# Лекция 9

- Как работает сверточная сеть
- Модификациии сверток
- Пример применения из астрофизики частиц
- Генеративные модели

# Повторение: глубокие нейронные сети, оптимизация

Проблемы:

- обнуление градиентов весов для начальных слоев
- ошибка округления
- большое число оптимизируемых весов

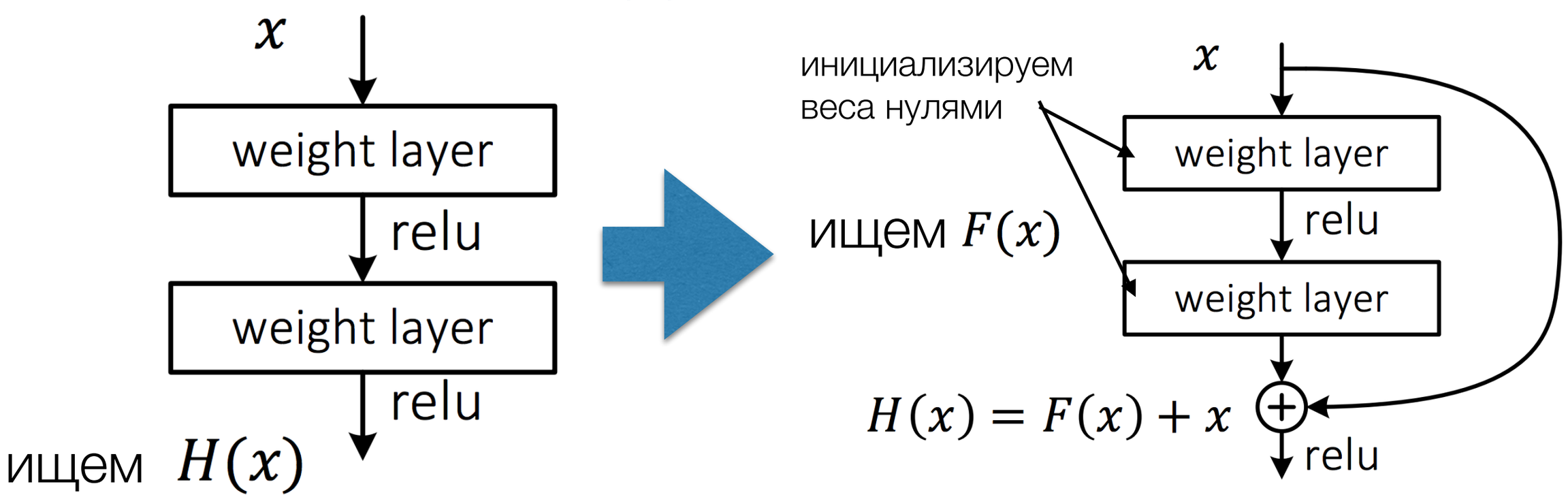
Решения:

- Альтернативные функции активации
- Остаточные (residual) слои
- Уменьшение числа весов за счет учета симметрии задачи
  - сверточные сети

# Повторение: глубокие нейронные сети, оптимизация

1. Альтернативные функции активации
2. Остаточные (residual) слои
3. Уменьшение числа весов за счет учета симметрии задачи

Пусть есть готовая обученная сеть *x* и мы хотим ее улучшить, добавив два дополнительных промежуточных слоя. Итоговая сеть *H(x)*:




plain net

ResNet Kaiming He et al 2015

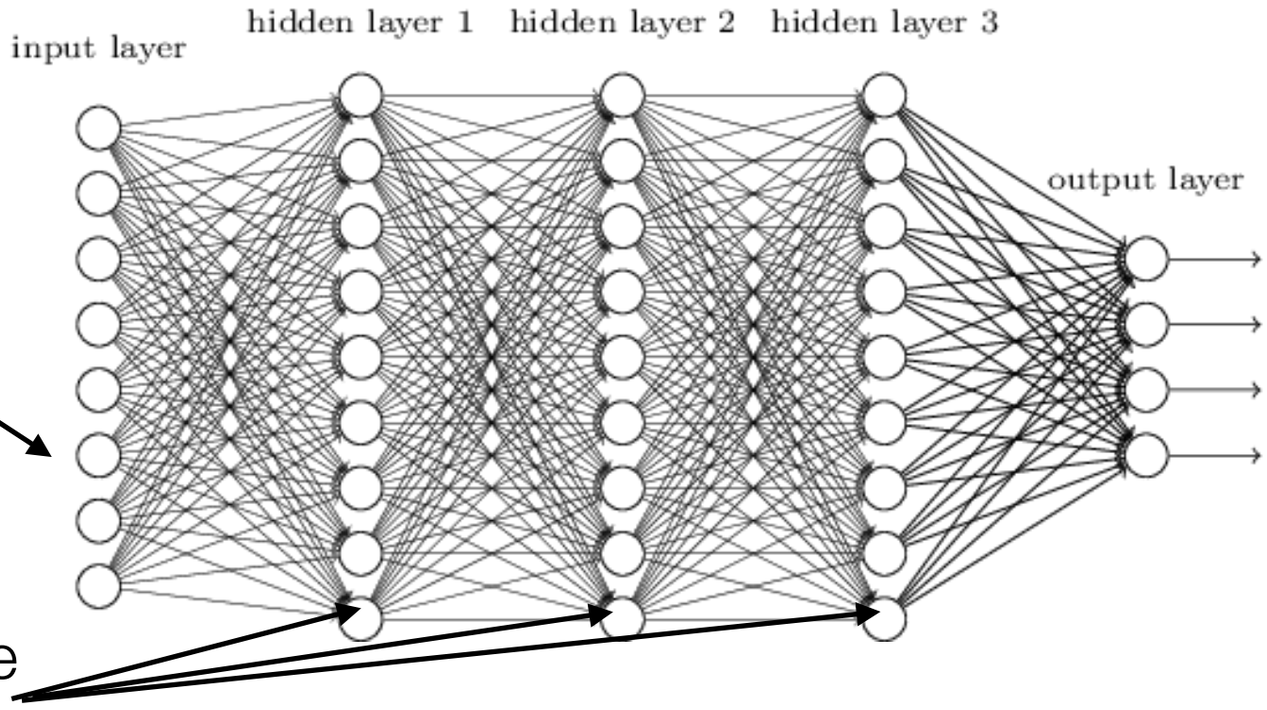
# Учет симметрии.

## Самый яркий пример - сверточные сети

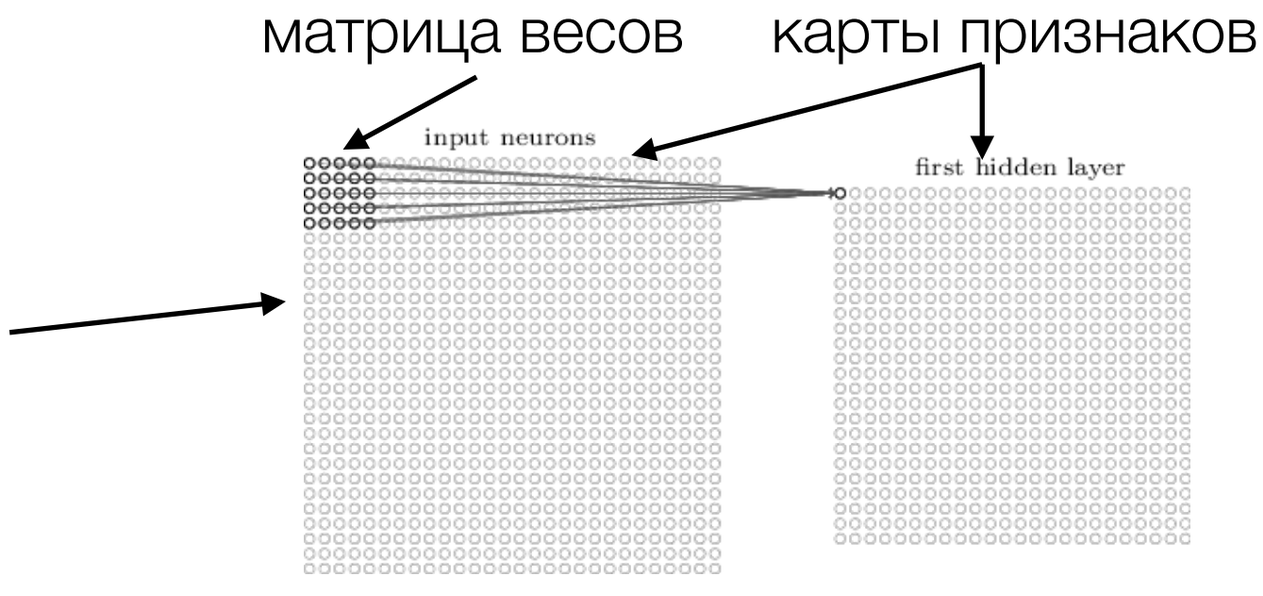
применяются для анализа изображений, временных рядов и т.п..

важное свойство: трансляционная инвариантность (см. дальше)

*MLP*




*Convnet*



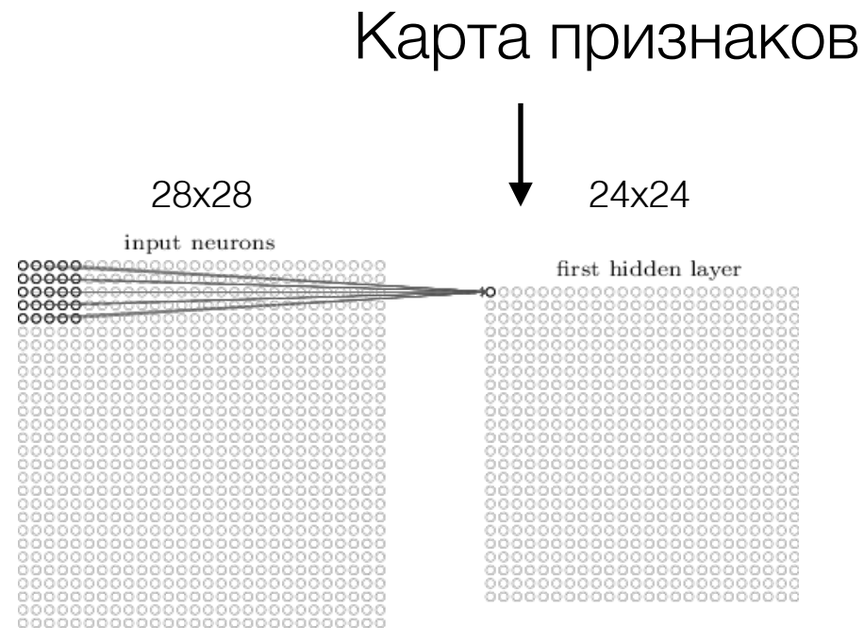

- нейрон скрытого слоя связан только с группой соседних нейронов внешнего слоя
- матрица весов (ядро фильтра) совпадает для всех нейронов

# Повторение: сверточные слои

- Математически эквивалентен полносвязанному слою где большинство весов нули, а остальные повторяются
- Число независимых весов:

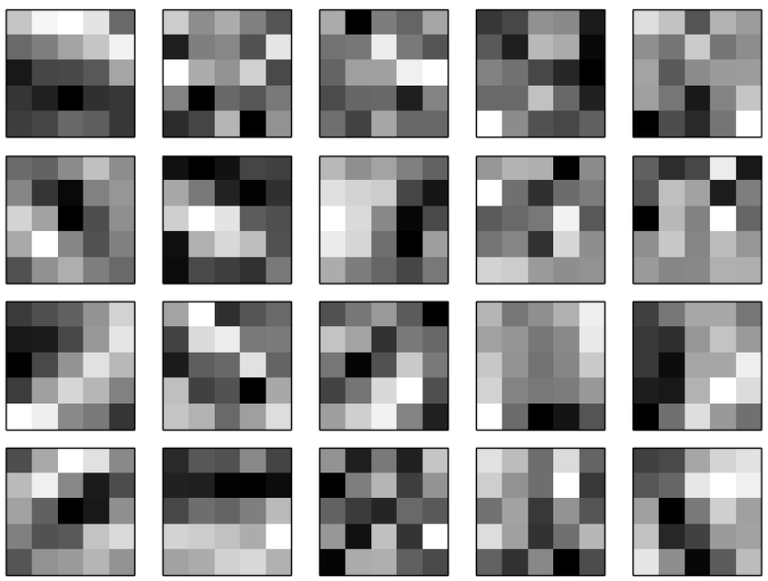
5x5x*N* против 28x28x24x24

*N* - число карт признаков




Примеры фильтров (MNIST):

(цвет кодирует вес)



Параметры сверточного слоя:

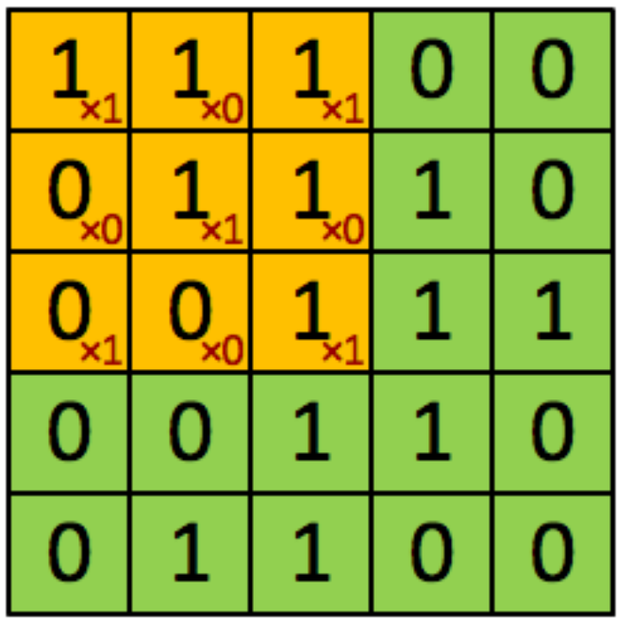
- Размер ядра
- Шаг (stride)
- Заполнение (padding): same/valid

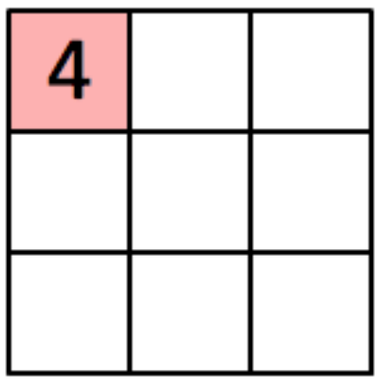
# Повторение: сверточные слои

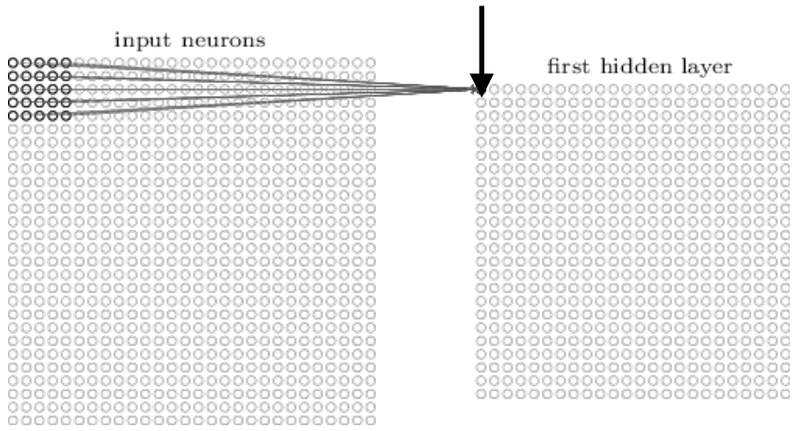
## Как это работает:

1.padding: ‘valid’ (без заполнения)

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$



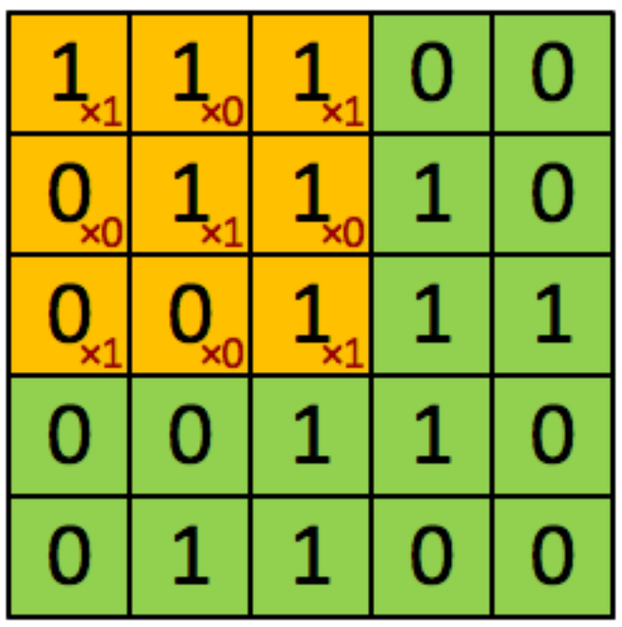



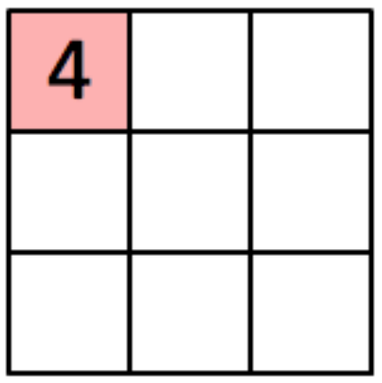

- размер карты признаков меньше исходного изображения
- пиксели на краях дают меньший вклад в карту признаков

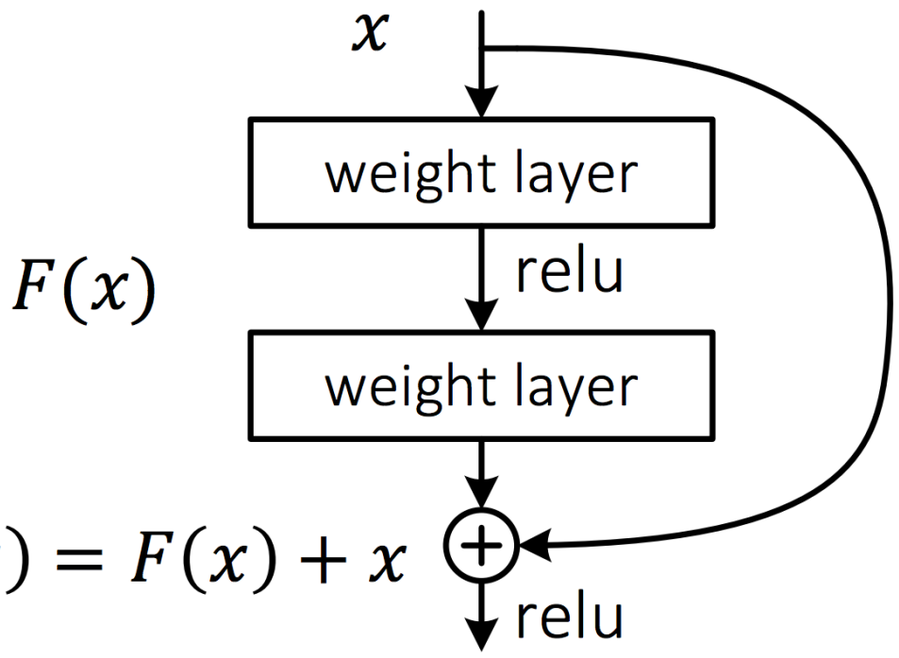
# Повторение: сверточные слои

Как это работает:

1.padding: ‘valid’ (без заполнения)



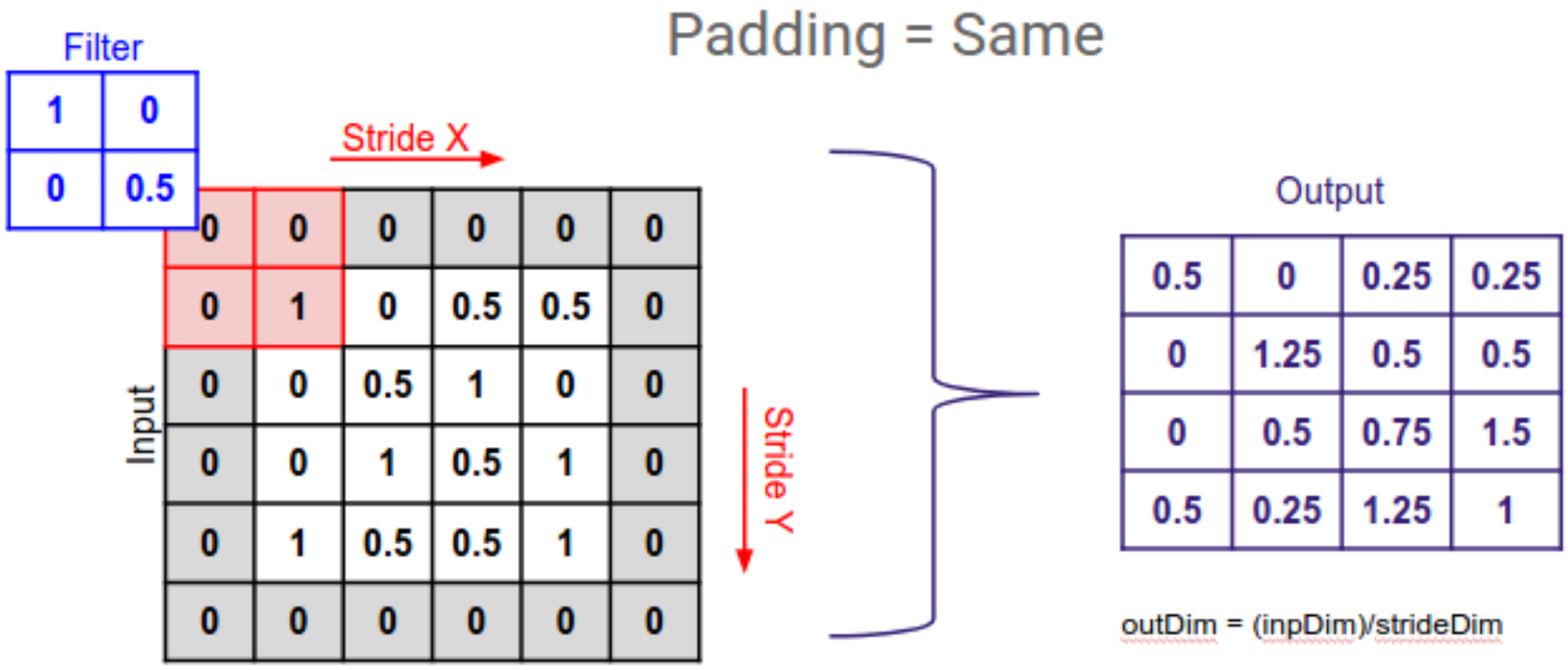

непригодно для остаточных сетей

- размер карты признаков меньше исходного изображения
- пиксели на краях дают меньший вклад в карту признаков

# Повторение: сверточные слои

Как это работает:

2.padding: ‘same’ (с заполнением)



# Повторение: сверточный слой

## обобщение на произвольное количество карт признаков в слое

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 156 | 155 | 156 | 158 | 158 | ... |
| 0 | 153 | 154 | 157 | 159 | 159 | ... |
| 0 | 149 | 151 | 155 | 158 | 159 | ... |
| 0 | 146 | 146 | 149 | 153 | 158 | ... |
| 0 | 145 | 143 | 143 | 148 | 158 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #1 (Red)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 167 | 166 | 167 | 169 | 169 | ... |
| 0 | 164 | 165 | 168 | 170 | 170 | ... |
| 0 | 160 | 162 | 166 | 169 | 170 | ... |
| 0 | 156 | 156 | 159 | 163 | 168 | ... |
| 0 | 155 | 153 | 153 | 158 | 168 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #2 (Green)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | ... |
| 0 | 163 | 162 | 163 | 165 | 165 | ... |
| 0 | 160 | 161 | 164 | 166 | 166 | ... |
| 0 | 156 | 158 | 162 | 165 | 166 | ... |
| 0 | 155 | 155 | 158 | 162 | 167 | ... |
| 0 | 154 | 152 | 152 | 157 | 167 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Input Channel #3 (Blue)

| | | |
|---|---|---|
| -1 | -1 | 1 |
| 0 | 1 | -1 |
| 0 | 1 | 1 |

Kernel Channel #1

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | -1 | -1 |
| 1 | 0 | -1 |

Kernel Channel #2

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | -1 | 1 |

Kernel Channel #3

$308 + (-498) + 164 + 1 = -25$

Bias = 1

Output

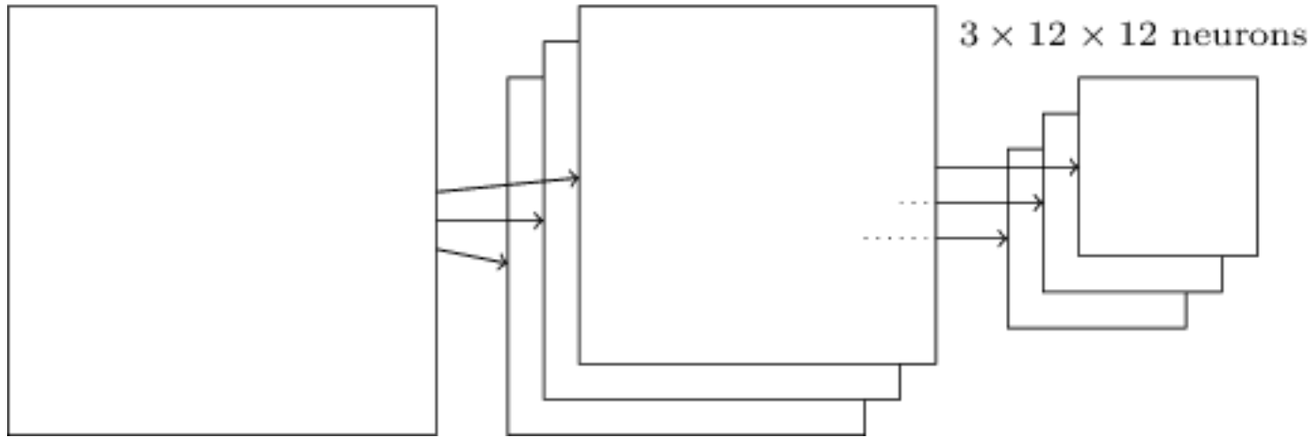
| | | | | |
|---|---|---|---|---|
| -25 | | | | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| ... | ... | ... | ... | ... |

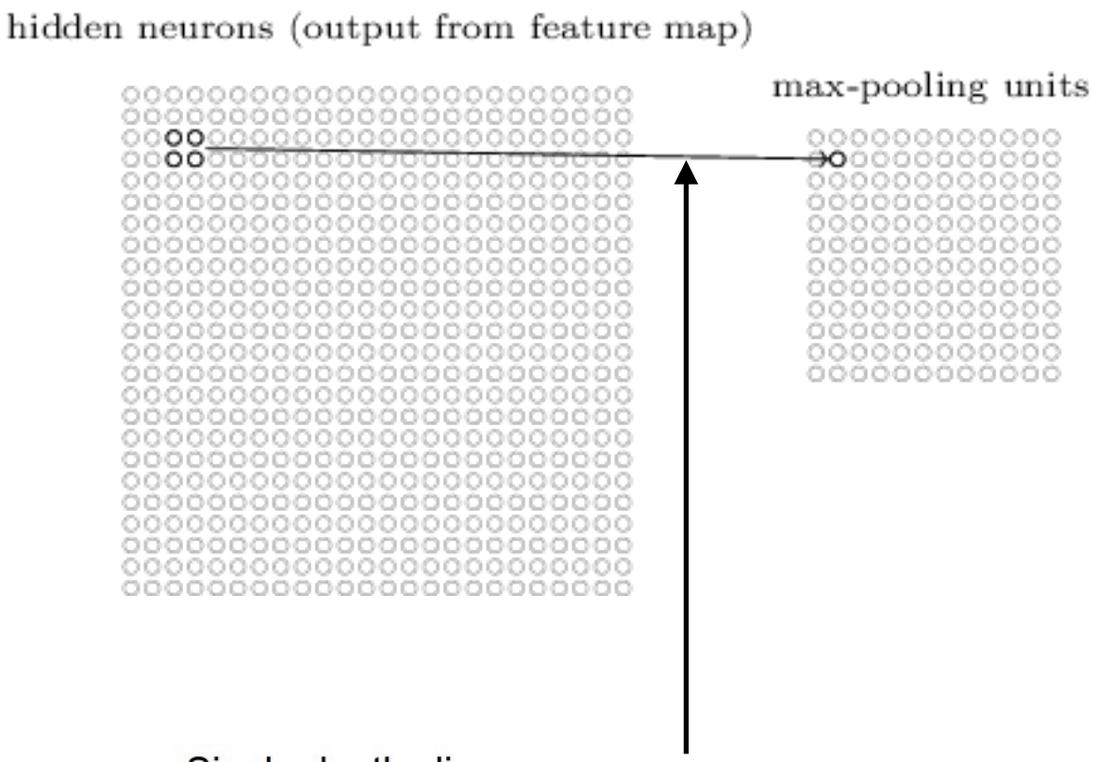
$$a_{d,m,n}^{l+1} = \sigma\left(b_d + \sum_{c,\alpha,\beta} W_{dc\alpha\beta} a_{c,m\delta_1+\alpha,n\delta_2+\beta}^{l}\right)$$

$c, d$ - индексы карт $\quad \delta_1$- шаг (stride) по вертикали $\quad \delta_2$ - шаг по горизонтали

# Масштабирующий слой

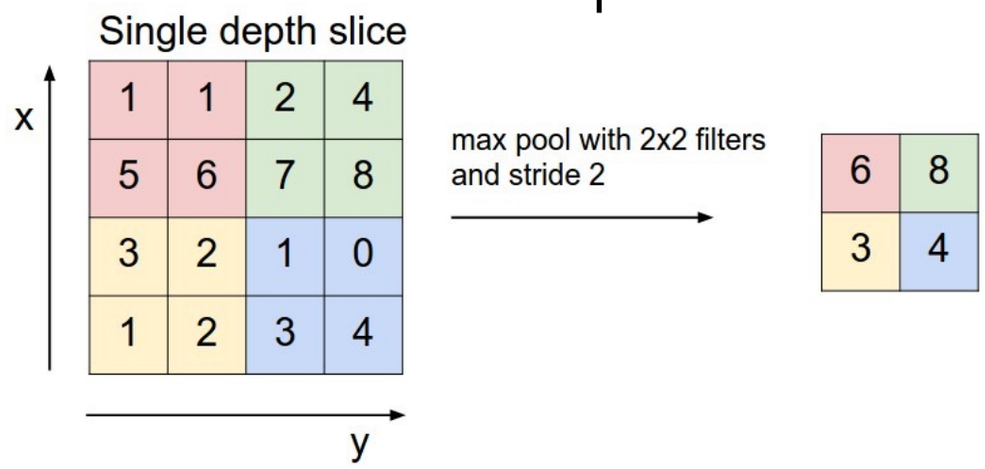
## Извлечение признаки на больших масштабах



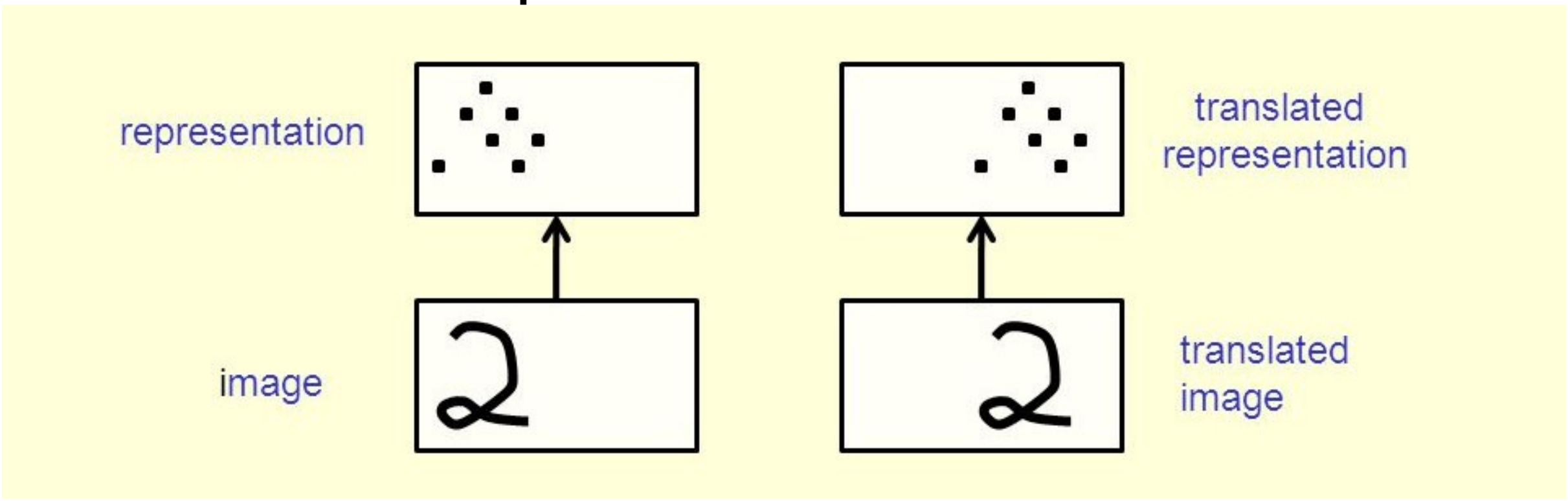




метод:

- максимум
- среднее
- среднеквадратичное
- …



# Трансляционная инвариантность сверточных сетей

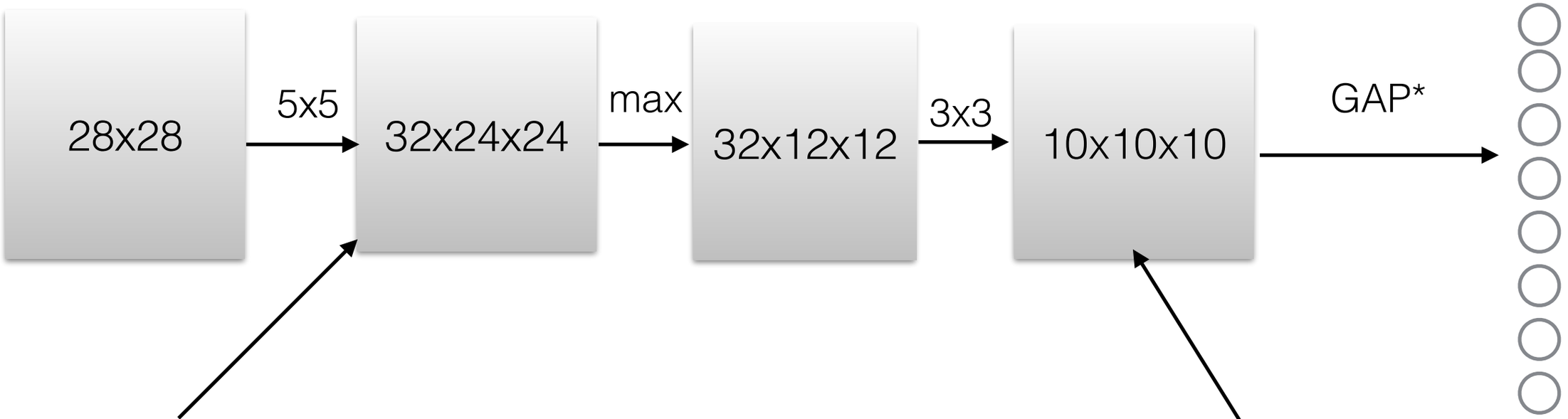



- Операция свертки коммутирует с операцией сдвига
- Результат применения последовательности операций свертки и масштабирования (MaxPool) инвариантен относительно небольших сдвигов изображения

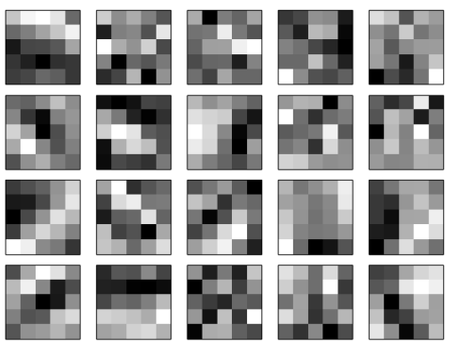
Задача: продемонстрировать трансляционную инвариантность на примерах MNIST. От чего зависит максимально допустимый сдвиг?

# Как увидеть, на каких признаках обучается сверточная нейронная сеть
# Задача

- Обучить сверточную нейронную сеть для распознавания рукописных цифр




Визуализировать фильтры



Для одной цифры из тестового набора, визуализировать активации соответствующего слоя, построив тепловую карту**. Смасштабировать карту до 28x28 и наложить ее на исходное изображение

*GAP- Global average pooling. Реализация в keras: GlobalAveragePooling2D

**Тепловая карта на seaborn: https://seaborn.pydata.org/generated/seaborn.heatmap.html

пример решения в google colab

# Оптимизация сверток

Свертка размером $(K,K)$ имеет $K^2 \times N_1 \times N_2$ весов

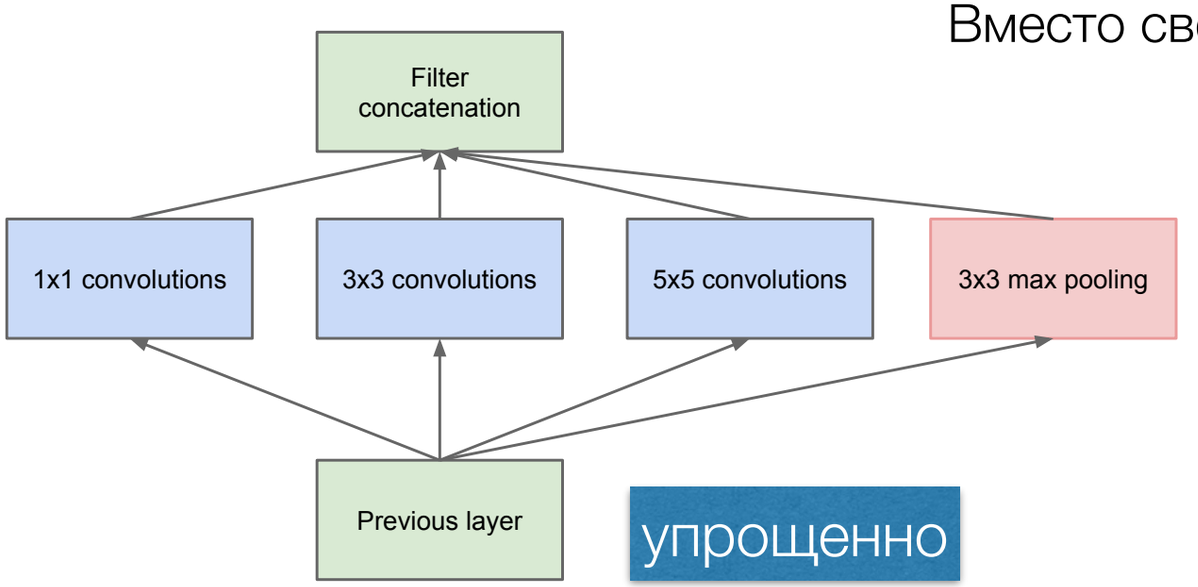
где $N_1,N_2$ - число карт признаков в предыдущем и текущем слоях (может быть большим)

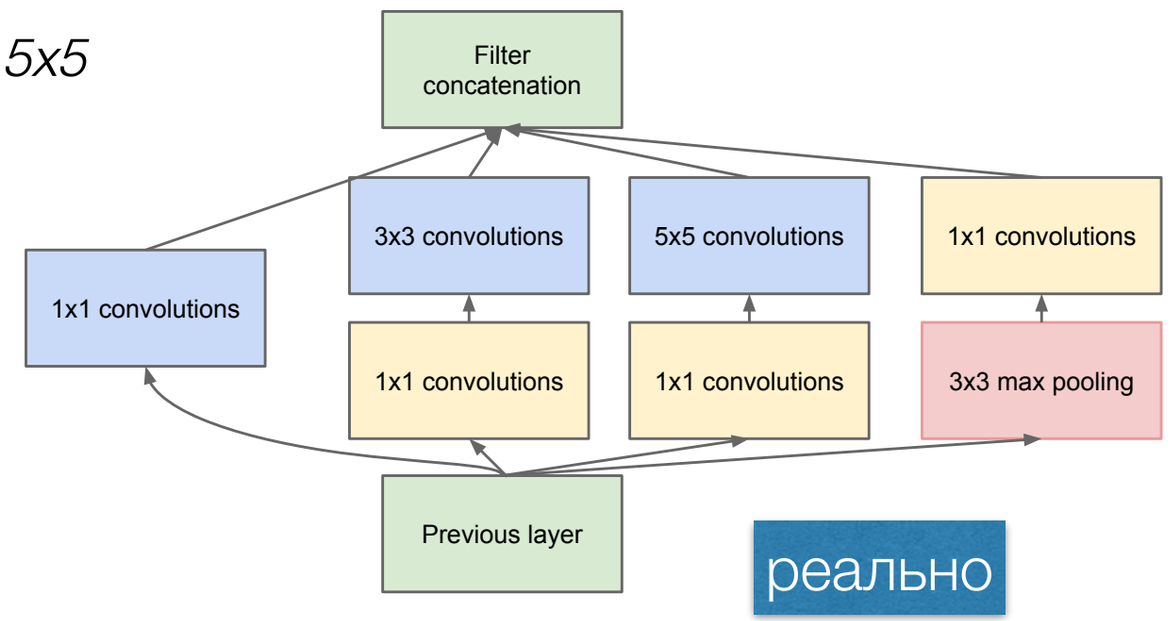
Архитектура ‘Inception’: Szegedy et al 2014, 2015

- прием 1: уменьшаем $N_2$

Разные признаки относятся к разным масштабам, для большинства признаков веса могут задаваться матрицей меньшего размера

Вместо свертки *5x5*



# Оптимизация сверток

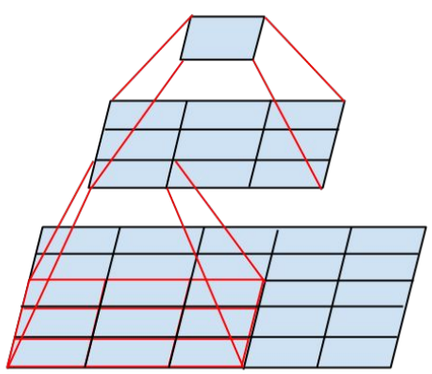
Свертка размером $(K,K)$ имеет $K^2 \times N_1 \times N_2$ весов

где $N_1,N_2$ - число карт признаков в предыдущем и текущем слоях

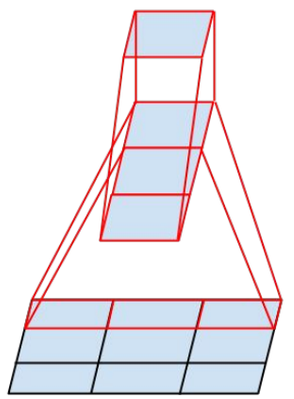
Архитектура ‘Inception’: Szegedy et al 2014, 2015

- прием 2: уменьшаем *K*

В реальных данных сигналы соседних нейронов коррелируют
Можно заменить большую свертку последовательностью небольших, что позволит уменьшить общее число весов



каков выигрыш в количестве весов?



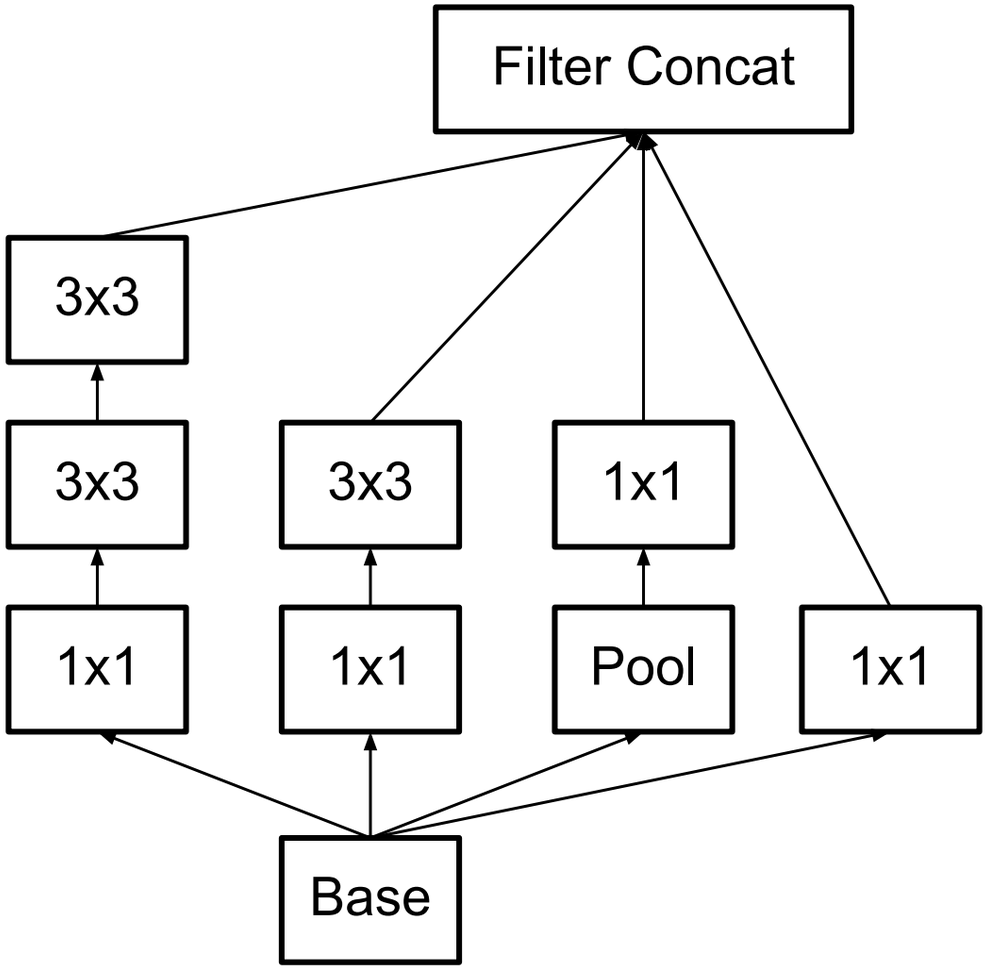
две свертки *3x3* вместо одной *5x5*

*3x1 и 1x3* вместо одной 3*x3*

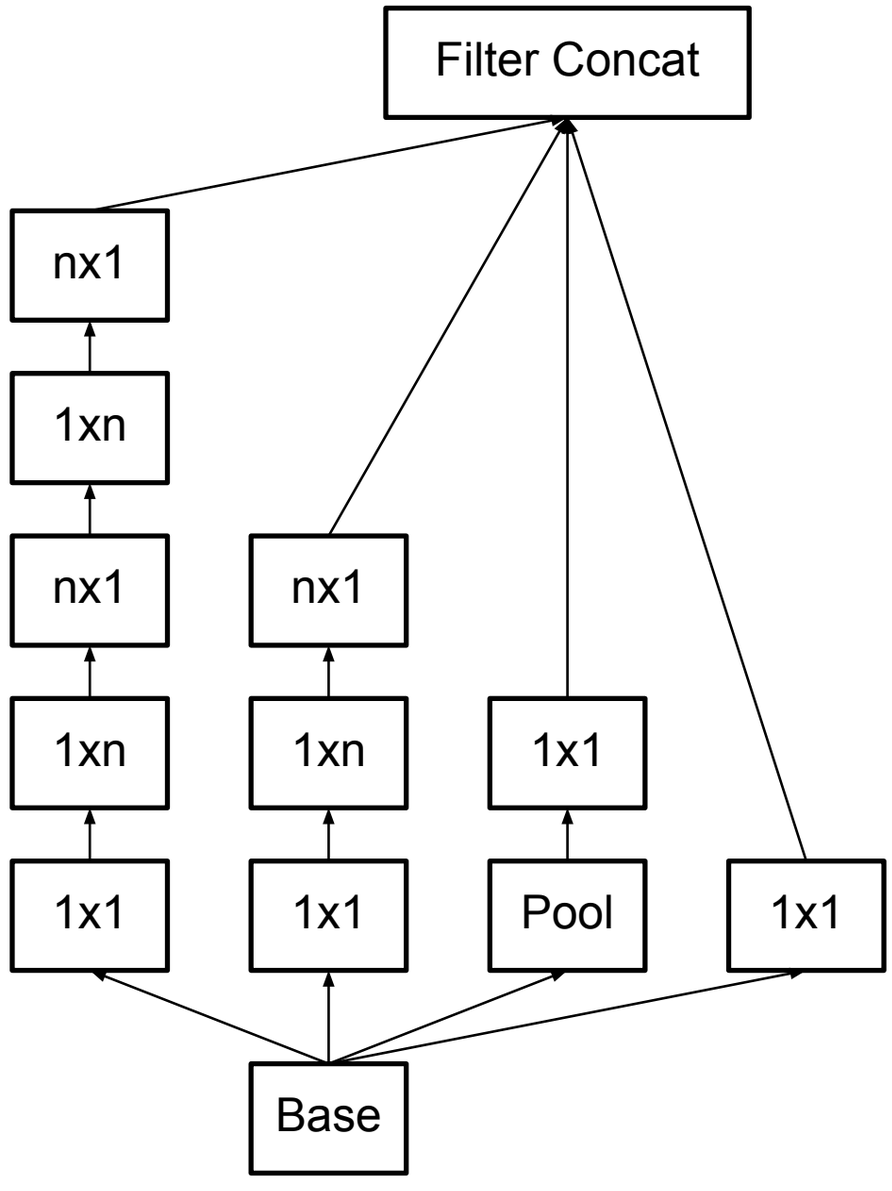
# Оптимизация сверток

Архитектура ‘Inception’: Szegedy et al 2014, 2015

Комбинируем приемы 1 и 2:




вместо $5 \times 5$

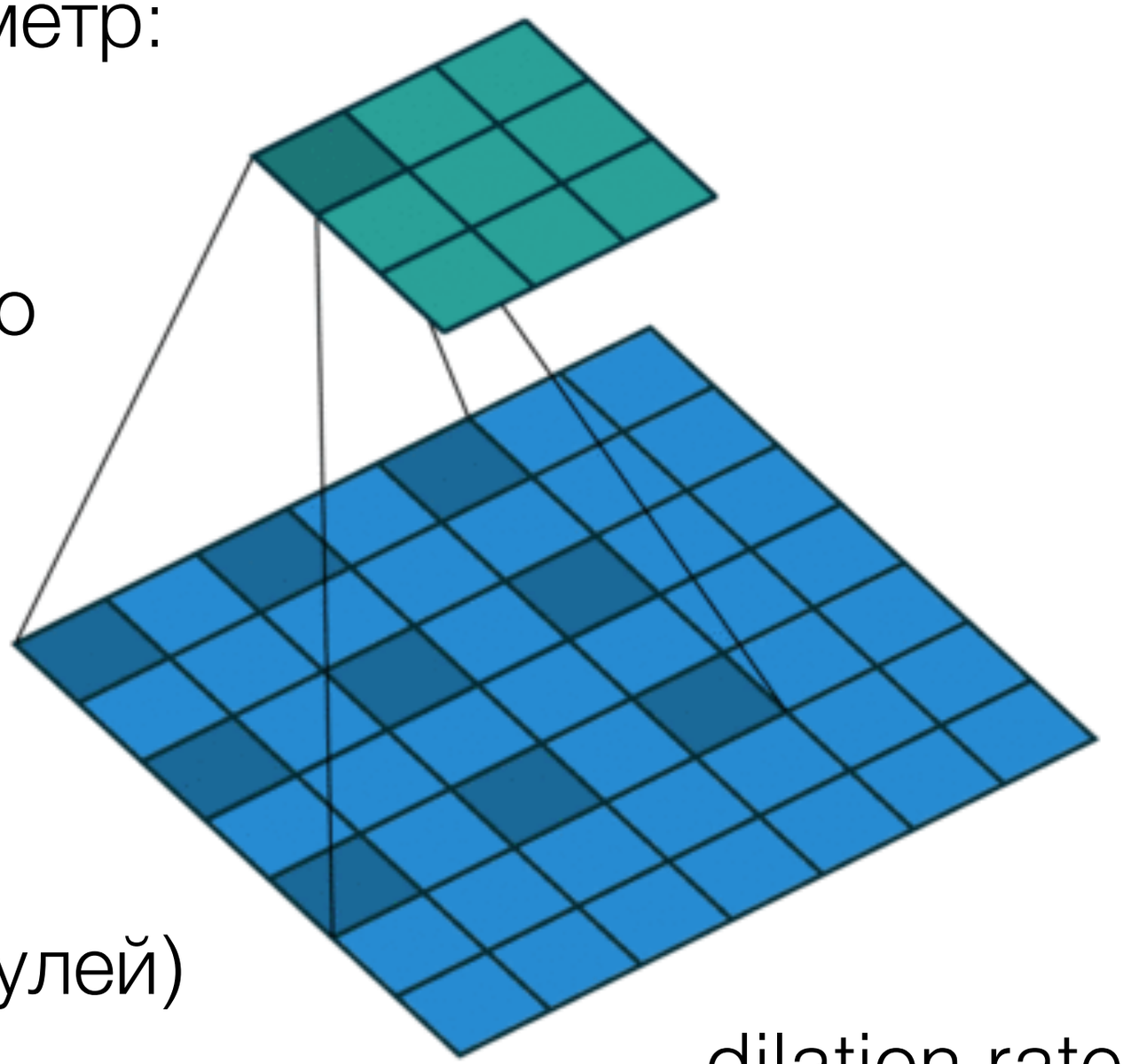



вместо $n \times n$, $n > 5$

# Свертки специального вида

## Разреженные (dilated)

- Дополнительный параметр: уровень разреживания (dilation rate) L
- Позволяют эффективно вычислять карты крупномасштабных признаков

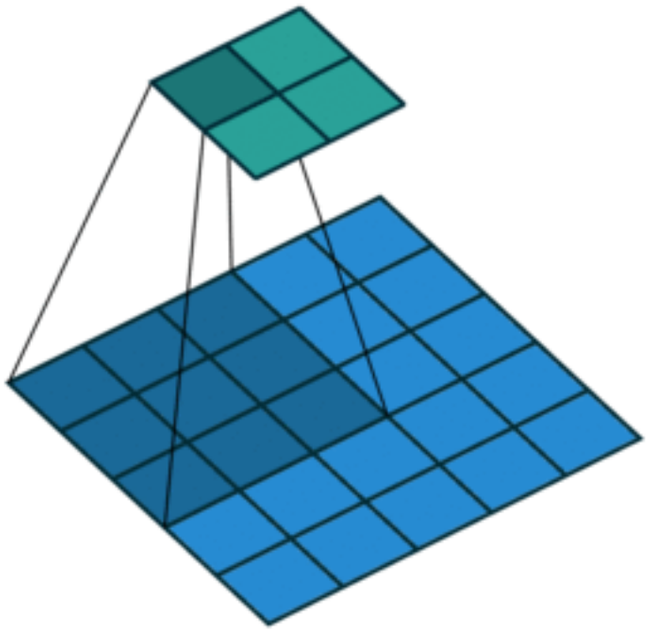


L-1 = число пробелов (нулей) между элементами ядра

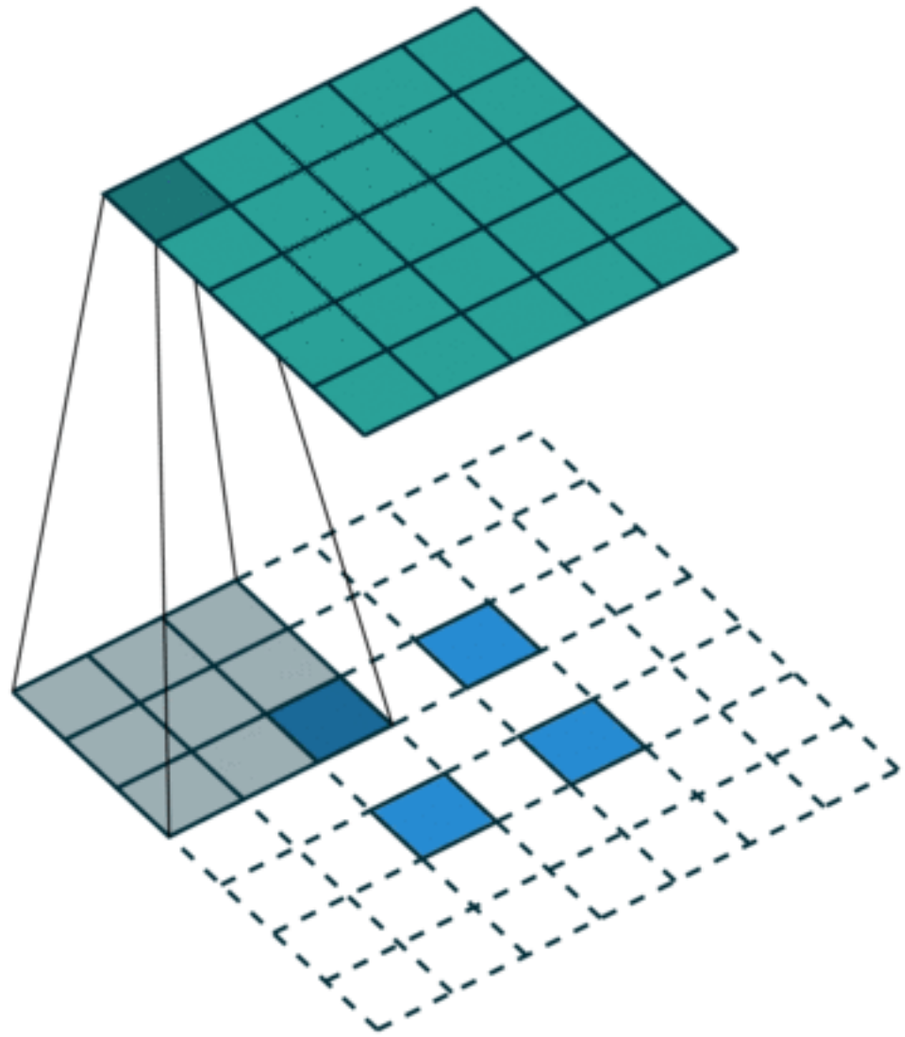
dilation rate = 2

# Свертки специального вида

## Транспонированные (transposed)



Обычная свертка
Padding: valid
Stride: 2



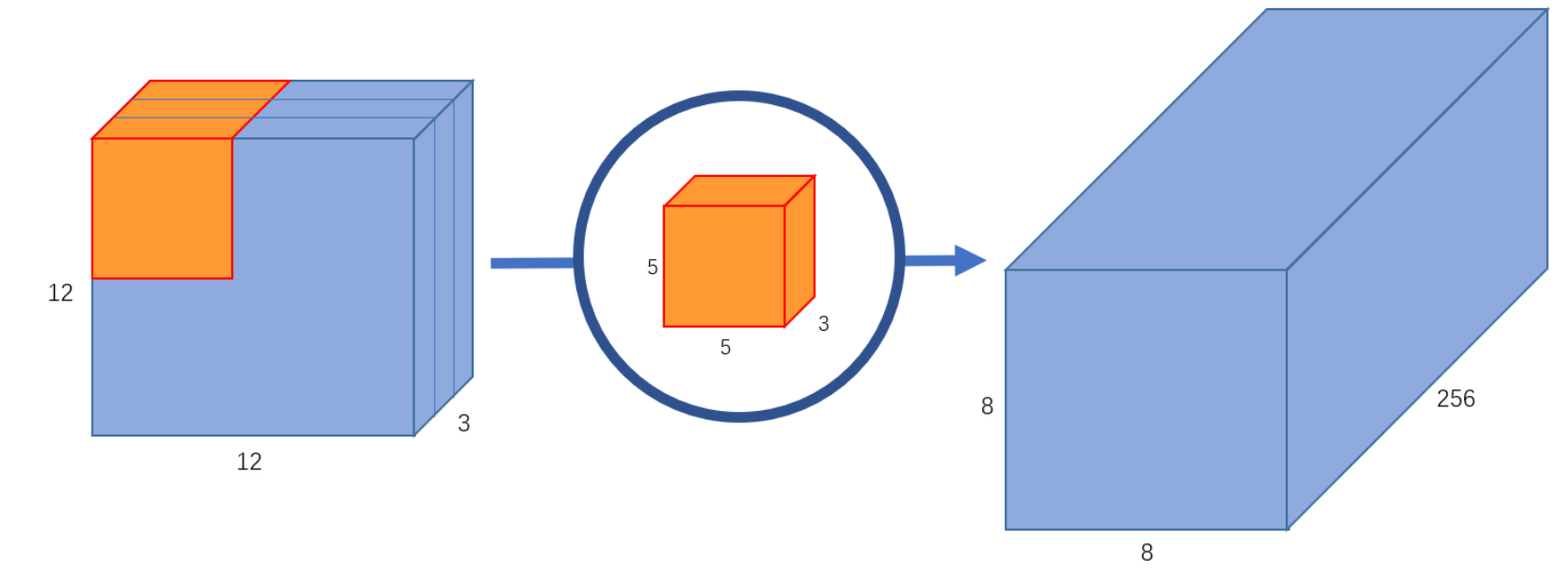
используется для увеличения размера карты признаков, например в генеративных моделях

# Свертки специального вида

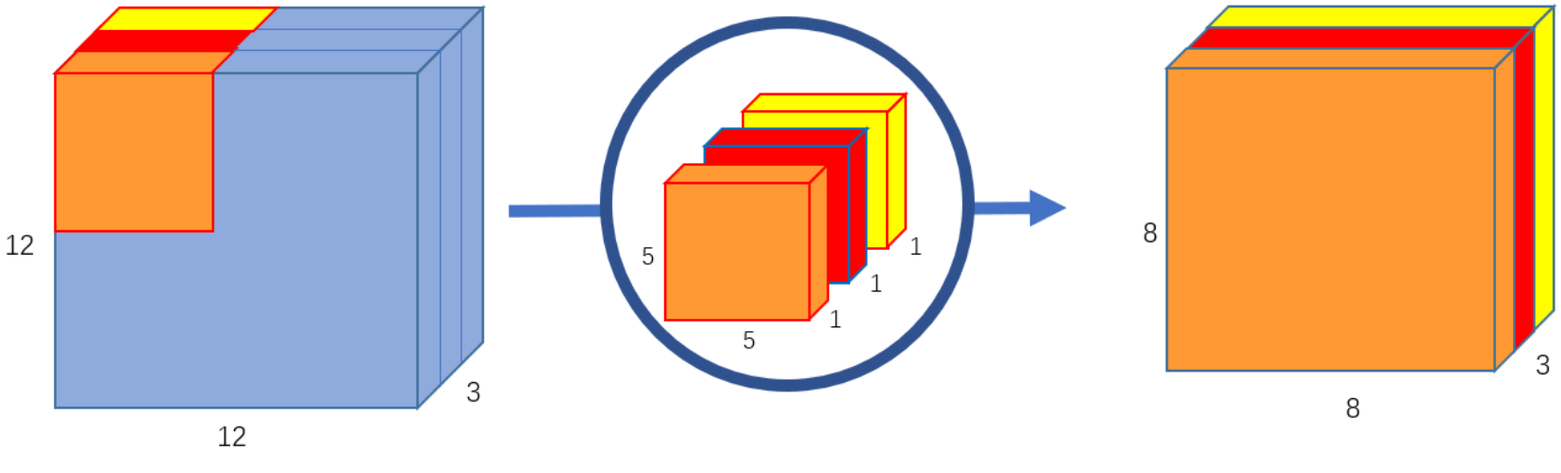
## Depthwise separable convolutions

разделимые по признакам

см. архитектура Xception (François Chollet, автор Keras)



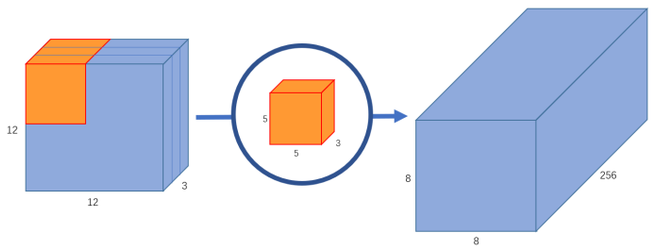

Ядро стандартной свертки
5 x 5 x 3 x 256



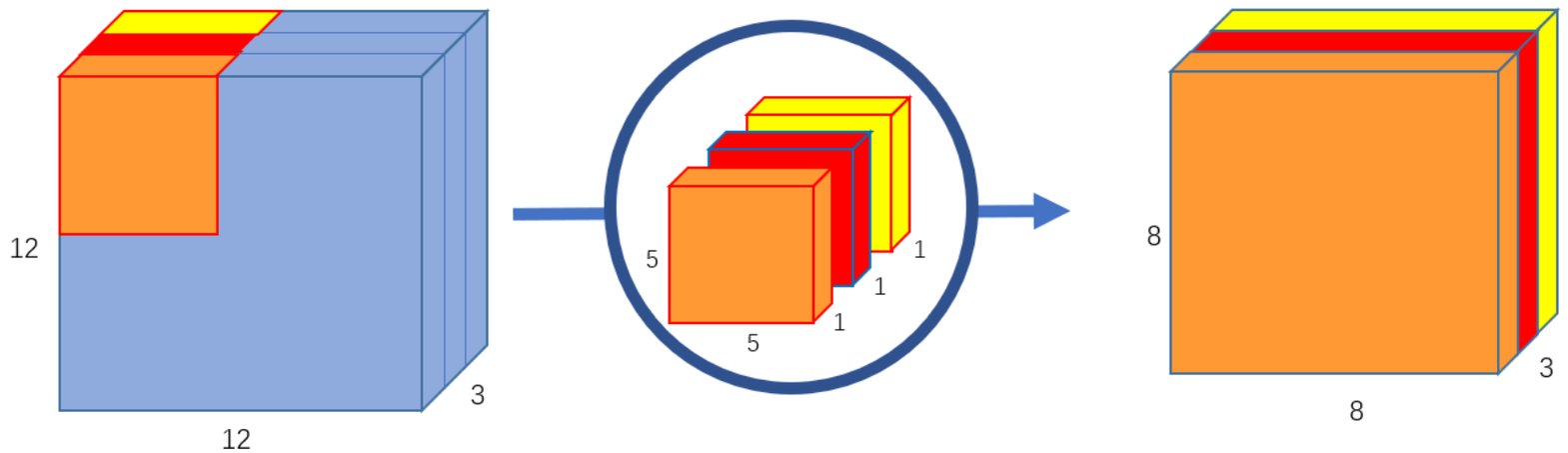

шаг 1
(пространственная свертка):
ядро 5 x 5 x 3

# Types of Convolutions in Deep Learning

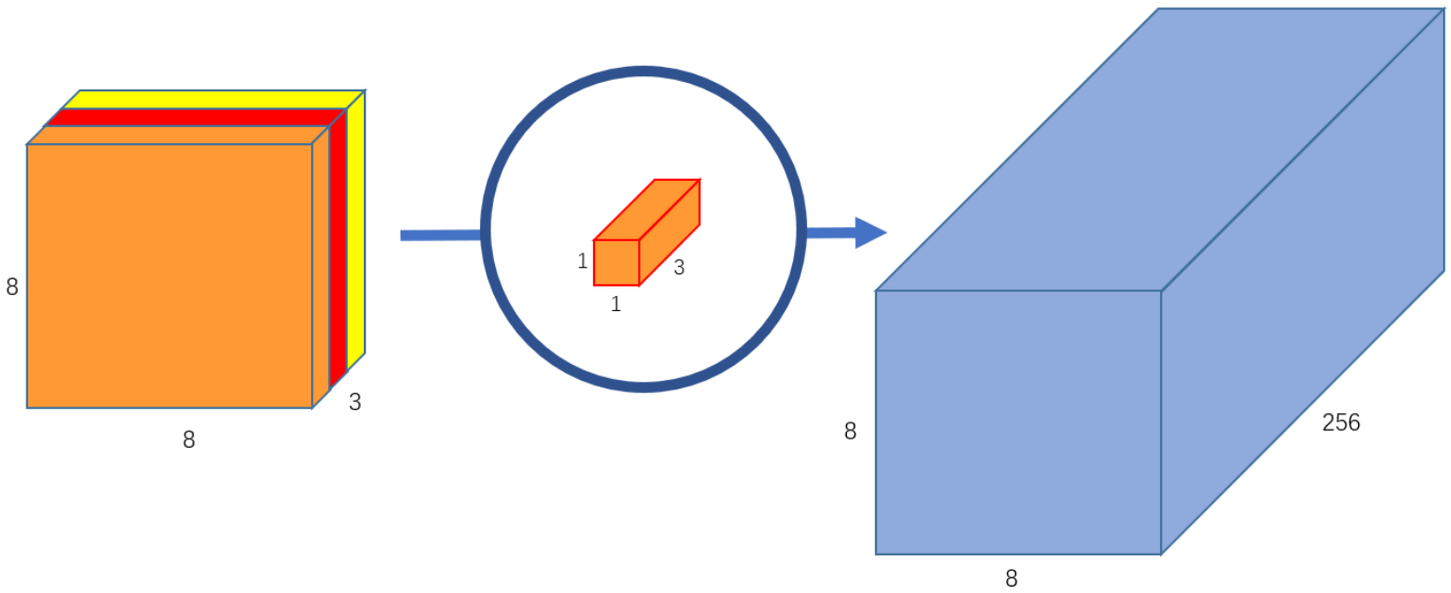
## Depthwise separable convolutions




Ядро стандартной свертки
5 x 5 x 3 x 256




шаг 1 (пространственная свертка): ядро 5 x 5 x 3



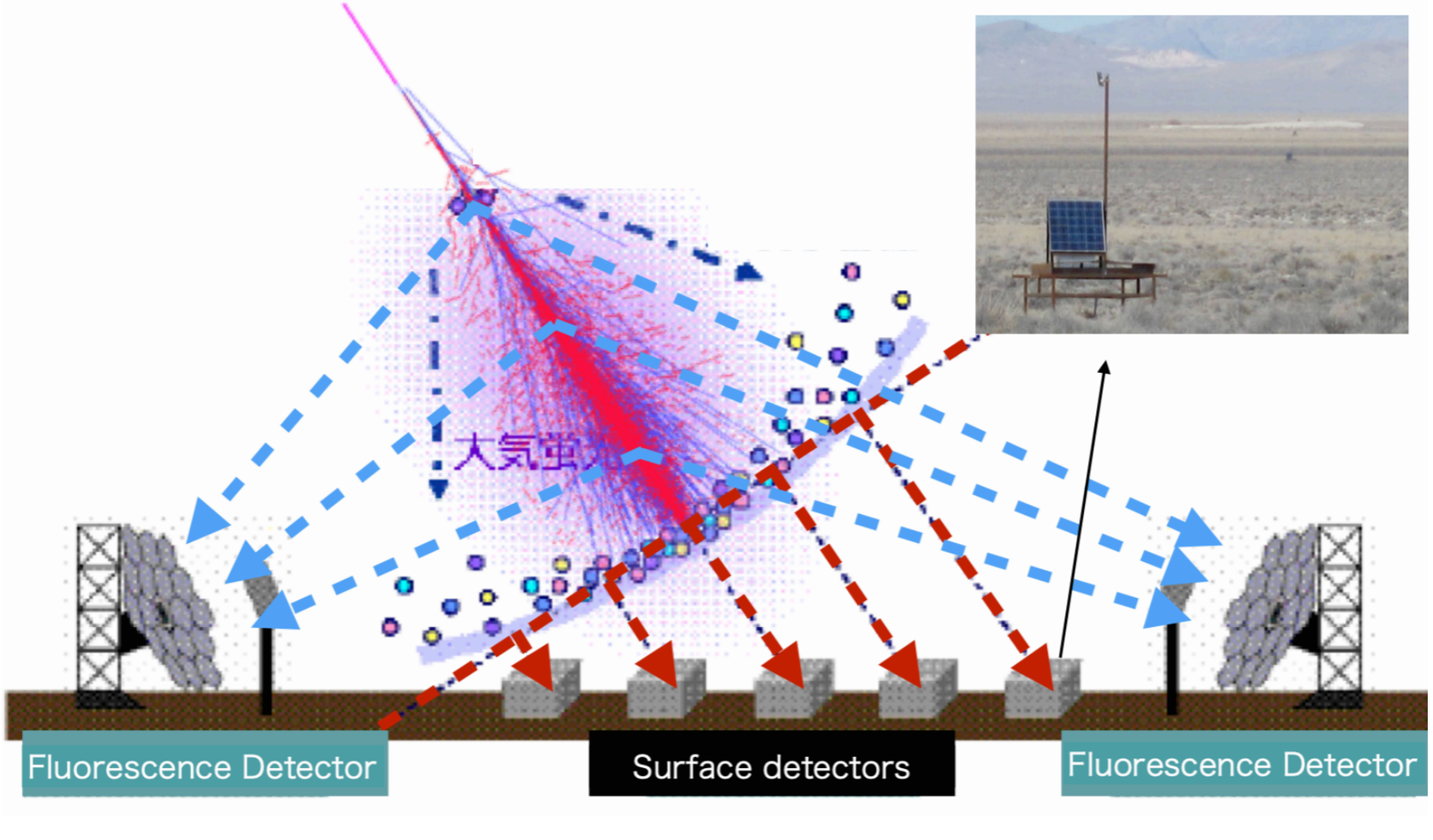

шаг 2:
ядро 1 x 1 x 3 x 256

Сравнение:

1228800 умножений против 3x5x5x8x8+256x1x1x3x8x8=53952

# Пример применения сверточных нейронных сетей в астрофизике частиц

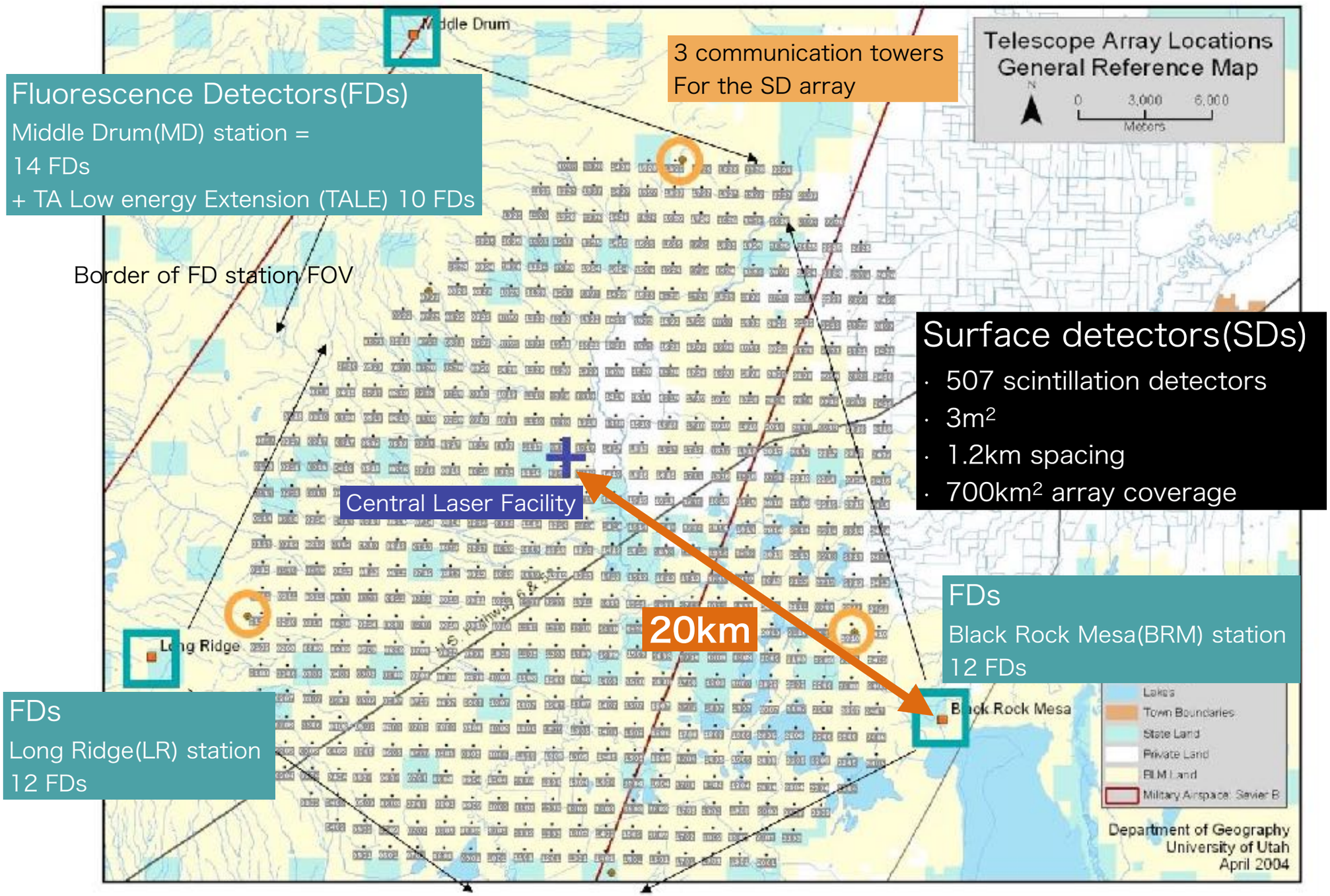
Реконструкция свойств первичных частиц по данным наземной решетки детекторов
Kalashev et al 2019, Ivanov et al 2020



# Telescope Array

- Самый большой эксперимент в северном полушарии (Юта, США).

USA,
Russia,
Japan,
Korea,
Belgium



# Аналитическая реконструкция

- LDF - эмпирическое распределение плотности ливня

$$f(r) = \left(\frac{r}{R_m}\right)^{-1.2} \left(1 + \frac{r}{R_m}\right)^{-(\eta - 1.2)} \left(1 + \frac{r^2}{R_1^2}\right)^{-0.6}$$

$$R_m = 90.0 \text{ m},\ R_1 = 1000 \text{ m},\ R_L = 30 \text{ m}, \quad \eta = 3.97 - 1.79\left(\sec(\theta) - 1\right),$$

$$r = \sqrt{(x_{\text{core}} - x)^2 + (y_{\text{core}} - y)^2},$$

- Timing - форма фронта ливня

$$t_r = t_o + t_{plane} + a \times (1 + r/R_L)^{1.5}\, LDF(r)^{-0.5}$$

$$LDF(r) = f(r)/f(800 \text{ m}) \qquad S(r) = S_{800} \times LDF(r)$$

*Свободные параметры:*

$x_{core},\ y_{core},\ \theta,\ \phi,\ S_{800},\ t_0,\ a$

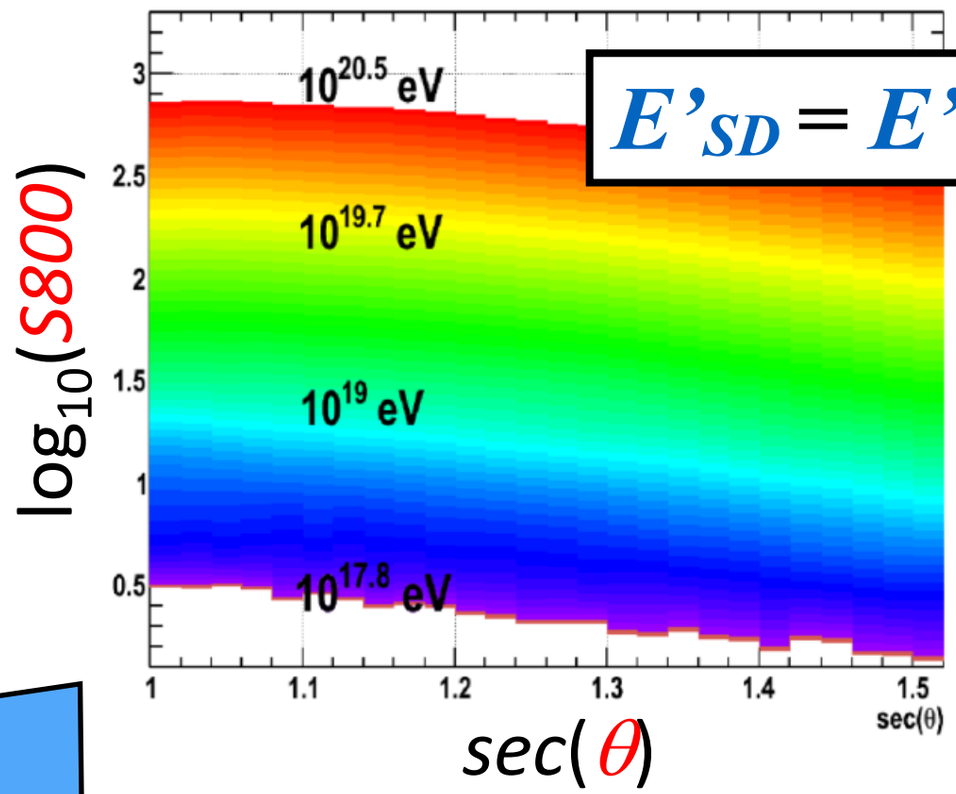
*Наблюдаемые:*

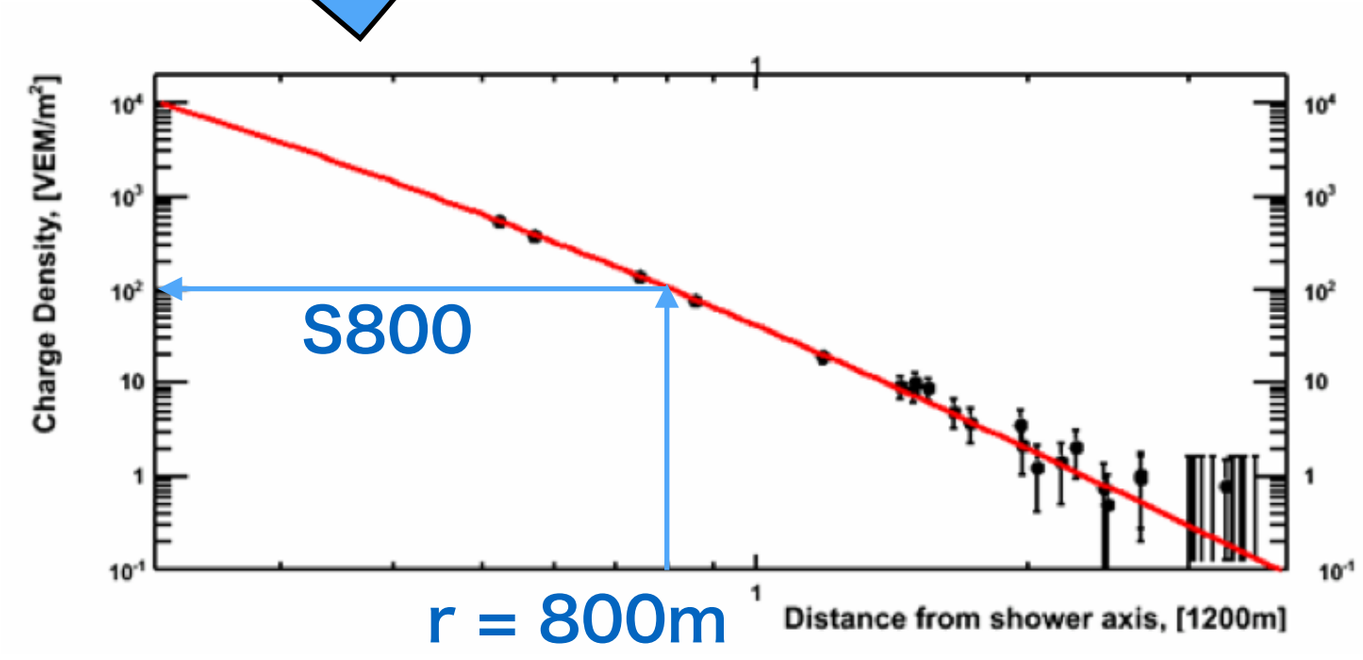
$t_r$ - время срабатывания детектора

$S_r$ - интегральный сигнал в детекторе

# Реконструкция энергии

*использует таблицу на основе Монте Карло моделирования*




S800 - плотность ливня на расстоянии 800 метров от оси

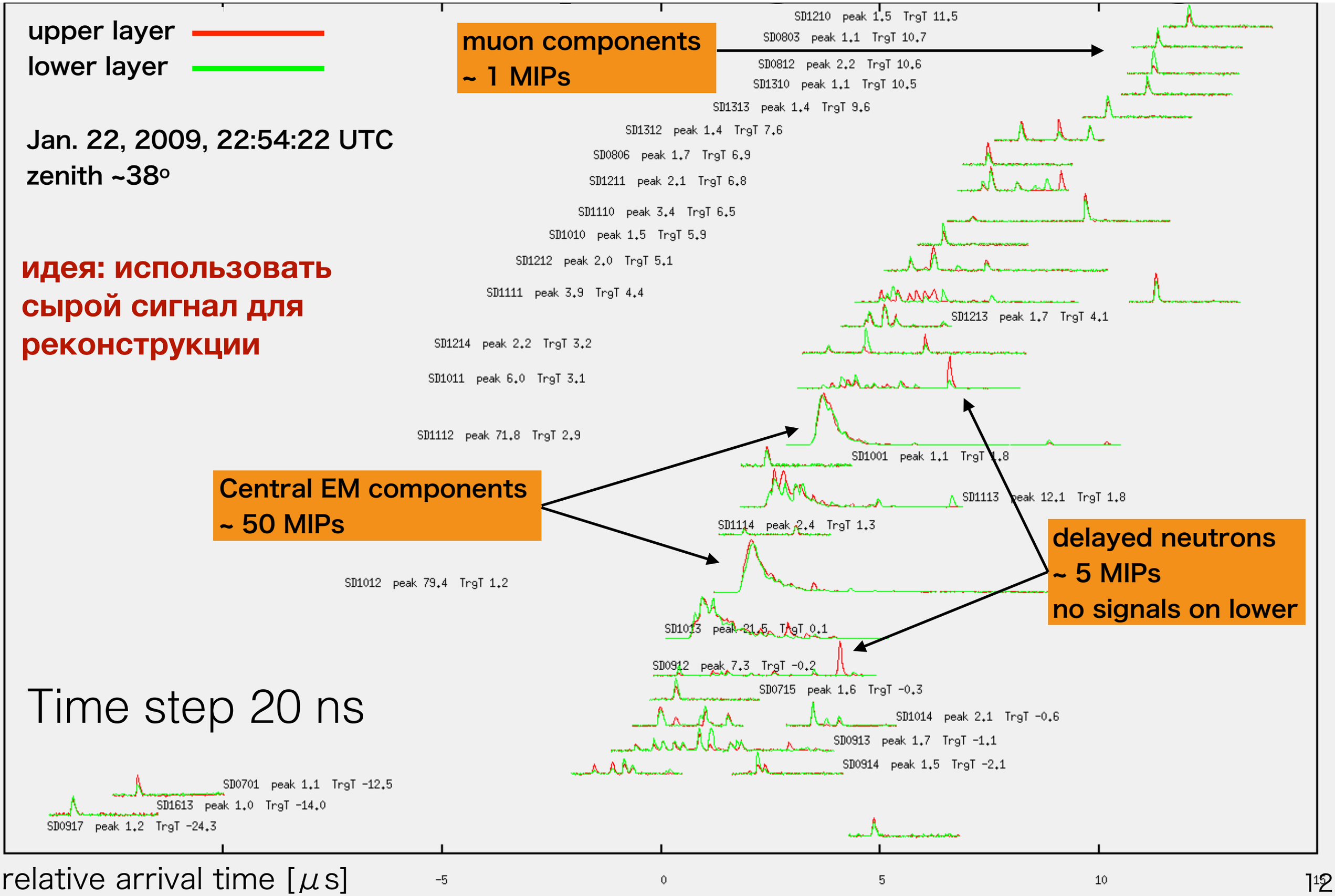
# Пример события

upper layer

lower layer

Jan. 22, 2009, 22:54:22 UTC
zenith ~38°

**идея: использовать сырой сигнал для реконструкции**



# Пример события

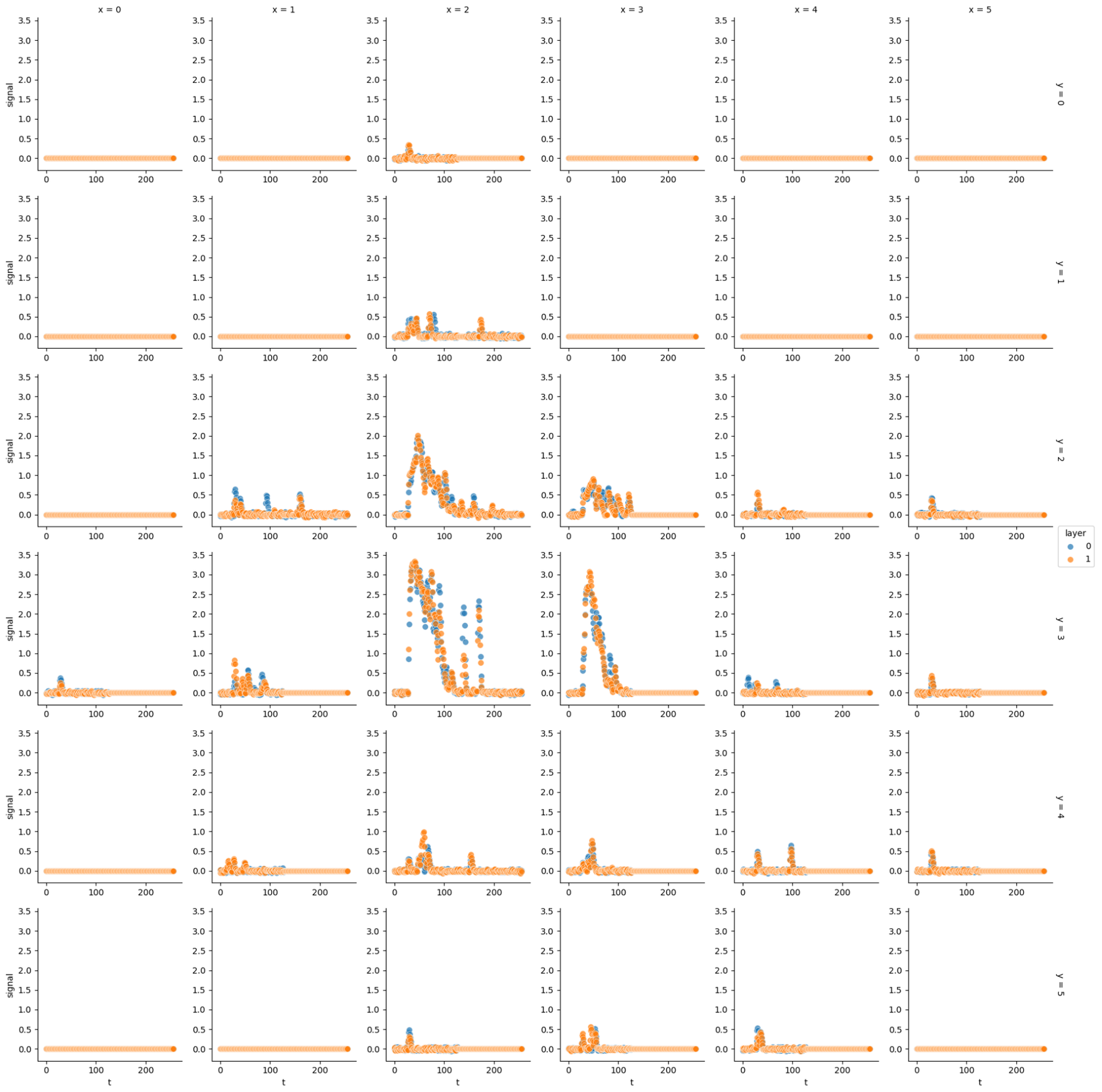
сырые данные

Размерность:
(N,N,T,2)

Waveform

detector layers

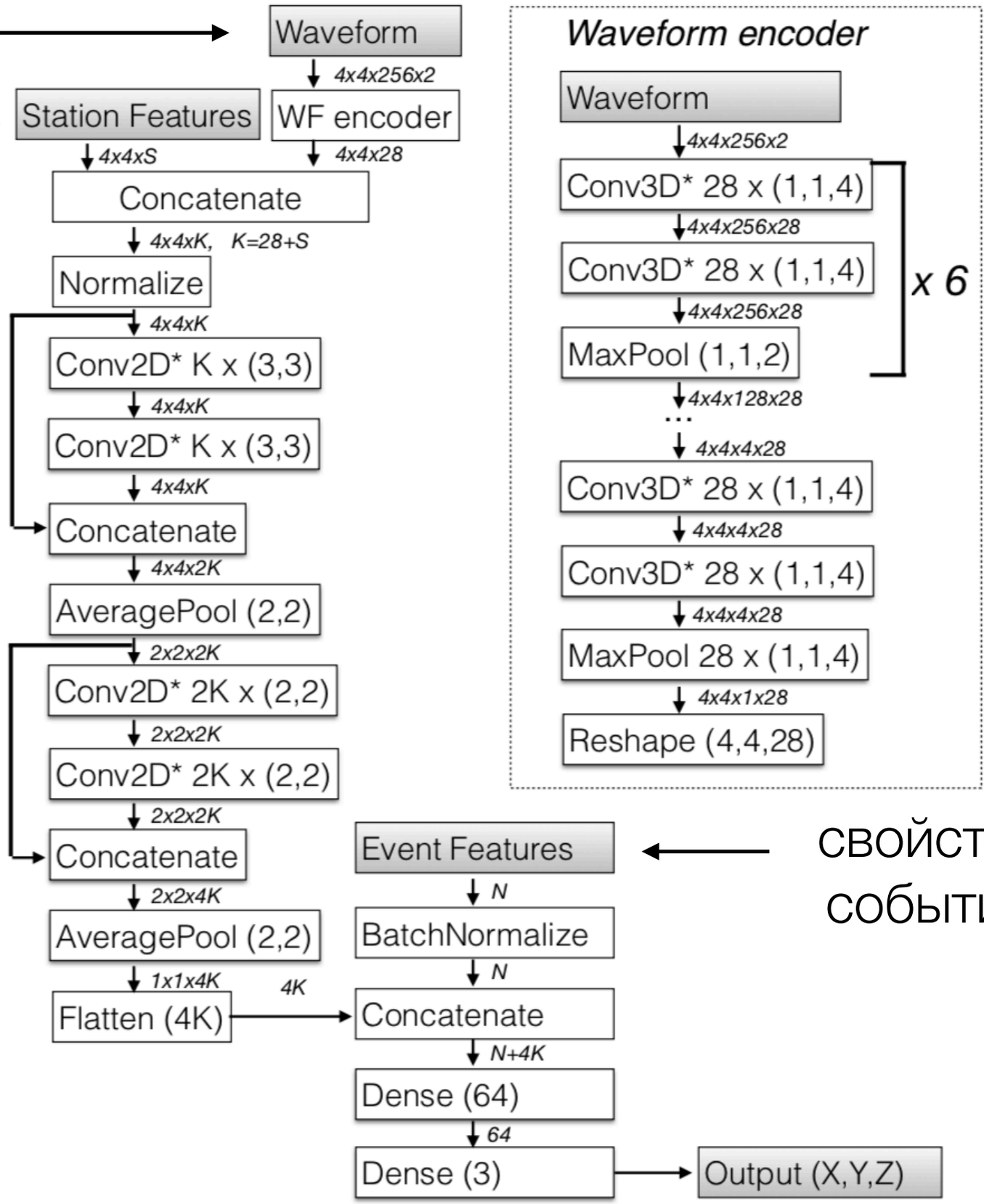
N=4-8, T=128-256



# Нейросетевая реконструкция

сырые данные детекторов

свойства детекторов




свойства событий

# Сравнение с аналитической реконструкцией

объясненная дисперсия
(explained variance)

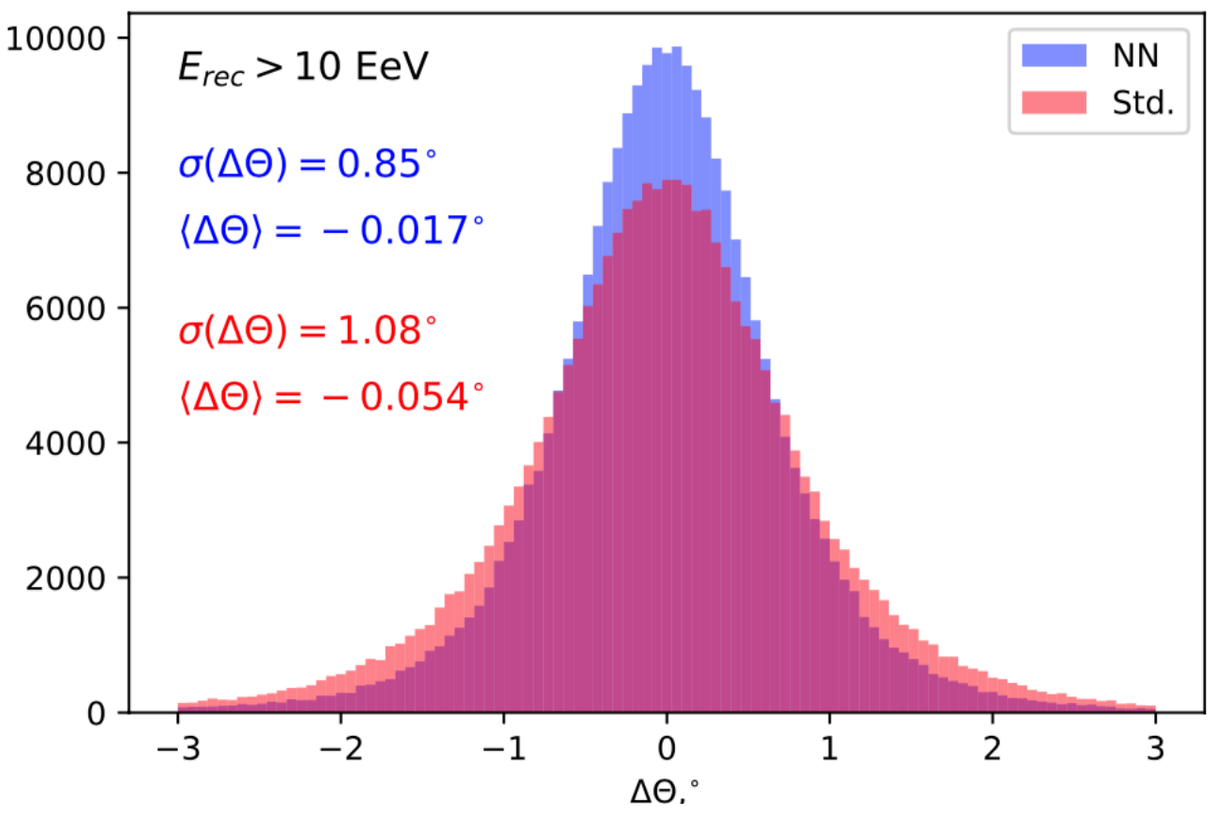
$$EV(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

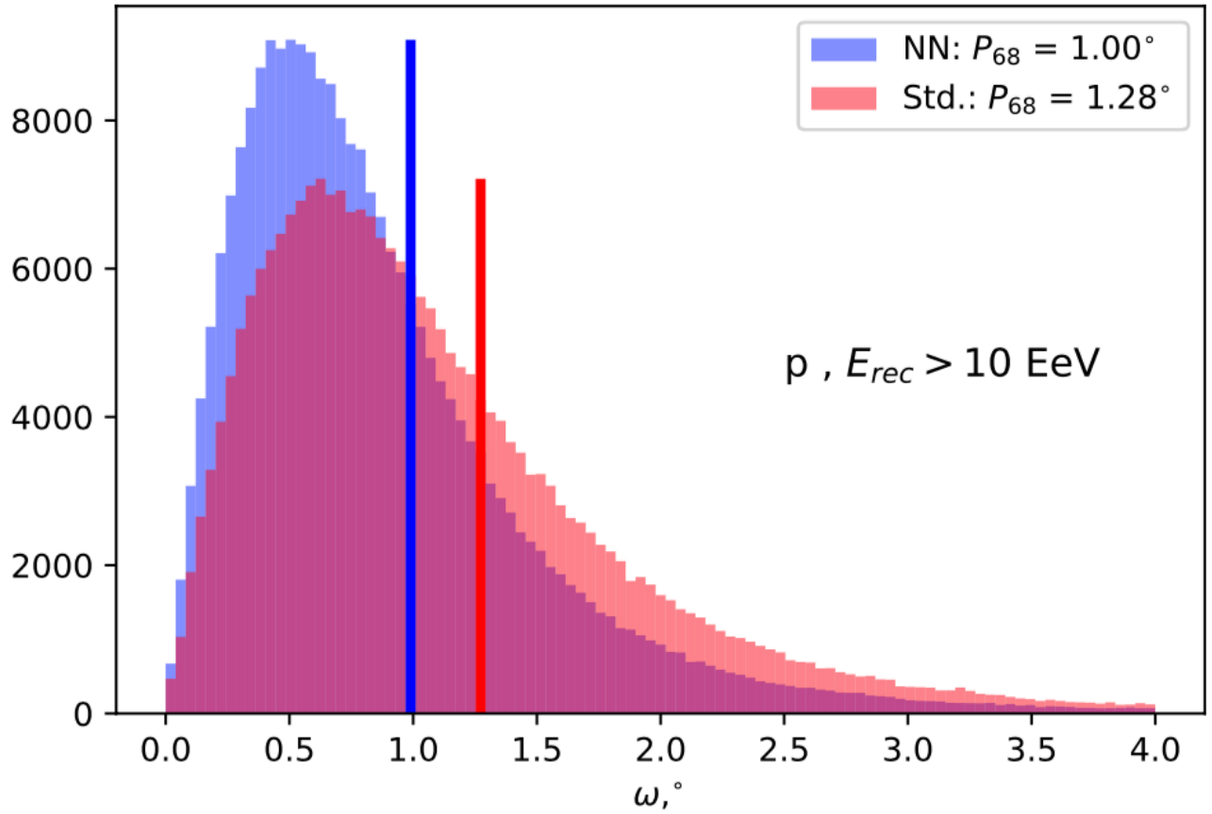
$y$ - предсказываемая величина

$\hat{y}$ - наша оценка предсказываемой величины

если $y$ - коррекция к старой реконструкции

| Event Features \ Waveform | Included | Excluded |
|---|---|---|
| Included | 0.37 | 0.22 |
| Excluded | 0.30 | 0.11 |

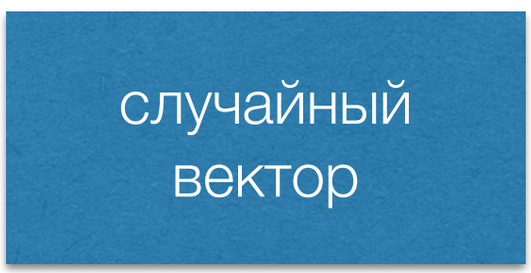


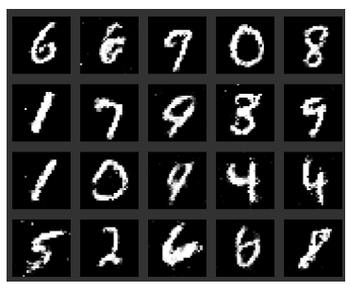



# Генеративные модели

Задача: генерация реалистичных данных

В идеале: с помощью небольшого количества параметров описать все разнообразие данных



Применение:

- анализ данных
- генерация данных по заданным параметрам
- модификация данных
  - шумоподавление
  - восстановление деталей, увеличение разрешения снимков
  - генерация речи, снимков, видео

# Генеративные модели

Задача: генерация реалистичных данных

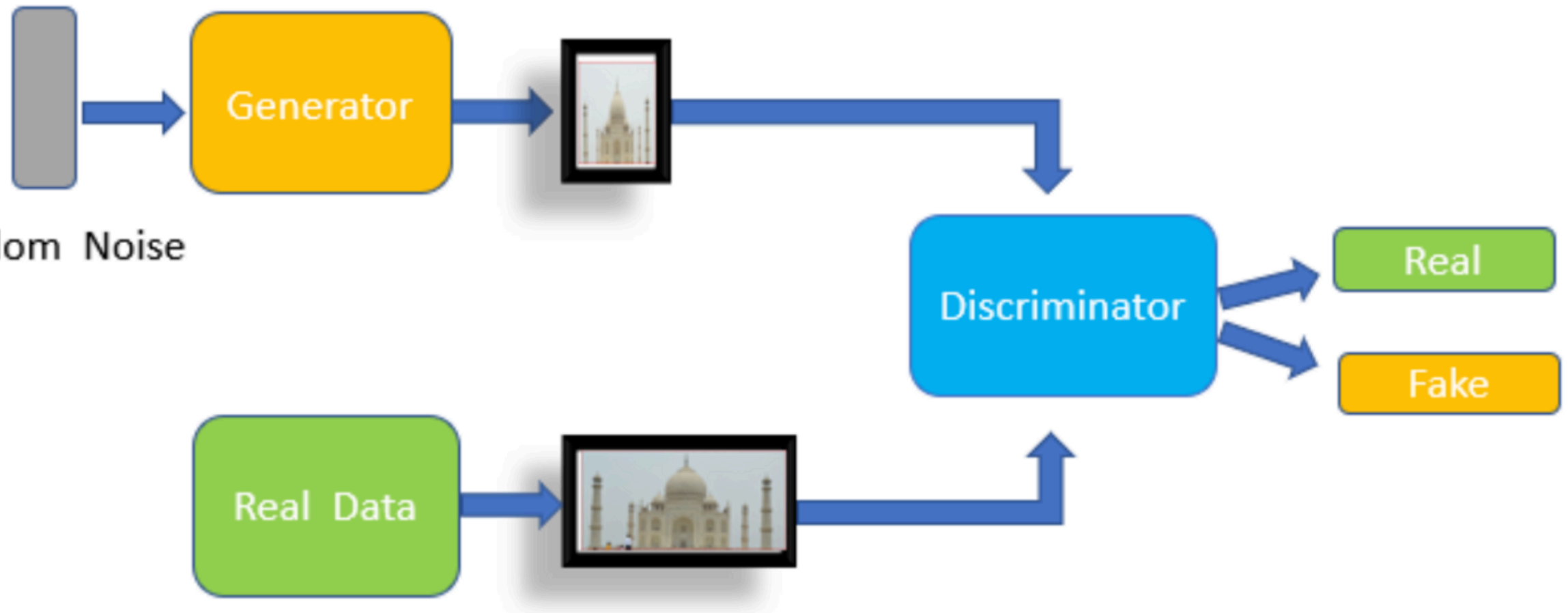
Машинное обучение:


Как это работает:

Если примеры данных содержат избыточную информацию, сравнительно небольшое число входных параметров (размерность вектора) и большое число данных, используемых для обучения, заставляет модель находить оптимальное описание данных

# Генеративно-состязательная сеть

Generative Adversarial Network (GAN)




- **Генеративная модель** учится генерировать образцы
- **Дискриминативная модель** учится отличать правильные («подлинные») образцы от сгенерированных

# Генеративно-состязательная сеть

Generative Adversarial Network (GAN)

## Обучение:

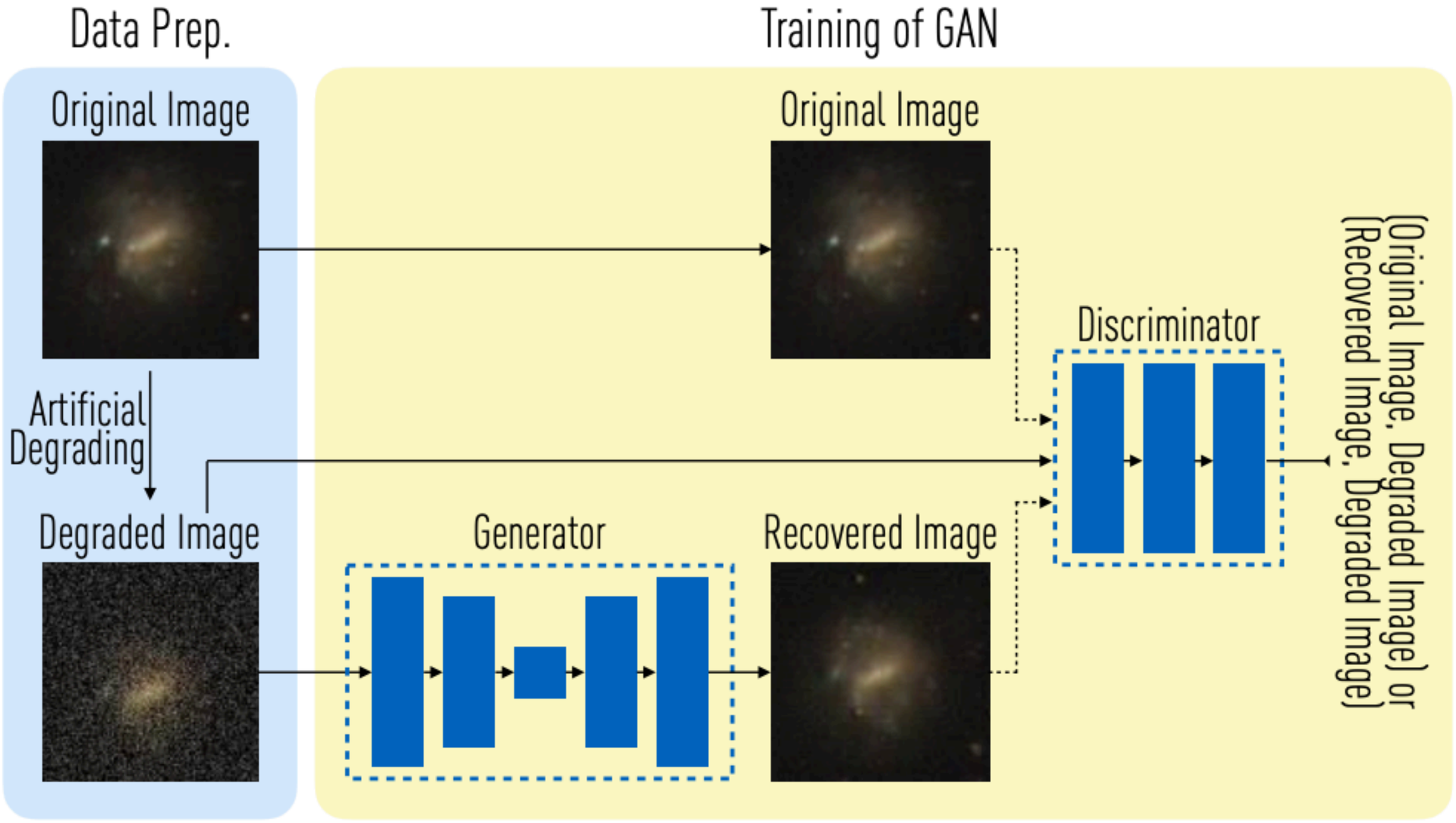
1. Генерируем $N$ образцов с помощью генеративной модели
2. Обучаем дискриминативную модель отличать сгенерированные образцы от реальных на примере $N$ сгенерированных и $N$ реальных
3. Фиксируем веса дискриминативной модели и обучаем генеративую модель обманывать дискриминативную
4. Повторяем весь цикл $n$ раз

# Применение GAN в астрофизике и физике частиц

- **генерация данных для грубого анализа в экспериментальной физике**
  https://arxiv.org/abs/1701.05927 генерация треков частиц в калориметре
  https://arxiv.org/abs/1802.03325 генерация ШАЛ от космических лучей

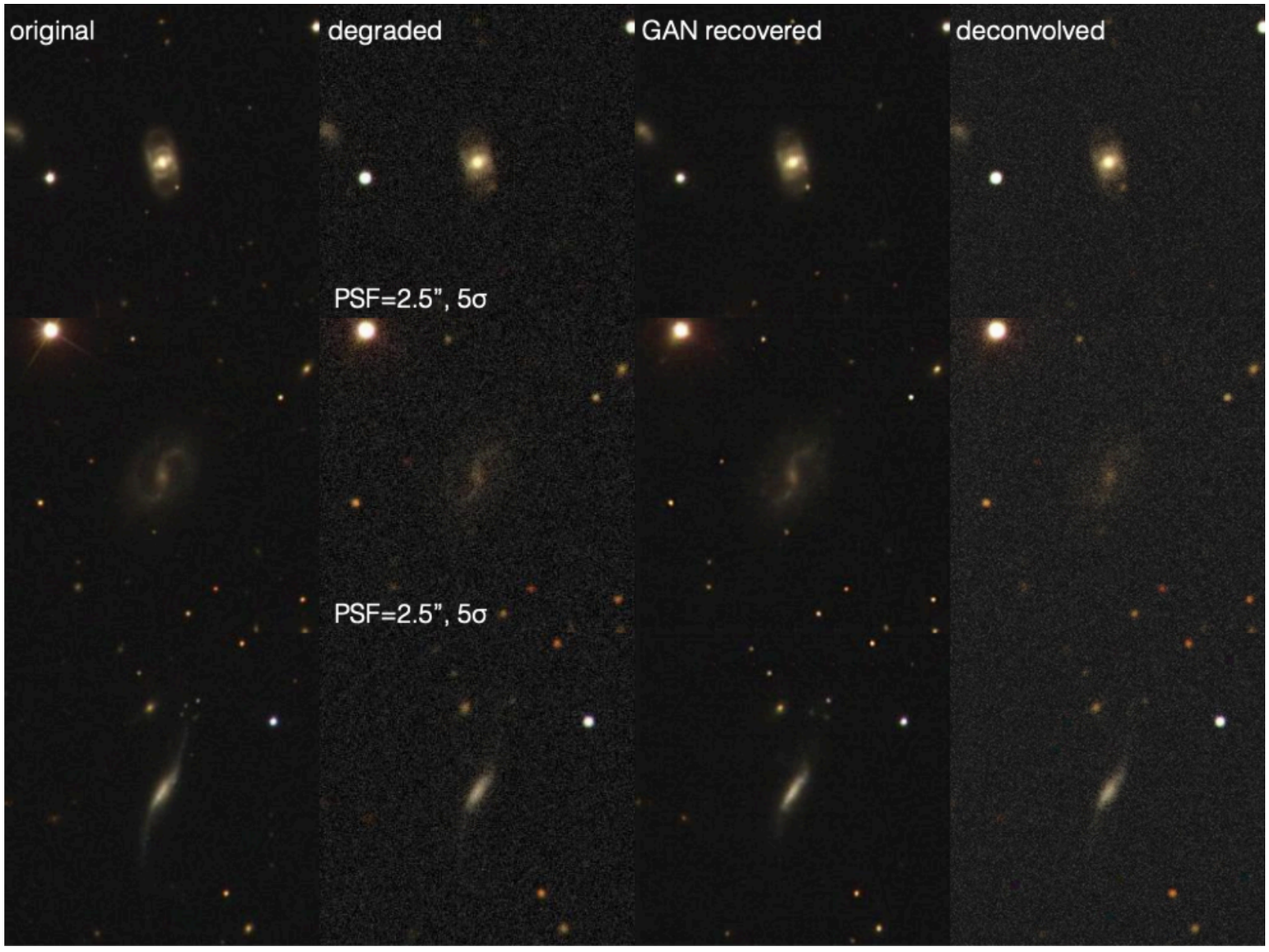
- **восстановление данных, генерация снимков с повышенным разрешением**
  Восстановление снимков галактик https://arxiv.org/abs/1702.00403

# Восстановление снимков галактик




https://arxiv.org/abs/1702.00403

# Восстановление снимков галактик




https://arxiv.org/abs/1702.00403

# Реализация GAN на Python с помощью Keras API

jupyter notebook:

https://colab.research.google.com/drive/1AkSKVJaEJkHfZJG0SACb7Nh1LcEbiBi8?usp=sharing