Sixteen Pixels is (Almost) All You Need: Crafting Parameterized Image Uncrumpling Models CS 231N Project Report

Maximilian Du*, Niveditha Iyer*, Tejas Narayanan*

Abstract

As smartphone technology continues to evolve, handheld document scanning is becoming more pervasive. These smartphone scanners must digitally remove creases and crinkles in a document to ensure the best scan possible. In a parameterized setting, this uncrumpling task can be considered as a form of image denoising, or as style transfer. In our work, we experiment with these approaches. First, we create a procedurally-generated dataset of pairwise crumpled and uncrumpled images. Then, we implement and compare denoising and style transfer architectures for our image uncrumpling task. We find that a U-Net architecture combined with a small (4x4) window PatchGAN performs better on standard metrics and also preserves more fine-grained details compared to non-adversarial paradigms and PatchGANs of larger sizes.

1. Introduction

In recent years, the use of smartphone document scanning has increased greatly. Unlike a flatbed scanner, which pushes a document against a pane of glass, smartphone scanning relies on a standard picture taken from a phone camera. Without being flattened against a pane of glass or the use of special illumination, any crinkles on the original document will show up in the picture. Traditional algorithms for text and line drawing enhancement [7] use strong transforms that may not generalize to removing crinkles from a full-color, high-detail image. In our work, we use parametric, adversarial approaches to develop a general uncrumpling algorithm.

The input to our models is an RGB image. We then implement a U-Net autoencoder, which maps from this input image into an output image of the same dimensions ($128 \times 128 \times 3$). We will add an adversarial component to the U-Net for later comparisons, which uses a discriminator network that maps from images to binary predictions.

In our experiments, we will see that the simple image regression U-Net will reduce the lighting artifacts of the crumpling but struggles to produce a sharp image. For adversarial approaches, we will see that reducing the receptive field of the discriminator to a drastically smaller (4×4) window will encourage the generator to output sharper images (Figure 1) and outperform the image regression U-Net on most standard metrics.

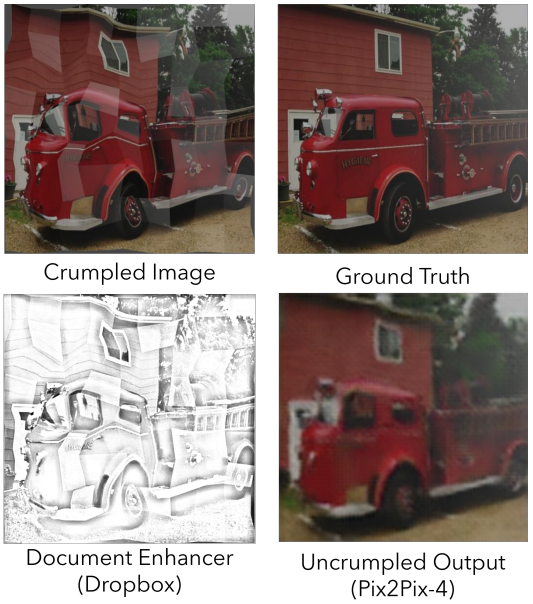

Figure 1. **Uncrumpling Images using Adversarial Learning.** Unlike existing document enhancement methods, our model is able to remove the lighting disruptions of crumpling while retaining color. It is also able to infer the image distortion and reverse the majority of its effects while keeping as many image details as possible.

2. Related work

To our knowledge, there is not explicit prior work that has done a similar uncrumpling task. However, as humans we can perform this uncrumpling in our minds as we look at a crumpled piece of paper, which means that such inferences is possible. In the following sections, we will look at

how existing methods might handle our uncrumpling task.

2.1. Document Enhancement

Document enhancement software is already implemented in mainstream scanning services, like Apple Notes [2] and the Dropbox smartphone application [1]. While they are highly accurate at isolating handwritten or typed text on a document, they often do not go beyond a linear transformation to reduce the effects of a tilted phone. They also apply aggressive contrast-enhancing algorithms which outputs a greyscale image. Taken together, it can be seen in the lower left corner of Figure 1 that these off-the-shelf algorithms are not optimized for full-color enhancements of crumpled images. Even full-color scans, like those created by Apple Notes, commonly apply simple lighting adjustments and do not account for the artifacts introduced by crumpling.

There exists more sophisicated classical algorithms that improve performance. For example, CleanPage [7] uses the natural motion of the operator's smartphone to take a sequence of images under different lighting conditions. Then, they use standard alignment algorithms to stack the pictures and take the median of each pixel. Different orientations create different lighting on the crumples, so a pixel-wise median can mitigate some of the crumpled image's lighting artifacts.

However, CleanPage is targeted for whiteboard writing, which is still a handwriting enhancement task. With images like those of Figure 1, the crumpling introduces visual distortions in addition to lighting artifacts, and visual distortions may not be easily removed through pixel-wise medians. With such a complicated image space, we might consider using a parametric model.

2.2. Uncrumpling as Denoising

We can consider the crumpling operation on the image as a form of special noise. Indeed, in procedural generation, the ground truth image is crumpled by applying a sequence of semi-reversible linear transformations. There are many works on denoising images, and many recent approaches rely on autoencoders [16, 4]. Autoencoders compress a noisy image to a lower-dimensional latent space and then decompress them into a denoised image [16]. Ideally, this bottleneck will force a compact, noiseless representation in the latent space. The autoencoder is trained through noisy-clean paired data, which can be generated through self-supervision. There are many variants to the autoencoder. For example, [11] proposes using dropout layers to increase robustness, and [17] proposes learning the pure noise from an image and then performing a pixel-wise difference operation. In total, the convolutional autoencoder is a simple yet strong baseline.

One major modification to the autoencoder is the *U-Network*. It was originally proposed as a model that outputs segmentations of cells [13], but it has been widely adopted. U-Net applies a bottleneck on the image features, but it also adds connections between the encoder and decoder layers. In this way, the latent space no longer needs to contain all the information necessary for image reconstruction. In our work, we try both simple convolutional autoencoders and U-Networks.

2.3. Uncrumpling as Style Transfer

However, unlike random noise, the crumpling is a special operation that has a distinctive visual properties. Therefore, we can also look at the crumpled and uncrumpled images as different *styles*. There has been significant work in the area of style transfer, starting from gram matrix methods [8] and moving more recently to adversarial approaches [9]. As a key example, Pix2Pix [10] was successful in mapping from images of two different domains, given the existence of style-style image pairs. Pix2Pix uses a U-Net as the generator and a lightweight discriminator that is applied patch-wise across the generated image. The authors experiment with relatively large patches, ranging from 16×16 to 286×286. In our work, we will explore using Pix2Pix on a smaller image and reducing the patch size to emphasize high-frequency details.

Later, CycleGAN [18] was also successful in mapping between image domains without the need of image pairs. Currently, CycleGAN is among the state of the art for style transfer between domains. In our work, we implement modified versions of Pix2Pix and CycleGAN architectures, then we compare them to U-Net image regression.

3. Methods

Mathematically, we can formulate the task as follows. Given `crumpled` images x, `smooth` images y, and loss function $\mathcal{L}$, we want to find a parameterized mapping f_θ such that

$$\theta := \arg\min_\theta \mathcal{L}(f_\theta(x), y)$$

We call $\hat{x} = f_\theta(x)$ as the `uncrumpled` image. In the following sections, we will discuss the various structures for f and loss functions $\mathcal{L}$.

3.1. U-Net Image Regression

The U-Network is a direct convolutional autoencoder that reduces the image down to a single feature vector through repeated convolutions, and then uses transposed convolutions to recreate the image from the feature vector. The encoding and decoding stacks yield symmetric intermediate features, which allows the encoding features to be stacked channel-wise with their corresponding decoding features before being run through a transposed convolution

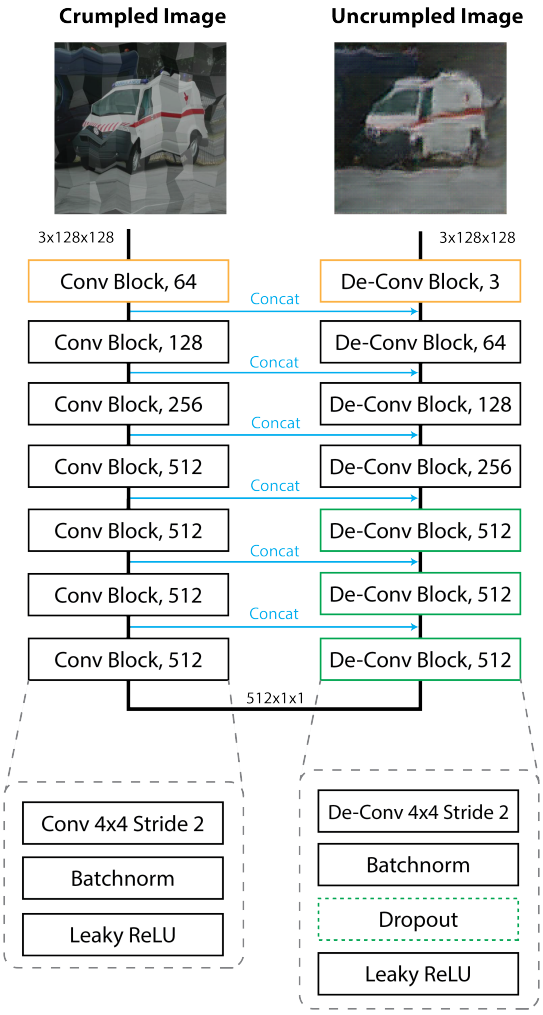

Figure 2. **U-Network Structure.** This is the basic convolutional autoencoder structure we used throughout this work. Each convolutional block reduces the width and height of the features by 2.

block. Channel-wise stacking allows for more information to pass between the encoder and decoder heads. The transposed convolution filters can learn to select the relevant information from the encoder features while ignoring the artifacts introduced by crumpling.

The architecture (Figure 2) is inspired by the generator found in Pix2Pix [10]. All layers use batchnorm except for the first, and the first three decoder layers use dropout for greater robustness. We removed one convolutional block to accommodate for the smaller image size.

For the simple U-Net, we formulate the regression loss function as follows:

$$\mathcal{L}(\hat{x}, y) = \frac{1}{K} \sum_i \sum_j \sum_k |\hat{x}_{i,j,k} - y_{i,j,k}|$$

where K is the size of the images. This pixelwise L1 loss encourages each pixel of the output to be as close to the ground truth as possible. We chose L1 loss because it reduces the blurriness of the final image [10]. In this form, the U-Net acts under the denoising paradigm. In the next section, we will look at how we can augment this structure to become under the style transfer paradigm.

3.2. Modified Pix2Pix

While the L1 regression loss is lightweight and intuitive, such pixel distance metrics are not very expressive. Instead, we can use the same U-Network architecture in an adversarial setup. We define a discriminator $g_\phi(x, \hat{x})$ that takes in a crumpled image x and a uncrumpled image $\hat{x}$. The images are concatenated channel-wise (Figure 3) and then run through a sequence of the same convolutional blocks used in the encoder.

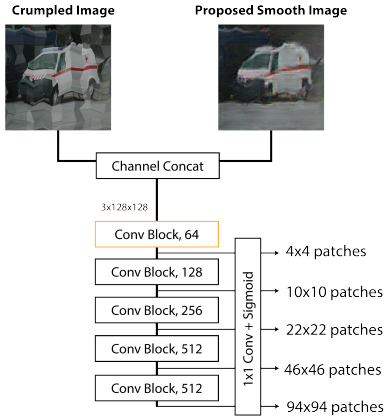

Figure 3. **Discriminator Structure.** This structure, known as a *PatchGAN*, is used in our Pix2Pix implementation

Here, unlike U-Net, we do not convolve into a single feature vector. Rather, we choose to terminate the convolutions early and get a set of predictions. For example, as can be seen in Figure 3, if the model is terminated after four convolutions, it will return an 8×8 matrix of discriminator scores. Each element in this matrix corresponds to a prediction made from a receptive field of 46×46. As such, this output is akin to taking a 46×46 convolutional classifier and convolving it with stride 16 over the original 128×128 image, with sufficient padding. This smaller effective viewing area is akin to looking at the "style" of the image [10] and may encourage more fine-grained details. The Pix2Pix paper calls it a *PatchGAN*.

The discriminator is trained to output high probabilities if the two images are from a crumpled–smooth ground truth pair, and low probabilities if the two images are from a crumpled–uncrumpled generated pair. To do this, we use binary cross entropy (BCE) through all of the patches (Figure 4). For crumpled–smooth pairs, we push the discriminator

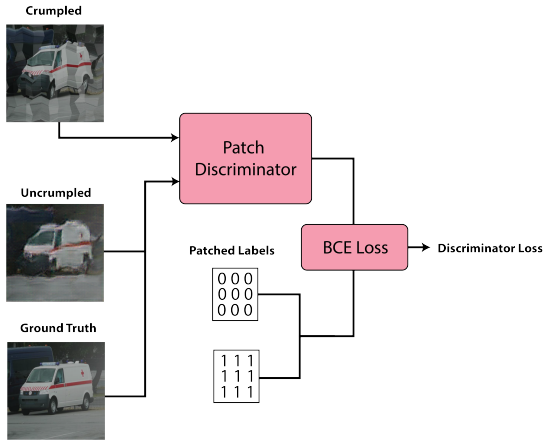

Figure 4. **Discriminator Training in Pix2Pix.** We use Binary Cross Entropy (BCE) to fit the discriminator

close to 1, and for crumpled–uncrumpled pairs, we push the discriminator close to 0.

The generator is trained with a hybrid loss function, as seen below:

$$\mathcal{L} = \lambda \mathcal{L}_{img} + \mathcal{L}_d$$

Here, $\mathcal{L}_{img}$ is the same L1 loss as described in the U-Network. The $\mathcal{L}_d$ is defined as follows:

$$\mathcal{L}_d = \text{BCE}(g_\phi(x, f_\theta(x)), \{1\})$$

where $\{1\}$ is a matrix of ones whose shape matches the PatchGAN output. Intuitively, we are trying to make the discriminator g_ϕ think that $\hat{x}$ is as real as possible. Between the two losses, we weigh them with λ, which is a hyperparameter.

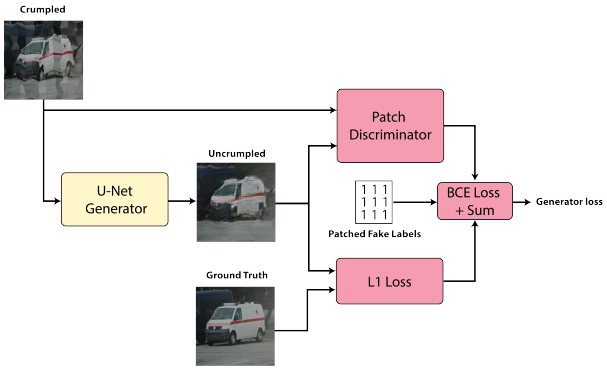

Figure 5. **Generator Training in Pix2Pix.** We use a combined loss function to encourage the generator to produce realistic images.

As [10] mentions, different patch sizes can yield different properties in the images. In our work, we try receptive field sizes of $1, 4, 10, 22, 46$, and 94 and we compare them with various metrics in our results.

3.3. CycleGAN

Pix2Pix is a much more expressive method than U-Net, but it still requires that the images be in crumpled-uncrumpled pairs. Instead, we can use a newer method, CycleGAN, that eases the restriction of paired data and instead utilizes cycle-consistency in its loss [18].

In addition to a forward generator f_{θ_f} that turns crumpled images into their uncrumpled form, we define a backward generator $f_{\theta_b}^{-1}$ that turns uncrumpled images back into their crumpled form. Then, we can incorporate cycle consistency into our loss by pushing $f_{\theta_b}^{-1}(f_{\theta_f}(x))$ close to x.

There are also two discriminators. We push g_{ϕ_f} to output 1 if an image is a real uncrumpled image, and 0 if an image is an uncrumpled image generated from f_{θ_f}. Similarly, we push a second discriminator $g_{\phi_b}^{-1}$ to output 1 if an image is a real crumpled image, and 0 if an image is a crumpled image generated from $f_{\theta_b}^{-1}$.

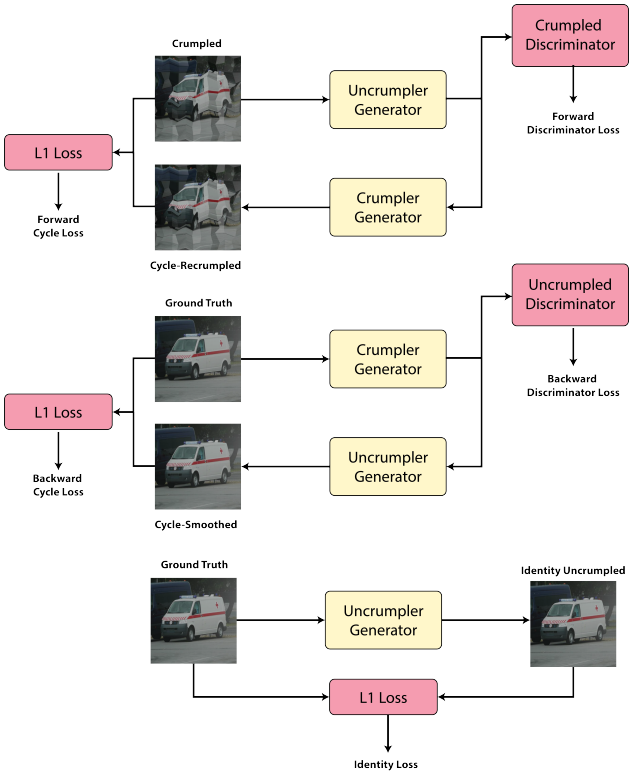

Figure 6. **CycleGAN Training Objectives**. There are five losses that all contribute to the CycleGAN objective.

The CycleGAN generator objective is as follows:

$$\begin{aligned}\mathcal{L}(f_{\theta_f}, f_{\theta_b}^{-1}, g_{\phi_f}, g_{\phi_b}^{-1}) &= \mathcal{L}_{\text{GAN}}(f_{\theta_f}, g_{\phi_f}, x, y) \\ &+ \mathcal{L}_{\text{GAN}}(f_{\theta_b}^{-1}, g_{\phi_b}^{-1}, y, x) \\ &+ \lambda_1 \mathcal{L}_{\text{cyc}}(f_{\theta_f}, f_{\theta_b}^{-1}, x, y) \\ &+ \lambda_2 \mathcal{L}_{\text{identity}}(f_{\theta_f}, f_{\theta_b}^{-1}, x, y)\end{aligned}$$

These loss components are defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(f, g, x, y) &= \text{MSE}(g(f(x)), 1) \\ \mathcal{L}_{\text{cyc}}(f, f^{-1}, x, y) &= \text{L1}(f^{-1}(f(x)), x) + \text{L1}(f(f^{-1}(y)), y) \\ \mathcal{L}_{\text{identity}}(f, f^{-1}, x, y) &= \text{L1}(f(y), y) + \text{L1}(f^{-1}(x), x)\end{aligned}$$

where λ_1 and λ_2 are hyperparameters to weight the cycle and identity losses.

3.4. Ablation: Baseline Autoencoder

For the baseline model, we used a simple convolutional autoencoder. The structure is the same as seen in Figure 2, but all the U-connections have been removed. By running this baseline, we want to see the impact of the U-connections on performance.

3.5. Code Adaptation

Nearly all the code was written by us. As cited in our repository, the CycleGAN implementation was taken from a blog post [5] and adapted into PyTorch [12]. The Pix2Pix was implemented from scratch, using some structure details as specified in the appendix of [10] and adjusting others based on our specific task.

4. Dataset and Features

We chose the face-blurred validation set of the ImageNet Large-scale Visual Recognition Challenge (ILSVRC) as our dataset [14], which contained 50,000 images.

Next, we used Blender [6], a free and open-source 3D modeling and rendering software, to generate the crumpled-uncrumpled pairs of images. We wrote a script that performed the following steps for each image in the dataset:

1. Generate a plane and apply the image as a texture
2. Render the output as the 512×512 uncrumpled image
3. Perform a random crumpling with the `subdivide` modifier
4. Render the output as the 512×512 crumpled image

The image generation took around 4 hours on an NVIDIA RTX 2080 GPU to generate all 50,000 crumpled-uncrumpled pairs. They are downsized to 128×128, normalized between 0 and 1, and stored in a sampling data structure for use in training, validation, and testing. Examples of the crumpled image can be seen in Figure 7.

We loaded 49k crumpled-uncrumpled image pairs to use as the training data. 128 image pairs from the training set were used as validation to monitor training progress, and 1k image pairs were used as a held-out test set across all models.

This method of data generation is highly parameterized. For future data generation, we are able to modify many parameters of the environment, including the lighting conditions, amount of crumple, and reflectivity of the surface. We can also use a larger subset of the ImageNet dataset with no modification to our data generation pipeline.

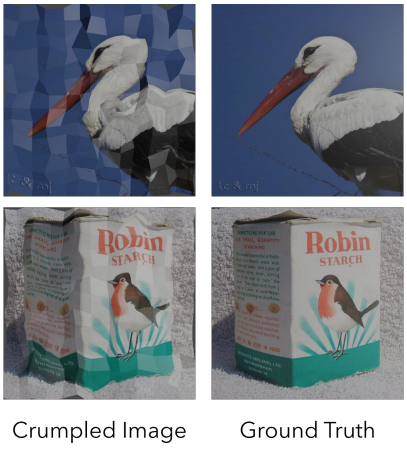

Figure 7. **Examples of Crumpled-Smooth Pairs in our Dataset.** Note how the crumpling imparts severe lighting artifacts and in some cases, irreversible distortion. The effect is most pronounced in images with sharp edges, like the second row.

5. Experiments

5.1. General Training

The models had varying levels of computational complexity and therefore required varying levels of hyperparameter turning. The non-adversarial models were more robust to hyperparameters and more lightweight, requiring around 4 hours to train on an NVIDIA RTX 2080 graphics card. For the adversarial paradigms, Pix2Pix took around 6 hours, and CycleGAN took around 10 hours. Batch sizes were adjusted to fit on 8GB of VRAM. For U-Net and Pix2Pix, the batch size was 32 images. For CycleGAN, the batch size was 8 images. All models were trained for 50k steps.

5.2. Metrics

As a simple metric, we used the pixel-wise mean-squared error (MSE). While this is a standard metric and

related to the L1 loss we train on, Figure 9 shows that it is not very expressive. During training, the MSE decreases sharply at the beginning and stays at a similar level throughout training, while the qualitative properties of the output are changing.

To resolve this, we chose to calculate the `Mutual Information` (MI) of the images x_1, x_2 as defined as follows:

$$\text{MI}(x_1, x_2) = D_{KL}(P(x_1, x_2)||P(x_1)P(x_2))$$

To compute $P(x_1, x_2)$, we flatten out both images and create a joint distribution intensities using k bins in the histogram. To compute $P(x_1)$ and $P(x_2)$, we marginalize across one image. A higher mutual information is better, as it indicates that the two images distributions are less independent from each other. In addition to the mutual information score, we can also qualitatively observe the joint distribution of the images, as seen in Figure 8

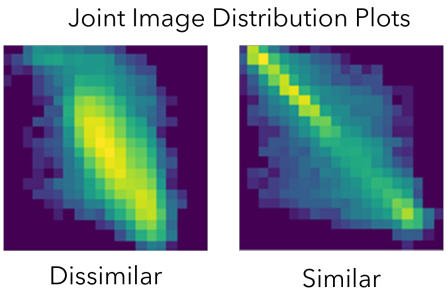

Figure 8. **Expressing Two Images as a Joint Distribution**. More similar images will have a heavier density on the diagonals. Two identical images will create a diagonal matrix.

As another metric, we used the Inception Score [3]. This score is defined between two images x_1, x_2 as the follows:

$$\text{IN}(x_1, x_2) = D_{KL}(I_\theta(x_1)||I_\theta(x_2))$$

where I_θ is an InceptionV3 network pretrained on ImageNet [15]. The Inception score is a more expressive form of the Mutual Information score because the features are extracted from a learned parameterization. However, even though inception score can be more correlated with qualitative image similarity, the number is less interpretable. As can be seen in Table 1, we will use all three metrics to compare our models.

5.3. Baseline Autoencoder

As mentioned in the methods, the baseline model is the U-Net structure inspired from Pix2Pix [10], with the pass-through connections removed. We train it using an L1 loss

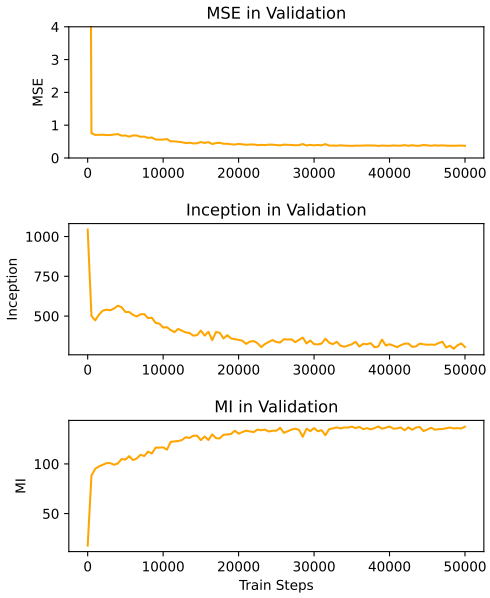

Figure 9. **Comparison of Loss Function Expressivity (Pix2Pix-4)**. Inception and Mutual Information metrics tell us more about model performance.

using standard parameters on an Adam Optimizer, as recommended by denoising literature [16]. As can be seen in Figure 13, the model learns to remove the lighting artifacts and some of the warping caused by the crumpling process. However, the output image is highly blurred. This reduced fidelity is expected, as the bottleneck layer imposes a heavy restriction on the data that flows through the network.

5.4. U-Net

Next, we added the pass-through connections and trained using the same loss objective and optimizer. As can be seen in Table 1, this model performs significantly better than the baseline across all metrics. This improvement is mostly attributed to the greater flow of information in the network. Intuitively, the decoder filters now can *choose* information, instead of *interpolating* information. Qualitatively, Figure 13 also shows that the output is less blurry.

Furthermore, the model attempts a stronger compensation against the warping effects. In Figure 10, we see that that the original crumple imparted a severe warp in the upper lid of a display case. The U-Net learns to fix this. The same can be said for the firetruck (right column of Figure 13). The house window was significantly damaged by the crumpling, but the U-Net repaired it. This perhaps even shows an emergent understanding of perspective.

While the U-Net shows promising behavior, it still yields

Figure 10. **The Surprising Efficacy of U-Net Edge Recovery**. Without any adversarial losses, the U-Net is able to infer straight edges in the crumpled image.

an image that is qualitatively blurry. Because the U-Net is getting enough encoder-decoder information transfer, the most likely explanation lies in the natural uncrumpling effect of a Gaussian blur. In this way, the U-Net has learned a simple shortcut to reducing the loss.

5.5. Pix2Pix

In an attempt to reduce the blurring in the final image, we experiment with our modified version of Pix2Pix. We trained with $\lambda = 100$ and all the default hyperparameters recommended by [10].

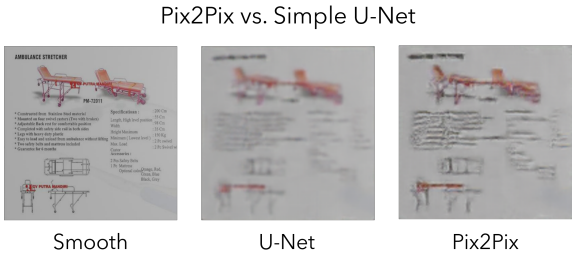

Figure 11. **Pix2Pix-2 Compared to Image Regression U-Net**. An adversarial loss encourages the reconstruction of high-frequency details.

In our experiments, we looked at various sizes of patches for the PatchGAN, with a focus on smaller patches ($4 \times 4, 10 \times 10$) than what was explored in the original paper.

Indeed, we found that smaller patch sizes yield better outputs. Qualitatively, in the abacus example (Figure 12) the 4×4 patch is able to reconstruct each bead without much distortion. In contrast, the $94, 46$, and 22 patches yield heavily distorted details. The 1×1 patch does not have distortion, but it fails to reconstruct the details, due to lack of spatial context in the discriminator. As another example, Figure 11 shows that the 4×4 PatchGAN is able to reconstruct high-frequency text-like details with better fidelity than the U-net.

The quantitative results agree with this qualitative assessment. Table 1 shows that the 4×4 Pix2Pix version was able to outperform or tie the U-Network across all metrics. Intuitively, the small window size forces the GAN to focus on the higher-frequency details. Performance drops with a 1×1 patch, as we remove all spatial contexts.

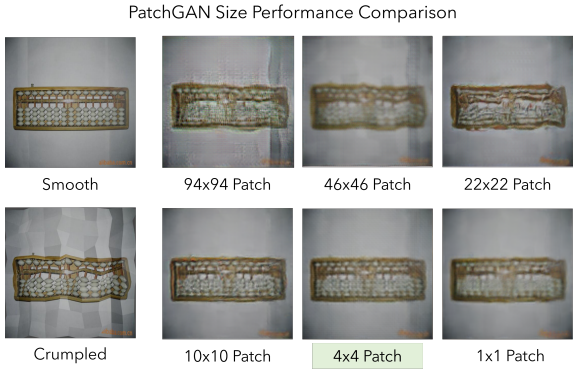

Figure 12. **Pix2Pix with Different Patch Sizes**. Smaller patch sizes generally yield higher quality images

5.6. CycleGAN

We trained the CycleGAN model on an Adam optimizer using hyperparameter values $\lambda_1 = 10$ and $\lambda_2 = 5$. As shown in Figure 13, the model fails to uncrumple the images; many of the original crumples are readily visible in the CycleGAN output. Moreover, it adds patches of black artifacts to the images and darkens the output image in its transformation. Both qualitatively and quantitatively (Table 1), CycleGAN performs the worst out of all the model structures tested.

We believe that CycleGAN was unable to converge to an optimal model due to its complex loss function and its departure from an image regression component. With Pix2Pix, the L1 image regression loss contributed 100 times more than the discriminator loss, meaning that the adversarial contribution only made minor corrections to the model. In the case of CycleGAN, there is no L1 image loss component due to the lack of image pairs in its formulation. This creates instability in the training of the model, even with the recommended hyperparameters. Additional tuning and structural modifications may be required for CycleGAN to perform well on this task.

5.7. Total Comparisons

Figure 13 shows our models run on five notable test images. The first two contain curves and straight lines that are severely interrupted by the crumpling operation. U-Net and Pix2Pix both reverse this transformation, while CycleGAN and our baseline autoencoder struggle to do so. The third and fourth image contains very fine-grained details, like text on a paper and the individual beads of an abacus. These

demonstrate high-fidelity reconstruction in the Pix2Pix-4. Finally, the last image was chosen as a combination of distorted curves and fine detail. Pix2Pix-4 and U-Net reconstructed nearly all the image features in a sensible manner.

Table 1 shows our model performance metrics as evaluated on 1k held out test images. The best scores are bolded. As predicted, the inception score reflects the greatest difference between Pix2Pix-4 and the U-Net, as the pretrained Inceptionv3 is able to extract similar features that contribute to our qualitative observations.

The training losses were slightly lower than validation and test losses, but even when evaluated on the held-out images, the models performed very similarly. Furthermore, we had a large dataset of generated images, and in our 50k steps of training, each training image would have only been seen around 33 times. Therefore, it is likely that our model did not significantly overfit on the training data.

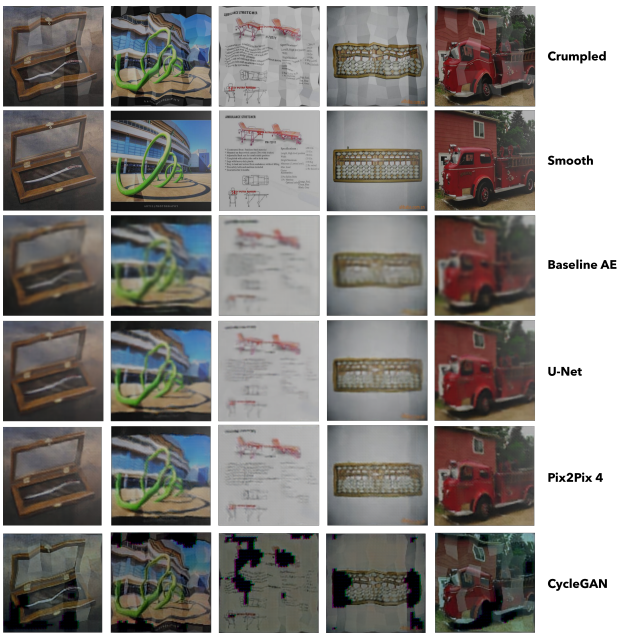

Figure 13. **Comparing All Models on the Same Data**. Pix2Pix with 2x2 PatchGAN yields the highest quality outputs.

6. Conclusion and Future Work

In our project, we wanted to find a parametric model that can remove the lighting artifacts and spatial warps imparted by crumpling the image. By using a convolutional U-Network structure combined with an adversarial small-window PatchGAN loss, we were able to remove these artifacts and warps while avoiding image blurring. This performance enhancement can be attributed to the high-frequency details that the discriminator is forced to pay attention to with such small of a window. The high-frequency-attentive discriminator loss augments the low-frequency-attentive L1 image regression loss, leading to a good quality reconstructed image.

While the CycleGAN did not show much promise on this domain, we suspect that with additional tuning and maybe more data, it may also be able to perform well. CycleGAN does not require crumpled-smooth image pairs in training, which will allow it to train on a larger dataset, potentially even with real-world examples of crumpled and smooth images. With more compute and memory, we can try increasing the randomization of our procedurally-generated crumpled images. Potentially we might randomize the lighting position, light type, and even the material of the paper. With these modifications, the model might be able to generalize zero-shot to real-world examples of crumpled images. This would provide a tool that may improve image scanning experiences in general.

	Metrics		
Models	100× MSE	Mutual Info	Inception
Baseline AE	0.42 ± .01	0.89 ± .01	3.99 ± .07
U-Net	0.26 ± .005	**1.12 ± .01**	2.56 ± .07
Pix2Pix 94	0.78 ± .01	0.64 ± .01	3.92 ± .07
Pix2Pix 46	0.91 ± .04	0.76 ± .01	3.93 ± .08
Pix2Pix 22	0.62 ± .01	0.8 ± .01	3.79 ± .07
Pix2Pix 10	0.29 ± .01	1.07 ± .01	2.4 ± .07
Pix2Pix 4	**0.25 ± .005**	**1.12 ± .01**	**2.27 ± .06**
Pix2Pix 1	0.29 ± .01	1.09 ± .01	2.73 ± .07
CycleGAN	14.91 ± .17	0.41 ± .005	5.87 ± .09

Table 1. **Cross Model Comparisons with Three Metrics**. The Pix2Pix model with a 4x4 PatchGAN ties or outperforms the non-adversarial U-Net model.

7. Contributions and Acknowledgements

All three authors contributed equally to this project. Maximilian Du implemented the Pix2Pix + U-Net structures and the Mutual Information metric, Niveditha Iyer and Tejas Narayanan implemented the CycleGAN, and Tejas Narayanan created the procedurally-generated dataset and the Inception metric. All members contributed to the writing of this paper. Briefly, to help parallelize final experiments, an A5000 GPU from the IRIS lab cluster was used.

References

[1] Dropbox: Document and PDF Scanner.
[2] How to scan documents on your iPhone, iPad, or iPod touch.
[3] S. Barratt and R. Sharma. A Note on the Inception Score.
[4] Y. Bengio. Learning Deep Architectures for AI. page 56.
[5] J. Brownlee. How to Implement CycleGAN Models From Scratch With Keras.

[6] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022.

[7] J. Courtney. CleanPage: Fast and Clean Document and Whiteboard Capture. 6(10):102.

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks.

[11] J. Liang and R. Liu. Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. In *2015 8th International Congress on Image and Signal Processing (CISP)*, pages 697–701.

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[13] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge.

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.

[16] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin. Deep Learning on Image Denoising: An overview.

[17] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. 26(7):3142–3155.

[18] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.