DeBlurGAN

Single Image Deblurring

NNFL Term Project

Group ID: 132

2017A7PS0077P - Aryan Mehra

2017A7PS0109P - Saujas Adarkar

2017A7PS0127P - Siddhant Khandelwal

Image Enhancement

Image enhancement includes various image cleaning tasks that are usually a pre-processing task for other major tasks like object detection, feature extraction, autonomous driving applications etc. Various image enhancement tasks include: Dehazing, Deblurring, Deraining....

Deblurring as a task (Traditional Approaches)

Blurry images are caused due to motion of the camera lense, rotational components, or slight movement on the part of the target itself.
The blur is modelled by the following equation:

$$IB = k(M) * IS + N$$

IB is the blurry image
IS is the sharp image
k(M) is the unknown blur chanel
* the convolution operation

GAN based Deep Learning

Generative Adversarial networks

- Main concept is very clear. Generator tries to fool the discriminator.
- In general training is tough
- Two networks need to converge simultaneously
- One has to be kept frozen while other trains.

Used mainly for image to image mapping problems where general methods fail.

DeblurGAN

DeblurGAN is an end-to-end learned method for deblurring based on conditional GAN and content loss. It achieves state of the art performance both in structural similarity and visual appearance.

The original paper is implemented in PyTorch and uses VGG-19 feature maps for perceptual loss, while our implementation uses Keras with Tensorflow backend, along with VGG-16 for perceptual loss.

Dataset

The dataset we chose was the Go_Pro-Large Dataset. It consists of 2103 train images and 1111 test images as a default train test split.

Preprocessing

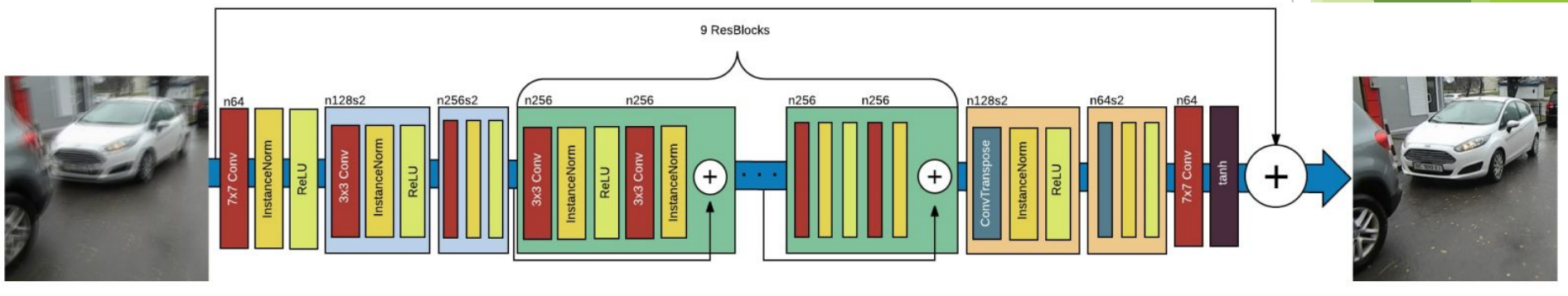
Each image was preprocessed using python scripts into a numpy file in a “.NPY” format. This reduced the dataset from 8.9GB to 1.3GB. Moreover while looping during training, the images in NPY format are loaded faster and thus the training process is faster. This was learnt and used by past experience of team members while training networks.

So, after preprocessing, all our files were present as 256x256x3 numpy arrays arranged suitably in files and folders to facilitate an appropriate train-test divide.

Referred implementation

We focussed our entire attention on the network architecture and training (which is the task given to us). The figure below shows the pictorial representation of the generator network from the original DeblurGAN paper.

[Link to Google Colab Implementation]



Training parameters and choices along the way..

- We trained for 200 epochs (original implementation was 300 epochs)
- Perceptual loss is implemented using VGG16 one by one instead of MobileNetv2 as communicated with TA and also with Russian data scientist Raphael Meudec who is the owner of the official DeblurGAN keras implementation repository on github.

Raphaël Meudec
to me, SIDDHANT, SAUJAS

Wed, Nov 6, 8:13 PM (8 days ago)

Hello Aryan,

It's actually interesting questions, I just re-read the original paper and authors are using VGG19 for their perceptual loss. When I implemented it, I took the definition of the perceptual loss from its original paper: Perceptual Losses for Real-Time Style Transfer and Super-Resolution, which uses a VGG-16 based perceptual loss. I guess it does not make much difference as the architectures are quite similar until the block3_conv3.

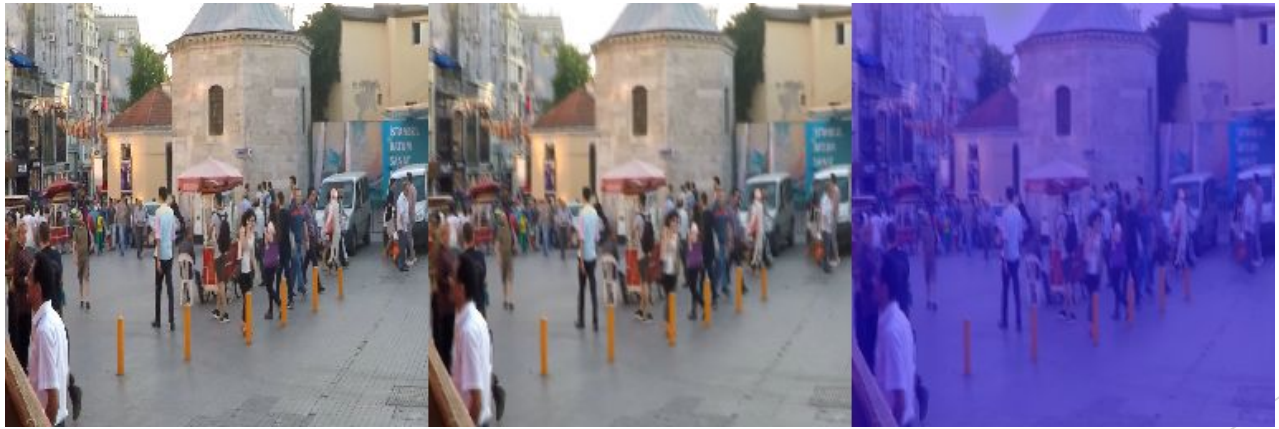
If you plan to take a totally different network, I guess you would have to look at the activations and take a layer which effects would be similar. I'm not sure why you would take the Mobilnet though? The advantage of mobilenet is mainly processing time, but this would help to only compute the loss a bit faster?

Note that they also have published a V2 of their deblurgan here: https://arxiv.org/abs/1908.03826

Hope this help, feel free to ask if you have other questions,
Raphael

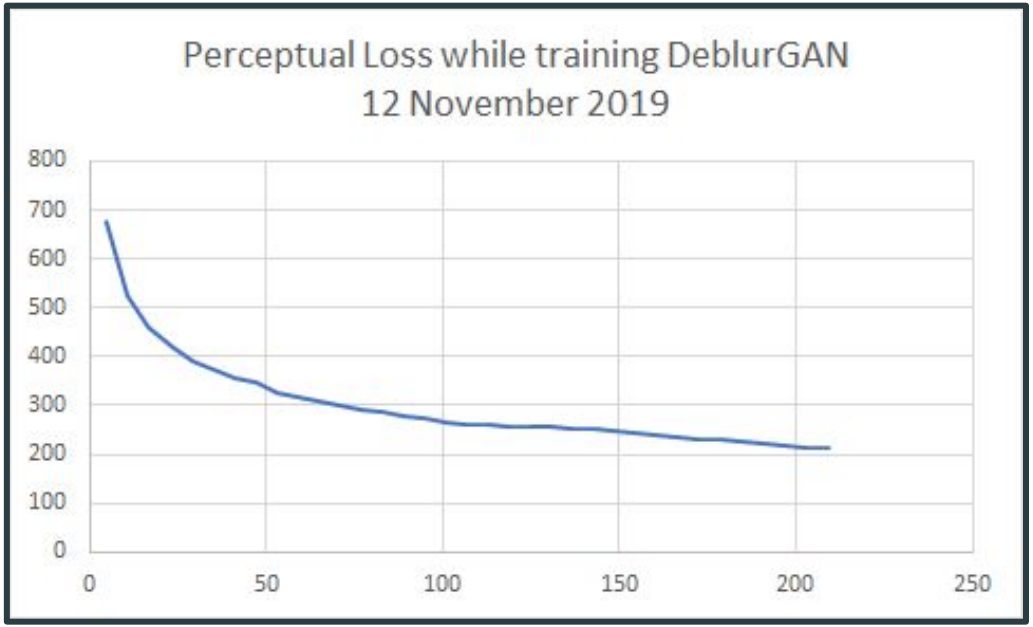
Nevertheless, we tried with MobileNetv2 as well !

Results with MobileNetv2 have a clear bluish tinge, and deblurring was also not visible, which means that layers analogous to block3-conv3 of VGG-16 are not easily identifiable in MobileNetv2 because of a completely different architecture.



Visualising training and model convergence, with VGG-16

- We trained for 200 epochs (original implementation was 300 epochs)
- We stopped after 200 because of time constraints and more importantly because of early stopping to prevent overfitting. The number of images is only 2000. The dataset, although large, is not very diverse.



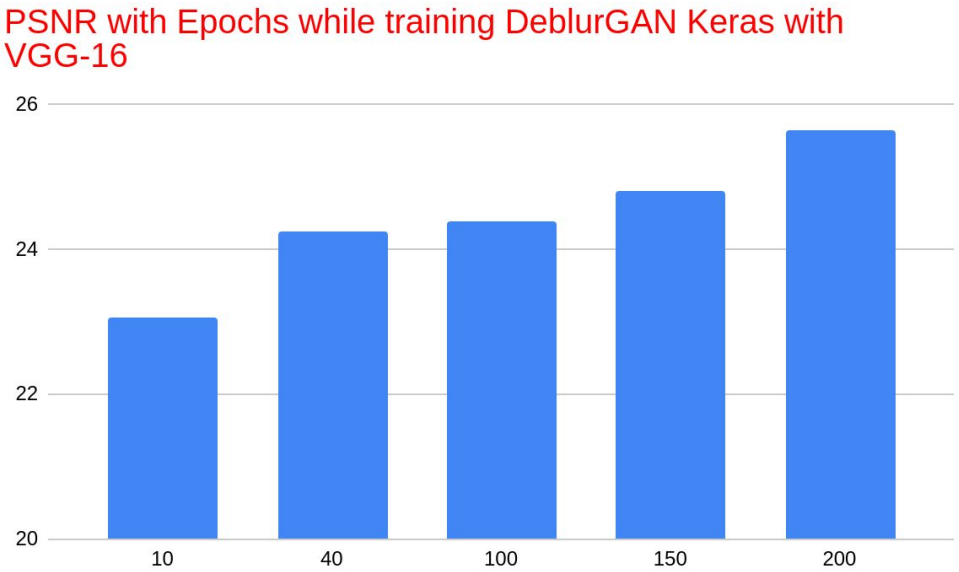
38%| | 38/100 [2:14:42<3:38:10, 211.14s/it]
-0.49999999979138376 228.23302

63%| | 63/100 [3:43:01<2:10:16, 211.25s/it]
-0.49999999623000624 215.76013

100%| | 100/100 [5:53:41<00:00, 211.88s/it]
-0.5 205.53262

Live Epoch-wise PSNR reduction.

The PSNR is seen to be steadily increasing while training. So we knew our model was converging well.



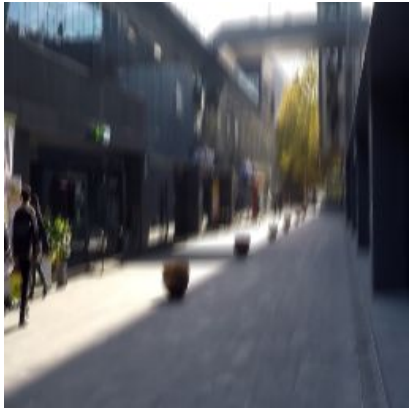
Final Quantitative Analysis

Model	PSNR	SSIM
DeblurGAN Original Paper	28.7	0.9580
Our Implementation	25.4	0.8208

	Sun *et al.*	Xu *et al.*	DeblurGAN		
Metric			*WILD*	*Synth*	*Comb*
PSNR	24.6	25.1	27.2	23.6	**28.7**
SSIM	0.842	0.89	0.954	0.884	**0.958**

- Our preliminary implementation trained for only 200 epochs is able to beat 3 out of 5 relevant state of the art in Deblurring, that are equal to or below the stated implementation of DeblurGAN

Qualitative Analysis



Blurry Predicted Target



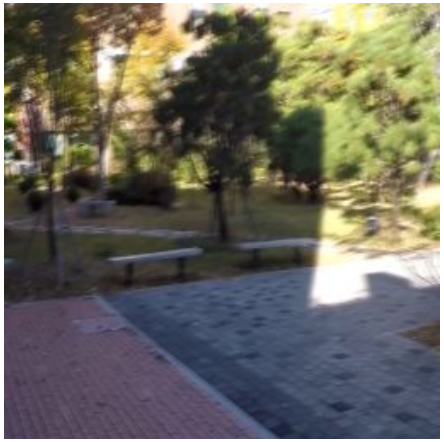
- No contextual loss
- No additional haze/blur introduced
- Visual appeal
- No contrast present
- No image enhancement required

Qualitative Analysis

- No contextual loss
- No additional haze/blur introduced
- Visual appeal
- No contrast present
- No image enhancement required

Blurry Predicted Target

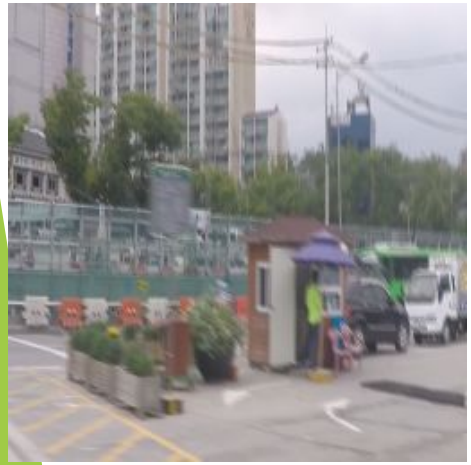
Qualitative Analysis

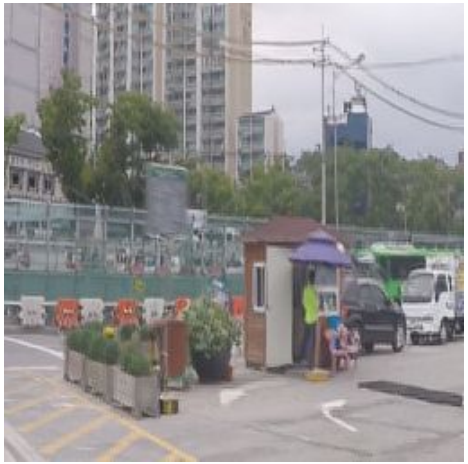


Blurry

Predicted

Target



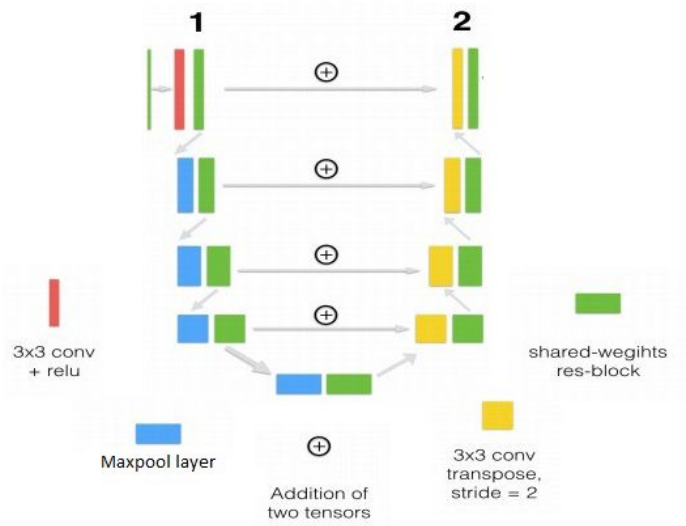


- No contextual loss
- No additional haze/blur introduced
- Visual appeal
- No contrast present
- No image enhancement required

CNN based extension...

We designed a shallow CNN just to test and learn more about why GANs were chosen for this problem statement?

Our CNN was based on UNET like structure with 3 encoder and 3 decoder blocks and three skip connections.



RESULTS
PSNR:
Ranges between 22.707 and 27.277512 !!!

And the image is still blurry !!

Timeline and Work Summary

15 Day Timeline based on correspondence with Saksham Consul (TA)

- Day1: Preprocessing (NPY file generation and cropping)
- Day 2 and 3: GAN coding
- Day 4: MobileNetv2 exploration for perceptual loss
- Day 5: Communication with data scientist (Raphael Meudec), peers and TA mentor
- Day 6, 7 and 8: GAN coding and training
- Day 9 and 10: Training on Co-lab, IPC servers and local machines
- Day 11 and 12: Testing for PSNR and SSIM
- Day 13 and 14: Extension of work for CNN based Deblurring
- Day 15: Presentation work

Results and Conclusions !

- We successfully implemented perceptual loss and how it is better than PSNR for our problem.
- We successfully understand how GANs are trained by freezing weights of discriminator while training the generator. And how 2 lossed are combined while compiling the model.
- We got a PSNR of 25.4 and SSIM of 0.8208
- Our model successfully deblurs the images without loss of context and contrast
- Our model successfully converged (Both Generator and Discriminator together !!)
- We extended the project and compared our results to a shallow CNN, showing why GANs are prefered over CNN in such tasks.



Thank You for your attention!

Any Questions?