Генеративные модели

Лекция 6: $GANs$

Повторение

Likelihood based models

- Способность генерировать качественные объекты не гарантирует высокого правдоподобия модели

- Высокое правдоподобие не гарантирует, что модель будет генерировать качественные образцы

- Правдоподобие – не идеальная мера качества для генеративных моделей

- Правдоподобие часто может быть невычислимым

- Это заставляет задуматься о новой идее, свободной от правдоподобия – ***идее состязательного обучения***

$Likelihood\ free - learning$

- Мы отказываемся от прямой аппроксимации плотности $p_{data}(\mathbf{x})$

- Вместо этого будем учить функцию генератор $p_{\boldsymbol{\theta}}(\mathbf{x})$ создавать образцы, которые выглядят как будто они взяты и $p_{data}(\mathbf{x})$

У нас будет 2 набора данных:

- ***Реальные образцы*** $(\boldsymbol{real\ samples})$ $\{\mathbf{x}_i\}_{i=1}^{n_1}$ из распределения $p_{data}(\mathbf{x})$

- ***Сгенерированные образцы*** $(\boldsymbol{fake\ samples})$ $\{\mathbf{x}_i\}_{i=1}^{n_2}$ из распределения $p_{\boldsymbol{\theta}}(\mathbf{x})$

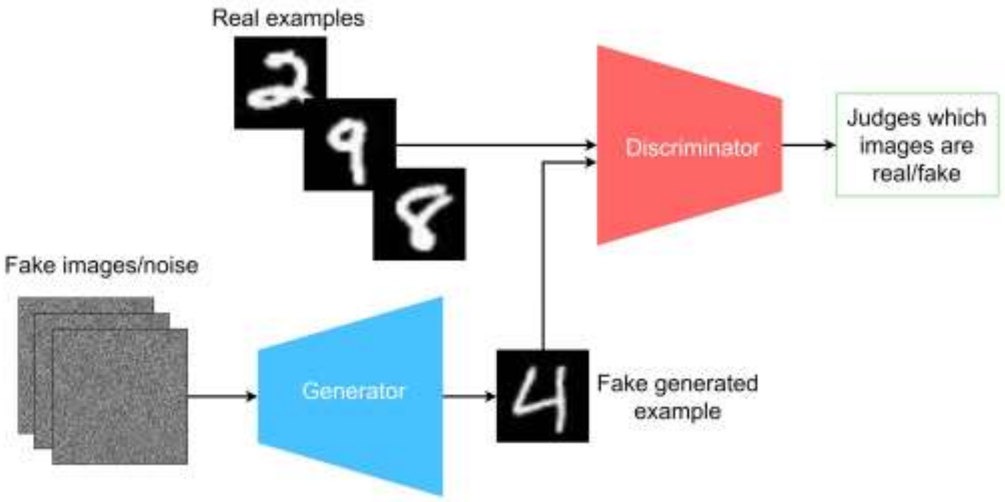
Вводим ***классификатор*** $(\boldsymbol{Discriminator})$, который будет учиться различать эти 2 набора:

- $p(y=1|\mathbf{x})-$ вероятность того, что образец $\mathbf{x}$ является реальным
- $p(y=0|\mathbf{x})-$ вероятность того, что образец $\mathbf{x}$ является сгенерированным

Generative Adversarial Networks

Обучение $GANs$ реализуется через состязание двух нейросетей*:*

- **Генератор G**: создаёт реалистичные данные $\mathbf{x} = G(\boldsymbol{z})$ из вектора $\boldsymbol{z} \sim p(\boldsymbol{z})$
- **Дискриминатор D**: бинарный классификатор, который получает на вход объект $\mathbf{x}$ и выдает вероятность $D(\mathbf{x})$, что $\mathbf{x} -$ реальный



Функция потерь

- Дискриминатор решает задачу бинарной классификации с метками $y = 1$ (реальный) и $y = 0$ (сгенерированный) и выдает вероятность $p(y = 1|\mathbf{x})$

- Хотим обучить D так, чтобы он максимизировал вероятность (правдоподобие) присвоения правильных меток всем классам:

$$\max_{p(y|\mathbf{x})} \left(\mathbb{E}_{p_{data}(\mathbf{x})} \log p(y = 1|\mathbf{x}) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \log p(y = 0|\mathbf{x})\right) =$$

$$= \max_{D} \left(\mathbb{E}_{p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \log\left(1 - D(\mathbf{x})\right)\right)$$

- Генератор, наоборот, пытается минимизировать эту функцию:

$$\min_{G} \max_{D} \left(\mathbb{E}_{p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \log\left(1 - D(\mathbf{x})\right)\right)$$

$$\min_{G} \max_{D} \left(\mathbb{E}_{p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p(\boldsymbol{z})} \log\left(1 - D(G(\boldsymbol{z}))\right)\right)$$

Оптимальный дискриминатор

Найдем оптимальный дискриминатор D^* при любом фиксированном генераторе G

Хотим найти D, который максимизирует $V(G, D)$:

$$V(G, D) = \mathbb{E}_{p_{data}(\mathbf{x})} \log D(\mathbf{x}) + \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})} \log\left(1 - D(\mathbf{x})\right) =$$

$$\int \left[p_{data}(\mathbf{x}) \log D(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x}) \log\left(1 - D(\mathbf{x})\right)\right] d\mathbf{x}$$

$$\frac{dy(D)}{dD} = \frac{p_{data}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{1 - D(\mathbf{x})} = 0$$

$$D^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Генератор

Найдем цель генератора при условии оптимального дискриминатора D^*

Подставим D^* в нашу функцию $V(G, D)$:

$$V(G, D^*) = \int p_{data}(\mathbf{x}) \log D(\mathbf{x})\, d\mathbf{x} + \int p_{\boldsymbol{\theta}}(\mathbf{x}) \log\big(1 - D(\mathbf{x})\big)$$

$$= \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x} + \int p_{\boldsymbol{\theta}}(\mathbf{x}) \log\left(1 - \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}\right)$$

$$= KL\left(p_{data}(\mathbf{x}) \,||\, \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) + KL\left(p_{\boldsymbol{\theta}}(\mathbf{x}) || \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) - 2\log 2$$

$$= 2JSD\big(p_{data}(\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{x})\big) - 2\log 2$$

Оптимальный GAN

Дивергенция Йенсена-Шеннона:

$$JSD\big(p_{data}(\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{x})\big) = \frac{1}{2}\left[KL\left(p_{data}(\mathbf{x}) \,||\, \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) + KL\left(p_{\boldsymbol{\theta}}(\mathbf{x}) || \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right)\right]$$

Истинная цель генератора – минимизация дивергенции Йенсена-Шеннона*:*

$$2JSD\big(p_{data}(\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{x})\big) - 2\log 2 \to \min_{\boldsymbol{\theta}}$$

- $JSD(p||q) = 0$ когда $p = q$
- В этой точке $D^*(\mathbf{x}) = \frac{1}{2}$, $V(G^*, D^*) = -2\log 2$

- Теоретическое равновесие достигается, когда генератор идеально воспроизводит данные, а дискриминатор не может отличить реальные объекты от сгенерированных

Проблемы GAN

Теоретическое равновесие требует, чтобы модели G и D были достаточно мощными (любыми функциями)

В действительности G и D – нейросети с конечным набором параметров

Из-за этой ограниченной мощности настоящее теоретическое равновесие для многих GAN архитектур может не существовать

Это и является основной причиной многих проблем при обучении GAN

Обучение GAN

Теперь наша $min\,max$ задача будет происходить над конкретными параметрами

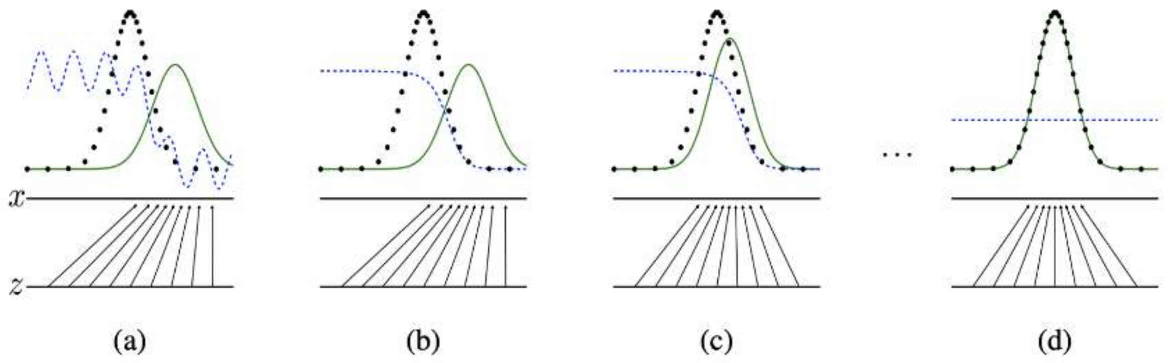
$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \left[\mathbb{E}_{p_{data}(\mathbf{x})} \log D_{\boldsymbol{\phi}}(\mathbf{x}) + \mathbb{E}_{p(\boldsymbol{z})} \log(\mathbf{1} - D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))) \right]$$

Обучаем D:

- Фиксируем веса генератора $\boldsymbol{\theta}$
- Делаем шаг градиентного спуска по $\boldsymbol{\phi}$, чтобы максимизировать $V(G, D)$

Обучаем G:

- Фиксируем веса дискриминатора $\boldsymbol{\phi}$
- Делаем шаг градиентного спуска по $\boldsymbol{\theta}$, чтобы минимизировать $V(G, D)$



План

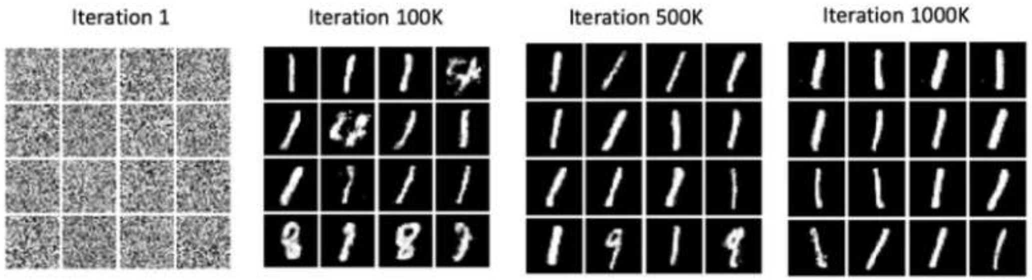
- GAN′s Problems
- Wasserstain Distance
- Wasserstain GANs
- Развитие GANs

GAN's Problems

Mode Collapse

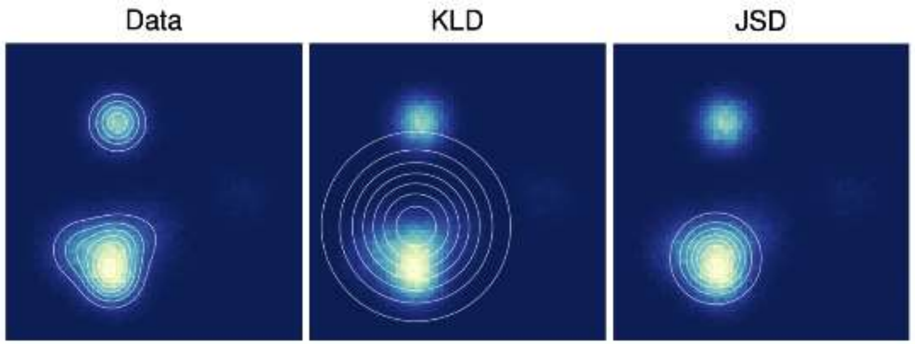
Коллапс мод (***mode collapse***) – главная практическая проблема *GANs*

Генератор «схлопывается» и начинает генерировать только одну или несколько мод распределения



Причина:

- Стандартный *GAN* минимизирует *JSD*, которая склонна к ***поиску мод*** (***mode seeking***)



Генератору проще найти одну моду и идеально ее воспроизводить

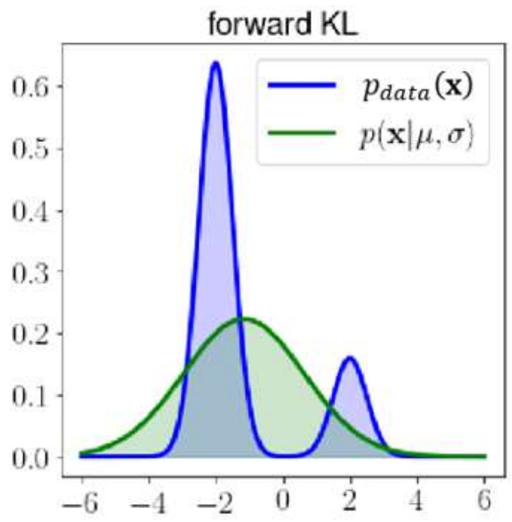
$Jensen - Shannon\ vs\ Kullback - Leibler$

Пусть
- $p_{data}(\mathbf{x})$ – истинное распределение, смесь двух гауссиан
- $p(\mathbf{x} \mid \mu, \sigma) = \mathcal{N}(\mu, \sigma^2)$ – модельное распределение с одной модой

Forward KL (***mode covering***):

$$KL(p_{data} || p_{\boldsymbol{\theta}}) = \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

- Сильно штрафует, если модель $p_{\boldsymbol{\theta}}$ не покрывает область, где есть реальные данные p_{data}
- Чтобы избежать ∞, модель вынуждена покрывать все моды

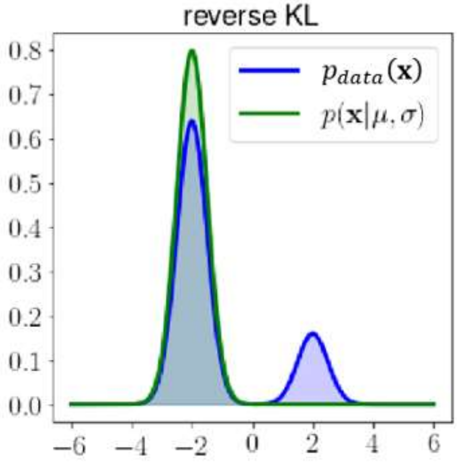


$Jensen - Shannon\ vs\ Kullback - Leibler$

Reverse KL (***mode seeking***):

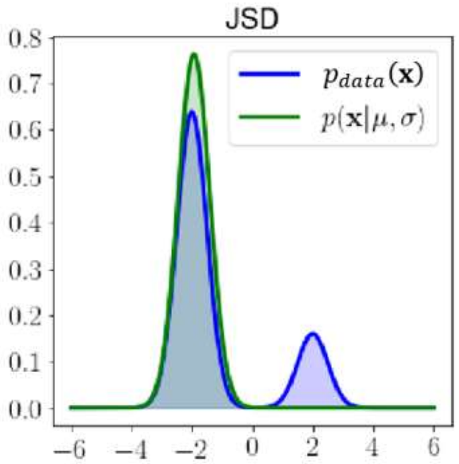
$$KL(p_{\boldsymbol{\theta}}||p_{data}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_{data}(\mathbf{x})}$$

- Сильно штрафует, если модель $p_{\boldsymbol{\theta}}$ генерирует там, где реальных данных p_{data} нет
- Чтобы избежать ∞, модель вынуждена избегать места, где $p_{data}(\mathbf{x}) \to 0$



JSD (***mode seeking***):

- На практике оказывается, что $\mathcal{JSD}$ близка по поведению к ***Reverse KL***



Низкоразмерное многообразие

- В пространствах высокой размерности почти каждая точка – это бессмысленный шум

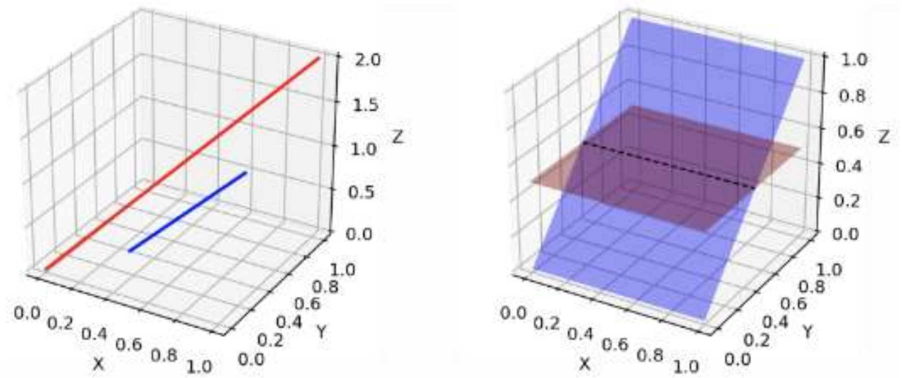
- Многообразие – это небольшая «осмысленная» область внутри этого пространства

- Многие многомерные данные, которые встречаются в реальном мире, на самом деле лежат вдоль низкоразмерных скрытых многообразий внутри этого многомерного пространства ($\boldsymbol{manifold\ hypothesis}$)

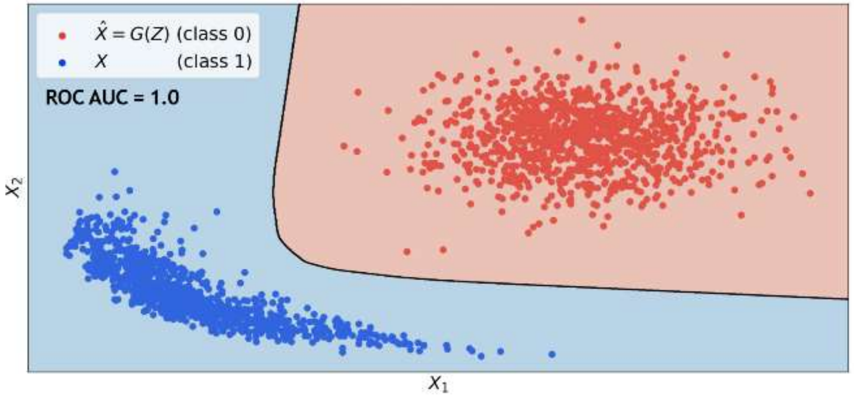
- Сгенерированные изображения также имеют низкую размерность, поскольку создаются из низкоразмерного источника $\boldsymbol{z}$

Низкоразмерное многообразие

- В пространстве высокой размерности эти два многообразия почти гарантированно не пересекаются (***непересекающиеся носители, disjoint supports***)



- Поскольку эти многообразия не пересекаются, то дискриминатору очень легко найти разделяющую плоскость и стать идеальным классификатором



Низкоразмерное многообразие

Forward KL:

- В случае непересекающихся носителей в области, где $p_{data}(\mathbf{x}) > 0$, мы имеем $p_{\boldsymbol{\theta}}(\mathbf{x}) = 0$:

$$KL(p_{data}||p_{\boldsymbol{\theta}}) = \int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x} \to \infty$$

Reverse KL:

- В случае непересекающихся носителей в области, где $p_{\boldsymbol{\theta}}(\mathbf{x}) > 0$, мы имеем $p_{data}(\mathbf{x}) = 0$:

$$KL(p_{\boldsymbol{\theta}}||p_{data}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_{data}(\mathbf{x})} d\mathbf{x} \to \infty$$

Низкоразмерное многообразие

$\boldsymbol{Jensen - Shannon}$:

$$JSD(p_{data} \,||\, p_{\boldsymbol{\theta}}) = \frac{1}{2} KL\left(p_{data}(\mathbf{x}) \,||\, \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) + \frac{1}{2} KL\left(p_{\boldsymbol{\theta}}(\mathbf{x}) || \frac{p_{data}(\mathbf{x}) + p_{\boldsymbol{\theta}}(\mathbf{x})}{2}\right) =$$

$$\frac{1}{2}\int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_m(\mathbf{x})} d\mathbf{x} + \frac{1}{2}\int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_m(\mathbf{x})} d\mathbf{x} = \frac{1}{2}\int p_{data}(\mathbf{x}) \log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})/2} d\mathbf{x} +$$

$$+ \frac{1}{2}\int p_{\boldsymbol{\theta}}(\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})/2} d\mathbf{x} = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = \log 2$$

- Как только дискриминатор становиться идеальным, JSD дивергенция становится константой
- Обучение останавливается, поскольку градиенты становятся равными нулю

Wasserstain Distance

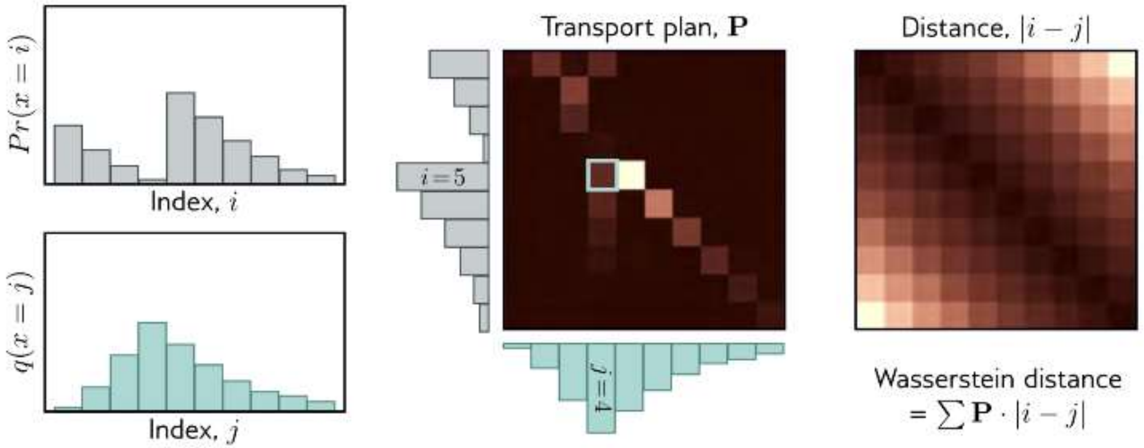
Wasserstain Distance

Интуиция:

- Мы можем представить два распределения как две кучи земли

- Стоимость перемещения рассчитывается следующим образом*:*

$$cost = amount \times distance$$



Расстоянием Вассерштейна (***Wasserstain Distance, Earth Mover's Distance***) будем называть самый эффективный транспортный план P, который минимизирует общую стоимость

Wasserstain Distance

В непрерывном случае расстояние Вассерштейна определяется следующим образом:

$$W(\pi, p) = \inf_{\gamma \in \Gamma(\pi,p)} \mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\gamma}||\mathbf{x} - \boldsymbol{y}|| = \inf_{\gamma \in \Gamma(\pi,p)} \int ||\mathbf{x} - \boldsymbol{y}||\,\gamma(\mathbf{x}, \boldsymbol{y})d\mathbf{x}d\boldsymbol{y}$$

- γ – совместное распределение, показывающее сколько массы нужно перенести из $\mathbf{x}$ в $\boldsymbol{y}$, чтобы p превратилось в π (***транспортный план***
- множество всех возможных транспортных планов γ
- $\mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\gamma}||\mathbf{x} - \boldsymbol{y}||$ – ожидаемая стоимость перемещения массы из точки $\mathbf{x}$ в точку $\boldsymbol{y}$ согласно плану γ

Существует целое семейство ***метрик Вассерштейна*** $W_s(\pi, p)$:

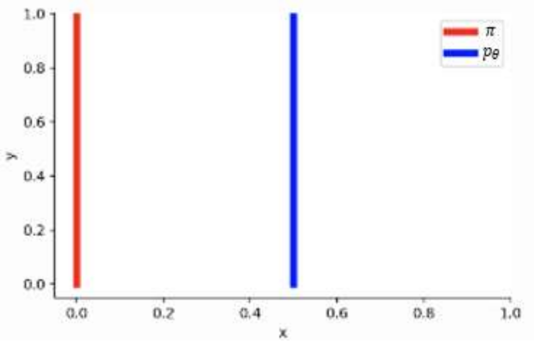
$$W_s(\pi, p) = \inf_{\gamma \in \Gamma(\pi,p)} \left(\mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\gamma}||\mathbf{x} - \boldsymbol{y}||^s\right)^{\frac{1}{s}}$$

Мы будем использовать метрику $W_1(\pi, p)$, то есть когда $s = 1$

$Wasserstain\ Distance\ vs\ KL\ vs\ JSD$

Рассмотрим 2 распределения π и p_{θ}:

$$\pi(x,y) = (0, U[0,1])$$
$$p_{\theta}(x,y) = (\theta, U[0,1])$$



При $\theta = 0$ распределения совпадают:

$$KL(\pi||p_{\boldsymbol{\theta}}) = KL(p_{\boldsymbol{\theta}}||\pi) = JSD(\pi||p_{\boldsymbol{\theta}}) = W(\pi, p_{\boldsymbol{\theta}}) = 0$$

При $\theta \neq 0$:

$$KL(\pi||p_{\boldsymbol{\theta}}) = \int_{U[0,1]} 1 \log\frac{1}{0} dy = \infty$$

$$JSD(\pi||p_{\boldsymbol{\theta}}) = \frac{1}{2}\left(\int_{U[0,1]} 1 \log\frac{1}{1/2} dy + \int_{U[0,1]} 1 \log\frac{1}{1/2} dy\right) = \log 2$$

$$W(\pi, p_{\boldsymbol{\theta}}) = |\theta|$$

Wasserstain Distance vs KL vs JSD

Вывод:

- Расстояние Вассерштейна – непрерывная и дифференцируемая функция, даже если носители не пересекаются (***теорема* 1**)
- Оно всегда даёт осмысленный градиент, который генератор использует во время обучения

***Теорема* 2**:

- Сходимость по W более слабое условие, чем сходимость по JSD или KL
- Даже если обучение не сходится по JSD, она может сойтись по по W, что делает обучение более стабильным

$Wasserstain\ GAN$

Теорема Канторовича-Рубинштейна

Проблема:

- Вычислить W напрямую через inf невозможно:

$$W(\pi, p) = \inf_{\gamma \in \Gamma(\pi, p)} \mathbb{E}_{(\mathbf{x}, \boldsymbol{y}) \sim \gamma} ||\mathbf{x} - \boldsymbol{y}||$$

Будем использовать ***теорему Канторовича−Рубинштейна***

$$W(\pi, p) = \frac{1}{K} \max_{||f||_L \leq K} [\mathbb{E}_{\pi(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})} f(\mathbf{x})]$$

- Этот трюк работает только для функций f, которые являются K −Липшицевыми:

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq K|\mathbf{x}_1 - \mathbf{x}_2|$$

- Это означает, что крутизна (градиент) функции f ограничена константой K

Wasserstain GAN

Мы не можем найти идеальную f, поэтому будем аппроксимировать ее с помощью нейросети критика $f_{\boldsymbol{\phi}}$ (***Critic***)

Обеспечение Липшицевости:

- Будем принудительно обрезать (***clamped***) все веса в диапазон $[-c, c]$ после каждого шага градиентного спуска
- Это гарантирует, что функция будет K –Липшицевой

Множество наших нейросетей $f_{\boldsymbol{\phi}}$ – это подмножество всех K –Липшицевых функций f:

$$K \cdot W(\pi, p) = \max_{||f||_L \leq K} \left[\mathbb{E}_{\pi(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})} f(\mathbf{x})\right] \geq \max_{\boldsymbol{\phi} \in \boldsymbol{\Phi}} \left[\mathbb{E}_{\pi(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{p(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x})\right]$$

Wasserstain GAN

Функция потерь ***GAN***:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \left(\mathbb{E}_{p_{data}(\mathbf{x})} \log D_{\boldsymbol{\phi}}(\mathbf{x}) + \mathbb{E}_{p(\boldsymbol{z})} \log\left(1 - D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))\right)\right)$$

Функция потерь ***WGAN***:

$$\min_{\boldsymbol{\theta}} W(\pi, p) \approx \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi} \in \boldsymbol{\Phi}} \left[\mathbb{E}_{\pi(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{p(\boldsymbol{z})} f_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))\right]$$

Основные отличия:

- Дискриминатор $D_{\boldsymbol{\phi}}$ → Критик $f_{\boldsymbol{\phi}}$ выдает любое число (оценку)
- Параметры критика $\boldsymbol{\phi}$ ограничены

$WGAN - GP$

$\boldsymbol{Weight\,Clipping}$ – очень плохой способ заставить критика быть K –Липшицевым

Идея $\boldsymbol{WGAN - GP}$:

Вместо того, чтобы обрезать веса $\boldsymbol{\phi}$, будем наказывать критика, если он нарушает 1 –Липшицево ограничение

1 –Липшицевость означает, что норма градиента критика $||\nabla f_{\phi}(\mathbf{x})|| \leq 1$ ***в любой точке***

- Мы не можем проверять градиент везде
- Авторы доказали, что достаточно проверить его на случайных точках между реальными x_{data} и поддельными x_{fake} данными

$WGAN - GP$

Gradient Penalty:

- Создаем случайный объект с помощью интерполяции:
$$\hat{\mathbf{x}} = \epsilon \cdot \mathbf{x}_{data} + (1 - \epsilon) \cdot \mathbf{x}_{fake}$$

- Вычисляем градиент критика $f_{\boldsymbol{\phi}}$ в этой точке $\hat{\mathbf{x}}$ и штрафуем его если $||\nabla f_{\boldsymbol{\phi}}(\hat{\mathbf{x}})||_2 \geq 1$

Функция потерь $\boldsymbol{WGAN - GP}$:

$$\mathcal{L} = \underbrace{\left[\mathbb{E}_{\pi(\mathbf{x})} f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{p(\boldsymbol{z})} f_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(\boldsymbol{z}))\right]}_{WGAN\ Loss} + \underbrace{\lambda \cdot \mathbb{E}_{\hat{\mathbf{x}}}\left[\left(||\nabla f_{\boldsymbol{\phi}}(\hat{\mathbf{x}})||_2 - 1\right)^2\right]}_{Gradient\ Penalty}$$

Развитие $GANs$

$GAN's\ Problems$

$\boldsymbol{WGAN}$ дал нам стабильные градиенты, но создание HR-изображений (1024×1024) все равно сложная задача

***Проблема* 1**:

- Чем выше разрешение, тем больше пространство, в котором живут объекты
- В таком пространстве наши два многообразия почти гарантировано не пересекаются

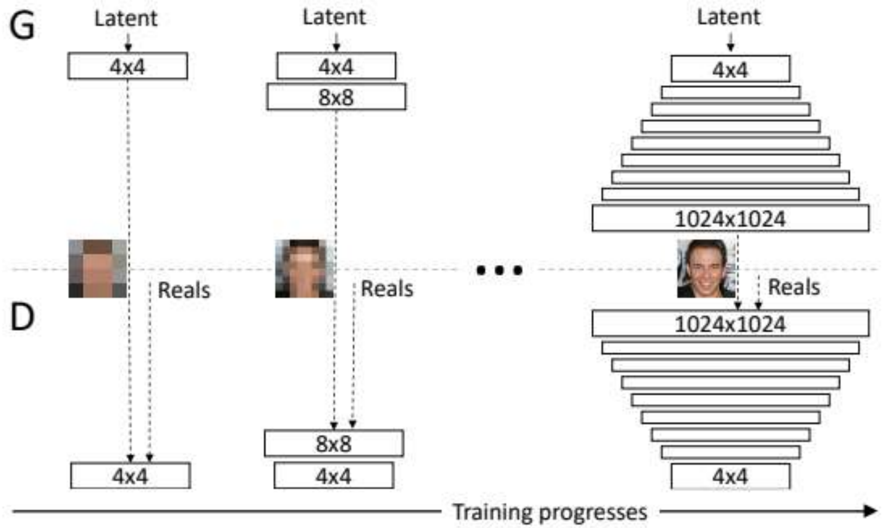
***Проблема* 2**:

- Генерация HR-изображений требует огромных моделей, которые занимают всю память
- Мы вынуждены использовать небольшие батчи, обучение на которых даёт шумные и нестабильные градиенты

Progressive Growing GAN

Идея: Будем постепенно увеличивать разрешение генерируемого изображения

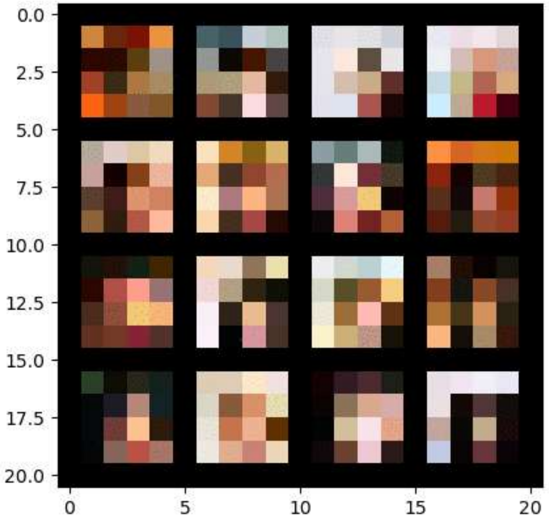
- Обучаем модель генерировать мелкие изображения 4×4
- Добавляем новые слои к обеим сетям (***Upsampling***)
- Обучаем новые слои, пока старые заморожены
- Повторяем процесс $8 \times 8 \rightarrow 16 \times 16 \ldots \rightarrow 1024 \times 1024$



Это хорошо стабилизирует обучение:

Модель сначала изучает общую структуру (поза, форма лица) на низких, а затем добавляет мелкие детали на высоких разрешениях

Progressive Growing GAN



$GAN's\ Problems$

- $\boldsymbol{PG - GAN}$ позволил генерировать изображения в высоком разрешении, но они все еще использовали свёрточные сети
- Сверточные сети по своей природе локальны

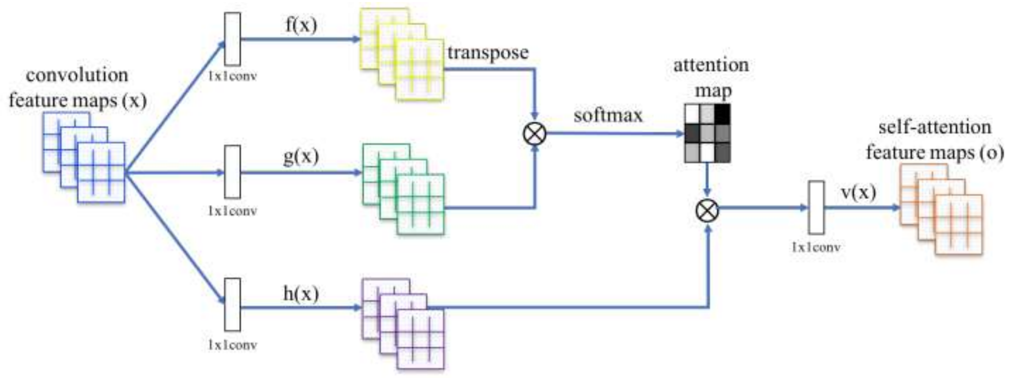
Проблемы:

- Требуются глубокие сети, чтобы моделировать дальние зависимости
- Через много слоев информация о каких-то признаках забывается

$Self - Attention\ GAN$

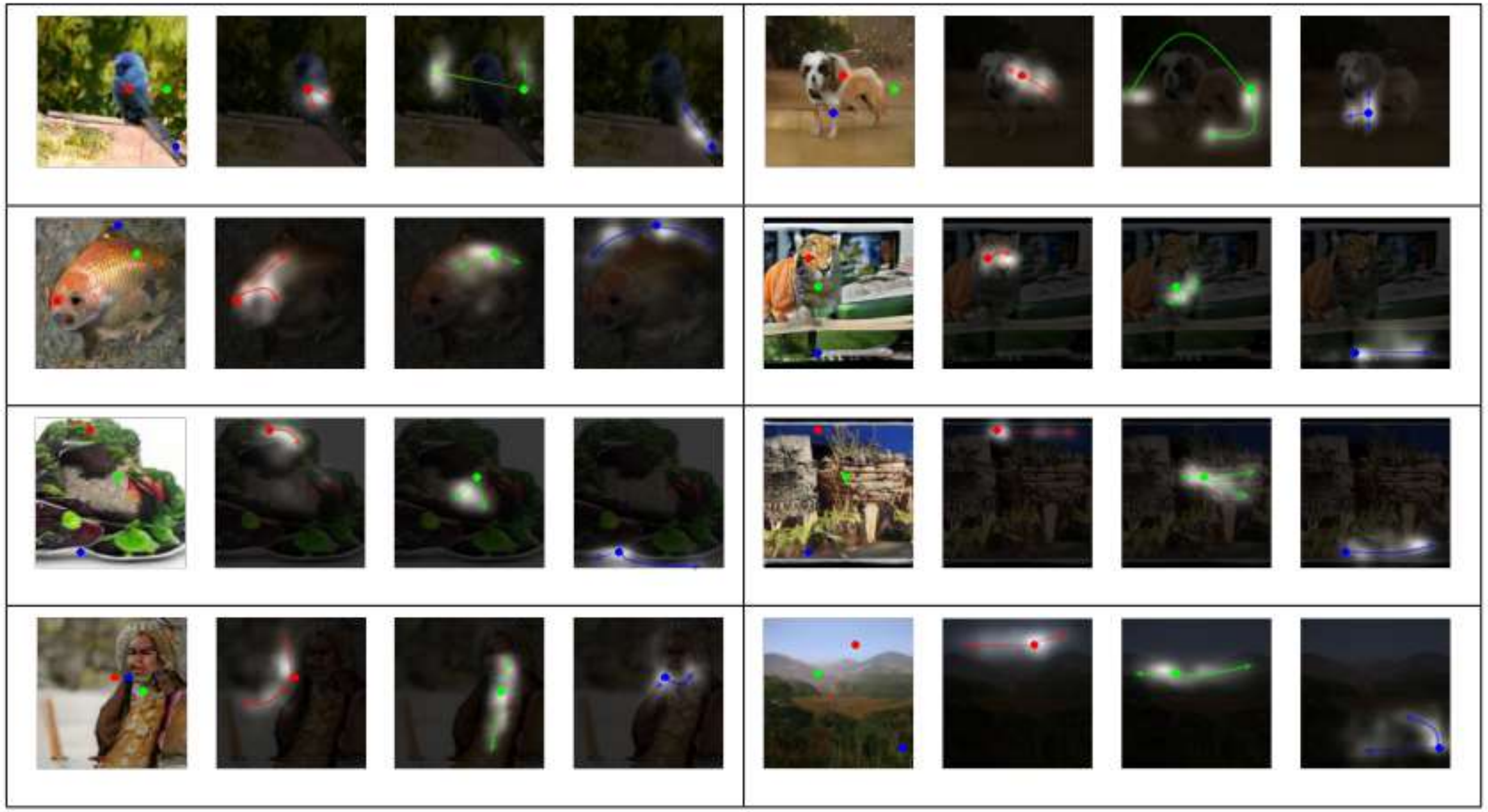
Идея:

- Дадим возможность пикселям смотреть друг на друга ($\boldsymbol{self - attention}$)



- Используется $\boldsymbol{Hinge\ Loss,}$ который тоже решает проблемы затухания градиентов JSD
- Липшицевость критика достигается с помощью спектральной нормализации ($\boldsymbol{Spectral\ Normalization}$)
- Дискриминатор и генератор обучаются с разной скоростью ($\boldsymbol{Separate\ Learning\ Rate}$)

$Self - Attention\ GAN$



$StyleGAN$

Идея:
- Возьмем за основу $PG - GAN$ и научим модель делать контролируемую генерацию

Проблема:

- В обычных $GANs$ латентный вектор $\boldsymbol{z}$ – это черный ящик

- Хотим сделать распутанное пространство, где мы смогли бы контролировать конкретные черты изображения

В $StyleGAN$ предложили разделить черты на 3 уровня:

- ***Грубые*** (***Coarse***) $(4 \times 4, 8 \times 8)$ – поза, форма лица

- ***Средние*** (***Middle***) $(16 \times 16, 32 \times 32)$ – черты лица, детали прически

- ***Мелкие*** (***Fine***) $(64 \times 64 \to 1024 \times 1024)$ – микро-детали, цвет глаз, морщины

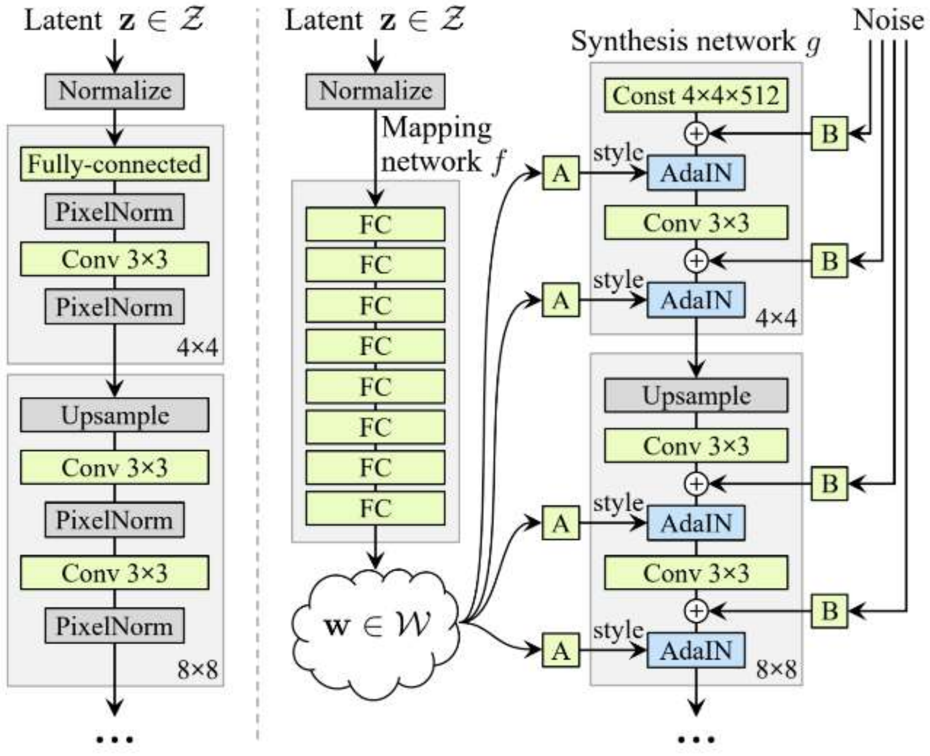
StyleGAN

Обычный GAN:

- Латентный вектор $\boldsymbol{z}$ подается на вход первому слою и проходит через всю сеть перемешиваясь

Style – based GAN:

- Генератор разделён на 2 сети
- ***Mapping Network*** – 8 слойный *MLP*, который распутывает $\boldsymbol{z}$ и превращает его в стилевой вектор $\boldsymbol{w}$
- ***Synthesis Network*** – сверточная сеть, которая рисует изображение, используя $\boldsymbol{w}$ как инструкцию

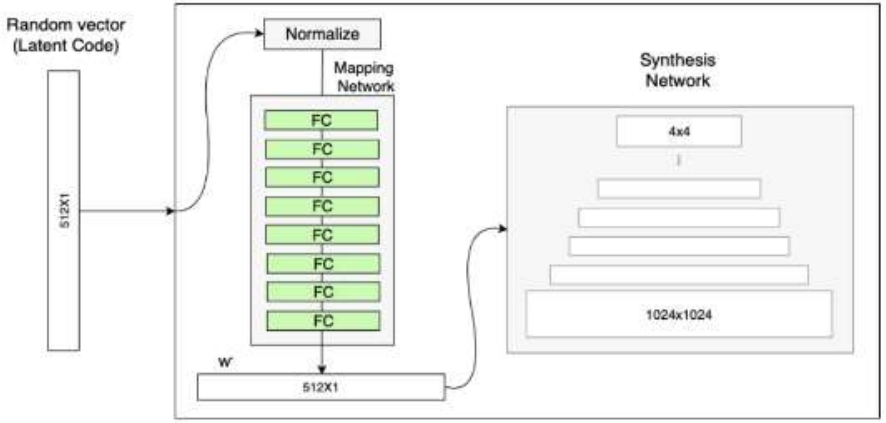


(a) Traditional (b) Style-based generator

StyleGAN: *Mapping Network*

Цель:
- Хотим, чтобы каждая компонента вектора $\boldsymbol{z}$ отвечала за некоторую черту

- В реальности $\boldsymbol{z}$ следует обучающим данным
- Будем использовать ***Mapping Network***, которая будет распутывать $\boldsymbol{z}$ и превращать его в стилевой вектор $\boldsymbol{w}$
- ***Synthesis Network*** будет требовать от ***Mapping Network*** понятных инструкций



(a) Distribution of features in training set

(b) Mapping from $\mathcal{Z}$ to features

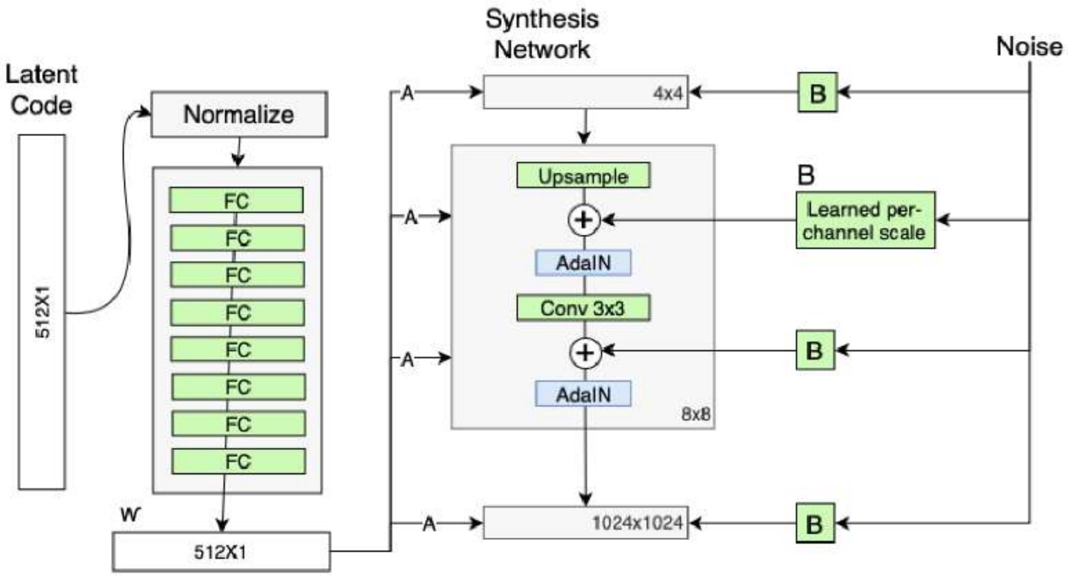
(c) Mapping from $\mathcal{W}$ to features

StyleGAN: *Stohastic Variation*

Вектор ***w*** контролирует все главные черты, но мы хотим сделать лицо более реалистичным со случайными деталями

Решение:

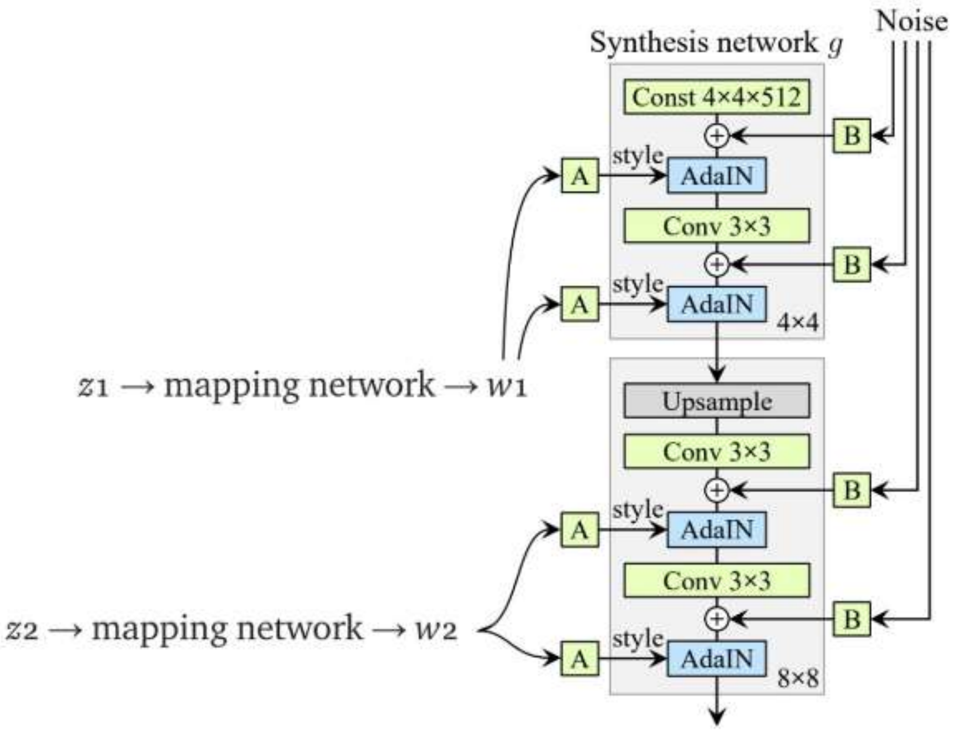
- Будем подавать шум в ***Synthesis Network*** на каждом уровне разрешения
- ***Mapping Network*** будет контролировать глобальный стиль (волосы, глаза)
- ***Noise*** будет отвечать за мелкие детали (морщины, веснушки)



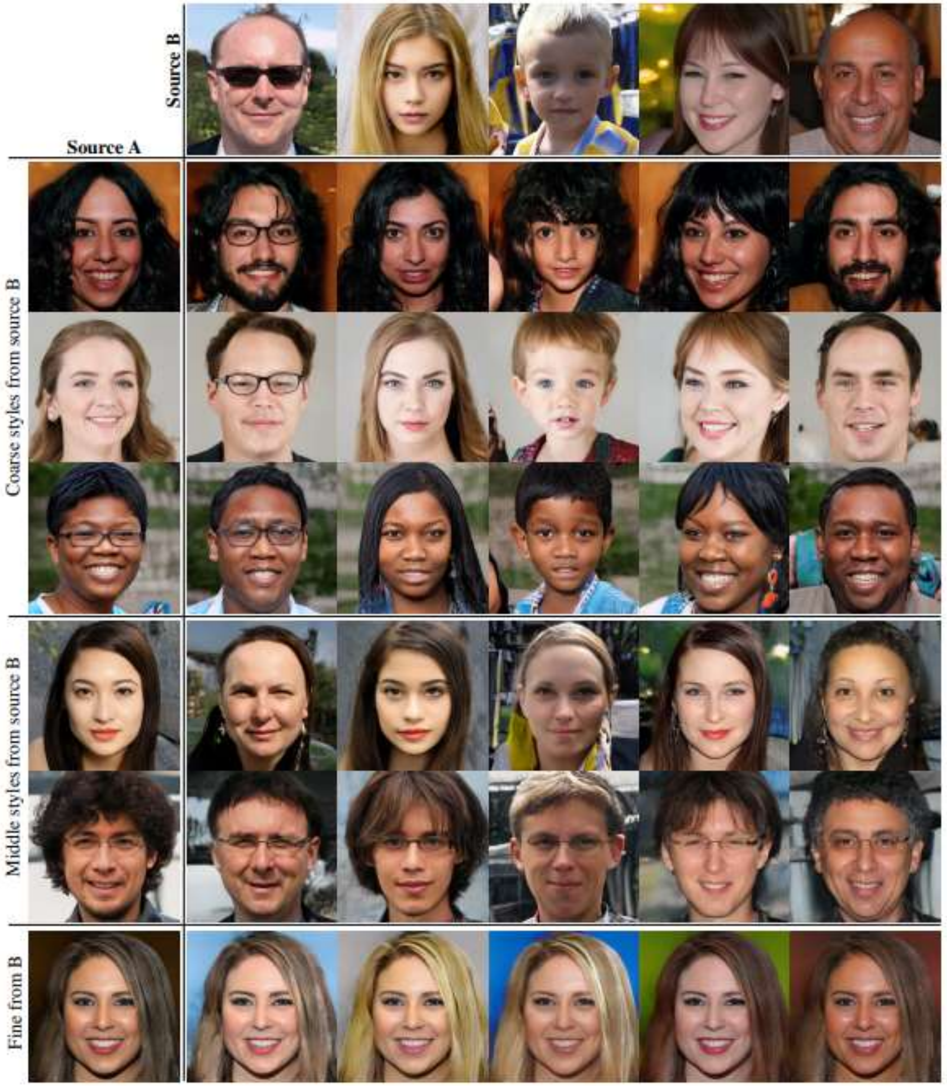
$StyleGAN: Style\ Mixing$

Можем попробовать взять 2 разных стилевых вектора $\boldsymbol{w}_1$ и $\boldsymbol{w}_2$

- На $coarse$ слоях возьмем $\boldsymbol{w}_1$, а на $fine$ $-\boldsymbol{w}_2$



StyleGAN: Style Mixing



Спасибо за внимание!