Reading Report

2019.5.27

1 Deblur GAN

1.1 Optimizer

Algorithm 1 a framework for optimizer

Input: Parameters to be optimized: w; target function $f(w)$; learning rate: α

1: Calculate the gradient of the target function: $g_t = \nabla f(w_t)$
2: Calculation of first and second order momentum according to historical gradient: $m_t = \phi(g_1, g_2, \cdots, g_t); V_t = \psi(g_1, g_2, \cdots, g_t)$
3: Calculate the descending gradient at the current moment $\eta_t = \alpha \cdot m_t / \sqrt{V_t}$
4: Update according to the descending gradient: $w_{t+1} = w_t - \eta_t$

Steps 3 and 4 are consistent for each algorithm, and the main difference is reflected in 1 and 2.

SGD

SGD has no notion of momentum, so
$m_t = g_t; V_t = I^2$
plug in step 3, the descent gradient is the simplest
$\eta_t = \alpha \cdot g_t$ SGD's biggest drawback is the slow rate of decline, and may stay at a local optimum.

SGD with Momentum

In order to suppress SGD oscillation, SGDM believes that inertia can be added in the gradient descent process. When going downhill, if you find it steep, use your inertia to run faster. The full name of SGDM is SGD with momentum, which introduces the first order momentum on the basis of SGD:

$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
In other words, the descending direction at time t is determined not only by the gradient direction of the current point, but also by the accumulated descending direction before. we always use $\beta_1 = 0.9$

SGD with Nesterov Acceleration

Nesterov Accelerated Gradient is a further improvement on SGD and SGD-M, as shown in step 1. Instead of calculating the direction of the gradient of the current position, we calculate the direction of the descent at that time if taking a step with the accumulated momentum.
$g_t = \nabla f(w_t - \alpha \cdot m_{t-1}/\sqrt{V_{t-1}})$
In case of SGD oscillating in the locally optimum. it can watch father.

AdaGrad

Adaptive learning rate. For parameters that are frequently updated, we have accumulated a lot of knowledge about them. We do not want to be influenced too much by a single sample, and we hope to learn slowly. For the occasionally updated parameters, we know too little information, and we hope to learn more from each accidental sample, that is, the learning rate is higher.
$V_t = \sum_{\tau=1}^{t} g_\tau^2$
$\eta_t = \alpha \cdot m_t/\sqrt{V_t}$
Learning rate changed from α to $\alpha/\sqrt{V_t}$.
in order to avoid the denominator is 0, it always add a small smooth term in the denominator. But there are some problems: because $\sqrt{V_t}$ is monotonically increasing, the learning rate will monotonically decrease to 0, which may make the training process end early

AdaDelta / RMSProp

Instead of accumulating all the historical gradients, we focus only on the descending gradient of the window in the past period.
$V_t = \beta_2 * V_{t-1} + (1 - \beta_2) * g_t^2$
It avoids the problem of continuous accumulation of seco.nd order momentum which leads to the early end of the training process

Adam

Adam = Adaptive + Momentum
$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
$V_t = \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot g_t^2$

Nadam

Nadam = Nesterov + Adam
$g_t = \nabla f(w_t - \alpha \cdot m_{t-1}/\sqrt{V_t})$
$V_t = \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot g_t^2$

others

However, ICLR 2018 "On the Convergence of Adam and Beyond", approve in some case Adam can not converge.
$V_t = max(\beta_2 * V_{t-1} + (1 - \beta_2)g_t^2, V_{t-1})$
" The Marginal Value of Adaptive Gradient Methods in Machine Learning "
Through a specific data example, it is shown that the adaptive learning rate algorithm may overfit the features that appear in the early stage, and the features that appear in the later stage are difficult to correct the fitting effect in the early stage
"Improving Generalization Performance by Switching from Adam to SGD"
In the later stage, Adam's learning rate is too low, which affects the effective convergence. so it add a lower bound for learning rate.

content

1.propose a loss and architecture which obtain state-of-the art results in motion deblurring
2.present a method based on random trajectories for generating a dataset for motion deblurring training in an automated fashion from the set of sharp image
3.present a novel dataset and method for evaluation of deblurring algorithms based on how they improve object detection results(loss function)

2019-05-27 下午 2.13.44.png 2019-05-27 下午 2.bb

Algorithm 1 Motion blur kernel generation.

Parameters:
$M = 2000$ – number of iterations,
$L_{max} = 60$ – max length of the movement,
$p_s = 0.001$ – probability of impulsive shake,
I – inertia term, uniform from (0,0.7),
p_b – probability of big shake, uniform from (0,0.2),
p_g – probability of gaussian shake, uniform from (0,0.7),
ϕ – initial angle, uniform from (0,2π),
x – trajectory vector.

procedure BLUR(Img, M, L_{max}, p_s)
  $v_0 \leftarrow cos(\phi) + sin(\phi) * i$
  $v \leftarrow v_o * L_{max}/(M-1)$
  $x = \text{zeros}(M, 1)$
  for $t = 1$ to $M - 1$ **do**
    if randn $< p_b * p_s$ **then**
      nextDir $\leftarrow 2 \cdot v \cdot e^{i*(\pi+(\text{randn}-0.5))}$
    else:
      nextDir $\leftarrow 0$
    $dv \leftarrow$ nextDir $+ p_s * (p_g * (\text{randn} + i * \text{randn}) * I * x[t] * (L_{max}/(M-1))$
    $v \leftarrow v + dv$
    $v \leftarrow (v/abs(v)) * L_{max}/(M-1)$
    $x[t+1] \leftarrow x[t] + v$
  Kernel $\leftarrow$ sub pixel interpolation(x)
  Blurred image $\leftarrow conv$(Kernel, Img)
  return Blurred image

Motion blur kernel generation algorithm

The author adopts the method of random generation of motion trajectory (generated by markov random process). Then "sub-pixel interpolation" is applied to the track to generate blur kernel.

loss function

$$L = L_{GAN} + \lambda L_X$$

Generative adversarial networks(WGAN_GP)+ Generative adversarial networks(perceptual loss)

perceptual loss

it is always used for Style Transfer but the author use it to deblur.

$$l^{SR}_{MSE} = \frac{1}{r^2WH}\Sigma_{x=1}^{rW}\Sigma_{y=1}^{rH}(I^{HR}_{x,y} - G_{\Theta_G}(I^{LR})_{x,y})^2$$

2 VAE

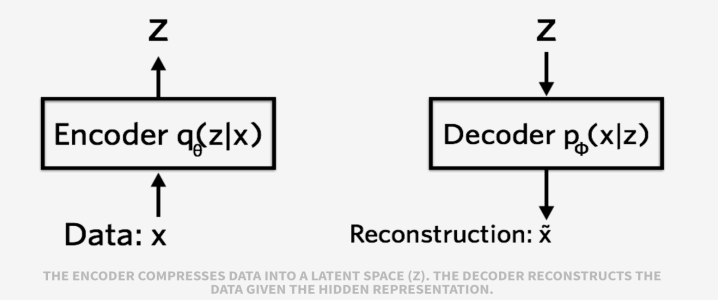
In the whole VAE model, we did not use the assumption that p(Z) (prior distribution) is normally distributed, but we used the assumption that p(Z|X) (posterior distribution) is normally distributed.

In this case, we can let the variational approximate posterior be a multivariate Gaussian with a diagonal covariance structure:

$$\log q_\Phi(z|x^{(i)}) = \log N(z; \mu^{(i)}; \sigma^{2(i)} I)$$

Neural network to fit p(Z|X). So we build two neural networks $\mu^{(k)} = f_1(X^k)$,$log\sigma^2 = f_2(X^k)$ to calculate them. Fitting the $og\sigma^2$ don't need to add the activation function, because it can be positive or negative. so we get p(Z|X) First, we want to reconstruct

2019-05-27 下午 1.00.55.png 2019-05-27 下午 1.bb



X, that is, to minimize $D(X_k, \widetilde{X_k})^2$, but the refactoring process is affected by the noise, because Z_k is through resampling, not directly by the encoder to calculate.

So in order for the final model to be better reconstructed, it's going to do whatever it takes to make the variance equal to 0. in this case, VAE is equal to AutoEncoder.Luckily, we have defined $\log q_\Phi(z|x^{(i)}) = \log N(z; \mu^{(i)}; \sigma^{2(i)} I)$, so p(Z) is also normally distributed. In order to make the model generative, VAE requires that each p(Z|X) should near to the normal distribution. So we add extra loss.

$$L_\mu = \|f_1(X^k)\|^2$$

$$L_{\sigma^2} = \|f_2(X^k)\|^2$$

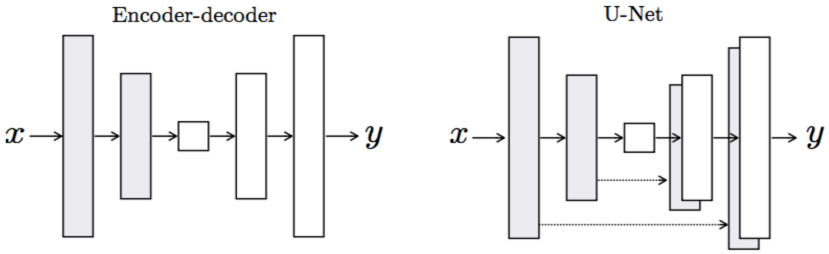
paper directly use $KL(N(\mu, \sigma^2)\|N(0, 1))$ as extra loss function.

3 Pix2Pix GAN

generator

U-net

2019-05-27 下午 1.43.58.png 2019-05-27 下午 1.bb



U-net is like Encoder-Decoder but add i_{th} layer into $(n-i)_{th}$ layer, since they have the same size and can get more information just like Res.

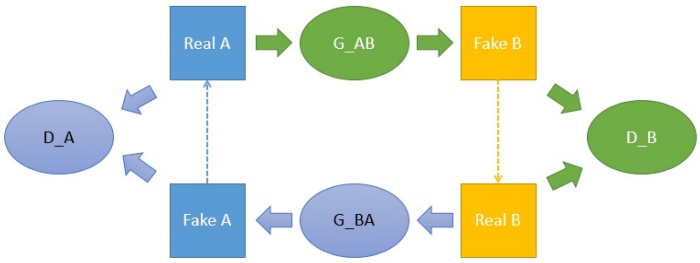
Discriminator

Patch GAN

L_1 and L_2 produces blurry results on image generation problems. Although these losses fail to encourage high-frequency crispness, in many cases they nonetheless accurately capture the low-frequencies. so this paper restrict the GAN discriminator to only model high-frequency structure, relying on an L_1 term to force low-frequency correctness
PatchGAN only penalizes structure at the scale of patches. This discriminator tries to classify if each N ×N patch in an image is real or fake. We run this discriminator convolutionally across the image, averaging all responses to provide the ultimate output of D.
a smaller PatchGAN has fewer parameters, runs faster, and can be applied to arbitrarily large images

4 Cycle GAN

like dual learning
$A_{real} \rightarrow B_{fake} \rightarrow A_fake$
$B_{real} \rightarrow A_{fake} \rightarrow B_fake$



5 some articles from google scholar

DSAL-GAN- Denoising based Saliency Prediction with Generative Adversarial Networks

this paper add SAKLIENCY PREDICTION to GAN, which close to salGAN. It use coupled dual step generative adversarial network. comprises of one network predicting saliency maps, another discriminator network to tell whether the map is a predicted one or ground truth.

The saliency generator adopts an encoder-decoder architecture, where the encoder part includes max pooling layers that decrease the size of the feature maps, while the decoder part uses up-sampling layers. VGG-16 is adopted as encoder part. The objective of discriminator is to fit a deterministic function that predict realistic saliency values from images, rather than realistic images from random noise

Differ from salGAN, this paper use joint optimization of denoising and saliency prediction.(add adversial loss to Loss)

Low-Dose CT Image Denoising Using a Generative Adversarial Network With Wasserstein Distance and Perceptual Loss

advanced image reconstruction from low-dose CT data using WGAN and perceptual similarity

the perceptual loss suppresses noise by comparing the perceptual features of a denoised output against those of the ground truth in an established feature space.

1. use VGG instead of pixel-wise MSE

2.project CT imagines onto a manifold and calculate the geodesic distance since CT imagines are not uniformly distributed in high-dimensional Euclidean space.

Image Blind Denoising With Generative Adversarial Network Based Noise Modeling

1. a GAN is trained to estimate the noise distribution over the input noisy images and to generate noise samples
2.the noise patches sampled from the first step are utilized to construct a paired training dataset, which is used, in turn, to train a deep CNN for denoising the given noisy images.

others

I also derived the formula of WGAN, but there are too many lemmas about measure theory and functional analysis, the formula is too complicated to type out.