Forecasting infrastructure deterioration with inverse GANs

Eric Bianchi[a] and Matthew Hebdon[b]

[a]Virginia Polytechnic and State University
[b]The University of Texas at Austin

ABSTRACT

Synthetic image generation using deep learning techniques like generative adversarial networks (GANS), has drastically improved since its inception. There has been significant research using human face datasets like FFHQ, and city-semantic datasets for self-driving car applications. Utilizing latent space distributions, researchers have been able to generate or edit images to exhibit specific traits in the resultant images, like face-aging. However, there has been little GAN research and datasets in the structural infrastructure domain. We propose an inverse-GAN application to embed real structural bridge detail images and incrementally edit them using learned semantic boundaries. Corrosion/non-corrosion and various steel paint colors were among the learned semantic boundaries discovered using the Interface-GAN methodology. The novel dataset used was procured from extracting hundreds of thousands of images from the Virginia Department of Transportation (VDOT) bridge inspection reports and was trained using the styleGAN2 generator. The trained model offers the ability to forecast deterioration incrementally, which is valuable to inspectors, engineers, and owners because it gives a window into the future on how and where damage may progress. As bridge inspectors typically review bridges every two years, this forecast could reinforce decisions for action or in-action.

Keywords: Inverse-GAN, bridge inspection, deep learning, structural health monitoring, infrastructure aging, GAN, Auto-encoder

Figure 1. Examples of semantic deterioration forecasting. Where the first image on the left is the real image and the following image is the embedded version of that image. The subsequent sequence of images are increments towards the target corrosion boundary by a relative gain of (3.0, 4.0, 5.0, and 6.0) respectively.

1. INTRODUCTION

Inspection of key transportation components, such as bridges, is a necessary and important facet to maintain a safe and functioning society. As more bridges approach and reach the end of their design-life, it becomes critical to monitor deterioration, assess potential risks, and triage individual components or entire structures which

Further author information: (Send correspondence to Eric Bianchi)
Eric Bianchi: E-mail: beric7@vt.edu

require action. Bridge inspectors must inspect each bridge, at a minimum, every two years, and sometimes more often to ensure safety of the public. Bridges pose a unique challenge in that inspectors must cover immense areas - often which are difficult to access and potentially during unfavorable weather conditions.

Research and technology is evolving to enhance the inspection process and meet this constant demand. The advent of machine learning and the rise in the use of Unmanned Aerial Vehicles (UAVs) has been at the forefront of this progress. Research has been focused on improving the means to detect damage such as cracking in concrete,[1] corrosion on steel,[2] and other common damage types.[3,4] After detecting the damage, inspectors are required to measure, quantify, and report the severity of the damage. There has also been research into methods for automatically quantifying the extent of the damage through semantic segmentation[3,5] or volumetric filling.[6,7] The aggregation of the damage for each portion of the bridge is typically compared to past inspections to measure the rate of change of its deterioration. Inspectors must use this quantifiable information to make an inference on whether or not maintenance or rehabilitative action is needed before the next inspection.

Because inspectors must use current and past information to make decisions to improve conditions years in advance, it would be advantageous to have a model that could forecast deterioration and its progression. The authors have chosen the word 'forecast' over 'prediction' due to the nuance that a forecast holds less certainty. Despite this perceived advantage, there has not been research into the visual forecast of a future condition based on a prior condition. In this paper, we explore the synthetic forecasting and stylization of damage onto existing and synthetically generated bridge images. A forecast model was trained using hundreds of thousands of image examples collected from bridge inspection reports. The forecasting model provides several advantages: [1] gains a window into the potential future based on the current damage state without further maintenance, and [2] generates synthetic data which can be used for virtual applications and expanding existing structural bridge datasets through synthetic data augmentation. As there has been little to no research on the forecasting of damage for infrastructure inspection, this paper uses grounded methods for an initial pass at the novel concept.

The main contributions of this research are the following:

- A forecasting model for damage deterioration and damage progression, with the ability to incrementally age synthetic and real image data.
- Implementation and application of synthetic data stylization.

2. RELATED WORK

Understandably, there has been little effort into predictive damage synthesis for infrastructure inspection. This may be due to the fact that generative networks have only been used since 2014,[8] or because forecasting damage based on images is not a natural task for a structural bridge inspector. For example, the closest forecasting application found was a researcher who used a cycle-GAN[9] network to add or remove cracks from a small patch of concrete.[10] While predicting damage may seem fantastical, it may not be as far-fetched as it sounds.

Noise vector generative networks, like styleGAN[11] and styleGAN2,[12] have proven that generating hyper-realistic images are possible when given enough training data, iterations, and a well structured network architecture. Within this research domain, some of the most common training datasets used are focused on faces (FFHQ,[11] celebA[13]), bedrooms,[14] churches,[14] city scenes for self-driving car applications,[15] and generic natural mountainous scenes from Flikr.[16] Manipulation of these synthetically generated images from their, embedded, latent space representation, is also a well researched area.[17] How well an image can be manipulated depends on how well the latent space is disentangled. For example, synthetically generated faces from the FFHQ dataset have been manipulated to change the color of the target's hair.[18] If only the color of the target's hair changes, and nothing else, than that control is well-disentangled because no other attribute, like facial expression or eye color, was also affected when making that change (Figure 2).

In addition to focused attributes like hair color, global attributes like the age or gender may also be disentangled and manipulated. While editing synthetically generated images may be useful, there are potentially many more use-cases for manipulating existing images. However, manipulating existing images is a more challenging task because one has to fit a real image into the learned embedded space that best matches a target region in the

Figure 2. Example of well-disentangled semantic editing from StyleSpace[18]

latent space distribution. One way to do this is optimizing the projections from the latent space and comparing the generated synthetic image to the target image until convergence. Another method, is by training an encoder network to encode images into the latent space and re-project or decode them using a pre-trained generative network like styleGAN2.[12] The process of inverting an image into the latent space, where semantic manipulations are possible, and re-projecting the image using a fixed generator is referred to as GAN-Inversion,[17] Figure 3. This may sound very similar to an auto-encoder network, however auto-encoders often optimize both the encoder and generator, therefore the generator is not fixed. A few notable and recent successful auto-encoders are networks like the swapping auto-encoder[16] or adversarial latent-auto-encoders.[19]

Figure 3. Example of a GAN-Inversion network from In-Domain GAN Inversion for Real Time Editing[20]

The authors were inspired by the fidelity of the recent work on real image inversions and global aging manipulation. Coupled with the ability to incrementally edit an image based on a target attribute, like age. This technique was applied to forecasting corrosion damage on typical structural bridge components. To the authors knowledge this is the first application of this technique in the structural inspection domain, and offers an alternative data domain for researchers to include in the presentation of their novel networks.

3. METHODOLOGY

A high-level overview of networks and techniques for editing and generating images are compiled in Table 2 of the Appendix. The table comprises four general network categories: Vector-Image Generative Adversarial Networks, Image-Image Translations, Image-Image Style Transfer, and Image-Image GAN-Inversion. The author chose to use GAN-Inversion, because it met all the desired qualities - the ability to: randomly generate images, transfer in-domain global styles, interpolate between images, amplify projection edits, semantically target regions, and edit real images.

GAN-inversion required training an encoder to embed real images into the latent space, and a generator to decode the image from the latent space. The generator was trained using StyleGAN2, and the latent codes were taken from the $\mathcal{W}$ space, a more disentangled $\mathcal{Z}$ space. Feed-forward auto-encoders and iterative methods were used for inverting real images into the latent space for real image-editing. Semantic boundaries, like corrosion/non-corrosion, and color, were discovered using the methodology outlined in Interface-GAN.[21] These semantic boundaries were used to edit the real and synthetic embedded images.

3.1 Data Extraction

The images were extracted with permission from every available bridge inspection report within the Virginia Department of Transportation (VDOT) database. Over +300GB of data were extracted from the reports using

a custom image extraction algorithm. Given this large quantity of data, there was a need to sort out the useful data. Many images extracted from the bridge inspection reports did not contain bridge components, including, VDOT seals, CAD drawings, company logos, etc. The authors trained a binary bridge/non-bridge image classifier (Figure 4) to automate the sorting process. The authors incrementally trained the classifier starting first with 2000 bridge and 2000 non-bridge images, and was stopped after a few thousand image predictions. To aid in this incremental supervision the images were sorted into bins of classifier confidence, [50-75%, 75-90%, 90-95%, and 95-100%]. These bins helped the authors quickly find false-positive and false-negative images (as they were more likely to be in lower confidence bins), and helped share insight into how the average confidence of its inference predictions improved after each incremental training session.

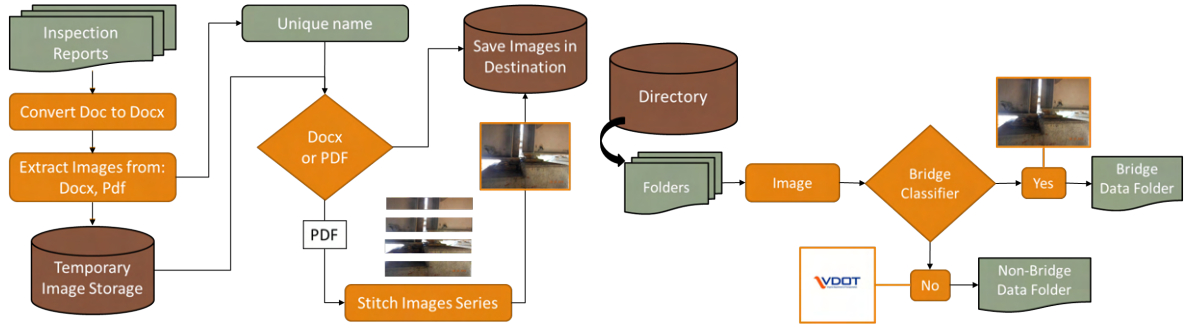


Figure 4. (a) Pre-processing Images (b) Sorting Images

This incremental training process continued until the 'bridge - non-bridge' classifier had a ¡1% error and the majority of the classification confidences were 90% or above for both bridge or non-bridge images. When the extraction process and sorting was completed there were over 300,000 images which were classified as containing bridge components. Of these, roughly half (about 150,000) images were selected from the higher confidence bridge images to be used used for training the StyleGAN2 generator. In an effort to focus the highly diverse image dataset to a more consistent subset of images, 4,000 images focused on specific structural bridge details were selected to fine-tune the model.

3.2 Training the Decoder

The first step was training a styleGAN2 generator (decoder). The authors used a batch size of 2, and an input-output image resolution of 512x512. This required just under 16 GB of GPU memory (spread across two 8 GB GPUs), and a large amount of CPU memory for calculating evaluation metrics like the frechet inspection distance (FID)[22] score at designated iteration checkpoints. The FID score is a metric which measures the generated image diversity.[17] There were two main datasets used when conducting these experiments. These datasets are summarized in Table 1.

At each evaluation checkpoint, useful metrics like the frechet inception distance (FID) scores[22] and the learned perceptual image patch similarity (LPIPS)[23] were calculated. FID score measures the diversity and likeliness of one dataset to another as a whole, and the LPIPS measures the images' perceptual quality. The models were allowed to run until the models' metrics and losses converged or right before the model became unstable. During the fine-tuning of the VDOT-detail dataset, there was a point where the model losses began to diverge, so the authors took several checkpoints prior to that inflection point. Training was stopped after these inflection points were reached, indicated by the number of days listed in Table 1, and all training data used two 8 GB V100 GPUs. Once training was completed for all of the experiments, the best VDOT-detail model checkpoint was chosen to generate 10,000 synthetic images from the $\mathcal{Z}$ latent space and $\mathcal{W}$ latent spaces.

3.3 Finding Semantic Boundaries in Latent Space

The latent boundaries were discovered following the methodology and code outlined in Interface-GAN.[21] The methodology is summarized as synthesizing a large enough amount of data from the generator's latent space to

Table 1. Summary of Datasets

Dataset	Image Size	Image count	Training Time	Description
VDOT-collection	512x512	150,000 images	(2) V100 GPUs, 12 days	All images from VDOT bridge inspection reports.
VDOT-detail	512x512	3,817 images	(2) V100 GPUs, 3 days	Images focused on structural bridge details.

sort 10,000 images into each desired semantic category. Then a simple classifier is trained for each category. The classifier model is run on the synthesized data to output confidence predictions. These predictions are paired with their respective latent space vectors. Together the latent vectors and the classifier predictions are used to train a support vector machine (SVM) and determine the normal unit vector which would push any given vector toward that boundary. Therefore, any given latent space image would be pushed towards the desired target category.

Roughly following this methodology, 10,000 images were synthesized from the $\mathcal{Z}$ space, and only 250 images of the most obvious, undeniable examples were sorted into specific categories (corrosion, non-corrosion, and steel paint colors). Some of these examples are shown in Figure 5. The authors believed that they could still get strong model predictions from the classifier with these two reduced samplings. Through experimentation on this particular dataset, the authors did find that the semantic editing qualities were not hurt by using a smaller sample size. The sorted data was used to train an image classifier to predict the confidence of each of the categories. These confidence scores coupled with the latent space codes of the synthetic images were used to build the SVM. By using the scores from the classifier, the model was able to establish a semantic boundary in the latent space which corresponds to each of the categories. Using these learned semantic boundaries, we were able to incrementally move images towards or away from these semantic boundaries while maintaining the original image structure.

Figure 5. Examples of corrosion and non-corrosion images used for training a latent semantic boundary

3.4 Choosing an Encoder

In order to manipulate and edit a real image using the learned semantic boundaries one must encode the image to a generator's embedded space. The success of the embedding depends on the diversity and robustness of the generator's latent space. There were two options which were explored. The first option used an Inverse-GAN auto-encoder network which trained an encoder to embed images into an existing latent space. The encoder network chosen was pixel2style2pixel.[24] The decoder network chosen was the styleGAN2 network,[12] pre-trained on the bridge images. The second option was an iterative method, using styleGAN2's own latent space codes

coupled with the decoder image generation to project predictions of the target image. Because this was an iterative method, the inference for arriving at the inverted image was much slower. However, this method does not require training an additional model, and it can be quite effective if the latent space is robust. Robust, in this case, meaning that the latent space is diverse and consistent. Diverse because it must be able to handle variations, yet consistent because it must be able to accurately place a foreign image into a region of similarly learned examples from its training.

For the optimization method, the authors began with the average latent vector. This method uses the projections from the latent space to make its next step. After 1000 iterations the optimization usually had converged on a latent vector for that image. Notably, this does guarantee that the optimization did not converge to a local minimum.[17] Through experimentation, it was found that the pixel2style2pixel[24] method was not able to converge and encode the images in an effective way, while the styleGAN2 optimization method was able to more realistically project the target image into the embedded latent space (Figure 6). The authors believe that the pix2style2pixel network needed a larger more consistent dataset for the training to be effective. While the 4000 bridge image dataset was focused on structural bridge details, it was still too diverse. If, perhaps, the dataset was composed of 4000 images of a single structural detail (like a bridge bearing), then the outcome of the training may have been more favorable for the pixel2style2pixel network.

Figure 6. Examples of failed pix2style2pix embedding convergence

4. RESULTS

The results section compares the generator and reconstruction quality of the two model variations: VDOT-collection and fine-tuned. The quality of the generator, reconstruction (inversion), was evaluated quantitatively and qualitatively, with points of failure identified and reasoned. The semantic editing of synthetic and real embedded images are qualitatively represented in the section as well.

4.1 Generator Quality

For generated image quality, the FID scores were used. The best FID scores for the VDOT-detail were between 15-16, and the best FID scores for the VDOT-collection were between 3-4. This result indicates that the VDOT-collection model generated images of higher quality than the fine-tuned model compared to its training data. Examples of the generated images for the VDOT-collection and the fine-tuned model are shown in Figures 7 and 8 respectively. Of the 10,000 randomly generated images for the VDOT-collection, many composed of entire bridge structures (Figure 7), culverts, pavement, and concrete walls. This was a reflection of the data which was within. The fine-tuned model had only a few hundred culverts within the VDOT-detail dataset, with the majority being structural bridge details. While the FID scores do measure the likeliness of one dataset to another as a whole, it does not necessarily directly translate into the perceptibly and fidelity of a generated dataset.[17] Additionally, the generative image quality, quantitatively measured by FID, does not translate to the model's ability to have a strong reconstruction accuracy.

4.2 Reconstruction Quality

The reconstruction accuracy is typically measured using the peak to signal noise ratio (PSNR) or the structural similarity (SSIM)[25] metrics. The goal was to forecast deterioration onto structural bridge details. Therefore, the reconstruction accuracy's for 20 randomly sampled real images of structural details were calculated for both the VDOT-collection and VDOT-detail fine-tuned model. The PSNR and SSIM were (16.0, 0.428) for VDOT-collection and (16.7, 0.452) for VDOT-detail fine-tuned. These quantitative results reinforces the qualitative results, shown in Figure 9, which indicate the better reconstruction ability for structural details of the fine-tuned

Figure 7. Examples of synthetically generated images from the model trained on the VDOT-collection dataset

Figure 8. Examples of synthetically generated images from the model fine-tuned on the VDOT-detail dataset

model. The authors believe that the large number of global scenes, like images of entire bridge captures or culverts, in the VDOT-collection model, overshadowed sparse subset of structural bridge details. This made it difficult for the latent space re-projection optimization for the VDOT-collection model compared to the fine-tuned model.

4.3 Semantic Editing

Images were synthetically edited using the trained latent boundaries and randomly generated latent noise vectors. The generator, StyleGAN2, offers the $\mathcal{W}$ latent space, which is a more disentangled representation of the $\mathcal{Z}$ space. The trained boundaries and vector gains used the $\mathcal{W}$ latent space. Figure 10 helps to visualize the difference between editing in the two spaces. Figure 10 (a) is a randomly generated image from the $\mathcal{Z}$ space. Figure 10 (b) is the same randomly generated image from the $\mathcal{Z}$ space after being transformed into the $\mathcal{W}$ space.

In Figure 10 (a), as the corrosion vector is being applied to the images from left to right in the $\mathcal{Z}$ space, there are noticeable tangential changes also occurring. The first observed tangential change is that the image appears to be zoom in as the corrosion vector is applied. This is most likely due to the fact that images which are corroded tend to be taken closer, since there are issues that need to reviewed. The second tangential change is that a ruler begins to materialize. This is explained by the fact that most inspectors give perspective to deteriorated structural details by putting a ruler, hammer, or other common object like a pen in the reference frame. In addition to these tangential changes, more global structural changes are also noticeable. However, in Figure 10 (b) the tangential and most of the global image structure remains constant as the structural detail in the image is progressively aged. More examples of the progressive deterioration forecasts for the $\mathcal{W}$ space are

Figure 9. Comparison of embedded images from the model trained on only the VDOT-collection and the model fine-tuned with the VDOT-detail dataset. Where the top row is the original image, the middle row are the embedded images from the VDOT-collection, and the bottom row are the embedded images from model fine-tuned on the VDOT-detail dataset.

(a)

(b)

Figure 10. Progressive corrosion from left to right, at unit increments from 0 to 3.0 over the course of five images. (a) and (b) are vectors produced from the latent $\mathcal{Z}$ space and $\mathcal{W}$ space respectfully.

shown in Figure 17 in the Appendix. Examples of the color swapping from its base color to brown are shown in Figure 18 in the Appendix.

Quantitatively evaluating how disentangled latent style spaces is another way to evaluate the success of a generative network. A common way to measure the success of the disentanglement of the latent spaces is to measure the performance of classification tasks within the latent space.[18] Should the model be able to predict a class based on a latent space boundary then one can argue that the model has disentangled its boundaries well. With this dataset being unlabeled for attributes and objects within the images, this metric was not used. While it would take time to label the structural data in this way in the future, it would provide useful insights into the performance of the generator.

4.4 Embedded Semantic Editing

The images were embedded using the iterative latent space optimization method. The learned semantic boundaries were corrosion and steel paint color (brown, green, and grey). An example of color changing is shown in Figure 11 and Figure 16 of the Appendix, and an example of the corrosion forecast is shown in Figure 1 and

Figure 15 of the Appendix. The color alterations appeared to have been easily transferable and apparent. While there is no ground-truth for a future state, results show a relatively likely progression of deterioration.

From a qualitative perspective, there were three main issues, [1] blur - Figures 15 and 16 of the Appendix, [2] pose - row 1 of Figure 16 of the Appendix, [3] object warping (i.e. merging of structural details) - bottom row of Figure 16 of the Appendix. The blurriness of the embedded real images was most likely due to the latent space optimization process. The pose miss-alignment and the object warping/merging was most likely a result of being stuck at some local minimum during the optimization process.

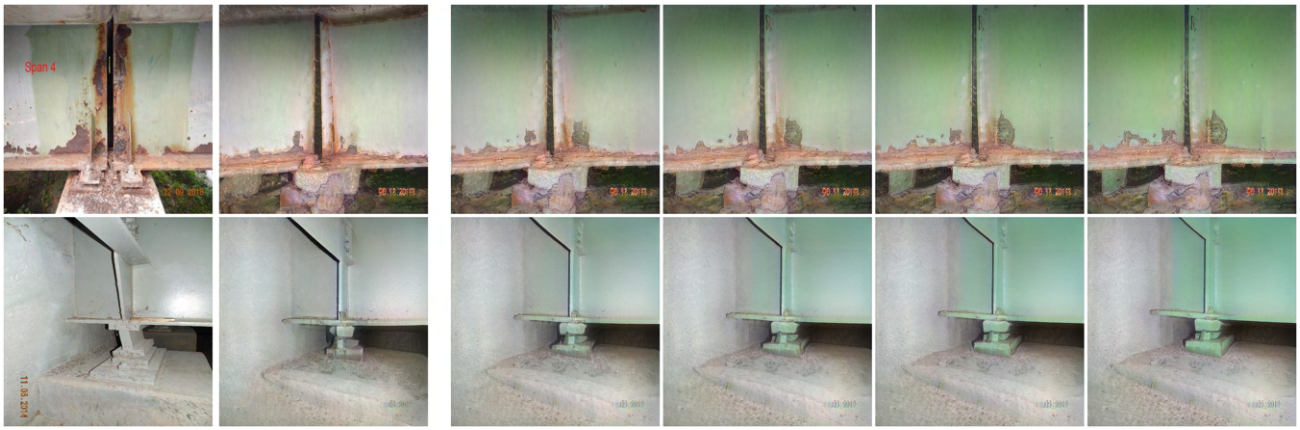


Figure 11. Examples of semantic color swapping (base color to green). Where the first image on the left is the real image and the following image is the embedded version of that image. The subsequent sequence of images are increments towards the target color boundary by a relative gain of (3.0, 4.0, 5.0, and 6.0) respectively.

4.5 Generator Failed Cases

There were a significant number of examples of what the authors consider failed-cases. For the synthetically generated images there were four main issues [1] unrecognizable structural details as shown in Figure 12, [2] recognizable - but highly unrealistic details as shown in the bearing in Figure 12 (second row, image 2), [3] image tear-artifacts seen in Figure 10 (a) image 5 in the top left corner, and [4] particular issues with angled diaphragm bracing (second row, image 5) Figure 12.

Figure 12. Examples of failed synthetically generated images

5. DISCUSSION

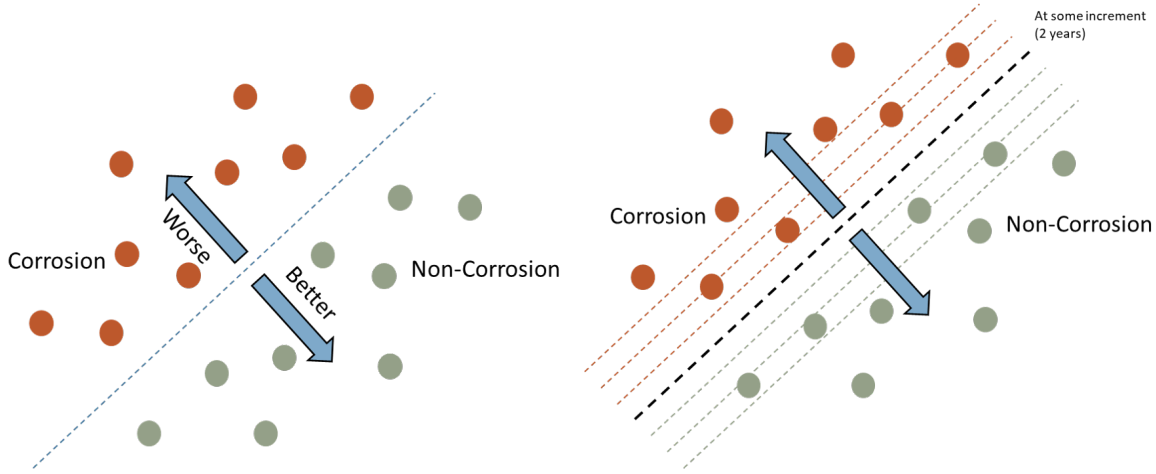
The concentrated, VDOT-detail dataset, fine-tuned on the VDOT-collection generated better fidelity results than the model trained on the VDOT-collection dataset on its own. Qualitatively, the synthesized structural images did reach the same level of hyper-realistic results as the much larger and more consistent datasets like FFHQ,[11] celebA,[13] etc. produced. However, with more of the 'structural detail' focused images, we would expect more realistic synthesized imagery. While the produced results are not the same quality as the larger and more widely used datasets, like FFHQ, it does show a promising avenue for a new application of GANs and inverse-GANs in the context of structural inspection analysis.

5.1 Future Avenues

There were three avenues which the authors have identified for continuing this application research [1] measurable semantic layering, [2] auto-encoders, and [3] memory-based networks with auto-encoders.

5.1.1 Measurable Semantic Layering

This paper applied inverse-GANs to forecast visual structural damage onto typical structural components based upon a current damage state. However, the arbitrary unit gain applied to the embedded images were not specific to any time domain. This begs the question: How can we couple each unitless increase with some measurable unit gain of time. Instead of discovering a single semantic boundary (13 (a)), one would discover a series of boundary layers (13 (b)) indicating layers of time. This concept of semantic layering has been researched when projecting certain age ranges onto human-faces.[26]



(a) (b)

Figure 13. (a) Arbitrary Incremental Corrosion (b) Defined Incremental Corrosion

5.1.2 Auto-encoders

A second option was explored for real image manipulations. However, this was not an inverse-GAN model it was an auto-encoder model which separated structure and style into two vectors - swapping-auto-encoder.[16] For this network, the operator was able to swap out the style vectors from a source image and insert it onto a target image while preserving the content or structure of the target image. One of the distinct advantages of this network architecture was that it was able to maintain the integrity of the overall structure of the image. For this application, only the embedded style vector would be manipulated away from learned semantic boundaries. Initial experiments on the VDOT-detail dataset show that this form of auto-encoder structure provides superior reconstruction ability (Figure 14) than the reconstruction through optimization of the styleGAN2 latent space. While the swapping-auto-encoder paper does suggest that the latent style space vectors can be disentangled for a learned domain, the latent style space was not yet tested for the VDOT-detail dataset.

Figure 14. Examples of real embedded images from swapping auto-encoder training. Top row are real images and the bottom row are their respective embedded versions of the image.

5.1.3 Memory-based Networks with Auto-encoders

Another approach to the unit time gain (semantic layering), is by leveraging a time-series of images to make a future prediction. Given a time-series of images from a specific bridge component location, it may be possible to more accurately forecast a certain component's damage. This concept leverages networks which use a series of sequential inputs to predict the next object in the sequence. These over-arching frameworks such as Recurrent Neural Networks RNN,[27] Long short-term memory convLSTM[28],[29] combined with an auto-encoder structure have been used for predicting next-image sequences.[30] Imagine if there was a database time series of thousands of bridge component locations collected over many inspection years. Using the built-up prior knowledge, an inspector could forecast next-inspection images based on reliable and real bridge deterioration progressions taken at common and known intervals.

6. CONCLUSION

This paper presented a novel GAN application to semantically edit typical structural bridge detail images taken by inspection engineers. Through common inverse-GAN methodologies real and synthetic images had incremental deterioration forecasts applied to them. These types of forecast capabilities have potential to integrate into a bridge inspector's report. If the owners or the engineers had insight into how the bridge may look at the time of their next inspection (i.e. 2 years into the future), then it may provide the catalyst for repair or the confidence for in-action. The trained model also enables the ability to synthetically generate data which could be used for other computer vision tasks related to this domain such as: object detection, semantic segmentation of materials, or damage detection. After comparing the swapping-auto-encoder and stylGAN2 optimization image reconstructions, it was clear the auto-encoder had better re-projections. Because of the need to preserve the original structure and detail of the image, the authors believe that future research, in this domain of forecasting deterioration, should investigate the use of auto-encoders.

REFERENCES

[1] Dorafshan, S., Thomas, R. J., and Maguire, M., "Sdnet2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks," *Data in Brief* **21**, 1664 – 1668 (2018).

[2] Cha, Y., Choi, W., Suh, G., Mahmoudkhani, S., and Büyüköztürk, O., "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," *Comput. Aided Civ. Infrastructure Eng.* **33**, 731–747 (2018).

[3] Hoskere, V., Narazaki, Y., Hoang, T. A., and Spencer, B. F., "Madnet: multi-task semantic segmentation of multiple types of structural materials and damage in images of civil infrastructure," *Journal of Civil Structural Health Monitoring* , 1–17 (2020).

[4] Hoskere, V., Narazaki, Y., Hoang, T. A., and Spencer, B., "Vision-based structural inspection using multi-scale deep convolutional neural networks," *ArXiv* **abs/1805.01055** (2018).

[5] Cheng, J., Xiong, W., Chen, W., Gu, Y., and Li, Y., "Pixel-level crack detection using u-net," *TENCON 2018 - 2018 IEEE Region 10 Conference* , 0462–0466 (2018).

[6] Rau, J., Hsiao, K., Jhan, J., Wang, S. H., Fang, W., and Wang, J., "Bridge crack detection using multi-rotary uav and object-base image analysis," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **42**, 311–318 (2017).

[7] Kalfarisi, R., Wu, Z., and Soh, K., "Crack detection and segmentation using deep learning with 3d reality mesh model for quantitative assessment and integrated visualization," *Journal of Computing in Civil Engineering* **34**, 04020010 (2020).

[8] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial networks," (2014).

[9] Zhu, J., Park, T., Isola, P., and Efros, A. A., "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR* **abs/1703.10593** (2017).

[10] Zhang, K., Zhang, Y., and Cheng, H.-D., "Self-supervised structure learning for crack detection based on cycle-consistent generative adversarial networks," *Journal of Computing in Civil Engineering* **34**, 04020004 (2020).

[11] Karras, T., Laine, S., and Aila, T., "A style-based generator architecture for generative adversarial networks," *CoRR* **abs/1812.04948** (2018).

[12] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T., "Analyzing and improving the image quality of stylegan," *CoRR* **abs/1912.04958** (2019).

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

[14] Yu, F., Zhang, Y., Song, S., Seff, A., and Xiao, J., "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365* (2015).

[15] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B., "The cityscapes dataset for semantic urban scene understanding," in [*Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], (2016).

[16] Park, T., Zhu, J., Wang, O., Lu, J., Shechtman, E., Efros, A. A., and Zhang, R., "Swapping autoencoder for deep image manipulation," *CoRR* **abs/2007.00653** (2020).

[17] Xia, W., Zhang, Y., Yang, Y., Xue, J., Zhou, B., and Yang, M., "GAN inversion: A survey," *CoRR* **abs/2101.05278** (2021).

[18] Wu, Z., Lischinski, D., and Shechtman, E., "Stylespace analysis: Disentangled controls for stylegan image generation," *CoRR* **abs/2011.12799** (2020).

[19] Pidhorskyi, S., Adjeroh, D. A., and Doretto, G., "Adversarial latent autoencoders," *CoRR* **abs/2004.04467** (2020).

[20] Zhu, J., Shen, Y., Zhao, D., and Zhou, B., "In-domain GAN inversion for real image editing," *CoRR* **abs/2004.00049** (2020).

[21] Shen, Y., Yang, C., Tang, X., and Zhou, B., "Interfacegan: Interpreting the disentangled face representation learned by gans," *CoRR* **abs/2005.09635** (2020).

[22] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S., "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR* **abs/1706.08500** (2017).

[23] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O., "The unreasonable effectiveness of deep features as a perceptual metric," *CoRR* **abs/1801.03924** (2018).

[24] Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., and Cohen-Or, D., "Encoding in style: a stylegan encoder for image-to-image translation," *CoRR* **abs/2008.00951** (2020).

[25] Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E., "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing* **13**(4), 600–612 (2004).

[26] Shoshan, A., Bhonker, N., Kviatkovsky, I., and Medioni, G., "Gan-control: Explicitly controllable gans," (2021).

[27] Elman, J. L., "Finding structure in time," *Cognitive Science* **14**(2), 179–211 (1990).

[28] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural Comput.* **9**, 1735–1780 (Nov. 1997).
[29] Song, H., Wang, W., Zhao, S., Shen, J., and Lam, K.-M., "Pyramid dilated deeper convlstm for video salient object detection," in [*Computer Vision – ECCV 2018*], Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., eds., 744–760, Springer International Publishing, Cham (2018).
[30] Zhou, Y., Dong, H., and El Saddik, A., "Deep learning in next-frame prediction: A benchmark review," *IEEE Access* **8**, 69273–69283 (2020).
[31] Isola, P., Zhu, J., Zhou, T., and Efros, A. A., "Image-to-image translation with conditional adversarial networks," *CoRR* **abs/1611.07004** (2016).
[32] Luan, F., Paris, S., Shechtman, E., and Bala, K., "Deep photo style transfer," *CoRR* **abs/1703.07511** (2017).
[33] Park, T., Liu, M., Wang, T., and Zhu, J., "Semantic image synthesis with spatially-adaptive normalization," *CoRR* **abs/1903.07291** (2019).
[34] Kolkin, N. I., Salavon, J., and Shakhnarovich, G., "Style transfer by relaxed optimal transport and self-similarity," *CoRR* **abs/1904.12785** (2019).
[35] Gatys, L. A., Ecker, A. S., and Bethge, M., "A neural algorithm of artistic style," *CoRR* **abs/1508.06576** (2015).

7. APPENDIX

Table 2. Desired Qualities for Synthetic And Real Image Editing

Network	Example Method	Rand. Gen. Image	Transfer In-Domain Global Style	Interpolate between Images	Amplify Edits	Targeted Semantic Styling	Edit Real Image
GAN: (noise vector input; image output)	StyleGAN2[12]	Yes	Yes	Yes	Not easily	Not easily	Not easily
Image Translation: (image input; image output)	Cycle-GAN[9]	No	Yes, but once per model	No	No	No	Yes
	Image to Image Trans.[31]	No	Yes, but needs image pairs	No	No	No	Yes
Style Transfer: (image and style input; image output)	Deep Photo Trans-fer,[32] SPADE,[33] STROTSS,[34] Gatys NST[35]	No	Yes	No	No	Yes	Yes
	Swapping Auto-Encoder[16]	No	Yes	Yes	Yes	Yes	Yes
GAN Inversion: (image input; image output)	In-Domain GAN Inversion[20]	Yes	Yes	Yes	Yes	Yes	Yes

Figure 15. Examples of real embedded images with semantic corrosion forecasting

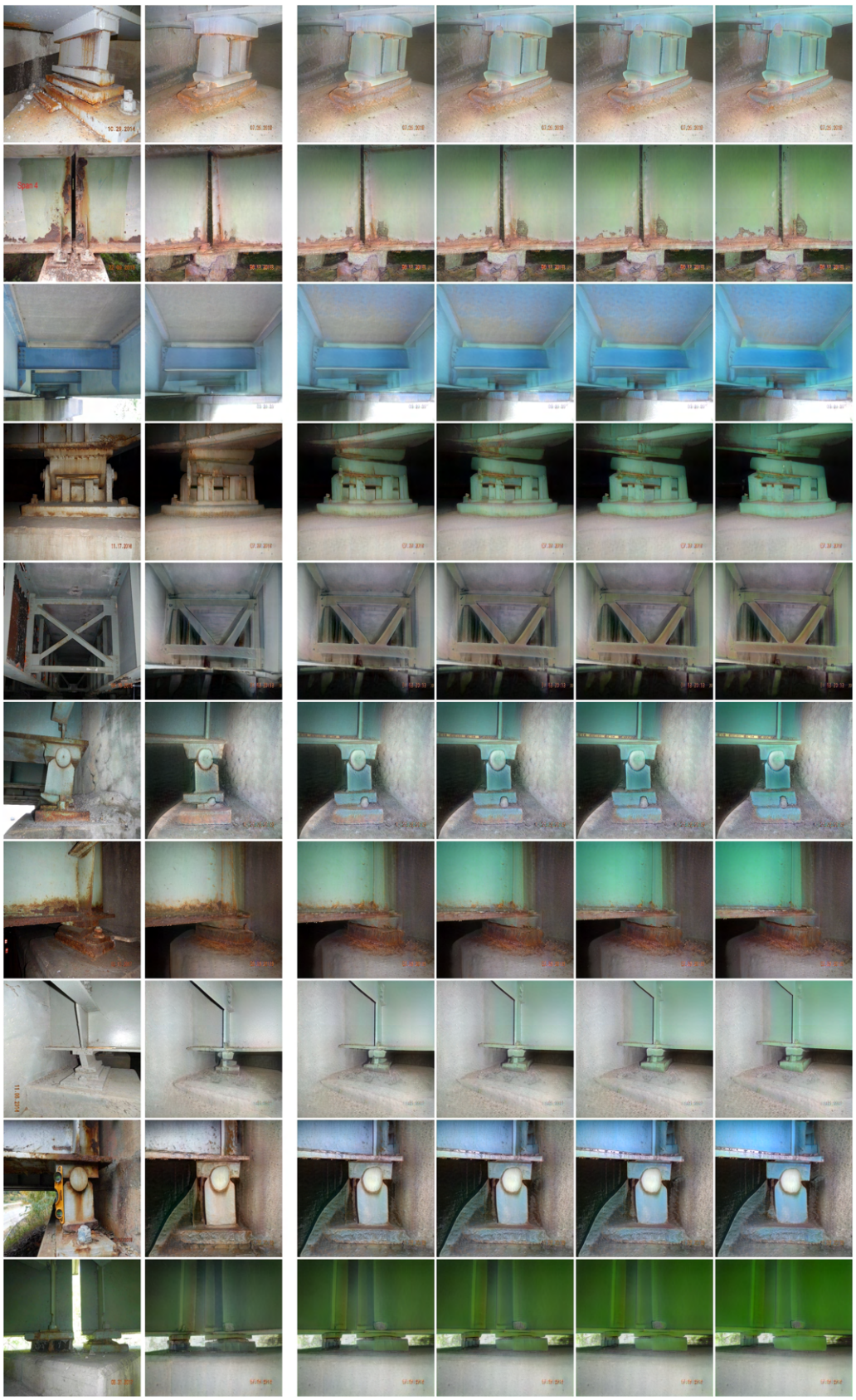


Figure 16. Examples of real embedded images with semantic paint color changing to green/blue

Figure 17. Examples of synthetically generated images with semantic corrosion forecasting

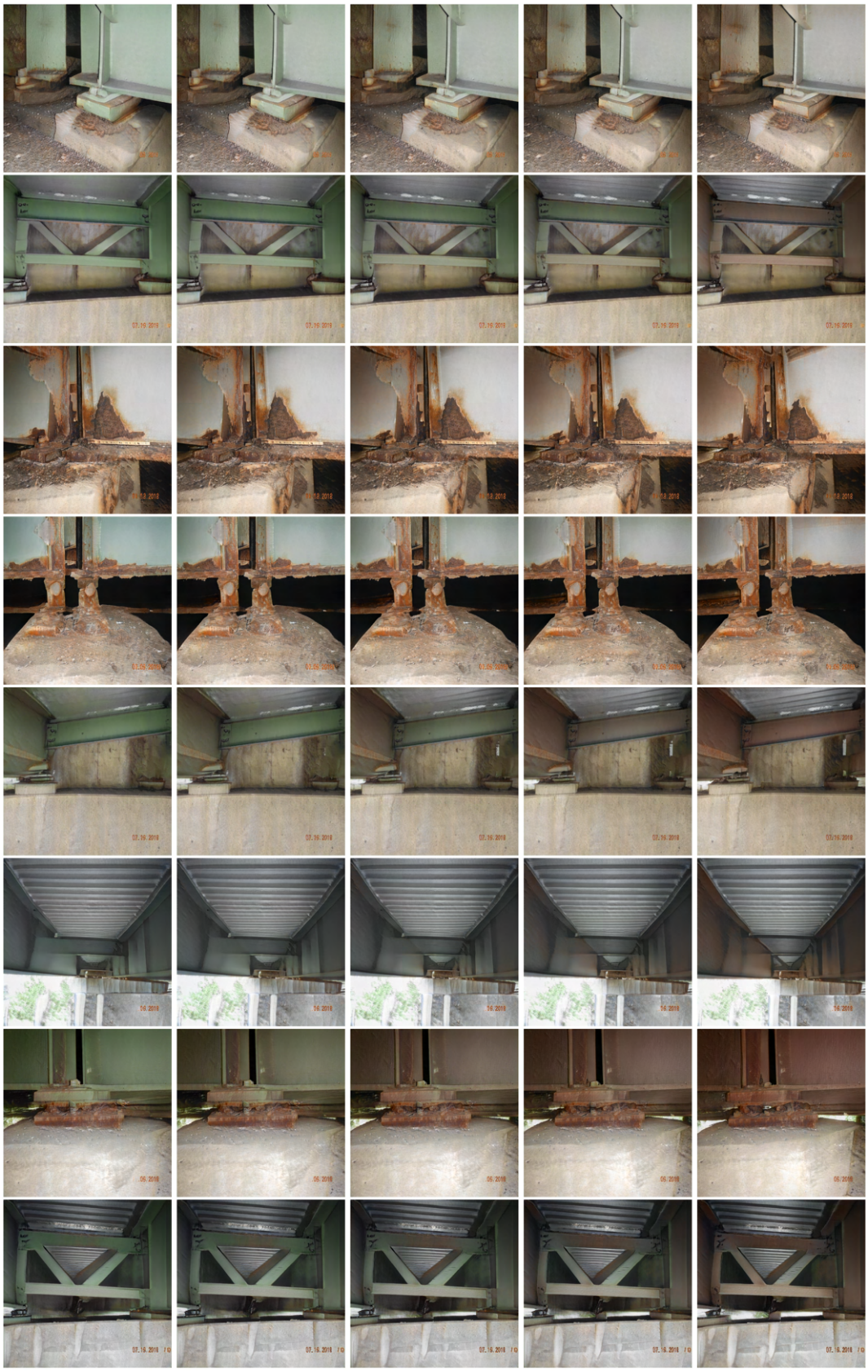


Figure 18. Examples of synthetically generated images with semantic paint color changes to brown