GANs

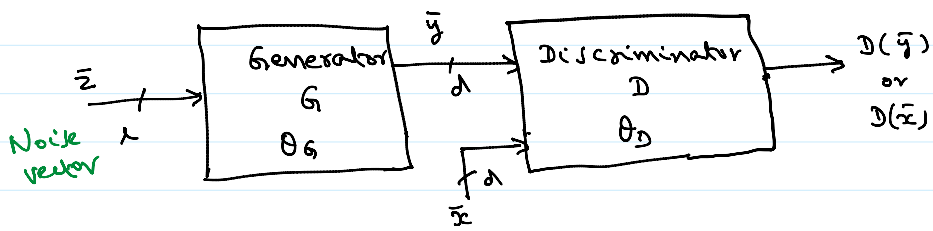
Saturday, 2 May 2020 09:48

Generative Adversarial Networks (GANs)

GAN: Generative model with adversarial learning

Purpose of GAN:

To synthesize examples that are similar to the real examples



Input to GAN: $\bar{z} \sim \mathcal{N}(\bar{0}, \sigma^2 I)$

Synthesized example: $\bar{y} \sim p(\bar{y})$

Real example: $\bar{x} \sim p(\bar{x})$

Objective of GAN: $p(\bar{y}) \approx p(\bar{x})$

Desired output for D:

$D(\bar{x}) = 1$

$D(\bar{y}) = 0$

Training a GAN:

Optimization problem

$$\underset{\theta_D}{\text{Maximize}} \quad \underset{\bar{x} \sim p(\bar{x})}{E}\left[\log D(\bar{x})\right] + \underset{\bar{y} \sim p(\bar{y})}{E}\left[\log(1 - D(\bar{y}))\right]$$

$\downarrow$

$$\underset{\theta_D}{\text{Maximize}} \quad \underbrace{\underset{\bar{x} \sim p(\bar{x})}{E}\left[\log D(\bar{x})\right]}_{\text{Independent of } G} + \underset{\bar{z} \sim \mathcal{N}(\bar{0}, \sigma^2 I)}{E}\left[\log(1 - D(G(\bar{z})))\right]$$

$$\underset{\theta_G}{\text{Minimize}} \quad \underset{\bar{y} \sim p(\bar{y})}{E}\left[\log(1 - D(\bar{y}))\right]$$

$\downarrow$

$$\text{Minimize} \quad E\left[\log(1 - D(G(\bar{z})))\right]$$

$\theta_G \qquad \bar{z} \sim N(\bar{0}, \sigma^2 I)$

Minmax optimization problem

$$\underset{\theta_G}{\text{Minimize}} \quad \underset{\theta_D}{\text{Maximize}} \quad \underset{\bar{x} \sim p(\bar{x})}{E}\left[\log D(\bar{x})\right] + \underset{\bar{z} \sim N(\bar{0}, \sigma^2 I)}{E}\left[\log(1 - D(G(\bar{z})))\right]$$

Discriminator training using gradient-ascent:

$$\theta_D^{New} = \theta_D^{old} + \eta \, \bar{\nabla}_{\theta_D}$$

Generator training using gradient-descent:

$$\theta_G^{New} = \theta_G^{old} - \eta \, \bar{\nabla}_{\theta_G}$$

Deep Convolutional GAN for Image data:

Discriminator : CNN with one output node
- Two-class classifier

Generator : CNN with transposed convolution or CNN with fractional-strided convolution

Deconvolution

$W_1 \times H_1 \qquad W_2 \times H_2$

$F \times F$

S

$$W_2 = \frac{W_1 - F}{S} + 1$$

$S \geq 1 : W_2 < W_1$

$S < 1 : W_2 > W_1$

Effective stride $S < 1$

Transposed Convolution:

Convolutional layer using 2-d kernel

$$W_1 \times H_1 \xrightarrow[K: F \times F]{} W_2 \times H_2$$

$S = 1 \quad P = 0 \qquad W_2 = W_1 - F + 1$

$H_2 = H_1 - F + 1$

$W_1 = H_1 = 4 \qquad W_2 = H_2 = 2$

$F = 3$

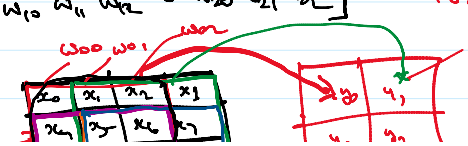
Kernel, K : $\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}_{3 \times 3}$

Convolution matrix : 4×16

$$C = \begin{bmatrix} w_{00} & w_{01} & w_{02} & 0 & w_{10} & w_{11} & w_{12} & 0 & w_{20} & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{00} & w_{01} & w_{02} & 0 & w_{10} & w_{11} & w_{12} & 0 & w_{20} & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{00} & w_{01} & w_{02} & 0 & w_{10} & w_{11} & w_{12} & 0 & w_{20} & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{00} & w_{01} & w_{02} & 0 & w_{10} & w_{11} & w_{12} & 0 & w_{20} & w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}_{16 \times 1}$$

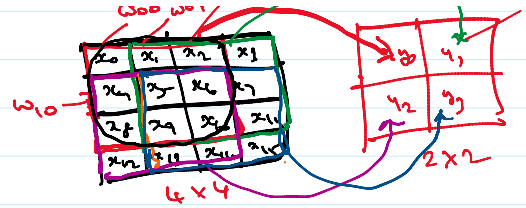
Flattened Input $\bar{x}$: 16×1

for 4×4



Flattened Input $\bar{x}$: 16×1
for 4×4

Flattened output $\bar{y}$ = 4×1
for 2×2



$$\bar{y} = C\,\bar{x}$$
4×1 $\quad 4\times 16 \quad 16\times 1$

w_{00} w_{01} w_{02}
x_4 x_5 x_6
x_8 x_9 x_{10}
x_{12} x_{13} x_{14}

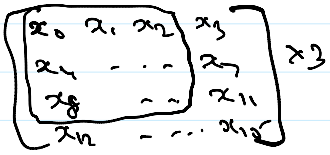
Transposed convolution : $\underset{16\times 1}{\hat{\bar{x}}} = \underset{16\times 4}{C^T}\ \underset{4\times 1}{\bar{y}}$

$\hat{\bar{x}}$ and $\bar{x}$ will not be the same, but are of same size

It can be extended for 3-d kernel :

$W_1\times H_1\times D \xrightarrow[K:\ F\times F\times D]{} W_2\times H_2$

$4\times 4\times 3 \qquad F\times F\times 3 \qquad 2\times 2$



Transposed convolution is also called as fractional-strided convolution
– Effective stride is less than 1

$$W_2 = \frac{W_1 - F}{S} + 1$$

When $W_2 > W_1$, $S < 1$

Another method for fractional-strided convolution

Standard convolution

$W_1\times H_1 \xrightarrow[K:\ F\times F]{} W_2\times H_2$

$4\times 4 \qquad 3\times 3 \qquad 2\times 2$

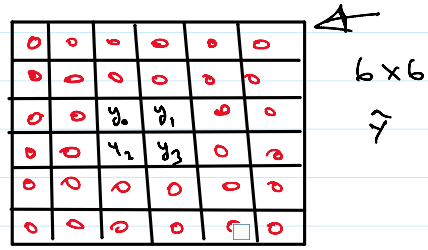
$X \xrightarrow[K]{} Y$

$W_2 < W_1$
$H_2 < H_1$

Fractional-strided convolution

$Y \xrightarrow[K]{} X$

$2\times 2 \qquad 3\times 3 \qquad 4\times 4$
$W_1\times H_1 \qquad W_2\times H_2$



6×6
$\hat{Y}$

Pad Y with 2 rows and 2 columns of zeros to get $\hat{Y}$

$\hat{Y} \xrightarrow[K]{} \hat{X}$

$6\times 6 \qquad 3\times 3 \qquad 4\times 4$
$\hat{W}_1\times\hat{H}_1 \qquad \hat{W}_2\times\hat{H}_2$

Effective stride $\hat{S} < 1$

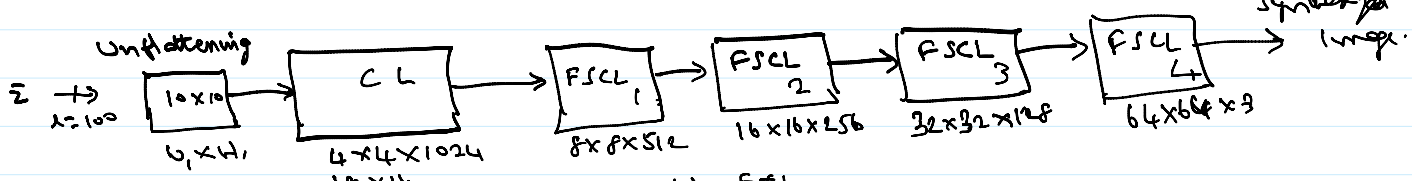
$\hat{W}_1 = 6$
$F = 3$
$\hat{W}_2 = \hat{W}_1 - F + 1$
$4 = 6 - 3 + 1$

3-d kernel fractional-strided convolution

$\hat{W}_1\times\hat{H}_1\times D \xrightarrow[K]{} W_2\times H_2$

$6\times 6\times 3 \qquad 3\times 3\times D \qquad 4\times 4$
$3\times 3\times 3$

Generator in DC GAN for Image Generation



$w_2 = w_1 - F + 1$ $\quad F = 7$

4 $\quad$ 10

Training DCGAN:

Training data set:

N random vectors : $\{\bar{z}_n\}_{n=1}^{N}$ $\quad \bar{z}_n$: $d \times 1$

N images : $\{x_n\}_{n=1}^{N}$

Training examples for Generator: $\{\bar{z}_n, x_n\}_{n=1}^{N}$

Training examples for Discriminator:

$\{x_n, t_n\}_{n=1}^{N} \quad t_n = 1$

$\{\tilde{x}_n, \tilde{t}_n\}_{n=1}^{N} \quad \tilde{t}_n = 0$

$\tilde{x}_n = G(\bar{z}_n)$

For a mini-batch

1. Perform parameter update for discriminator using gradient ascent method
2. Perform parameter updates for generator using gradient descent method.

Conditional GAN:

Use class label as the condition for generation of data.

- Class label information in the form of $\bar{t}_n$, one-hot vector, is concatenated to $\bar{z}$ to get $\hat{z}_n = \text{concat}(\bar{z}_n, \bar{t}_n)$

  $\hat{z}_n$ is given as input to generator to generate examples of a class.

BigGAN : Trained for generation of data of 1000 classes in ImageNet