An Introduction to Generative Adversarial Networks and Image Deblurring Applications

Dimitrios Gotsis

April 27, 2020

Table of Contents

Main Message

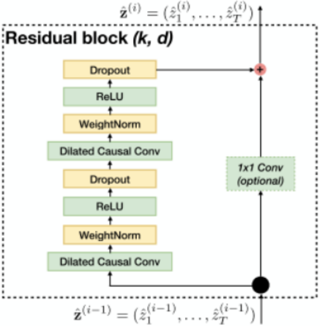
- Brief introduction into neural networks and certain well known layers
- Give an introduction to Generative Adversarial Networks (GANs) and their potential applications
- Introduction to image deblurring problem and potential solutions with GANs and its modifications

Basic Layers

- Will cover some basic layers, many more exist including variations on the ones mentioned
- ReLU (Rectified Linear Unit): activation function, $\mathrm{ReLU}(z) = \max\{0, z\}$, other activation functions exist like sigmoid, tanh, etc.
- Convolutional Layers
- Layers that upsample or downsample, e.g. MaxPooling
- Linear Layer, $y = Wx + b$

Structure of a Neural Network

- Neural Networks usually pass output from one layer to next layer
- More complex structures can be created that do not just ‘stack’, i.e. are not only compositions of functions



Training a Neural Network

- Typically use gradient descent, i.e. $w_{t+1} = w_t - \alpha \nabla f(w_t)$, or some variant to update weights based on objective function
- Need to update weights of all layers, use chain rule to update weights due to composition of functions, i.e. $[f(g(x))]' = f'(g(x))g'(x) = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$
- Start with weights from last layer and move backwards to take advantage of chain rule

Basic Defintions

- A Generative Adversarial Network (GAN) uses two neural networks, a generator (G) and a discriminator (D)
- G and D trained in competition to generate images from noise to look like they derive from training set
- G takes in noise-only images, outputs created images
- D takes in created images and real images and differentiates between them
- When training is complete, discard D and keep G

Mathematical Description

- Have 0 signify created image and 1 signify real image
- Mathematically we have: $D_\phi : \mathbb{R}^{M \times N} \to [0, 1]$ and $G_\theta : \mathbb{R}^{L \times K} \to \mathbb{R}^{M \times N}$
- M, N dimensions of dimensions of created and real images, K, L dimensions of noise-only image, ϕ parameters of D and θ parameters of G

Loss Function

The discriminator's loss function can be shown to be:

$$\mathcal{L}_D = -\frac{1}{2}\mathop{\mathbb{E}}_{x}[\log(D_\phi(x))] - \frac{1}{2}\mathop{\mathbb{E}}_{z}[\log(1 - D_\phi(G_\theta(z)))] \tag{1}$$

To derive the generator loss function we assume zero sum game, i.e. $\mathcal{L}_D + \mathcal{L}_G = 0$ allowing us to say that:

$$\theta^* = \arg\min_{\theta}\max_{\phi}\mathop{\mathbb{E}}_{x}[\log(D_\phi(x))] + \mathop{\mathbb{E}}_{z}[\log(1 - D_\phi(G_\theta(z)))] \tag{2}$$

Training a GAN

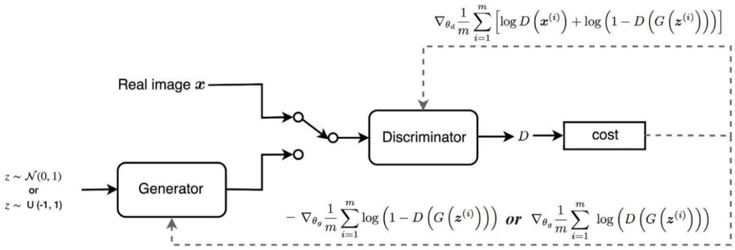


Figure: Training a GAN

Some Applications

- Maps high resolution images to low resolution images well, applications in superresolution and deblurring
- Image-to-image translations, such as mapping sketches to images and aerial photos to maps
- Also used in creating artwork

Application Examples

Figure: Image-to-Image Translation and Superresolution Examples

Mode Collapse

- Problem where G maps noise input z to only one potential output point
- In practice G maps to some output points but not all when mode collapse occurs
- Caused by the fact that $\min\max$ is not the same as $\max\min$ and during training we simultaneously update both networks hoping that solution will be similar
- Simultaneous updates do not privilege $\min\max$ over $\max\min$ or vice versa

Mode Collapse (cont.)

Figure: No Mode Collapse vs Mode Collapse

Vanishing Gradient

- During training G or D may overpower one another potentially causing a vanishing gradient value
- A small gradient means small updates to weights, if this happens too soon in training we get bad weights for both G and D
- Many tricks used to avoid problem such as training one network less often during training or changing the model size of either network until no vanishing gradient is seen

Non-Convergence

- Gradient Descent does not converge for all GANs since it is a minimax problem, a competition between two players G and D
- Sometimes you may get oscillatory results meaning that equilibrium point never reached

Deblurring Background

- Blurry images very common when taking pictures/video, inconvenient since image information lost
- Mathematically can be modeled as $I_B = h * I_S + n$, I_B blurry image, I_S sharp image, h blur kernel, n noise
- For simplicity if we assume no noise, and a known kernel then $I_S = \mathcal{F}^{-1}\left\{\frac{I_B(\omega)}{H(\omega)}\right\}$, where $\mathcal{F}^{-1}$ is inverse Fourier Transform

Deblurring Background (cont.)

- However, even with very small noise may lead to huge distortions from fraction if $H(\omega)$ has small values
- Additionally, h not known in reality and without I_S being known we can have many potential solutions
- Create realistic blur kernels and transform sharp images into blurry images to make dataset

Using GANs for Deblurring

- Main modification: change input generator input from noise-only image to blurry image or blurry image plus noise-only image pair
- Design algorithm for creating realistic blur kernels
- In practice other modifications/additions to loss function also present in other NN solved problems
- See Kupyn et al. (2018)

Conclusion and Summary

- NN made up of function compositions, use gradient descent (or variant) plus chain rule to train weights, more complex NN structures also exist
- GANs can be used for a variety of different applications but difficult to train due to mode collapse, vanishing gradient and potential non-convergence
- Deblurring problem hard to solve due to noise and unknown blur kernel

References

- Goodfellow, Ian. "NIPS 2016 Tutorial: Generative Adversarial Networks." ArXiv:1701.00160 [Cs], December 31, 2016. http://arxiv.org/abs/1701.00160.
- Kupyn, Orest, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks." In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8183–92. Salt Lake City, UT: IEEE, 2018. https://doi.org/10.1109/CVPR.2018.00854.