DeblurGAN

Analysis of DeblurGAN Paper

Intro

DeblurGAN is an end-to-end learned method specifically for motion deblurring. It is based on `conditional GAN (cGAN)` and a multi-component loss function. It uses `Wasserstein GAN (WGAN)` with `gradient penalty` and `perceptual loss`.

The highlighted terms are explained first and then we head to the exact model architecture.

classical GAN and minimax loss function

Generative adversarial network defines a game between two competing networks, a generator and a discriminator. The generator receives noise as input and generates a sample. A discriminator receives a real and generated sample as is trying to distinguish between them. The goal of the generator is to fool the discriminator by generating convincing samples. The generator and discriminator compete with each other and keep getting better. At the end of the training loop, we will have a generator ready for our generation purposes.
The classical GAN uses the minimax loss function.
The generator tries to minimize while the discriminator tries to maximize the given function
$log(D(x)) + log(1 - D(G(z)))$

- D(x) is the discriminator's probability that x (real data) is real
- G(z) is generator's output when given noise z
- D(G(z)) is discriminator's probability that G(z) (fake data) is real is real.

Breaking it down
The generator tries to minimize $log(1 - D(G(z)))$ as it doesn't have control over the other term. Minimizing this is equivalent to maximizing D(G(z)) (probability that fake data is labelled real by discriminator) which is what we want. Meanwhile the discriminator tries to maximize $log(D(x)) + log(1 - D(G(z)))$ which is equivalent to *maximizing D(x)* - making sure real data is classified as real and *minimizing D(G(z))* - making sure fake data (generator's output) is classified as fake. This is the minimax loss.

The original GAN paper notes that the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy. The paper therefore suggests modifying the generator loss so that the generator tries to maximize $logD(G(z))$ instead.

WGAN and Wasserstein loss function

Classical GAN suffers from many problems due to some properties of it's loss function, so it was proposed to use Wasserstein-1 (also known as Earth mover's) distance to design a loss function. In a regular GAN, the loss criterion for generator is probability of generated image being real of fake. In WGAN, the generator tries to minimize the Wasserstein distance.

Wasserstein loss depends on a modification of the GAN scheme where discriminator doesn't actually classify instances (real or fake), but it gives a numerical score which can be from any range (unlike 0 to 1 in previous case). So we can't really draw a line between real and fake here, it just tries to make the output bigger for real instances than for fake instances. It is more of a critic (gives a score) than a discriminator (differentiate between real and fake). The discriminator tries to maximize it while the generator tries to minimize it.
It is given by (D(x)-D(G(z)))

- D(x) critic's output for real instance
- G(z) generator's output on noise z
- D(G(z)) critic's output for fake instance (generator's output).
- These formulas derive from the earthmover distance between the real and generated (fake) distributions.

Critic loss- the discriminator tries to *maximize D(x) - D(G(z))*. Basically it is trying to increase the gap between real and fake inputs.
Generator loss- the generator tries to *minimize D(x) - D(G(z))* which is equivalent to maximizing D(G(z)) which is the only term it can control. Maximizing D(G(z)) is equivalent to reducing the gap between real and fake inputs.

For this loss function to work, the discriminator needs to be a 1-Lipschitz (MA-106💀) function. To ensure this, one method is to add a **gradient penalty** term $\lambda(\|\nabla D(x)\| - 1)^2$. Ignoring the lambda part, it is just pushing the norm of

gradient of D(x) to be closer to one (basically limiting the values of gradient which ensures Lipschitz condition) and λ is a hyperparameter which defines the strength of this penalty.

Conditional GAN

It is an extension of traditional GAN framework that incorporates additional information to guide the generation process. Ex: when we want to generate pictures of a particular dog breed instead of random dogs.
A classic GAN generator learns a mapping from random noise z to y, a cGAN learns a mapping from random noise z and extra data x to y. In our case, we want our Generator to learn the mapping to a sharp image given blurred image and random noise.

Perceptual loss

Also known as feature loss, It is used when comparing two different images that look very similar. Instead of pixel-level differences, perceptual loss uses CNNs to extract meaningful features and the differences are calculated upon these higher features. It is often used in conjunction with other loss functions.

Building the GAN

Notations
I_S - sharp image
I_B - blurred image

θ_G , θ_D - parameters of generator and discriminator
G_{θ_G} , G_{θ_D} - generator and critic

Defining the loss function

$\mathcal{L} = \mathcal{L}_{GAN} + \lambda\mathcal{L}_X$

lambda λ is a hyperparameter
adversarial loss corresponds to the loss of GAN. We use WGAN-GP which corresponds to Wasserstein GAN with gradient penalty as the critic function.
$\mathcal{L}_{GAN} = -\Sigma D_{\theta_G}(G_{\theta_D}(I^B))$
content loss - this component is to preserve the nature of the image, L1 or L2 loss leads to blurry artifacts on generated images, so perceptual loss is used. Perceptual loss is a simple L2 loss based on difference of output of a CNN on generated and target image.
We use VCG-19 conv3.3 neural network which maps the images to a 2 dimensional matrix. Then content loss is simply the average of squares of difference between them.

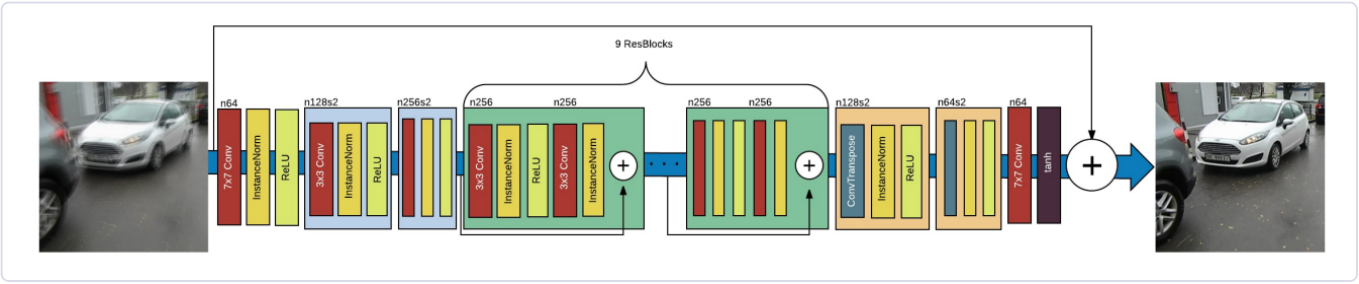
$\mathcal{L}_X = \frac{1}{W.H}\Sigma\Sigma(\phi(I^S) - \phi(I^B))^2$

W and H are dimensions of the image map.

The perceptual loss focuses on restoring general content while adversarial loss focuses on restoring the texture.

Regularization is not recommended according to the paper.

Generator architecture



It has two strided convolutional blocks with stride 1/2 , nine residual blocks (ResBlocks) and two transposed convolutional blocks. Each ResBlock consist of a convolutional layer , instance normalization layer and ReLU activation. Dropout regularization with a probability of 0.5 is added after the first convolutional layer in each ResBlock.In addition, a global skip connection is introduced referred as ResOut. CNN learns a residual connection I_R to a the blurred image I_B , so $I_S = I_R + I_B$

Critic architecture

Critic network is a WGAN with GP (Gradient penalty). It's architecture is similar to PatchGAN. All convolutional layers except the last are followed by InstanceNorm layer and LeakyReLU with α = 0.2

Training

Data used

Three ways of training this model are mentioned

- Training on random 256x256 crops of GoPro dataset downscaled by a factor of 2
- 256x256 patches from MS COCO dataset blurred by artificial method

- On a combination of synthetically blurred images and images taken in high frame-rate camera (2:1 ratio). As the images are fully convolutional and trained on patches, they can be trained on images of arbitrary size.

Optimization step

1. 5 gradient descent steps on D_{θ_D}
2. 1 gradient descent step on G_{θ_G} with adam optimizer
3. Learning rate is set to 1e-4 and linearly decreased to 0 every 150 epochs.
4. Batch size of 1 is used