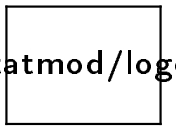
Deep learning, Neural Nets for images

Попов Владимир
Шаповал Егор
Леонович Роман



Санкт-Петербург
2022 г.

Свёрточная нейронная сеть

Definition

Изображение — тензор $M \in \mathbb{R}^{m \times n \times d}$, m — ширина изображения, n — длина. Чаще всего $d = 3$ (3 канала — Red, Green, Blue).

Пусть $X \in \mathbb{R}^{m \times n \times d}$ — случайная величина "изображение".
Задачи:

- Классификация $f : X \to \{1, \ldots, K\}$, K — число классов
- Сегментация $f : X \to Y$, $Y \in [0, 1]^{m \times n}$, $Y_{ij} = \mathsf{P}(X_{ij} \in \text{segment})$

Свёртка изображений

Definition

Рассмотрим $M \in \mathbb{R}^{m \times n}$ — изображение по одному из каналов, $K \in \mathbb{R}^{k \times l}$ — ядро, *свёрткой изображения относительно отображения* $h : \mathbb{R}^{m \times n} \times \mathbb{R}^{k \times l} \to \mathbb{R}$ *называется функция* $*$,

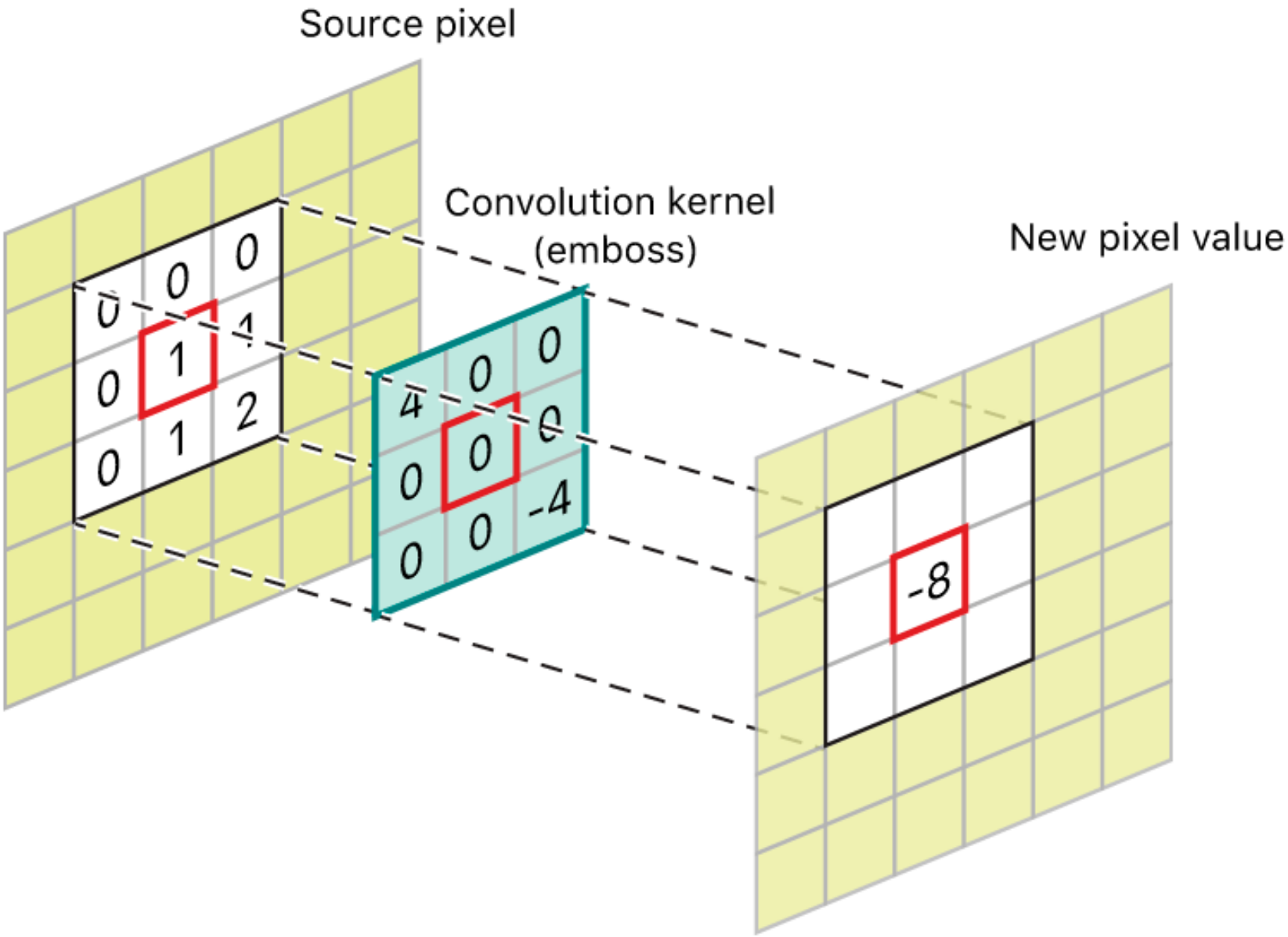
рассмотрим $h(\mathbf{X}, \mathbf{Y}) = \sum_{a=1}^{k} \sum_{i=1}^{l} X_{ij} Y_{ij}$,

$$(M * K)(i, j) = \sum_{a=1}^{k} \sum_{b=1}^{l} M_{i+a, j+b} K_{ab}.$$

$(M * K) \in \mathbb{R}^{(m-k+1) \times (n-l+1)}$

Можем заполнить как-нибудь края чтобы результат свёртки имел размерность $m \times n$ (padding).

Свёртка наглядно



Свёрточный слой

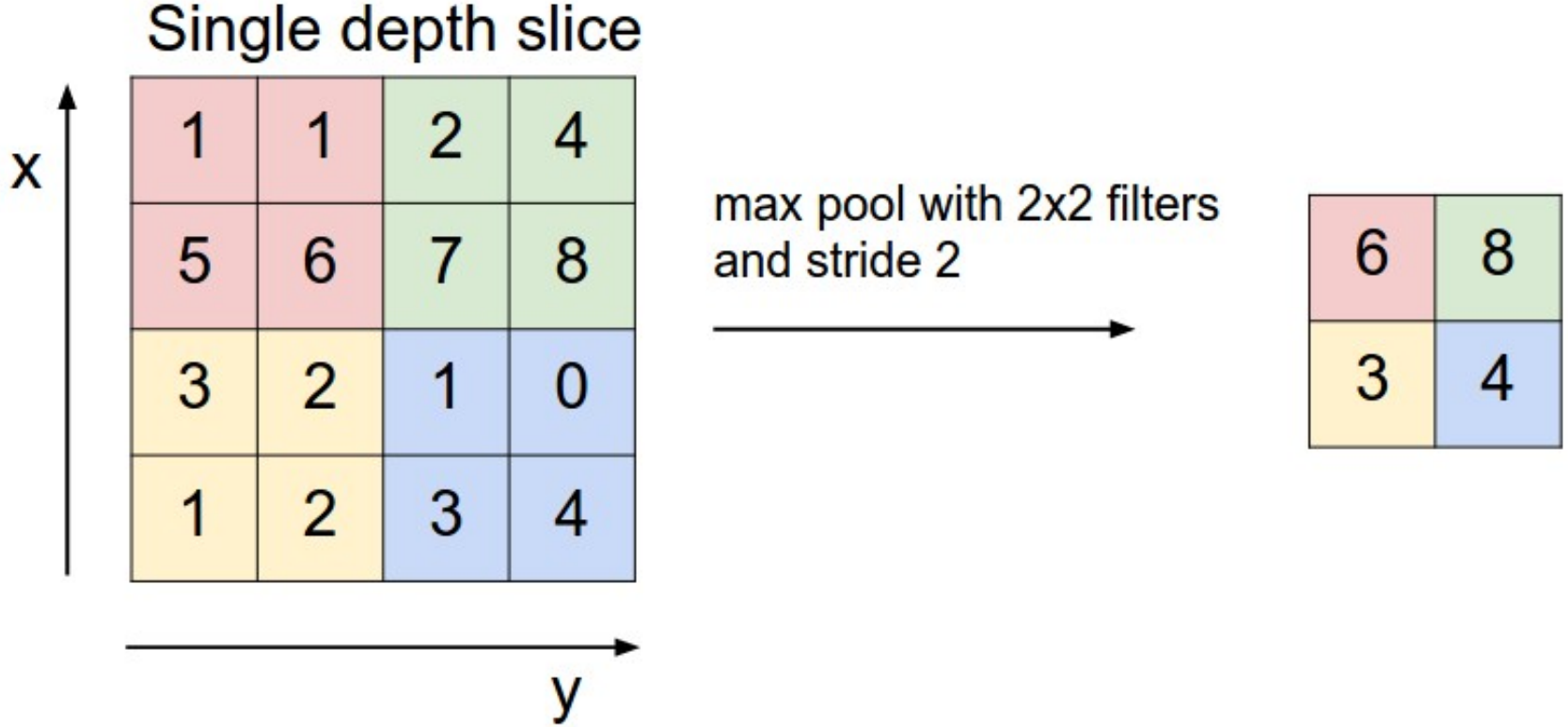
Сверточный слой CNN задаётся следующими параметрами:

- Размер фильтра $k \times l \times r$, $k \leq m$, $l \leq n$, $r \leq d$.
- Способ заполнения краёв (padding), например можно заполнить нулями, равносильно окаймлением изображения чёрной рамкой. Есть другие способы: max, reflect, replicate.
- Размер заполнения краёв $P \in \mathbb{R}^d$.
- Величина сдвига ядра (stride) $S \in R^3$. При $S = (1, 1, 1)$ получаем операцию, похожую на операцию вложения в SSA.

На выходе получаем тензор размерности $m_1 \times n_1 \times d_1$, после этого применяем к полученным элементам функцию активации.

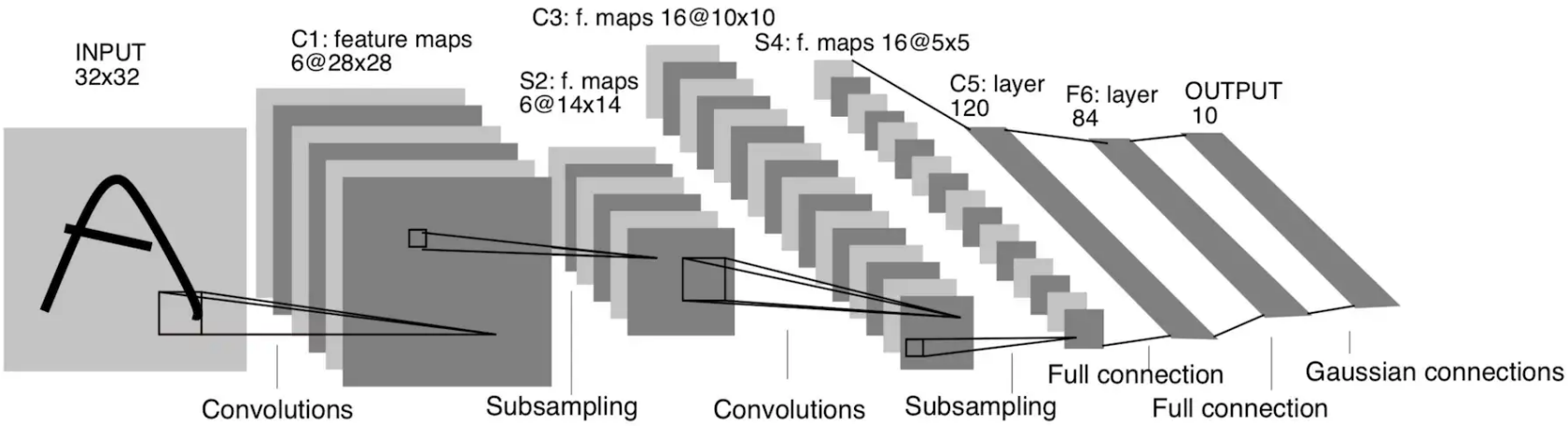
Pooling слой

Pooling операцию можно понимать как свёртку с $S = (k, l, d)$.
Основная цель pooling — уменьшение размерности изображения. Часто используют max-pooling, иногда применяют sum-pooling, average-pooling



Свёрточная нейронная сеть

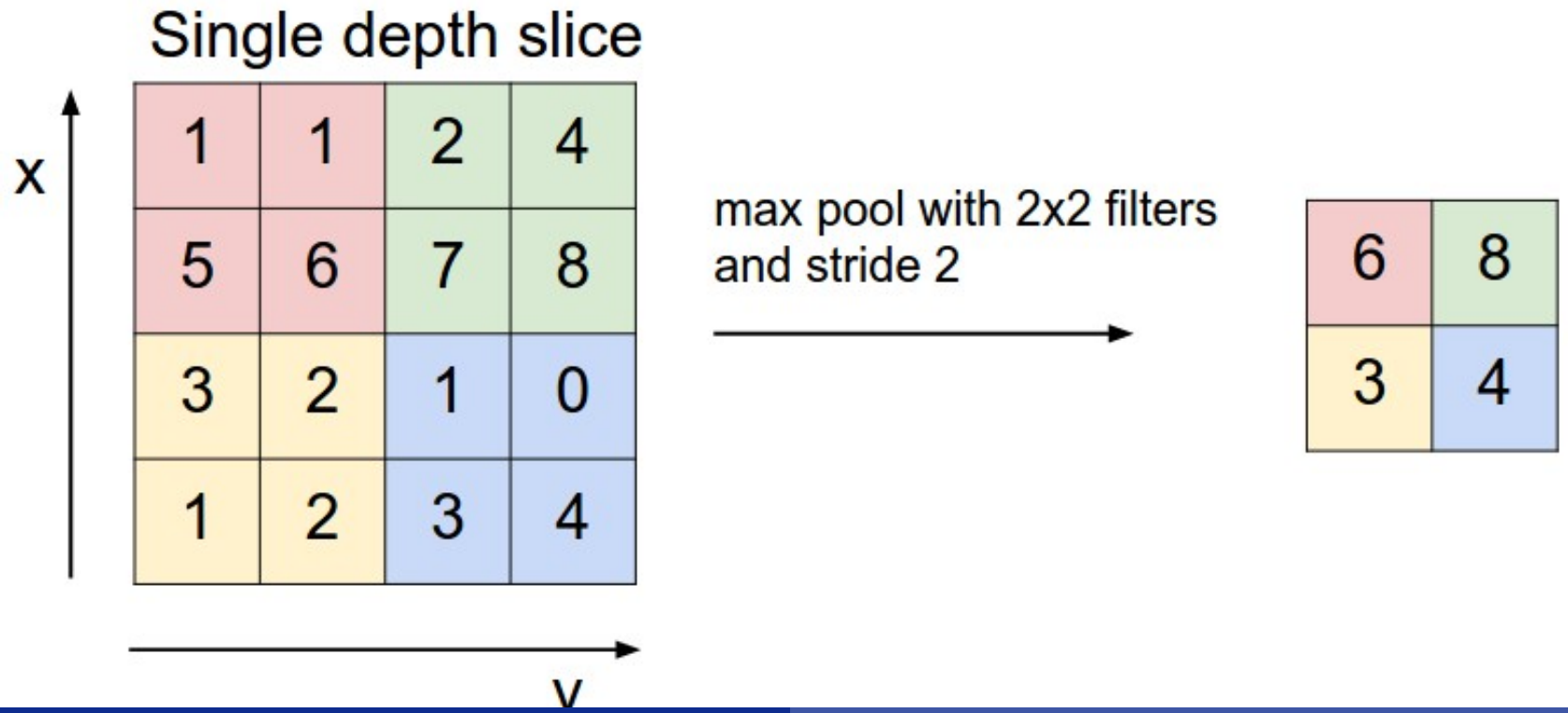
В зависимости от постановки задачи после применения свёрточных и pooling слоёв и применения функций активации могут следовать полносвязные или свёрточные слои.



Backpropogation

В случае отсутствия max-pooling слоёв применим алгоритм обратного распространения ошибки. Если присутствуют max-pooling слои, то градиенты пробрасываются в ту клетку, на которой достигается максимум, остальные градиенты равны нулю.

В примере ненулевые градиенты будут в точках $(2,2),(2,4),(3,1),(4,4)$.

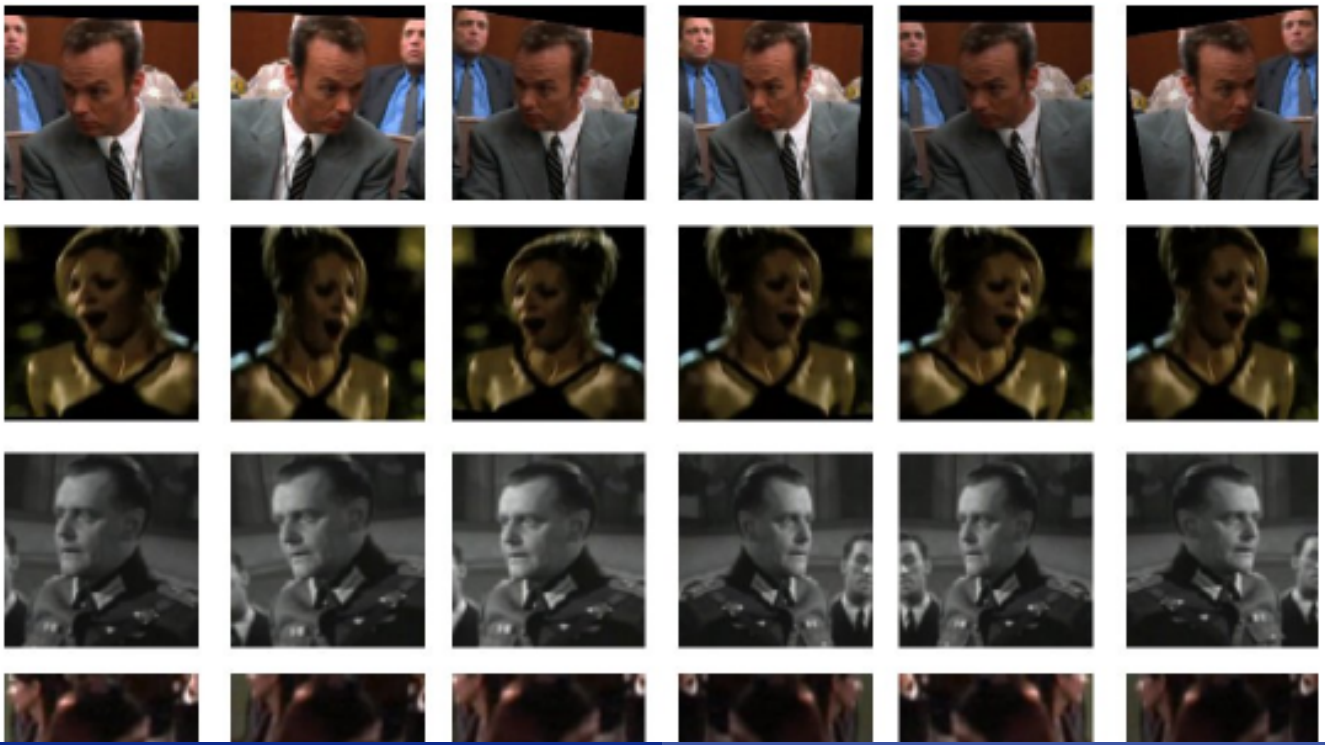


Augumentation

Definition

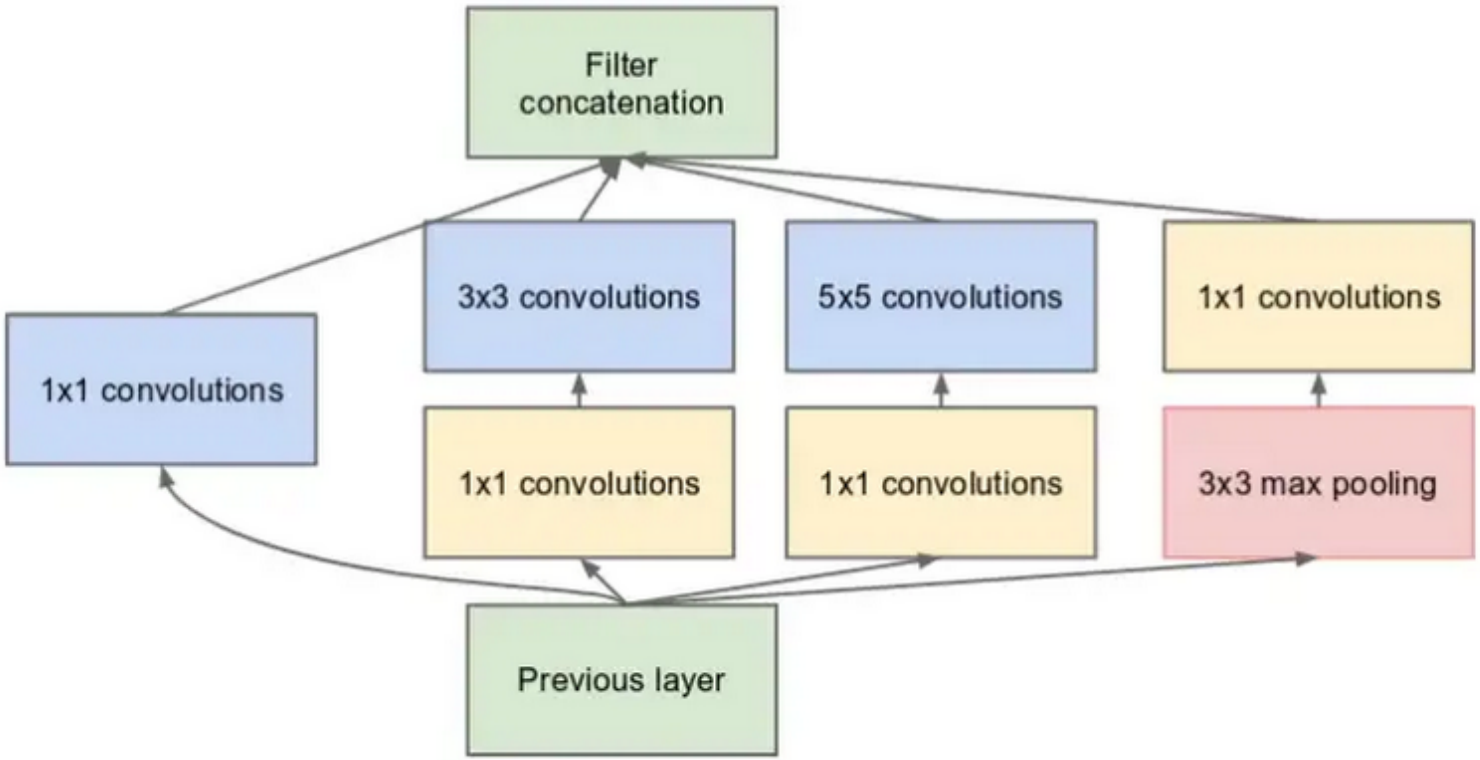
Аугументация (Augumentation) — увеличение объёма тренировочной выборки с помощью различных афинных преобразований изображений: зеркальное отражение, поворот, сдвиг, изменение масштаба.

Используется для борьбы с переобучением.



Inception

Проблема: непонятно какой размер ядра на каждом слое будет давать минимальную ошибку
Решение: Рассмотрим разные комбинации свёрток и pooling слоёв, а затем сконкатенируем их

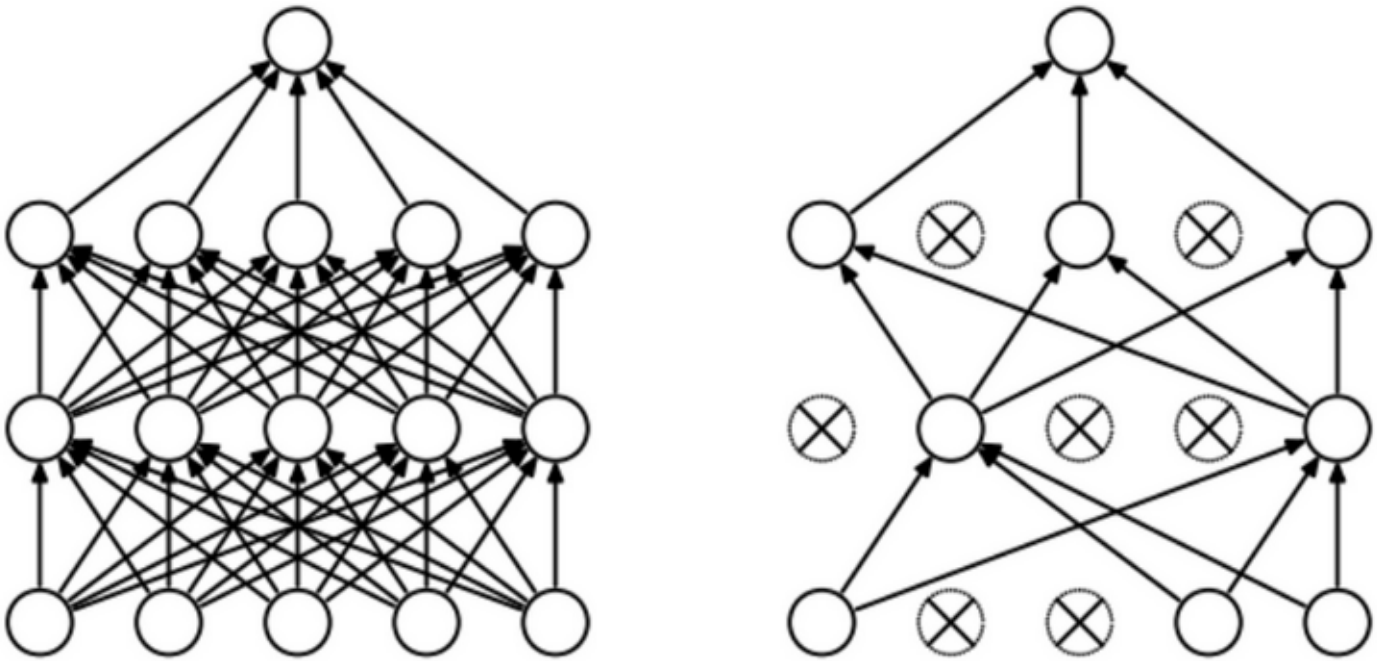


Dropout (только для полносвязанных сетей)

Definition

Dropout — отключение (зануление) случайных нейронов во время обучения нейросети. Параметр p — доля отключаемых нейронов. Оставшимся ненулевым нейронам присваиваем вес, равный $\frac{1}{1-p}$.

Цель: борьба с переобучением



Практические проблемы

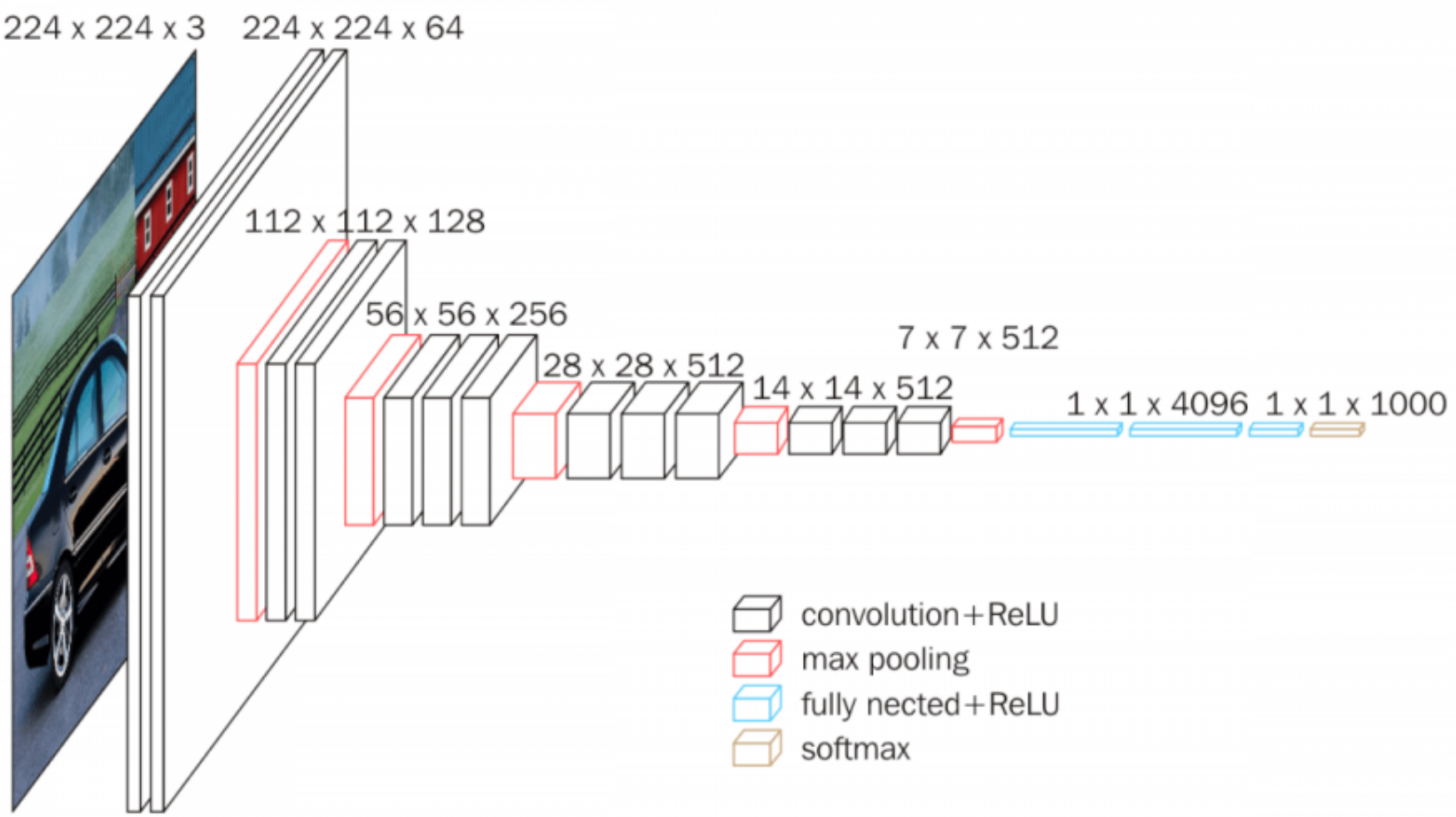
Проблемы:

- Необходимость разметки данных для обучения
- Большое количество параметром, следовательно долгое обучение, даже на GPU

Решения:

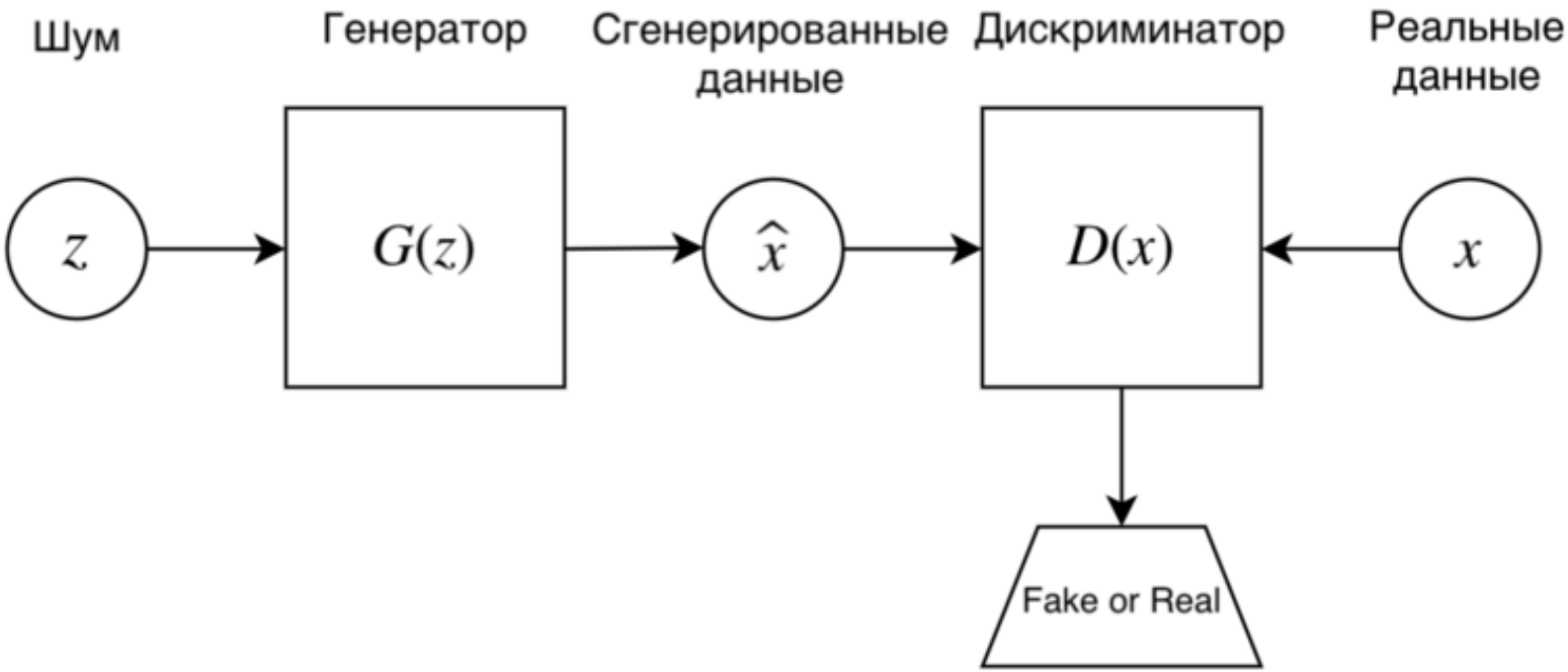
- Использование размеченных библиотек изображений: Imagenet (14M изображений, 1000 категорий), OpenImages (9M изображений, 60K меток, 20K категория)
- Использование предобученной модели (Alexnet, vgg net, Resnet)

vgg16


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

GAN — Порождающие состязательные сети

- Увеличение разрешения изображений.
- Преобразование текста в изображение.
- Раскрашивание изображений.
- Генерация большего количества данных и заполнение пробелов в них.



GAN. Генератор

Цель генератора: Сгенерировать такие данные, чтобы дискриминатор не отличил их от реальных.
Цель дискриминатора: Определить, входные данные реальны, или были сгенерированы искусственно.

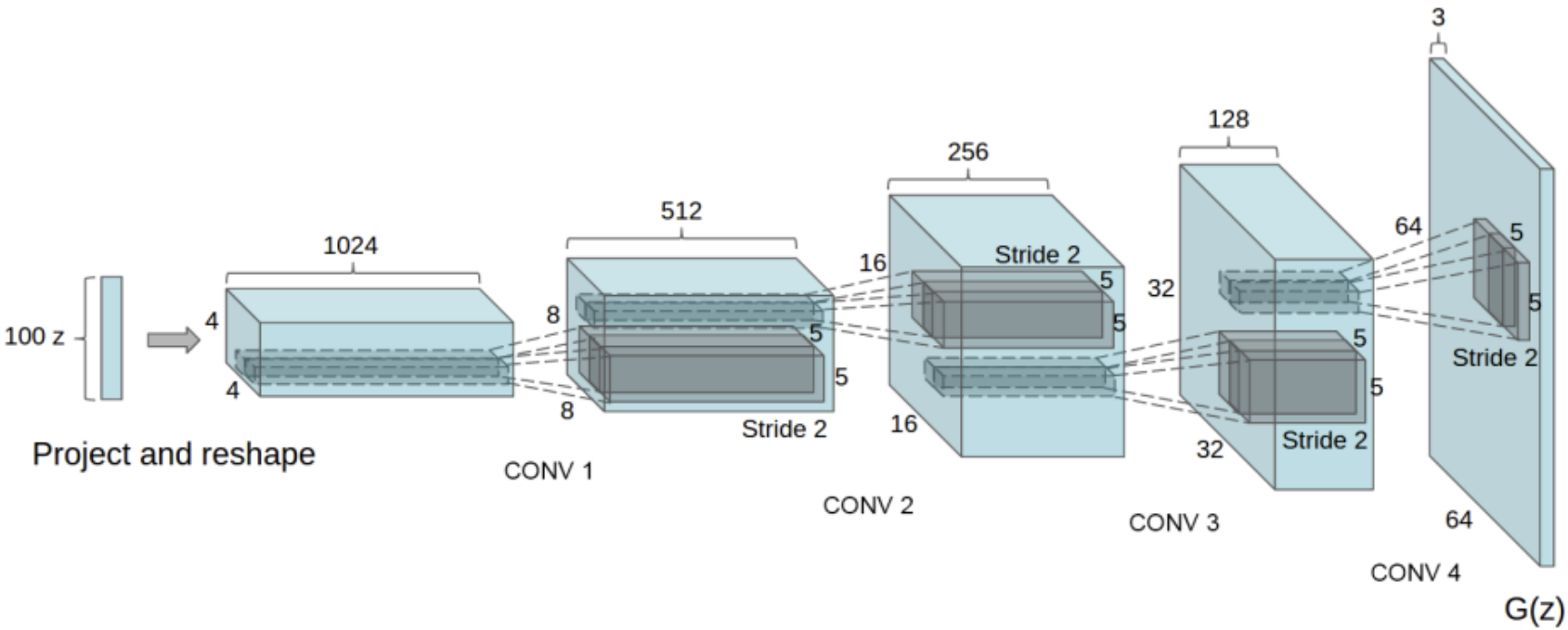


Рис.: Генератор

Постановка задачи

- $X \in \mathbb{R}$ — Набор данных;
- p_g — Вероятностное распределение генератора;
- $p_z(z)$ — Априорная вероятность шума;
- $G(z, \gamma_g)$ — Генератор, где G многослойный перцептроном с параметром γ_g;
- $D(z, \gamma_d)$ — Дискриминатор, который на выход подает вероятность того, что x пришло из тренировочных данных, а не p_g.

Задача:

$$\min_G \max_D V(D, G) = \mathop{\mathbb{E}}_{x \sim p_{data}} [\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z} [\log (1 - D(G(z)))]$$

Алгоритм обучения

1. Получаем мини-батч $z_1, ..., z_m$ из распределения p_z,
2. Получаем мини-батч $x_1, ..., x_m$ из распределения p_{data}
3. Обновляем дискриминатор в сторону возрастания его градиента

$$d_w \leftarrow \nabla_{\gamma_d} \frac{1}{m} \sum_{t=1}^{m} [\log D(x_t)] + [\log (1 - D(G(z_t)))]$$

4. Повторяем шаги 1-3 k раз.
5. Получаем мини-батч $z_1, ..., z_m$ из распределения p_z
6. Обновляем генератор в сторону убывания его градиента

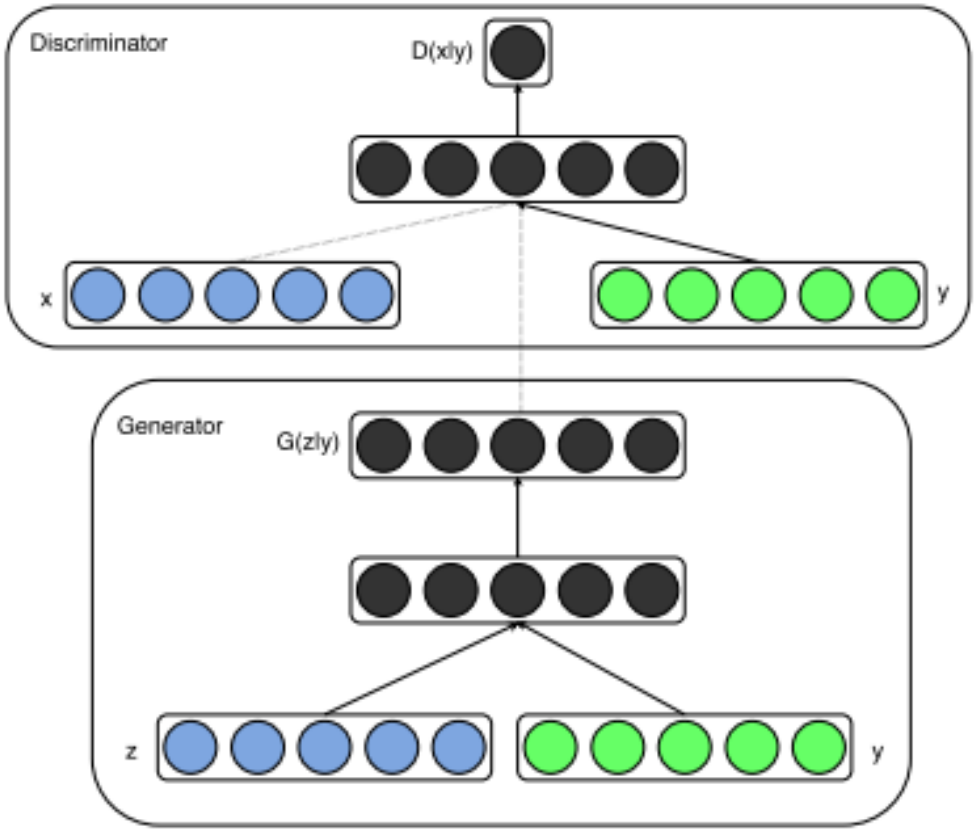
$$g_w \leftarrow \nabla_{\gamma_d} \frac{1}{m} \sum_{t=1}^{m} [\log (1 - D(G(z_t)))]$$

Проблемы обучения GAN

- Генератор выдает ограниченное количество разных образцов.
- Параметры модели дестабилизируются и не сходятся.
- Дискриминатор становится слишком сильным, а градиент генератора исчезает и обучение не происходит.
- Выявление корреляции в признаках, не связанных (слабо связанных) в реальном мире.
- Высокая чувствительность к гиперпараметрам.

y — Дополнительное условие для генератора и дискриминатора (Метка класса, изображение или данные из других моделей)

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x \sim p_{data}} [\log D(x|y)] + \mathop{\mathbb{E}}_{z \sim p_z} [\log (1 - D(G(z|y)))]$$



StackGAN, Stage-I

- Conditioning Augmentation — $\mathcal{N}\left(\mu(\phi_t), \Sigma(\phi_t)\right)$, где t — текстовое описание, а ϕ_t — векторное представление
- Регуляризация: $r = D_{KL}\left(\mathcal{N}\left(\mu(\phi_t), \Sigma(\phi_t)\right) || \mathcal{N}\left(0, I\right)\right)$
- Тренировка дискриминатора D_0 и генератора G_0:

  $L_{D_0} = \mathbb{E}_{(I_0,t)\sim p_{data}}\left[\log D_0\left(I_0, \phi_t\right)\right] +$
  $E_{z\sim t, t\sim p_{data}}\left[\log\left(1 - D_0\left(G_0(z, \hat{c_0}), \phi_t\right)\right)\right]$

  $L_{G_0} = \mathbb{E}_{z\sim t, t\sim p_{data}}\left[\log\left(1 - D_0\left(G_0(z, \hat{c_0}), \phi_t\right)\right)\right] + \lambda r,$

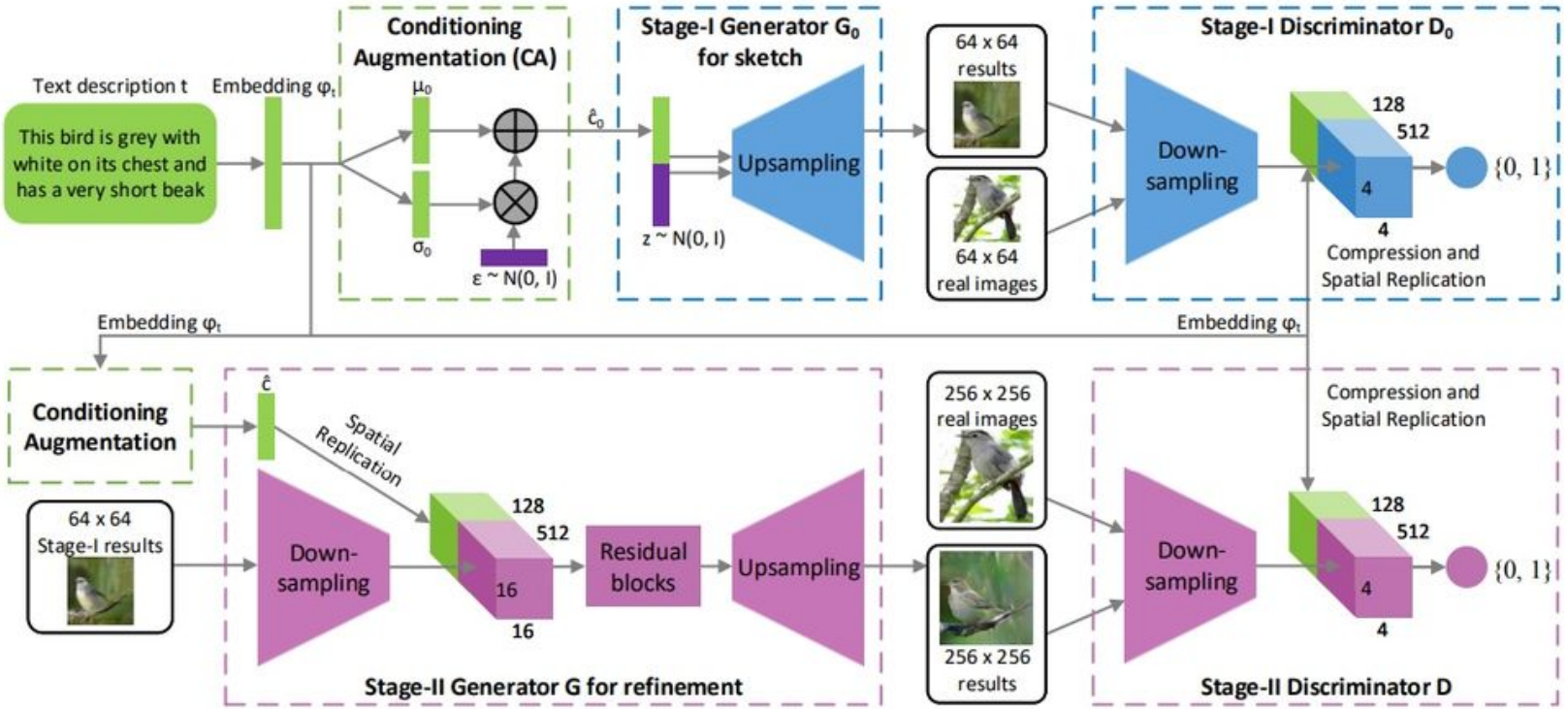
  где реальное изображение I_0 и описание текста t берутся из реального распределения данных p_{data}, z — шумовой вектор.

StackGAN, Stage-II

$$L_D = \mathbb{E}_{(I,t)\sim p_{data}}[\log D(I, \phi_t)] + \mathbb{E}_{s_0\sim p_{C_0}, t\sim p_{data}}[\log(1 - D(G(s_0, \hat{c}), \phi_t))]$$

$$L_G = \mathbb{E}_{s_0\sim p_{C_0}, t\sim p_{data}}\left[\log\left(1 - D\left(G(s_0, \hat{c}), \phi_t\right)\right)\right] + \lambda r,$$

где $s_0 = G_0\left(z, \hat{c}_0\right)$ — результат работы генератора Stage-I GAN.



LAPGAN

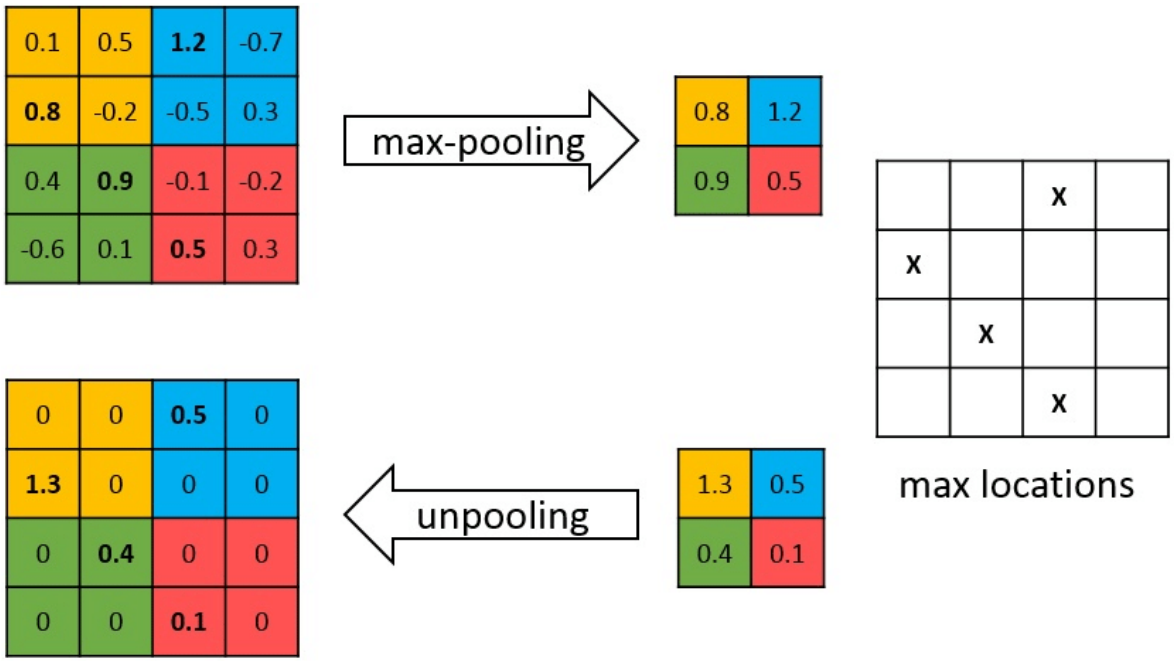
- Пусть $d(\cdot)$ — операция сжатия изображения размера $j \times j$ так, что новое изображение $d(I)$ имеет размеры $j/2 \times j/2$
- $u(\cdot)$ — операция расширения такая, что $u(I)$ имеет размеры $2j \times 2j$.

Тогда пирамида гауссианов имеет вид $\mathcal{G}(I) = [I_0, I_1, \ldots, I_k]$, где $I_0 = I$, и I_k представляет собой k раз выполненное применение $d(\cdot)$.
Коэффициенты h_k на каждом уровне пирамиды:

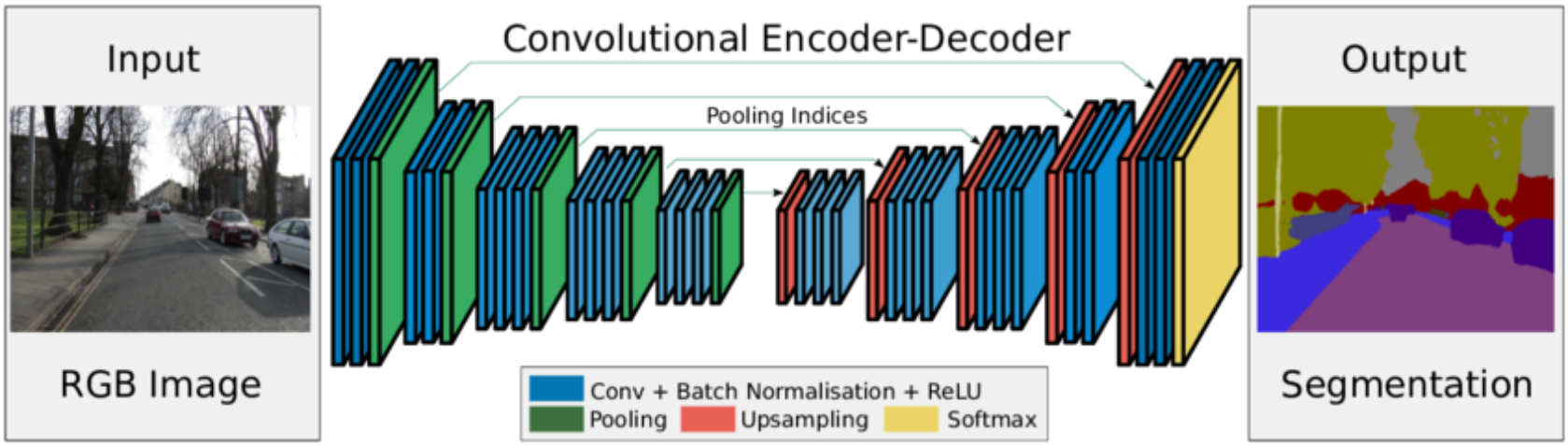
$$h_k = \mathcal{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = I_k - u\left(I_{k+1}\right)$$

Unpooling

Сохраняем позиции, где достигается максимум и используем их в unpooling слое.



Segnet


Input
RGB Image
Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Output
Segmentation

Vanilla Unet