FoodGAN: SuperResolution on Low-Quality Food Images

Federico Reyes Gomez
Dept. of Computer Science
Stanford University
frg100@stanford.edu

Amay Aggarwal
Dept. of Computer Science
Stanford University
amayagg@stanford.edu

Chinmay Patel
Dept. of Computer Science
Stanford University
chinmayp@stanford.edu

Abstract

In this paper, we develop and compare various Generative Adversarial Networks for single image super resolution (SISR) using a high quality food image training dataset. Extending previous approaches used to conduct a more generalized form of super resolution (SRGAN)[1], we find that improving the generator loss function to incorporate a more complex VGG loss and to optimize for structural similarity improves the performance of our model. Overall, we conclude that SRGAN-like models can have improved outputs with a modification as small as changing the generator loss function, leading to noticeable results with a dataset consisting of under 10k images.

1 Introduction

This paper trains various generative models to enhance low quality food images, by performing superresolution and de-blurring. The input to our algorithm is a blurred low resolution image of food. We then use various modified versions of the SRGAN (Super Resolution GAN) architecture [1] to output a higher resolution image, which is a deblurred higher resolution version of the input.

The number of food images posted on social media such as Instagram and VSCO grows by the day, with over 300 million photos [2] currently posted under food related hashtags on Instagram. However, these images are often taken in settings that only permit low quality resolution. Although a significant amount of work has been conducted on performing superresolution on human faces [3], applying some of these same architectures and techniques to the context of food is an interesting and widely applicable domain of study. In doing so, we hope to build on some of this architecture and leverage the power of Generative Adversarial Networks (GANs) to conduct similar image enhancement on low quality food images.

2 Related Work

The literature on superresolution tasks has exploded in the last couple of years, with SRGAN still as the most salient architecture. The SRGAN paper borrowed from various previous attempts to arrive at a then state of the art 16-block SRResNet with a loss calculated on VGG-derived features instead of an MSE-based content loss. They focused on Single Image Super Resolution (SISR) with a large, nonconstrained domain. They used 350,000 randomly sampled images from the ImageNet dataset to train on. We decided to take the great progress done by SRGAN and suggest some improvements, testing with a constrained domain, high quality food images. This should be an easier task that a general-purpose super resolution algorithm, allowing us to see results more quickly without having to train on such a large dataset.

We found one previous approach [4] that used SRGANs to perform super resolution specifically on a food image dataset. This paper compares the benefits of adversarial (SRGAN) and non-adversarial (SRResNet) models, and notes that adversarial models such as the SRGAN tend to perform better and have higher perceptual similarity, or lower pixel wise loss. However, this paper simply uses noise injection to create their dataset. We attempted to create a better synthetic dataset with multiple quality-lowering effects. Another approach [5] focuses on SISR through three different non-adversarial methods, and concluded that using a LAPGAN or SRGAN may serve to be more effective.

3 Dataset and Features

We scraped 8000 images from Instagram that were tagged with the four most common food-related hashtags: #food, #foodporn, #yummy, and #yum. We then divided these images into training set, validation set and test set as follows:

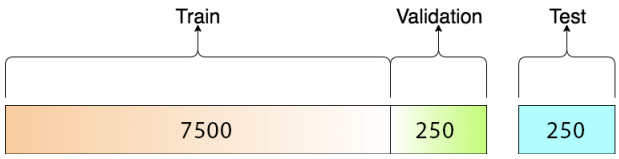


Figure 1: Distribution of images into training, validation and test sets

As the resolution of the images varies, we filtered out all images with resolutions lower than 1080×1080. When loading the dataset, we took top-left crops of all our input images in order to have a consistent high-res image size of 1080×1080. Additionally, in order to be able to train the model and not run out of memory, we took random 96×96 crops of the images. Since the generator is fully convolutional, we are able to feed in any sized image at test time.

Since we only have full-resolution reference images, we needed to pre-process them into blurry, scaled-down versions for the generator to learn from. Also note that this processing is done at train-time, not beforehand. This slows down our iteration speed, but was a much simpler implementation than implementing a complex preprocessing pipeline. To do this, we created a non-trainable model in Keras to implement two steps:

1. **Blur**: In order to blur input images, we first generate a 3×3 Gaussian filter with $\sigma = 0.5$ and then apply a *depth-wise* 2D convolution. This means that for every channel of the input image, we apply a 2D convolution layer, and then stack the channels back up.
2. **Downsample**: In order to downsample the image (by 4 times in our case), we just applied a 4×4 average pooling layer. Note that we could have used max-pooling here as well.

4 Methods

4.1 Model 0: SRGAN

For our baseline, we read the original SRGAN paper [1] and re-implemented it using the Keras framework and TensorFlow 2.0, which has only been available for a little over two months.

In order to implement the custom ResNet architectures specified in the paper, shown below, we had to use the Keras Functional API. However, with the ease of Keras we were able to get a working version of the SRGAN, as described in the original paper, to serve as 1) our baseline and 2) a working model that we can build off of and modify for our specific task.

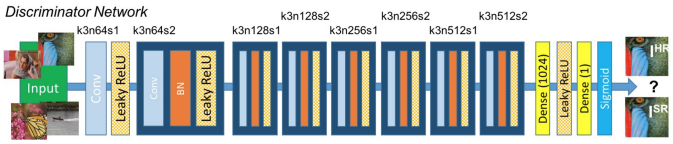


Figure 2: SRGAN Discriminator Architecture (Ledig et al.)

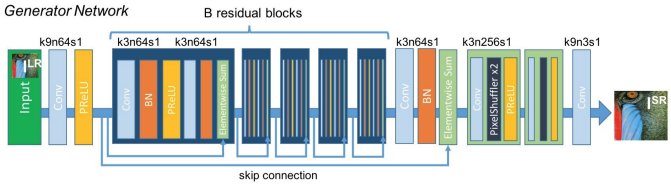


Figure 3: SRGAN Generator Architecture (Ledig et al.)

We also implemented the custom loss functions described in the article, most interestingly one of their most significant results, a loss function based off of feature similarity after running images through the VGG-19 network and stopping before the final classification.

4.2 Model 1: VGGAN

As our first modification, we decided to focus on improving the generator loss. In the SRGAN paper, this consists of three parts: a sigmoid cross entropy loss to measure how well the generator is fooling the discriminator, a mean squared error loss between the generated images and the hi-res images, and a VGG loss consisting of the mean squared error between the image features of the generated and original hi-res images.

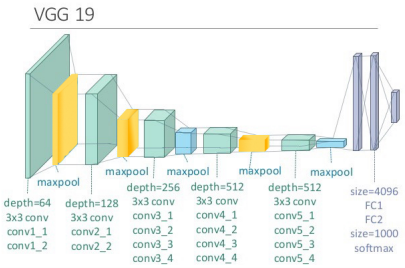


Figure 4: VGG Architecture with feature layers highlighted in yellow

This VGG loss was one of the more interesting parts of the paper, so we decided to explore that further. SRGAN uses the representations of the images after the fourth pooling layer (`pool4`) as the features. However, we decided to *also* use the outputs at two other layers in our loss. Specifically, we focused on earlier layers, `pool1` and `pool2`, highlighted in yellow in the image above. We decided on these layers since the vanilla SRGAN model was outputting images that still looked very blurry. As earlier layers tend to learn more low-level features, forcing the model to optimize the low-level features of the generated images to resemble the high-quality images should hopefully gives us a sharper output.

4.3 Model 2: SSIMGAN

For our final modification, we focused on structural similarity, one of the measures we will be using to evaluate our models. Although we were unable to find other models that included structural similarity as an objective for the optimization of the generator, we decided to experiment with it using the following equation:

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

We can see that SSIM depends only on the average values, variances, covariance, constants, and stabilization factors of the two images. Thus it is differentiable, and can be used in our loss function.

4.4 Evaluation Metrics

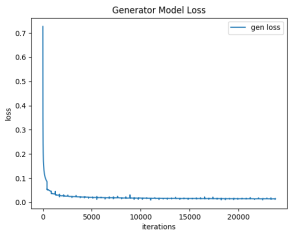
To evaluate our model, we standardize our test set to 1080 x 1080 crops of each image, which we then apply gaussian blur ($\sigma = 0.8$) and downsample by a factor of 4. We pass this lower resolution image (270 x 270) into our trained generator, which outputs a higher resolution image (1080 x 1080). We then compare the original full resolution image, the downsampled image, and output of our model using a variety of quantitative and qualitative metrics.

Firstly, we use Structural Similarity (SSIM) and Peak Signal-to-Noise Ratio (PSNR), two of the most commonly used metrics to evaluate super resolution and to compare how close our output is to the original full resolution image. We also calculate the blur using a Variance of Laplacian Blur Detector (LBD) [7] to measure the difference in blur between the downsampled image and the generated image. Lastly, we use a qualitative form of evaluation. At first, we wanted to use Mean Opinion Score (MOS) testing to validate our results, but preliminary MOS testing wasn't giving us interesting results. Thus, in order to get a more expressive metric to differentiate between the outputs of each of our models, we asked participants to rank our images and then took the average of all the rankings.

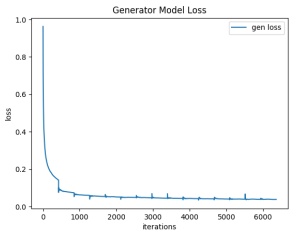
5 Results

5.1 Training Losses

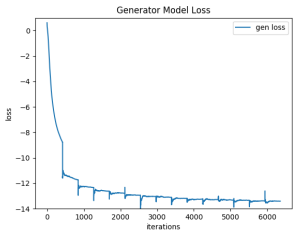
We ran each model for at least 15 epochs and all comparisons are at the 15th epoch, although we did train the SRGAN version for over 50 epochs to see if we could get even better results. As the figures show, we got convergence in the losses for most of the models, with the notable exception of the SSIMGAN.



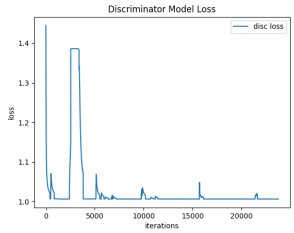
(a) SRGAN Generator Loss



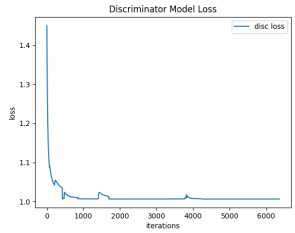
(b) VGGAN Generator Loss



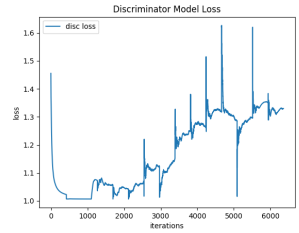
(c) SSIMGAN Generator Loss



(a) SRGAN Discriminator Loss



(b) VGGAN Discriminator Loss



(c) SSIMGAN Discriminator Loss

Figure 6: Generator and Discriminator Losses

5.2 Evaluation at 15 Epochs

As mentioned above, we used four metrics to rate our results, 1) SSIM, 2) PSNR, 3) LBD, 4) RANK. We calculated each of the metrics for 10 test images and took the averages for each model, 1) SRGAN, 2) VGGAN, 3) SSIMGAN, and 4) the original High-Resolution image as a baseline. Our results are presented below:

Model	SSIM	PSNR	LBD	RANK
SRGAN	0.804	67.7	138.5	1.94
VGGAN	0.781	66.0	150.9	2.3
SSIMGAN	0.848	68.6	30.2	1.76
HR	1	N/A	451.1	N/A

Table 1: Quantitative Results

(a) Downsampled image (b) VGGAN output (c) SSIMGAN output (d) Original image

Figure 7: Outputs for comparison

6 Discussion of Results

It has been found that the dataset used significantly affects the quality of super resolution models [6]. Thus in using a very specific dataset for training consisting of high quality food images, we were able to get reasonable results with a fraction of the compute power and dataset size, as compared to the general-purpose SRGAN.

6.1 SSIM

Structural Similarity is a metric that takes into account general perceptual phenomena and is important in determining whether high-level structural information is retained. Thus is makes perfect sense for SSIMGAN, a model trained with SSIM as an objective, to have the highest average SSIM score. Additionally, we see that SRGAN and VGGAN are close to each other with SRGAN slightly outperforming VGGAN. This also makes sense since VGGAN was optimized for sharpness and low-level detail, not high level structural information.

6.2 PSNR

The PSNR values for all our models are very similar, and thus this metric does not serve to be useful in our analysis. However, given its prevalence in super resolution literature, we decided to include it in our evaluation.

6.3 LBD

This was the most interesting metric that we calculated. Although not prevalent in literature, LBD helped confirm our suspicions that the training of SRGAN-like models depends non-trivially on the loss function. Initially, when looking only at SSIM and PSNR, we were seeing that SRGAN was generally outperforming our other models, yet the generated images from VGGAN seemed a bit clearer and sharper. Once we calculated the LBD scores, we saw that this was, in fact, the case. The output of VGGAN outperforms SRGAN in this measure of image sharpness and clarity. Interestingly, SSIMGAN performed significantly worse on this metric. We think this may be because the model focuses on optimizing for high-level structural similarity and cares less about image sharpness, leading to a blurrier image. However, we note that the model did improve the sharpness from an average of 9.9 for the downsampled images to 30.2.

6.4 Qualitative Results

We asked 25 participants to qualitatively evaluate our results by ranking each image generated from our model, as well as the original and downsampled image. Across our sample size, it was unanimous that the generated images from all three models were better than the downsampled version and worse than the original, as expected. Qualitatively the differences in the images were very small but still notable. SSIMGAN had the lowest average rank (indicating clearer generated images) followed by the baseline SRGAN, and VGGAN after that. At a high-level however, there is significant super resolution from the downsampled image.

7 Conclusion and Next Steps

We are able to come to the conclusion that small modifications to SRGAN-like models, such as changing the generator loss function, can result in noticeably different outputs. This leads us to believe that targeted improvements in the loss function can help better performance of SRGAN-like models for a specific task.

We note that our optimizations of the loss function were very minimal due to the time we had, so it would be very interesting to try to tune the scales of the losses more finely to see if the results can be even further improved. It would also be very interesting to try other modifications to SRGAN in order to get even better results, such as Implicit Maximum Likelihood Estimation [8]. Notably, using pre-trained models other than VGG for feature representations, and trying out different architectures for the generator.

As another next step, we notice that a lot of our output images contain faded colors compared to the original and down-sampled images. Although this saturation can be corrected manually on the output of the generator, one way to more directly address this would be to include color difference as an objective in the loss function.

8 Contributions and Code

In terms of contributions to the project, Federico worked primarily on the implementation and analysis of the models. Amay focused on generating output, addressing GAN compilation issues, and evaluation metrics. Chinmay worked on compiling the dataset, pre-processing our input, and evaluation metrics. Please find our implementation code of all three models on our Github repository at **https://github.com/frg100/cs230_foodGAN**

Acknowledgements

We would like to thank our project TA, Jo Chuang. He was incredibly helpful in giving us direction for our project and in debugging a key memory issue we were facing.

References

[1] Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[2] https://thenextweb.com/opinion/2015/09/01/why-sharing-photos-of-food-is-about-more-than-whats-on-the-plate/

[3] Shu, Yujie. "Human Portrait Super Resolution Using GANs." CS 230.

[4] Yudai Nagano and Yohei Kikuta. 2018. SRGAN for super-resolving low-resolution food images. In Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management (CEA/MADiMa '18). ACM, New York, NY, USA, 33-37. DOI: https://doi.org/10.1145/3230519.3230587

[5] Jaffe et al. "Super-Resolution to Improve Classification Accuracy of Low-Resolution Images"

[6] Nao Takano and Gita Alaghband. "SRGAN: Training Dataset Matters". 2019

[7] https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/

[8] Ke Li, Shichong Peng, and Jitendra Malik "Super-Resolution via Implicit Maximum Likelihood Estimation" 2018