Реализация Conditional GAN с помощью Keras

Общие сведения о рекомендованном в литературе принципе передачи номера класса в сеть GAN

Conditional GAN (CGAN) - это тип GAN, который позволяет при обучении учитывать информацию о принадлежности изображения к тому или иному классу, а после обучения целенаправленно генерировать изображения, относящиеся к требуемому классу. Например, при генерации цифр одна цифра является классом и после обучения можно генерировать, например, только восьмерки и тройки.

Соответственно, по сравнению с обычной GAN в CGAN на вход как дискриминатора, так и генератора дополнительно передается номер класса.
Существующие Best practices рекомендуют добавлять информацию о номере класса (как в генератор, так и в дискриминатор) с помощью слоя Embedding.

По определению из документации Keras: слой Embedding превращает положительные целые числа (индексы, они же номера классов) в плотные вектора фиксированного размера. Значения компонент векторов, как правило, меняются в процессе обучения. Фиксированный размер вектора является параметром сети, который выбирается при разработке архитектуры сети и не меняется в процессе обучения (обычно это значение выбирают из диапазона от 32 до 100 (иногда больше)). Номера классов обязательно должны быть положительными числами из фиксированного диапазона.

По сути слой Embedding создает матрицу размерностью "количество классов на фиксированный размер вектора", содержащую отображение каждого класса на плотный вектор. Когда мы передаем в слой Embedding конкретный номер класса, делается срез и оставляется только тот вектор, который соответствует запрошенному классу.

Выход слоя Embedding передается на полносвязный (Dense) слой, имеющий такое количество выходных каналов, которое позволяет масштабировать вектор до размеров входного (для дискриминатора) или генерируемого (для генератора) изображения.

Масштабированный вектор затем объединяется с изображением с помощью метода Concatenate, который добавляет масштабированный вектор как дополнительный канал (карту признаков). В случае черно-белой картинки это будет второй канал (первый - само изображение). Для цветного RGB-изображения это будет четвертый канал (первые три - это соответственно красная, зеленая и синяя составляющие изображения).

После объединения архитектура CGAN не отличается от обычной GAN.

Подробности работы генератора и дискриминатора с номером класса рассмотрим на примере, когда входное изображение имеет размер 32x32 пикселя, а количество классов равно 5.

Добавление информации о номере класса в дискриминатор
На вход дискриминатора передается изображение (матрица 32x32) и номер класса (одно число), к которому относится изображение.
1.1) Входное изображение преобразуется в тензор размерностью (32x32x1).
1.2) Номер класса передается на слой Embedding, который представляет собой матрицу размером 5 (количество классов) на 50 (выбранный на этапе проектирования сети размер вектора, этот размер не обучаемый). На выходе слоя Embedding получим вектор

размерностью 1 на 50, соответствующий номеру класса. За слоем Embedding следует полносвязный слой, который в рассматриваемом случае должен содержать 32x32 = 1024 нейрона. На выходе полносвязного слоя получаем вектор размерностью 1 на 1024. Затем выход полносвязного слоя преобразуется в тензор размерностью (32x32x1).
1.3) После этого изображение и полученный тензор объединяются методом Concatenate в тензор (32x32x2).
1.4) Полученный объединенный тензор передается на конволюционные слои и обрабатываются дальше как в обычной GAN.

Добавление информации о номере класса в генератор
Генератор CGAN позволяет при генерации изображения указать класс, к которому изображение будет относиться.
На вход генератора подается случайный шум (вектор) и номер класса (одно число).
2.1) Пусть на вход генератора подается шум в виде вектора 1x100, где 100 - это выбранный на этапе проектирования сети размер вектора, этот размер не обучаемый.
Дальше этот шум передается на полносвязный слой, содержащий 8x8x128 = 8192 нейрона, где 8x8 - это базовый размер изображения (выбранный на этапе проектирования сети), а 128 - количество карт признаков, которое также выбирается на этапе проектирования сети.
На выходе полносвязного слоя применяется функция активации (например, LeakyReLU), после чего результат преобразуется в тензор размерностью 8x8x128. Это своеобразная "заготовка" будущего изображения.
2.2) Номер класса передается на слой Embedding, который представляет собой матрицу размером 5 (количество классов) на 50 (выбранный на этапе проектирования сети размер вектора, этот размер не обучаемый). На выходе слоя Embedding получим вектор размерностью 1 на 50, соответствующий номеру класса. За слоем Embedding следует полносвязный слой, который в рассматриваемом случае должен содержать 8x8 = 64 нейрона. Здесь 8x8 - это базовый размер изображения, который мы уже использовали на этапе 2.1. Затем выход полносвязного слоя преобразуется в тензор размерностью (8x8x1).
3) После этого тензоры, полученные на этапе 2.1 и 2.2 объединяются методом Concatenate в тензор (8x8x129). Таким образом, тензор, полученный на этапе 2.2, может рассматриваться как еще одна дополнительная карта признаков.
4) Полученный объединенный тензор передается на слои, выполняющие обратную конволюцию, и обрабатывается дальше как в обычной GAN.