# Context-Aware Image Inpainting for Automatic Object Removal

Nicholas Russell

# What is Image Inpainting?

Overview:

- Realistically filling or reconstructing parts of an image that are damaged or missing.

Applications:

- Image Editing
- Image Reconstruction
- Painting or Image Restoration
- Object Removal

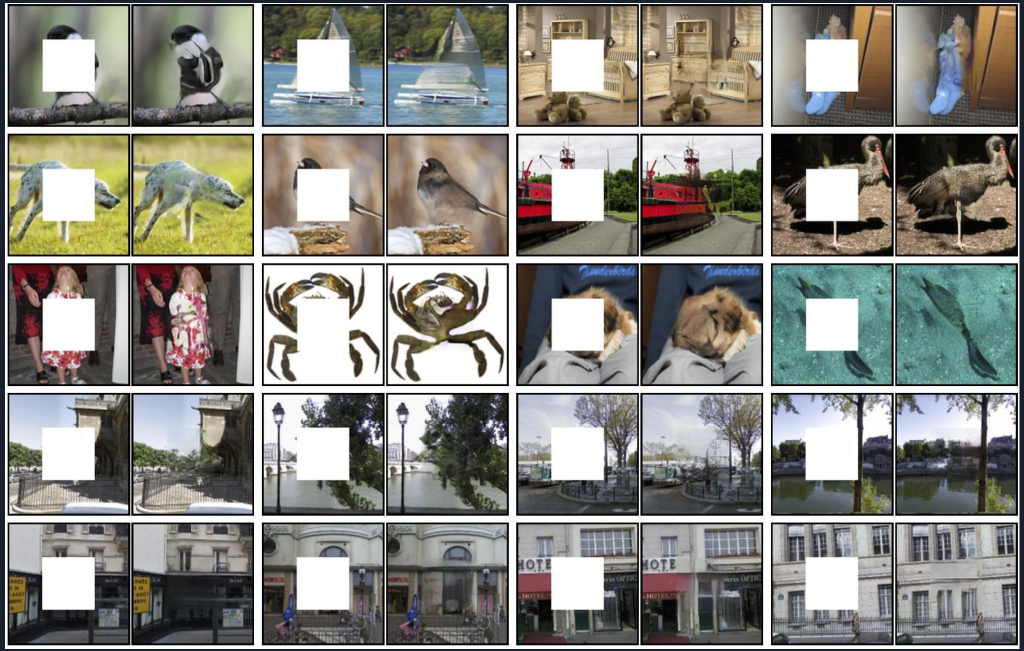
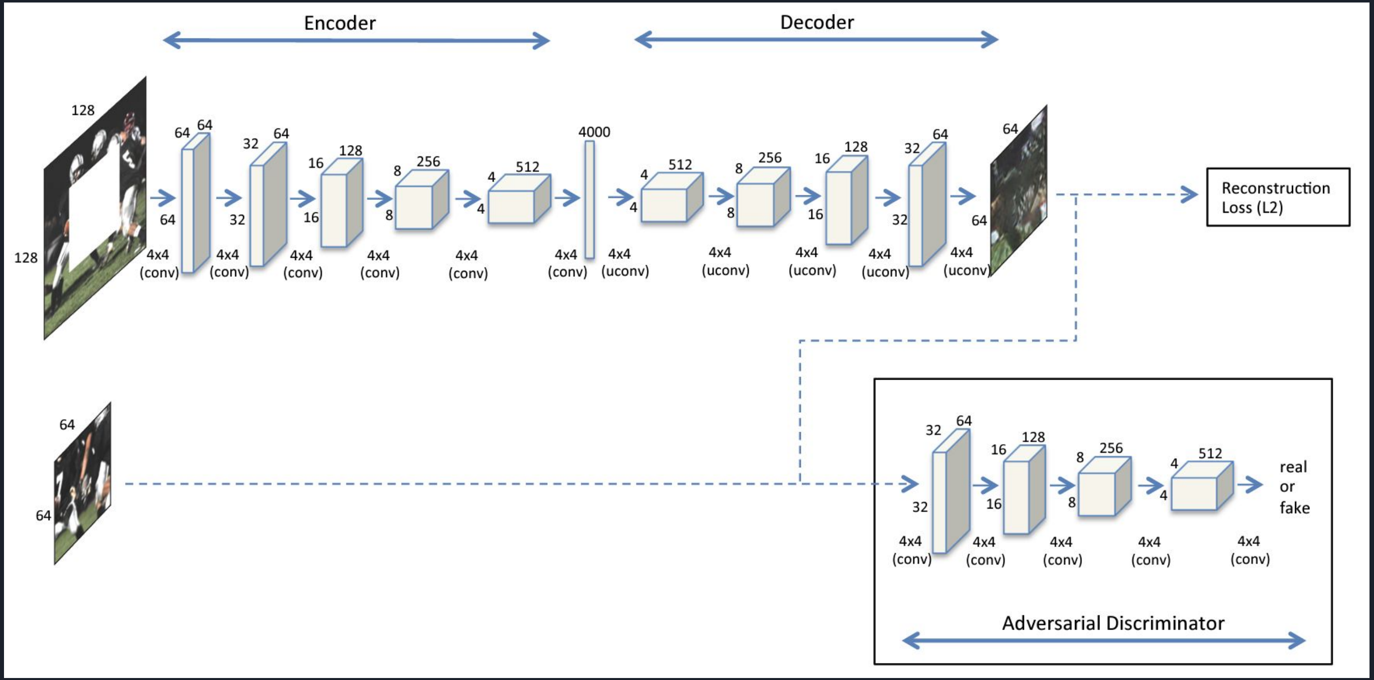



Image from https://www.cs.cmu.edu/~dpathak/context_encoder/

# Image Inpainting Examples


Images from https://advimman.github.io/lama-project/

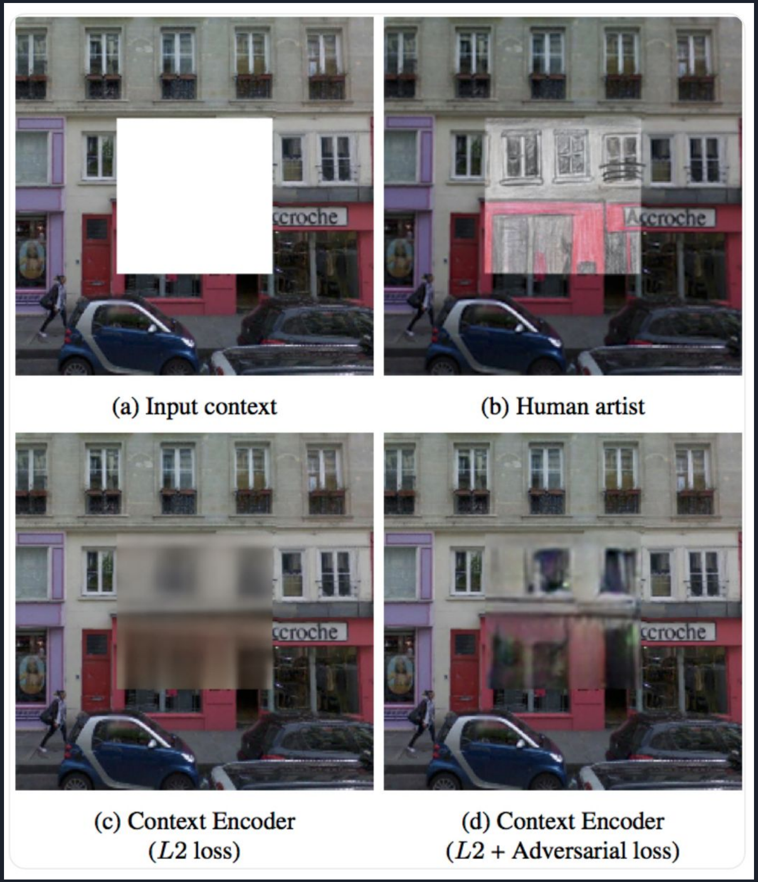
# Context Encoders



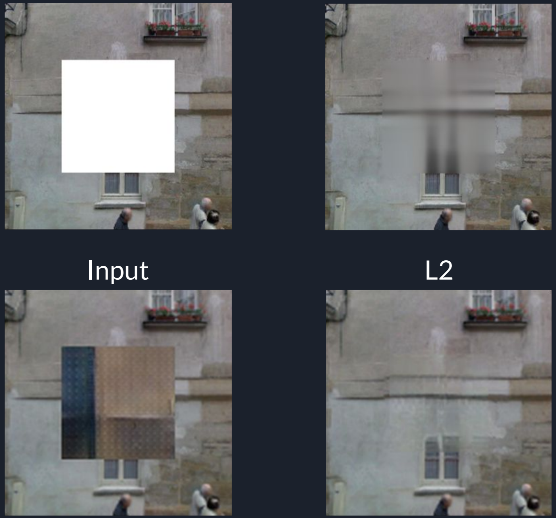

Context Encoders Architecture [1]

Loss: $\mathcal{L} = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec}$ with BCE adversarial and L2 reconstruction losses.

# Context Encoders Example



(a) Input context (b) Human artist
(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)




- Reconstruction loss alone results in good structure, but blurry results.
- Solution: Use a decoupled loss with an adversarial factor to exploit the structure of the reconstruction loss and the sharpness of the adversarial loss.

# Methodology

- Experiment with reconstruction metrics and implement different joint reconstruction losses to achieve better visual results.
- Modify the model for automatic object removal:
    - Modify the baseline Context Encoders model to support evaluation with arbitrary masks.
    - Integrate YOLOv8, an object detection model, with Context Encoders to generate masks to remove objects from the scene.

# Dataset

- We used the MiniPlaces dataset, which is a subset of MIT's Places dataset that only contains 100,000 128x128 images from 100 scene categories.

https://github.com/CSAILVision/miniplaces

# Metrics

- L1

$$L1 = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

- L2

$$L2 = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

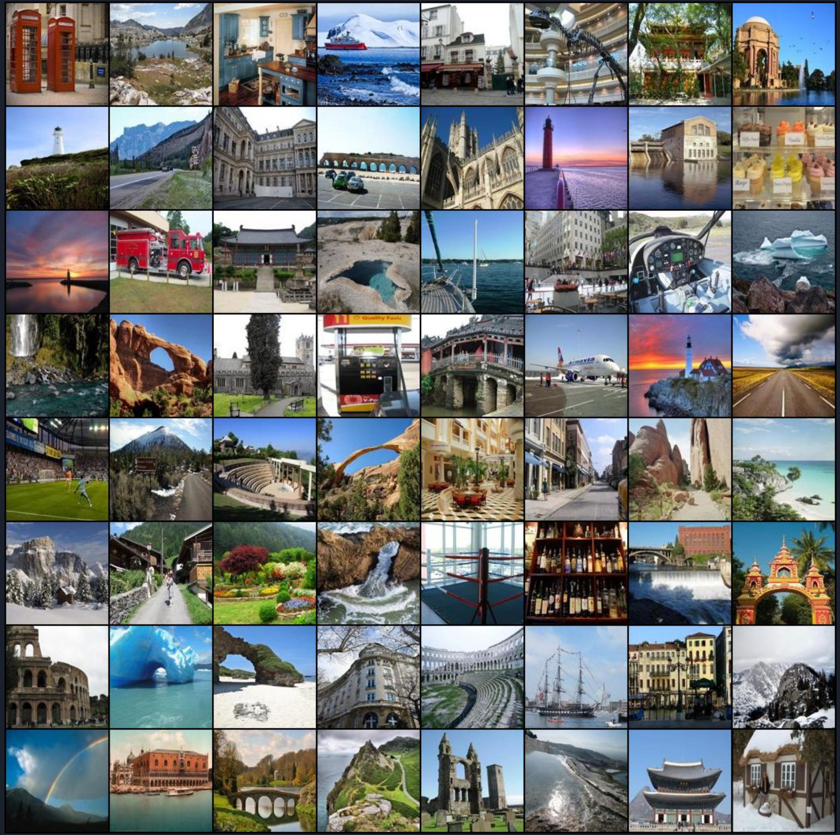
- PSNR

$$PSNR = 10 \cdot \log_{10}\left(\frac{(MAX_I)^2}{\sqrt{MSE}}\right)$$

- SSIM

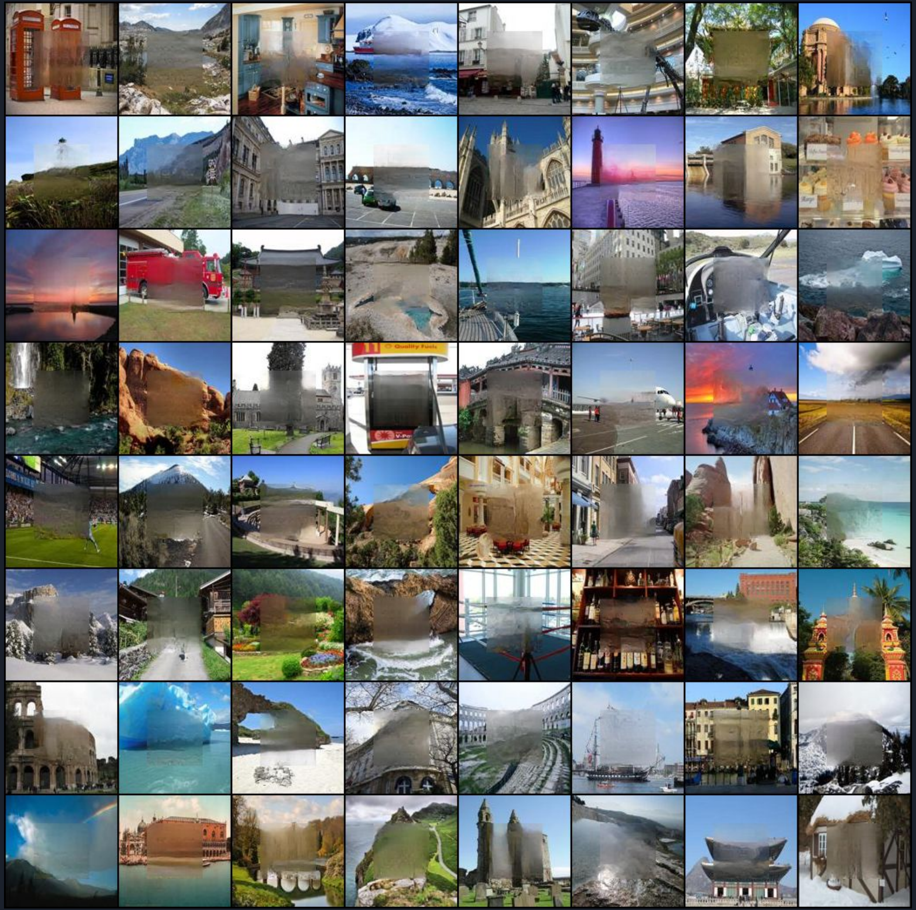
$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

# Initial Results & Motivation for Proposed Solution

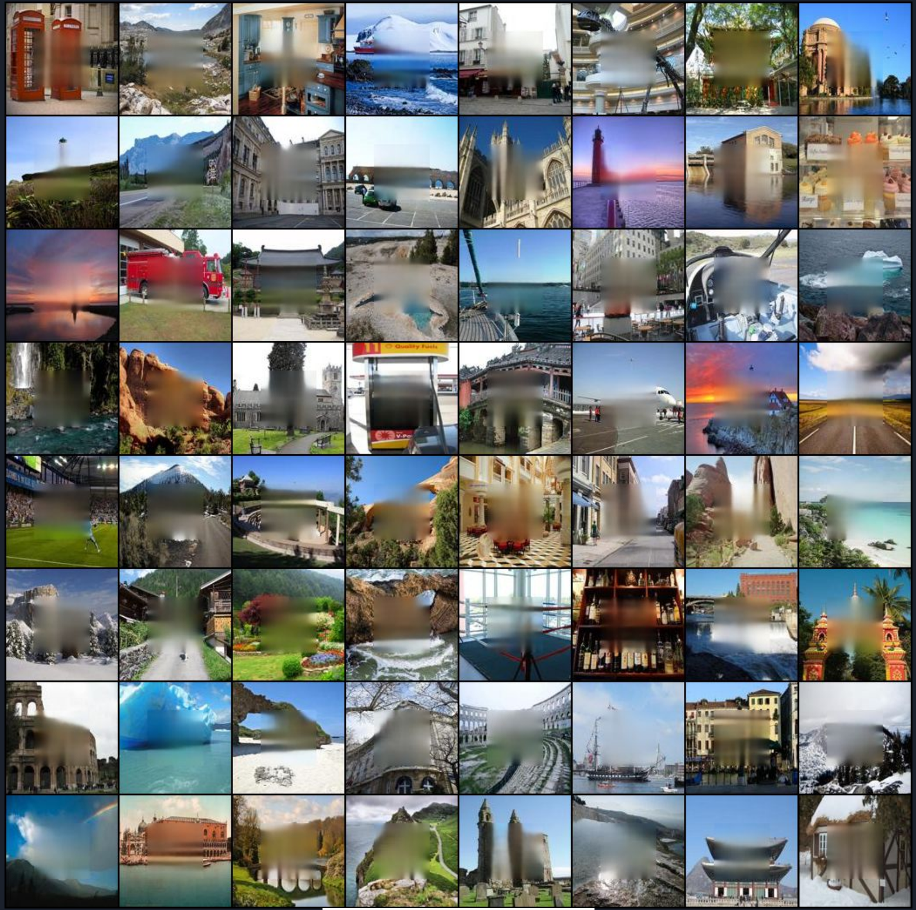


Input

Masked



Base



$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

# Baseline vs L2 Adversarial + L1 Reconstruction

Base

$$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$$

# Baseline vs L2 Adversarial + L1 Reconstruction

Base

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

# Baseline vs L2 Adversarial + L1 Reconstruction

Base

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

# Experiments

- What we want:
  - Bring back sharp edges, but create smoother results with less noise than some of the patches created by the baseline Context Encoders model.
- Proposed Solution:
  - Implement a Joint Reconstruction Loss specifically designed for visual appearance

$$\text{Reconstruction Loss: } \lambda_{rec1}\mathcal{L}_{rec1} + \lambda_{rec2}\mathcal{L}_{rec2}$$

- Use a factor of SSIM (or variations of SSIM):

$$\text{Loss: } \mathcal{L} = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}(\lambda_{L1}\mathcal{L}_{L1} + \lambda_{SSIM}(1 - \mathcal{L}_{SSIM}))$$

# Results

Base

L1: 0.0774
L2: 0.0386
PSNR: 14.1375
SSIM: 0.7189

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0698
L2: 0.0347
PSNR: 14.5935
SSIM: 0.7403

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0734
L2: 0.0396
PSNR: 14.0252
SSIM: 0.7287

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0744
L2: 0.0400
PSNR: 13.9811
SSIM: 0.7345

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.0844
L2: 0.0552
PSNR: 12.5826
SSIM: 0.7232

# Results

Base

L1: 0.0551
L2: 0.0199
PSNR: 17.0017
SSIM: 0.7063

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0511
L2: 0.0180
PSNR: 17.4431
SSIM: 0.7164

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0562
L2: 0.0221
PSNR: 16.5612
SSIM: 0.7139

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0560
L2: 0.0213
PSNR: 16.7069
SSIM: 0.7128

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.0661
L2: 0.0298
PSNR: 15.2601
SSIM: 0.7060

# Results

Base

L1: 0.0509
L2: 0.0209
PSNR: 16.8047
SSIM: 0.7701

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0415
L2: 0.0181
PSNR: 17.4288
SSIM: 0.8060

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0507
L2: 0.0224
PSNR: 16.4965
SSIM: 0.7920

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0448
L2: 0.0205
PSNR: 16.8840
SSIM: 0.8065

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.0522
L2: 0.0290
PSNR: 15.3750
SSIM: 0.7975

# Results

Base

L1: 0.0691
L2: 0.0296
PSNR: 15.2889
SSIM: 0.7111

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0648
L2: 0.0268
PSNR: 15.7234
SSIM: 0.7194

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0640
L2: 0.0279
PSNR: 15.5403
SSIM: 0.7256

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0669
L2: 0.0289
PSNR: 15.3979
SSIM: 0.7197

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.0789
L2: 0.0415
PSNR: 13.8219
SSIM: 0.7187

# Results

Base

L1: 0.0652
L2: 0.0277
PSNR: 15.5769
SSIM: 0.7183

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0596
L2: 0.0244
PSNR: 16.1256
SSIM: 0.7237

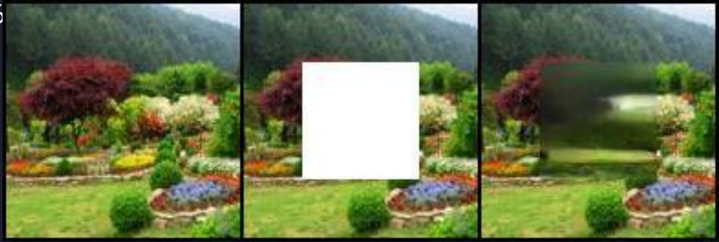
$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0613
L2: 0.0277
PSNR: 15.5784
SSIM: 0.7278

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0607
L2: 0.0260
PSNR: 15.8559
SSIM: 0.7234

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$



L1: 0.0662
L2: 0.0313
PSNR: 15.0510
SSIM: 0.7267

# Results

Base

L1: 0.0740
L2: 0.0358
PSNR: 14.4571
SSIM: 0.7388

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0688
L2: 0.0389
PSNR: 14.1060
SSIM: 0.7421

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0724
L2: 0.0417
PSNR: 13.8002
SSIM: 0.7382

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0734
L2: 0.0464
PSNR: 13.3350
SSIM: 0.7479

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.0811
L2: 0.0553
PSNR: 12.5762
SSIM: 0.7232

# Results

Base

L1: 0.0783
L2: 0.0360
PSNR: 14.4411
SSIM: 0.7140

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0846
L2: 0.0468
PSNR: 13.2932
SSIM: 0.7185

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0941
L2: 0.0572
PSNR: 12.4256
SSIM: 0.7129

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0859
L2: 0.0474
PSNR: 13.2443
SSIM: 0.7118

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.1020
L2: 0.0716
PSNR: 11.4505
SSIM: 0.7136

# Results

Base

L1: 0.0954
L2: 0.0521
PSNR: 12.8282
SSIM: 0.7108

$\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$

L1: 0.0918
L2: 0.0516
PSNR: 12.8775
SSIM: 0.7213

$\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$

L1: 0.0962
L2: 0.0609
PSNR: 12.1527
SSIM: 0.7269

$\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$

L1: 0.0915
L2: 0.0543
PSNR: 12.6535
SSIM: 0.7365

$\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$

L1: 0.1137
L2: 0.0873
PSNR: 10.5899
SSIM: 0.7164

very high SSIM in combination with adversarial loss tends to give strange results or artifacts

# Evaluation

- Each model was trained for 40 epochs.
- Models are evaluated on the MiniPlaces validation dataset containing 10,000 128x128 images from 100 different scene categories.
- All methods, except the second, use BCE adversarial loss.

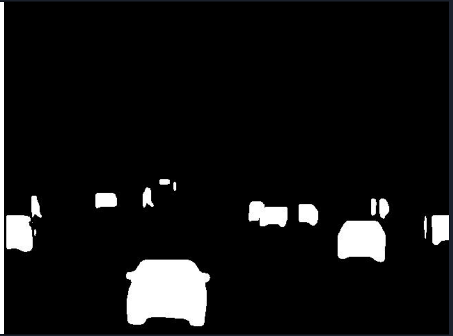
| Model | Mean L1 Loss | Mean L2 Loss | Mean PSNR | Mean SSIM |
|---|---|---|---|---|
| Base Context Encoders | 0.0653 | 0.0319 | 15.7722 | 0.7385 |
| $\mathcal{L}_{adv} = \mathcal{L}_{L2}, \mathcal{L}_{rec} = \mathcal{L}_{L1}$ | **0.0619** | **0.0319** | **15.9358** | **0.7538** |
| $\mathcal{L}_{rec} = 0.3\mathcal{L}_{L1} + 0.7\mathcal{L}_{SSIM}$ | 0.0651 | 0.0366 | 15.3956 | 0.7504 |
| $\mathcal{L}_{rec} = 0.5\mathcal{L}_{L1} + 0.5\mathcal{L}_{SSIM}$ | 0.0645 | 0.0363 | 15.4440 | 0.7512 |
| $\mathcal{L}_{rec} = 0.005\mathcal{L}_{L1} + 0.995\mathcal{L}_{SSIM}$ | 0.0740 | 0.0485 | 14.3107 | 0.7441 |

# Object Removal – Mask Extraction

Input

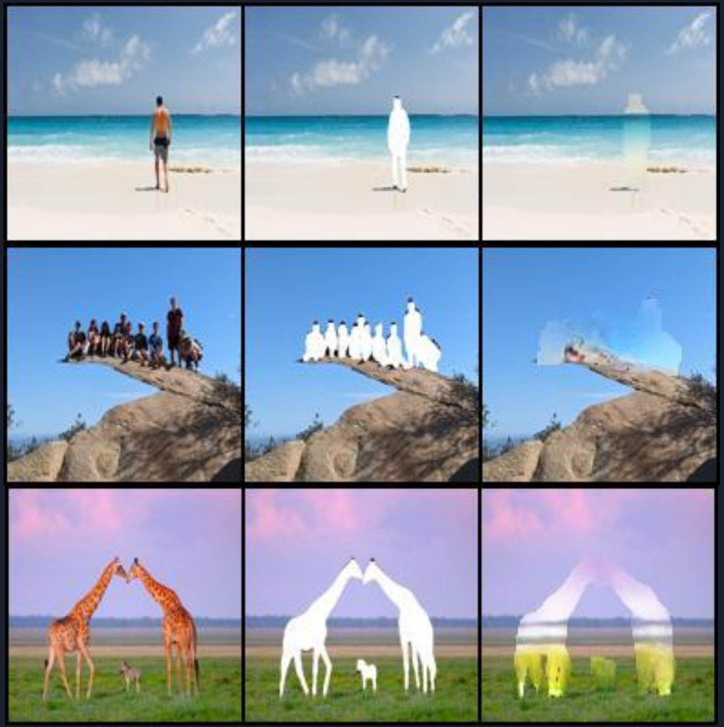
Predictions


Mask



Extracting Mask of People and Cars

# Automatic Object Removal Examples



- Note: The model is trained 12.5x less than in the Context Encoders paper, so the inpainting results are not perfect and object outlines can still be seen.

# Object Removal Limitations

- Problem:  Removal with arbitrary masks sometimes gives strange artifacts or inpaints a near solid color (especially on small objects).
- Potential Solution: Implement random masking when training the model.

# Conclusion

- Modified the baseline Context Encoders model to support evaluation with arbitrary shaped masks.
- Implemented the below joint reconstruction loss to significantly improve visual results of the Context Encoders model.

$$\text{Loss: } \mathcal{L} = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}(\lambda_{L1}\mathcal{L}_{L1} + \lambda_{SSIM}(1 - \mathcal{L}_{SSIM}))$$

- Integrated image inpainting with object detection and segmentation to automatically remove objects from an image.

# References


[1] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. "Context Encoders: Feature Learning by Inpainting". In: *Computer Vision and Pattern Recognition (CVPR)*. 2016.

[2] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. "Loss Functions for Image Restoration With Neural Networks". In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 47–57.

[3] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).