Super-Resolution on Image and Video

Jason Liu
Stanford University
liujas00@stanford.edu

Max Spero
Stanford University
maxspero@stanford.edu

Allan Raventos
Stanford University
aravento@stanford.edu

Abstract

In this project, we explore image super-resolution using generative adversarial networks. Super-resolution is a problem that has been addressed using signal processing methods, but has only recently been tackled using deep learning, especially generative models. We start with a network inspired by Ledig et al [9], explore changes to the network and test our models on various datasets. We train models for both 2x and 4x upsampling and arrive at results that beat simple interpolations. Our 4x GAN is visually comparable to industry standards in deep learning, although trained on a dataset with comparatively few class labels. We also have preliminary results on super-resolution with video that are promising but computationally expensive.

1. Introduction

Image downscaling is an innately lossy process. No matter how good an upscaling algorithm is, there will always be some amount of high-frequency data lost from a downscale-upscale function performed on an image. Ultimately, even the best upscaling algorithms cannot effectively reconstruct data that does not exist. We propose a fix to certain situations where that problem appears, by using Generative Adversarial Networks (GANs) to hallucinate high-frequency data in a super-resolved image that does not exist in the smaller image. Using this method, we claim no perfectly accurate reconstruction of lost data, but rather a plausible guess at what this lost data might be, constrained to reality by a loss function penalizing deviations from the ground truth image.

As a motivating example, we consider the set of images in Figure 1. To produce the third image we low-pass filter the high resolution image, by zeroing out a center block in the 2D Fourier Transform. Similarly, for the fourth image we zero out the complement of the center block. We can see that the low resolution is actually quite similar to the low-pass filtered high resolution image. The high frequency portions of the Fourier Transform, in turn, mainly contain sharp edge detail. In essence, we want our GAN to generate the high frequency portions of the Fourier Transform that are lost through down-sampling.

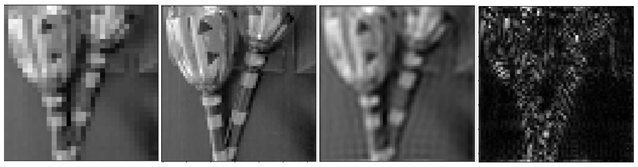


Figure 1. (a) low resolution 32x32, (b) high resolution 64x64, (c) 64x64 low-pass filtered, (d) 64x64 high-pass filtered; (a) and (b) from ImageNet

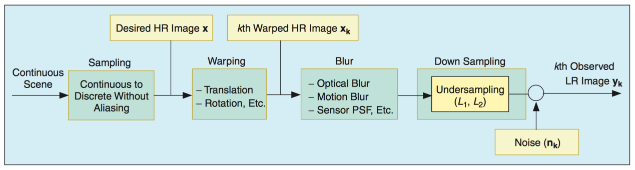
For the case of image super-resolution, the GAN takes in a low resolution 32x32 image and outputs a super-resolved 64x64 version of the image. Through our experiments with the 2x upsampling GAN, we create a GAN that produce 4x upsampling, from a 32x32 image to a 128x128 image.

2. Related Work

Super resolution is a task that had been addressed previously by methods other than deep learning. In their paper titled 'Super-Resolution Image Reconstruction: A Technical Overview'[13], Park et. al describe various reconstruction methods. These methods assume that one can model the transition from high-resolution to low-resolution using the sequence of transformations described in Figure 2. The first method Park et. al describe is the Nonuniform Interpolation Approach. [19] Its advantages lies in its fast computation, which makes real-time super resolution possible. However, this model assumes that all LR images were created with the same transformations. The second is a frequency domain approach, in where one can take the Discrete Fourier Transform of the low-resolution image and the continuous Fourier Transform of the high-resolution image. These properties make it possible to find a relationship between the aliased DFT coefficients of the observed LR images to a sample of the CFT of an unknown image. [18]. Another approach is the regularized SR Reconstruction approach. In this approach, one method was to construct a maximum-likelihood super-resolution image estimation problem to estimate subpixel shifts, the noise variance of each image, and the HR image simultaneously. [17]. This approach did well in staying flexible in regards to noise characteristics between the low-resolution and high-resolution images.

In 2014, Dong, et al. explored super resolution by using Convolutional Neural Networks, calling their network SR-CNN. [4]. In their paper, they explained how CNN methods had many similarities to sparse-coding-based SR methods [20]. Later, Kim, et. al improved upon SRCNN's result in their paper, titled Accurate Image Super-Resolution Using Very Deep Convolutional Networks" [7]. In their model, they took inspiration from VGG-net, a network that had been designed for image classification tasks. [16]

Generative Adversarial Networks (GANs) are relatively new models that were first proposed by Goodfellow, et al. [2] GANs are a form of unsupervised learning in which a generator model attempts to learn about the distribution of some true set, and a discriminator model learns to distinguish between data that is from the original set and data that is artificially created by the generator. Since then, a variety of work has been done towards improving and understanding GANs. Radford et. al developed DCGAN, which they demonstrated could learn general image representations. [14]. Mirza et. al developed Conditional GANs, in which one streams the data one wishes to condition on to both the generator and the discriminator. [12] Arjovsky, et. al introduced the Wasserstein GAN that changed the original loss function and thus improve the stability of learning. [1] [5]. Ledig et al., then applied GANs to super-resolution (SRGAN), using a network inspired by Res-net. [9] [6] SR-GAN works well for single image super-resolution as it also uses an intelligent content loss function that uses pre-trained VGG-net layers. However, Ledig et al, noted that further information could be used if this network were to be adapted to video, such as temporal information.



▲ *3. Observation model relating LR images to HR images.*

Figure 2. Diagram showing the transformations turning an image from high-res to low-res [13]

3. Methods

A generative network, G, is meant to learn the underlying distribution of a data set, Y. We can, for example, train a generative convolutional network on a dataset of face images to produce face images similar to those contained in the dataset. With just a generative network, however, we must then visually assess the quality of network outputs and judge how we can adapt the network to produce more convincing results.

With a discriminative network, D, we can incorporate this tweaking directly into training. The discriminative network takes in both fabricated inputs generated by G and inputs coming from the "real" dataset, Y. Its purpose is to classify whether the input comes from G or from Y. The key idea is that we can then backpropagate gradients from the results of D's classification to G, so that G gets better at producing fabricated images that can fool D.

For this project, we have data separated into two categories: Y, which contains 64x64 images, serving as labels for X, which contains 32x32 images downscaled from Y. G takes in an low resolution (LR) $x \in X$ and outputs $\hat{y}$, a super-resolved (SR) 64x64 version of x. The discriminator, in turn, takes in a 64x64 image and outputs the probability that the image comes from Y, instead of the set of outputs of G, G(X). As such, if the discriminator isnt being fooled by our SR images, it should generally output a probability larger than 0.5 for HR inputs coming from Y and a probability smaller than 0.5 for SR inputs coming from G(X).

We started by basing our GAN architecture off of SR-GAN, a recent paper [9] attempting to perform the same task as ours: image super-resolution. Their generator used residual blocks and a pixel-shuffle operation to upscale the input image. Their discriminator used many convolution layers with kernel size 3.

The generator and discriminator network architectures are reproduced in Figure 3 below.

The block consisting of convolution and pixel shuffler layers is responsible for upsampling the image. One of these blocks will upsample by a factor of 2 in both width and height dimensions, thus producing a 64x64 output from a 32x32 input. Adding a second one of these blocks will upsample the image by an additional factor of 2, resulting in a 128x128 output.

Similar to [9], the generator network is trained to minimize:

$$l^{SR} = l^{SR}_{C} + \beta l^{SR}_{Gen}$$

where we use the L2 distance between the Conv[4,5] features of $\hat{y}$ and y (features taken from a pre-trained VGG-19 network) for the content loss (left term). The adversarial

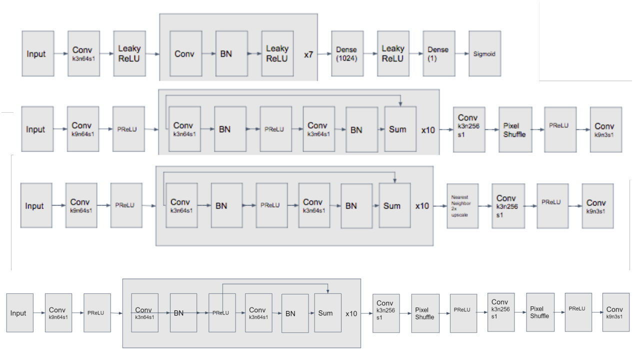


Figure 3. From top to bottom: Discriminator architecture, Generator architecture using pixel-shuffle layers, Generator architecture using resize-convolution layers, and Generator architecture for 4x super-resolution

loss (right term) is:

$$l^{SR}_{Gen} = \frac{1}{N}\sum_{i=1}^{N} H(1, D(G(x^{(i)})))$$

H is the cross-entropy, and 1 refers to a distribution that is identically 1 for all $x^{(i)}$. Cross-entropy reaches a minimum when the input distributions are identical. As such, G is trying to produce outputs $G(x^{(i)})$ to which the discriminator then assigns high probabilities of belonging to Y.

β represents the tradeoff between images that are more correct in the VGG-19 feature L2 sense and an images that are doing a better job at fooling the discriminator.

The discriminator, in turn, is trained to minimize:

$$l^{SR}_{Dis} = \frac{1}{N}\sum_{i=1}^{N}(H(0, D(G(x^{(i)})) + H(1, D(y^{(i)})))$$

where 0 is a distribution that is identically 0. That is, we want the discriminator to output high probabilities for images coming from Y, $y^{(i)}$ and low probabilities for images coming from $G(X)$, $G(x^{(i)})$.

These cross-entropy based losses are a variation on the approach used by [9], which uses, for example:

$$l^{SR}_{Gen} = \sum_{i=1}^{N} -\log(D(G(x^{(i)})))$$

We use cross-entropy based losses for numerical stability. The β hyperparameter is obtained through experimentation. [9] suggests 10^{-3}, but our adversarial losses, although having the same intention, produce different numerical values. We arrive at 10^{-2}. In later experiments we use the loss from Least-Squares GAN [11] (with the addition of content loss). Generator loss was:

$$\ell_G = \beta\frac{1}{2}\mathbb{E}_{z\sim p(z)}\left[(D(G(z))-1)^2\right] + l^{SR}_C$$

And discriminator loss:

$$\ell_D = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[(D(x)-1)^2\right] + \frac{1}{2}\mathbb{E}_{z\sim p(z)}\left[(D(G(z)))^2\right]$$

We use Adam optimizer [8] to update our parameters, because it generally requires comparatively less hyperparameter tuning than other methods. Adam stores the first and second moments of the gradients in order to converge to optima faster and not overshoot. We find it to be successful, starting with the default parameters, although we use 1/10 of the recommended learning rate to help make sure neither the generator nor the discriminator overshot, since the optimal target that each network trains towards changes with each iteration.

We implement these networks from scratch and train them by doing alternating steps of batch gradient descent on G and gradient descent on D. For a batch of size m, we do gradient descent using m inputs from X on G, and gradient descent using m inputs from G(X) and m inputs from Y on D.

4. Dataset and Features

We started off with a training set consisting of 9000 images from ImageNet [15] (9000 32x32 images in X and 9000 64x64 images in Y). For the discriminator, the class label distribution is always 50% real and 50% generated, given that half of each discriminator training batch is taken from $G(X)$ and half is taken from Y.

We wanted to see whether the GAN performance was affected by the variety of classes in the image set. ImageNet consists of a large number of image classes, which might make learning harder for the GAN than if restricting to, say, just images with faces or a dataset like ImageNet but with fewer classes. As a result we also tested on CelebA [10] (all faces) and STL-10 [3] (essentially ImageNet but with fewer classes). Our CelebA dataset also had 64x64 images, but the STL-10 dataset had 96x96 images, which allowed us to evaluate how our model performed with across different resolutions. In these datasets as well, we used the first 9000 images as our training set.

As a step towards video, we also tested single image super-resolution on the center crop of an anime episode.

We do not perform any explicit feature extraction, but instead have our generator and discriminator work directly on image pixels. We normalized all of our low-resolution images between 0 and 1, and we normalized all of our high-resolution images between -1 and 1, in accordance with the preprocessing that Ledig et al. performed [9].

5. Experiments

With our GAN, we ran several experiments in order to determine the best training methods and architecture for our task of super-resolution. Across all experiments, we used a batch size of 64, and used various datasets, architectures, and optimizers.

One problem we found with our networks is that it takes the generator some time to learn how to reproduce shapes, and then colors, from a low-resolution image. However, in the iterations that the generator is learning to recreate these basic things, the discriminator becomes too strong and the generator is unable to fool it again. Lowering the discriminator's learning rate helped solve this problem, but later in training the generator surpassed the discriminator and started to always fool the discriminator, which was also a problem.

To solve this, we decided to pre-train the generator on mean-squared-error loss with the original high resolution image. We trained for 15 epochs - long enough that results from the generator alone were passable as super-resolved images. From there, we saved the weights and started training the generator with a different loss function - adversarial loss plus content loss - against the discriminator. This worked and ended up not only giving us more stable training, but also allowed us in development to see within a few iterations whether any change we made was working the way we wanted it to. Figure 4 shows a plot of the content loss while pre-training the generator on STL-10 data. Generative and discriminative losses after pre-training are not as monotonic and smooth, because having losses jumping back and forth rather than converging is the desired goal.

We found that training the discriminator two steps for every one step that we trained the generator was the best for our use case. Training one for one meant sometimes the generator would surpass the discriminator to the point of no return. Training the discriminator for more than two, however, significantly increased training time for the generator, which was a negative.

We tried two different architectures for upscaling our image. The first was PixelShuffler and the second a resize followed by a convolution. Figure 5 shows the consistent gridding artifact we observed when using the PixelShuffler. Although this artifact generally went away after several epochs, we found the using a resize convolution mitigated any gridding effects much earlier in training. Ultimately, after enough training, PixelShuffler got rid of its gridding artifacts and produced higher-quality high-frequency data, so we ended up using it for our 4x GAN.

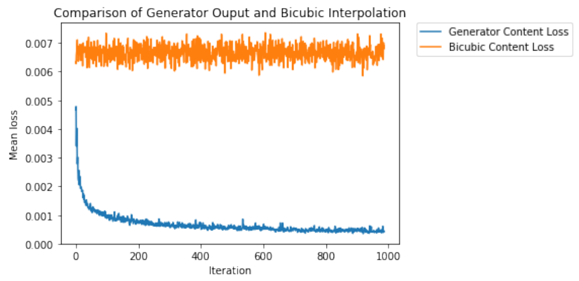


Figure 4. Mean content loss (Mean-squared error) while pre-training the generator on STL-10 vs content loss from bicubic interpolation

We tried three different loss function/optimizer combinations in training our GAN. We originally used softmax cross-entropy loss with an Adam optimizer with learning rate of 1e-4 and beta1=0.5, beta2=0.999. That got us training, however we were unable to train for more than a few hours before either the generator or the discriminator gained an advantage on the other. Next we tried a Wasserstein GAN with gradient penalties to hopefully train more stably, however results were not great. It seemed like having content loss in addition to discriminator loss was possibly creating problems and it was unclear whether further tuning would provide an easy fix, so we put it on hold. Lastly we tried the loss function from Least-Squares GAN, using Adam (this time with beta1=0.9) and least-squares loss, resulting in the most-stable training and best-looking images of the three approaches.

We made an attempt to make our super-resolution better by borrowing structure from surrounding frames. Instead of a standard 2D convolution across the low-res input image, we did a 3D convolution across the input frame, the two frames before it, and the two frames after it. Unfortunately, this didn't work as well as we expected it to. Training the generator to learn the basics of super-resolution took many more iterations, probably because of the increased input size and not-always-clear connection between the surrounding frames. Additionally, iterations took very very long, since every convolution in our 2D GAN was multiplied across so many frames. It was worthwhile thing to try, and maybe something to revisit, but because we were having so much success with our 4x GAN at the same time, we decided to drop this experiment and make our computing power more useful running other experiments.

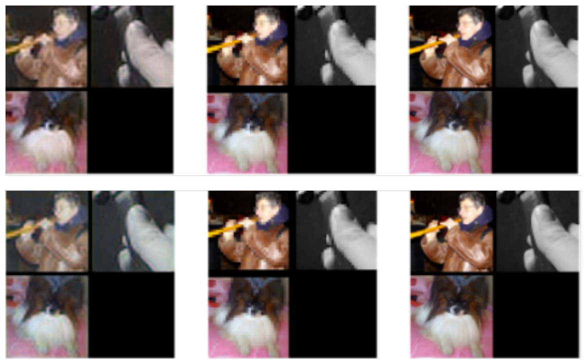


Figure 5. Example results using PixelShuffler (top) and resize convolution (bottom). From left to right: generated, low resolution, high resolution.

6. Results and Discussion

Figure 6 contains a sampling of our single image 2x (32x32 to 64x64) super-resolution results across the various datasets.

Qualitative discussion: In Row 1, we can see how lines in the generated image are bolder and sharper than those in the bicubic interpolation. In Row 2 edges appear sharper in generated, but with some color noise. In Row 3, there is blur in both generated and bicubic, but generated is able to more accurately produce dog fur texture. In Row 4, our GAN seems to hallucinate eye shadow underneath the eye when in fact it is only above the eye - most likely an artifact from other images in CelebA. In Row 5, here the GAN very accurately reproduces the CelebA face. In Rows 6 and 7 we see strong performance on STL-10, with some minor background color noise added by the generator.

All in all, we find that on 2x upsampling the GAN produces sharper 64x64 images of its 32x32 inputs than bicubic interpolation, sometimes at the expense of added background color noise and artifacts. On a different note, we find that when the dataset has too many class labels (e.g. ImageNet), the network has a hard time generating fine detail across all classes. Conversely, for a dataset with too few class labels (e.g. CelebA), our GAN sometimes generates detail that should not be there, such as the eye-shadow in Figure 6. Overall, it seems that our single image GAN performs best when there is enough variety of class labels in the dataset (anime crops and STL-10), but not too much variety.

In Figure 7 we include some preliminary results for super-resolution on video. For these, there is an extra temporal dimension in all convolution filters. These results are just for pre-training the generative network, which given the number of parameters is very computationally expensive. We find that the generative network is able to learn color quite well and is starting to learn some spatial structure. However, there seems to be blur possibly coming from adjacent frames. We decided to leave the remaining work in video to future work, and perfect single image super-resolution instead.

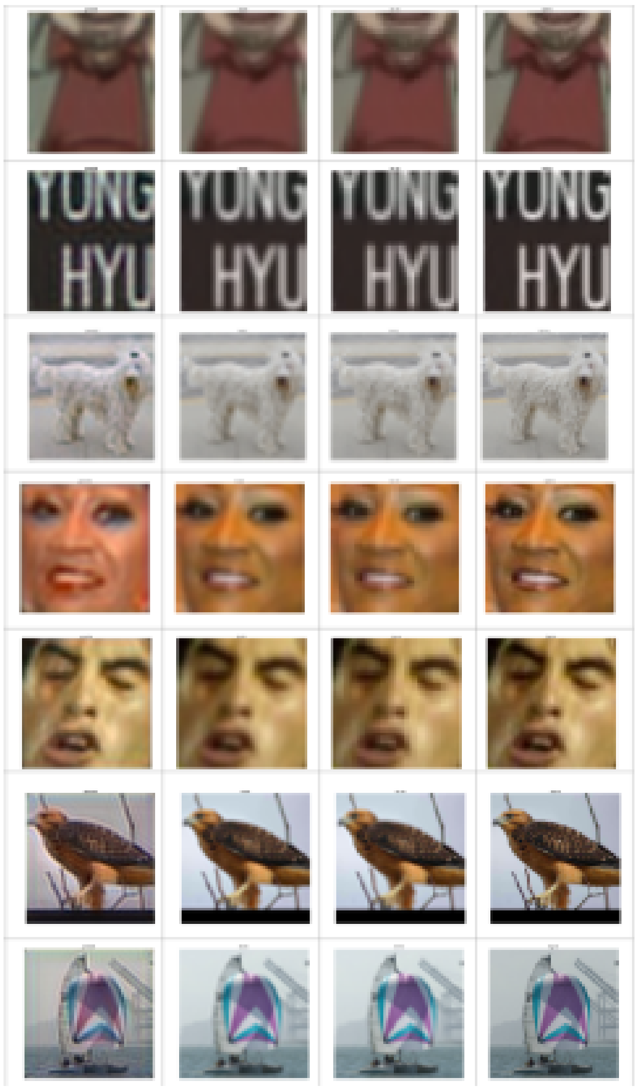


Figure 6. Columns from left to right: GAN output (SR image), result of bicubic interpolation, original LR image, true HR image; Rows from top to bottom: 1,2 from anime, 3 from ImageNet, 4, 5 from CelebA, and 6, 7 from STL-10

We combined the findings of our experiments into a GAN that successfully upscales images by 4x, from 32x32 pixels to 128x128. We used the pixel-shuffle generator al-

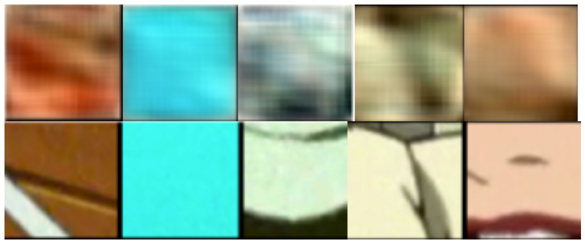


Figure 7. Preliminary video super-resolution results just using generative network. Top: generator output, Bottom: high resolution

gorithm from before, with one additional conv-pixelshuffle-prelu layer to go from 2x to 4x. We pretrained the generator on mean-squared-error pixel loss for several hours. We then used least-square loss (adding content loss and mean-squared-error pixel loss to the generator's loss) and an Adam optimizer, training the discriminator two iterations for each iteration of training the generator, in order to keep training stable. In Figure 8 we can see results. They look possibly as good as an industry standard super-resolution would be. However, it is important to note that this dataset was fairly small and this trained generator would be unlikely to perform as well upsampling real images as it does on drawn lines. We would need a dataset with a broader set of image classes and much more training time in order to create a industry-standard super-resolution tool from our 4x GAN.

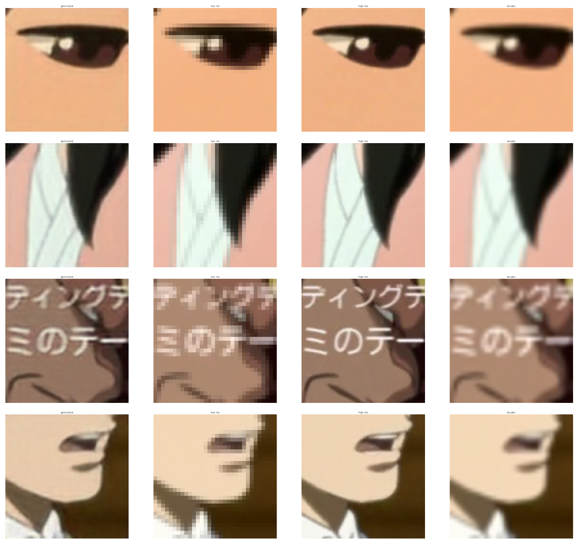


Figure 8. 4x GAN example validation results after training. From left to right: Generator outputs, low-resolution generator inputs, high-resolution ground truth images, bicubic upscale

7. Conclusion and Future Work

Our goal for this project was to explore single-image super resolution and extend our findings to video. In the end, we received results that beat standard bicubic interpolation methods. We are unable to make a direct comparison between our results and those of Ledig et. al [9], because we mainly focused on a 2x up-sampling for general purpose images, and 4x up-sampling only within the space of animated images, whereas they tackled 4x super-resolution on ImageNet. However, we suspect we would need to spend far more computation time training in order to obtain results comparable or better than theirs.

More computational resources would enable more expedited progress on the video section of this project. The number of trainable parameters when adding the temporal dimension into our networks (and each convolutional filter therein) grows significantly. We also did no pre-processing on the video before feeding it as input, but something we could try in the future would be to keep the frame to be super-resolved the same, but turn all other frames into a diff with the center frame. This could possibly make it easier for the network to learn what temporal information is useful in fewer iterations.

Based on Figure 1, we ran some experiments in which we trained a GAN directly on 2D Fourier Transforms, rather than the images themselves. The idea here is that if what we are trying to add is high frequency content, it might be better to work with images in the frequency domain instead of the spatial domain. Part of the issue with this approach is that the highest magnitudes in a 2D Fourier Transform are generally at low frequencies, and as such, we would need to adapt the architecture for the network to focus mainly on altering high frequency content (perhaps by working with log transforms). On top of this, we need to extend TensorFlow convolutions to work with complex numbers as, from experimentation, working with just the magnitude (or real part) of the FFT is insufficient. Figure 9 shows preliminary results when training a generator just on the real part of the FFT.

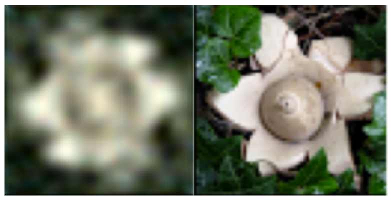


Figure 9. Preliminary generator results training on real part of FFT using ImageNet

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. *Nature*, 521:436–444, 2015.

[3] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014.

[5] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[10] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[11] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, and Z. Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

[12] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[13] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.

[14] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[16] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[17] B. C. Tom and A. K. Katsaggelos. Reconstruction of a high-resolution image by simultaneous registration, restoration, and interpolation of low-resolution images. In *Image Processing, 1995. Proceedings., International Conference on*, volume 2, pages 539–542. IEEE, 1995.

[18] R. Tsai and T. Huang. Multiframe image restoration and register. *Advances in Computer Vision and Image Processing, JAL Press Inc*, 1984.

[19] H. Ur and D. Gross. Improved resolution from subpixel shifted pictures. *CVGIP: Graph. Models Image Process.*, 54(2):181–186, Mar. 1992.

[20] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.