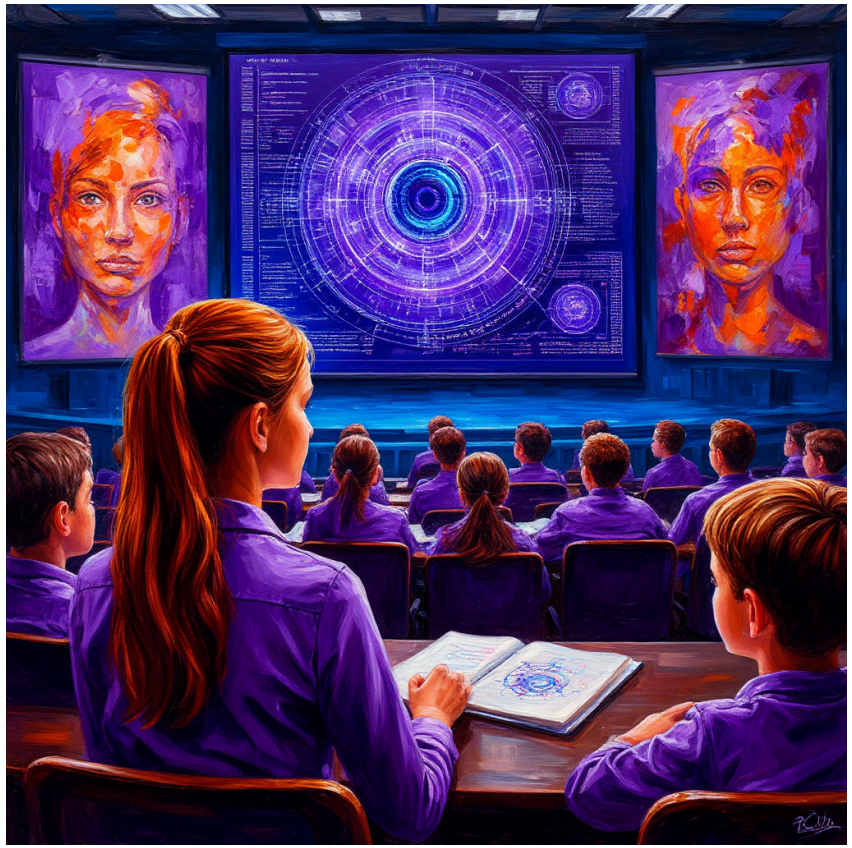
Введение в анализ данных



Генерация изображений

Задача генерации изображений

Хотим научиться генерировать изображения. Какие задачи бывают:

- Перенос стиля
- Генерация произвольных изображений
- Генерация изображений по тексту
- Super Resolution
- Inpainting / Outpainting
- Image-to-image Translation
- ...

Генерация изображений

- **Перенос стиля**
- Генерация произвольных изображений

Перенос стиля. Задача

Gatys et al. (2016)

Дано: изображение контента $\vec{p}$ и изображение стиля $\vec{a}$.

Задача: получить изображение $\vec{x}$, объединяющее данный контент и стиль.

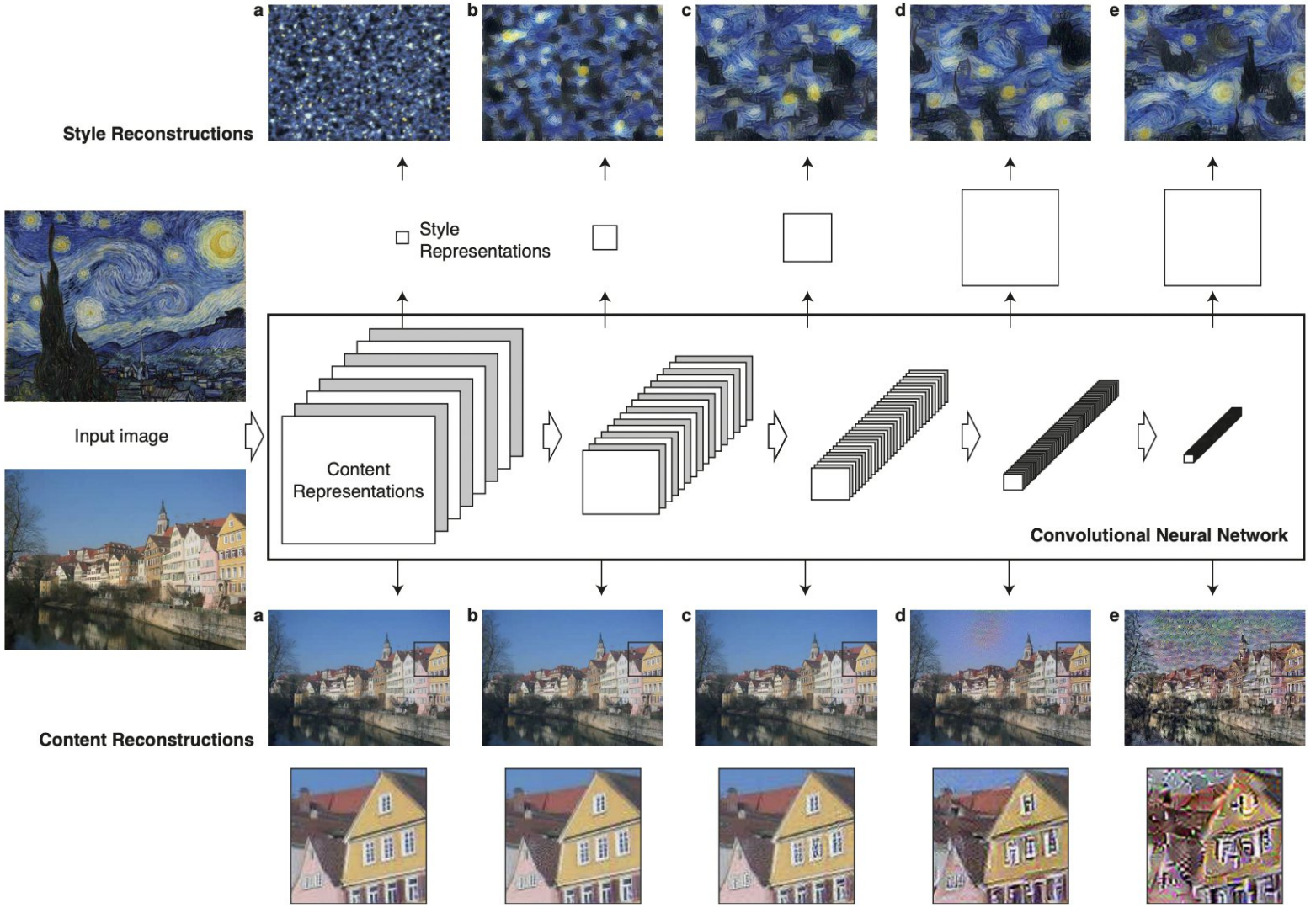
Перенос стиля. Применение свойств CNN

Content features

Выходы глубоких слоев сверточной нейросети хранят высокоуровневую информацию про изображенные объекты.

Style features – матрицы Грама по выходам слоев CNN, где

- начальные слои представляют мелкие текстуры
- более глубокие – особенности стиля художника



Перенос стиля. Алгоритм

Идея

Хотим оптимизировать картинку $\vec{x}$ так, чтобы:

- Content Features $\vec{x}$ были близки к Content Features картинки контента $\vec{p}$;
- Style Features $\vec{x}$ были близки к Style Features картинки стиля $\vec{a}$.

Алгоритм

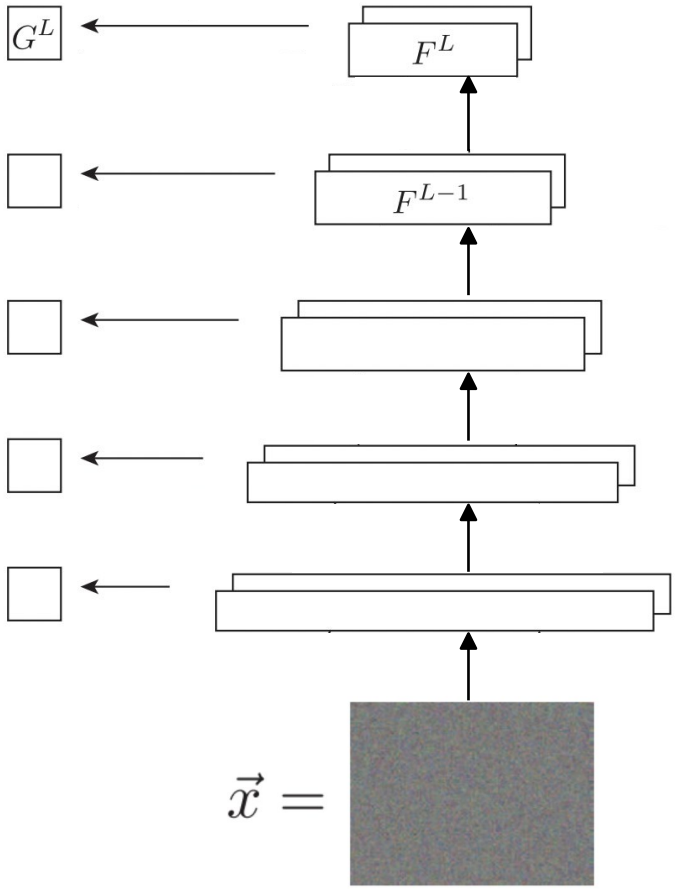
- инициализируем $\vec{x}$ нормальным шумом;
- задаем лосс близости контента $\mathscr{L}_{content}$ и лосс близости стиля $\mathscr{L}_{style}$;
- оптимизируем $\vec{x}$ градиентным спуском.

Перенос стиля. Алгоритм

Применяем предобученную сверточную сеть (VGG19) к картинке $\vec{x}$.

G^L – матрица Грама для выходов F^L сверточного слоя L:

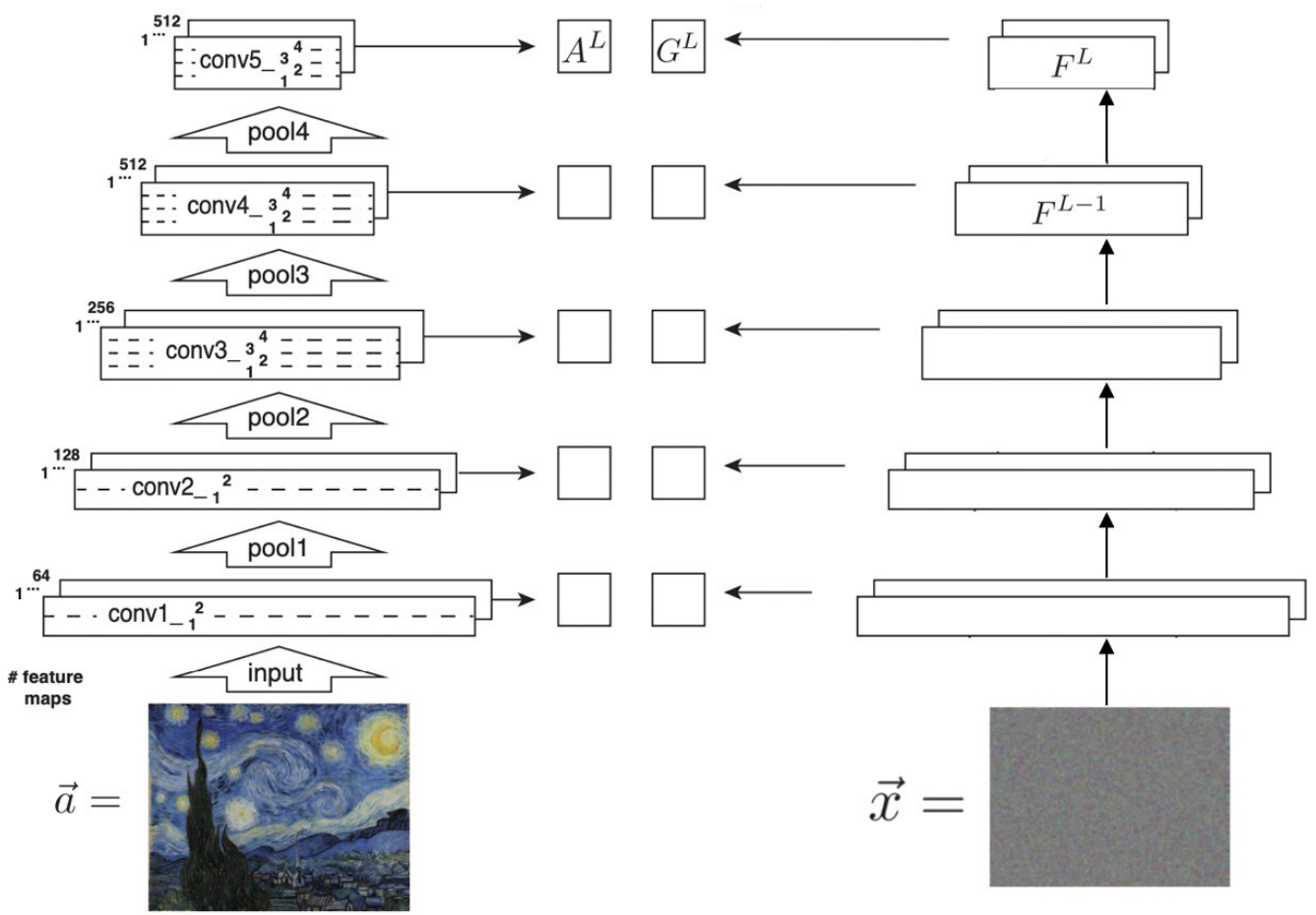
$$G^L_{ij} = \sum_k F^L_{ik} F^L_{jk}.$$



Перенос стиля. Алгоритм

Применяем сверточную сеть к картинке стиля $\vec{a}$.

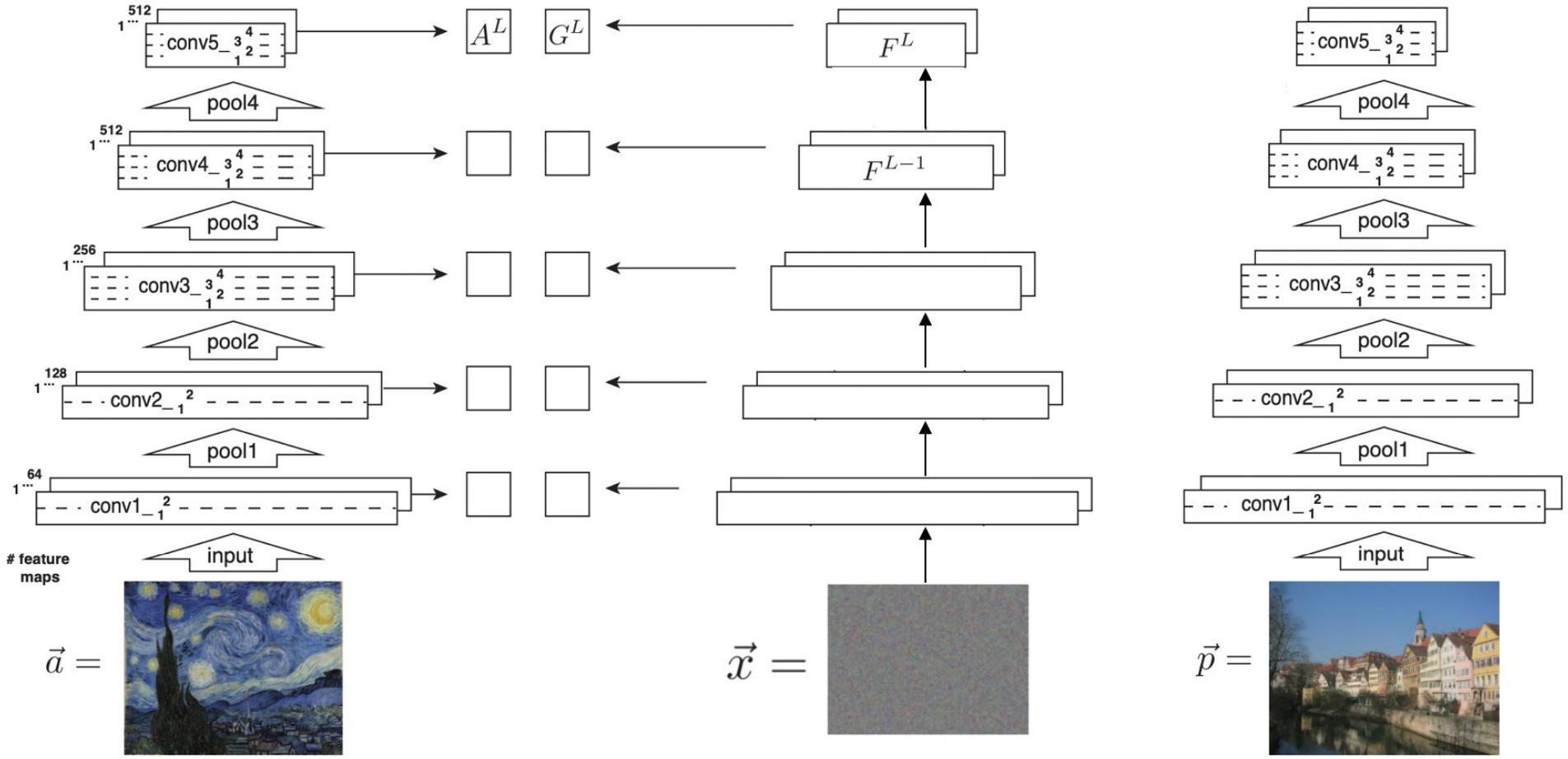
A^L – матрица Грама для выходов сверточного слоя L.



Перенос стиля. Алгоритм

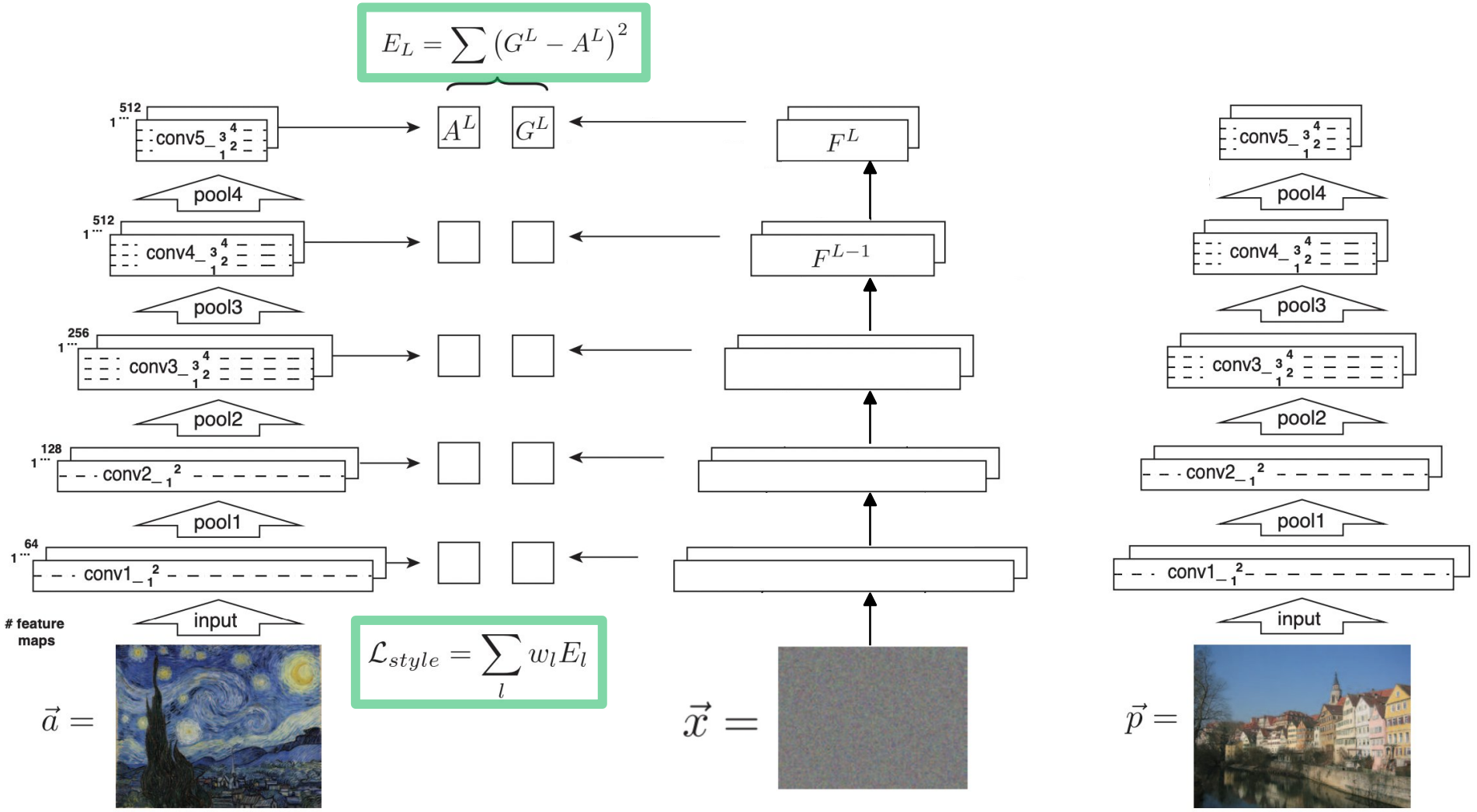
Применяем сверточную сеть к картинке стиля $\vec{p}$.

P^L – выход сверточного слоя L.



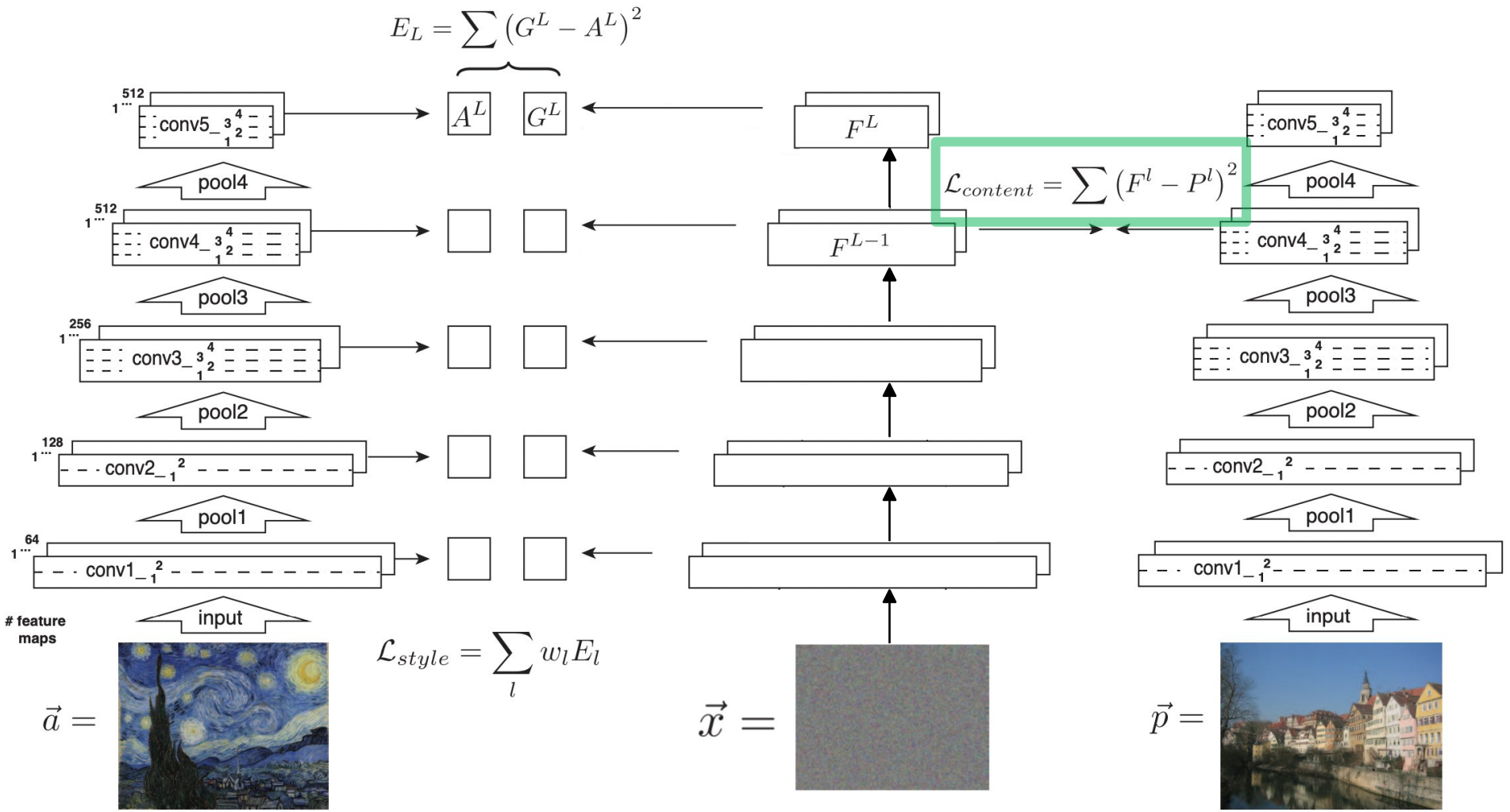
Перенос стиля. Алгоритм

$\mathscr{L}_{style}$ – сумма
по слоям MSE
для матриц Грама



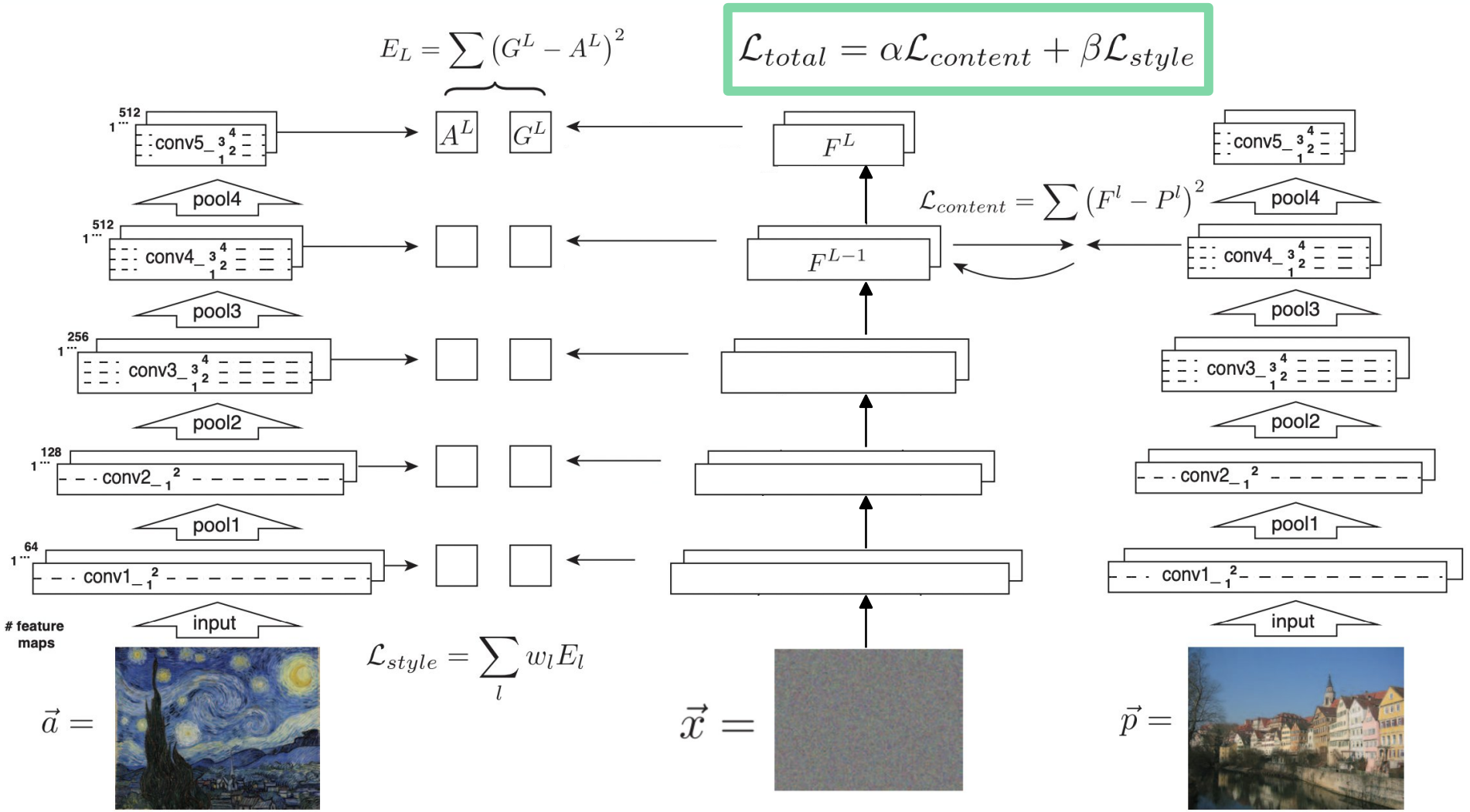
Перенос стиля. Алгоритм

$\mathscr{L}_{content}$ – сумма по слоям MSE для выходов сверточных слоев



Перенос стиля. Алгоритм

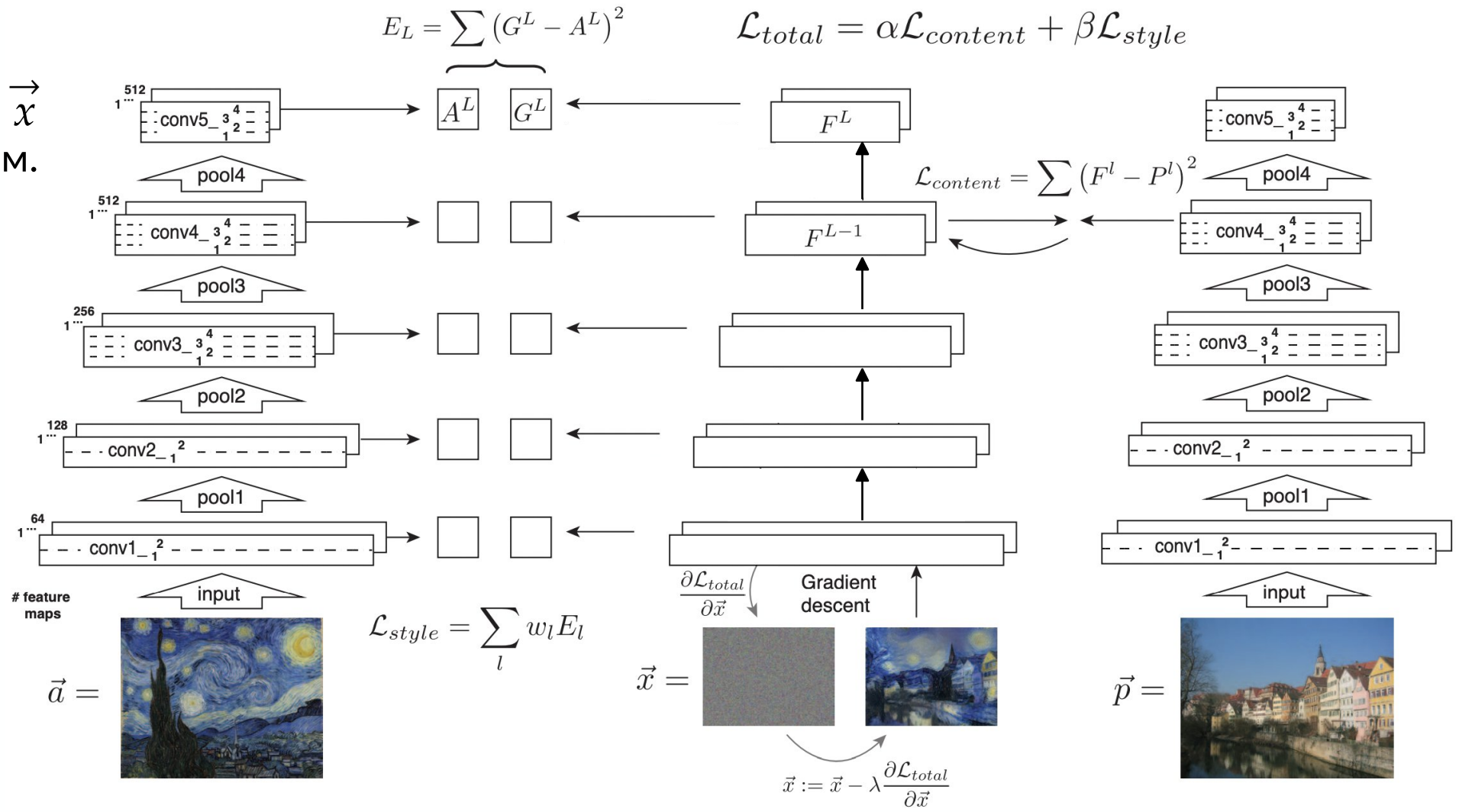
$\mathcal{L}_{total}$ – взвешенная сумма $\mathcal{L}_{content}$ и $\mathcal{L}_{style}$



Перенос стиля. Алгоритм

Делаем обновление $\vec{x}$ градиентным спуском.

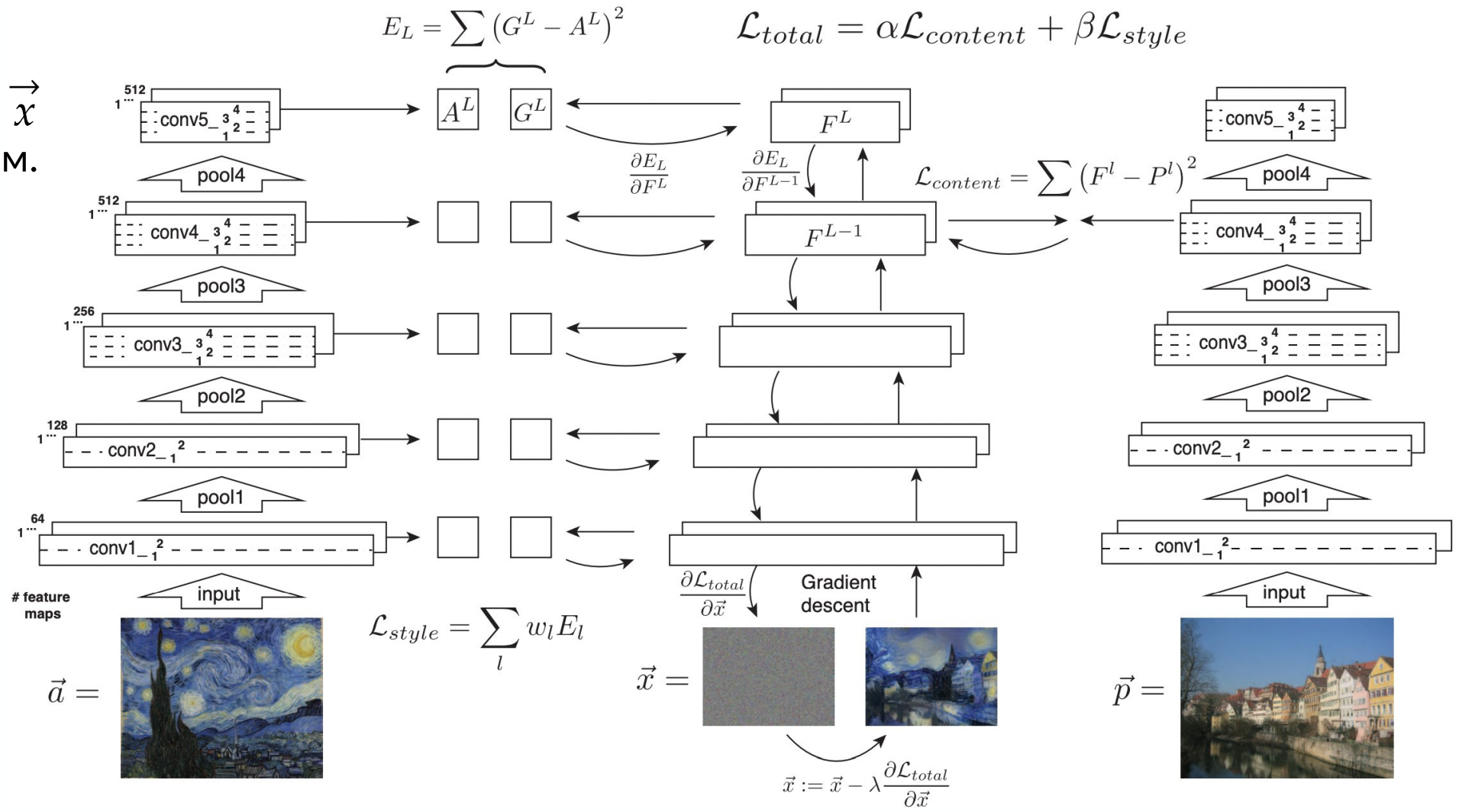
В итоге получаем искомую картинку.



Перенос стиля. Алгоритм

Делаем обновление $\vec{x}$ градиентным спуском.

В итоге получаем искомую картинку.



Влияние коэффициентов лосса

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$$

Результаты при разных значениях отношения α/β коэффициентов компонент лосса:

10^{-4}

10^{-3}

10^{-2}

10^{-1}

Генерация изображений

- Перенос стиля
- **Генерация произвольных изображений**

Генеративные модели. Задача

Пусть P – некоторое распределение данных, например, распределение картинок с котиками. Хотим сгенерировать новые объекты, которые:

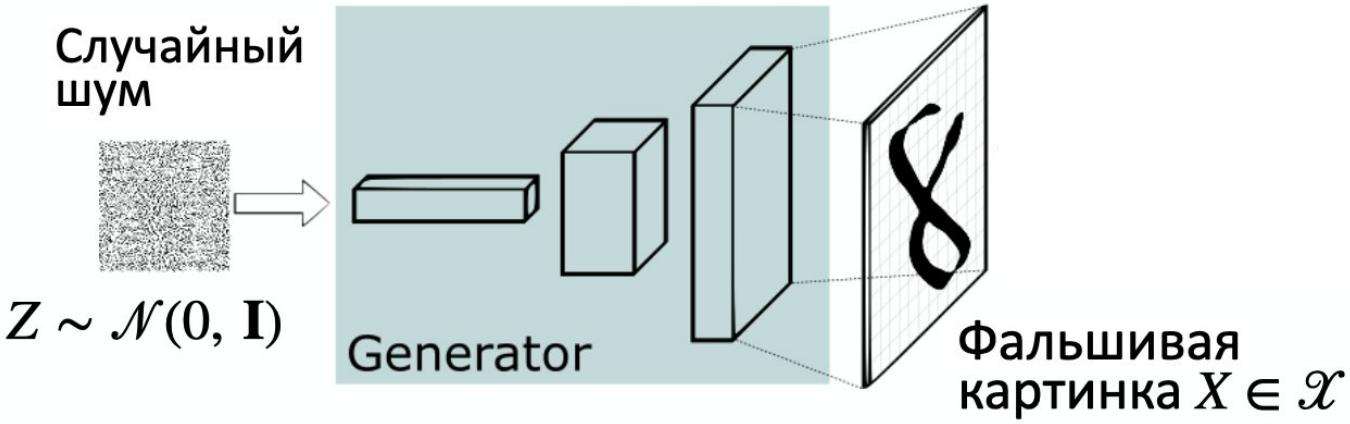
- выглядят так, будто получены из P;
- отличаются от уже существующих.

Генеративные модели. Генератор

P – некоторое распределение картинок $X \in \mathcal{X}$.

Как получить модель, генерирующую картинки из P, которых мы еще не видели?
Будем давать ему на вход шум - случайный вектор Z.
Это будет нашим генератором.

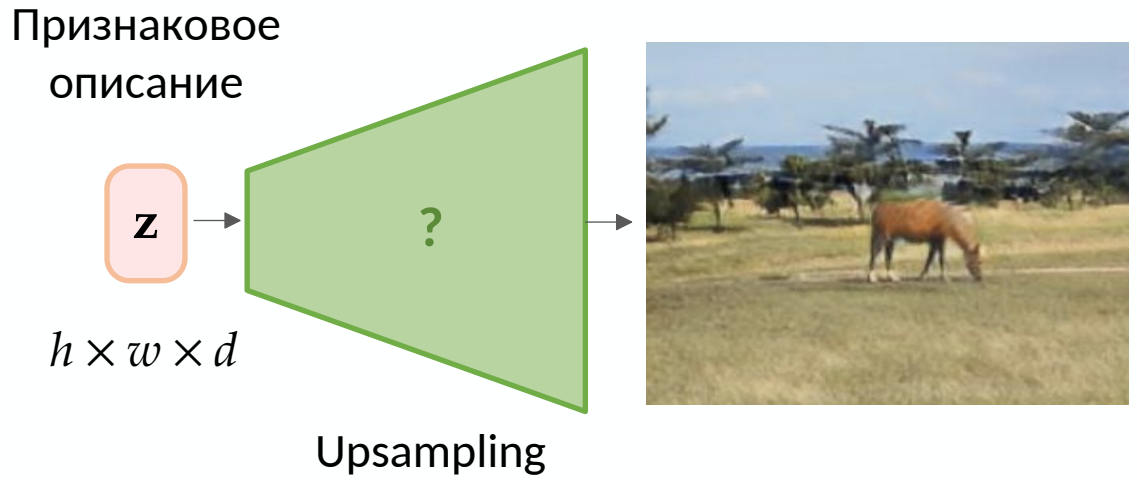
Но как построить такую сеть?



Про природу пространства представлений Z узнаем на 4 курсе DS-потока.

Генеративные модели. Генератор

Обычно для генерации используется модель, повышающая размерность входа. Нужна операция, которая из тензора размера $h \times w \times d$ получает тензор размера $H \times W \times K$, где $h < H$, $w < W$, то есть операция повышения размерности. Такая операция называется **Upsampling**.

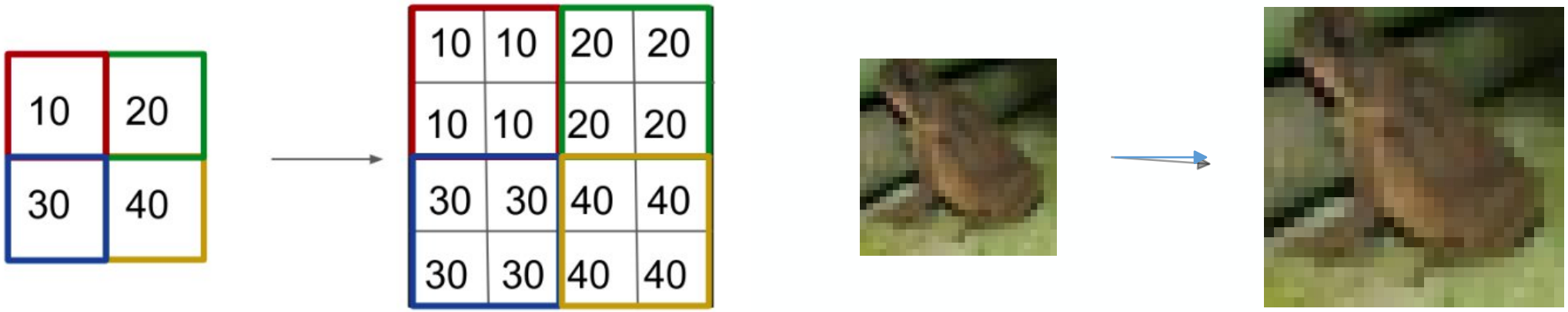


Про природу пространства представлений Z узнаем на 4 курсе DS-потока.

Upsampling: интерполяция

Нужна операция, которая из тензора размера $h \times w \times d$получает тензор размер $H \times W \times K$, где $h < H, w < W$, то есть операция повышения размерности. Такая операция называется **Upsampling**. Рассмотрим очевидный способ:

По ближайшему соседу
(Nearest Neighbors)



Upsampling: интерполяция

Билинейная (Bilinear)

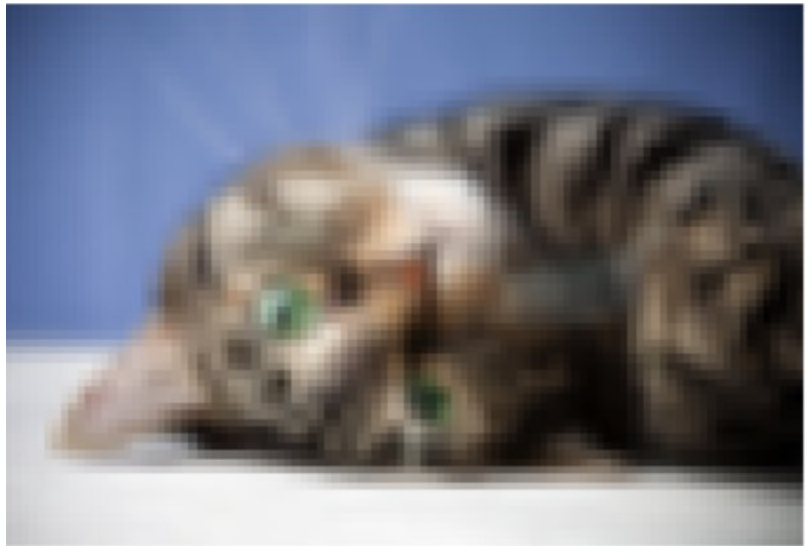
Перенесем значения исходной матрицы таким образом, чтобы по краям новой матрицы матрицы оказались краевые значения исходной матрицы.

10	20
30	40

→

10	13	17	20
17	20	23	27
23	27	30	33
30	33	37	40

→

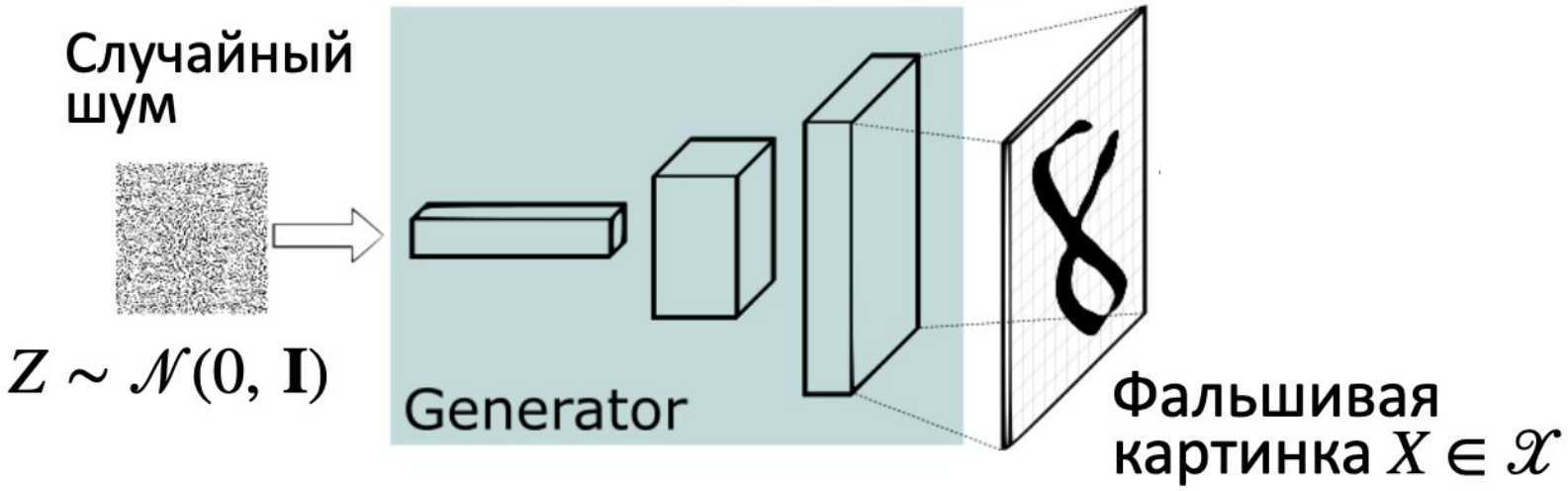


Генератор картинок

P – некоторое распределение картинок $X \in \mathscr{X}$.

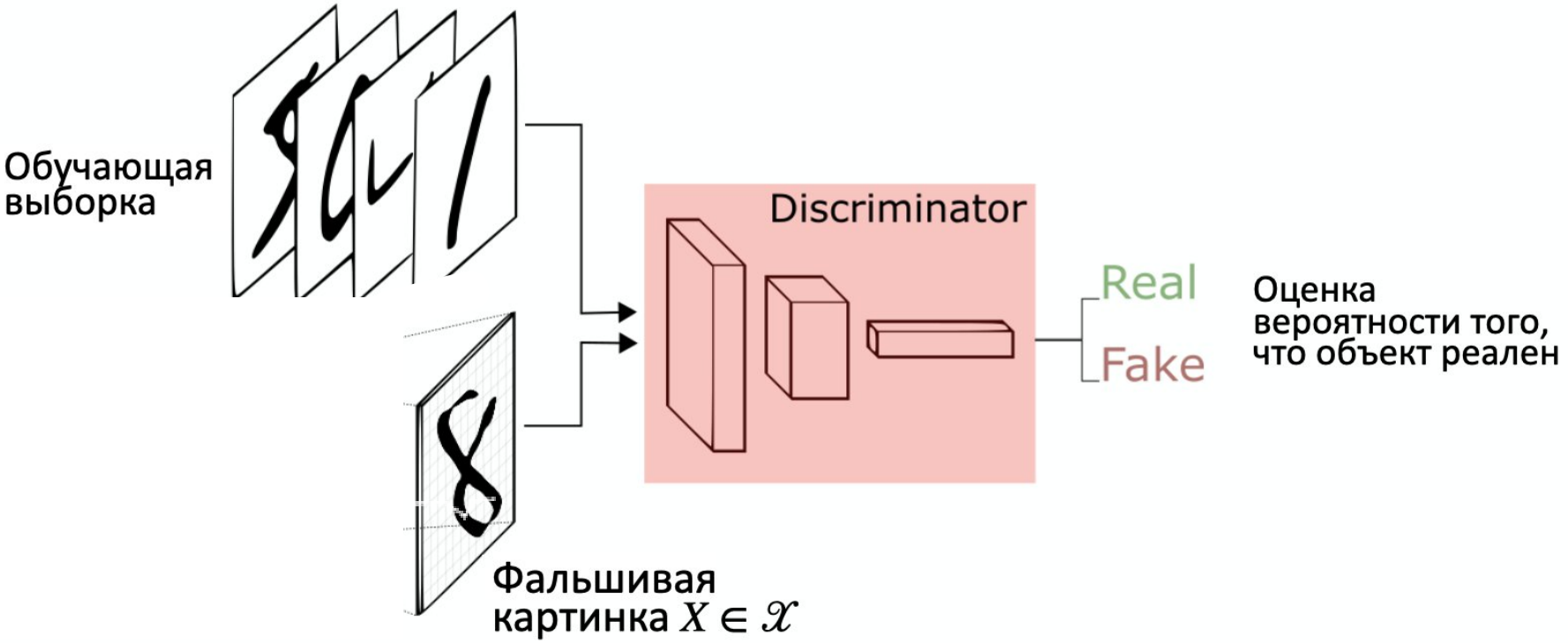
Как получить модель, генерирующую картинки из P, которых мы еще не видели?
Будем давать ему на вход шум - случайный вектор Z.
Это будет нашим генератором.

Но как такую нейросеть обучать?



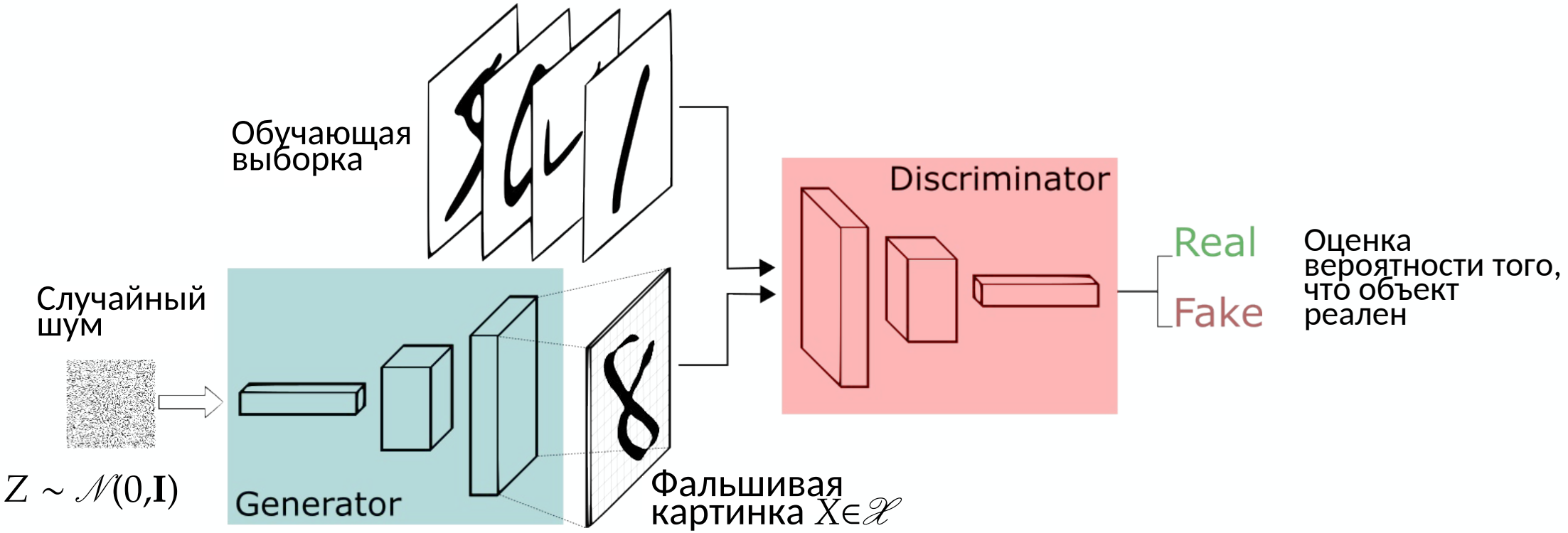
Дискриминатор

Для определения того, настоящая картинка или сгенерированная, заведем модель Discriminator. Это будет обычный классификатор, принимающий на вход $X \in \mathscr{X}$ и выдающий 0 фальшивкам, и 1 примерам из датасета.



GAN: Generative Adversarial Network

Модель GAN призвана генерировать реалистичные объекты
из распределения картинок и состоит из генератора и дискриминатора.
Они обучаются попеременно.



GAN: Generative Adversarial Network

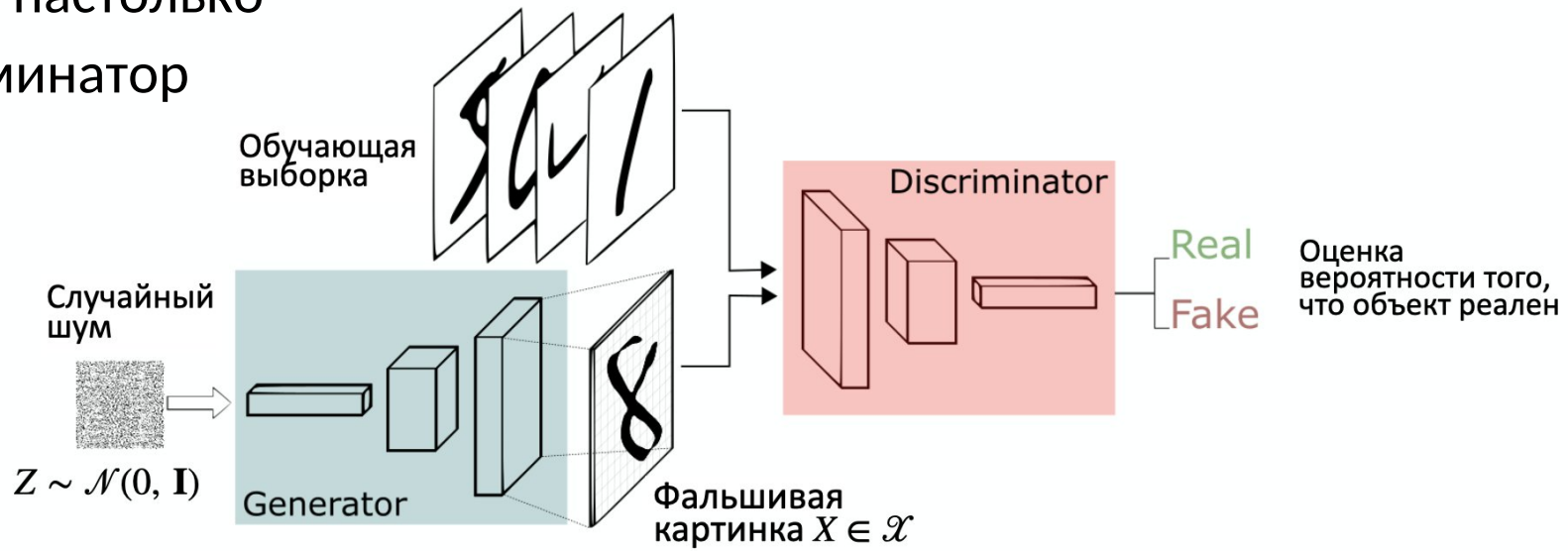
Обучение называется **состязательным**, так как генератор и дискриминатор решают противоположные задачи:

- **Дискриминатор** хочет выдавать 1 для истинных картинок и 0 для сгенерированных:

$$\mathrm{E} \log D(X) + \mathrm{E} \log [1 - D(G(Z))] \longrightarrow \max_{D}$$

- **Генератор** хочет уметь обманывать дискриминатор – генерировать настолько хорошие картинки, что дискриминатор подумает, что они реальны:

$$\mathrm{E} \log D(G(Z)) \longrightarrow \max_{G}$$



Современная генерация картинок

Развитие генеративных моделей

VAE
2013

GAN
2014

DCGAN
2015

StyleGAN
2018

DaLLE-2
2022

VAE разберем подробнее на байесовских методах, StyleGAN – на курсе по генеративным моделям на 4 курсе DS-потока.

Диффузионная модель

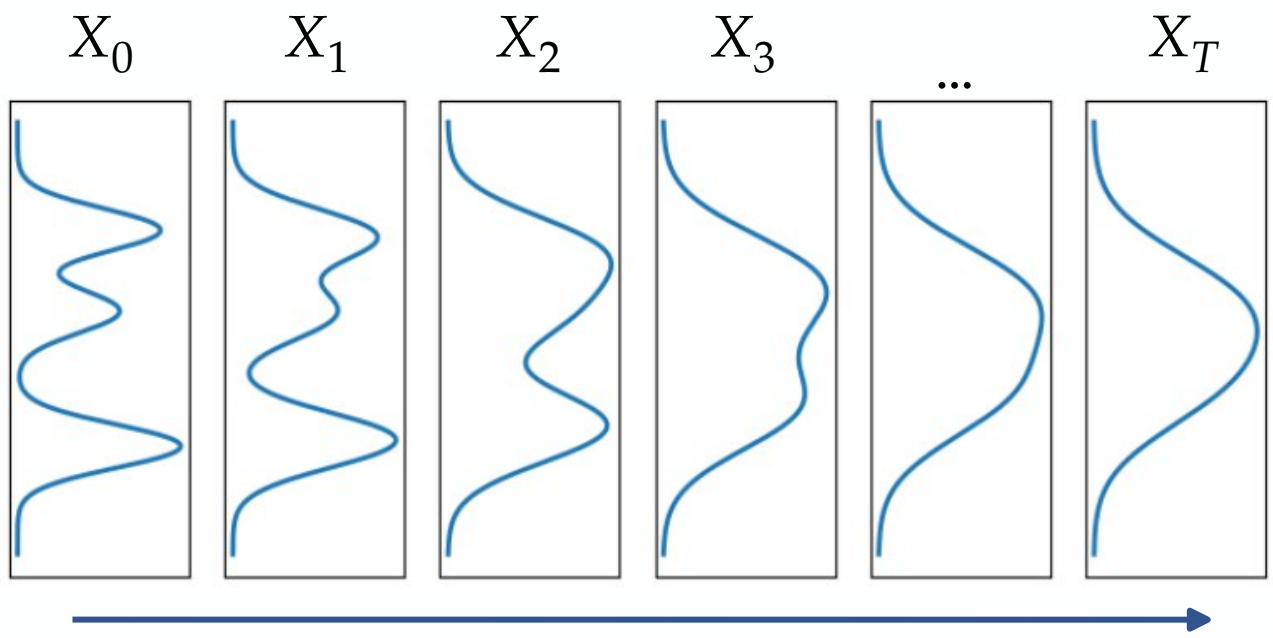
Диффузионная модель состоит из 2 процессов.

- **Прямой процесс** — постепенно добавляем шум ко входу.
- **Обратный процесс** — модель постепенно восстанавливает данные из шума.

Прямой диффузионный процесс

При $T \to \infty$, $X_T \xrightarrow{d} N(0, I)$.

На последнем шаге итераций получаем гауссовский шум.



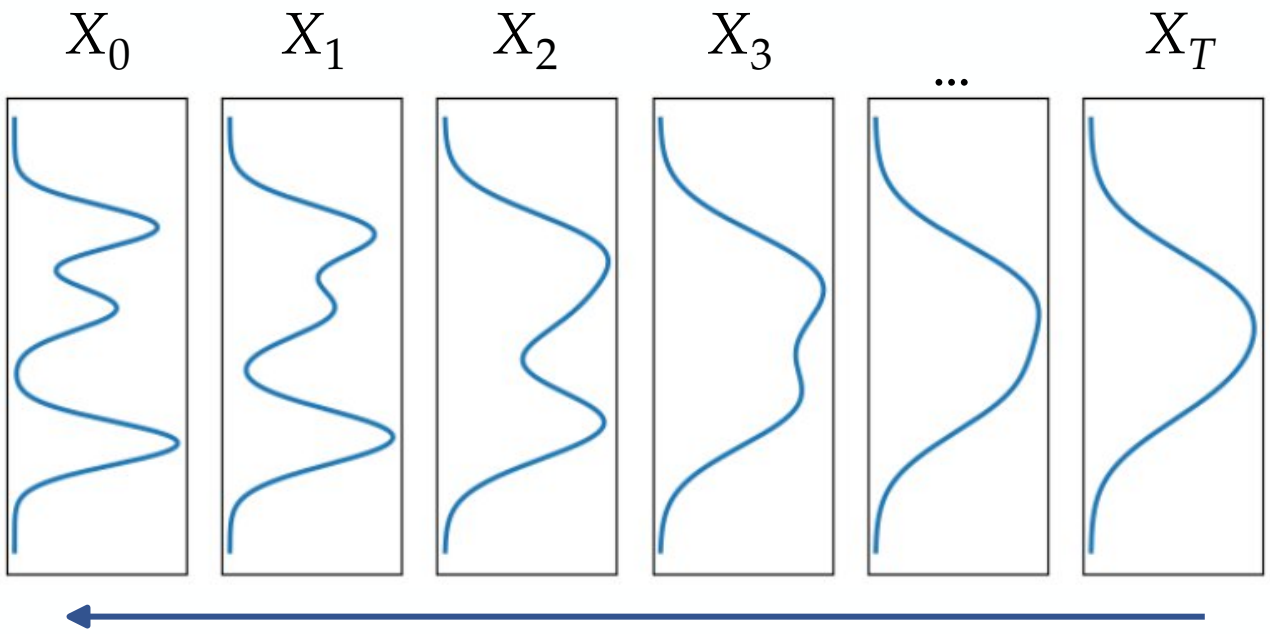
Прямой диффузионный процесс

Обратный диффузионный процесс

Хотим восстановить исходное изображение.

Знаем $X_T \sim N(0, I)$. Будем итеративно семплировать из условного распр. $X_{t-1}|X_t$.

Но как?



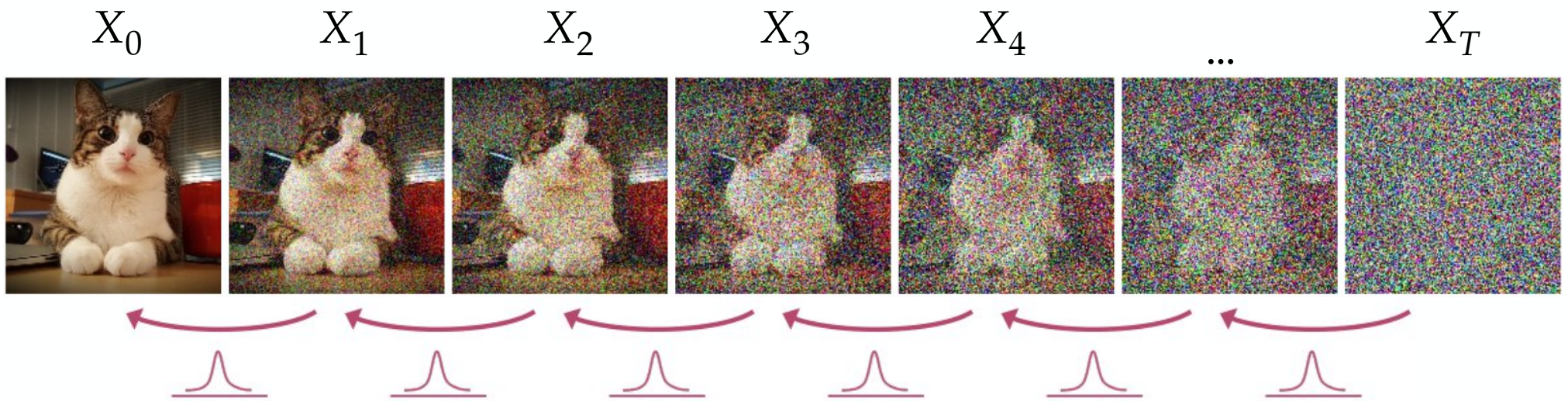
Обратный диффузионный процесс

Обратный диффузионный процесс

Будем **аппроксимировать условное распр. $X_{t-1}|X_t$ нормальным распределением**, среднее которого получается из нейросети, параметризуемой θ.

$$X_{t-1}|X_t \sim N\left(\mu_\theta(X_t, t), \sigma_t^2 I\right).$$

В результате X_0 – сгенерированная из шума картинка.



Обратный диффузионный процесс

Обучение диффузионной модели

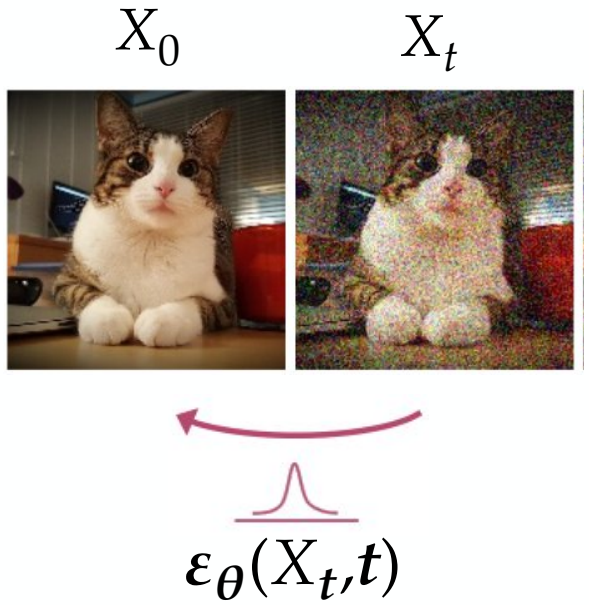
На практике учим нейронную сеть предсказывать не μ_θ,

а **реконструкцию шума** ε_θ, т.е. шум, который был добавлен к X_0 на итерации t прямого диффузионного процесса.

При этом $\mu_\theta(X_t, t)$ выражается через $\varepsilon_\theta(X_t, t)$.

Почему?

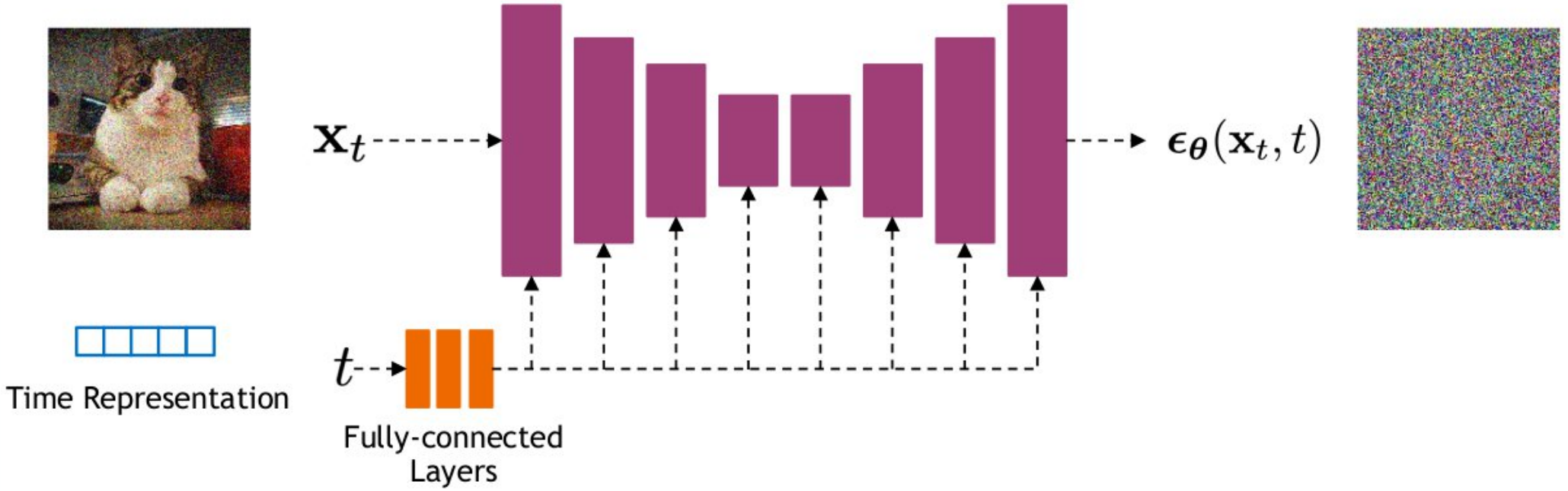
В таком случае сеть учится решать задачу в пространстве с простым распределением выходов сети, что приводит к быстрой сходимости.



Подробнее – на 4 курсе DS-потока.

Диффузионная модель. Реализация

- Обычно для предсказания $\varepsilon_\theta(X_t,t)$ используется U-Net-подобная архитектура
- Для времени t подсчитываются некоторые признаки и представления через MLP-сеть
- Преобразованный вектор времени подается на слои U-Net как дополнительная информация



ВСЁ!