Advanced Computer Vision

Практический курс

Wasserstein GAN

Архитектура, инициализация и препроцессинг

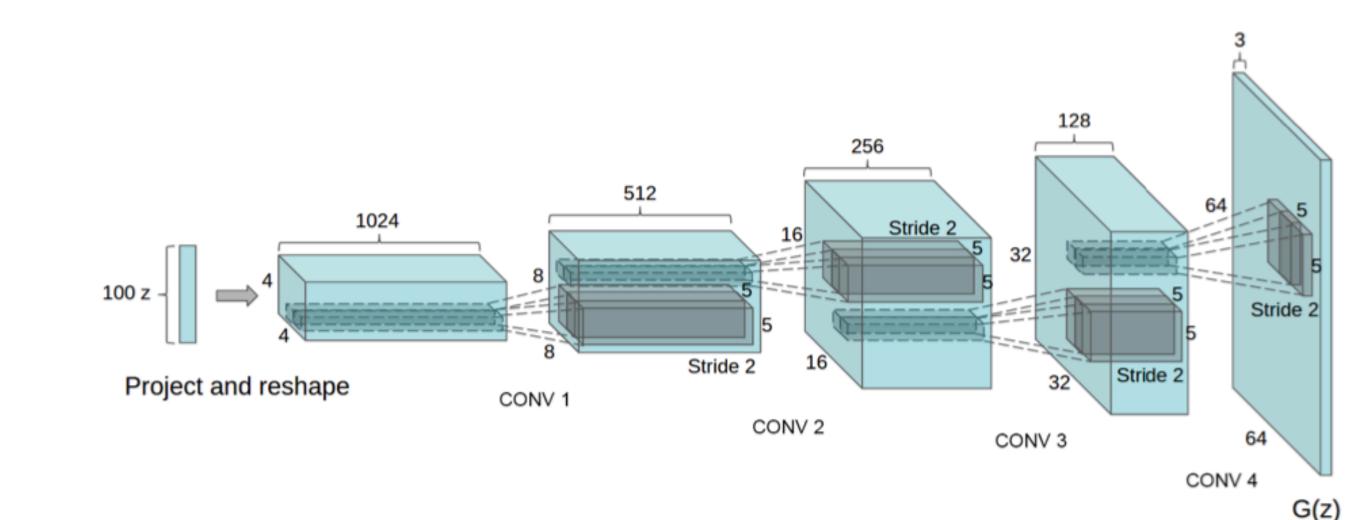
Реализовать нейронную сеть из статьи DCGAN:

https://arxiv.org/pdf/1511.06434.pdf

Здесь можно посмотреть реализацию архитектуры, инициализации весов и препроцессинга (но избавиться от Sigmoid в дискриминаторе!):

https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

P.S. В датасете MNIST картинки 28х28, просто применяем к ним такой же препроцессинг, чтобы они растянулись до 64х64.



Wasserstein GAN

Целевая функция

Статья, на которую опираться при реализации:

https://arxiv.org/pdf/1704.00028.pdf

Гиперпараметры из псевдокода оставить такими же.

Реализация подсчета gradient penalty можно подсмотреть в функции TOGradOfDisc:

https://github.com/jalola/comparetensorflow-pytorch/blob/master/test/ gan_pytorch_model.py , где

interpolates = eps * real + (1 - eps) * fake

```
Algorithm 1 WGAN with gradient penalty. We use default values of \lambda = 10, n_{\text{critic}} = 5, \alpha =
0.0001, \beta_1 = 0, \beta_2 = 0.9.
Require: The gradient penalty coefficient \lambda, the number of critic iterations per generator iteration
      n_{\text{critic}}, the batch size m, Adam hyperparameters \alpha, \beta_1, \beta_2.
Require: initial critic parameters w_0, initial generator parameters \theta_0.
  1: while \theta has not converged do
            for t = 1, ..., n_{\text{critic}} do
                  for i = 1, ..., m do
                       Sample real data x \sim \mathbb{P}_r, latent variable z \sim p(z), a random number \epsilon \sim U[0,1].
                       \tilde{\boldsymbol{x}} \leftarrow G_{\theta}(\boldsymbol{z})
                       \hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}
                       L^{(i)} \leftarrow D_w(\tilde{x}) - D_w(x) + \lambda(\|\nabla_{\hat{x}}D_w(\hat{x})\|_2 - 1)^2
                  end for
                  w \leftarrow \operatorname{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)
            end for
10:
           Sample a batch of latent variables \{z^{(i)}\}_{i=1}^m \sim p(z).
11:
            \theta \leftarrow \operatorname{Adam}(\nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} -D_{w}(G_{\theta}(\boldsymbol{z})), \theta, \alpha, \beta_{1}, \beta_{2})
13: end while
```

```
class DiscriminatorLoss(nn.Module):
   def __init__(self, config):
        super(DiscriminatorLoss, self).__init__()
        self.penalty_lambda = config.penalty_lambda
        self.epsilon = torch.FloatTensor(config.batch_size, 1, 1, 1).to(device)
   def forward(self, netD, fake, disc_fake, real, disc_real):
        loss = disc_fake - disc_real
        self.epsilon.uniform_(0, 1)
        interpolates = self.epsilon * real + (1 - self.epsilon) * fake
       disc_interpolates = netD(interpolates)
        gradients = autograd.grad(outputs=disc_interpolates, inputs=interpolates,
                                  grad_outputs=torch.ones(disc_interpolates.size()).cuda(), create_graph=True)[0]
        gradients = gradients.view(gradients.size(0), -1)
        gradient_norm = gradients.norm(2, dim=1)
        gradient_penalty = ((gradient_norm - 1) ** 2).mean() * self.penalty_lambda
        loss += gradient_penalty
        return loss
```

Wasserstein GAN

Процесс обучения

При обучении советую мониторить значение целевой функции генератора и дискриминатора, а также точность дискриминатора на реальных и сгенерированных картинках.

Обучение лучше начать с генерации только одной буквы из MNIST!

Этот тьюториал полезен для того, чтобы понимать, как процесс обучения будет идти и сколько ждать результатов (архитектуру отсюда не использовать):

https://
machinelearningmastery.com/
practical-guide-to-gan-failuremodes/

Дедлайн 01.12.2022 23:59