CS499 Project Report

DeblurGAN

Guide: Prof. Shanmuganathan Raman

Ph.D. Mentor: Prajwal Singh

Madhav Kanda	Dhruv Patel	Ksheer Sagar Agrawal
20110104	20110129	20110098

DeblurGAN Report

Introduction:

As a part of the project course, we studied the paper "Deep Generative Filter for Motion Deblurring", which proposed a Generative Adversarial Network (GAN) for deblurring motion images. The paper used Lua code, making it difficult to perform further experimentation. Hence, we implemented the entire GAN for deblurring in Pytorch. This report will explain our experiments, modifications, and results that we did on top of our Pytorch implementation.

Implementation:

The original code associated with the paper is in Lua, but we implemented the entire research paper in Pytorch to perform comprehensive experimentation. We faced several issues while implementing the research paper. This includes:

1. The Go Pro dataset that we have used has sharp and blur images in different folders. Thus, we had to come up with an approach where both the blur and the real image could be fed together into the model. Thus, based on the implementation of Pix2Pix in Pytorch, we concatenated the two images and fed the model through the dataloader.
2. Owing to the high number of filters, we weren't able to fit the entire model on the GPU. Thus, we initially reduced the number of filters and reduced the batch size. Later on, we modified the codebase to use the two GPU cores parallely using DataParallel from the torch. Further, we reduced the usage of GPU by incorporating autocast which is a context manager that runs the code in mixed precision.

We weren't able to compare our implementation based on the metrics mentioned in the research paper as they used a combination of GoPro images and randomly sampled images from MS-COCO and Imagenet. Thus, recreating the exact dataset is not possible. Hence, for all our experiments we have used the GoPro dataset containing **2103 pairs of training images (blur and real)** and **534 pairs of test images (blur and real).**

These images were pre-processed to obtain images of the size 512*256 (width* height). Each such image was a concatenation of corresponding real (256*256) and fake (256*256) images.
Link to our GitHub repository: https://github.com/ksheersagaragrawal/DeblurGAN

Loss

For the loss, we use the summation of three losses generative adversarial loss, L1 loss and LPIPS loss.

$$\mathcal{L}\mathsf{GAN}(G, D) = \mathbb{E}x \sim p_{\mathsf{data}}(x)[\log D(x)] + \mathbb{E}z \sim pz(z)[\log(1 - D(G(z)))]$$

$$\mathsf{LPIPS}(I_1, I_2) = \sum_{i=1}^{N} w_i \cdot d_i(f_i(I_1), f_i(I_2)),$$
$$d_{\mathsf{i}}(a, b) = \frac{1}{2}\left(1 - \mathsf{SSIM}(a, b)\right)^2,$$
$$\mathsf{SSIM}(a, b) = \frac{(2\mu_a\mu_b + c_1)(2\sigma_{ab} + c_2)}{(\mu_a^2 + \mu_b^2 + c_1)(\sigma_a^2 + \sigma_b^2 + c_2)}.$$

Experimentation and Methods Proposed:

➔ Original

- We trained the implemented GAN model on the training dataset and evaluated it on the test dataset to obtain the following results:

PSNR	SSIM	MS - SSIM	F - SIM	VIF
21.01	0.789	0.904	0.891	0.412

➔ Experiment - 1

- To normalize the outputs from each layer we were earlier applying batch normalisation, but since the batch size = 1 (owing to the computation resource constraints) it made no sense as there is no other image in the set to normalize over. Thus, we introduced Instance Normalisation which normalizes the image over the channels.
- We replaced the VGG perceptual loss with LPIPS loss in the generator. LPIPS loss has shown to exhibit better results in many cases owing to its robustness to color and contrast, better correlation, etc.
- The results that we obtained using this approach were significantly better than the previous results. Further, this significantly decreased the time to obtain the results. The results obtained using this experimentation were as follows:

PSNR	SSIM	MS - SSIM	F - SIM	VIF
23.47	0.8438	0.948	0.903	0.506

➔ **Experiment - 2**

- We explored the pytorch library for reducing the computation and for parallelization so as to increase the batch size. Based on it used the autocast and DataParallel method from the torch library. Autocast is a context manager that allows regions of code to run with mixed precision, this maintains accuracy while improving performance significantly. DataParallel method helped us to use the two GPU's so as to get more computation power.

➔ **Experiment - 3**

- We used the WGAN loss for better training stability, gradient flow, and image quality. The results obtained using this are as follows:

PSNR	SSIM	MS - SSIM	F - SIM	VIF
22.022	0.819	0.9316	0.8972	0.536

➔ **Experiment - 4**

- We introduced differential augmentation to control the problem of vanishing gradients in discriminators while learning. This method feeds the model with images having different augmentations so that the learning keeps on happening for the discriminator. The results obtained are as follows:

PSNR	SSIM	MS - SSIM	F - SIM	VIF
23.51	0.8396	0.9470	0.900	0.503

Conclusion:

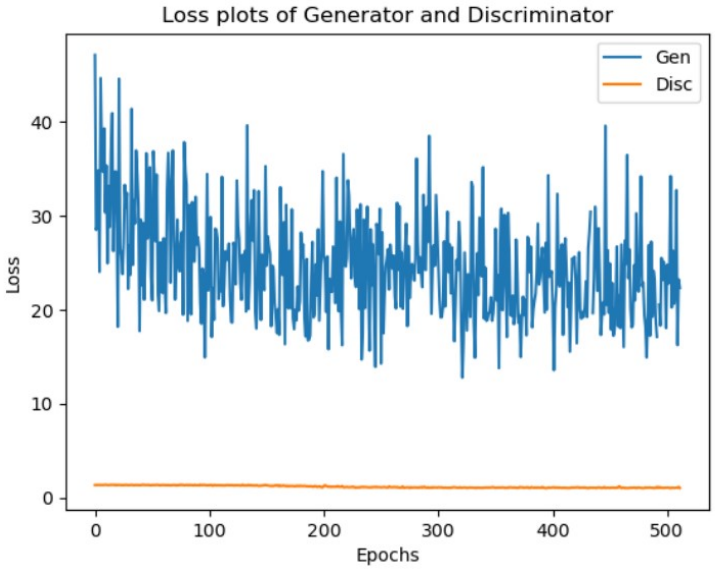
As can be seen, we have significantly improved the results by introducing many new methods in the initial model. Since we have been using a small dataset for training the GAN model and have been able to produce significant results, we wish to continue our research in this field and incorporate other methods to significantly boost our results.

Methods	PSNR	SSIM	MS - SSIM	F - SIM	VIF
Original	21.01	0.789	0.904	0.891	0.412
Experiment - 1	23.47	0.8438	0.948	0.903	0.506
Experiment - 3	22.022	0.819	0.9316	0.8972	0.536

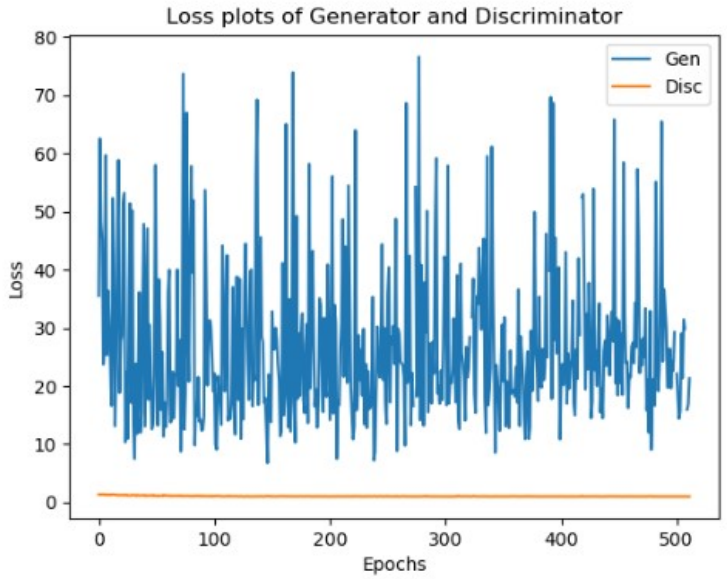
Experiment - 4	23.51	0.8396	0.9470	0.900	0.503

Loss Plots

1. **Differential augmentation**



2. **LPIPS loss**

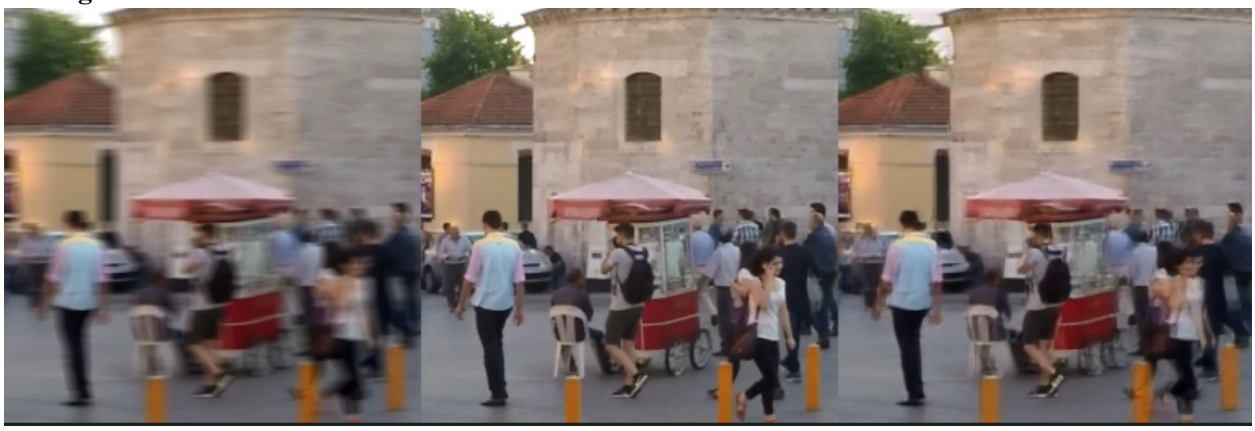


RESULTS

LPIPS

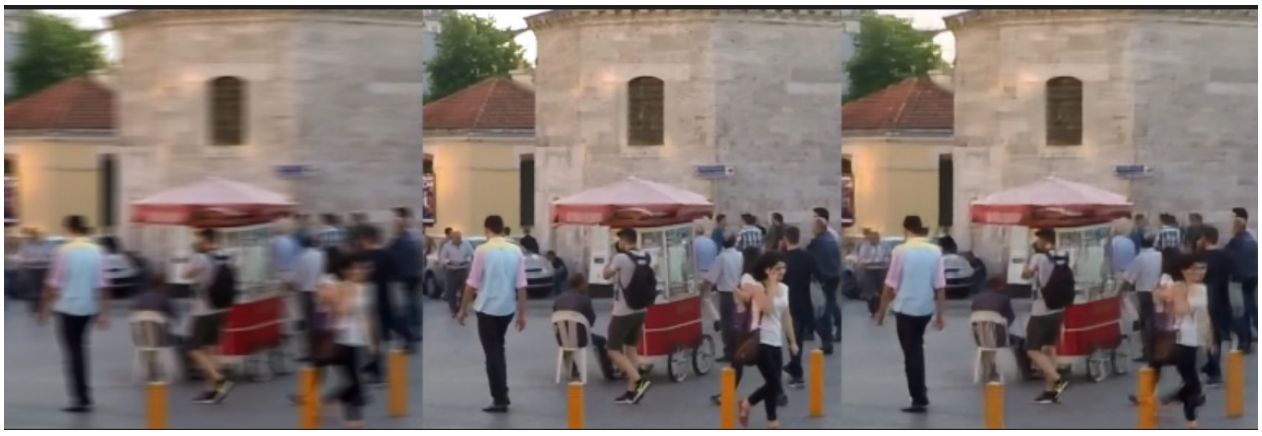
Blur **Sharp** **Generated**

Diffaug



Blur **Sharp** **Generated**

Wgan



Blur **Sharp** **Generated**

Future Work

A comparison with similar works done using diffusion models would be helpful in providing context and insights into the performance of the model. This comparison could involve assessing the strengths and weaknesses of different diffusion models used in previous studies and how they compare with the current model. This would enable a better understanding of the limitations of the current model and potential areas for improvement. Regarding the current model's performance in a low data regime, further techniques could be explored to enhance the results. One approach could involve exploring transfer learning techniques that can leverage pre-trained models or knowledge from related tasks to improve the model's performance in low data regimes.