Generative models for images

Алеся Иванова

Обучение модели

Я взяла датасет 100-bird-species, который состоит из фотографий 525 видов птиц, всего в датасете 84635 train картинок $224 \times 224 \times 3$, я их сжимала до $64 \times 64 \times 3$.

Пример картинок из датасета:

Использовала модель DCGAN, вдохновлялась PyTorch тьюториалом.

У меня есть следующие параметры при обучении:

- workers = 2 - количество worker-ов в даталодере
- batch_size = 128 - размер батча
- image_size = 64 - размер стороны картинки

- nc = 3 - количество каналов у картинки
- nz = 100 - размер латентного вектора для генерации
- ngf = 64 - размер feature map в генераторе
- ndf = 64 - размер feature map в дискриминаторе
- num_epochs = 1000 - максимальное количество эпох
- lr = 0.0002 - learning rate
- beta1 = 0.5 - параметр оптимайзера Adam

Значения параметров я брала в всё том же тьюториале.

К картинкам я применяла Resize, CenterCrop и Normalize без каких-то дополнительных аугментаций.

Инициализировала веса модели из нормального распределения $N(0, 0.02)$.

Архитектура генератора:

Generator(
(main): Sequential(
(0): ConvTranspose2d(100, 512, kernel$_size = (4,4), stride = (1,1), bias = False)$
(1) : $BatchNorm2d(512, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(2) : $ReLU(inplace = True)$
(3) : $ConvTranspose2d(512, 256, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(4) : $BatchNorm2d(256, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(5) : $ReLU(inplace = True)$
(6) : $ConvTranspose2d(256, 128, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(7) : $BatchNorm2d(128, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(8) : $ReLU(inplace = True)$
(9) : $ConvTranspose2d(128, 64, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(10) : $BatchNorm2d(64, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(11) : $ReLU(inplace = True)$
(12) : $ConvTranspose2d(64, 3, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(13) : $Tanh()$
)
)

Архитектура дискриминатора:

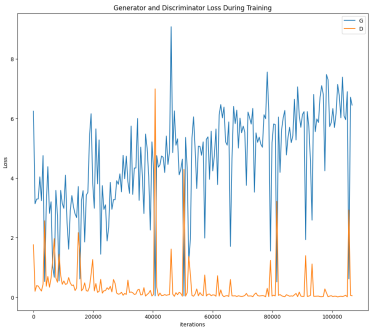
Discriminator(
(main): Sequential(
(0): Conv2d(3, 64, kernel$_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(1) : $LeakyReLU(negative_slope = 0.2, inplace = True)$
(2) : $Conv2d(64, 128, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(3) : $BatchNorm2d(128, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(4) : $LeakyReLU(negative_slope = 0.2, inplace = True)$
(5) : $Conv2d(128, 256, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(6) : $BatchNorm2d(256, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(7) : $LeakyReLU(negative_slope = 0.2, inplace = True)$
(8) : $Conv2d(256, 512, kernel_size = (4,4), stride = (2,2), padding = (1,1), bias = False)$
(9) : $BatchNorm2d(512, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)$
(10) : $LeakyReLU(negative_slope = 0.2, inplace = True)$
(11) : $Conv2d(512, 1, kernel_size = (4,4), stride = (1,1), bias = False)$
(12) : $Sigmoid()$

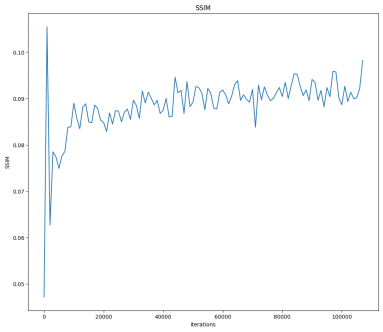
)
)

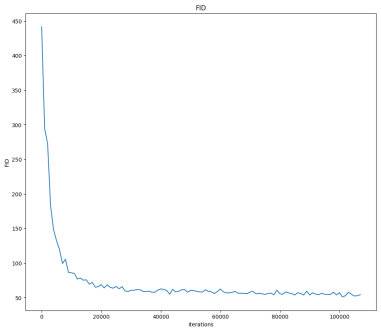
Обучала с critetion BCELoss. Сделала фиксированный вектор fixed_noise и раз в 1000 итераций подавала его на вход генератору для генерации примеров картинок. Так же раз в 1000 итераций я считала SSIM и FID.

Результаты

Моя модель в итоге училась 107000 итераций, итоговое качество генерации следующее:





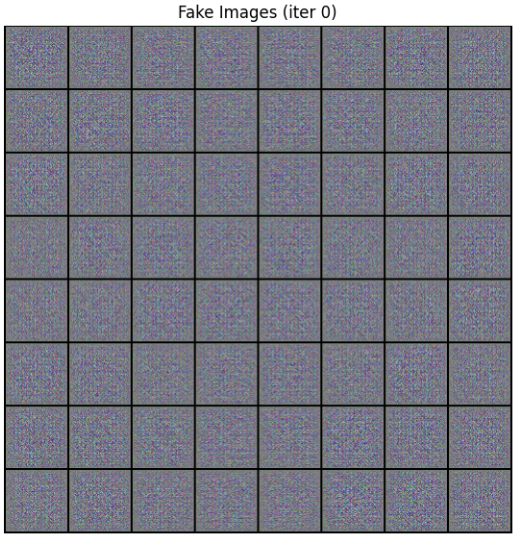


Видно, что FID уменьшается в процессе обучения, то есть распределение реальных картинок постепенно приближется распределением сгенерированных картинок. SSIM остаётся на примерно одном уровне. Мне кажется, это может объясняться тем, что картинки очень разные и сопоставляя случайные пары картинок сложно объективно оценить их сходство.

Можно пронаблюдать улушение качества по мере обучения, это говорит о том, что FID хорошо отражает качество генерации картинок:

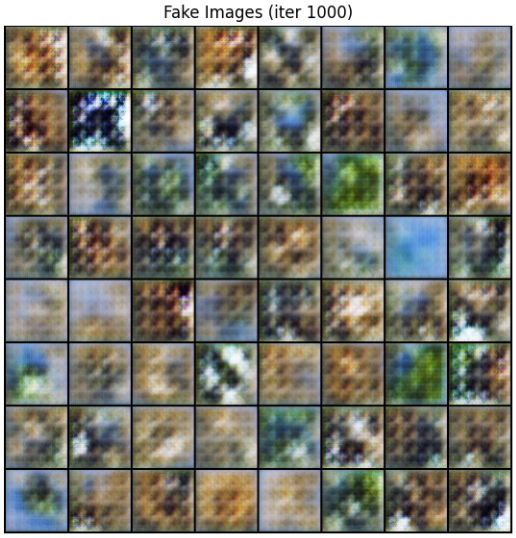
Real Images

Fake Images (iter 0)



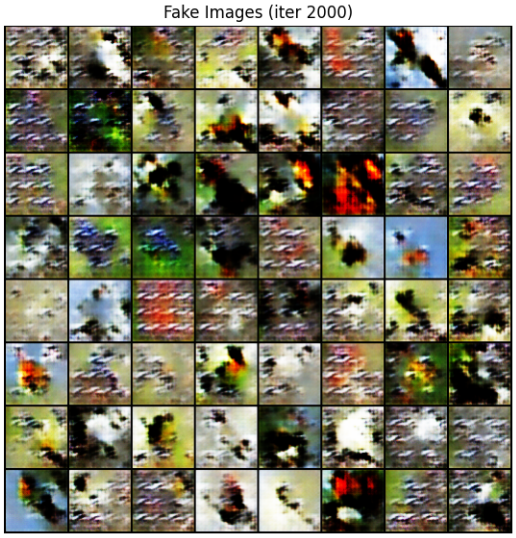
Real Images

Fake Images (iter 1000)



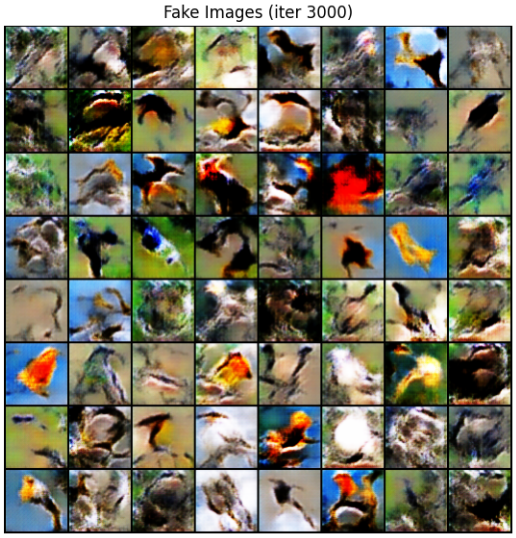
Real Images

Fake Images (iter 2000)



Real Images

Fake Images (iter 3000)

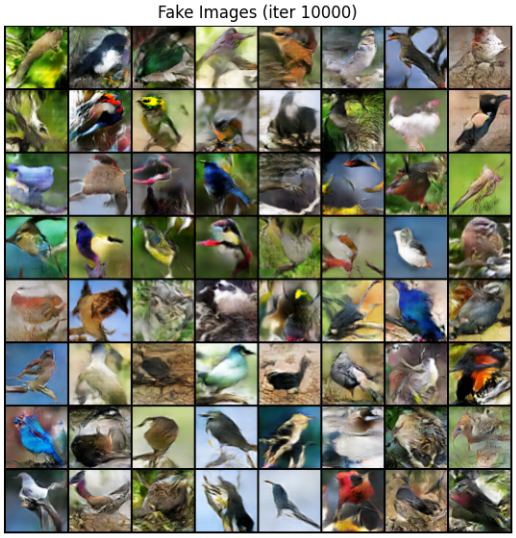


Real Images

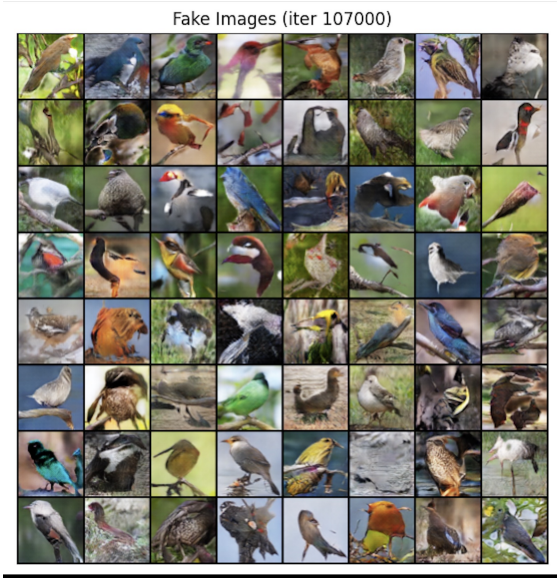
Fake Images (iter 4000)

Real Images

Fake Images (iter 5000)



Давайте посмотрим на каждую сгенерированную картинку повнимательнее и оценим, насколько это птица.



птица спряталась в кустах	синяя птица на прогулке	вообще отличная зеленая птица	странно, но похоже на голову птицы	птицу размазало	супер-птица	strange	сова?
непон	???	птица занимается йогой	?	?	птица в листве	птица на лужайке	голова птицы с краными полосками
пушистая белая птица на ветке	толстая птица на ветке	странно	синяя птица отдыхает	?	какой-то орёл	голова попугая	цапля?
снегирь на дереве	птица на солнечной прогулке	тропическая птица в кустах	голова какой-то птицы?	птица спряталась	почти получилось	чайка?	маленькая пушистенькая птичка
птица гуляет	непон	пингвин в траве?	?	?	?	синяя птица лучше чем в реальности	strange
птица кого-то ждёт на ветке	голова птицы	?	просто вау зелёная птица	не менее крутая коричневая птица	вау, белая птица	непон	?
ну какая красивая блестящая птица	?	колибри	как живая	прелестная жёлтая птица	что-то картинка без птицы	странно	аист?
милая разноцветная птичка	?	крыло птицы	модернизм	постмодернизм	пост-пост	длинная птица сидит отдыхает	голубь спрятал голову в кусты

Как видно, во-первых модель старалась, во-вторых, картинки очень разнообразные, в-третьих, издалека вообще не отличить от птиц, в-четвертых, есть реально крутые сгенерированные птицы!

Качество модели, конечно, не идеальное. Возможно, для такого достаточно большого и разнообразного датасета нужна модель побольше. Так же стоило бы попытаться подобрать опитмальные гиперпараметры, но в силу ограниченных ресурсов это не получилось сделать. Можно было бы попробовать другие модели, такие как StyleGAN или DDPM. Также можно было бы при обучении и генерации учитывать метки классов, так как в датасете для каждой картинки известен вид изображенной на ней птицы. Так бы могла получиться более точная генерация каждого вида птицы, и модели бы было проще понимать, какие картинки считать схожими, то есть на которых изображены одинаковые птицы. Но текущей модели, в принципе, достаточно для генерации птиц, так как она раз в несколько семлов выдаёт реалистичные изображения.