# Landscape inpainting with convolutional autoencoders

Alexander Viala Bellander
*Department of Computer Science and Engineering*
*Chalmers University of Technology*
Gothenburg, Sweden
viala@chalmers.se

Edwin Holst
*Department of Computer Science and Engineering*
*Chalmers University of Technology*
Gothenburg, Sweden
edwinh@chalmers.se

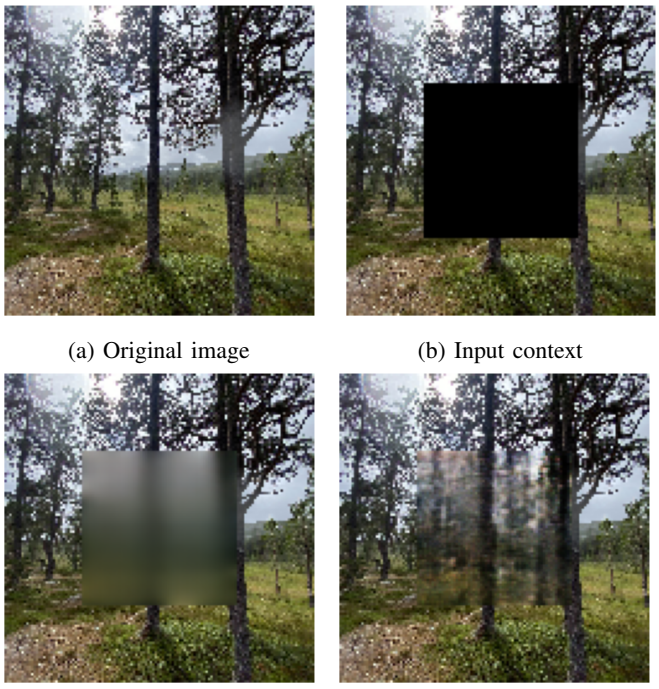
***Abstract*—We present a traditional unsupervised visual feature learning algorithm driven by context-based pixel prediction using Convolutional Autoencoders (CAE). We attempt to reproduce the original paper *Context Encoders: Feature Learning by Inpainting* by Pathak et al. We reconstruct, to the best of our ability, a variant of the original Context Encoder – a convolutional neural network trained to generate the contents of a masked region, conditioned on its surroundings. Similar to the original paper, when training the context encoder, we experimented with both standard-pixel-wise reconstruction loss, as well an adversarial loss using a traditional discriminator generator setup. The combination of the two loss methods produces much sharper results than either one alone. We found that we were able to reproduce a context encoder that learns a representation that captures not just appearance but also the semantics of visual structures. We quantitatively and qualitatively demonstrate the capabilities of the context encoder as an inpainting method.**

## I. Introduction

Denoising images is an active topic within machine learning [1]. But changing the context from removing local noise into recreating larger chunks of an image changes both the meaning and the usefulness of such a model. Image inpainting, and in this task specifically; semantic hole-filling, is when you try to replace a large portion of an image, such as a missing square region. This may seem like a trivial task for a human, but to do it effectively, we need a semantic understanding of the rest of the image. This is the main difference to denoising images, since denoising has previously relied on mainly local features [2]. Image inpainting is therefore both a useful tool in itself, but can also be seen as a training task to create a semantically meaningful understanding of images. This is what is explored in [2], which reached state of the art performance in image inpainting at the time of publication. Our project seeks to investigate and reproduce the method of image inpainting presented in the paper.

## II. Related work

Pathak et al. presented state of the art performance in image inpainting, achieving remarkable reconstruction results, at the time of publication [2]. They put the hole filling task into an encoder-decoder pipeline utilising a pixel-wise and adversarial loss. Their approach leverages the powerful feature learning function of convolutional neural networks (CNN). Variations of GAN with curriculum learning, progressive generative networks (PGN) have later demonstrated superior results in semantic inpainting where the hole filling process is split into several different phases [3]. There have also been attempts with patch-based methods that fill in the missing region by searching for well-matching replacement patches (i.e., candidate patches) in the undamaged part of the image. For instance, Alilou et al. [4] uses singular value decomposition (SVD) and an approximation matrix to reconstruct the missing regions which produces highly detailed textural results. A literature review for imagine inapinting published in 2020 summarises this problems past history well [5]. In recent years we have seen an increasing usage of powerful



(a) Original image (b) Input context

(c) Context Encoder ($L2$ loss) (d) Context Encoder ($L2$ + Adversarial loss)

Fig. 1: Qualitative illustration of the task. Given an image with a missing region (a), our generative model is given the input (b). Automatic inpainting using our *context encoder* trained with $L2$ reconstruction loss is shown in (c), and using both $L2$ and adversarial losses in (d).

large scale transformers which have also successfully been applied to semantic image inpainting [6]. Lastly, we have Zero-Shot Text-to-Image Generation which can be seen as the final boss to the original semantic inpainting problem, where almost any context can even be supplied [7] and generated with remarkable results.

Since this study is an attempt to reconstruct [2], a more detailed and field oriented background to related work can be found in the original paper.

## III. The Context Encoder

We now introduce landscape inpainting using Context Encoders: A CAE that predict the content of a missing region of a scene from their surroundings. Initially, we present our reproducibility methods including any limitations applied, then we introduce the Context Encoder with its general architecture and learning procedure.

### A. Context Encoders for semantic hole-filling

The architecture can generally be described as a traditional CAE; an encoder-decoder pipeline where the encoder takes an input image, see Figure 1b, with a large region missing and produces a latent feature representation of the image context, then the decoder takes this encoded feature representation and attempts to fill the missing region in the image (inpainting). Figure 2 described an overview of our architecture for the context encoder.

*1) Encoder:* Our encoder is based on the encoder used in the original context encoder paper [2]. Given input images of size $128 \times 128$ we use a set of convolutional layers without pooling to compute a latent feature space of size $1 \times 1 \times 4000$. In contrast to the original paper, where Pathak. et al. experimented with *AlexNet* architecture, our solution meerly replicates their simpler architecture of 6 downsampling convolutional layers and 5 upsampling convolutional layers. However, similar to [2] our network is also trained for context prediction "from scratch" with randomly initialized weights. In our architecture this information propagation is handled by a fully connected layer whilst [2] used channel-wise fully connected layers instead to reduce the number of trainable parameters.

*2) Decoder:* Here we describe the second part of the context encoder pipeline, the decoder, which generates pixels using the encoded features. The "encoder features" are connected to the "decoder features" using a fully connected layer with 65M trainable parameters (92% of the entire model). The encoded feature space is upscaled with transposed convolution layers from $1 \times 1 \times 4000$ to the predetermined patch size of $64 \times 64$. Transposed convolution is a non-linear weighted upsampling of the feature produced by the encoder until we roughly reach the original target size. It can be understood as an upsampling followed by convolution, or convolution with fractional stride (as described in [8] and [9]).

### B. Loss function

We train our context encoders by regressing to the ground truth content of the missing (masked out) region. However,

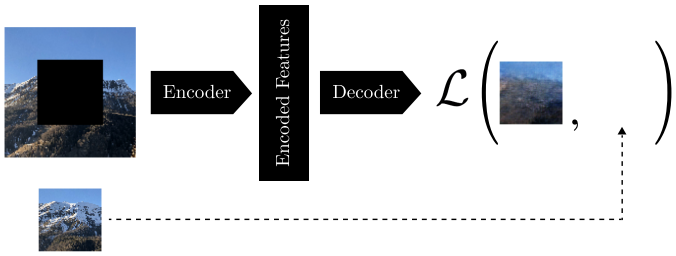



Fig. 2: Context Encoder. The input image (the context) is passed through the encoder which obtains the relevant features which the decoder uses to fill the missing region in the image. The loss is a function of the generated patch and the original patch.

there are often multiple variations of content that can fill the masked region that are consistent with the surrounding context. We model our loss function as a joint problem. The reconstruction (L2) loss is responsible for capturing the overall structure of the missing region and ensures coherence with regards to its context. However, a side effect is that it tends to average the different modes in the underlying prediction distribution. This results in blurry images that are often semantically accurate, however, with very little detail.

We use an adverserial loss function which, in contrast to the L2 loss, tries to make the prediction look real.

*1) Reconstruction loss:* Much like in [2] we use a normalised masked L2 distance as our reconstruction loss function, $\mathcal{L}_{rec}$,

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

where $\odot$ is the element-wise product operation. In contrast to [2] we only train our models with L2 loss. There were reportedly no significant difference between L1 and L2 loss [2]. While this simple loss method encourages the decoder to produce a rough prediction of the masked region, it remains insufficient as a loss method with this setup for the decoder (generator) to produce and capture any high frequency detail (see Figure 1c). We believe this happens because predicting a blurry solution over an highly accurate solution is "safer" for the L2 loss. Predicting the mean of some underlying distribution likely minimises the mean pixel-wise error but results in a blurry averaged image. We alleviated the blur problem by adding an adversarial loss.

*2) Adverserial loss:* Our adversarial loss is based on Generative Adversarial Networks (GAN) [10]. To learn a generative model $G$, GAN models jointly learn an adversarial discriminative model $D$ to provide loss gradients to the generative model. The learning process can be seen as a two-player game where the generator $G$ attempts to fool the discriminator $D$, which for ground truth image $x$ learns to differentiate from generated images $G(x)$ and $x$. The objective function for discriminator is logistic likelihood. Thus, the discriminator tries to predict whether an image is real or generated.

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

Fig. 3: Semantic Inpainting results on *held-out* images. The last two sets of images on the last row shows one very complicated scene (airplane) and the last image shows a trivial inpaint task (chessboard) for humans but for the model rather difficult.

This method has shown encouraging results in a variety of generative applications over the last few years [11], [12]. We do not condition the discriminator on a noise vector since the model in [2] performed better without. The loss function for the adverserial network: $\mathcal{L}_{adv}$, is

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

where, $F$ and $D$ are optimised using alternating SGD. Note that this objective function differs from our implementation since this encourages the entire output of the context encoder to look realistic, not just the missing regions. Our discriminative model is given an image of size $78 \times 78$, that is the original patch size with 7 pixel padding. The idea is that the discriminator will encourage smooth edges over hard edges when looking at the original context + the generated content.

*3) Joint loss:* We define the overall loss function as

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv} \tag{3}$$

where $\lambda_{rec}, \lambda_{adv}$ are scaling parameters. We used the same values as in [2]

### C. Implementation details

We have used the machine learning framework PyTorch with 15K training images and 150 test images of various landscape images (mountains, forest, sea, etc.). Our solution is limited to landscape inpainting. The models were trained using a Nvidia A100 40GB GPU with a learning rate of $1e-4$ for the generator and $1e-5$ for the discriminator, both using the Adam optimiser, for 100 epochs. The training took roughly 30 seconds per epoch. The original intent was to create a grey-scale model in order to make the model smaller, but all colour channels was used in the final model, as discussed in the following chapter. The final architecture of the model can be seen in figure 4.

## IV. Results & Discussion

We now present our two implementations of the model, with some example generations of our model as well as a numerical comparison to the original paper. However, it is important to note that there is no perfect metric for image inpainting, as a complement to our presented metrics, we encourage the reader to draw their own conclusions from looking at our presented images and by trying the model.

### A. Gray-scale versus colour

Due the small time frame of this project, our first implementation of the context encoder was limited to grey scale images. The idea behind this was to decrease the complexity of the model, and speed up training time by making a smaller model. However, the resulting images of the grey scale model were in our opinion significantly worse than that of our coloured version, so this simplification was not used in our final model. It is unclear why this would be the case, but a theory is that it could either be because of a lack of optimising hyper parameters, or that the colours themselves make it easier for the model to understand the semantics of the scene, due to semantically similar object sharing similar colours.

### B. Quantitative results

There are some useful metrics when evaluation image inpainting, mainly mean L1 and L2 pixel loss, PSNR, and SSIM [2]. We used the same metrics as in Pathal et al.'s paper in addition to SSIM. These values are found in Table I, along with the results from the original paper as a reference [2]. From the table, we see that our models almost perform up to par with the original author's. Note however that this is not a fair comparison, since we compare the model performance on vastly different data sets, with our images probably being more homogeneous. However, one conclusion that can be drawn is that context encoders seem to be able to perform well on an "easier" dataset, with less training.

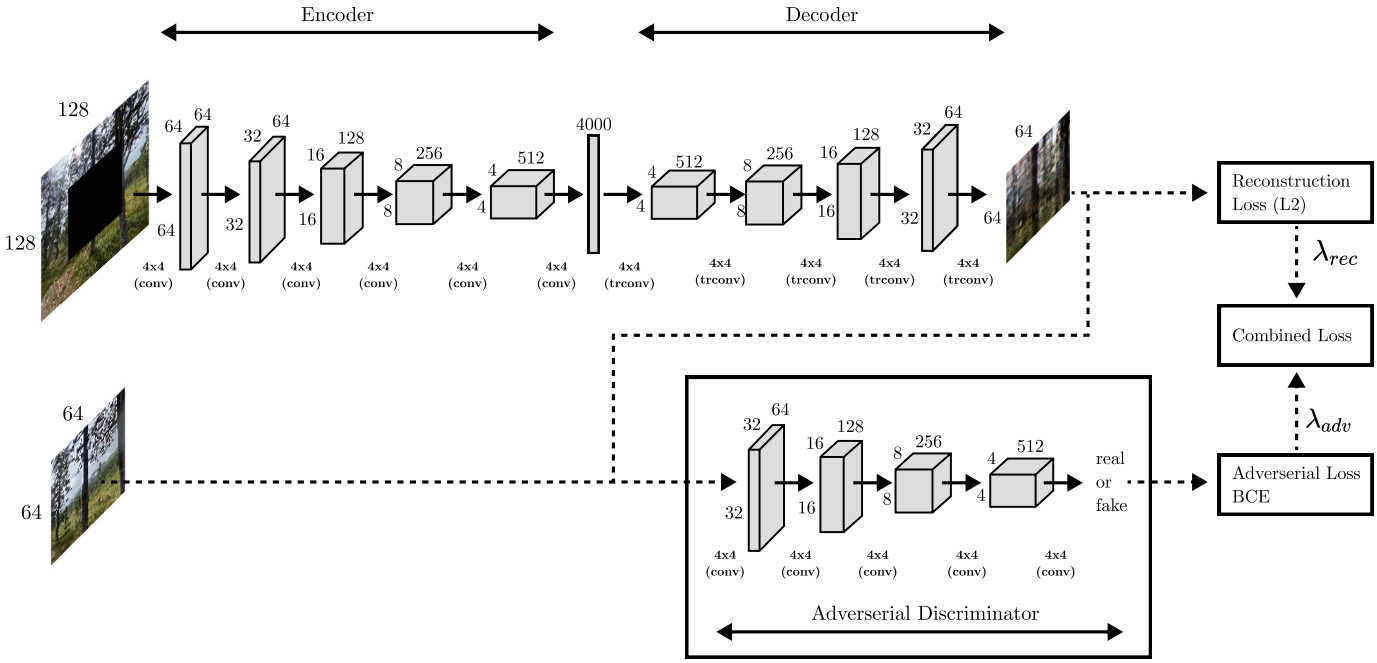



Fig. 4: Architectural overview of the Context encoder trained with joint reconstruction and adversarial loss for semantic inpainting with central region dropout.

TABLE I: Evaluation metrics

| Method | Mean L1 Loss | Mean L2 Loss | PSNR | SSIM |
|---|---|---|---|---|
| Pathak et al. | 15.10% | 4.30% | 14.70 dB | - |
| Ours, B&W | 20.25% | 32.43% | 12.95 dB | 11.1% |
| Ours, Colour | **18.16%** | **29.60%** | **13.43 dB** | **15.4%** |

### C. Visual results

The visual results is presented in 3. Drawing hard conclusions from these images is hard, but we would argue that it is quite obvious that our model has been able to draw atleast some semantic meaning from the context. This is most obvious in cases with horizons or similar features. It is also obvious in 1d where the model seems to figure out that we should construct a tree like object with a trunk in the middle. It also seems to have a more difficult time reconstruction objects, such as the airplane, however, this is expected since the training images was specifically selected to focused more on landscapes and less on different object.

## V. Conclusion

We succeed in creating an context-encoder model, as presented in the paper by Pathal et al [2], and trained it to generate landscape images conditioned on context. The model is somewhat successful in the task of inpainting, but perhaps more importantly seems to be able to interpret some semantic meaning of the images. This is not a state of the art inpainting model anymore, but it could still be considered as an unsupervised training task, useful where semantic context is key.

## References


[1] M. M. Thomas, G. Liktor, C. Peters, S. Kim, K. Vaidyanathan, and A. G. Forbes, "Temporally stable real-time joint neural denoising and supersampling," Intel Corporation, Tech. Rep., 2022 [Online]. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/temporally-stable-denoising-and-supersampling.html

[2] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," *CoRR*, vol. abs/1604.07379, 2016. [Online]. Available: http://arxiv.org/abs/1604.07379

[3] H. Zhang, Z. Hu, C. Luo, W. Zuo, and M. Wang, "Semantic image inpainting with progressive generative networks," *ACM international conference on Multimedia*, vol. MM'18, 2018. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3240508.3240625

[4] V. K Alilou and F. Yaghmaee, "Exemplar-based image inpainting using svd-based approximation matrix and multi-scale analysis," *Multimedia Tools and Applications*, vol. 76, no. 5, pp. 7213–7234, 2017.

[5] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image inpainting: A review," *Neural Processing Letters*, vol. 51, no. 2, pp. 2007–2028, 2020.

[6] Y. Yu, F. Zhan, R. Wu, J. Pan, K. Cui, S. Lu, F. Ma, X. Xie, and C. Miao, "Diverse image inpainting with bidirectional and autoregressive transformers," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 69–78.

[7] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.

[8] A. Dosovitskiy, J. T. Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," *CoRR*, vol. abs/1411.5928, 2014. [Online]. Available: http://arxiv.org/abs/1411.5928

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[11] S. C. Alec Radford, Luke Metz, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1511.06434

[12] M. Liu, X. Huang, J. Yu, T. Wang, and A. Mallya, "Generative adversarial networks for image and video synthesis: Algorithms and applications," *CoRR*, vol. abs/2008.02793, 2020. [Online]. Available: https://arxiv.org/abs/2008.02793