GlareNet: A Deep Learning Approach To Removing Glare From Images Taken in Proximity Operations

Sournav Sekhar Bhattacharya* and Aaron Su†
Texas A&M University, College Station, TX, 77843

Gregory E. Chamitoff‡
Texas A&M University, College Station, TX, 77843

Optical sensors have been used aboard satellites for a variety of purposes. In the case of proximity operations in low Earth orbit, these sensors may be used to capture images of a target satellite from a chaser. From these images state information may be obtained about the target. However, due to adverse lighting conditions caused by glare from the Sun, the images that are input to the algorithms responsible for computing state information may produce erroneous results. We propose GlareNet, a deep learning based solution to this problem. In recent years Generative Adversarial Networks (GAN) have been making strides in harnessing neural networks to generate images, and alter images to remove possible unwanted qualities, such as blur[1, 2] GlareNet utilizes a conditional deep convolutional GAN (DCGAN) in order to receive an image where there is significant glare around a target satellite and subsequently remove the glare while retaining the structure of the satellite even in cases where the glare occludes parts of the target vehicle.

I. Nomenclature

GAN	=	Generative Adversarial Network
$DCGAN$	=	deep convolutional generative adversarial network
X	=	image with glare in training set
Y	=	image without glare in training set
G	=	image generated by generator
L_P	=	perceptual loss
L_W	=	wasserstein loss
Y_{true}	=	ground truth image with no glare in training set
Y_{pred}	=	predicted image from generator
y_{true}, g	=	true discriminator output label
y_{pred}	=	predicted discriminator output
n	=	batch size
$VGG16$	=	Deep convolutional network used for image classification

II. Introduction

Images taken in proximity operations often consist of a target satellite in frame. These images may be used as estimates of a satellites orientation, or may be used for extracting other data concerning the target satellite such as a classification. These algorithms will use the input images in order to extract key features in order to ascertain the required parameters concerning the target. The glare present in the image around the satellite may obfuscate key structural components from the image. These can include edges, corners, and other key features that may be used for tracking. Common solutions to these approaches may range from hardware implements such as filters to software algorithms such as brightness correction algorithms. In the former case not every satellite may be equipped with such hardware, and may become

*Graduate Student Researcher, Computer Science and Engineering, ASTRO Center H.R Bright Building, sournav@tamu.edu
†Undergraduate Student Researcher, Computer Science and Engineering, ASTRO Center H.R Bright Building, aa_ron_su@tamu.edu
‡Professor of Engineering Practice, Department of Aerospace Engineering, H.R. Bright Building, AIAA Associate Fellow, chamitoff@tamu.edu

damaged in flight, or on the ground. The typical software solutions rely solely on a single captured image, with no semantic understanding of the object (in this case the target satellite) present in the image. Thus such algorithms are not able to infer the content occluded by glare, as this information is completely missing from the image.

We propose a deep learning method by which during deployment a convolutional neural network (CNN) can be fed a glared image of a satellite and output an image with no glare while retaining the satellite itself. Note that the semantic information previously missing about a particular satellite is encoded during the training process where glare can be artificially be introduced to images of the satellite with no glare. These pairs of images with and without glare form the desired input and desired output of the supervised learning problem that can be learned via GlareNet. At its core, GlareNet is a GAN, or generative adversarial network. GANs are becoming increasingly more prevalent in image generation and image alteration tasks, and thus are suited to tasks such as removing glare.

Fig. 1 (LEFT) Image in test set with glare. (RIGHT) Test image after it has been passed through GlareNet to remove glare. Note that the glare occludes part of the panel, but GlareNet is able to reconstruct this occluded region.

III. Background On Deep Convolutional Neural Networks

A. Introduction

Neural networks have become pervasive in several applications, such as natural language processing, speech recognition and more. In particular in the field of computer vision and image processing, convolutional neural networks have proven successful in tasks such as object detection, image classification, and in various image generation problems.

Convolutional neural networks consist of layers, where each layer consists of a set of filters. As input each layer takes either an image, or the output of a previous layer. Thus the input to a given layer, or "feature map" can be characterized by its height h, width w, and depth or number of channels d. Each filter is a tensor of dimension $f_w \times f_h \times d$. Note that that f_h and f_w are user defined parameters and correspond to the height and width of the filter respectively. The depth of each filter is fixed as the same as the number of input channels. An element-wise product is taken of the filter and a section of the input feature map of the same dimension. This results in a single output number, that corresponds to an element in a single channel of a feature map. Thus by convolving the filter across the input feature map, a two dimensional output matrix is produced. The dimensions of these output matrices is dependant on the size of the filter itself, as well as the convolution itself. If the filter is convolved with a stride s then the output size is reduced. Furthermore, the input feature map can be expanded by a padding p. The output width and height are thus dependant on that of the input feature map and that of the filter.

$$\text{Width} = \frac{w - f_w + 2p}{s} \tag{1}$$

$$\text{Height} = \frac{h - f_h + 2p}{s} \tag{2}$$

If in a given layer there are n filters, then the n such matrices are output. The depth of the feature map is thus n. With respect to dimensions, there are two forms of padding, namely same padding and valid padding. Same padding is such that the width and height of the feature map output of the layer remains the same as that of the input. Valid padding is a padding of size 0, resulting in a reduced width and height.

B. Pooling Layers

Pooling layers are similar to convolutional layers in that they involve filters that must be convolved along the input feature map. However instead of each element of the feature map being a learnable weight, the average of elements of feature map overlapping the filer is taken, or the maximum. Pooling layers are thus not learnable layers. Their primary function is to reduce the dimensionality of the input layer, and average features present in the feature map. [**?**]

C. Up Sampling

Up sampling layers have the opposite effect of that found in pooling layers. [**?**] There are learned and unweighted variations to the up sampling procedure. Simple bicubic interpolation or other computer vision and image processing scaling primitives may be employed. Alternatively the transposed convolution is a learned alternative where the convolution operation is transposed (where normally a convolution would downscale the image, here the image is upscaled). In general for a pixel at position x, y in the input feature map, the output pixel locations are x', y' for an upsampling layer to be scaled by a factor of n:

$$x' = \left\lfloor \frac{x}{n} \right\rfloor, y' = \left\lfloor \frac{y}{n} \right\rfloor \tag{3}$$

This is also a non-trainable operation, meaning there are no learnable weights involved.[**?**]

D. Dropout

When a dropout layer is added a random number of neurons in the neural network do not fire, and are ignored. This is to reduce the chance of overfitting. Note that as a result of using dropout, a network may take more training cycles to converge. Dropout is associated with a user tune-able parameter that corresponds to the chance of a neuron to be deactivated. [**?**]

E. Batch Normalization

One possible cause of overfitting is having weights of large magnitude. In order to limit this batch normalization is used. A batch normalization layer normalizes the outputs from a given layer of a neural network in a parameterized way, when an optimizer or stochastic gradient descent is used during training. [**?**] Consider a mini training batch with n training data samples, if the output of a given layer with k number of neurons compose the vector V_i with a mean of μ_B and standard deviation of σ_B across all the training inputs of the minibatch where $1 \leq i \leq n$, then the resultant vector V_i' after batch normalization is:

$$\mu_B = \frac{1}{n} \sum_{i=1}^{n} V_i \tag{4}$$

$$\sigma_B = \sqrt{\frac{1}{n} \sum_{i=1}^{n} V_i - \mu_B} \tag{5}$$

$$V_i' = \frac{V_i - \mu_B \cdot \mathbf{1}^k}{\sigma_B} \cdot \gamma + \beta \tag{6}$$

Here B represents the mini-batch. In the above equations, γ and β are learnable parameters that do not need to be set explicitly by the user. The individual mini batch means and standard deviations computed in the procedure above can be used to update a rolling global set of statistics, namely the mean μ_G and standard deviation σ_G:

$$\mu_G = \alpha \cdot \mu_G + (1 - \alpha) \cdot \mu_B \tag{7}$$

$$\sigma_G = \sqrt{\alpha \cdot \sigma_G^2 + (1 - \alpha) \cdot \sigma_B^2} \tag{8}$$

Note that in these equations, α, known as momentum, is a user set hyper-parameter such that $0 \leq \alpha \leq 1$. This is used as a weight over the previous batch's statistics.

IV. Conditional Generative Adversarial Networks

Conditional Deep Convolutional GANs or Conditional DCGANs make use of two convolutional neural networks, namely the generator and the discriminator. The generator and discriminator play an adversarial role against each other. The generator will use its convolutional layers in order to produce an output image corresponding to a given input. The discriminator will attempt to classify whether a given image is generated or in the training set. The purpose of the generator is to produce images that cannot be differentiated from the one corresponding to the given input in the training set. [3]

The generator and discriminator are combined into a single network at training time. During deployment and testing the generator is used alone. The training data consists of X, Y image pairs, where X corresponds to an image that is unaltered (in our case an image with glare), and Y corresponds to the desired output image (an image without glare). During training, the X image is fed into the generator, which generates an image G. G is labeled as a generated image, and fed through the discriminator. Y is labeled as a training set image and then fed through the discriminator. The discriminator back-propagates the resulting loss. The discriminator and generator are then combined into a single network, where the weights of the discriminator are frozen. The input X is fed into this network with the target discriminator output being set to "training set". Thus the generator over time will learn to generate images that the discriminator classifies as no different than Y. [4]

Conditional GANs are different that pure GANs in that, they require a conditional input, whereas pure GANs take in some random noise or a distribution vector. Thus there is often no qualitative correlation with the input of a pure GAN and its output. In the case of this work, the condition is an image itself, rather than a binary choice.

V. Variations and Additions to the Convolutional Neural Network

A. Residual Network

Residual neural networks change the learning target objective. If a neural network is a function mapping a given input x to an output y, then a residual network builds on this model by instead learning the residual between x and y. Thus the residual neural network can be thought of as a function that learns $y - x$. This learned output can then be added back onto the original input. This is ideal for our use case, as the final output desired is a modification (glare removal) of the original image. Residual networks have been used in other works in conditional GANs, such as the task of motion blur removal from images.

B. Reflection Padding

As mentioned in the section explaining convolutional neural networks, there are various forms of padding. The simplest of which expands the input feature map in the width and height axes by appending pixels of value zero. A problem with using this approach is that the corner pixels of the original image are not weighted the same as those in the center of a feature map. This may cause artefacting in the final output of a GAN. Reflection padding works by reflecting a window of the edge of the image across the required padding border. Since the content of the reflected edge is similar to the edge itself, the pixels on the edge of the feature map are not weighted differently than those on the center.

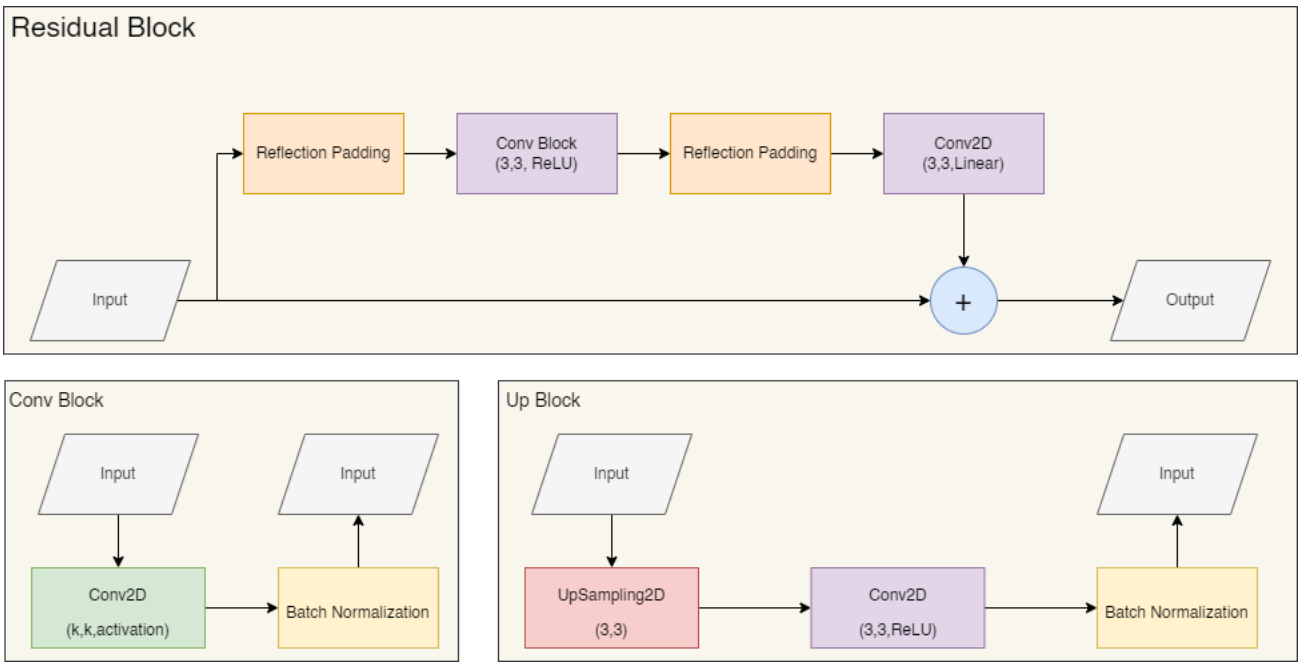

Fig. 2 Various components needed for construction of the Generator and Discriminator

VI. Construction of the GlareNet Conditional Generative Adversarial Network

A. Generator

The generator is composed of various convolutional components. The generator itself follows the residual architecture, in that the input is added to the output of its convolutional layers. Thus only the difference between the input signal (the image with glare), and the output signal (the image with no glare) needs to be learned. The generator's convolutional layers are composed of Residual Blocks, Convolutional Blocks, and Up Sampling Blocks as detailed in the diagram below. Additionally convolutional layers are also used.

The following is the assembled generator layers. The output itself is a $256px \times 256px$ image. All layers not preceded by a reflection padding layer that are convolutional use the "same" padding scheme (the convolutional output size is the same as the input under this padding type).

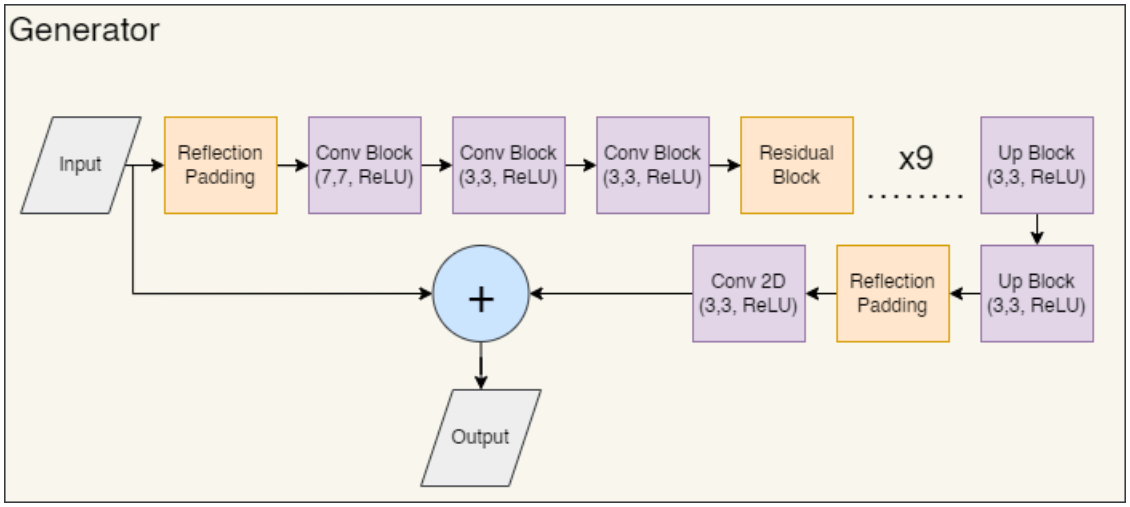

Fig. 3 Generator architecture

B. Discriminator

The discriminator is a much smaller network than the generator, and is composed of convolutional and dense layers. The final output is a single node, corresponding to the discriminator's prediction of whether the input image was part of the training set or generated by the generator.

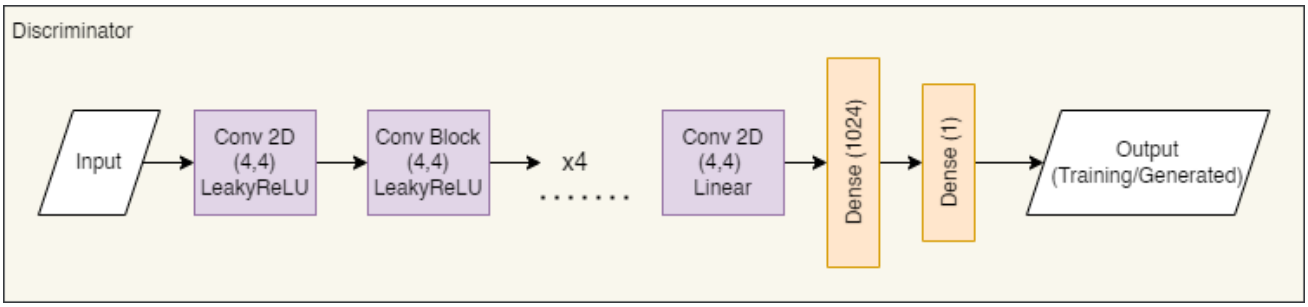

Fig. 4 Discriminator architecture

C. Loss

There are two main loss functions used, namely a perceptual loss, and the wasserstein loss. The perceptual loss requires the use of a VGG16 network without its classification head trained on the "ImageNet" dataset (Note that the head is the last layer of the VGG16 network). Thus the perceptual loss is:

$$P_L = \frac{\sum_{i=1}^{n}(VGG16(Y_{true_i}) - VGG16(Y_{pred_i}))^2}{n} \tag{9}$$

Here Y_{true} is the batch of ground truth images with no glare in the dataset, where as Y_{pred} corresponds to the predicted images. n is the batch size specified during training. The wasserstein loss is used for the discriminator:

$$P_W = \left| \frac{\sum_{i=1}^{n} y_{true_i} \cdot y_{pred_i}}{n} \right| \tag{10}$$

Here y_{pred} corresponds to the predicted discriminator output. Note that $y_{pred}, y_{true} \in \{0, 1\}$. y_{true} corresponds to the ground truth discriminator target.

"ImageNet" is a large dataset composed of images of various objects. It is often used to characterize the performance of machine learning classifiers on image data. By training the network on image net, and removing the head we are able to retain the VGG16's capabilities of identifying features in images without the need to be tied to the actual prediction classes provided by ImageNet. This is a common form of transfer learning, and has been used with success in other tasks such as object detection. [5]

VII. Dataset

A. Introduction

Getting imagery data from proximity operations environments in space is difficult if each glared image needs a corresponding image with no glare. Thus we have opted to use computerized graphics to render a training set, where lighting is adjusted without changing camera parameters, thereby producing pairs of images that have and do not have glare respectively.

In particular we choose to use Blender, the open source modeling and visualization software. We have also made use of NASA's repository of 3D models. In particular we chose the Aura satellite as our learning target. This is because the Aura satellite is has an asymetric axis, providing a challenging learning objective for the machine learning components of GlareNet.

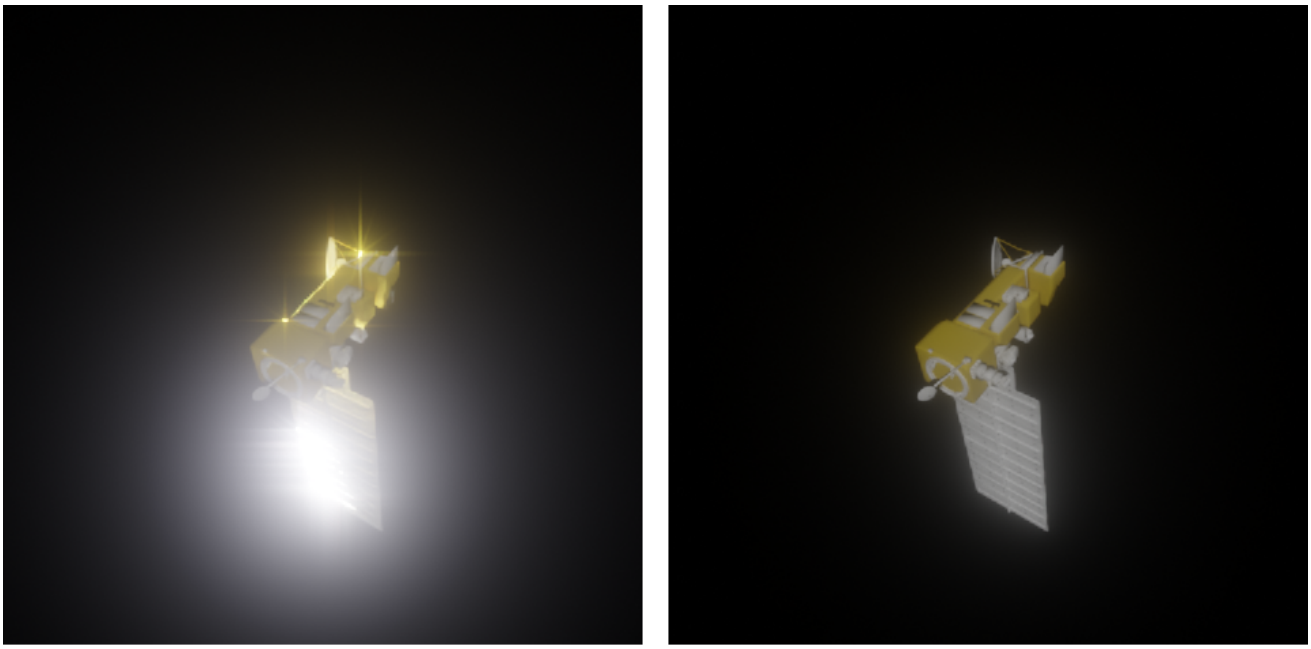

Fig. 5 (LEFT) Image with Glare from dataset, (RIGHT) Same satellite, with same attitude and camera parameters with no glare generated for dataset by Blender

B. Data Generation

The model of Aura is centered along the central axis of the camera frame, and is randomly rotated. This alone produces the render of the image with no glare. In order to add glare we position a light behind the object. The intensity, and location of the light relative to the camera is changed in order to allow for a variety of lighting conditions, and to avoid repetitive scenarios in our dataset. For enhancing the quality of this glare we use the "glare" node found in Blender's compositing tool. There are several possible glare types to choose from. The "ghosts" emulates haze, "streaks" simulates lens flares, "fog glow" is similar to ghosts but muted in effect. We opt to use fog glow, as it achieves results that are qualitatively similar to real world images.

Thus with the use of blender we were able to create a 1000 image pair dataset to train our generative adversarial network on.

VIII. Training and Results

A. Training Specifications

In order to train GlareNet we use a NVidia GeForce 1080ti graphics card. Furthermore we use the Tensorflow 2.0 and Keras libraries to implement the neural network in Python 3. This abstracts away several of the key convolutional neural network components into simple callable functions, thereby decreasing development time. We only consider the last 10% of our dataset as test. The rest is used for training, and validation. The test set images are not seen by the neural network during training.

B. Training

By looking at the internal feature maps of the convolutional neural network it is possible to understand the learning process. We feed in a test image that the neural network has not seen before, and sample the feature maps.

During the training of gans since the qualities desired in the output, in this case the removal of glare from an image, are subjective there is not an accuracy metric that can be assigned during the training process. Thus we focus on understanding the learned representation present in the convolutional neural network. In particular we choose to look at the output of the 9th convolutional layer in the VGG16 net. The satellite is distinguished from the foreground, however

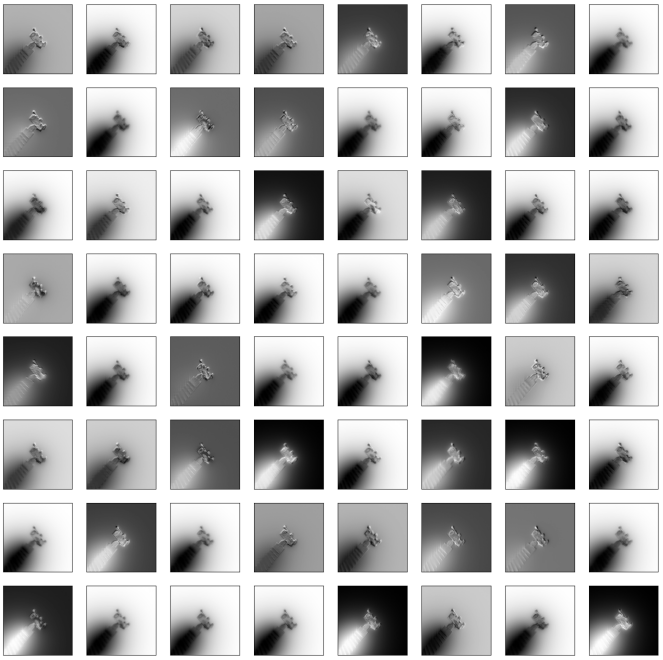

Fig. 6 Collection of feature maps

different feature maps highlight different regions of the satellites.

It is important to note that if instead of 2D upsampling, transposed convolutions are used, then heavy artefacting is present in these internal layers that percolates to the end of the network, and creates a gridlike pattern in the output image.

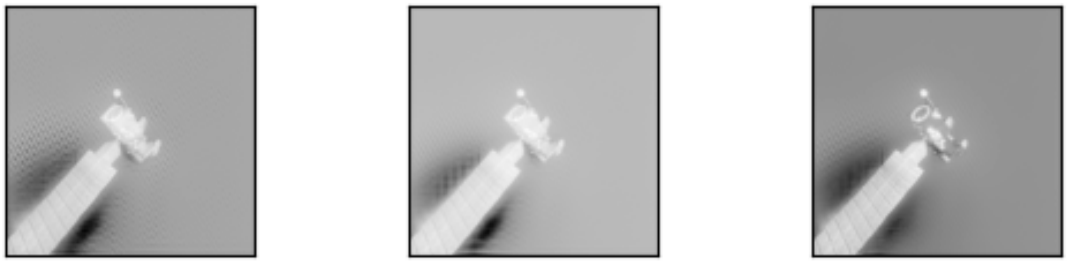

Fig. 7 RGB Output channels with repetetive grid like artefacting present due to the use of transposed convolutions instead of UpSampling layers

IX. Results

A. Analysis Methodology

As stated in the previous section, the GAN itself is producing an image that attempts to remove glare. However, the efficacy of this is difficult to establish numerically as there is no measure for how much glare is present in an image. Rather we opt to use the algorithms that take images as input to perform processing tasks, and then validate that images passed through our proposed GlareNet produce more favorable results. We focus on two main use cases, edge detection and corner point detection.

B. Corner Point Detection

1. Introduction

Corner detection aims to identify corners in images. We choose the Harris Corner Detector written in OpenCV. In this algorithm, edges that meet are considered corners. Corner detection is useful for operations such as motion tracking, and feature identification. In a glared image, the number and placement of corners in the image will be inaccurate.

2. Results

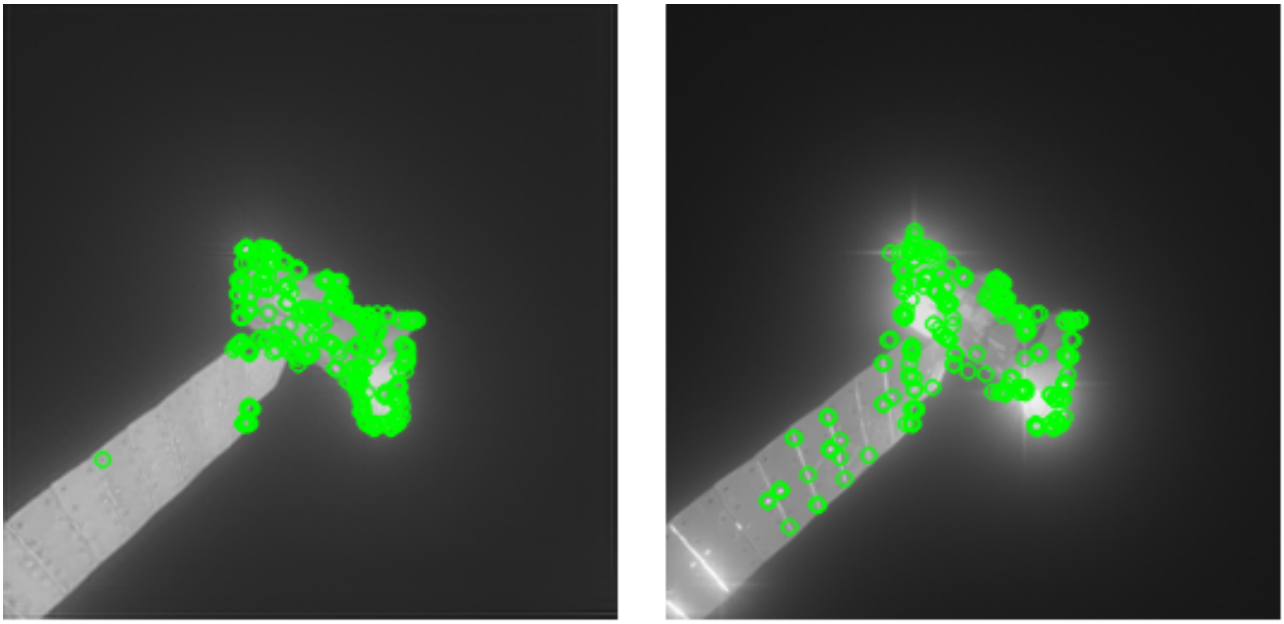

Fig. 8 (LEFT) Corner points on the image without any glare (RIGHT) Corner points on the image with glare

The concentration of corner points is on the satellite body and not the panel in the unglared image. This is due to removal of some edges between panel elements by the GAN, and is understood as an artefact. However the coverage of corner points improves around the body of the satellite, as the glare from these regions were obstructive.

In general the number of keypoints detected is a good indicator of how closely the image aligns with the ground truth image that has no glare. Taking the images in our test set, the average number of keypoints detected for the ground truth was 268.1, and our GlareNet processed images on average detected 254.1 corner points. Images that were glared averaged 310.44 corner points. Our approach thus produces images more closely resembling images that have no glare.

C. Edge Detection

1. Introduction

Edge detection is a convolutional image processing approach that aims to highlight edges in an image. An edge can be defined as a line of pixels such that they interface between two contrasting portions of the image. The convolutional approach consists of sliding a filter along an image, the structure of this filter will determine the type of edges that the image responds to (eg. horizontal or vertical). In particular we choose canny edge detection, this is provided by the computer vision toolkit OpenCV in python 3.

2. Results

It is seen that in some cases there is some distortion caused by artefacting along the edges, if this artefacting is removed by simple cropping then the average sum of absolute differences between the edge detection of the images with glare and the control ground truth is 106335, wheras when the image with glare is passed through glare net the average sum of differences between pixels is 39780. This reduction in the error between the edge detection results shows that GlareNet effectively reduces the impact of glare on a source image. In our results we observed that the edge detection

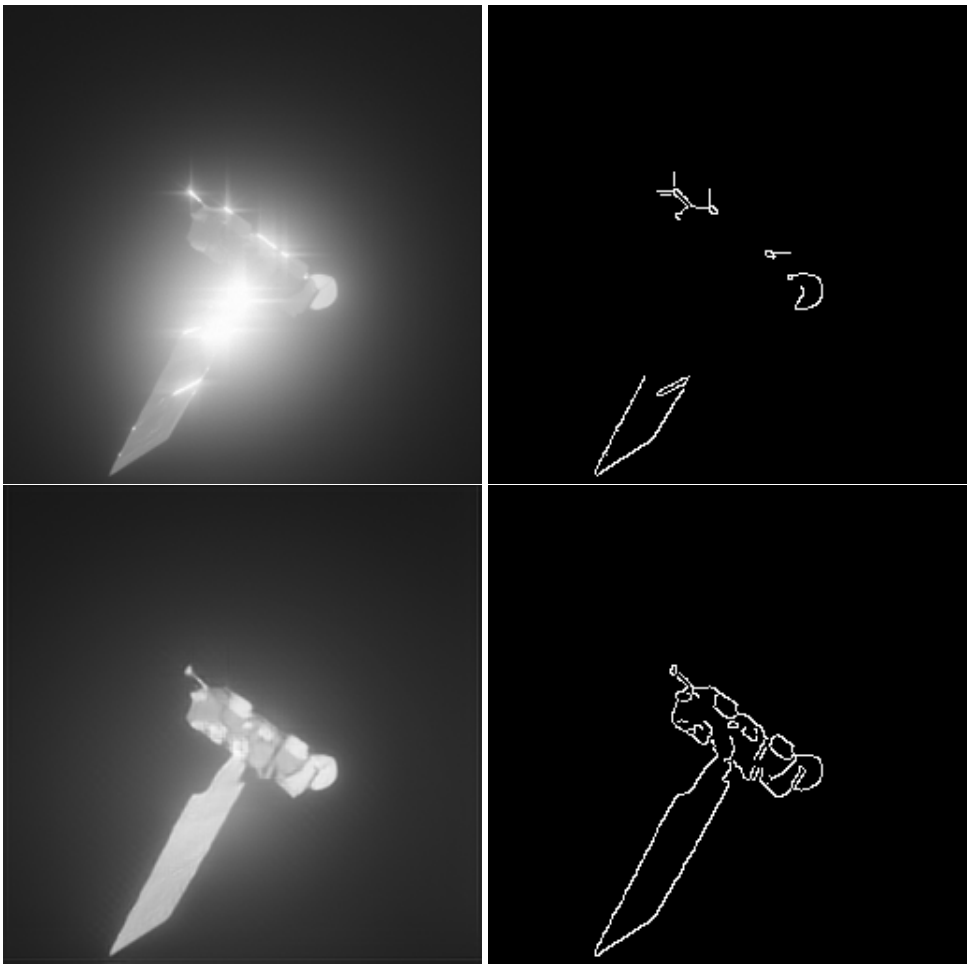

Fig. 9 (LEFT) Corner points on the image without any glare (RIGHT) Corner points on the image with glare

results produced missing portions where the glare broke up the regular pattern on the boundary of the satellite. Results of images that contained glare, passed through GlareNet produced consistently complete results mirroring the ground truth.

X. Future Work

We present this paper as a first step into exploring the use of conditional generative adversarial networks as a means of reducing the effects of glare on an image. As such, there are many ways to improve our approach that we may pursue in the future. We have not yet added other celestial objects to our database. Having the Earth or Moon as a backdrop in the scene would provide a more difficult challenge, as glare from multiple objects in the scene will need to be removed. Furthermore, multiple types of spacecraft are not considered. This would be a step towards a more general approach to glare removal that could work on any satellite. We have used use cases as a measure of our effectiveness, however discovering a metric that is solely related to glare may help more effectively evaluate future attempts at the problem.

XI. Conclusion

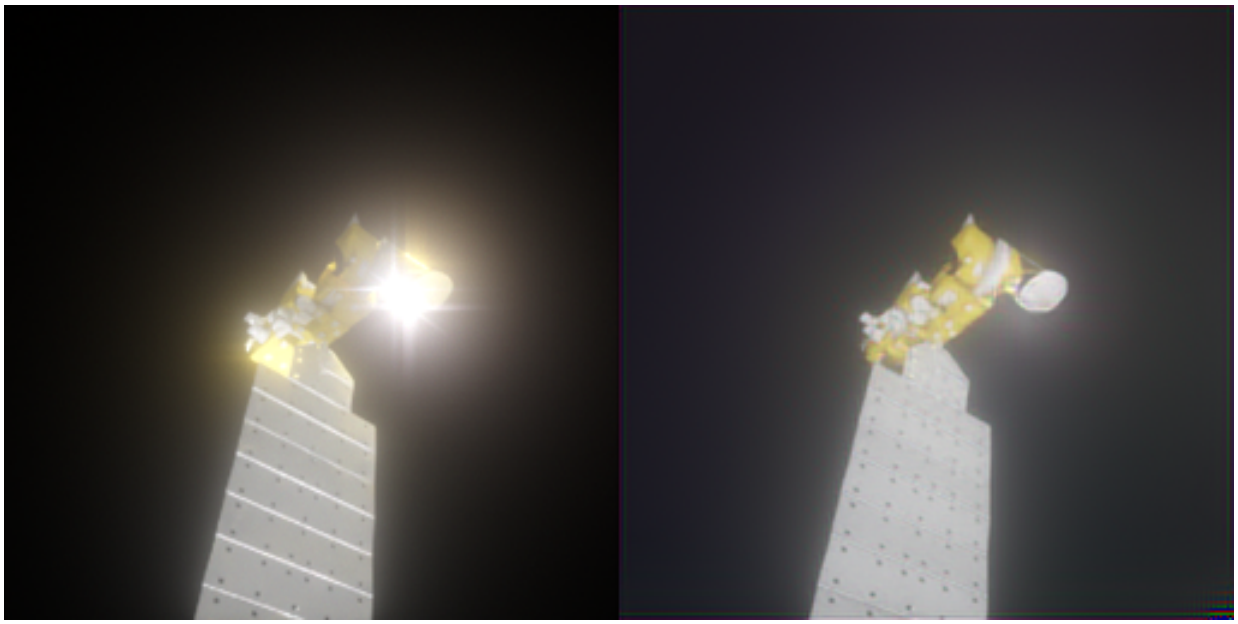

Fig. 10 (RIGHT) Image with glare from testing set (LEFT) Same testing image after GlareNet

By making use of conditional generative adversarial networks we have proposed a method by which to reduce the glare present in images that may be taken of satellites in proximity operations scenarios. This network is able to remove substantial amounts of glare from an image without introducing unwanted artefacting, and is even able to reproduce portions of the satellite that may be occluded. We present methods to effectively evaluate the performance of our network without the need to develop a specific metric to evaluate glare itself, by focusing on common application use cases for these images. This is an important step in being able to use optical systems in adverse conditions during proximity operations. We hope to see its integration and use in a diverse array of systems in the future.

References

[1] Kaneko, T., and Harada, T., "Blur, Noise, and Compression Robust Generative Adversarial Networks," , 2020. https://doi.org/10.48550/ARXIV.2003.07849, URL https://arxiv.org/abs/2003.07849.

[2] Bhattacharya, S. S., Summers, Z., Panchal, A., Koock, E. A.-M., McHenry, N., and Chamitoff, G. E., "Using GANs to Generate Random Surface Materials for High Fidelity Low Altitude Imaging Simulations," *2020 IEEE Aerospace Conference*, 2020, pp. 1–9. https://doi.org/10.1109/AERO47225.2020.9172565.

[3] Radford, A., Metz, L., and Chintala, S., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv e-prints*, 2015, arXiv:1511.06434.

[4] Mirza, M., and Osindero, S., "Conditional Generative Adversarial Nets," , 2014. https://doi.org/10.48550/ARXIV.1411.1784, URL https://arxiv.org/abs/1411.1784.

[5] Frogner, C., Zhang, C., Mobahi, H., Araya-Polo, M., and Poggio, T., "Learning with a Wasserstein Loss," , 2015. https://doi.org/10.48550/ARXIV.1506.05439, URL https://arxiv.org/abs/1506.05439.