Navigating the GAN Parameter Space for Semantic Image Editing

Введение

Генеративно-состязательные сети (GANs) предназначены для генерации изображений, которые затруднительно отличить от настоящих. Во время обучения сеть состоит из двух частей: генератор G и дискриминатор D. Генератор G по вектору z производит изображение I. Дискриминатор D по изображению I предсказывает вероятность того, что изображение является реалистичным. Во время обучения используется принцип состязания: 1) генератор G пытается обмануть дискриминатор D, то есть старается генерировать такие изображения, которые будут классифицироваться как реалистичные 2) дискриминатор D пытается правильно классифицировать изображения.

Существуют методы, позволяющие по заданному изображению I находить вектор **z**, такой что G(**z**) ~= I. Также известно, что в латентном пространстве (это пространство, в котором находится вектор **z**) существуют такие преобразования, которые отвечают каким-то визуальным преобразованиям картинки. Преобразования могут принимать самый разный вид. Самый простой из них (но который уже является достаточно выразительным) — это сдвиг на заданный вектор, то есть **z** превращается в **z** + epsilon * **z_delta**. К примеру, существуют преобразования, удаляющие фон с картинки, выполняющие вращение головы (для GAN, обученного на изображениях лиц людей), изменяющие толщину написания цифры (для GAN, обученного на MNIST). Примеры таких преобразований можно посмотреть в статье arXiv:2002.03754.

В статье Navigating the GAN Parameter Space for Semantic Image Editing авторы пытаются найти преобразования весов генератора

G, а не латентного пространства **z**. Мотивацией к этому стала работа arXiv:2007.15646, в которой предлагается находить сдвиг весов генератора G, который соответствует заранее заданному преобразованию картинки. Таким образом, этот метод позволяет искать лишь заранее известные преобразования картинки. Авторы же обозреваемой статьи описывают метод, позволяющий находить преобразования в unsupervised режиме (то есть заранее не требуется никакой разметки). Этот метод на вход получает обученный генератор G и на выходе предоставляет набор преобразований весов (количество преобразований является гиперпараметром метода). Преимуществом метода является то, что преобразования заранее не известны, а значит на выходе может получиться много неожиданных картинок.

Сеть-реконструктор

В текущей работе используется подход для поиска различных преобразований, который называется сеть-реконструктор. Он впервые был описан в arXiv:2002.03754. Сеть-реконструктор R — это обучаемая нейросеть, которая на вход принимает два изображения. На выходе эта нейросеть предоставляет два значения: номер преобразования и величина сдвига. То есть, если k — номер преобразования (из [1...K], где K - количество искомых преобразований), epsilon — величина сдвига, то R обучается так, чтобы оно предсказывало R(I, I_shifted) = (k, epsilon).

Направления, получаемые из SVD-разложения

Авторы статьи фиксируют один сверточный слой генератора G и делают SVD-разложение его весов: W = UDV, где D — диагональная матрица. Далее рассматривается преобразования весов, выражаемые как W -> W + UD'V, где D' - диагональная матрица, у которой единица стоит ровно в одном месте, а в

остальных местах стоят нули. Оказывается, что такое преобразование уже позволяет получать осмысленные модификации картинок.

Optimization-based method

Данный метод заключается в том, чтобы искать преобразования в базисе SVD (в базисе, описанном в предыдущем пункте) с помощью сети-реконструктора.

Spectrum-based method

Данный метод использует LPIPS-сеть, которая по двум картинкам выдает число d, соответствующее визуальному расстоянию между ними. Можно d(G_w(**z**), G_{w+shift}(**z**)) разложить в ряд тейлора, где первые два слагаемых зануляются и остается только вторая производная (гессиан). Поэтому наибОльшим изменениями картинки соответствуют сдвиги вдоль собственных векторов гессиана, упорядоченных по убыванию модулей собственных значений.

Hybrid method

Данный метод ищет преобразования в базисе собственных векторов гессиана с помощью сети-реконструктора.

Результаты

Авторы заявляют, что лучше всего себя показал метод Hybrid. В статье есть много картинок интересных преобразований. Авторы показывают, что не все из этих преобразований могут быть получены преобразованием латентного пространства **z**.