Image-to-Image Translation with Conditional Adversarial Networks

Authored by:-
Phillip Isola, Jun-Yan Zhu, Tinghui Zou, Alexei A. Efros (BAIR, UC Berkeley)

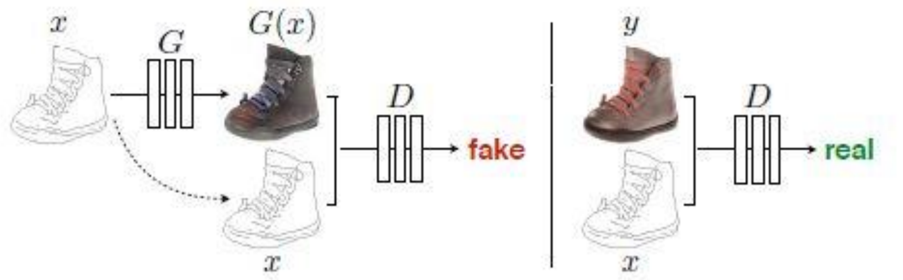
Nikhil Agrawal - 2016A4PS0248P
Pavan R - 2016A4PS0405P
Sampras Lopes - 2016A7PS0125P

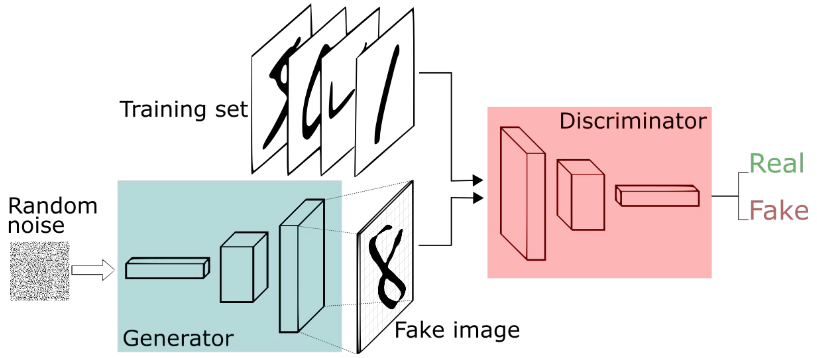
Introduction:-

- Conditional GANs or cGANs in short are CNNs that can be used for image translations.
- Normal methods which use L2 losses between the target pixels and input pixels, creates blurred images. This has been overcome by GANs, which train the data to be indistinguishable from the input.
- In the paper, the authors have used GAN in a slightly modified setting with a modification in the loss function.
- Some significant differences in GAN and the proposed method (cGAN) by authors are as follows:

Some differences between unconditional GAN and cGAN

- Below images show the main difference between cGAN and GAN.





- In cGAN, the input image is shown to the generator as well but in unconditional GAN, random noise is shown to the generator.
- As a consequence, there is also a difference is in the loss function.

Methodology

- The datasets used are images which needed to be pre-processed.
- U-net architecture was implemented for the generator. The parameters which were used by the authors were used.
- A discriminator to distinguish real and fake images.
- Model was trained on datasets ‘cityscapes’ and ‘facades’. The loss function was ‘conditioned cross-entropy loss’ and L1 loss with weights of 1 and 100 respectively.
- Images were generated from the validation set.

Dataset and pre-processing

- We have chosen datasets which ‘cityscapes’ and ‘facades’ which have 2975 and 400 training images respectively. Sample image is shown below.

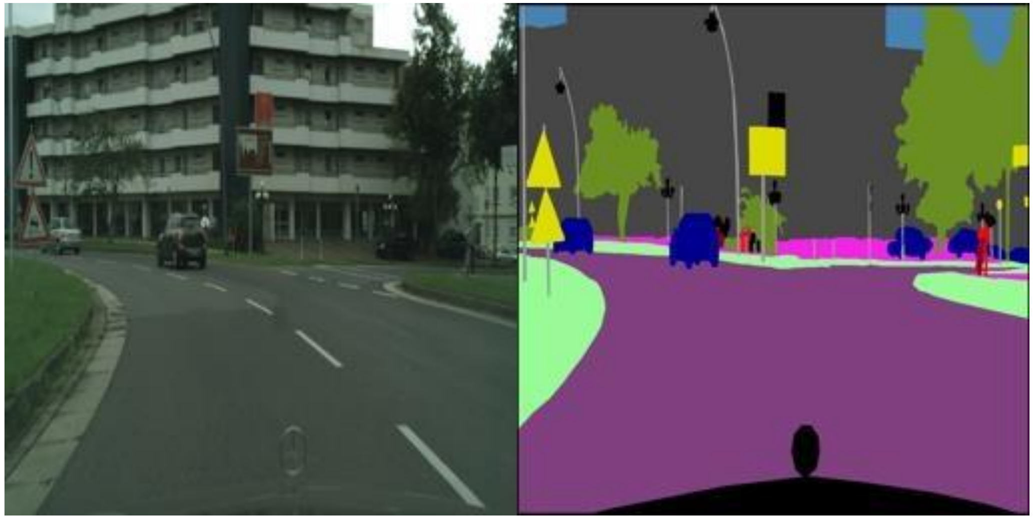
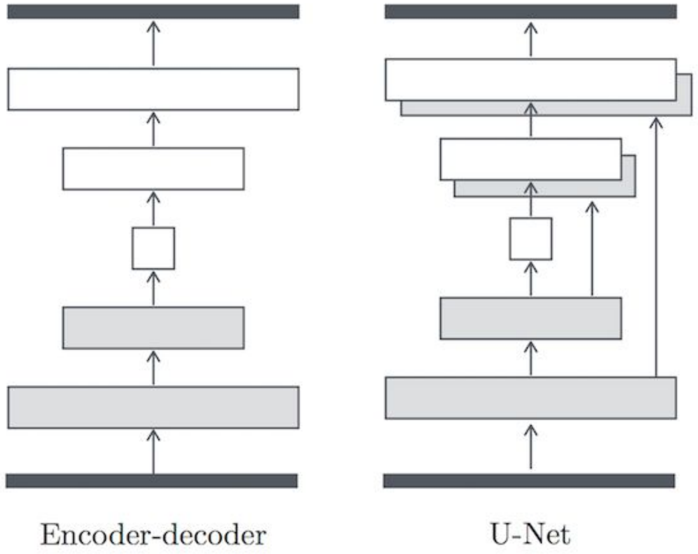


Image size is 512x256. The left half is the ground truth and right half is the input image.

Network Architecture

- U-net architecture was implemented for the generator. This architecture is similar to encoder decoder architecture except for the fact that there are skip connections from the i layer of encoder to n-i layer of the decoder.



Network Architecture

- In the present paper, the following architecture was used:-
- Every encoder layer was down-sampled by a factor of 2 and decoder was upsampled by 2.
- The number of channels in each layer are as follows:-

  3(input) -> C64 -> C128 -> C256 -> C512 -> C512 -> C512 -> C512 -> C512

  All the encoder layers are convolutions with 4x4 filter with stride 2, have leaky ReLU activation function with slope 0.2. Batch-Norm was applied for every layer except C64.

- The U-net decoder is as follows:-

  CD512 -> CD1024 -> CD1024 -> C1024 -> C1024 -> C512 -> C256 -> C128.

  All decoder layers have normal ReLU activations except for output which is tanh. CD in layers refers to dropout of 0.5. Concatenation was used to implement U-net architecture.

Discriminator

Markovian discriminator (PatchGAN) was used which outputs the probability of a patch (16x16) of being real or fake rather the complete image.

4 layer convolution with leaky ReLU activation of slope 0.2 was used.

C64 -> C128 -> C256 -> C512

Batch Norm done on all layers except C64.

Generator generates images to fool the discriminator, while discriminator learns to classify real and fake images. They both improve by competing against each other.

Loss function and Optimizer

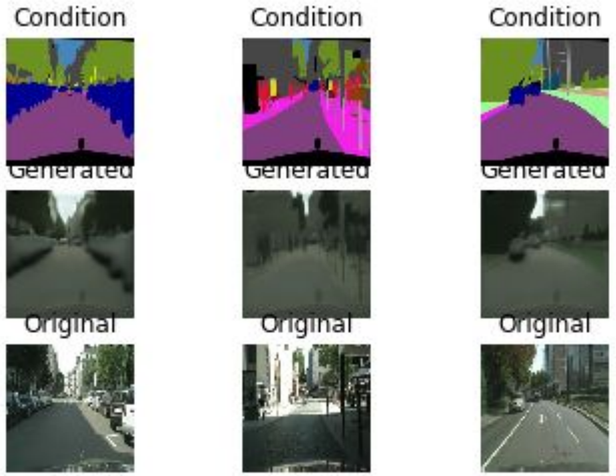
The loss function is shown below

$$\begin{aligned}\mathcal{L}_{cGAN}(G,D) =&\mathbb{E}_{x,y}[\log D(x,y)]+\\ &\mathbb{E}_{x,z}[\log(1-D(x,G(x,z))],\end{aligned}$$

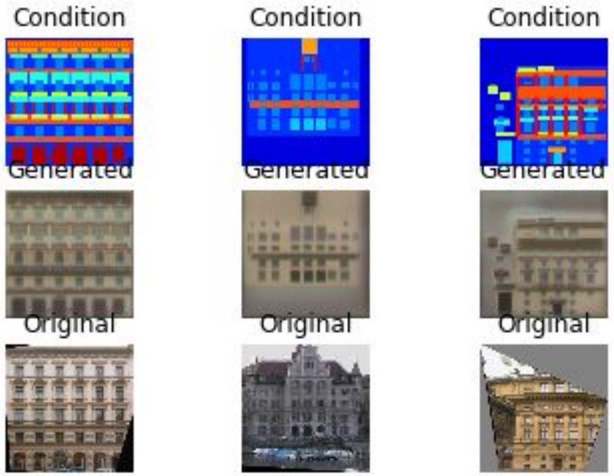
The L1 loss was added to this with weight 100, i.e 100*L1 was added to the above cross-entropy loss function. The idea behind adding L1 is to make the images closer to ground truth while fooling the discriminator.

The optimizer used was Adam with lr = 0.0002, betas = (0.5, 0.999).

Results and discussion



CityScapes



Facades

Conclusions

- Due to low computational power, we were able to run the cityscapes dataset for 80 epochs (the authors have run it for 200 epochs) and facades dataset for 40 epochs. This may be the reason for loss of details in the images.
- The authors have also shown that cGAN performs well for highly detailed images but L1 losses are enough and perform better for images with lower details.

References

- https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/
- https://towardsdatascience.com/pix2pix-869c17900998
- https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09
- Original GAN paper by Ian. J. Goodfellow et al. (https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf)