Ablative Image Compression

Phillip Kuznetsov
Machine Learning at Berkeley
philkuz@ml.berkeley.edu
Utkarsh Singhal
Machine Learning at Berkeley
utkarsh@ml.berkeley.edu

Abstract

We propose a method to apply Generative Adversarial Networks to image compression. The proposed method removes significant portions of an image while retaining some assistant information, and fills the gaps using generative model inpainting. We use Plug and Play Generative Networks as our inpainting framework, and explore several different ablation schemes in order to determine the most useful information present in an image. Finally, we demonstrate the performance of this method in comparison to JPEG.

1 Introduction

With the rise in popularity of neural networks there have been numerous works pushing the boundaries of image compression. These methods beat both the quality and compression ratio of JPEG. However, many of these approaches focus on making the compression perceptually unnoticeable. In reality, not all details are equally important and there are often whole regions where the details don't need to be preserved well. Even though there are methods that allow for spatially varying compression rates, they don't target specific features of an image to degrade; instead they degrade the overall quality of the image.

Ideally, a compression algorithm would be able to selectively ignore different kinds of details in different parts of the image, thus optimizing compression rate for the given application. We call this approach Ablative Image Compression. In this project, most of our focus is on approaching Ablative Image Compression through Image inpainting.

A previously popular method for weighted compression utilized image inpainting [5] (figure 1). The method removes parts of the image, compresses the leftover portions of the image while encoding - and infers the removed portion of the image using previously saved 'assistant information'. By varying the types and amount of assistant information retained, the algorithm can be adapted to achieve Ablative Compression.

We combine this result with the recent inpainting success of Plug and Play Generative Networks [7]. PPGN utilizes conditional iterative generation to ascend the natural prior gradient.

We utilize PPGN to power our image compression algorithm. PPGN is particularly adept at inpainting relatively large missing portions of an image as shown in figure 2.

2 Related Work

Over the last few years, neural networks have shown great promise for image compression through transform coding. Transform coding involves transforming an image into a domain where its code is easily compressible, and since neural networks are able to learn non-linear transformations (as opposed to linear transformations used in JPEG), these architectures can be used to compress images at any point on the rate-distortion curve, with spatially-variant fidelity for content-weighted compression [1],[11],[10]. These networks are more flexible than JPEG, and they surpass JPEG in both compression ratio and image quality, making them ideal for compression and super-resolution tasks.

On the other end of the spectrum, analytical methods like [2],[5] explore a completely different approach from transform-coding. These methods retain a small part of the image, and exploit the spatial redundancy and structure of natural images to reconstruct the discarded part through inpainting. These methods show great promise, especially for cartoon-like images [2] or images with low texture detail, but for more complex images, the reconstruction quality suffers, and finding the best pixels to retain becomes computationally intractable. To deal with this intractibility, approaches like [5] preserve pixels near edges to retain structure, and some pixels away from edges to retain texture. With this approach, reconstruction quality and compression ratio are highly dependent on the quality of the image prior used in the inpainting method. The traditional inpainting approaches like texture synthesis and exemplar-based inpainting use geometric properties of images as the prior, but neural inpainting approaches like [8],[7] show great potential by going one step further and learning more complicated image statistics, facilitated by recent developments in generative adversarial networks [4], [13]

3 Approach and Architecture

We explored numerous approaches to achieve ablative image compression.

3.1 PPGN

In order to re-create the results from [5] and be able to experiment with a varying degree of assistant information, we wanted to use a flexible network architecture that we would not have to re-train for every experiment. After

trying several approaches, we settled on Plug and Play Generative Networks [7] as our architecture (as demonstrated in figure

To compress our image, we use an approach similar to [5] and mask away the uninformative pixels on an image. The exact choice of pixels depends on the application, but in order to keep a fair comparison with [5] we mask the pixels that are distant from the edges and don't convey much structural or textural information.

Our image code thus descends the loss function

$$\mathcal{L}_{ppgn}(\vec{x}, \vec{p}) = \mathcal{L}_E(\vec{x}, \vec{p}) + \mathcal{L}_C(\vec{x}, \vec{p}) + \mathcal{L}_A(\vec{x})$$

where $\mathcal{L}_E$ is the autoencoder loss for the E network, $\mathcal{L}_A(\vec{x})$ is the assistant information loss, and $\mathcal{L}_C$ is the loss from the conditional network C. See figure 3 for network details.

3.1.1 Masking Experiments

For simplicity and demonstration of the technique, we first observed the image produced when we preserved a 100×100 pixel square in the centre of the image. We found that PPGN could readily fill in the details around the square itself. However, we also noticed that the borders of the square were oftentimes fairly apparent - something we drastically wanted to change.

In order to test the efficacy of the inpainting approach for compression, we tried reconstructing images with three varying levels of information: 1) Randomly chosen pixel values, 2) Pixel values at all the edges, and 3) Pixel blocks close to the edges and some randomly chosen pixel blocks far away from the edges.

The authors in [2] were able to achieve relatively good inpainting results using a sample of 1% of the pixels in an image and inpainting using PDEs based on the Heat Equation. In comparison, our reconstructions with randomly chosen 1% pixels were quite poor, even if the pixels were chosen to be close to the edges. This hints at the importance of the exact mask shape.

We also tried using just pixel blocks close to edges, and pixel blocks both close and far away from edges. In general, the number of pixel blocks did not seem to matter as much as their position.

3.1.2 Adding up loss functions.

We hypothesized that adding perceptual losses to our image would improve the reconstruction quality at the cost of compression ratio. We explored three different losses

1. Edge Loss - We utilized the Laplacian of the Gaussian to get the edges of the original image and took the L2 difference from the reconstructed image to determine the edge loss

$$\mathcal{L}_{edge}(\vec{p}, \vec{x}) = \|\textsc{Laplace}(\vec{x}) - \textsc{Laplace}(\vec{p})\|_2$$

2. Style Loss - borrowed from Image Style Transfer Using Convolutional Neural Networks [3]

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)$$

Where F^l and P^l are the filter responses of the original image x and p at a layer l in a CNN trained on ImageNet.

3. Content Loss - also borrowed from [3]

$$\mathcal{L}_{style}(\vec{p}, \vec{x}) = \frac{1}{2} \sum_{l=0}^{L} (w_l E_l)$$

where

$$E_l = \frac{1}{4N_l^2 M_l^2} \|G - A\|_{\mathcal{F}}$$

and G, A are the grammian self-similarity matrix of layer activations F_l and P_l respectively. In style loss, we choose several layers, weighted by the constant w_l, to use for self-similarity.

Our overall loss function was thus

$$\mathcal{L}(\vec{x}, \vec{p}) = \mathcal{L}_{ppgn} + \lambda_{edge}\mathcal{L}_{edge} + \lambda_{style}\mathcal{L}_{style} + \lambda_{content}\mathcal{L}_{style}$$

Where λ's were hyperparameter scalars that weighted the influence of each loss function to the overall loss.

4 Results

We needed a metric to compare our results with the state of the art, standard JPEG encoding. We chose to use SSIM [12] because of it's known correlations with human perception of genuine images, as well as it's simplicity to calculate. The results are tabulated in Table 1 As you can observe from the table, we were unable to beat the perceptual quality of JPEG compression, even at rates of 1% quality. There are certainly a number of reasons for this phenomenon. If you observe the image comparison in figure 4 you'll clearly see that the images generated using our method did not out compete JPEG's ability to handle Despite this, it should be noted that there are a number of artifacts that are a result of our sampling method as well as naive assumptions made by the GAN. This suggests that we can improve our method by a loss function around the edges of the masks, encouraging gradient descent to reduce the boundary between the masked image and the produced image.

Table 1: SSIM metric similarities. Entries with letters next to them have a corresponding image in Figure 4

Method	SSIM
JPEG 1% quality (a)	0.438
JPEG 5% quality	0.555
JPEG 25% quality	0.797
JPEG 50% quality	**0.864**
Pixel sampling (b)	0.126
Pixel sampling + style loss (c)	0.149
Pixel sampling + content loss (d)	0.156
Pixel sampling + edge loss (e)	0.154
Block sampling (e)	0.366
Block+Random sampling (f)	0.389
B+R sampling + style loss (g)	0.398
B+R sampling + content loss (h)	0.403
B+R sampling + edge loss (i)	0.399

5 Discussion and Future Work

Our experiments for assistant information efficacy agreed with the results as seen in analytical approaches for inpainting. Our reconstructions suffered from the GANs limited reconstruction capabilities, but we expect that to be fixed as we explore different architectures, training methods, and masks. We would also like to explore training feedforward networks as used in [8] for better reconstruction. Another direction for improving inpainting would be to look at perceptual losses such as that used in Deep Photo Style Transfer in addition to the edge loss and style transfer losses from this work.

6 Conclusion

We observed that the network was able to synthesize an approximation of the target image with spatially varying degrees of accuracy and efficiency. We found that using randomly chosen pixels as the data for reconstruction works, but using edge information, perceptual style loss, and similar assistant information increases the performance significantly. We also found that the position of pixels was much more important than their number, hinting at possible optimizations in future. In the long run, improvements in GAN technology will allow for higher accuracy and efficiency, thus allowing for compression of low-priority regions with much less data than conventional transform coding techniques.

References

[1] Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image com-

pression. *CoRR abs/1611.01704* (2016).

[2] Galić, I., Weickert, J., Welk, M., Bruhn, A., Belyaev, A., and Seidel, H.-P. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision 31*, 2 (2008), 255–269.

[3] Gatys, L. A., Ecker, A. S., and Bethge, M. A neural algorithm of artistic style. *CoRR abs/1508.06576* (2015).

[4] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. *ArXiv e-prints* (June 2014).

[5] Liu, D., Sun, X., Wu, F., Li, S., and Zhang, Y.-Q. Image Compression With Edge-Base Inpainting. *IEEE Transactions on Circuits and Systems for Video Technology 17* (2007).

[6] Luan, F., Paris, S., Shechtman, E., and Bala, K. Deep Photo Style Transfer. *ArXiv e-prints* (Mar. 2017).

[7] Nguyen, A., Yosinski, J., Bengio, Y., Dosovitskiy, A., and Clune, J. Plug play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint 1612.00005* (2016).

[8] Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. *CoRR abs/1604.07379* (2016).

[9] Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR abs/1511.06434* (2015).

[10] Toderici, G., O'Malley, S. M., Hwang, S. J., Vincent, D., Minnen, D., Baluja, S., Covell, M., and Sukthankar, R. Variable rate image compression with recurrent neural networks. *CoRR abs/1511.06085* (2015).

[11] Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., and Covell, M. Full resolution image compression with recurrent neural networks. *CoRR abs/1608.05148* (2016).

[12] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE 13* (2004).

[13] Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., and Metaxas, D. N. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR abs/1612.03242* (2016).

Appendix

StackGAN

We explored the StackGAN architecture [13], the PPGN architecutre [7], and finally adding style transfer losses [3] to the PPGN model loss.

StackGAN Architecture

One of the approaches we tried before settling on PPGN were conditional GANs like StackGAN. In StackGAN, the authors use two GAN architectures, layered on top of one another as demonstrated in figure 5. An image caption is fed through the first GAN, generating a low resolution and sometimes structurally ambiguous image. The next GAN is then conditioned with this low-resolution image, along with the original caption, and generates a high resolution image with near photo-like qualities.

We adapted this framework by removing the first layer of the network, and instead down sampling an image to the size of the first GAN's output. The results of this approach are located in figure 6.

The results of the approach were non-ideal. Although we were able to reduce the dimensions of an image by 1/4 (which we could then just use any compression method to preserve), the resulting images were oftentimes much further from the original image than we expected. This is likely because the generation of an image was heavily dependent upon the caption itself, as well as the stochasticity of the network caused by dropout.

Furthermore, the approach we were using was limited to specific domains, a trade off that enabled the GAN to generate photo-realistic images. We decided to explore other options.

Hyperparameters

We conducted a number of surveys on our hyperparameters of our original PPGN masking approach

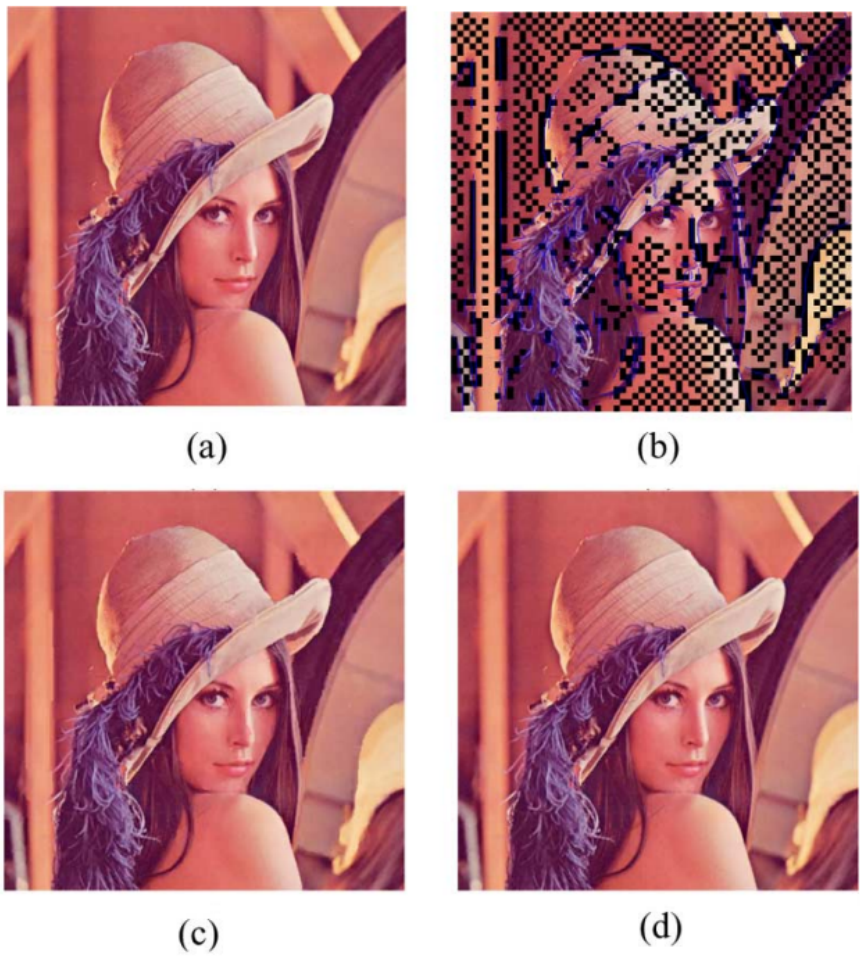

Figure 1: Inpainting from [5] (a) Original Image (b) Blocks removed for inpainting (c) Image after reconstruction (d) JPEG comparison

Figure 2: PPGN inpainting [7]

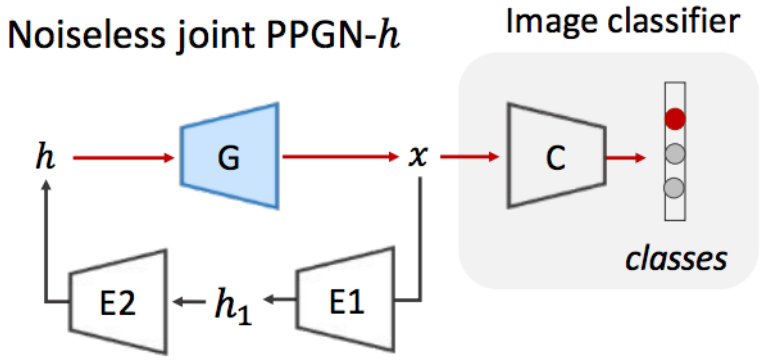

Figure 3: Noiseless joint PPGN-h architecture from [7]. x denotes an image, and h denotes the vector code representing that image. G transforms the h code into perceivable image space as x. C is the condition network - included as part of the overall optimization to ensure that the reconstructed image matches the code of the original image. $E1$ and $E2$ compose an encoding network. The h_1 vector represents the vector representing the image after pool5 of the network. We did not tinker with this code in our experiments. G is the generative component of the DCGAN architecture [9] and is pretrained on ImageNet. C and E are the same CaffeNet architecture (a modification of AlexNet).

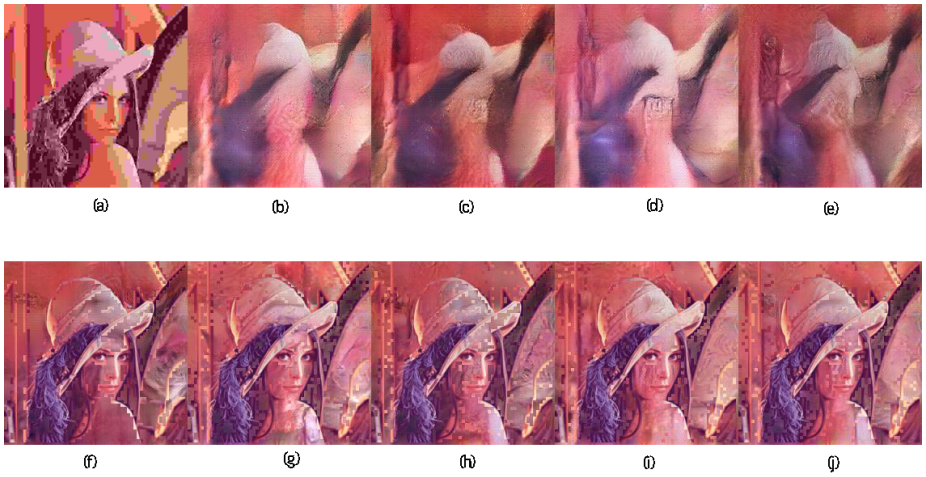

Figure 4: Image Quality demonstration. A guide for each compression/reconstruction method is listed in Table 1.

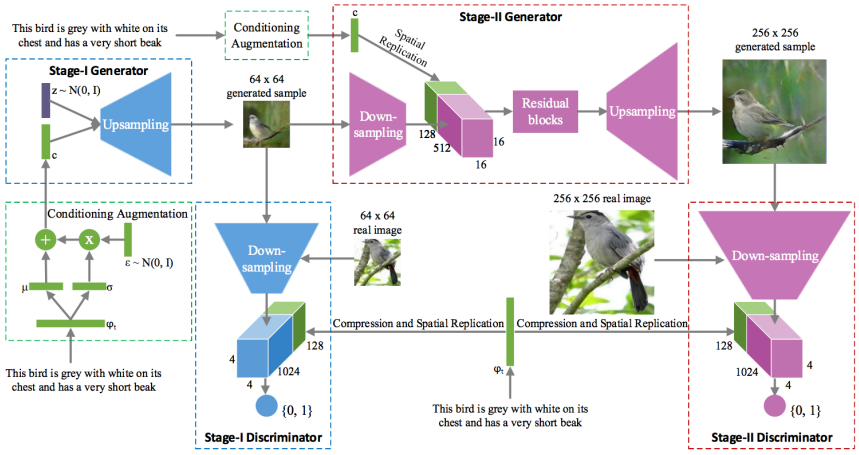

Figure 5: The StackGAN architecture from [13]

Figure 6: Reconstruction of an image using StackGAN. The input to the GAN are the down-sampled images in the row labelled 'Stage-I'. The image caption from the original dataset is displayed on top.

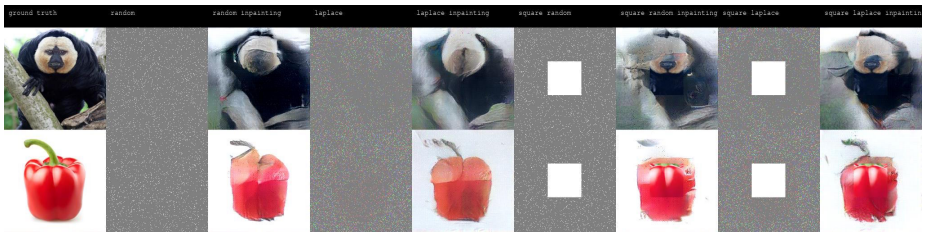

Figure 7: Example of different masks used for inpainting

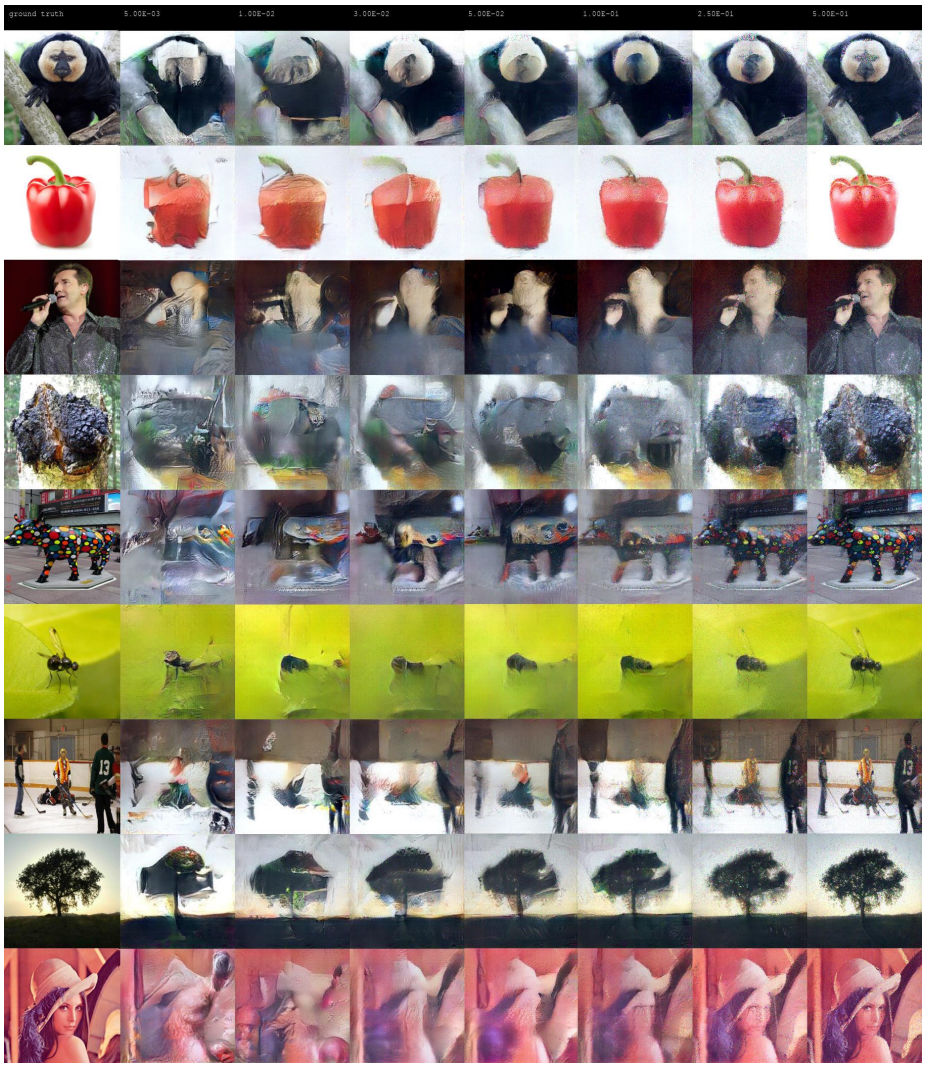

Figure 8: Inpainted results with different levels of image sampling

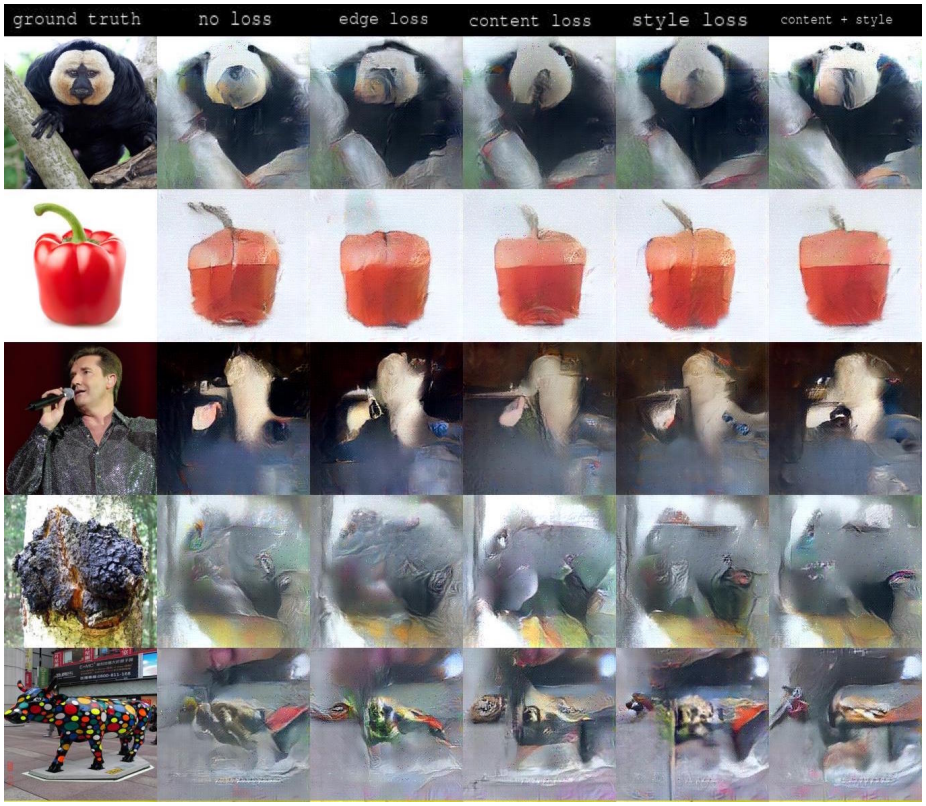

Figure 9: Demonstrations of different loss values