

Context Encoders: Feature Learning by Inpainting

Deepak Pathak

Philipp Krähenbühl

Jeff Donahue

Trevor Darrell

Alexei A. Efros

University of California, Berkeley

{pathak, philkr, jdonahue, trevor, efros}@cs.berkeley.edu

Abstract

We present an unsupervised visual feature learning algorithm driven by context-based pixel prediction. By analogy with auto-encoders, we propose Context Encoders – a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. In order to succeed at this task, context encoders need to both understand the content of the entire image, as well as produce a plausible hypothesis for the missing part(s). When training context encoders, we have experimented with both a standard pixel-wise reconstruction loss, as well as a reconstruction plus an adversarial loss. The latter produces much sharper results because it can better handle multiple modes in the output. We found that a context encoder learns a representation that captures not just appearance but also the semantics of visual structures. We quantitatively demonstrate the effectiveness of our learned features for CNN pre-training on classification, detection, and segmentation tasks. Furthermore, context encoders can be used for semantic inpainting tasks, either stand-alone or as initialization for non-parametric methods.

1. Introduction

Our visual world is very diverse, yet highly structured, and humans have an uncanny ability to make sense of this structure. In this work, we explore whether state-of-the-art computer vision algorithms can do the same. Consider the image shown in Figure 1a. Although the center part of the image is missing, most of us can easily imagine its content from the surrounding pixels, without having ever seen that exact scene. Some of us can even draw it, as shown on Figure 1b. This ability comes from the fact that natural images, despite their diversity, are highly structured (e.g. the regular pattern of windows on the facade). We humans are able to understand this structure and make visual predictions even when seeing only parts of the scene. In this paper, we show

The supplementary material, trained models and code are available at the author’s website.

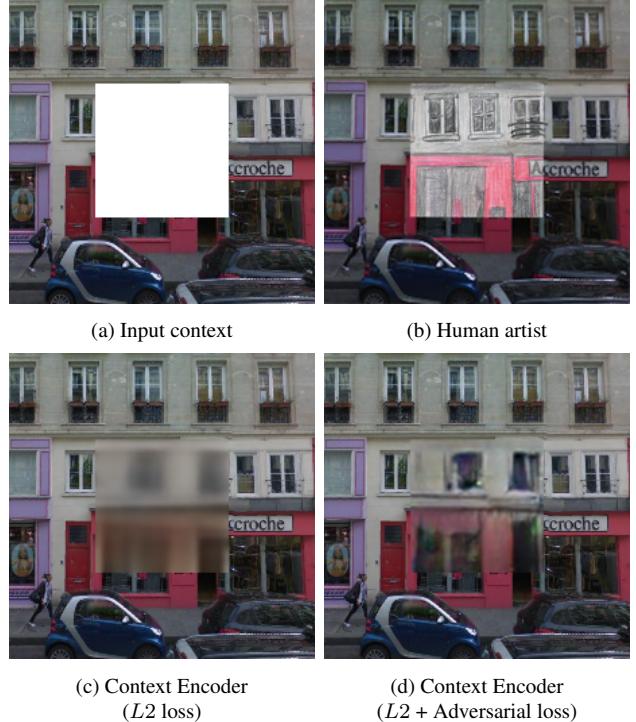


Figure 1: Qualitative illustration of the task. Given an image with a missing region (a), a human artist has no trouble inpainting it (b). Automatic inpainting using our *context encoder* trained with L_2 reconstruction loss is shown in (c), and using both L_2 and adversarial losses in (d).

that it is possible to learn and predict this structure using convolutional neural networks (CNNs), a class of models that have recently shown success across a variety of image understanding tasks.

Given an image with a missing region (e.g., Fig. 1a), we train a convolutional neural network to regress to the missing pixel values (Fig. 1d). We call our model *context encoder*, as it consists of an encoder capturing the context of an image into a compact latent feature representation and a decoder which uses that representation to produce the missing image content. The context encoder is closely related to autoencoders [3, 20], as it shares a similar encoder-decoder architecture. Autoencoders take an input image and try

to reconstruct it after it passes through a low-dimensional “bottleneck” layer, with the aim of obtaining a compact feature representation of the scene. Unfortunately, this feature representation is likely to just compresses the image content without learning a semantically meaningful representation. Denoising autoencoders [38] address this issue by corrupting the input image and requiring the network to undo the damage. However, this corruption process is typically very localized and low-level, and does not require much semantic information to undo. In contrast, our context encoder needs to solve a much harder task: to fill in large missing areas of the image, where it can’t get “hints” from nearby pixels. This requires a much deeper semantic understanding of the scene, and the ability to synthesize high-level features over large spatial extents. For example, in Figure 1a, an entire window needs to be conjured up “out of thin air.” This is similar in spirit to word2vec [30] which learns word representation from natural language sentences by predicting a word given its context.

Like autoencoders, context encoders are trained in a completely unsupervised manner. Our results demonstrate that in order to succeed at this task, a model needs to both understand the content of an image, as well as produce a plausible hypothesis for the missing parts. This task, however, is inherently multi-modal as there are multiple ways to fill the missing region while also maintaining coherence with the given context. We decouple this burden in our loss function by jointly training our context encoders to minimize both a reconstruction loss and an adversarial loss. The reconstruction (L2) loss captures the overall structure of the missing region in relation to the context, while the the adversarial loss [16] has the effect of picking a particular mode from the distribution. Figure 1 shows that using only the reconstruction loss produces blurry results, whereas adding the adversarial loss results in much sharper predictions.

We evaluate the encoder and the decoder independently. On the encoder side, we show that encoding just the context of an image patch and using the resulting feature to retrieve nearest neighbor contexts from a dataset produces patches which are semantically similar to the original (unseen) patch. We further validate the quality of the learned feature representation by fine-tuning the encoder for a variety of image understanding tasks, including classification, object detection, and semantic segmentation. We are competitive with the state-of-the-art unsupervised/self-supervised methods on those tasks. On the decoder side, we show that our method is often able to fill in realistic image content. Indeed, to the best of our knowledge, ours is the first parametric inpainting algorithm that is able to give reasonable results for semantic hole-filling (i.e. large missing regions). The context encoder can also be useful as a better visual feature for computing nearest neighbors in non-parametric inpainting methods.

2. Related work

Computer vision has made tremendous progress on semantic image understanding tasks such as classification, object detection, and segmentation in the past decade. Recently, Convolutional Neural Networks (CNNs) [13, 27] have greatly advanced the performance in these tasks [15, 26, 28]. The success of such models on image classification paved the way to tackle harder problems, including unsupervised understanding and generation of natural images. We briefly review the related work in each of the sub-fields pertaining to this paper.

Unsupervised learning CNNs trained for ImageNet [37] classification with over a million labeled examples learn features which generalize very well across tasks [9]. However, whether such semantically informative and generalizable features can be learned from raw images alone, without any labels, remains an open question. Some of the earliest work in deep unsupervised learning are autoencoders [3, 20]. Along similar lines, denoising autoencoders [38] reconstruct the image from local corruptions, to make encoding robust to such corruptions. While context encoders could be thought of as a variant of denoising autoencoders, the corruption applied to the model’s input is spatially much larger, requiring more semantic information to undo.

Weakly-supervised and self-supervised learning Very recently, there has been significant interest in learning meaningful representations using weakly-supervised and self-supervised learning. One useful source of supervision is to use the temporal information contained in videos. Consistency across temporal frames has been used as supervision to learn embeddings which perform well on a number of tasks [17, 34]. Another way to use consistency is to track patches in frames of video containing task-relevant attributes and use the coherence of tracked patches to guide the training [39]. Ego-motion read off from non-vision sensors has been used as supervisory signal to train visual features *et al.* [1, 21].

Most closely related to the present paper are efforts at exploiting spatial context as a source of free and plentiful supervisory signal. Visual Memex [29] used context to non-parametrically model object relations and to predict masked objects in scenes, while [6] used context to establish correspondences for unsupervised object discovery. However, both approaches relied on hand-designed features and did not perform any representation learning. Recently, Doersch *et al.* [7] used the task of predicting the relative positions of neighboring patches within an image as a way to train an unsupervised deep feature representations. We share the same high-level goals with Doersch *et al.* but fundamentally

differ in the approach: whereas [7] are solving a *discriminative* task (is patch A above patch B or below?), our context encoder solves a pure *prediction* problem (what pixel intensities should go in the hole?). Interestingly, similar distinction exist in using language context to learn word embeddings: Collobert and Weston [5] advocate a discriminative approach, whereas word2vec [30] formulate it as word prediction. One important benefit of our approach is that our supervisory signal is much richer: a context encoder needs to predict roughly 15,000 real values per training example, compared to just 1 option among 8 choices in [7]. Likely due in part to this difference, our context encoders take far less time to train than [7]. Moreover, context based prediction is also harder to “cheat” since low-level image features, such as chromatic aberration, do not provide any meaningful information, in contrast to [7] where chromatic aberration partially solves the task. On the other hand, it is not yet clear if requiring faithful pixel generation is necessary for learning good visual features.

Image generation Generative models of natural images have enjoyed significant research interest [16, 24, 35]. Recently, Radford *et al.* [33] proposed new convolutional architectures and optimization hyperparameters for Generative Adversarial Networks (GAN) [16] producing encouraging results. We train our context encoders using an adversary jointly with reconstruction loss for generating inpainting results. We discuss this in detail in Section 3.2.

Dosovitskiy *et al.* [10] and Rifai *et al.* [36] demonstrate that CNNs can learn to generate novel images of particular object categories (chairs and faces, respectively), but rely on large labeled datasets with examples of these categories. In contrast, context encoders can be applied to any unlabeled image database and learn to generate images based on the surrounding context.

Inpainting and hole-filling It is important to point out that our hole-filling task cannot be handled by classical inpainting [4, 32] or texture synthesis [2, 11] approaches, since the missing region is too large for local non-semantic methods to work well. In computer graphics, filling in large holes is typically done via scene completion [19], involving a cut-paste formulation using nearest neighbors from a dataset of millions of images. However, scene completion is meant for filling in holes left by removing whole objects, and it struggles to fill arbitrary holes, e.g. amodal completion of partially occluded objects. Furthermore, previous completion relies on a hand-crafted distance metric, such as Gist [31] for nearest-neighbor computation which is inferior to a learned distance metric. We show that our method is often able to inpaint semantically meaningful content in a parametric fashion, as well as provide a better feature for nearest neighbor-based inpainting methods.

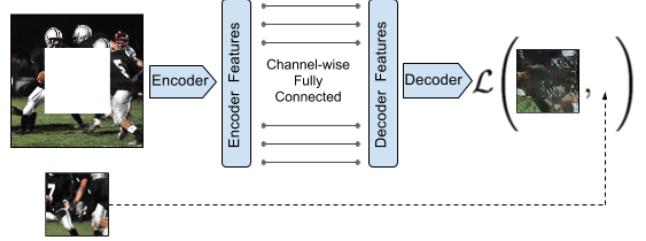


Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

3. Context encoders for image generation

We now introduce context encoders: CNNs that predict missing parts of a scene from their surroundings. We first give an overview of the general architecture, then provide details on the learning procedure and finally present various strategies for image region removal.

3.1. Encoder-decoder pipeline

The overall architecture is a simple encoder-decoder pipeline. The encoder takes an input image with missing regions and produces a latent feature representation of that image. The decoder takes this feature representation and produces the missing image content. We found it important to connect the encoder and the decoder through a channel-wise fully-connected layer, which allows each unit in the decoder to reason about the entire image content. Figure 2 shows an overview of our architecture.

Encoder Our encoder is derived from the *AlexNet* architecture [26]. Given an input image of size 227×227 , we use the first five convolutional layers and the following pooling layer (called *pool5*) to compute an abstract $6 \times 6 \times 256$ dimensional feature representation. In contrast to *AlexNet*, our model is not trained for ImageNet classification; rather, the network is trained for context prediction “from scratch” with randomly initialized weights.

However, if the encoder architecture is limited only to convolutional layers, there is no way for information to directly propagate from one corner of the feature map to another. This is so because convolutional layers connect all the feature maps together, but never directly connect all locations within a specific feature map. In the present architectures, this information propagation is handled by *fully-connected* or *inner product* layers, where all the activations are directly connected to each other. In our architecture, the latent feature dimension is $6 \times 6 \times 256 = 9216$ for both encoder and decoder. This is so because, unlike autoen-

coders, we do not reconstruct the original input and hence need not have a smaller *bottleneck*. However, fully connecting the encoder and decoder would result in an explosion in the number of parameters (over 100M!), to the extent that efficient training on current GPUs would be difficult. To alleviate this issue, we use a channel-wise fully-connected layer to connect the encoder features to the decoder, described in detail below.

Channel-wise fully-connected layer This layer is essentially a fully-connected layer with groups, intended to propagate information within activations of each feature map. If the input layer has m feature maps of size $n \times n$, this layer will output m feature maps of dimension $n \times n$. However, unlike a fully-connected layer, it has no parameters connecting different feature maps and only propagates information within feature maps. Thus, the number of parameters in this channel-wise fully-connected layer is mn^4 , compared to m^2n^4 parameters in a fully-connected layer (ignoring the bias term). This is followed by a stride 1 convolution to propagate information across channels.

Decoder We now discuss the second half of our pipeline, the decoder, which generates pixels of the image using the encoder features. The “encoder features” are connected to the “decoder features” using a channel-wise fully-connected layer.

The channel-wise fully-connected layer is followed by a series of five *up-convolutional* layers [10, 28, 40] with learned filters, each with a rectified linear unit (ReLU) activation function. A up-convolutional is simply a convolution that results in a higher resolution image. It can be understood as upsampling followed by convolution (as described in [10]), or convolution with fractional stride (as described in [28]). The intuition behind this is straightforward – the series of up-convolutions and non-linearities comprises a non-linear weighted upsampling of the feature produced by the encoder until we roughly reach the original target size.

3.2. Loss function

We train our context encoders by regressing to the ground truth content of the missing (dropped out) region. However, there are often multiple equally plausible ways to fill a missing image region which are consistent with the context. We model this behavior by having a decoupled joint loss function to handle both continuity within the context and multiple modes in the output. The reconstruction (L2) loss is responsible for capturing the overall structure of the missing region and coherence with regards to its context, but tends to average together the multiple modes in predictions. The adversarial loss [16], on the other hand, tries to make prediction look real, and has the effect of picking a particular mode from the distribution. For each ground truth

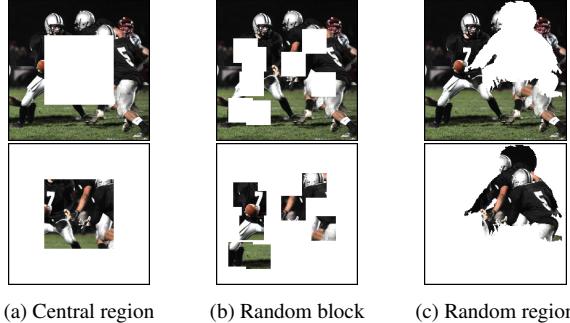


Figure 3: An example of image x with our different region masks \hat{M} applied, as described in Section 3.3.

image x , our context encoder F produces an output $F(x)$. Let \hat{M} be a binary mask corresponding to the dropped image region with a value of 1 wherever a pixel was dropped and 0 for input pixels. During training, those masks are automatically generated for each image and training iterations, as described in Section 3.3. We now describe different components of our loss function.

Reconstruction Loss We use a normalized masked L2 distance as our reconstruction loss function, \mathcal{L}_{rec} ,

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2, \quad (1)$$

where \odot is the element-wise product operation. We experimented with both L1 and L2 losses and found no significant difference between them. While this simple loss encourages the decoder to produce a rough outline of the predicted object, it often fails to capture any high frequency detail (see Fig. 1c). This stems from the fact that the L2 (or L1) loss often prefer a blurry solution, over highly accurate textures. We believe this happens because it is much “safer” for the L2 loss to predict the mean of the distribution, because this minimizes the mean pixel-wise error, but results in a blurry averaged image. We alleviated this problem by adding an adversarial loss.

Adversarial Loss Our adversarial loss is based on Generative Adversarial Networks (GAN) [16]. To learn a generative model G of a data distribution, GAN proposes to jointly learn an adversarial discriminative model D to provide loss gradients to the generative model. G and D are parametric functions (e.g., deep networks) where $G : \mathcal{Z} \rightarrow \mathcal{X}$ maps samples from noise distribution \mathcal{Z} to data distribution \mathcal{X} . The learning procedure is a two-player game where an adversarial discriminator D takes in both the prediction of G and ground truth samples, and tries to distinguish them, while G tries to confuse D by producing samples that appear as “real” as possible. The objective for discriminator is logistic likelihood indicating whether the input is real sam-

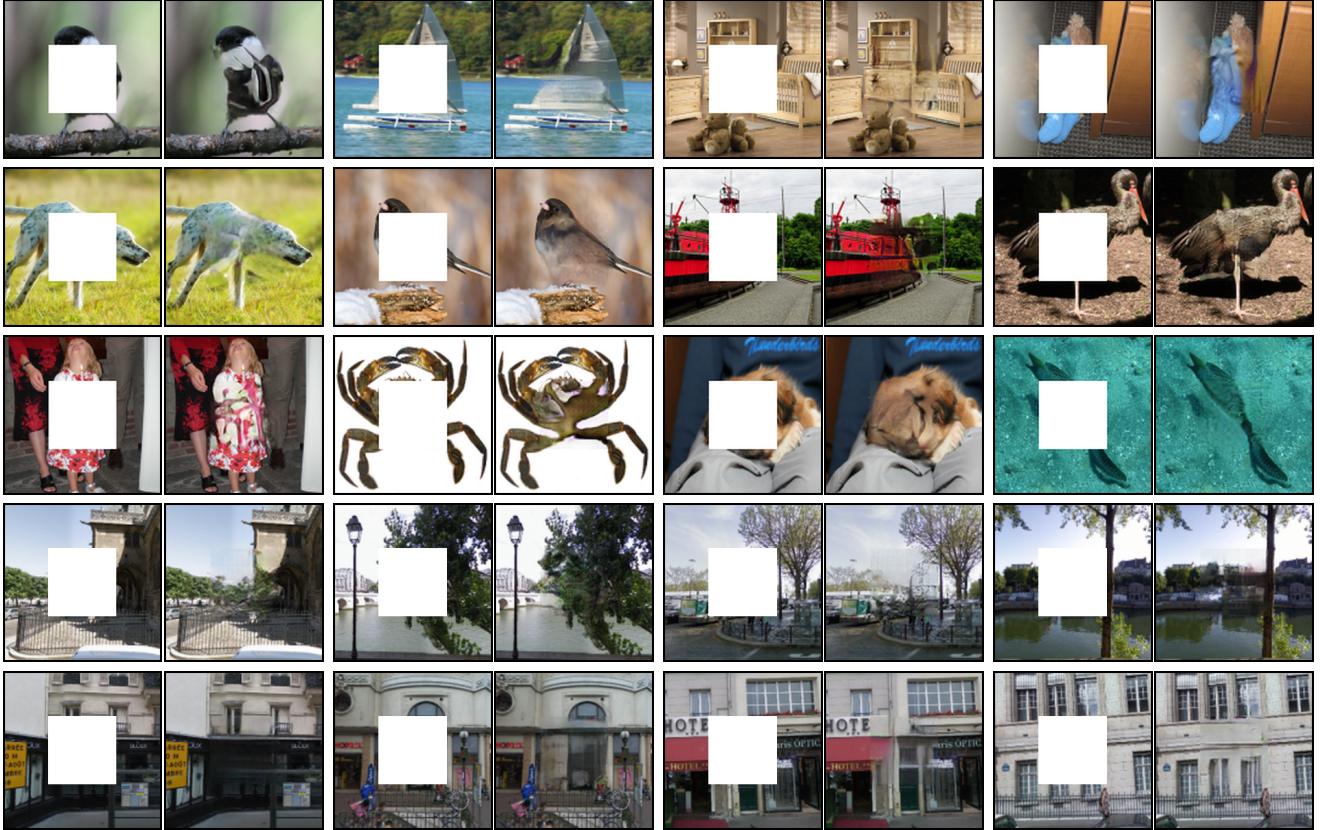


Figure 4: Semantic Inpainting results on *held-out* images for context encoder trained using reconstruction and adversarial loss. First three rows are examples from ImageNet, and bottom two rows are from Paris StreetView Dataset.

ple or predicted one:

$$\min_G \max_D \mathbb{E}_{x \in \mathcal{X}}[\log(D(x))] + \mathbb{E}_{z \in \mathcal{Z}}[\log(1 - D(G(z)))]$$

This method has recently shown encouraging results in generative modeling of images [33]. We thus adapt this framework for context prediction by modeling generator by context encoder; i.e., $G \triangleq F$. To customize GANs for this task, one could condition on the given context information; i.e., the mask $\hat{M} \odot x$. However, conditional GANs don't train easily for context prediction task as the adversarial discriminator D easily exploits the perceptual discontinuity in generated regions and the original context to easily classify predicted versus real samples. We thus use an alternate formulation, by conditioning only the generator (not the discriminator) on context. We also found results improved when the generator was not conditioned on a noise vector. Hence the adversarial loss for context encoders, \mathcal{L}_{adv} , is

$$\begin{aligned} \mathcal{L}_{adv} = \max_D & \mathbb{E}_{x \in \mathcal{X}}[\log(D(x))] \\ & + \log(1 - D(F((1 - \hat{M}) \odot x))), \end{aligned} \quad (2)$$

where, in practice, both F and D are optimized jointly using alternating SGD. Note that this objective encourages the

entire output of the context encoder to look realistic, not just the missing regions as in Equation (1).

Joint Loss We define the overall loss function as

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}. \quad (3)$$

Currently, we use adversarial loss only for inpainting experiments as AlexNet [26] architecture training diverged with joint adversarial loss. Details follow in Sections 5.1, 5.2.

3.3. Region masks

The input to a context encoder is an image with one or more of its regions “dropped out”; i.e., set to zero, assuming zero-centered inputs. The removed regions could be of any shape, we present three different strategies here:

Central region The simplest such shape is the central square patch in the image, as shown in Figure 3a. While this works quite well for inpainting, the network learns low level image features that latch onto the boundary of the central mask. Those low level image features tend not to generalize well to images without masks, hence the features learned are not very general.

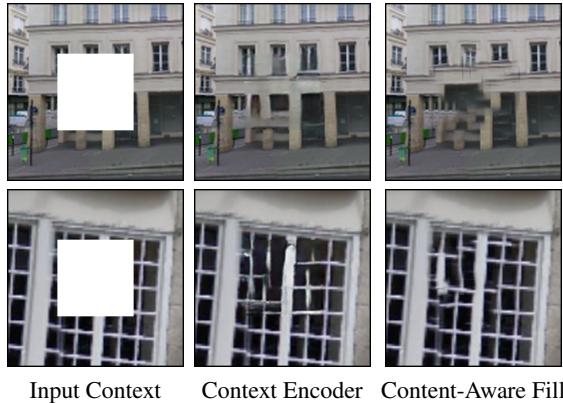


Figure 5: Comparison with Content-Aware Fill (Photoshop feature based on [2]) on *held-out* images. Our method works better in semantic cases (top row) and works slightly worse in textured settings (bottom row).

Random block To prevent the network from latching on the constant boundary of the masked region, we randomize the masking process. Instead of choosing a single large mask at a fixed location, we remove a number of smaller possibly overlapping masks, covering up to $\frac{1}{4}$ of the image. An example of this is shown in Figure 3b. However, the random block masking still has sharp boundaries convolutional features could latch onto.

Random region To completely remove those boundaries, we experimented with removing arbitrary shapes from images, obtained from random masks in the PASCAL VOC 2012 dataset [12]. We deform those shapes and paste in arbitrary places in the other images (not from PASCAL), again covering up to $\frac{1}{4}$ of the image. Note that we completely randomize the region masking process, and do not expect or want any correlation between the source segmentation mask and the image. We merely use those regions to prevent the network from learning low-level features corresponding to the removed mask. See example in Figure 3c.

In practice, we found region and random block masks produce a similarly general feature, while significantly outperforming the central region features. We use the random region dropout for all our feature based experiments.

4. Implementation details

The pipeline was implemented in *Caffe* [22] and *Torch*. We used the recently proposed stochastic gradient descent solver, ADAM [23] for optimization. The missing region in the masked input image is filled with constant mean value. Hyper-parameter details are discussed in Sections 5.1, 5.2.

Pool-free encoders We experimented with replacing all pooling layers with convolutions of the same kernel size and stride. The overall stride of the network remains the same, but it results in finer inpainting. Intuitively, there is no reason to use pooling for reconstruction based networks.

Method	Mean L1 Loss	Mean L2 Loss	PSNR (higher better)
NN-inpainting (HOG features)	19.92%	6.92%	12.79 dB
NN-inpainting (our features)	15.10%	4.30%	14.70 dB
Our Reconstruction (joint)	09.37%	1.96%	18.58 dB

Table 1: Semantic Inpainting accuracy for Paris StreetView dataset on *held-out* images. NN inpainting is basis for [19].

In classification, pooling provides spatial invariance, which may be detrimental for reconstruction-based training. To be consistent with prior work, we still use the original AlexNet architecture (with pooling) for all feature learning results.

5. Evaluation

We now evaluate the encoder features for their semantic quality and transferability to other image understanding tasks. We experiment with images from two datasets: Paris StreetView [8] and ImageNet [37] without using any of the accompanying labels. In Section 5.1, we present visualizations demonstrating the ability of the context encoder to fill in semantic details of images with missing regions. In Section 5.2, we demonstrate the transferability of our learned features to other tasks, using context encoders as a pre-training step for image classification, object detection, and semantic segmentation. We compare our results on these tasks with those of other unsupervised or self-supervised methods, demonstrating that our approach outperforms previous methods.

5.1. Semantic Inpainting

We train context encoders with the joint loss function defined in Equation (3) for the task of inpainting the missing region. The encoder and discriminator architecture is similar to that of discriminator in [33], and decoder is similar to generator in [33]. However, the bottleneck is of 4000 units (in contrast to 100 in [33]; see supplementary material. We used the default solver hyper-parameters suggested in [33]. We use $\lambda_{rec} = 0.999$ and $\lambda_{adv} = 0.001$. However, a few things were crucial for training the model. We did not condition the adversarial loss (see Section 3.2) nor did we add noise to the encoder. We use a higher learning rate for context encoder (10 times) to that of adversarial discriminator. To further emphasize the consistency of prediction with the context, we predict a slightly larger patch that overlaps with the context (by 7px). During training, we use higher weight ($10\times$) for the reconstruction loss in this overlapping region.

The qualitative results are shown in Figure 4. Our model performs generally well in inpainting semantic regions of an image. However, if a region can be filled with low-level textures, texture synthesis methods, such as [2, 11], can often perform better (e.g. Figure 5). For semantic inpainting, we compare against nearest neighbor inpainting (which forms the basis of Hays *et al.* [19]) and show that

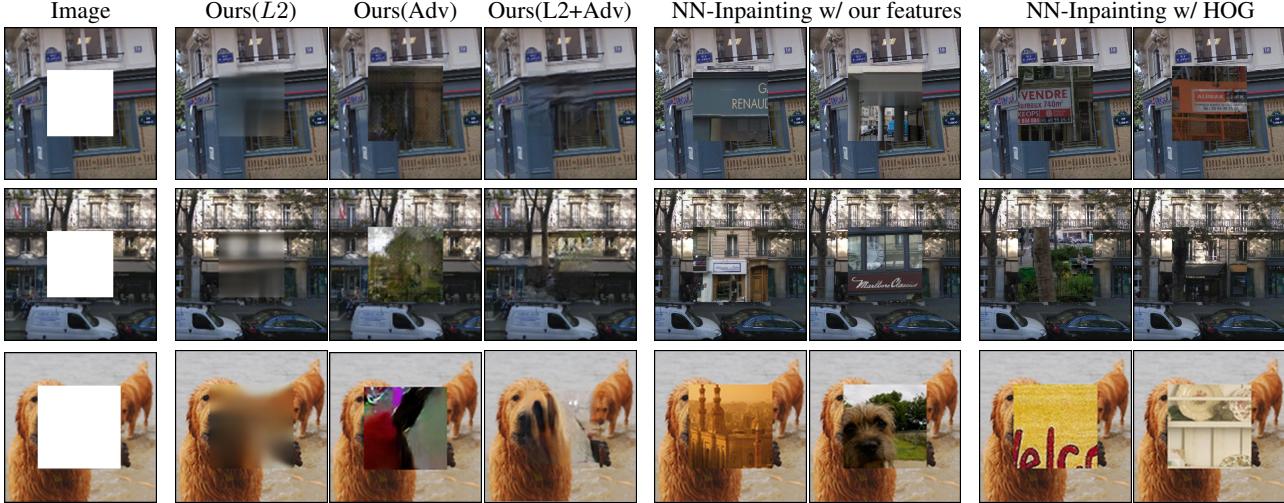


Figure 6: Semantic Inpainting using different methods on *held-out* images. Context Encoder with just L2 are well aligned, but not sharp. Using adversarial loss, results are sharp but not coherent. Joint loss alleviate the weaknesses of each of them. The last two columns are the results if we plug-in the best nearest neighbor (NN) patch in the masked region.

our reconstructions are well-aligned semantically, as seen on Figure 6. It also shows that joint loss significantly improves the inpainting over both reconstruction and adversarial loss alone. Moreover, using our learned features in a nearest-neighbor style inpainting can sometimes improve results over a hand-designed distance metrics. Table 1 reports quantitative results on StreetView Dataset.

5.2. Feature Learning

For consistency with prior work, we use the AlexNet [26] architecture for our encoder. Unfortunately, we did not manage to make the adversarial loss converge with AlexNet, so we used just the reconstruction loss. The networks were trained with a constant learning rate of 10^{-3} for the center-region masks. However, for random region corruption, we found a learning rate of 10^{-4} to perform better. We apply dropout with a rate of 0.5 just for the channel-wise fully connected layer, since it has more parameters than other layers and might be prone to overfitting. The training process is fast and converges in about 100K iterations: 14 hours on a Titan X GPU. Figure 7 shows inpainting results for context encoder trained with random region corruption using reconstruction loss. To evaluate the quality of features, we find nearest neighbors

to the masked part of image just by using the features from the context, see Figure 8. Note that none of the methods ever see the center part of any image, whether a query or dataset image. Our features retrieve decent nearest neighbors just from context, even though actual prediction is blurry with L2 loss. AlexNet features also perform decently as they were trained with 1M labels for semantic tasks, HOG on the other hand fail to get the semantics.

5.2.1 Classification pre-training

For this experiment, we fine-tune a standard AlexNet classifier on the PASCAL VOC 2007 [12] from a number of supervised, self-supervised and unsupervised initializations. We train the classifier using random cropping, and then evaluate it using 10 random crops per test image. We average the classifier output over those random crops. Table 2 shows the standard mean average precision (mAP) score for all compared methods.

A random initialization performs roughly 25% below an ImageNet-trained model; however, it does not use any labels. Context encoders are competitive with concurrent self-supervised feature learning methods [7, 39] and significantly outperform autoencoders and Agrawal *et al.* [1].

5.2.2 Detection pre-training

Our second set of quantitative results involves using our features for object detection. We use *Fast R-CNN* [14] framework (FRCN). We replace the ImageNet pre-trained network with our context encoders (or any other baseline model). In particular, we take the pre-trained encoder weights up to the *pool5* layer and re-initialize the fully-

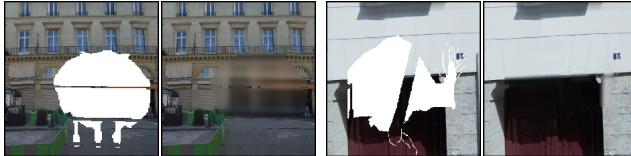


Figure 7: Arbitrary region inpainting for context encoder trained with reconstruction loss on *held-out* images.



Figure 8: Context Nearest Neighbors. Center patches whose context (not shown here) are close in the embedding space of different methods (namely our context encoder, HOG and AlexNet). Note that the appearance of these center patches themselves was never seen by these methods. But our method brings them close just from their context.

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	78.2%	56.8%	48.0%
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal <i>et al.</i> [1]	egomotion	10 hours	52.9%	41.8%	-
Doersch <i>et al.</i> [7]	context	4 weeks	55.3%	46.6%	-
Wang <i>et al.</i> [39]	motion	1 week	58.4%	44.0%	-
Ours	context	14 hours	56.5%	44.5%	29.7%

Table 2: Quantitative comparison for classification, detection and semantic segmentation. Classification and Fast-RCNN Detection results are on the PASCAL VOC 2007 test set. Semantic segmentation results are on the PASCAL VOC 2012 validation set from the FCN evaluation described in Section 5.2.3, using the additional training data from [18], and removing overlapping images from the validation set [28].

connected layers. We then follow the training and evaluation procedures from FRCN and report the accuracy (in mAP) of the resulting detector.

Our results on the test set of the PASCAL VOC 2007 [12] detection challenge are reported in Table 2. Context encoder pre-training is competitive with the existing methods achieving significant boost over the baseline. Recently, Krähenbühl *et al.* [25] proposed a data-dependent method for rescaling pre-trained model weights. This significantly improves the features in Doersch *et al.* [7] up to 65.3% for classification and 51.1% for detection. However, this rescaling doesn't improve results for other methods, including ours.

5.2.3 Semantic Segmentation pre-training

Our last quantitative evaluation explores the utility of context encoder training for pixel-wise semantic segmentation. *Fully convolutional networks* [28] (FCNs) were proposed as an end-to-end learnable method of predicting a semantic label at each pixel of an image, using a convolutional network pre-trained for ImageNet classification. We replace the classification pre-trained network used in the FCN method with

our context encoders, afterwards following the FCN training and evaluation procedure for direct comparison with their original *CaffeNet*-based result.

Our results on the PASCAL VOC 2012 [12] validation set are reported in Table 2. In this setting, we outperform a randomly initialized network as well as a plain autoencoder which is trained simply to reconstruct its full input.

6. Conclusion

Our context encoders trained to generate images conditioned on context advance the state of the art in semantic inpainting, at the same time learn feature representations that are competitive with other models trained with auxiliary supervision.

Acknowledgements The authors would like to thank Amanda Buster for the artwork on Fig. 1b, as well as Shubham Tulsiani and Saurabh Gupta for helpful discussions. This work was supported in part by DARPA, AFRL, Intel, DoD MURI award N000141110688, NSF awards IIS-1212798, IIS-1427425, and IIS-1536003, the Berkeley Vision and Learning Center and Berkeley Deep Drive.

References

- [1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. *ICCV*, 2015. 2, 7, 8
- [2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 2009. 3, 6
- [3] Y. Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2009. 1, 2
- [4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Computer graphics and interactive techniques*, 2000. 3
- [5] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008. 3
- [6] C. Doersch, A. Gupta, and A. A. Efros. Context as supervisory signal: Discovering objects with predictable context. In *ECCV*, 2014. 2
- [7] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. *ICCV*, 2015. 2, 3, 7, 8
- [8] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 2012. 6
- [9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *ICML*, 2014. 2
- [10] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. *CVPR*, 2015. 3, 4
- [11] A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 3, 6
- [12] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes challenge: A retrospective. *IJCV*, 2014. 6, 7, 8
- [13] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 1980. 2
- [14] R. Girshick. Fast r-cnn. *ICCV*, 2015. 7
- [15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2
- [16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 2, 3, 4
- [17] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. *ICCV*, 2015. 2
- [18] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 8
- [19] J. Hays and A. A. Efros. Scene completion using millions of photographs. *SIGGRAPH*, 2007. 3, 6
- [20] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006. 1, 2
- [21] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015. 2
- [22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014. 6
- [23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6
- [24] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014. 3
- [25] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *ICLR*, 2016. 8
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 3, 5, 7, 8, 10
- [27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 2
- [28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 4, 8
- [29] T. Malisiewicz and A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NIPS*, 2009. 2
- [30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 2, 3
- [31] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 2006. 3
- [32] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 2005. 3
- [33] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016. 3, 5, 6, 10
- [34] V. Ramanathan, K. Tang, G. Mori, and L. Fei-Fei. Learning temporal embeddings for complex video analysis. *ICCV*, 2015. 2
- [35] M. Ranzato, V. Mnih, J. M. Susskind, and G. E. Hinton. Modeling natural images using gated mrf's. *PAMI*, 2013. 3
- [36] S. Rifai, Y. Bengio, A. Courville, P. Vincent, and M. Mirza. Disentangling factors of variation for facial expression recognition. In *ECCV*, 2012. 3
- [37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 2, 6
- [38] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 2
- [39] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. *ICCV*, 2015. 2, 7, 8
- [40] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 4

Supplementary Material

In this section, we present the architectural details of our context-encoders, and show additional qualitative results. Context encoders are not only able to inpaint semantic details in the missing part of an input image, but also learn features transferable to other tasks. We discuss the implementation details for each of these in following sections.

A. Semantic Inpainting

Context encoders for inpainting are trained jointly with reconstruction and adversarial loss as discussed in Section 5.1. The inpainting results are slightly worse if we use 227×227 directly. So, we resize images to 128×128 and then train our joint loss with the resized images. The encoder and discriminator architecture is similar to that of discriminator in [33], and decoder is similar to generator in [33]; the bottleneck is of 4000 units. We used batch normalization in both context encoder and discriminator. ReLU [26] non-linearity is used in decoder, while leaky ReLU [33] is used in both encoder and discriminator.

In case of arbitrary region inpainting, adversarial discriminator compares the full real image and the full generated image. We do not condition the adversarial discriminator with mask, see (2). If the discriminator sees the mask, it figures out the perceptual discontinuity of generated part from the real part and easily classifies the real v/s the generated image, i.e., the process doesn't train. Moreover, particularly for center region inpainting, this process can be computationally simplified by producing center only and not showing discriminator the context boundary (or in other words, not showing the mask). The exact architecture for center region dropout is shown in Figure 9a.

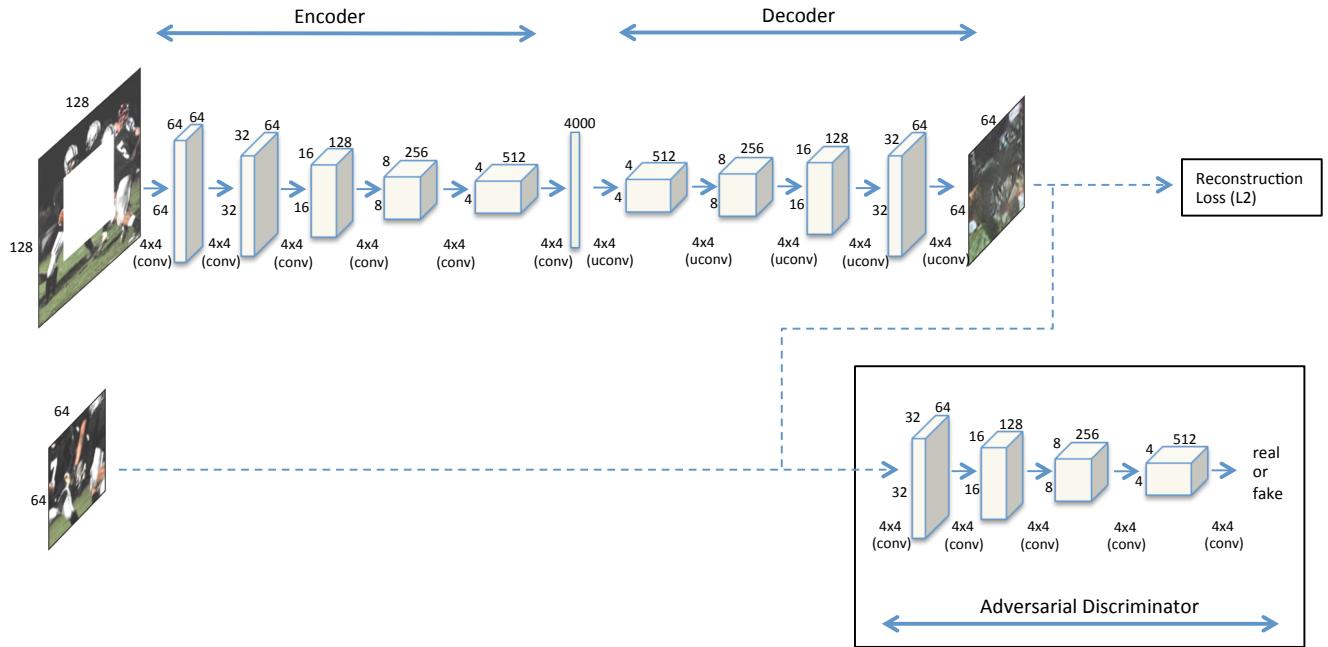
B. Feature Learning

We use the AlexNet [26] architecture for encoder so that we can compare the learned features with the prior works, which are trained using Imagenet labels and other un/self-supervised techniques. The encoder is Alexnet until pool5, followed by channel-wise fully connected layer and decoder is a series of upconvolutional layers until we reach the target size. The input image size is 227×227 . Unfortunately, we couldn't train adversary with Alexnet Encoder, so it is trained with reconstruction loss. See Figure 9b for exact architecture details. For pre-training experiments in Section 5.2, we randomly initialize the fully-connected layers, i.e., $fc6$ and $fc7$, while starting from context encoder weights.

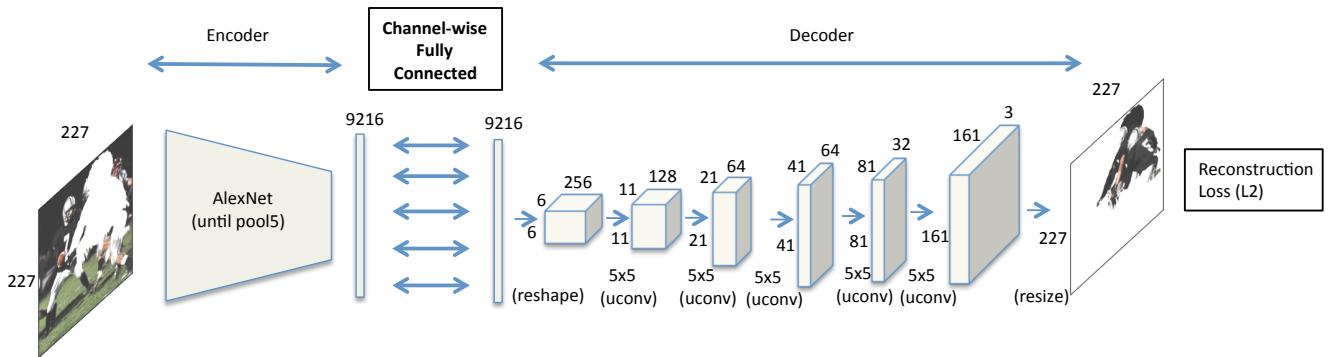
C. Additional Results

Finally, we show additional inpainting results using our context-encoders in Figure 10. These results, in comparison to nearest-neighbor inpainting, show that: (a) The fea-

tures learned by context-encoder are semantically meaningful and retrieve neighboring patches just by looking at the context. This is also verified quantitatively in Table 2. (b) Our context encoder doesn't memorize the examples from training set. It rather produces realistic and coherent inpainting results which are much better than nearest neighbor inpainting both qualitatively (Figure 10) and quantitatively (Table 1).



(a) Context encoder trained with joint reconstruction and adversarial loss for semantic inpainting. This illustration is shown for *center region dropout*. Similar architecture holds for arbitrary region dropout as well. See Section 3.2.



(b) Context encoder trained with reconstruction loss for feature learning by filling in *arbitrary region dropouts* in the input.

Figure 9: Context encoder training architectures.



Figure 10: Semantic Inpainting using different methods on *held-out* images. Context Encoder with just L_2 are well aligned, but not sharp. Using adversarial loss, results are sharp but not coherent. Joint loss alleviate the weaknesses of each of them. The last two columns are the results if we plug-in the best nearest neighbor (NN) patch in the masked region.