GAN-based Inpainting Study

Nik Bhatt; UNI: nb3097

I. Synopsis

Inpainting (image repair, object removal, and image synthesis) has a long history in computer science. Early techniques used algorithmic approaches like patch finding, while the latest research leverages Generative Adversarial Networks (GANs). After studying several inpainting algorithms and literature reviews, I noticed certain tendencies. Each research team uses the same datasets, looks for small improvements in aggregate, and shows cherry-picked results. I noticed only small overall improvements in standard metrics from paper to paper. No variety in the datasets, combined with similar results, might indicate that the solutions or approaches are highly tuned to one problem domain and lack generality. So, I set out to test these GANs on a different public dataset to see how well the algorithms performed on novel images.

Further, the latest GANs focus primarily on image synthesis, where most of the image is obscured and the goal is to create a realistic new image. That image does not need to closely resemble the ground truth – just be appealing and realistic. While that is an important problem, I'm interested in object removal because that is more applicable to the photo editing domain in which I work. As a result, I decided to test GANs using segmentation masks rather than the randomly generated masks found in research papers. I would collect a consistent set of statistics using two datasets and compare the results across multiple GANs.

First, I had to get them to run on my Apple Silicon Mac, which turned out to be more difficult than I anticipated. Some GitHub repos were missing code. Others assumed the presence of CUDA (with no CPU fallback), or the CPU code required an Intel architecture. Each researcher calculates statistics differently, using custom code in many cases. Modifying each research group's code to output a standard set of statistics became an ordeal.

Since all GAN researchers must compare their solution to pre-existing ones, they may encounter similar problems. My solution is to create a reusable library that future researchers can use to evaluate GANs. By writing a small amount of python code, different GANs can be tested and measured consistently. My solution does not make it easier to get a GAN to run, but once it is running, the effort to evaluate it is greatly simplified. The code is freely available on GitHub with instructions for its use. Here are its features:

- Flexible. This is important because each GAN uses different ways of storing originals, masks, and generated images. For example, some GANs use masks that are bitwise-inverted from others or have specific image size requirements. Some GANs require the masks to be in PNG format, while others want JPEGs. A researcher can evaluate a GAN by writing a small adapter subclass in Python. The adapter is described below.
- Statistics calculations: FID, LPIPS, and SSIM are calculated. Those are described below.
- Automatic generation of a PNG that is the original image with the mask translucently composited, like this:

- Automatic generation of an animated GIF that shows the original, the above PNG, and the generated image to make it easier to visualize the inpainting result.
- A detailed CSV file with paths to input images, masks, and output images, statistics for every image, and mean, standard deviation and variance for FID, LPIPS and SSIM. By producing a CSV, researchers can create their own charts and graphs either in a spreadsheet or in, say, Jupyter Notebook.

II. Novelty and Value

In my previous literature survey, I found other papers, such as Xiang [1], tested a wide range of GANs. However, they used the same datasets as the papers did and used random masks as well. They did not supply any code to replicate their findings or make it possible for future researchers to leverage their hard work. My novelties and benefits are four-fold:

	Xiang	Mine	Benefit
Datasets	ImageNet, CelebA, Places, Paris StreetView	Open Images v7 [2]	Open Images is not used in inpainting research, providing a novel dataset to test GANs.
Models	A wide range of models up to 2021	Newer models	Using newer models provides better comparison data for new research papers.
Masks	Randomly generated masks	Open Images includes segmentation masks for over 13,000 images.	Because these masks are based on object boundaries, they are more useful for object removal, compared to random masks which may be better for image synthesis.
Reusable Library	None. No library or code provided	Reusable library on GitHub with instructions for use and links to datasets and CSVs.	Researchers can use my library to evaluate both their own design and others without reinventing the wheel.

III. Reusable Library

The reusable library consists of three main files. adapter.py, mask-maker.py, and analysis.py. I use the term "library" to mean "independent module of code." The python files could be added to a GAN project, or they can be run separately from GANs by pointing them to the desired paths on disk. I took the second approach for my own analysis to avoid modifying the GAN source code.

Adapter.py bridges the GAN code with the mask-maker and analysis components. I provide a base class with functionality already implemented. I determined the necessary class methods by evaluating three different GANs.

Method	Purpose
`__call__`	Input: source image path Output: actual mask image path, actual result path
`overlay_name`	Input: source image path Output: desired composite image path
`needs_scaling`	True if images need to be resized
`resolution`	If resizing, what size to make images and masks
`img_extensions`	Returns valid file extensions for source images
`mask_is_black`	Returns True if masked pixels should be black. False if masked pixels should be white.

For any given GAN, the researcher first determines what the GAN expects (and produces) in terms of input files, output files, and masks. Then, they can write a subclass, implementing whichever methods are necessary. For example, below is the adapter for MI-GAN, which uses a particular naming format for its mask images.

```
class Migan512Adapter(Adapter):
    def __init__(self):
        super().__init__()

    # return mask path and result path
    # customize this to match your own naming conventions
    def __call__(self, real_image_name, mask_path_str, result_path_str):
        stem = Path(real_image_name).stem
        mask_image_path = mask_path_str / f"img000{stem}.png"
        result_image_path = result_path_str / f"{stem}.png"
        return mask_image_path, result_image_path

    def resolution(self):
        return 512, 512
```

Below is an adapter for HiFill; this algorithm uses a particular naming convention for output images and a different resolution for the images.

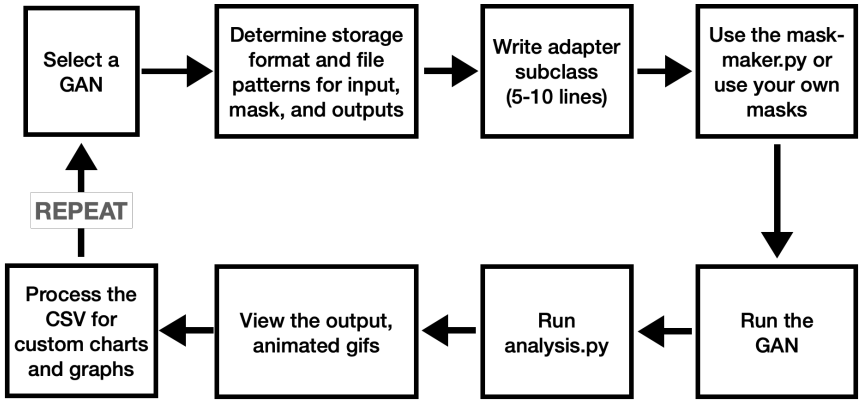
```
class HiFillAdapterPlaces(Adapter):
    def __init__(self):
        super().__init__()

    def resolution(self):
        return 512, 512

    # return mask path and result path
    # customize this to match your own naming conventions
    def __call__(self, real_image_name, mask_path_str, result_path_str):
        stem = Path(real_image_name).stem
        mask_image_path = mask_path_str / f"{stem}.png"
        result_image_path = result_path_str / f"{real_image_name}_inpainted.jpg"
        return mask_image_path, result_image_path
```

As can be seen, the adapter subclass is quite small and simple to write. The adapter classes are used by the mask-maker to generate mask files of the right file format, naming convention, resolution, and bit pattern (white-on-black or black-on-white). The adapter is used by the analysis code to match input and output files (to calculate statistics), and to generate animated gifs and other visualizations that show the repair.

Mask-maker.py is responsible for assigning masks to input images. For the Open Images dataset, a subset of the validation images includes segmentation masks. The mask maker filters the validation image list to those images with masks because only images with valid segmentation masks are appropriate for this object removal test. It also composites multiple segmentation masks if more than one is present for an image (this happens if more than one object was identified and masked). It will resize the original image if the GAN has specific image size requirements, it will resize the mask to match the original image's size, and it will invert the mask if the GAN requires it. The final output of the mask-maker is a set of directories: the input directory, the mask directory, and a directory of mask visualizations so researchers can see what will be inpainted before running the GAN. These directories can then be fed into the GAN without modifying it because the files and formats match the GAN's requirements.

Analysis.py is used after the GAN has been executed. The analysis code creates an animated GIF of the repair, showing the original image, the mask composited on top of it, and the repaired image. It also calculates the three statistics (LPIPS, SSIM, and FID). Finally, it outputs a CSV. At the top of the CSV are combined statistics such as LPIPS mean and standard deviation (and FID, which is a single number for the entire dataset). Then there are a series of rows, with the paths to the files and the calculated statistics for each input image. By processing the CSV, researchers can create their own charts and graphs that meet their needs.

Workflow:



IV. Setup and Metrics

I evaluated several recent GANs for suitability for this final project: HiFill [3], which is from 2020, CoModGAN [4] (2021), MI-GAN [5] (2023), LaMa [6] (2021) and FCF [7] (2023). All of these were candidates because they provided code on GitHub, including pre-trained models. However, I excluded LaMa because the code provided was incomplete and I excluded FCF because it had a dependency on CUDA, which I do not have on my Mac. That left three GANs: HiFill, CoModGAN, and MI-GAN. Though that is fewer GANs than I hoped, I believe there is still enough variety to answer my research questions. I used pre-trained models because training the models would be time- and cost-prohibitive.

All three GANs were pre-trained on two datasets: a dataset of scenery called Places [8], and a dataset of human faces (there are a few different face datasets commonly used). For my project, I used the Places pre-trained models, as that is much closer to the Open Images dataset in content. Both contain a variety of scenes, with and without people present. Below are the sizes of the two datasets.

	Train	**Test**	**Validation**	**Segmentation Masks**
Places	1.8 M	45,000	36,500	0
Open Images	1.7 M	125,000	41,620	13,524

Because the Places dataset does not include segmentation masks, researchers randomly generate masks, using a variety of approaches (resulting in a wide range of sizes and shapes). A subset of the Open Images validation set contains segmentation masks which define a particular object or objects in the image. Not all objects in a scene have segmentation masks but the masks that exist are generally accurate. Because the randomly generated masks for Places are never provided in research papers (perhaps due to data size), I randomly assigned segmentation masks from Open Images to the Places images.

There are several metrics in use for inpainting GANs, though three are most computed and compared. The following table is adapted from my mid-term paper, and I computed all three of these in my library.

Metric	Description	Normalized	Best Score
Fréchet Inception Distance (FID)	A low-level metric comparing the distribution of the generated images to the distribution of the real images using the InceptionNet v3 model.	No	Lower
Learned Perceptual Image Patch Similarity (LPIPS)	Measures perceptual similarity by calculating the distance in VGGNet (or similar) feature space	Yes	Lower
Structural Similarity (SSIM)	Calculates the difference in luminance, contrast, and structure, weighted to apply relative importance to each.	Yes	Higher

V. Research Questions

1. Can I reproduce the results as described in the papers?

The short answer is no, because researchers do not provide the randomly generated masks that they use. When they provide sample images and masks, those are cherry-picked, and the results of those images are far better than the mean values published in the papers. However, since I used the Places dataset as a control (with my own randomly assigned masks), I tried to compare those results. Again, however, I was stymied. The HiFill paper calculated FID and MS-SSIM (which is the multi-scale version of SSIM), but not LPIPS. The MI-GAN paper and code, which I used for both MI-GAN and CoModGAN, calculated a confusing mix of statistics. For some test cases, they would compute FID and LPIPS (but not SSIM). For other cases, they calculated SSIM, but not LPIPS or FID. The paper did not explain why.

2. How different are the reproduced metrics vs. ones using the Open Images dataset?

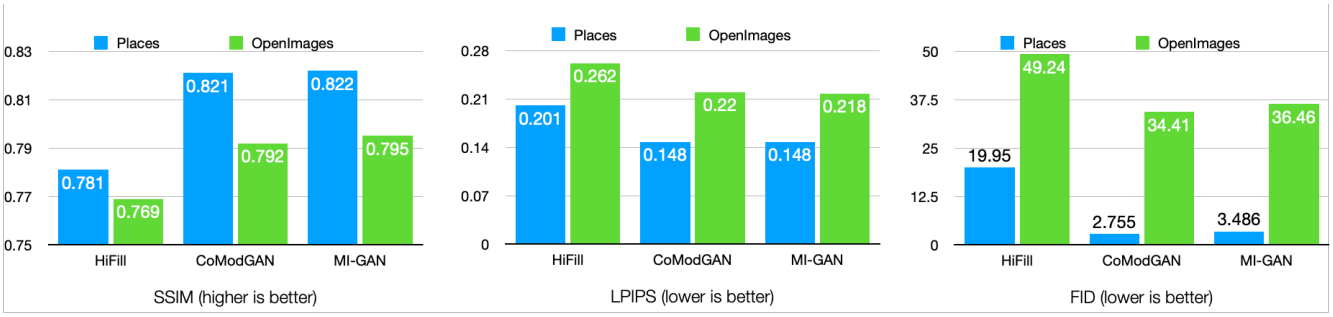
As mentioned above, I cannot reproduce the metrics because the papers do not provide their masks. I was able to compare my results with Places with my results with Open Images.

3. [NEW] What type of ablation study might be reasonable for this project?

In GAN research, a typical way of handling ablation is to remove portions of the network – typically by using a vanilla network without the improvements described in the paper. That would not be practical in my case, because I am using pre-trained models, which include all the defined advantages in each paper. I cannot change the masks used because those are not provided by any papers. Instead, I chose to compare the results from two different datasets with the same masks, where one of the datasets is the same one used for training the models. This can indicate how much the dataset matters in terms of repair quality. This is not the same thing as GAN ablation, but has some analogous characteristics.

4. How different are the results from their datasets to the Open Images dataset and masks?

The results using the Places validation images (which come from the same overall dataset each GAN is trained on) is noticeably better than the results with the unseen Open Images dataset. Again, my results are not identical to what the papers did because they use different masks (and do not provide them).



Each metric is worse when Open Images are used. The inferior SSIM results for Open Images would indicate that the luminance and contrast of the generated images are inferior, which would make the Open Images generated images appear worse to humans than the ones from the Places validation set. That said, the difference between the two datasets is relatively small. For LPIPS, the results are noticeably worse than SSIM. LPIPS attempts to measure human perceptual similarity of pairs of real and generated images. For CoModGAN and MI-GAN, the Open Images values are about 50% worse than the Places values. Finally, the FID stats are dramatically worse – more than an order of magnitude for MI-GAN and CoModGAN. FID attempts to calculate how "real" the generated images would appear to people by comparing them to a pre-existing set of real images. Here the GANs really falter with very poor results across the board for the Open Images dataset.

5. If they are different, what does the difference imply?

The inferior results across the board imply that the choice of dataset may matter. I'm using random masks for the Places dataset, so that is similar to what the researchers are doing. The poor LPIPS and FID results may indicate either overfitting or overly-specific algorithmic design, inadvertently targeting features of the Places dataset.

6. Are newer GANs able to remove objects with good quality results, or has that been diminished in the race for image synthesis? Are they better than the control (HiFill)?

There are a few ways to try to answer this. One way is to look at the statistics and analyze them based on the size of masks. Smaller masks should be easier to repair faithfully than larger masks. The histograms in the Appendices provide some information there. What they show is a correlation between LPIPS and mask size. Smaller masks have better LPIPS values with a significant drop-off after the first bin. The conclusion is, for smaller repairs, both MI-GAN and CoModGAN are effective at repairing images and removing objects.

Another way is to look at sample output. Given that the Open Images dataset has 13,000 images in it, it is not practical to review every image, though my initial scan began to reveal some disturbing trends with LPIPS and SSIM which are discussed below.

The newer GANs (MI-GAN and CoModGAN) are indeed better than the older HiFill algorithm in all three metrics. In my proposal, I referred to HiFill as the control because I was expecting to evaluate more newer GANs than it turned out I was able to.

7. [NEW] How well do LPIPS, FID, and SSIM correlate to human subjective quality evaluations?

I began to wonder about this when reviewing different GAN papers earlier in the semester, especially when papers seemed to have fairly similar statistics to each other. Older papers used PSNR (peak signal-to-noise ratio) and MSE (mean-squared error), but those have two problems. First, those metrics are absolute values, not normalized, which makes them more difficult to compare. Second, they are not perceptual in nature and are deceived by noise and blurring [9]. Algorithms such as SSIM and LPIPS are intended to address these shortcomings, and they generally do that. However, they are not necessarily ideal for all image comparison use cases. Here are two examples of inpainting from MI-GAN.

Mask Percentage	LPIPS	SSIM
5.86%	0.0573	0.9444

This is a very good, though not perfect, result with the object removed and some detail restored (image is cropped).

Here is another example from the same GAN and dataset, with nearly identical mask percentage, and better metrics (the image also has been cropped).

Mask Percentage	LPIPS	SSIM
5.85%	0.0395	0.9681

The result is quite poor. More worrisome, an LPIPS score of 0.0395 is exceptional, since 0.0 is perfect. And, SSIM is also excellent at 0.9681 (perfect is 1.0). For reference, the entire dataset for MI-GAN has an LPIPS of 0.220 and SSIM of 0.792, so the "bad" example has well-above results for this GAN. Sadly, there is no shortage of such cases in the dataset, whereas it was harder to find good results.

That would indicate, at least some of the time, LPIPS and SSIM do not indicate high quality repairs. Based on my review of the result images, I would argue that these statistics are *often* not valid for inpainting and there is a simple explanation why.

Both of these metrics are calculating some level of similarity to the ground truth image. This results in a catch-22. If the mask is small, then most of the repaired image and the ground truth image are identical. So, the stats are going to be very good, *regardless of the quality of the repair*, because only a small percentage of the image is affected. If the mask is very large, the images are going to be very different, and the stats will seem poor, even if the repair makes an attractive image because a large percentage of the image is being replaced and thus are not very similar to the ground truth.

A quick solution might be to confine the comparison to the masked area. However, this will not work either. In the case of a small mask, comparing just the masked region before and after repair is analogous to the case of a large masked area. In both cases, most of the pixels will be changed and the similarity metrics will be bad. In addition, this is missing the point of inpainting. The algorithm is not trying to make new pixels that look like the bad ones. It is trying to fashion new pixels that fit with the image.

The CoModGAN paper introduced some new metrics, called P-IDS and U-IDS (Paired/Unpaired Inception Discriminative Score) [4]. These are intended to be superior to metrics like SSIM, but even that paper still included LPIPS and FID. For unknown reasons, newer papers have generally not adopted these metrics. Future work could add them to the library's analysis code in the hopes of a more credible statistic.

8. Which recent algorithms are best at object removal with an unseen dataset, both in terms of quality and runtime? How well do they handle larger images on a standard laptop?

CoModGAN and MI-GAN performed similarly to each other. I was not able to run enough GANs to draw more of a conclusion. I was also not able to try larger images, as the pre-trained models and code were limited to 256- or 512-pixel square images.

VI. Deliverables

My work is in the following location on GitHub: `https://github.com/nikbhatt-cu/GANBench.git`.

VII. Self-Evaluation

Accomplishments: I was able to create a reusable library, which I believe will be valuable to GAN researchers because it lessens the effort of comparing algorithms, while providing consistent metrics. I was also able to critically evaluate three different GANs using an unseen dataset to assess their applicability to novel data; the results would indicate that models are somewhat overfit to the Places dataset.

Challenges: It was challenging to read, understand, and work with these research papers and their code. A lot of the code doesn't work without modification, or it requires special hardware and software that is not accessible to everyone. At one point, I hoped to drop GAN models into my library and use them directly. I thought so, because fundamentally each GAN takes an input image and a mask, and produces an output image, so a simple API seemed possible. However, the models are tightly bound to (and dependent on) the accompanying python code in each repository.

I also had hoped to use Facebook's Detectron2 library to make the segmentation masks and I spent a long time trying to get it to run. However, that library is designed to work in Docker containers running Linux, or on desktop computers running Intel CPUs. Fortunately, the Open Images dataset included segmentation masks so I could still run my object removal tasks.

What I learned: I learned a lot about GAN-based inpainting, their strengths and weaknesses, the metrics used for assessing them, and potential issues with those metrics. I was very disappointed to see that despite the impressive results cited in the papers (both sample images and metrics), most of the example repairs I examined are not actually usable.

VIII. Course Feedback

I have two suggestions for future iterations of this class. Both are related to the student presentations. I noticed that presentations would often run long, which either gave the second student too little time to present or caused the class to run over its allotted time. Presenters (understandably) tend to lose track of time, so it's easy for them to run long.

My recommendation is that the professor or the IA actively monitor the lengths of presentations to keep students on track. Remind them at the start that they have 30 minutes, and then give a verbal five-minute warning (interrupting as needed) and a one-minute warning (if necessary).

If the presentations continue to run long, then I suggest that the professor walk through her slides before turning things over to the students. That way, important course information and questions are sure to be handled. The lecture section can also be time-bound, to ensure that the presenters have enough time.

IX. References

[1] Xiang, et al. Deep learning for image inpainting: A survey. In Pattern Recognition, Volume 134, Feb 2023. https://www.sciencedirect.com/science/article/abs/pii/S003132032200526X

[2] Open Images Dataset v7: https://storage.googleapis.com/openimages/web/download_v7.html. Last accessed April 21, 2024.

[3] Yi, et al. Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting. In *Computer Vision and Pattern Recognition*, 2020. http://openaccess.thecvf.com/content_CVPR_2020/html/Yi_Contextual_Residual_Aggregation_for_Ultra_High-Resolution_Image_Inpainting_CVPR_2020_paper.html

[4] Zhao, et al. Large Scale Image Completion via Co-Modulated Generative Adversarial Networks. In *International Conference on Learning Representations* (ICLR), 2021. https://arxiv.org/abs/2103.10428

[5] Sargsyan, et al. MI-GAN: A Simple Baseline for Image Inpainting on Mobile Devices. In *International Conference on Computer Vision* (ICCV), 2023. https://openaccess.thecvf.com/content/ICCV2023/papers/Sargsyan_MI-GAN_A_Simple_Baseline_for_Image_Inpainting_on_Mobile_Devices_ICCV_2023_paper.pdf

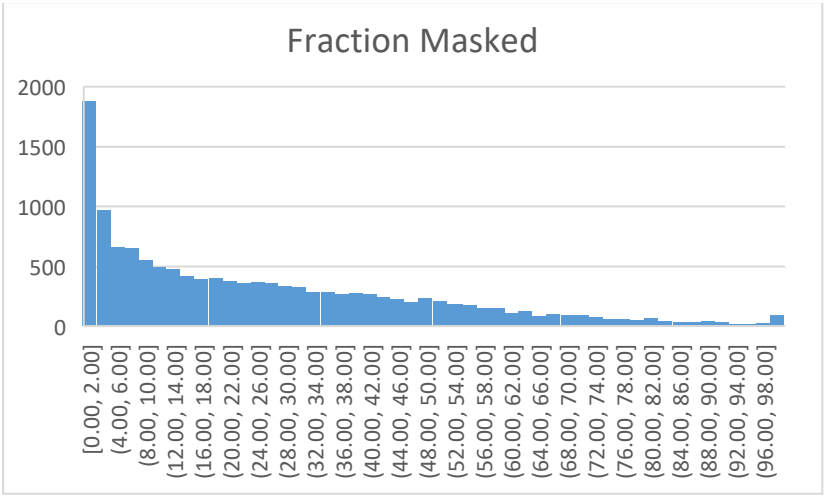
[6] Suvorov, et al. Resolution-robust Large Mask Inpainting with Fourier Convolutions. In *Winter Conference on Applications of Computer Vision*, 2022. https://openaccess.thecvf.com/content/WACV2022/papers/Suvorov_Resolution-Robust_Large_Mask_Inpainting_With_Fourier_Convolutions_WACV_2022_paper.pdf

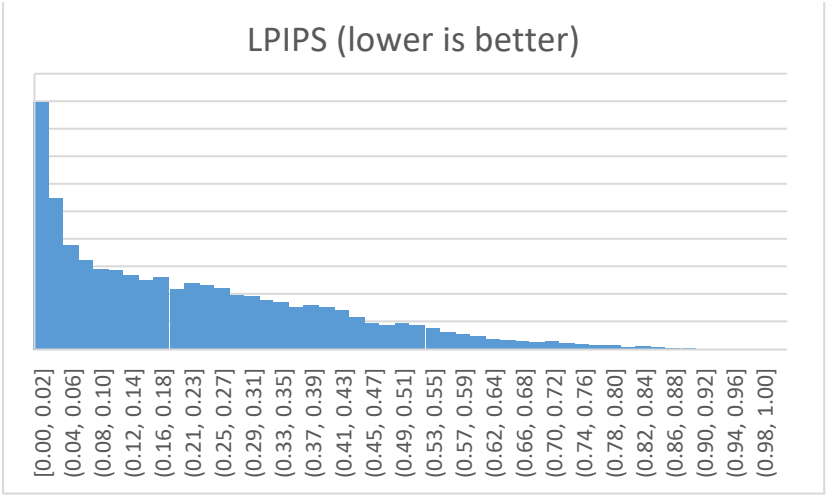
[7] Jain, et al. Keys to Better Image Inpainting: Structure and Texture Go Hand in Hand. In: *IEEE Winter Conference on Applications of Computer Vision*, 2023. https://openaccess.thecvf.com/content/WACV2023/html/Jain_Keys_To_Better_Image_Inpainting_Structure_and_Texture_Go_Hand_WACV_2023_paper.html

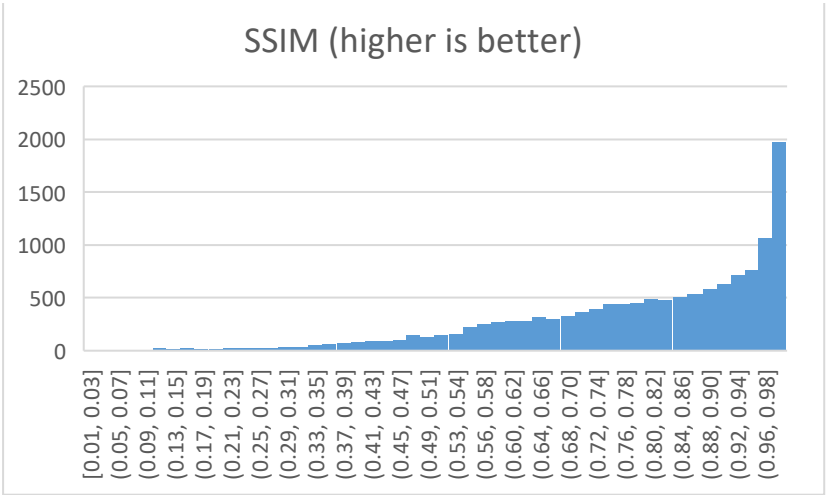
[8] Places365-Standard: http://places2.csail.mit.edu/download-private.html. Last accessed April 21, 2024.

[9] Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study. In Journal of Computer and Communications > Vol.7 No.3, March 2019. https://www.scirp.org/journal/paperinformation?paperid=90911.

X. APPENDIX: HISTOGRAMS FOR MI-GAN

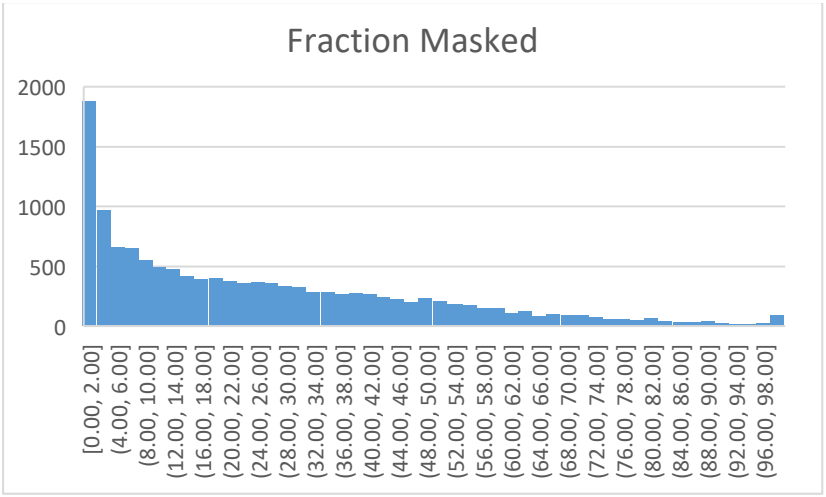


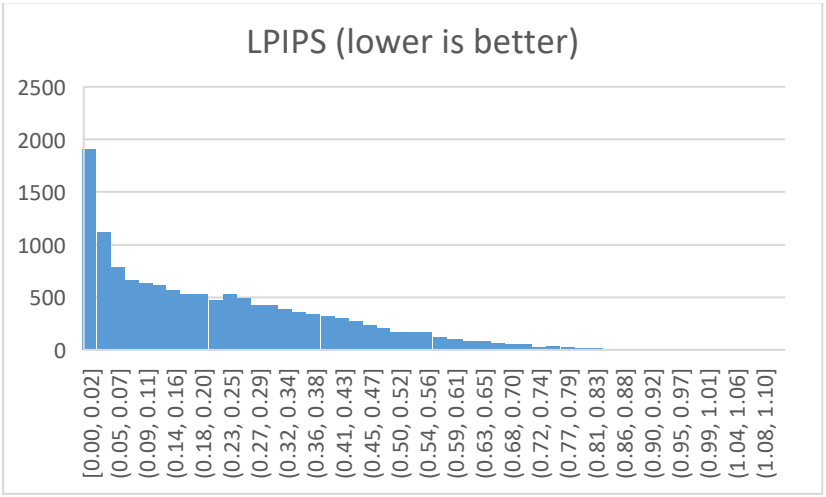


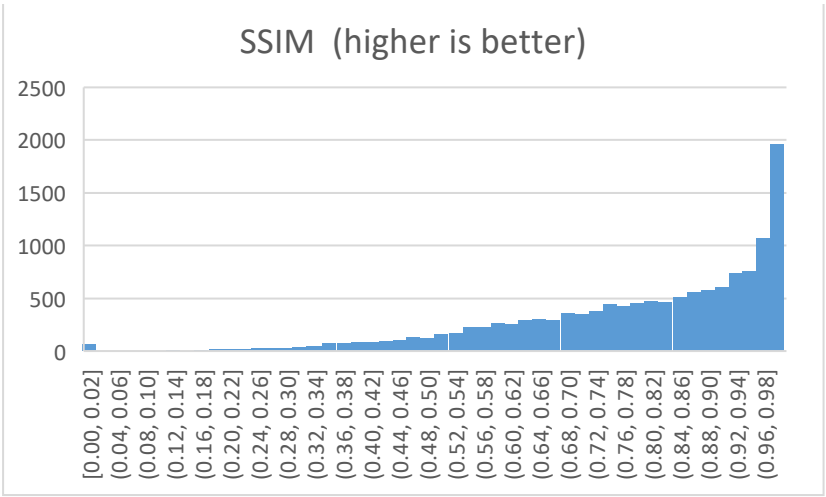


I generated these histograms in Excel, using the CSV from CoModGAN using the Open Images dataset. One clear relationship is that smaller masks result in better LPIPS and SSIM results. There is also a sharp drop-off after the first bin, indicating that the quality of the repair initially declines precipitously with mask size before slowly declining with larger masks.

XI. Appendix: Histograms for CoModGAN

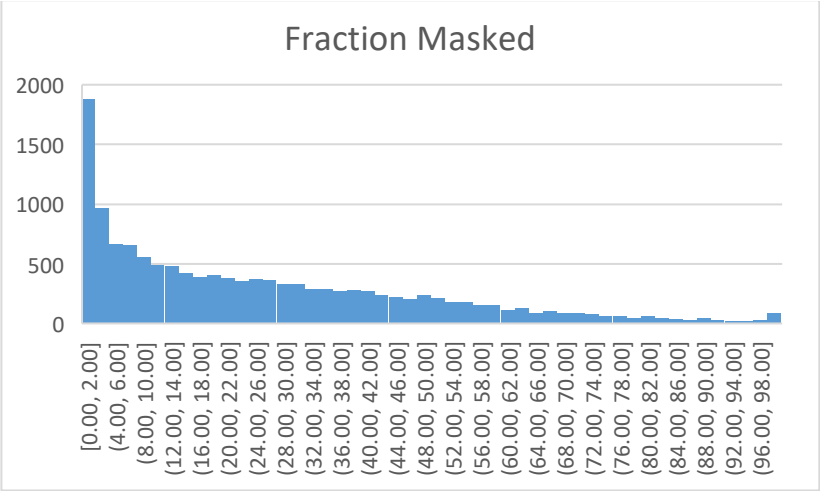


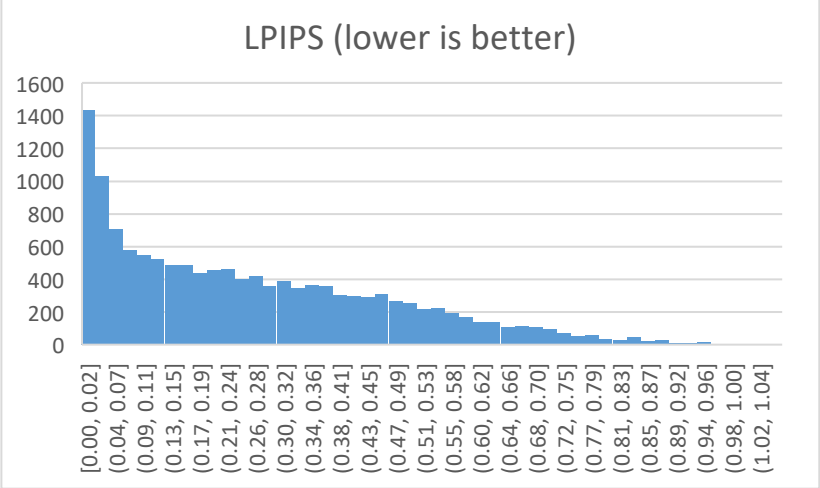


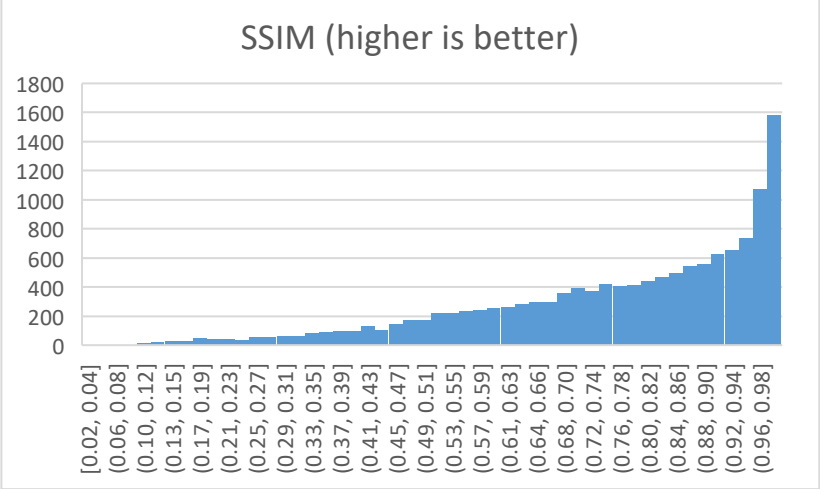


CoModGAN shows similar results to MI-GAN, which is not surprising, as MI-GAN uses CoModGAN as a teacher network.

XII. Appendix: Histograms for HiFill







HiFill is similar but shows less of a drop off as the mask size increases (at least with LPIPS).