SharinGAN - Image DeBlurring with Conditional GANs

Debojeet Chatterjee	Rohit JV	Karthik Buddha	Vivek Vaidyanathan
5295271	5418983	ID 5416304	5416749
chatt086@umn.edu	jakku004@umn.edu	buddh009@umn.edu	vaidy083@umn.edu

Abstract

The objective of this project is to produce a deblurred image from a picture that has blur. The implementation has a Conditional Generative Adversarial Networks with a multi-component loss based off of the approach used in DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks [**primary**]. The implemented model achieved state-of-the-art performance in terms of both structural similarity and visual appearance as mentioned in the base paper[**primary**]. Also this method claims to be 5 times faster than the closest competitor Deep-Deblur [**deep-deblur**]. We also implemented their novel method for generating artificial motion blurred images from sharp original images as a powerful data augmentation technique. We also used a novel technique of evaluation by running the Faster R-CNN [**faster-r-cnn**] object detection model on our deblurred images.

1 Introduction

1.1 Problem Statement

Our work tries to restore a single image with blind motion blur. The common formulation of a non-uniform blur model is given by

$$I_b = k(M) * I_s + N, \tag{1}$$

where I_b is a blurry image, *k(M)* are the unknown motion *blur kernels* given by the motion field *M*. I_s is the sharp latent image, *N* is an additive noise factor and * denotes a convolution operation. Normally, the blur function is treated as unknown, and *blind deblurring* algorithms estimate both the latent sharp image I_s as well as the blur kernels *k(M)*.

1.2 Significance and Impact

1.2.1 Problem Significance

No one likes it when they move their hands a bit while taking a picture and the shot is completely ruined by motion blur. Having a robust deblurring system as part of an image editing suite would be extremely helpful to salvage some of these ruined shots. Object Detection is one of the most popular and heavily researched problems in computer vision with applications in different domains from autonomous driving to security and agriculture. During the last few years, approaches based on Deep Convolutional Neural Networks have showed state of the art performance when compared to older traditional methods. However, these object detection and classification networks are trained on limited datasets and in the real-world, images are often degraded by various natural artifacts, including motion blur. With this work of restoring sharp images from their blurred counterparts, we improve the confidence and performance of object detection and classification models such as Faster R-CNN [**faster-r-cnn**]. Autonomous Driving systems will greatly benefit from better object detection and even Motion Capture systems that quickly capture frames of fast moving objects will be able to perform better.

Figure 1: Deblurring helps Object Detection [**primary**]. YOLO [**yolo**] detections on the blurred image (top), the restored image (middle) and the real sharp image (bottom) from the GoPro [**deep-deblur**] dataset.

1.2.2 Prior Work

Early work [**cv-algos**] has mostly focused on non-blind deblurring, making an assumption that the blur kernels *k(M)* are known. Most of them relied on the classical Lucy-Richardson algorithm, Wiener or Tikhonov filter to perform the deconvolution operation and obtain the I_s estimate. Blind Motion Deblurring is more common, however, as blur kernels are usually unknown and they try to find both I_s and *k(M)*. We can see how finding one blur function for each pixel is an ill-posed problem, and most of the existing algorithms rely on heuristics, image statistics and assumptions on the sources of the blur. This family of methods addresses the blur caused by camera shake by considering a uniform blur across the entire image. First they estimate the camera's motion in terms of the induced blur

kernel, and then reverse the effect by using a transposed convolution operation. Following the success of Fergus et al. [**remove-camera-shake**], many methods have been developed over the last decade based on an iterative approach [**sparse-deblurring**], which improve the estimate of the motion kernel and sharp image iteratively using parametric models. These algorithms, however, suffer greatly in terms of running time and stopping condition. A few others use assumptions of local linearity of the blur function and simple heuristics to quickly estimate the unknown kernel. These methods while being fast, only work well on a small subset of images.

1.2.3 Motivation

This is inspired by the success of approaches in the related areas of image super-resolution [**image-super-resolution**], image-inpainting [**image-inpainting**] and image-to-image translation [**image-to-image-translation**] by employing Generative Adversarial Networks [**gans**]. GANs are known for their ability to preserve texture details in images and generate synthetic images that are very close to the actual image manifold and also look perceptually convincing. We consider Deblurring as a special case of image-to-image translation and utilize their approach as inspiration for our own. Unlike previous work in this direction, we use a Wasserstein GAN [**wgans**] with a gradient penalty term [**wgan-gp**] and perceptual loss [**perceptual-loss**]. This makes the model encourage images that are perceptually hard to distinguish from real sharp images and allows us to restore finer texture details than if we would use traditional pixel-wise Mean Squared Error(L2) or Mean Absolute Error(L1) as a perceptual optimization target.

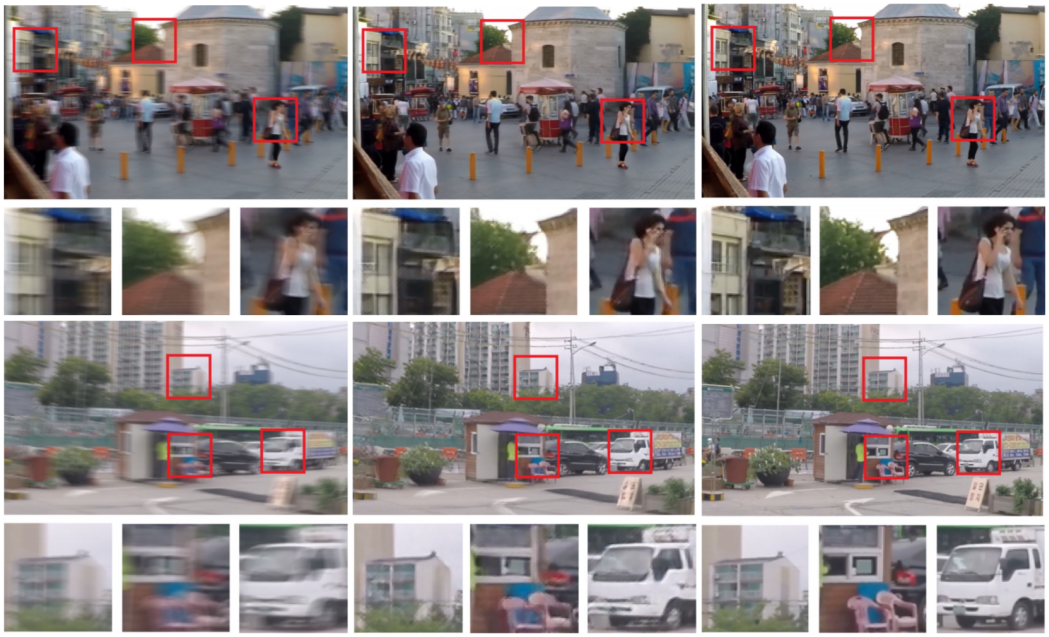

Figure 2: GoPro Images [**deep-deblur**] processed by SharinGAN. Blurred Image (left), Deblurred Image (center), Real Sharp Image (right)

1.2.4 Contributions

Our solution comprises of three contributions. First, we utilise a loss function and a network architecture which can obtain state-of-the art results in motion deblurring, while being 5x faster than the closest competitor. Secondly, we provide a method based on random trajectories for effective dataset augmentation for motion deblurring training in an automated fashion from just a collection of original sharp images. Finally, we present a novel evaluation method of deblurring algorithms based on how well they improve object detection results using Precision, Recall and F1-Scores.

2 Background

2.1 Generative Adversarial Networks

Generative adversarial networks invented by Ian Goodfellow et. al[**gans**] can be defined as a min-max game between two competing networks, one generative and one discriminative. This is analogous to a team of money counterfitters and law enforcement, one trying to trick the other with counterfit money and the other trying to distinguish a counterfit from the real deal. A discriminative network D by definition is a classifer network that, when given the features of a data instance, attempts to predict a label or category to which that data belongs. [**intro-to-gan**] On the other hand, a generative network G tries to understand the underlying model and produces a distribution that accurately captures it. The generative model in a GAN produces synthetic data from noise. [**wgan-gp**]

To learn the generator's distribution p_g over data x, we use a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z, \Theta_g)$, where G denotes a differentiable function represented by a multilayer perceptron with parameters Θ_g corresponding to the model of the generator. We also define a second multilayer perceptron $D(x, \Theta_d)$ that outputs a single scalar corresponding to the class label. $D(x)$ is the probability that x came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $log(1 - D(G(z)))$. In other words, the game V can be defined as

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)}[log(D(x))] + E_{z \sim p_z(z)}[log(1 - (D(G(z))))] \tag{2}$$

2.2 Limitations of a Vanilla GAN

Another way of looking at this is, training a GAN requires finding a Nash equilibrium of a non-convex game with continuous, highdimensional parameters. Additionally, GANs are typically trained using gradient descent techniques that are designed to find a low value of a cost function, rather than to find the Nash equilibrium of a game. However, as we know, these algorithms may fail to converge. [**improved-techniques-gans**]

So, despite being a breakthrough in the world of machine learning, a vanilla GAN is hard to train and even after that, it proves to be non-robust in multiple scenarios. Apart from failing to converge, it also suffers from mode collapse, vanishing gradients etc in the context of training. [**improved-techniques-gans**] [**min-max-gen-dis**]

In addition to the pitfalls of training a vanilla GAN, we also plan to feed the generative model with an actual input (a blurry image) as opposed to just noise that a GAN usualy works with. It is also important to keep in mind that we are using the project in order to improve object detection among others, and apart from "tricking" the discriminative model, an important job is to make sure that our generative model guarantees images that can perform well when used for object detection.

2.3 Conditional GAN

A conditional GAN works in with the techniques that GAN does, except it operates in the conditional scenario. The applications of GANs can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y could be any kind of auxiliary information, such as class labels or data from other modalities. We can perform the conditioning by feeding y into the both the discriminator and generator as additional input layer. [**cgans**]

In the generator the prior input noise $p_z(z)$, and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. This application is especially useful for training our model as we would like our generator to generate sharp images from an image with a blur kernel applied to it, as opposed to producing one out of thin air.

Unlike vanilla GAN, a cGAN learns a mapping from observed image x and random noise vector z, to $y : G : x, z \rightarrow y$.

In the discriminator x and y are presented as inputs and to a discriminative function. Now the objective function of the two-player minimax game would be:

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[log(D(x|y))] + E_{z\sim p_z(z)}[log(1-(D(G(z|y))))] \quad (3)$$

2.4 Wasserstein GAN

The inventors of WGANs [**wgans**] also mention the difficulties of training a GAN by saying that the divergences which GANs typically minimize are potentially not continuous with respect to the generator's parameters. They propose instead using the Earth-Mover distance (also called Wasserstein-1) distance $W(q,p)$, which is informally defined as the minimum cost of transporting mass in order to transform the distribution q into the distribution p (where the cost is mass times transport distance). Under mild assumptions, $W(q,p)$ is continuous everywhere and differentiable almost everywhere. This means that these flavours of problems are atleast solvable and will most certainly converge. The WGAN value function is constructed using the Kantorovich-Rubinstein duality [**optimal-transport**] to obtain the modified equation:

$$\min_G \max_{D\in\mathcal{D}} = E_{x\sim\mathcal{P}_r}[D(x)] - E_{\tilde{x}\sim\mathcal{P}_g}[D(\tilde{x})] \quad (4)$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions and $\mathcal{P}_g$ is the model distribution implicitly defined by $\tilde{x} = G(z), z \sim p(z)$. In that case, under an optimal discriminator (called a critic [**wgans**], since it's not trained to classify), minimizing the value function with respect to the generator parameters minimizes $W(\mathcal{P}_r, \mathcal{P}_g)$.

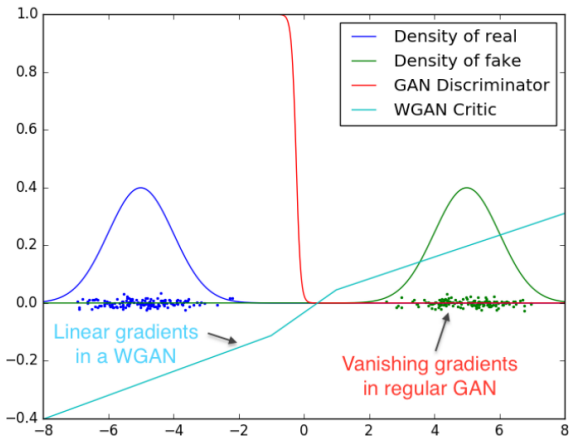

Figure 3: The discriminator of a minimax GAN saturates and results in vanishing gradients in an experiment conducted in[**wgans**]. WGAN critic, on the other hand is seen to provide very clean gradients on all parts of the space.

The WGAN value function results in a critic function whose gradient with respect to its input is better behaved than its GAN counterpart, making optimization of the generator easier. Additionally, WGAN has the desirable property that its value function correlates with sample quality, which is not the case for GANs

It is argued in [**wgan-gp**] that the WGAN optimization process is also because of interactions between the weight constraint and the cost function, which result in either vanishing or exploding gradients without careful tuning of the clipping threshold c.

To demonstrate this, WGAN is trained on the Swiss Roll toy dataset, varying the clipping threshold c in [10^{-1}, 10−2, 10^{-3}], and plot the norm of the gradient of the critic loss with respect to successive layers of activations. Both generator and critic are 12-layer ReLU MLPs without batch normalization, and as a result, it is shown that for each of these values, the gradient either grows or decays

exponentially as we move farther back in the network. We find our method results in more stable gradients that neither vanish nor explode, which has some unexplored options. [**wgan-gp**]

2.5 WGAN - Gradient Penalty

In the same paper, there is an alternate way proposed to impose the Lipschitz constraint. A differentiable function is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere, so we consider directly constraining the gradient norm of the critic's output with respect to its input. To circumvent tractability issues, we enforce a soft version of the constraint with a penalty on the gradient norm for random samples $\hat{x} \sim \mathcal{P}_{\hat{x}}$. Our new objective along with this "penalty" can be written as:

$$\min_G \max_D V(D,G) = \underbrace{E_{x\sim\mathcal{P}_r}[D(x)] - E_{\tilde{x}\sim\mathcal{P}_g}[D(\tilde{x})]}_{\text{WGAN Loss}} + \underbrace{\lambda E_{\hat{x}\sim\mathcal{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{Gradient penalty suggested in [wgan-gp]}} \quad (5)$$

The following definitions were supplied in [**wgan-gp**] in order to understand the nuances of the equation.

2.5.1 Sampling distribution

$P_{\hat{x}}$ is defined implicitly sampling uniformly along straight lines between pairs of points sampled from the data distribution $\mathcal{P}_r$ and the generator distribution $\mathcal{P}_g$. This is motivated by the fact that the optimal critic contains straight lines with gradient norm 1 connecting coupled points from $\mathcal{P}_r$ and $\mathcal{P}_g$. Given that enforcing the unit gradient norm constraint everywhere is intractable, enforcing it only along these straight lines seems sufficient and experimentally results in good performance.

2.5.2 Penalty coefficient

(λ) We used a lambda value of 100 to penalize.

2.5.3 No critic batch normalization

Some GAN implementations use batch normalization in both the generator and the discriminator to help stabilize training, but batch normalization changes the form of the discriminator's problem from mapping a single input to a single output to mapping from an entire batch of inputs to a batch of outputs. The gradient penalized training objective is no longer valid in this setting, since we penalize the norm of the critic's gradient with respect to each input independently, and not the entire batch. To resolve this, we simply omit batch normalization in the critic in our models, finding that they perform well without it. Our GAN works with normalization schemes which don't introduce correlations.

3 The Model

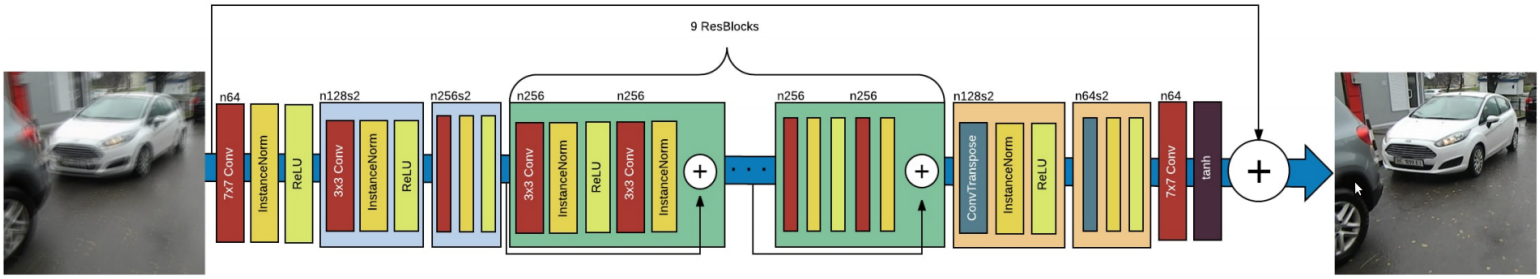

Figure 4: DeblurGAN generator architecture [**primary**]. It contains two strided convolution blocks with stride length of 2, 9 ResBlocks and two transposed Convblocks. Each ResBlock consists of a convolution layer, instance normalization layer, and ReLU activation

3.1 Networks

The CNN generator model is as described in [**perceptual-loss**] and shown in the figure. This model was initially used for style transfer task in the same paper. It contains two strided convolution blocks with stride legnth of 2 blocks, nine residual blocks[**resnet**] (ResBlocks) and two transposed convolution blocks. Each ResBlock consists of a convolution layer, instance normalization layer [**wgan-gp**], and ReLU activation. Dropout regularization with a probability of 0.5 is added after the first convolution layer in each ResBlock. In addition, we introduce the global skip connection which we refer to as ResOut. CNN learns a residual correction I_R to the blurred image I_B, so $I_S = I_B + I_R$. We find that such formulation makes training faster and resulting model generalizes better. During the training phase, we define a critic network D_{Θ_D} , which is Wasserstein GAN [**wgans**] with gradient penalty. The architecture of critic network is identical to PatchGAN [**patchgan**]. All the convolutional layers except the last are followed by InstanceNorm layer and LeakyReLU with $\alpha = 0.2$[**primary**]

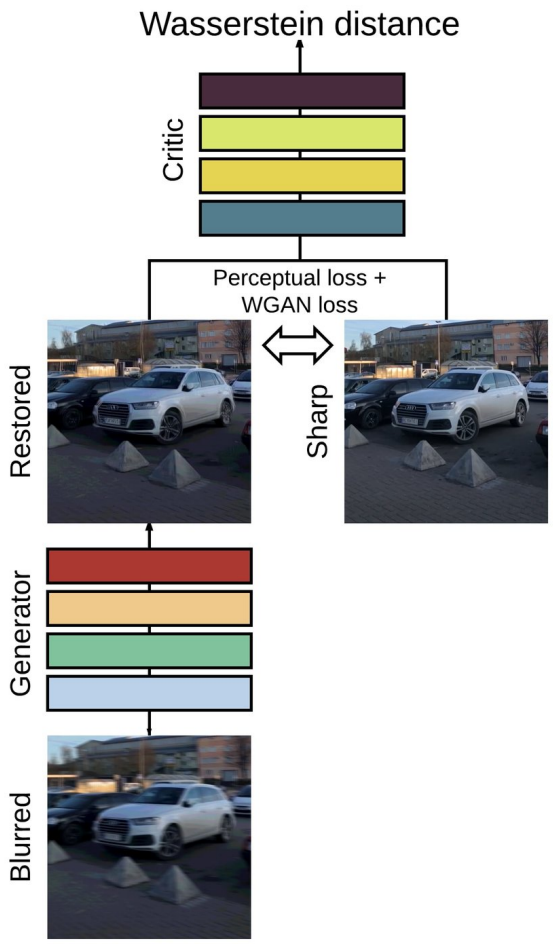

Figure 5: DeblurGAN training. The generator network takes the blurred image as input and produces the estimate of the sharp image. The critic network takes the restored and sharp images and outputs a distance between them. The total loss consists of the WGAN loss from critic and the perceptual loss [**perceptual-loss**]. The perceptual loss is the difference between the VGG-19 [**very-deep-cnn**] $conv3.3$ feature maps of the sharp and restored images. At test time, only the generator is used. [**primary**]

3.2 Loss function

The loss function used in the GAN is decomposed into 2 parts:

$$\mathcal{L} = \mathcal{L}_{Adversarial} + \lambda \mathcal{L}_{Perceptual} \tag{6}$$

The two components are defined as follows:

1. **Adversarial loss :**
   It is important to note that we do not use the vanilla GAN objective as the loss function, which is shown below.

   $$\min_G \max_D V(D,G) = E_{x\sim\mathcal{P}_r} log[D(x)] + E_{\tilde{x}\sim\mathcal{P}_g} log[1 - D(\tilde{x})] \tag{7}$$

   Instead, we use the WGAN-GP loss function, which was discussed in the sections above.

   $$\min_G \max_D V(D,G) = \underbrace{E_{x\sim\mathcal{P}_r}[D(x)] - E_{\tilde{x}\sim\mathcal{P}_g}[D(\tilde{x})]}_{\text{WGAN Loss}} + \underbrace{\lambda E_{\hat{x}\sim\mathcal{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{Gradient penalty suggested in [wgan-gp]}} \tag{8}$$

   As mentioned by Orest Kupyn during his talk at EECVC 2018 [**deblurgan-talk**], it was shown by Ian Goodfellow that the choice of loss function doesn't really matter, however in practice, using WGAN-GP is more robust to the selection of the generator architecture.

2. **Perceptual loss :**
   While the GAN adversarial loss aims to let the discrimintor network and the generator network to converge to a state of equilibrium, the perceptual loss supplements the adversarial loss, and helps to not only converge, but converge to a more meaningful state.

   Multiple choices for perceptual loss are possible, including the *L1* or *MAE* loss, *L2* or *MSE* loss on raw pixels. However, using these functions mean that the optimal solution would be a pixel-wise average of the possible solutions. This often leads to blurry results. [**primary**]

   A more meaningful loss to use is the *L2* loss on some feature maps of the generated and the original sharp image. This leads to better comparision, since we do not take the loss on the actual pixel values, but on the *more representative* feature maps of the two images. Particularly, we use the pretrained VGG-19 network to extract the feature maps of the 2 images. VGG-19 is trained on a large dataset, and is therefore a good default choice for extract feature maps of a random image. We extract the feature maps on layer 14 of the VGG-19 network for both the images, and compute the *L2* loss.[**primary**]

In summary, while the perceptual loss helps in restoring the general content of the image, the adversarial loss aims at restoring the texture details.

When it comes to backpropagating the loss through the networks, it is worth noting that the adversarial loss decomposes into two terms, one each for the generator network and the discriminator network. Further the perceptual loss, which is calculated on the original sharp image and the generated sharp image, is relevant only to the generator network. As a result, the discriminator network's component of the adversarial loss is backpropagated through the discriminator network, while the complementary part of the adversarial loss is added to the perceptual loss, and then backpropagated through the generator network.

3.3 Training

Similar to the base paper [**primary**] we implemented all of our models using PyTorch deep learning framework. The training was performed on a single core of a 1.5GHz 12GB GDDR5X NVIDIA GTX Titan-X GPU on the GoPro dataset, with 1000 images.

These are 256x256 RGB images which were downscaled by a factor of 2. In order to optimize, we use the methods suggesed in the [**wgans**][**wgan-gp**] and perform 5 steps over D_{Θ_D} whith one step on G_{Θ_G} with Adam Optimizer as a solver [**adamopt**]. This model is trained over 300 epochs for 17 hours with nearly 210 seconds per epoch.

4 Results

The results on the test images is shown in Figure 6. The generator and the discriminator networks are trained on a dataset where majority of the images are not plants whereas the test dataset majorly

contains the aerial images of weed fields. From this, it is evident that the model learns the blur features rather than the objects in the training dataset. All the test input, output and the ground truth images are passed through an object detection and classification network (Faster RCNN). The results in Figure 7 through 9 show object detection results on original, blurred and deblurred images. From the results, it is clear that our GAN greatly increases the objectness scores. This deblurring technic not only helps in cleaning the blurred images but also helps in getting sharp resized aerial images captured from high altitudes. The Peek Signal to Noise Ratio (PSNR) on a test set of size 300 is 38.910459 which clearly indicates the deblurred image has very close resemblance to the original image. The Structural Similarity Index (SSIM) values averaged over 300 images is 0.856854 also manifests the good performace of the model.

Figure 6: Test Images - the original image (left), blurred image(center), and SharinGAN's deblurred image (right)

Figure 7: Object Detection & Classification on the Original Image

Figure 8: Object Detection & Classification on the Blurred Image

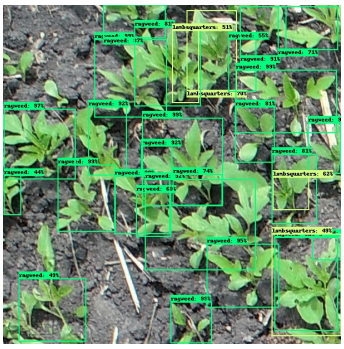

Figure 9: Object Detection & Classification on the Deblurred Image

5 Conclusion

This project helped us in exploring conditional GANs and its application in deblurring the motion blurred Images. One application of this project is for improved Object Detection & Classification results which can solve various problems like plant detection and classification in Precision Agriculture and several other applications such as autonomous driving.