

# Context Encoders: Feature Learning by Inpainting

COMPUTER VISION PROJECT

INSTRUCTOR: DR. ANOOP NAMBOODIRI  
TA MENTOR: SHREYA REDDY TERUPALLY

# Team Members

- Rajashekhar Reddy (2018122010)
- Amitesh Singh (20171131)
- Tanmai Mukku (20171145)



# The problem

## Context

Our visual world is very diverse, yet highly structured, and humans have an uncanny ability to make sense of this structure like filling up the missing parts. In this work, few state-of-the-art computer vision algorithms have been presented.

## Problem statement

Design and test 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.

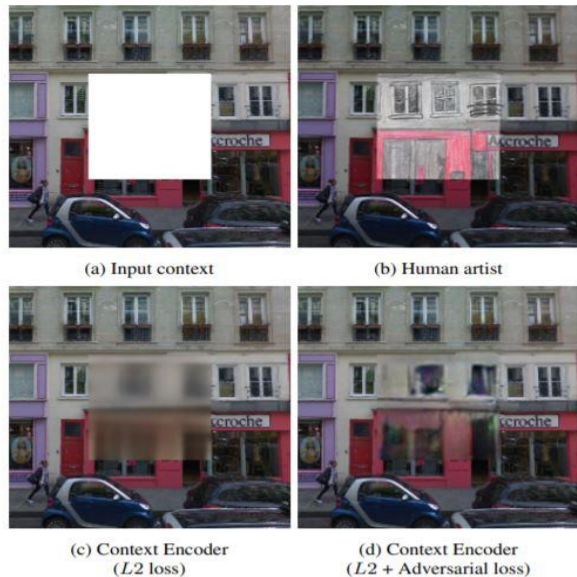
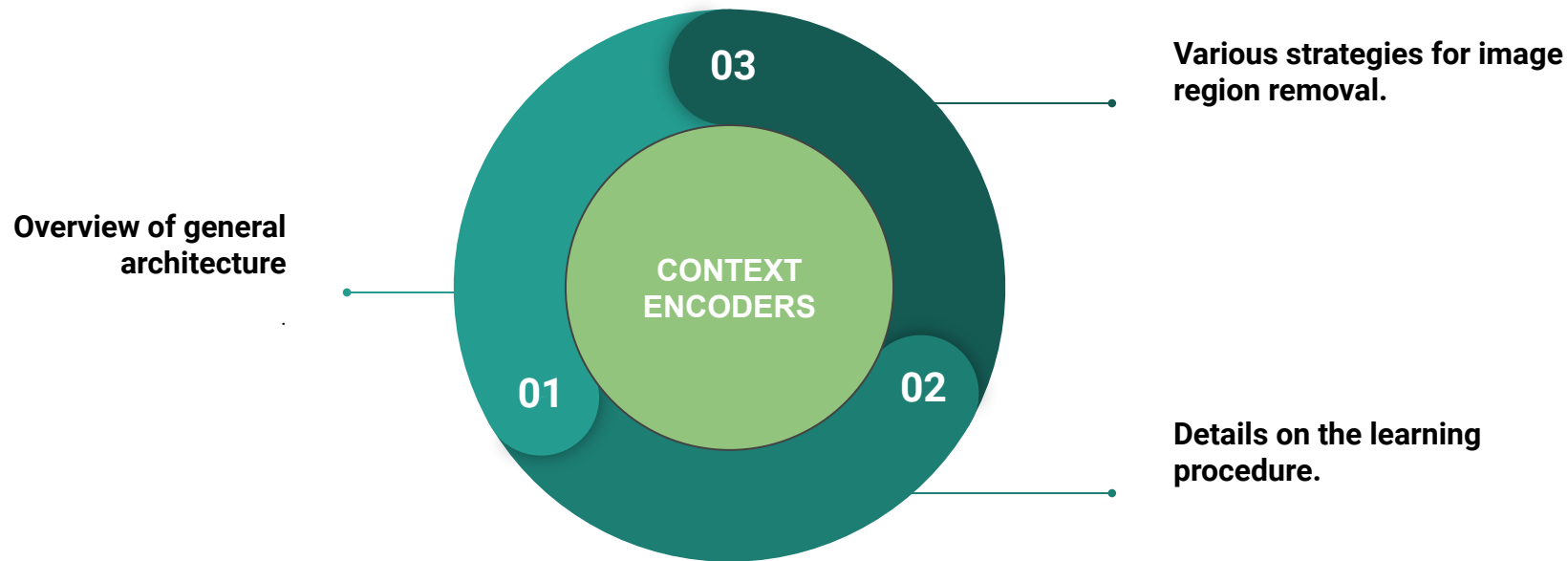


Figure 1: Qualitative illustration of the task. Given an image with a missing region (a), a human artist has no trouble inpainting it (b). Automatic inpainting using our *context encoder* trained with  $L_2$  reconstruction loss is shown in (c), and using both  $L_2$  and adversarial losses in (d).

# Overview



# Overview of General Architecture

- 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.
- The encoder and the decoder are connected through a channelwise fully-connected layer, which allows each unit in the decoder to reason about the entire image content.

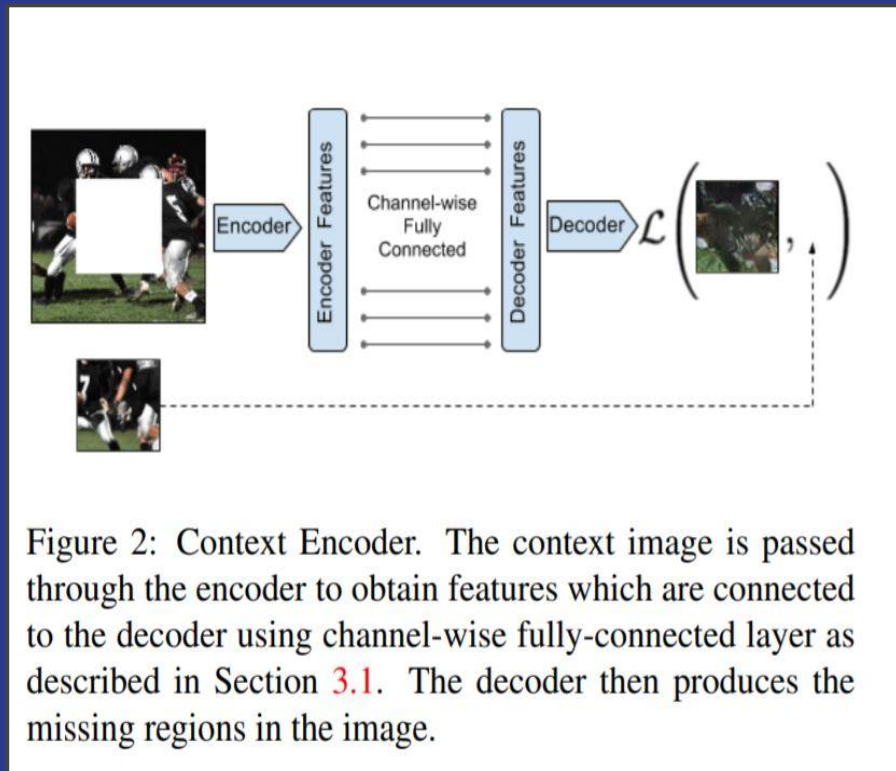


Figure 2: Context Encoder. The context image is passed through the encoder to obtain features which are connected to the decoder using channel-wise fully-connected layer as described in Section 3.1. The decoder then produces the missing regions in the image.

# Encoder Decoder Pipeline

- **Encoder** : The encoder is derived from the AlexNet architecture. Given an input image of size  $227 \times 227$ , 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.
- **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  $n \times n$ .
- **Decoder** : The channel-wise fully-connected layer is followed by a series of five up-convolutional layers with learned filters, each with a rectified linear activation function.



# Details on the learning Procedure

- We train the 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.
- This behavior is modeled by having a decoupled joint loss function (reconstruction loss and adversarial loss) to handle both continuity within the context and multiple modes in the output.
- For each ground truth image  $x$ , the context encoder  $F$  produces an output  $F(x)$ . Let  $\mathbf{M}$  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.

# Different components of the loss function.

- **Reconstruction Loss** : 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.

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

- **Adversarial Loss** : The adversarial loss tries to make prediction look real, and has the effect of picking a particular mode from the distribution.

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

- **Joint Loss** :

$$\mathcal{L} = \lambda_{rec} \mathcal{L}_{rec} + \lambda_{adv} \mathcal{L}_{adv}.$$

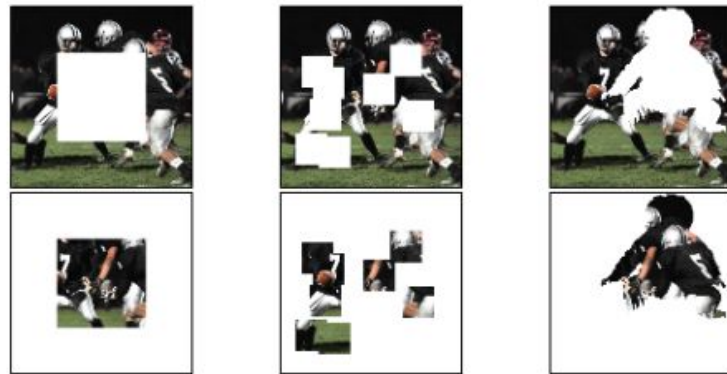




# Various strategies for image region removal

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
- Random block
- Random region



(a) Central region

(b) Random block

(c) Random region

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

### Central Region

The simplest such shape is the central square patch in the image. 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.



### Random Block

To prevent the network from latching on to 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 25% 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, removing arbitrary shapes from images can be tried. We deform those shapes and paste in arbitrary places in the other images, again covering up to  $\frac{1}{4}$  of the image. We 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.

## Various Strategies for image region removal

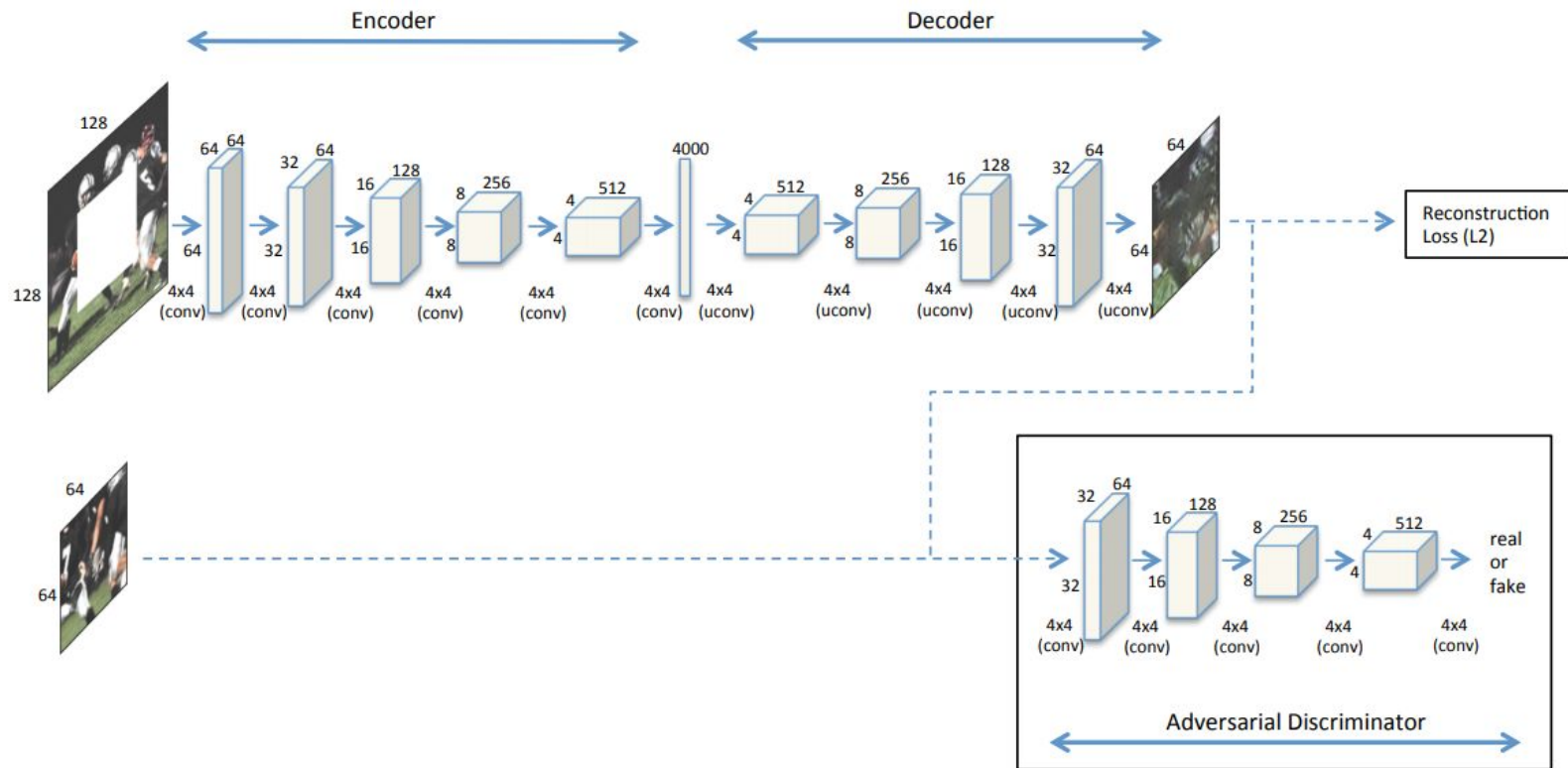
# Experiments

- The pipeline was implemented in tensorflow.
  - The stochastic gradient descent solver, ADAM was used for optimization. The missing region in the masked input image is filled with constant mean value.
  - The training and the other parameters are taken according to the paper.
  - Pool-free encoders have been used instead as shown in the paper 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.
- 

# Evaluation

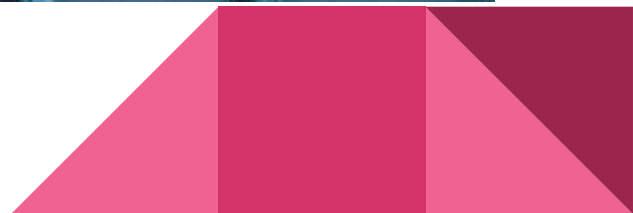
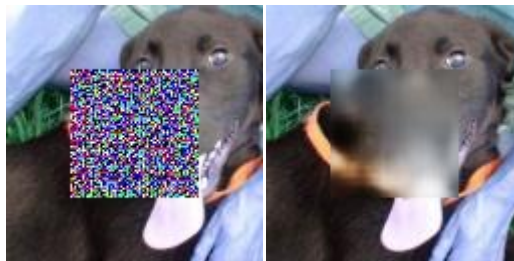
- For now we are working mainly on Paris streetview dataset without the labels
- To be experimented on the Isun dataset
- Feature learning part is still pending and to be presented using Pretraining time, Classification accuracy, Detection, and Segmentation



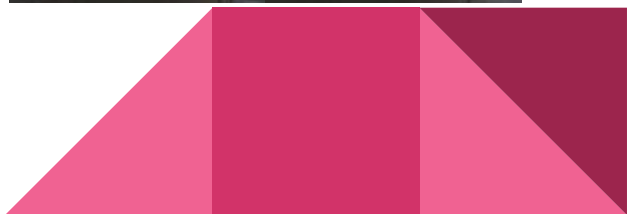


(a) Context encoder trained with joint reconstruction and adversarial loss for semantic inpainting. This illustration is shown for *center region dropout*. Similar architecture holds for arbitrary region dropout as well. See Section 3.2.

# Results



# Results (Paris Street View Dataset)





# Results (increasing the iterations)





# Results (increasing the iterations)



# Comparison



# REFERENCE PAPER -

## Context Encoders: Feature Learning by Inpainting

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell Alexei A. Efros, University of California, Berkeley

Link - <https://arxiv.org/pdf/1604.07379.pdf>