UPSCALING IMAGES USING A GENERATIVE ADVERSARIAL NETWORK

Martin ben Ahmed[1,2], Patrick Hoffmann[1,3], Sebastian Zielinski[1,4]

***Abstract*—New state of the art neural network architectures and ever increasing computational resources offer a large potential for new applications of deep learning in real life problems. In the context of images, GANs play a big role in creating new plausible sample data from a specified input. Amongst others, GANs can solve the problem of upscaling an image while increasing its level of detail, making the image look better when enlarging it. Upscaling is especially useful in context of video games, surveillance cameras or old data with a very low resolution. In this report we describe our approach on recreating a GAN for image super resolution, heavily inspired by Ledig et al. [1].**

I. MOTIVATION

With TVs and screens getting bigger and an ever increasing resolution the computitional effort to render images for these resolutions rises. Especially in the context of video games, powerful GPUs are needed to render images in high resolutions fast enough. While today most of these graphics cards are really expensive, most of the time they are sold out and it is hard to get a hand on one of them.

A solution to this problem is to render frames in a lower quality and upscale them to a bigger resolution in constant time using a neural network, typically utilizing a GAN architecture [2]. This upscaling is also reffered to as Super-Resolution. As a general upscaling would keep the level of detail the same and therefore visibly reduce the quality of the image when viewed on a larger screen with higher resolution, the GAN is supposed to increase the resolution while also increasing the level of detail, which is not possible with computer vision algorithms. The idea is, that the GAN creates an intuition on how to add detail to scenes during training.

This reduces the time to render frames drastically, which makes it possible to play games on high resolution without the need to buy the most powerful graphics card on the market. This technique also proves useful in many other szenarios. Using upscaling, images with low resolution (e.g. historical data like old photos or films) can be reprocessed, increasing its usability and asthetic. Further, data from surveillance cameras, which typically come at a low resolution, especially when they are older, can be made more useful, as details are increased. Therefore it can be also used to help in crime investigation, especially with further postprocessing steps, like face or object recognition.

Though there are many different approaches on Super-Resolution like using a ResNet architecture [3], a Deep Laplacian Pyramid Network [4] or a Sub-Pixel Convolutional Neural Network [5], we choose to use the GAN architecture proposed in [1], as it states to deliver extraordinary results. Even though GANs are usually harder to train than the other networks proposed in the previous paragraph, we decided to face the challenge of a training a GAN, as we wanted to examine wether it is worth the extra effort. In the following, we describe our recreation of this approach.

II. METHOD

A. Network architecture

The network architecture shown in figure 1 is also heavily inspired by Ledig et al. [1].
The generator starts by utilizing a convolutional layer activated by PReLU. After that, it is followed by a series of residual blocks, which aim to remember the input of the network. After another convolutional layer the input will be added to the current state by a skip connection spanning over all residual blocks. Finally, the network uses a series of upscaling blocks, consisting of convolutional and conv. transpose layers followed by PReLU activation to perform the actual upscaling process. Afterwards, another convolutional layer will be applied and generate the output image. The generator tries to recreate the full resolution image as exactly as possible from the low resolution image it gets as an input.

The discriminator starts with the same convolutional layers as the generator, activated by Leaky ReLU in-

[1]Institute of Computer Science, University of Osnabrück
[2]mbenahmed@uos.de [3]pahoffmann@uos.de [4]szielinski@uos.de

stead of PReLU. Afterward, it is followed by a series of convolution, batch normalization and Leaky ReLU activation blocks, varying in the number of channels and stride size. Finally, the images will run through a dense layer with the size of 1024, activated by Leaky ReLU and a dense layer of size 1, activated by Sigmoid. The discriminator then decides whether the input image was upscaled by the generator or is an original full resolution image.

All convolutional layers are described by a code consisting of k (kernel size), n (number of blocks) and s (stride size). Example: k9-n64-s1 describes a layer with a kernel size of 9x9, 64 channels, and a stride of 1.

B. *Loss Function*

The loss of the discriminator is defined as

$$L_D = L_{fake} + L_{real}$$

where L_{fake} is the binary cross entropy between the discriminator prediction and the ground truth for real images and where L_{real} is the binary cross entropy between the discriminator prediction and the ground truth for fake images.

The loss of the generator is defined as

$$L_G = L_{G_1} + L_{G_2}$$

where

$$L_{G_1} = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}(Y_{x,y} - \hat{Y}_{x,y})^2$$

is the pixel-wise Mean Square Error with Y being the original image and $\hat{Y}$ being the generated image. L_{G_2} is the loss of the discriminator.

We decided against a more complex approach as described by Ledig et al. [1], as it seemed to not fit the scope of the project in terms of complexity. However, we expect our results to be better with more sophisticated loss functions.

III. EXPERIMENTS

A. *Dataset*

The 2017 Unlabeled images dataset from the Common Objects in Context Dataset (COCO) by Microsoft [6] was used as the training dataset. This dataset consists of 123,000 unlabeled images. A dataset of a relatively smaller size has been chosen to minimize the training effort. The images are sourced from 80 object categories.

As the dataset consisted of both greyscale and RGB images, all non-RGB images were converted to RGB. This was done via a python script as a one-time operation.

B. *Training*

We trained our networks on a NVIDIA RTX 3090 GPU with 24 GB of memory. First of all, the paths of the dataset images were read and then shuffled with a large enough buffer size. After shuffling, the images were loaded and modified by flipping, cropping, darkening, changing the saturation and brightness with a certain probability. Finally, after the images were modified, we create full and low-resolution pairs by resizing the original images. We chose the number of upscaling blocks u = 2, which leads to an upscaling factor of $2^u = 4$. The low-resolution images with the size 64 by 64 pixels are upscaled to full resolution images with the size 256 by 256 pixels. The optimizer we chose was Adam with ß1 = 0.9, as proposed by Ledig et al. [1]. The learning rate was set to 10^{-4}, and we trained the network for 10^3 iterations. The number of residual blocks was set to b = 8 in order to decrease training effort. After encountering a memory problem that leads our GPU's memory usage to increase over time and eventually cause allocation errors, we restructured our training loop and chose to load and save our network after each training iteration. After each iteration, one test sample was upscaled by the network to visualize training progress. This was the only practical way to train the network for a decent number of iterations. One training iteration took approximately 800 seconds. The training of 1000 iterations took about 220 hours in total.

C. *Inference*

Since the generator network was trained with an input size of 64 by 64 pixels, upscaling larger images is not directly possible. For this reason, we implemented a program to split images into patches of size 64 by 64, to be able to run them through the generator. After each patch is scaled up to 256 by 256 pixels, the patches are set back together, resulting in an image that is four times larger then the input. Our git repository features a version of the trained network and the script to perform upscaling on any jpg images.

DISCUSSION

D. Results

After 1000 training iterations, the generated images show some of the criteria the upscaling process aims to fulfill. As seen in figure 2, all upscaled images appear a bit more clear and sharp and thus make details easier for the human eye to observe. In figure 2b, the nostrils of the baby are way more naturally shaped than in the low resolution image. In figure 2d, the patterns on the butterfly's wings are smoothed out. Nevertheless, all images are biased towards a specific color and some artifacts show around the edges of the images. Furthermore, the edges are sharpened by a fair amount, but this could be improved further. The problem on the edges gets way more obvious if an image gets scaled up from patches, as seen in figure 3b. At the same time, especially the nose of the dog gets smoothed out and looks subjectively more pleasant.

CONCLUSION

TODO

REFERENCES

[1] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016.

[2] A. Watson, "Deep learning techniques for super-resolution in video games," *arXiv preprint arXiv:2012.09810*, 2020.

[3] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144, 2017.

[4] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 11, pp. 2599–2613, 2018.

[5] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016.

[6] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[7] M. Bevilacqua, A. Roumy, C. Guillemot, and M. line Alberi Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proceedings of the British Machine Vision Conference*, pp. 135.1–135.10, BMVA Press, 2012.

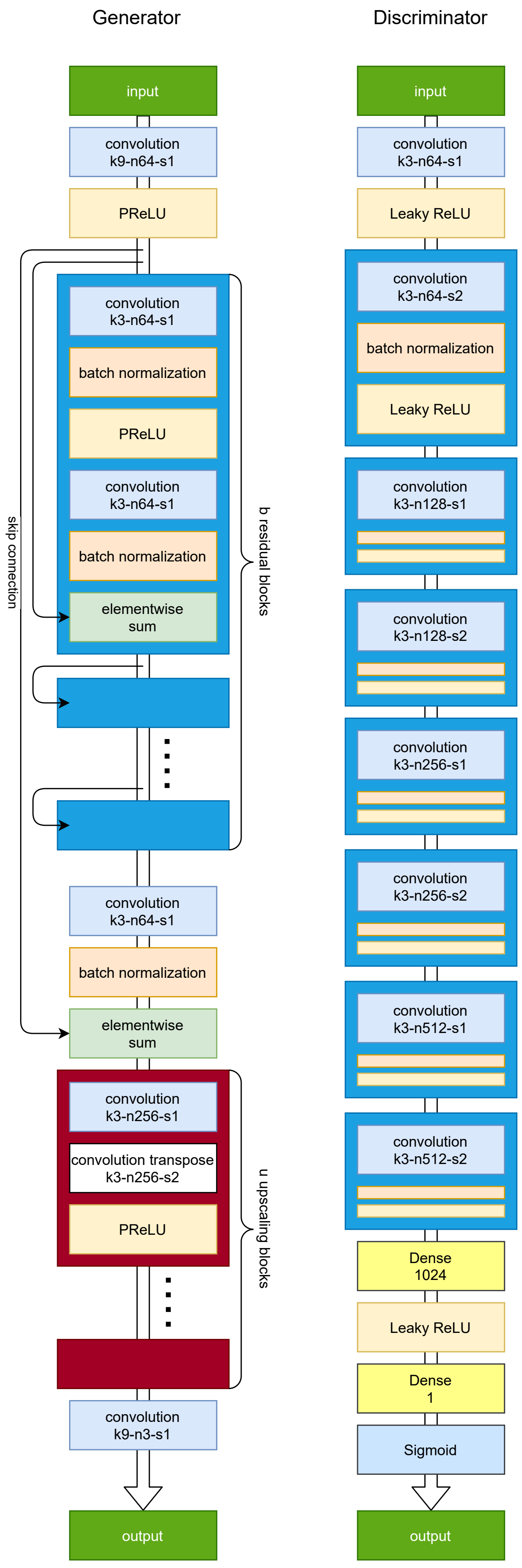
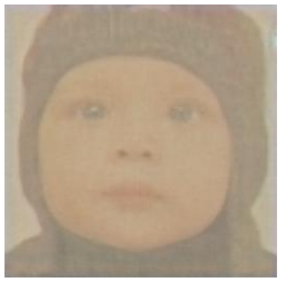
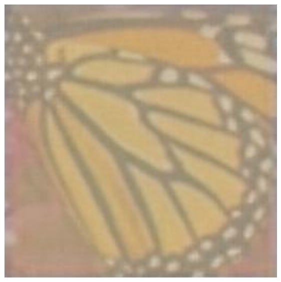


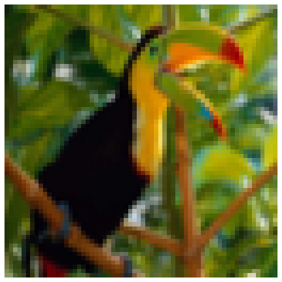
Fig. 1: Generator and Discriminator Network similar to Ledig et al. [1]

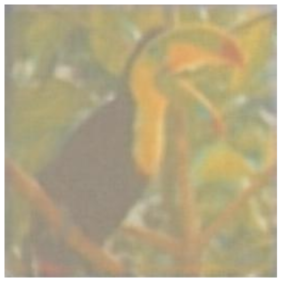
(a) Baby low

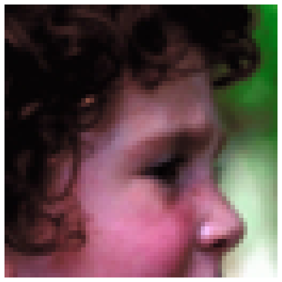

(b) Baby up

(c) Butterfly low


(d) Buttferfly up


(e) Bird low


(f) Bird up


(g) Head low

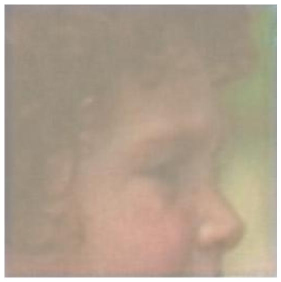
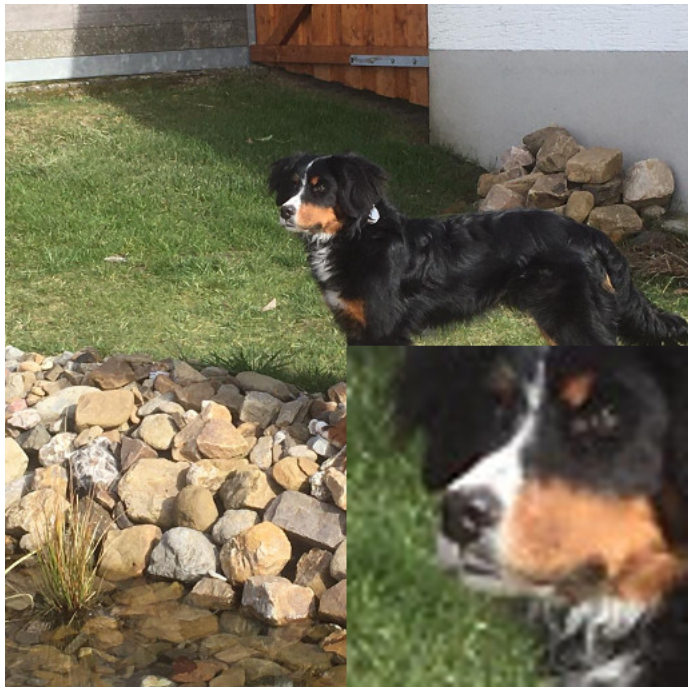

(h) Head up

Fig. 2: Sample images scaled up by the generator. Images by Bevilacque et al [7]


(a) Dog low

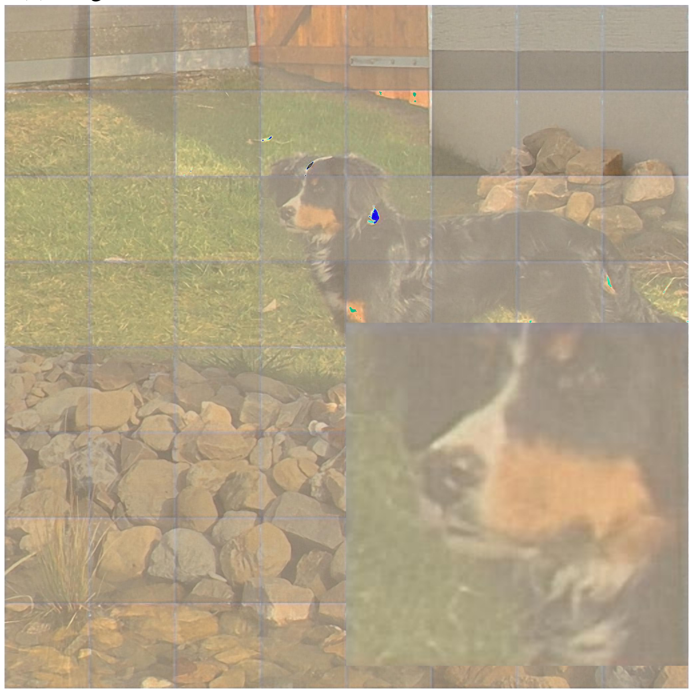

(b) Dog up

Fig. 3: Large image scaled up in patches.