Практика

Генеративно-состязательные сети (Generative Adversarial Networks, GAN) помогают моделям генерировать реалистичные данные, такие как изображения. При использовании GAN две нейронные сети — генератор и дискриминатор — обучаются вместе в конкурентной среде, где генератор создаёт синтетические изображения, а дискриминатор учится отличать их от реальных. Такое состязательное обучение со временем улучшает обе сети, что приводит к созданию высококачественных изображений.

Генератор

**Пошаговое объяснение:**

1. **Входной слой (nn.Linear)**:
   * Принимает 100-мерный вектор шума.
   * Преобразует его в тензор размерности 7\*7\*256 = 12544 элементов.
   * Функция активации ReLU добавляет нелинейность.
2. **Преобразование в 3D-форму (nn.Unflatten)**:
   * Изменяет форму тензора из (batch\_size, 12544) в (batch\_size, 256, 7, 7).
   * Это подготовка для сверточных слоев.
3. **Первая транспонированная свертка (ConvTranspose2d)**:
   * Вход: (256, 7, 7), выход: (128, 7, 7).
   * Ядро 5x5, stride=1, padding=2 (сохраняет размер).
   * BatchNorm нормализует активации, ускоряя обучение.
   * ReLU добавляет нелинейность.
4. **Вторая транспонированная свертка**:
   * Вход: (128, 7, 7), выход: (64, 14, 14).
   * Stride=2 увеличивает размер в 2 раза.
   * output\_padding=1 корректирует размер выхода.
5. **Финальная транспонированная свертка**:
   * Вход: (64, 14, 14), выход: (1, 28, 28).
   * Stride=2 увеличивает размер до 28x28.
   * Tanh сжимает значения пикселей в диапазон [-1, 1].

**Как работает генератор:**

* Начинает с случайного шума.
* Постепенно "разворачивает" его в изображение через серию преобразований.
* Каждый слой увеличивает пространственное разрешение и уменьшает глубину (количество каналов).

Дискриминатор

**Пошаговое объяснение:**

1. **Первая свертка (Conv2d)**:
   * Вход: (1, 28, 28), выход: (64, 14, 14).
   * Ядро 5x5, stride=2 уменьшает размер в 2 раза.
   * LeakyReLU (с отрицательным наклоном 0.2) помогает избежать "мертвых" нейронов.
   * BatchNorm стабилизирует обучение.
2. **Вторая свертка**:
   * Вход: (64, 14, 14), выход: (128, 7, 7).
   * Дальнейшее уменьшение размера.
3. **Классификатор**:
   * Flatten() преобразует (128, 7, 7) в вектор (6272).
   * Полносвязный слой Linear сокращает до 1 числа — вероятности "реальности".

**Как работает дискриминатор:**

* Принимает изображение 28x28.
* Последовательно уменьшает его размер, извлекая признаки.
* В конце выдает скаляр (логит), который через сигмоиду превращается в вероятность.