PatchVAE: Learning Local Latent Codes for Recognition
Supplementary Materials

Kamal Gupta[1] Saurabh Singh[2] Abhinav Shrivastava[1]
[1]University of Maryland, College Park [2]Google Research
{kampta,abhinav}@cs.umd.edu saurabhsingh@google.com

A. Training Details

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

B. Visualization of Weighted Loss

Figure 1 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.

C. Model Architecture

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.

C.1. Architecture for unsupervised learning task

Tables 1 and 2 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 5. Rest of the encoder architecture varies depending on the model in consideration and is described in the Tables 1 and 2.

Tables 3 and 4 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.

C.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 5 shows the architecture for classifier used in our experiments.

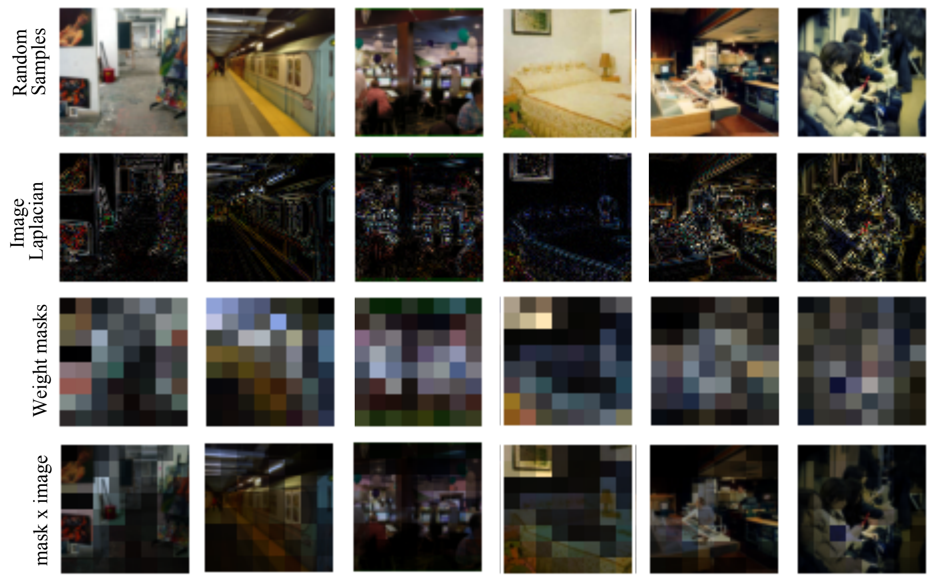

Figure 1: Masks used for weighted reconstruction loss $\mathcal{L}_{\mathrm{w}}$. First row contains images randomly samples from MIT Indoor datatset. Second and third rows have the corresponding image laplacians and final reconstruction weight masks respectively. In the last row, we take the product of first and third row to highlight which parts of image are getting more attention while reconstruction.

Table 1: 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^{O}	-	-	$(3 \times 3, 16, 1, 1)$
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

Table 2: 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^{O}	-	-	$(3 \times 3, 16, 1, 1)$
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 3: Decoder architecture for unsupervised earning task on CIFAR100 - All 'deconvolutional' layers are represented as (kernel_size × 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 4: Decoder architecture for unsupervised learning task on Indoor67 and Places205 - All 'deconvolutional' layers are represented as (kernel_size × 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 5: Architecture for supervised learning task - same for all baselines and our model. All convolutional layers are represented as (kernel_size $\times$ 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 $\times$ 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