstruct the image. We achieve this by transforming $\mathbf{f}$ to $\hat{\mathbf{z}}$ which encodes a set of parts at a small subset of L locations on the grid cells. We refer to $\hat{\mathbf{z}}$ as "patch latent codes" for an image. Next we describe how we re-tool the β-VAE framework to learn these local latent codes. We first describe our setup for a single part and follow it up with a generalization to multiple parts (Section 3.3).

Image Encoding. Given the image representation $\mathbf{f} = \phi(x)$, we would like to learn part representations at each grid location l (where $l \in \{1, \ldots, L\}$). A part is parameterized by its appearance $\mathbf{z}_{\text{app}}$ and its visibility $\mathbf{z}^l_{\text{vis}}$ (i.e., presence or absence of the part at grid location l). We use two networks, $Q^{\text{A}}_{\mathbf{f}}$ and $Q^{\text{V}}_{\mathbf{f}}$, to parameterize posterior distributions $Q^{\text{A}}_{\mathbf{f}}(\mathbf{z}_{\text{app}} \mid \mathbf{f})$ and $Q^{\text{V}}_{\mathbf{f}}(\mathbf{z}^l_{\text{vis}} \mid \mathbf{f})$ of the part parameters $\mathbf{z}_{\text{app}}$ and $\mathbf{z}^l_{\text{vis}}$ respectively. Since the mapping $\mathbf{f} = \phi(\mathbf{x})$ is deterministic, we can re-write these distributions as $Q^{\text{A}}_{\mathbf{f}}(\mathbf{z}_{\text{app}} \mid \phi(\mathbf{x}))$ and $Q^{\text{V}}_{\mathbf{f}}(\mathbf{z}^l_{\text{vis}} \mid \phi(\mathbf{x}))$; or simply $Q^{\text{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x})$ and $Q^{\text{V}}(\mathbf{z}^l_{\text{vis}} \mid \mathbf{x})$. Therefore, given an image $\mathbf{x}$ the encoder networks estimate the posterior $Q^{\text{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x})$ and $Q^{\text{V}}(\mathbf{z}^l_{\text{vis}} \mid \mathbf{x})$. Note that $\mathbf{f}$ is a deterministic feature map, whereas $\mathbf{z}_{\text{app}}$ and $\mathbf{z}^l_{\text{vis}}$ are stochastic.

Image Decoding. We utilize a generator or decoder network G, that given $\mathbf{z}_{\text{vis}}$ and $\mathbf{z}_{\text{app}}$, reconstructs the image. First, we sample a part appearance $\hat{\mathbf{z}}_{\text{app}}$ (d_p dimensional, continuous) and then sample part visibilities $\hat{\mathbf{z}}^l_{\text{vis}}$ (L dimensional, binary) one for each location l from the posteriors

$$\begin{aligned} \hat{\mathbf{z}}_{\text{app}} &\sim Q^{\text{A}}(\mathbf{z}_{\text{app}} \,|\, \mathbf{x}) \\ \hat{\mathbf{z}}^l_{\text{vis}} &\sim Q^{\text{V}}\left(\mathbf{z}^l_{\text{vis}} \,|\, \mathbf{x}\right), \quad \text{where } l \in \{1, \ldots, L\} \end{aligned} \tag{4}$$

Next, we construct a 3D representation $\hat{\mathbf{z}}$ by placing $\hat{\mathbf{z}}_{\text{app}}$ at every location l where the part is present (i.e., $\hat{\mathbf{z}}^l_{\text{vis}} = 1$). This can be implemented by a broadcasted product of $\hat{\mathbf{z}}_{\text{app}}$ and $\hat{\mathbf{z}}^l_{\text{vis}}$. We refer to $\hat{\mathbf{z}}$ as **patch latent code**. Again note that $\mathbf{f}$ is deterministic and $\hat{\mathbf{z}}$ is stochastic. Finally, a deconvolutional network takes $\hat{\mathbf{z}}$ as input and generates an image $\hat{\mathbf{x}}$. This image generation process can be written as

$$\hat{\mathbf{x}} \sim G\left(\mathbf{x} \mid \mathbf{z}^1_{\text{vis}}, \mathbf{z}^2_{\text{vis}}, \ldots, \mathbf{z}^L_{\text{vis}}, \mathbf{z}_{\text{app}}\right) \tag{5}$$

Since all latent variables ($\mathbf{z}^l_{\text{vis}}$ for all l and $\mathbf{z}_{\text{app}}$) are independent of each other, they can be stacked as

$$\mathbf{z}_{\text{p}} = \left[\mathbf{z}^1_{\text{vis}}; \mathbf{z}^2_{\text{vis}}; \ldots; \mathbf{z}^L_{\text{vis}}; \mathbf{z}_{\text{app}}\right]. \tag{6}$$

This enables us to use a simplified the notation (refer to (4) and (5)):

$$\begin{aligned} \hat{\mathbf{z}}_{\text{p}} &\sim Q^{\{\text{A,V}\}}(\mathbf{z}_{\text{p}} \,|\, \mathbf{x}) \\ \hat{\mathbf{x}} &\sim G\left(\mathbf{x} \mid \mathbf{z}_{\text{p}}\right) \end{aligned} \tag{7}$$

Note that despite the additional structure, our model still resembles the setup of variational auto-encoders. The primary difference arises from: (1) use of discrete latents for part visibility, (2) patch-based bottleneck imposing additional structure on latents, and (4) feature assembly for generator.

Training. We use the training setup of β-VAE and use the maximization of variational lower bound to train the encoder and decoder jointly (described in Section 3.1). The posterior Q^{A}, which captures the appearance of a part, is assumed to be a zero-mean Normal distribution with diagonal covariance $\mathcal{N}(0, \mathcal{I})$. The posterior Q^{V}, which captures the presence or absence a part, is assumed to be a Bernoulli distribution $\text{Bern}\left(\mathbf{z}^{\text{prior}}_{\text{vis}}\right)$ with prior $\mathbf{z}^{\text{prior}}_{\text{vis}}$. Therefore, the ELBO for our approach can written as (refer to (3)):

$$\mathcal{L}_{\text{PatchVAE}}(\mathbf{x}) = -\,\mathbb{E}_{\mathbf{z}_{\text{p}} \sim Q^{\{\text{A,V}\}}(\mathbf{z}_{\text{p}} \,|\, \mathbf{x})}\left[G\left(\mathbf{x} \mid \mathbf{z}_{\text{p}}\right)\right] + \beta D_{\text{KL}}\left[Q^{\{\text{A,V}\}}(\mathbf{z}_{\text{p}} \,|\, \mathbf{x}) \,\|\, p(\mathbf{z}_{\text{p}})\right] \tag{8}$$

where, the D_{KL} term can be expanded as:

$$D_{\mathrm{KL}}\left[Q^{\{\mathrm{A,V}\}}(\mathbf{z}_{\mathrm{p}} \mid \mathbf{x}) \parallel p(\mathbf{z}_{\mathrm{p}})\right] = \beta_{\text{app}} \sum_{l=1}^{L} D_{\mathrm{KL}}\left(Q^{\mathrm{V}}(\mathbf{z}^{l}_{\text{vis}} \mid \mathbf{x}) \parallel \mathrm{Bern}\left(\mathbf{z}^{\text{prior}}_{\text{vis}}\right)\right) + \beta_{\text{vis}} D_{\mathrm{KL}}\left(Q^{\mathrm{A}}(\mathbf{z}_{\text{app}} \mid \mathbf{x}) \parallel \mathcal{N}(0, \mathcal{I})\right) \tag{9}$$

Implementation details. As discussed in Section 3.1, the first and second terms of the RHS of (8) can be trained using L2 reconstruction loss and reparameterization trick (Kingma & Welling, 2013). In addition, we also need to compute KL Divergence loss for part visibility. Learning discrete probability distribution is a challenging task since there is no gradient defined to backpropagate reconstruction loss through the stochastic layer at decoder even when using the reparameterization trick. Therefore, we use the relaxed-bernoulli approximation (Maddison et al., 2016; Agustsson et al., 2017) for training part visibility distributions $\mathbf{z}^{l}_{\text{vis}}$.

For an $H \times W$ image, network $Q(\mathbf{f} \mid \mathbf{x})$ first generates feature maps of size $(h \times w \times d_e)$, where $(h,\, w)$ are spatial dimensions and d_e is the number of channels. Therefore, the number of locations $L = h \times w$. Encoders $Q^{\mathrm{A}}_{\mathbf{f}}(\mathbf{z}_{\text{app}} \mid \mathbf{f})$ and $Q^{\mathrm{V}}_{\mathbf{f}}(\mathbf{z}^{l}_{\text{vis}} \mid \mathbf{f})$ are single layer neural networks to compute $\mathbf{z}_{\text{app}}$ and $\mathbf{z}^{l}_{\text{vis}}$. $\mathbf{z}^{l}_{\text{vis}}$ is $(h \times w \times 1)$-dimensional multivariate bernoulli parameter and $\mathbf{z}_{\text{app}}$ is $(1 \times 1 \times d_p)$-dimensional multivariate gaussian. d_p is length of the latent vector for a single part. Input to the decoder $\hat{\mathbf{z}}$ is $(h \times w \times d_p)$-dimensional. In all our experiments, we fix $h = \frac{H}{8}$ and $w = \frac{W}{8}$

Constructing $\mathbf{z}_{\text{app}}$. Notice that $\mathbf{f}$ is an $(h \times w \times d_e)$-dimensional feature map and $\mathbf{z}^{l}_{\text{vis}}$ is $(h \times w \times 1)$-dimensional binary output, but $\mathbf{z}_{\text{app}}$ is $(1 \times 1 \times d_p)$-dimensional feature vector. If $\sum_l \mathbf{z}^{l}_{\text{vis}} > 1$, the part occurs at multiple locations in an image. Since all these locations correspond to same part, their appearance should be the same. To incorporate this, we take the weighted average of the part appearance feature at each location, weighted by the probability that the part is present. Since we use the probability values for averaging the result is deterministic. This operation is encapsulated by the Q^{A} encoder (refer to Figure 1b). During image generation, we sample $\hat{\mathbf{z}}_{\text{app}}$ once and replicate it at each location where $\hat{\mathbf{z}}^{l}_{\text{vis}} = 1$. During training, this forces the model to: (1) only predict $\hat{\mathbf{z}}^{l}_{\text{vis}} = 1$ where similar looking parts occur, and (2) learn a common representation for the part that occurs at these locations. Note that $\mathbf{z}_{\text{app}}$ can be modeled as a mixture of distributions (e.g., mixture of gaussians) to capture complicated appearances. However, in this work we assume that the convolutional neural network based encoders are powerful enough to map variable appearance of semantic concepts to similar feature representations. Therefore, we restrict ourselves to a single gaussian distribution.

3.3 PatchVAE with multiple parts

Next we extend the framework described above to use multiple parts. To use N parts, we use $N \times 2$ encoder networks $Q^{A(i)}\left(\mathbf{z}^{(i)}_{\text{app}} \mid \mathbf{x}\right)$ and $Q^{V(i)}\left(\mathbf{z}^{l(i)}_{\text{vis}} \mid \mathbf{x}\right)$, where $\mathbf{z}^{(i)}_{\text{app}}$ and $\mathbf{z}^{l(i)}_{\text{vis}}$ parameterize the i^{th} part. Again, this can be implemented efficiently as 2 networks by concatenating the outputs together. The image generator samples $\hat{\mathbf{z}}^{(i)}_{\text{app}}$ and $\hat{\mathbf{z}}^{l(i)}_{\text{vis}}$ from the outputs of these encoder networks and constructs $\hat{\mathbf{z}}^{(i)}$. We obtain the final **patch latent code** $\hat{\mathbf{z}}$ by concatenating all $\hat{\mathbf{z}}^{(i)}$ in channel dimension. Therefore, $\hat{\mathbf{z}}^{(i)}$ is $(h \times w \times d_p)$-dimensional and $\hat{\mathbf{z}}$ is $(h \times w \times (N \times d_p))$-dimensional stochastic feature map. For this multiple part case, (6) can be written as:

$$\mathbf{z_P} = \left[\mathbf{z}^{(1)}_{\mathrm{p}};\, \mathbf{z}^{(1)}_{\mathrm{p}};\, \ldots;\, \mathbf{z}^{(N)}_{\mathrm{p}}\right], \text{where } \mathbf{z}^{(i)}_{\mathrm{p}} = \left[\mathbf{z}^{1(i)}_{\text{vis}};\, \mathbf{z}^{1(i)}_{\text{vis}};\, \ldots;\, \mathbf{z}^{L(i)}_{\text{vis}};\, \mathbf{z}^{(i)}_{\text{app}}\right]. \tag{10}$$

Similarly, (8) and (9) can be written as:

$$\mathcal{L}_{\text{MultiPatchVAE}}(\mathbf{x}) = -\,\mathbb{E}_{\mathbf{z_P}}\left[G\left(\mathbf{x} \mid \mathbf{z_P}\right)\right] + \beta_{\text{app}} \sum_{i=1}^{N} \sum_{l=1}^{L} D_{\mathrm{KL}}\left(Q^{\mathrm{V}(i)}\left(\mathbf{z}^{l(i)}_{\text{vis}} \mid \mathbf{x}\right) \parallel \mathrm{Bern}\left(\mathbf{z}^{\text{prior}}_{\text{vis}}\right)\right) + \beta_{\text{vis}} \sum_{i=1}^{N} D_{\mathrm{KL}}\left(Q^{\mathrm{A}(i)}\left(\mathbf{z}^{(i)}_{\text{app}} \mid \mathbf{x}\right) \parallel \mathcal{N}(0, \mathcal{I})\right). \tag{11}$$

The training details and assumptions of posteriors follow the previous section.

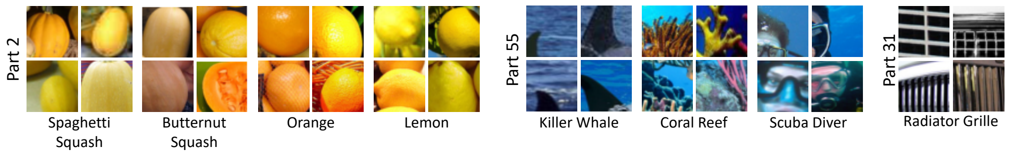


Figure 2: Concepts captured by parts: We visualize a few representative examples for several parts to qualitatively demonstrate the visual concepts captured by parts. For each part, we crop image patches centered on the part location where it is predicted to be present. Selected patches are sorted by part visibility probability as score. We have manually selected a diverse set from the top 50 occurrences from the training images. As visible, a single part may capture diverse set of concepts that are similar in shape or texture or occur in similar context, but belong to different categories. We show which categories the patches come from.

3.4 Improved Reconstruction Loss

The L2 reconstruction loss used for training β-VAEs (and other reconstruction based approaches) gives equal importance to each region of an image. This might be reasonable for tasks like image compression and image de-noising. However, for the purposes of learning semantic representations, not all regions are equally important. For example, “sky” and “walls” occupy large portions of an image, whereas concepts like “windows,” “wheels,”, “faces” are comparatively smaller, but arguably more important. To incorporate this intuition, we use a simple and intuitive strategy to weigh the regions in an image in proportion to the gradient energy in the region. More concretely, we compute laplacian of an image to get the intensity of gradients per-pixel and average the gradient magnitudes in 8×8 local patches. The weight multiplier for the reconstruction loss of each 8×8 patch in the image is proportional to the average magnitude of the patch. All weights are normalized to sum to one. We refer to this as **weighted loss ($\mathcal{L}_{\mathrm{w}}$)**. Note that this is similar to the gradient-energy biased sampling of mid-level patches used in Singh et al. (2012); Doersch et al. (2012). In Appendix 6.1, we show examples of weight masks for some of the images.

In addition, we also consider an adversarial training strategy from GANs to train VAEs as proposed by Larsen et al. (2015), where the discriminator network from GAN implicitly learns to compare images and gives a more abstract reconstruction error for the VAE. We refer to this variant by using ‘GAN’ suffix in experiments. In Section 4, we demonstrate that the proposed weighted loss ($\mathcal{L}_{\mathrm{w}}$) is complementary to the discriminator loss from adversarial training, and these losses result in better recognition capabilities for both β-VAE and PatchVAE.

4 Experiments

Datasets. We evaluate our proposed model on CIFAR100 (Krizhevsky et al., 2009), MIT Indoor Scene Recognition (Quattoni & Torralba, 2009), and Places (Zhou et al., 2017) datasets. Details of these datasets can be found in Appendix 6.2.

Learning paradigm. In order to evaluate the utility of features learned for recognition, we setup the learning paradigm as follows: we will first train the model in an unsupervised manner on all the images other than test set images. After that, we discard the generator network and use only part of the encoder network $\phi(\mathbf{x})$ to train a supervised model on the classification task of the respective dataset. We study different training strategies for the classification stage as discussed later.

Training details. In all experiments, we use the following architectures. For CIFAR100, Indoor67, and Place205, $\phi(\mathbf{x})$ has a conv layer followed by two residual blocks (He et al., 2016). For ImageNet, $\phi(\mathbf{x})$ is a ResNet18 model (a conv layer followed by four residual blocks). For all datasets, Q^{A} and Q^{V} have a single conv layer each. For classification, we start from $\phi(\mathbf{x})$, and add a fully-connected layer with 512 hidden units and a final fully-connected layer as classifier. More details can be found in Appendix 6.2 and 6.3.

During the unsupervised learning part of training, all methods are trained for 90 epochs for CIFAR100 and Indoor67, 2 epochs for Places205, and 30 epochs for ImageNet dataset. All methods use ADAM optimizer for training, with initial learning rate of 1×10^{-4} and a minibatch size of 128. For relaxed bernoulli in Q^{V}, we start with the temperature of 1.0 with an annealing rate of 3×10^{-5} (details in (Agustsson et al., 2017)). For training the classifier, all methods use stochastic gradient descent (SGD) with momentum with a minibatch size of 128. Initial learning rate is 1×10^{-2} and we reduce it by a factor of 10 every 30 epochs. All experiments are trained for 90 epochs for CIFAR100 and Indoor67, 5 epochs for Places205, and 30 epochs for ImageNet datasets.

Table 1: Classification results on CIFAR100, Indoor67, and Places205. We initialize the classification model with the representations $\phi(\mathbf{x})$ learned from unsupervised learning task. The model $\phi(\mathbf{x})$ comprises of a conv layer followed by two residual blocks (each having 2 conv layers). First column (called 'Conv1') corresponds to Top-1 classification accuracy with pre-trained model with the first conv layer frozen, second and third columns correspond to results with 3 conv layers and 5 conv layers frozen respectively. Details in Section 4.1.

	CIFAR100			Indoor67			Places205		
Model	Conv1	Conv3	Conv5	Conv1	Conv3	Conv5	Conv1	Conv3	Conv5
β-VAE	44.12	39.65	28.57	20.08	17.76	13.06	28.29	24.34	8.89
β-VAE + $\mathcal{L}_w$	44.96	40.30	28.33	21.34	19.48	13.96	29.43	24.93	9.41
β-VAE-GAN	44.69	40.13	29.89	19.10	17.84	13.06	28.48	24.51	9.72
β-VAE-GAN + $\mathcal{L}_w$	45.61	41.35	31.53	20.45	18.36	14.33	29.63	25.26	10.66
PatchVAE	43.07	38.58	28.72	20.97	19.18	13.43	28.63	24.95	11.09
PatchVAE + $\mathcal{L}_w$	43.75	40.37	30.55	23.21	21.87	15.45	29.39	26.29	12.07
PatchVAE-GAN	44.45	40.57	31.74	21.12	19.63	14.55	28.87	25.25	12.21
PatchVAE-GAN + $\mathcal{L}_w$	45.39	41.74	32.65	22.46	21.87	16.42	29.36	26.30	13.39
BiGAN	47.72	41.89	31.58	21.64	17.09	9.70	30.06	25.11	10.82
Imagenet Pretrained	55.99	54.99	54.36	45.90	45.82	40.90	37.08	36.46	31.26

Table 2: ImageNet classification results using ResNet18. We initialize weights from using the unsupervised task and fine-tune the last 2 residual blocks. Details in Section 4.1.

Model	Top-1	Top-5
β-VAE	44.45	69.67
PatchVAE	47.01	71.71
β-VAE + $\mathcal{L}_w$	47.28	71.78
PatchVAE + $\mathcal{L}_w$	47.87	72.49
Imagenet Supervised	61.37	83.79

Table 3: Effect of maximum number of patches (N)(left) and number of hidden units (d_p) for patch appearance $\hat{\mathbf{z}}_{\text{app}}$ (right) on classification accuracy. Details in Section 4.2.

N	CIFAR100	Indoor67
4	27.59	14.40
8	28.74	12.69
16	28.94	14.33
32	27.78	13.28
64	29.00	12.76

d_p	CIFAR100	Indoor67
3	28.63	14.25
6	28.97	14.55
9	28.21	14.55

Baselines. We use the **β-VAE** model (Section 3.1) as our primary baseline. In addition, we use weighted loss and discriminator loss resulting in the **β-VAE-*** family of baselines. We also compare against a **BiGAN** model from Donahue et al. (2016). We use similar backbone architectures for encoder/decoder (and discriminator if present) across all methods, and tried to keep the number of parameters in different approaches comparable to the best of our ability. Exact architecture details can be found in Appendix 6.3.

4.1 RESULTS

In Table 1, we report the top-1 classification results on CIFAR100, Indoor67, and Places205 datasets for all methods with different training strategies for classification. First, we keep all the pre-trained weights in $\phi(\mathbf{x})$ from the unsupervised task frozen and only train the two newly added conv layers in the classification network (reported under column 'Conv5'). We notice that our method (with different losses) generally outperforms the β-VAE counterpart by a healthy margin. This shows that the representations learned by PatchVAE framework are better for recognition compared to β-VAEs. Moreover, better reconstruction losses ('GAN' and $\mathcal{L}_w$) generally improve both β-VAE and PatchVAE, and are complementary to each other.

Next, we fine-tune the last residual block along with the two conv layers ('Conv3' column). We observe that PatchVAE performs better than VAE under all settings except the for CIFAR100 with just L2 loss. However, when using better reconstruction losses, the performance of PatchVAE improves over β-VAE. Similarly, we fine-tune all but the first conv layer and report the results in 'Conv1' column. Again, we notice similar trends, where our method generally performs better than β-VAE on Indoor67 and Places205 dataset, but β-VAE performs better CIFAR100 by a small margin. When compared to BiGAN, PatchVAE representations are better on all datasets ('Conv5') by a huge margin. However, when fine-tuning the pre-trained weights, BiGAN performs better on two out of four datasets. We also report results using pre-trained weights in $\phi(\mathbf{x})$ using *supervised* ImageNet

Table 4: Effect of $\mathbf{z}_{\text{vis}}^{\text{prior}}$: Increasing the prior on patch visibility has adverse effect on classification performance.

$\mathbf{z}_{\text{vis}}^{\text{prior}}$	CIFAR100	Indoor67
0.01	28.86	14.33
0.05	28.67	14.25
0.1	28.31	14.03

Table 5: Effect of β_{vis}: Too high or too low β_{vis} can deteriorate the performance of learned representations on classification.

β_{vis}	CIFAR100	Indoor67
0.06	30.11	14.10
0.3	30.37	15.67
0.6	28.90	13.51

classification task (last column, Table 1) for completeness. The results indicate that PatchVAE learns better semantic representations compared to β-VAE.

ImageNet Results. Finally, we report results on the large-scale ImageNet benchmark in Table 2. For these experiments, we use ResNet18 (He et al., 2016) architecture for all methods. All weights are first learned using the unsupervised tasks. Then, we fine-tune the last two residual blocks and train the two newly added conv layers in the classification network (therefore, first conv layer and the following two residual blocks are frozen). We notice that PatchVAE framework outperforms β-VAE under all settings, and the proposed weighted loss helps both approaches. Finally, the last row in Table 2 reports classification results of same architecture randomly initialized and trained end-to-end on ImageNet using supervised training for comparison.

4.2 Ablation Studies

We study the impact of various hyper-parameters used in our experiments. For the purpose of this evaluation, we follow a similar approach as in the 'Conv5' column of Table 1 and all hyperparameters from the previous section. We use CIFAR100 and Indoor67 datasets for ablation analysis.

Maximum number of patches. Maximum number of parts N used in our framework. Depending on the dataset, higher value of N can provide wider pool of patches to pick from. However, it can also make the unsupervised learning task harder, since in a minibatch of images, we might not get too many repeat patches. Table 3(left) shows the effect of N on CIFAR100 and Indoor67 datasets. We observe that while increasing number of patches improves the discriminative power in case of CIFAR100, it has little or negative effect in case of Indoor67. A possible reason for this decline in performance for Indoor67 can be smaller size of the dataset (i.e., fewer images to learn).

Number of hidden units for a patch appearance $\hat{\mathbf{z}}_{\text{app}}$. Next, we study the impact of the number of channels in the appearance feature $\hat{\mathbf{z}}_{\text{app}}$ for each patch (d_p). This parameter reflects the capacity of individual patch's latent representation. While this parameter impacts the reconstruction quality of images. We observed that it has little or no effect on the classification performance of the base features. Results are summarized in Table 3(right) for both CIFAR100 and Indoor67 datasets.

Prior probability for patch visibility $\mathbf{z}_{\text{vis}}^{\text{prior}}$. In all our experiments, prior probability for a patch is fixed to $1/N$, i.e., inverse of maximum number of patches. The intuition is to encourage each location on visibility maps to fire for at most one patch. Increasing this patch visibility prior will allow all patches to fire at the same location. While this would make the reconstruction task easier, it will become harder for individual patches to capture anything meaningful. Table 4 shows the deterioration of classification performance on increasing $\mathbf{z}_{\text{vis}}^{\text{prior}}$.

Patch visibility loss weight β_{vis}. The weight for patch visibility KL Divergence has to be chosen carefully. If β_{vis} is too low, more patches can fire at same location and this harms the the learning capability of patches; and if β_{vis} is too high, decoder will not receive any patches to reconstruct from and both reconstruction and classification will suffer. Table 5 summarizes the impact of varying β_{vis}.

5 Conclusion

We presented a patch-based bottleneck in a VAE framework that encourages learning useful representations for recognition. Our method, PatchVAE, constrains the encoder architecture to only learn patches that are repetitive and consistent in images as opposed to learning *everything*, and therefore results in representations that perform much better for recognition tasks compared to vanilla VAEs. We also demonstrate that losses that favor high-energy foreground regions of an image are better for unsupervised learning of representations for recognition.

REFERENCES

Lama Affara, Bernard Ghanem, and Peter Wonka. Supervised convolutional sparse coding. *CoRR*, abs/1804.02678, 2018. URL `http://arxiv.org/abs/1804.02678`.

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pp. 1141–1151, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. IEEE Conference on*, pp. 248–255. Ieee, 2009.

Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4):101:1–101:9, 2012.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Context as supervisory signal: Discovering objects with predictable context. In *European Conference on Computer Vision*, pp. 362–377. Springer, 2014.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *NIPS*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 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.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Mayank Juneja, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 923–930, 2013.

Salman H Khan, Munawar Hayat, and Nick Barnes. Adversarial training of variational auto-encoders for high fidelity image generation. *arXiv preprint arXiv:1804.10323*, 2018.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

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

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

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

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL `http://arxiv.org/abs/1511.06434`.

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.

Saurabh Singh, Abhinav Gupta, and Alexei A. Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, 2012. URL `http://arxiv.org/abs/1205.3137`.

Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 843–852, 2017.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016. URL `http://arxiv.org/abs/1601.06759`.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2015.

Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, 2017.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232, 2017.

6 Appendix

6.1 Visualization of Weighted Loss

Figure 3 shows an illustration of the reconstruction loss $\mathcal{L}_\text{w}$ proposed in Section 3.4. Notice that in first column, guitar has more weight that rest of the image. Similarly in second, fourth and sixth columns that train, painting, and people are respectively weighed more heavily by $\mathcal{L}_\text{w}$ than rest of the image; thus favoring capturing the foreground regions.

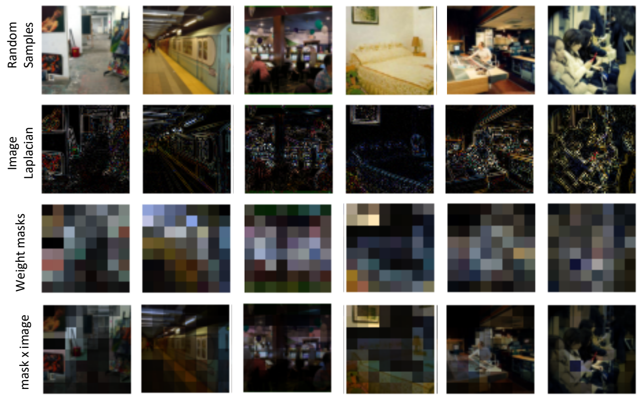


Figure 3: Masks used for weighted reconstruction loss $\mathcal{L}_\text{w}$. First row contains images randomly samples from MIT Indoor datatset. Second and third rows have the corresponding image laplacians and final reconstruction weight masks respectively. In the last row, we take the product of first and third row to highlight which parts of image are getting more attention while reconstruction.

6.2 Dataset and Training Details

CIFAR100 consists of 60000 32×32 color images in 100 classes, with 600 images per class. There are 50000 training images and 10000 test images. Indoor dataset contains 67 categories, and a total of 15620 images ('Indoor67'). Train and test subsets consist of 80 and 20 images per class respectively. Places dataset has 2.5 millions of images with 205 categories ('Places205'). Finally, we report results on the large-scale ImageNet (Deng et al., 2009) dataset, which has $\sim$1.28M training and 50k validation images spanning 1000 categories.

The generator network has two deconv layers with batchnorm (Ioffe & Szegedy, 2015) and a final deconv layer with tanh activation. When training with 'GAN' loss, the additional discriminator has four conv layers, two of with have batchnorm.

6.3 Model Architectures

In this section, we share the exact architectures used in various experiments. As discussed in Section 4, we evaluated our proposed model on CIFAR100, Indoor67, and Places205 datasets. We resize and center-crop the images such that input image size for CIFAR100 datasets is $32 \times 32 \times 3$ while for Indoor67 and Places205 datasets input image size is $64 \times 64 \times 3$. PatchVAE can treat images of various input sizes in exactly same way allowing us to keep the architecture same for different datasets. In case of VAE and BiGAN however, we have to go through a fixed size bottleneck layer and hence architectures need to be a little different for different input image sizes. Wherever possible, we have tried to keep the number of parameters in different architectures comparable.

6.3.1 Architecture for unsupervised learning task

Tables 6 and 7 show the architectures for encoders used in different models. In the unsupervised learning task, encoder comprises of a fixed neural network backbone $\phi(x)$, that given an image of size $h \times w \times 3$ generated feature maps of size $\frac{h}{8} \times \frac{w}{8} \times d_e$. This backbone architecture is common to different models discussed in the paper and consists of a single conv layer followed by 2 residual blocks. We refer to this $\phi(x)$ as Resnet-9 and it is described as Conv1-5 layers in Table 10. Rest of the encoder architecture varies depending on the model in consideration and is described in the tables below.

Tables 8 and 9 show the architectures for decoders used in different models. We use a pyramid like network for decoder where feature map size is doubled in consecutive layers, while number of channels is halved. Final non-linearity used in each decoder is $tanh$.

Table 6: Encoder architecture for unsupervised learning task on CIFAR100 - All 'convolutional' layers are represented as (kernel_size $\times$ kernel_size, channels, stride, pad). BN stands for batch normalization layer and ReLU for Rectified Linear Units.

Layer	β-VAE	BiGAN	PatchVAE
Features ϕ	Resnet-9	Resnet-9	Resnet-9
Q^{V}	-	-	$(3 \times 3, 16, 1, 1)$
Q^{A}	$(1 \times 1, 64, 1, 0)$ BN ReLU $\mu : (4 \times 4, 96, 1, 0)$ $\sigma^2 : (4 \times 4, 96, 1, 0)$	$(1 \times 1, 64, 1, 0)$ BN ReLU $(4 \times 4, 96, 1, 0)$	$\mu : (3 \times 3, 96, 1, 1)$ $\sigma^2 : (3 \times 3, 96, 1, 1)$
# Parameters	888,192	789,792	922,896

6.3.2 Architecture for supervised learning task

As discussed in Section 4, during the supervised learning phase, we discard rest of the encoder model and only keep $\phi(x)$ for classifier training. So the architectures for all baselines are exactly the same. Tables 10 shows the architecture for classifier used in our experiments.

Table 7: Encoder architecture for unsupervised learning task on Indoor67 and Places205 - All 'convolutional' layers are represented as (kernel_size $\times$ kernel_size, channels, stride, pad). BN stands for batch normalization layer and ReLU for Rectified Linear Units. Note that PatchVAE and β-VAE architectures are slightly different to account for sizes.

Layer	β-VAE	BiGAN	PatchVAE
Features ϕ	Resnet-9	Resnet-9	Resnet-9
Q^{V}	-	-	$(3 \times 3, 16, 1, 1)$
Q^{A}	$(1 \times 1, 64, 1, 0)$ BN ReLU $\mu : (8 \times 8, 96, 1, 0)$ $\sigma^2 : (8 \times 8, 96, 1, 0)$	$(1 \times 1, 64, 1, 0)$ BN ReLU $(8 \times 8, 96, 1, 0)$	$\mu : (3 \times 3, 96, 1, 1)$ $\sigma^2 : (3 \times 3, 96, 1, 1)$
# Parameters	1,478,016	1,084,704	922,896

Table 8: Decoder architecture for unsupervised earning task on CIFAR100 - All 'deconvolutional' layers are represented as (kernel_size $\times$ kernel_size, channels, stride, pad). BN stands for batch normalization layer and ReLU for Rectified Linear Units.

	β-VAE	BiGAN	PatchVAE
Model	$(4 \times 4, 64, 1, 0)$ BN LeakyReLU(0.2) $(1 \times 1, 256, 1, 0)$ BN LeakyReLU(0.2) $(4 \times 4, 128, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 64, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 3, 2, 1)$ $tanh$	$(4 \times 4, 64, 1, 0)$ BN LeakyReLU(0.2) $(1 \times 1, 256, 1, 0)$ BN LeakyReLU(0.2) $(4 \times 4, 128, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 64, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 3, 2, 1)$ $tanh$	$(1 \times 1, 256, 1, 0)$ BN LeakyReLU(0.2) $(4 \times 4, 128, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 64, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 3, 2, 1)$ $tanh$
# Parameters	774,144	774,144	683,904

Table 9: Decoder architecture for unsupervised learning task on Indoor67 and Places205 - All 'deconvolutional' layers are represented as (kernel_size $\times$ kernel_size, channels, stride, pad). BN stands for batch normalization layer and ReLU for Rectified Linear Units. Note that PatchVAE and β-VAE architectures are slightly different to account for sizes.

	β-VAE	BiGAN	PatchVAE
Model	$(8 \times 8, 64, 1, 0)$ BN LeakyReLU(0.2) $(1 \times 1, 256, 1, 0)$ BN LeakyReLU(0.2) $(4 \times 4, 128, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 64, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 3, 2, 1)$ $tanh$	$(8 \times 8, 64, 1, 0)$ BN LeakyReLU(0.2) $(1 \times 1, 256, 1, 0)$ BN LeakyReLU(0.2) $(4 \times 4, 128, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 64, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 3, 2, 1)$ $tanh$	$(1 \times 1, 256, 1, 0)$ BN LeakyReLU(0.2) $(4 \times 4, 128, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 64, 2, 1)$ BN LeakyReLU(0.2) $(4 \times 4, 3, 2, 1)$ $tanh$
# Parameters	1,069,056	1,069,056	683,904

Table 10: Architecture for supervised learning task - same for all baselines and our model. All convolutional layers are represented as (kernel_size × kernel_size, channels, stride, pad). BN stands for batch bormalization layer and ReLU for Rectified Linear Units. All pooling operations are MaxPool and are represented by (kernel_size × kernel_size, $stride, pad$). Like Resnet-18, downsampling happens by convolutional layers that have a stride of 2. In our model, downsampling happens during Conv1, Pool, and after Conv4-5.

Layer	CIFAR100 ($32 \times 32 \times 3$)	Indoor67 ($64 \times 64 \times 3$)	Places205 ($64 \times 64 \times 3$)
Conv1	$1 \times \begin{cases} (7 \times 7, 64, 2, 3) \\ \text{BN} \\ \text{ReLU} \\ \text{Pool}(3 \times 3, 2, 1) \end{cases}$	$1 \times \begin{cases} (7 \times 7, 64, 2, 3) \\ \text{BN} \\ \text{ReLU} \\ \text{Pool}(3 \times 3, 2, 1) \end{cases}$	$1 \times \begin{cases} (7 \times 7, 64, 2, 3) \\ \text{BN} \\ \text{ReLU} \\ \text{Pool}(3 \times 3, 2, 1) \end{cases}$
Conv2-3	$2 \times \begin{cases} (3 \times 3, 64, 1, 1) \\ \text{BN} \\ \text{ReLU} \\ (3 \times 3, 64, 1, 1) \\ \text{BN} \end{cases}$	$2 \times \begin{cases} (3 \times 3, 64, 1, 1) \\ \text{BN} \\ \text{ReLU} \\ (3 \times 3, 64, 1, 1) \\ \text{BN} \end{cases}$	$2 \times \begin{cases} (3 \times 3, 64, 1, 1) \\ \text{BN} \\ \text{ReLU} \\ (3 \times 3, 64, 1, 1) \\ \text{BN} \end{cases}$
Conv4-5	$2 \times \begin{cases} (3 \times 3, 128, 1, 1) \\ \text{BN} \\ \text{ReLU} \\ (3 \times 3, 128, 1, 1) \\ \text{BN} \end{cases}$	$2 \times \begin{cases} (3 \times 3, 128, 1, 1) \\ \text{BN} \\ \text{ReLU} \\ (3 \times 3, 128, 1, 1) \\ \text{BN} \end{cases}$	$2 \times \begin{cases} (3 \times 3, 128, 1, 1) \\ \text{BN} \\ \text{ReLU} \\ (3 \times 3, 128, 1, 1) \\ \text{BN} \end{cases}$
FC	2048×512 512×100	8192×512 512×67	8192×512 512×205
# Parameters	1,783,460	4,912,259	4,983,053