Свёртка как универсальный оператор обработки изображений

Изображение — дискретный двумерный сигнал I(x,y), где x и y — координаты пикселей. Свёртка возникает как естественное обобщение линейных фильтров для локальных преобразований.

Локальность

Значение в точке зависит только от ограниченной окрестности входа

Линейность

Оператор сохраняет суперпозицию сигналов

Инвариантность

Одно ядро применяется ко всем положениям изображения

Формальное определение двумерной свёртки

Двумерная дискретная свёртка изображения I с ядром K:

$$(I * K)(x, y) = \sum_{u=-m}^{m} \sum_{v=-n}^{n} I(x - u, y - v) \cdot K(u, v)$$

Линейность

Оператор линеен по входному сигналу

Пространственная инвариантность

Одно ядро для всех положений

Локальность

Суммирование по конечной окрестности

Свёртка и корреляция

Корреляция отличается отсутствием отражения ядра:

$$(I \star K)(x, y) = \sum_{u,v} I(x + u, y + v) \cdot K(u, v)$$

Классическая обработка

Различие принципиально для симметричных ядер

Нейронные сети

Фреймворки реализуют корреляцию, но называют свёрткой. Веса обучаются — ориентация подстраивается автоматически

Линейность и суперпозиция

Свёртка сохраняет линейные комбинации:

$$(I_1 + I_2) * K = I_1 * K + I_2 * K$$

$$(\alpha I) * K = \alpha (I * K)$$

1

Нет нелинейных искажений

Все эффекты — следствие формы ядра

2

Критично для CNN

Отдельный слой не аппроксимирует нелинейности без активации

Линейность позволяет анализировать фильтры через отклик на базисные сигналы — импульсы и синусоиды.

Граничные эффекты и padding

На границах изображения окрестность выходит за пределы. Размер выхода:

$$H_{out} = \frac{H_{in} + 2P - K}{S} + 1$$

Zero-padding

Дополнение нулями — затухание отклика у границ

Replication

Повторение крайних значений — сохранение структуры

Reflection

Отражение изображения — минимизация разрывов

Где P — размер padding, K — размер ядра, S — шаг (stride).

Шаг свёртки и подвыборка

Шаг $S > 1$ означает применение фильтра с интервалом. Эквивалентно свёртке с последующей дискретизацией.

- Потеря информации, но снижение вычислений
- Увеличение эффективного рецептивного поля
- Замена явных операций pooling в современных архитектурах

Инженерное расширение: stride не часть классической свёртки, но ключевой элемент CNN

Вычислительная сложность

Наивная реализация двумерной свёртки:

$$O(H \cdot W \cdot K_h \cdot K_w)$$

$O(n^4)$

Сложность

Для квадратных изображений и ядер

GEMM

Оптимизация

Матричные операции в современных фреймворках

Свёртка — основной потребитель ресурсов в CNN. Все архитектурные решения направлены на оптимизацию этого шага.

Свёртка как обобщение градиентных операторов

Оператор Собеля — частный случай свёртки:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Собель, Превитт

Выделение градиентов

Лаплас, DoG

Выделение границ

Гаусс

Сглаживание

Все локальные признаки — результат свёртки с фиксированными ядрами. Свёртка — универсальный язык классических фильтров.

Практическая реализация

Наивная реализация на NumPy:

```
def conv2d(image, kernel):
    h, w = image.shape
    kh, kw = kernel.shape
    pad_h, pad_w = kh // 2, kw // 2
    padded = np.pad(image, ((pad_h, pad_h),
                    (pad_w, pad_w)), mode='constant')
    output = np.zeros_like(image)
    for i in range(h):
        for j in range(w):
            region = padded[i:i+kh, j:j+kw]
            output[i, j] = np.sum(region * kernel)
    return output
```

PyTorch — оптимизированная реализация:

```
conv = nn.Conv2d(in_channels=3, out_channels=32,
 kernel_size=3, padding=1)
y = conv(x)
```

Переход к параметризуемым фильтрам

1

Классический CV

Ядро K задано вручную — фиксированный оператор

2

Обучаемые фильтры

Элементы ядра — параметры модели, оптимизируемые по данным

3

CNN

Свёртка превращается в обучаемое представление

Фундаментальное ограничение: фильтр не может адаптироваться к данным. Следующий шаг — рассматривать веса как параметры оптимизации.

Свёртка как параметризуемый оператор

Элементы ядра становятся параметрами модели:

$$K = \{w_{u,v}\}$$

$$y(x,y) = \sum_{u,v} I(x-u, y-v) \cdot w_{u,v}$$

Математика не меняется

Та же свёртка, но веса выбираются автоматически

Оптимизация

Параметры w подбираются минимизацией функции потерь

Обучаемое представление

Свёртка — элемент параметрической модели

Свёрточный слой в оптимизационной задаче

Обучение — минимизация функции потерь:

$$\min_{w} \sum_{i} \mathcal{L}\left(f(I_i; w), y_i^*\right)$$

Где:

- f(·; w) — композиция свёртки и последующих операций
- $\mathscr{L}$ — функция потерь
- w — веса ядра

Ядро — носитель статистических закономерностей данных. Оптимизация извлекает информативные признаки.

Структурированное линейное отображение

Свёрточный слой как матричное умножение:

$$\mathbf{y} = W_{conv} \cdot \mathbf{x}$$

Разреженность

Каждый выход зависит только от локальной окрестности

Блочная структура

Одно ядро применяется ко всем положениям

Инвариантность

Предположение о стационарности изображения

Жёстко ограниченный класс линейных преобразований для учёта геометрии изображений.

Многоканальный случай и банк фильтров

Входной тензор: I(x, y, c), где c = 1, ..., Cin

Выходной отклик для одного фильтра:

$$y(x,y) = \sum_{c=1}^{C_{in}} \sum_{u,v} I(x-u, y-v, c) \cdot w_{u,v,c}$$

Отображение

$\mathbb{R}^{H \times W \times Cin} \to \mathbb{R}^{Hout \times Wout \times Cout}$

Банк признаков

Cout фильтров — обучаемый базис представления

Рецептивное поле и наращивание контекста

Рецептивное поле — область входа, влияющая на один выход.

Для двух последовательных слоёв с ядрами K_1 и K_2:

$$R = K_1 + K_2 - 1$$

Один слой

Рецептивное поле = размер ядра

Два слоя

Поле растёт линейно

Глубокая сеть

Интеграция информации из больших областей

Глубина позволяет переходить от локальных градиентов к текстурам и сложным структурам.

Нелинейность — необходимый компонент

Без нелинейности композиция линейных слоёв эквивалентна одному линейному оператору.

ReLU-активация:

$$\sigma(z) = \max(0, z)$$

- Аппроксимация нелинейных зависимостей
- Разреженность активаций
- Иерархическая композиция признаков

Критично: без нелинейности глубина не даёт прироста выразительности

Обратное распространение и градиенты

Градиент функции потерь по весам фильтра:

$$\frac{\partial \mathcal{L}}{\partial w_{u,v,c}} = \sum_{x,y} I(x-u, y-v, c) \cdot \frac{\partial \mathcal{L}}{\partial y(x,y)}$$

> Градиент по весам — **корреляция входного сигнала с картой ошибки**. Фильтр обновляется в направлении паттернов, систематически связанных с уменьшением ошибки.

Первые слои переоткрывают детекторы границ — такие паттерны устойчиво коррелируют с целевой функцией.

Инженерная реализация слоя

Минимальный пример в PyTorch:

```
import torch.nn as nn

conv = nn.Conv2d(
    in_channels=3,
    out_channels=32,
    kernel_size=3,
    stride=1,
    padding=1
)

x = torch.randn(1, 3, 224, 224)
y = conv(x)
```

За лаконичной записью — оптимизированная реализация с GEMM, векторизацией и GPU-ускорением.

Связь с классическими признаками

Визуализация весов

Фильтры первых слоёв обученной сети напоминают:

- Операторы Собеля
- Фильтры Габора
- Разности Гауссов (DoG)

Непрерывность идей

CNN не отвергают классический CV, а обобщают его — автоматически подбирая параметры под данные.

Свёрточный слой — **обобщённый оператор признаков**, где форма фильтра — результат оптимизации

Переход к иерархическим архитектурам

1

Один слой

Локальный оператор с ограниченным контекстом

2

Композиция слоёв

Формирование иерархии признаков

3

Глубокая сеть

От простых структур к сложным абстрактным представлениям

Последовательное применение свёрток создаёт иерархию, где простые локальные структуры объединяются в сложные паттерны.

От локальных фильтров к многоуровневым представлениям

Ключевая идея CNN — последовательная композиция свёрточных слоёв.

01	02	03
Первый слой извлекает границы, ориентации, простые текстуры	Второй слой работает с картами признаков, формируя признаки более высокого порядка	Каждый уровень оперирует всё более абстрактными представлениями

Формально: вход каждого слоя — выход предыдущего. Концептуально: иерархия признаков.

Рост рецептивного поля в глубокой сети

Для L слоёв со свёртками размера K_1, K_2, ..., KL и шагом 1:

$$R = 1 + \sum_{l=1}^{L} (K_l - 1)$$

Фундаментальный механизм: глубина заменяет глобальные операции локальной композицией

Постепенная агрегация информации

Сохранение локальной структуры

Устойчивость к шуму

Иерархия визуальных признаков

Глубокие слои

Части объектов, конфигурации

Средние слои

Текстуры, композиционные паттерны

Ранние слои

Границы, ориентации

Каждый слой реализует отображение $\phi_l: \mathbb{R}^{H_l \times W_l \times C_l} \to \mathbb{R}^{H_{l+1} \times W_{l+1} \times C_{l+1}}$

Нелинейность разрушает линейность композиции, формируя семантическое пространство признаков.

Агрегация и подвыборка: роль pooling

Max-pooling размера 2×2:

$$y(x, y) = \max_{(u,v)\in\Omega} x(2x + u, 2y + v)$$

Свойства

- Нелинейная операция
- Без обучаемых параметров
- Агрегирование локальной информации

Роль

- Снижение чувствительности к сдвигам
- Уменьшение размера представления
- Локальная инвариантность

Индуктивные смещения CNN

Архитектура CNN кодирует предположения о структуре визуальных данных:

Локальность

Важная информация в локальных окрестностях

Инвариантность

Признак может появляться в разных частях изображения

Иерархичность

Сложные структуры из простых

Эти смещения сокращают пространство поиска параметров, обеспечивая устойчивое обучение.

Базовая архитектура CNN

Типичный блок свёрточной сети:

1	2	3
Conv	ReLU	Pooling
Свёрточный слой	Нелинейность	Агрегация

Композиция отображений:

$$f = f_n \circ f_{n-1} \circ \cdots \circ f_1$$

Несколько блоков формируют признаковый экстрактор, затем полносвязные слои для принятия решения. Все уровни представления формируются автоматически.

LeNet-5 как канонический пример

Архитектура LeNet-5 — чередование свёрточных и pooling-слоёв:

- Вход: 32×32 пикселей
- Постепенное увеличение числа каналов
- Уменьшение пространственного разрешения
- Переход от пространственного к признаковому представлению

Даже неглубокая сеть формирует иерархию, достаточную для распознавания сложных паттернов

Размерности и параметры

Число параметров свёрточного слоя:

$$K_h \cdot K_w \cdot C_{in} \cdot C_{out}$$

✓

Не зависит от размера входа

Ключевое преимущество CNN

O(HW)

Вычислительная стоимость

Масштабируется с размером изображения

Ранние слои наиболее дороги вычислительно. Понимание этих соотношений критично для проектирования архитектур.

Устойчивость к сдвигам и деформациям

Свёртка

Эквивариантность к сдвигам на ранних слоях

Pooling

Локальная инвариантность

Глубина

Возрастающая инвариантность на глубоких уровнях

Комбинация механизмов снижает чувствительность к шуму, неточным выравниваниям и локальным деформациям. Приближённая эквивариантность делает CNN эффективными в распознавании объектов.

Свёртка как матричное умножение (GEMM)

На практике свёртка приводится к матричному умножению для высокой производительности.

Развёртка входа:

$$X \in \mathbb{R}^{(H_{out}W_{out}) \times (K_h K_w C_{in})}$$

Веса фильтров:

$$W \in \mathbb{R}^{(K_h K_w C_{in}) \times C_{out}}$$

Выход:

$$Y = XW$$

Это позволяет использовать высокоэффективные библиотеки матричного умножения, оптимизированные под CPU и GPU.

Память, батчи и пропускная способность

Данные обрабатываются батчами размера B:

$$X \in \mathbb{R}^{B \times C \times H \times W}$$

Преимущества батчей

- Эффективная загрузка GPU
- Стабилизация оценки градиентов
- Параллелизация вычислений

Компромиссы

- Увеличение потребления памяти
- Узкое место при обучении глубоких моделей

Проектирование CNN — баланс между размером батча, разрешением, числом каналов и глубиной.

Численные эффекты и инициализация

Обучение чувствительно к начальной инициализации параметров.

Инициализация Xe для ReLU:

$$w \sim \mathcal{N}\left(0, \frac{2}{K_h K_w C_{in}}\right)$$

Поддержание дисперсии

Активации остаются стабильными при переходе между слоями

Предотвращение затухания

Градиенты не исчезают и не взрываются

Визуализация активаций и весов

Визуализация — инструмент анализа и диагностики CNN.

Пример визуализации весов:

```
import matplotlib.pyplot as plt

weights = conv.weight.detach().cpu()
plt.imshow(weights[0, 0], cmap="gray")
plt.show()
```

Веса фильтров

Напоминают классические операторы: границы, ориентации, текстуры

Карты активаций

Показывают, какие области возбуждают конкретные каналы

Первые слои как обученные фильтры

Первые свёрточные слои обучаются фильтрам, близким к:

- Операторам Собеля
- Фильтрам Габора
- Разностям Гауссов (DoG)

CNN автоматически находит оптимальные фильтры для данной задачи и распределения данных.

> Классические операторы — частные случаи оптимального решения задачи обучения

Связь с Sobel, Gabor и DoG

Двумерный фильтр Габора:

$$G(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)\cos(2\pi f x)$$

Классический CV

Параметры формы, частоты, ориентации задаются вручную

CNN

Аналогичные структуры возникают автоматически как результат оптимизации

> CNN не противопоставляются классическому CV, а включают его как частный случай

Ограничения базовых CNN

Локальность свёртки

Затрудняет моделирование дальних зависимостей

Жёсткая структура

Меньше гибкости при работе с глобальными контекстами

Рост вычислений

Увеличение рецептивного поля через глубину повышает стоимость

Эти ограничения мотивировали появление продвинутых архитектур. Но базовые CNN эффективны благодаря индуктивным смещениям.

Свёртка и эквивариантность

Свёртка эквивариантна к сдвигам:

$$(I(x - \Delta x, y - \Delta y) * K)(x, y) = (I * K)(x - \Delta x, y - \Delta y)$$

Эквивариантность

Сдвиг входа → сдвиг выхода

Отличие от FC

Фундаментальное преимущество CNN

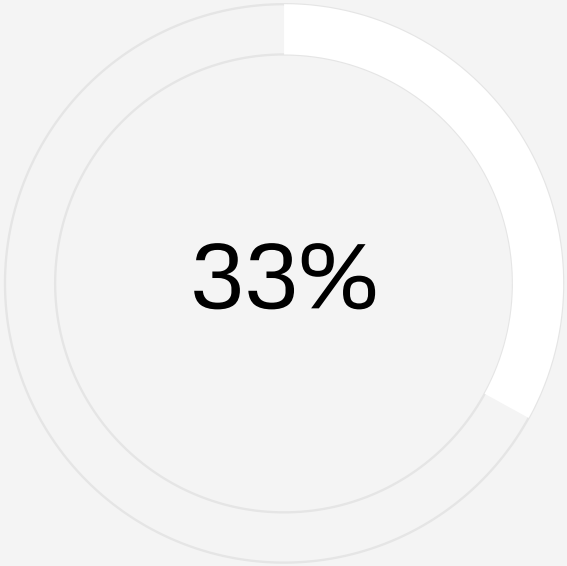
Устойчивость

К положению объекта в кадре

Pooling дополняет эквивариантность частичной инвариантностью для распознавания.

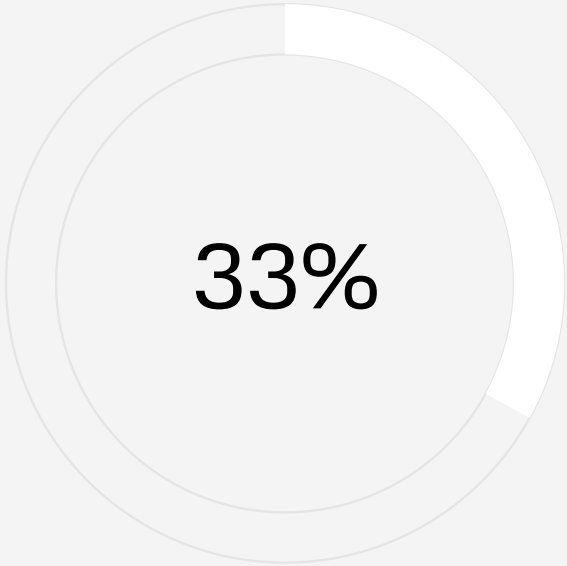
Инженерные компромиссы

Проектирование CNN — баланс между точностью, скоростью и ресурсами.



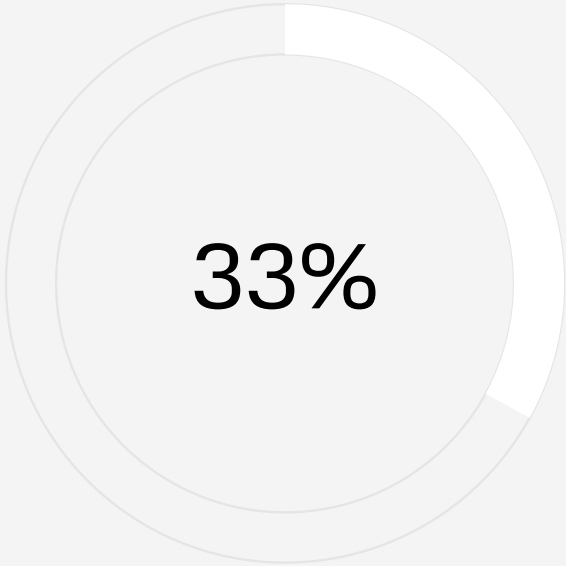
Точность

Больше каналов, глубже сеть



Скорость

Меньше разрешение, меньше слоёв



Память

Размер батча, число параметров

Инженерная практика — поиск баланса, где архитектура достаточно проста для обучения и достаточно мощна для задачи.

Заключение

Мы прошли путь от классических операторов к обучаемым свёрточным представлениям:

1. **Свёртка как универсальный оператор**
   Формальное рассмотрение, универсальность для фильтров и признаков
2. **Параметризуемая модель**
   Веса ядра — параметры оптимизации, обучение по данным
3. **Иерархия признаков**
   Композиция слоёв, рост рецептивного поля, переход к абстракциям
4. **Инженерная реализация**
   GEMM, визуализация, связь с классическим CV

CNN — не новая математика, а переосмысление классических операций в контексте обучения.