Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт
Высшая школа теоретической механики и математической физики

РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ВИЗУАЛЬНЫХ ОБРАЗОВ С ПОМОЩЬЮ НЕЙРОСЕТЕВЫХ АЛГОРИТМОВ КАК ЭЛЕМЕНТА КОНТРОЛЯ КАЧЕСТВА НА ПРОИЗВОДСТВЕ

Сфера применения – производство солнцезащитного оборудования

Выполнил
студент гр.5040103/20301 Д.Л. Оленчук

Руководитель
профессор ВШТМиМФ, д.ф.-м.н. В.М. Иванов

Консультант
ассистент ВШТМиМФ Д.С. Перец

Задача:

- Разработать нейросетевую модель для распознавания предметов в видеопотоке.
- Произвести оценку качества распознавания.
- Разработать дополнительный модуль для сравнения результатов распознавания со спецификацией.
- Проверить применимость разработанной программы

Задачи в распознавании:

Классификация

CAT

Локализация

CAT

Сегментация

CAT, DOG

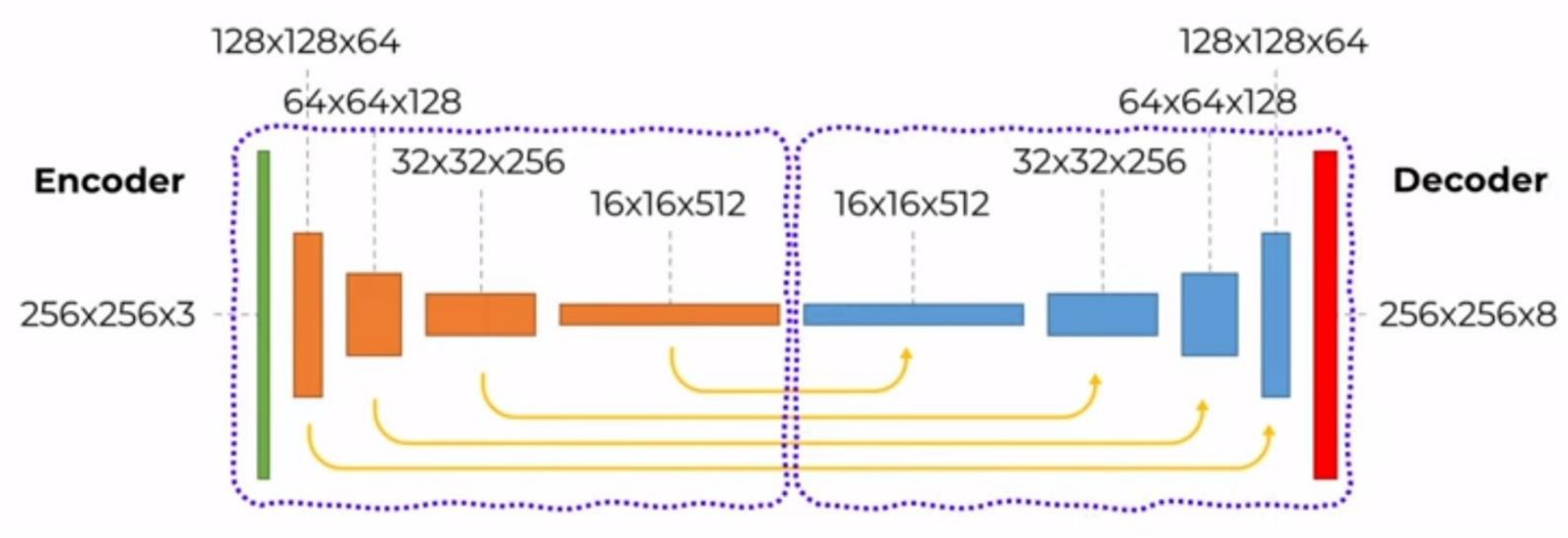
Метод решения:

Мультиклассовая сегментация

Выбор архитектуры нейросети

Архитектура	Вычисли-тельная эффектив-ность	Подходит для задач сегмен-тации	Простота настройки	Точность распознава-ния объектов сложных форм и разных размеров	Сум-ма баллов
R-CNN	1	2	2	3	8
Fast R-CNN	3	3	3	3	12
Faster R-CNN	4	3	3	4	14
YOLO	5	2	4	3	14
U-Net	3	5	5	5	18

Концепция U-Net



Основные шаги

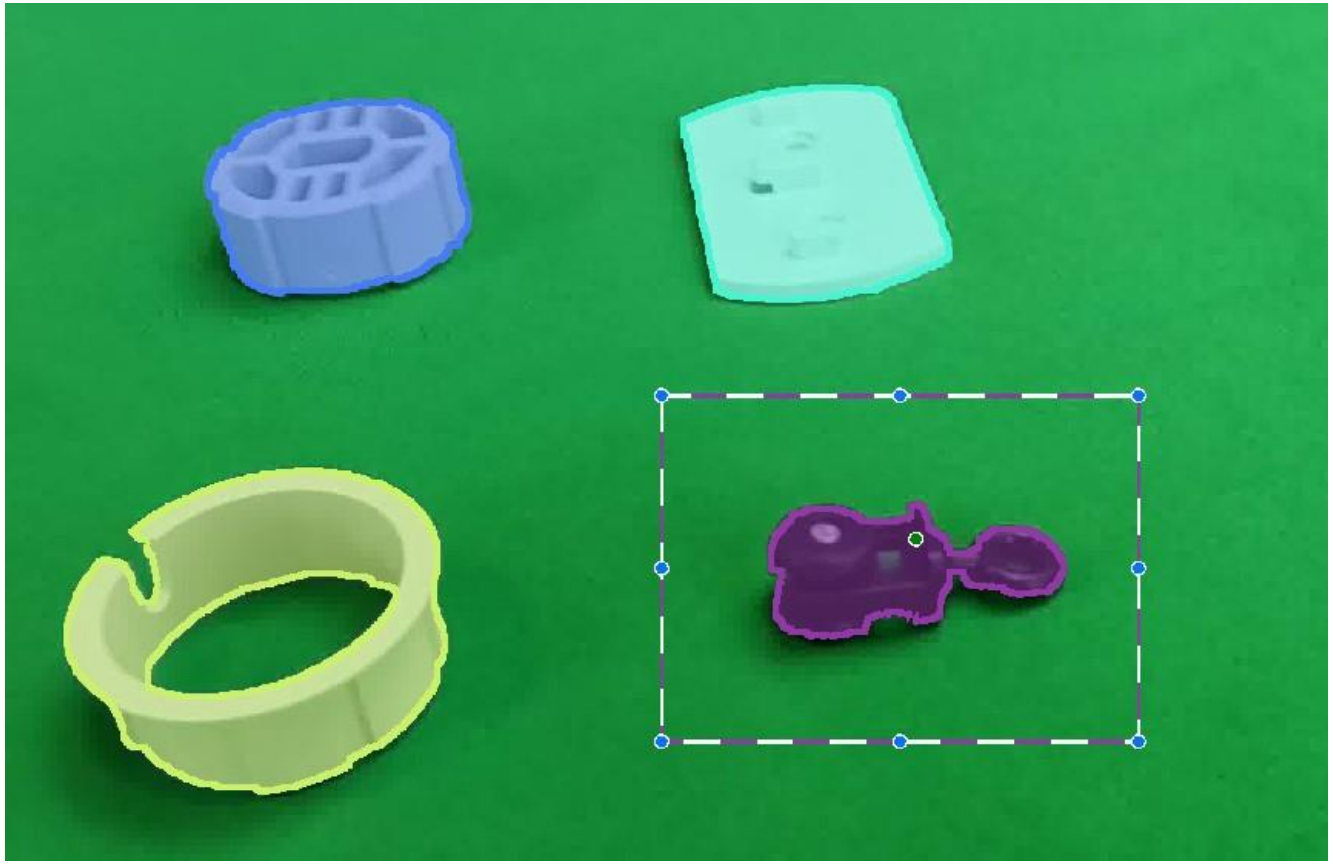
- Создание обучающего набора данных
- Построение модели нейросети
- Обучение
- Тестирование

Создание набора данных

- Съёмка видео комплектующих оборудования (38 сек.).
- Разделение на изображения (1338 шт.)
- Разметка и присвоение классов (42 изображения)

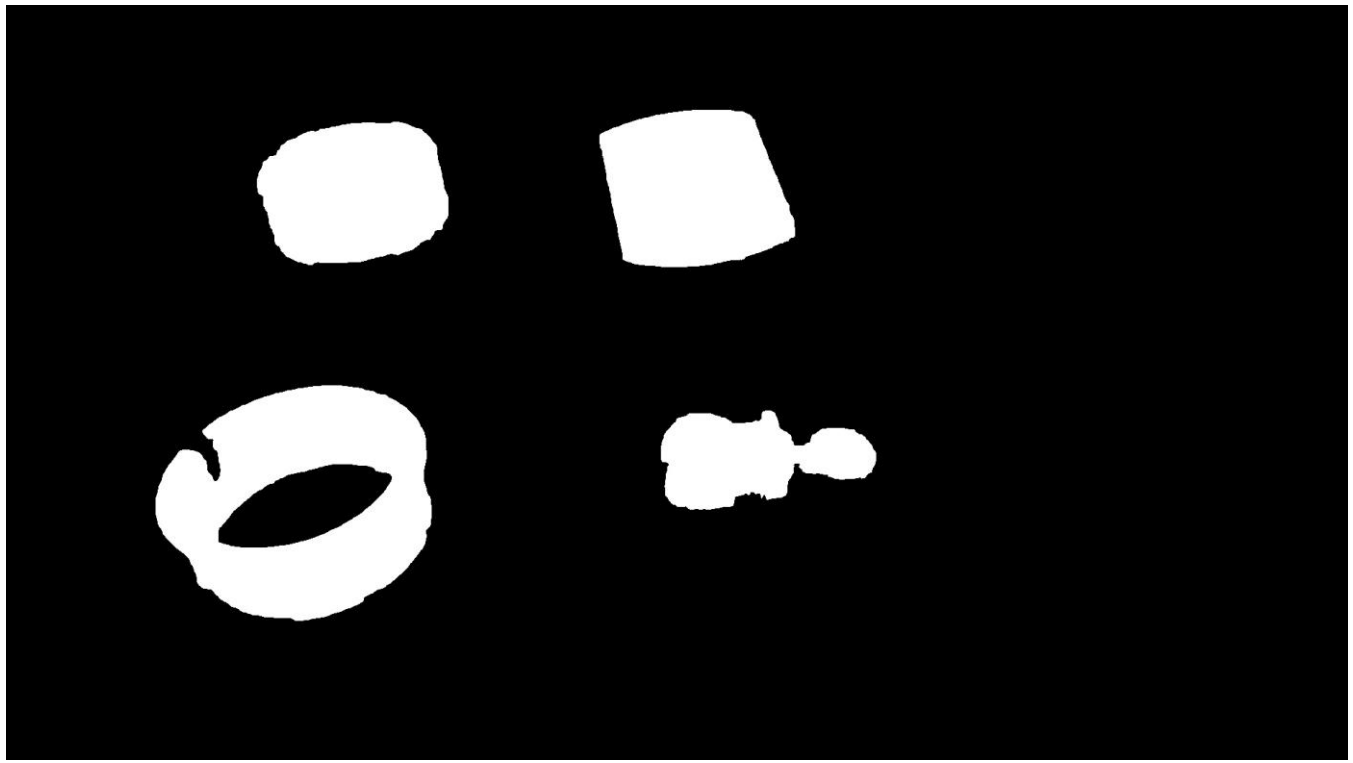
Разметка данных

Выделение контуров, назначение классов



Результат разметки:

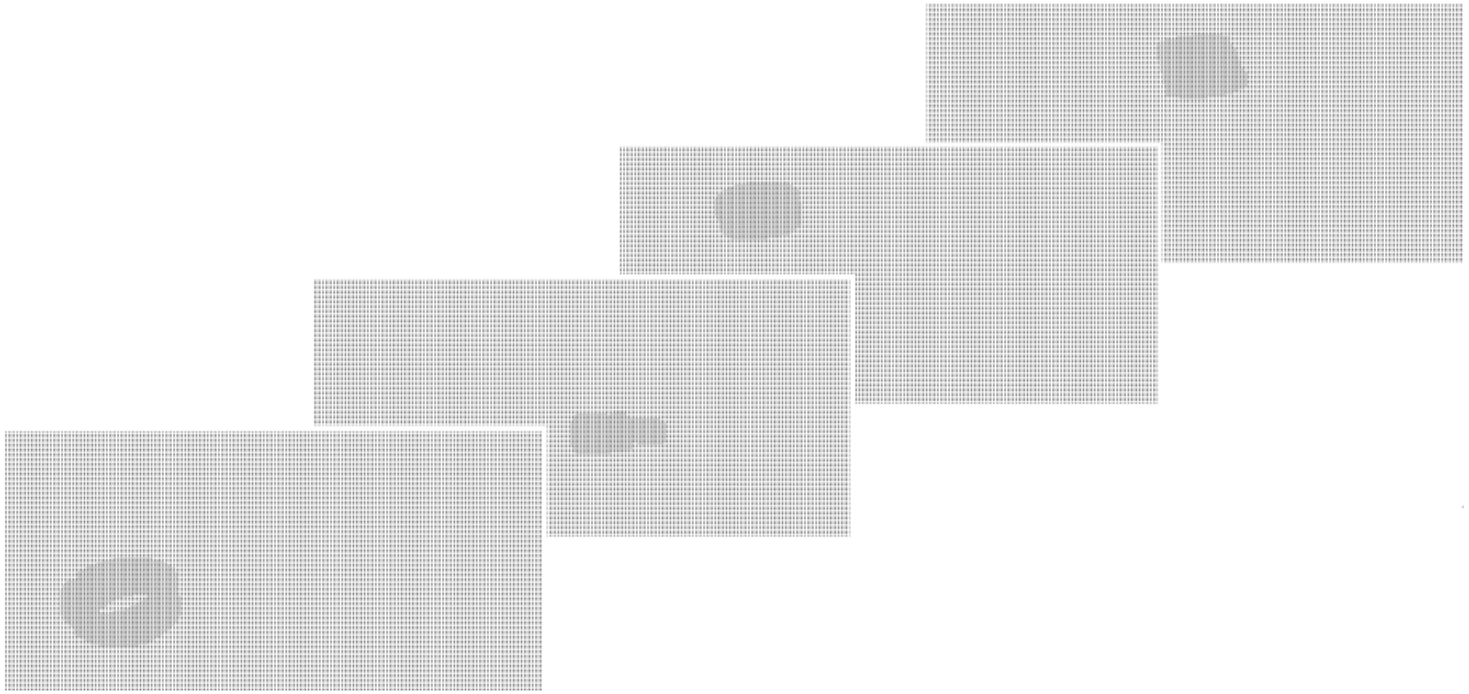
Маска для каждого кадра (png).

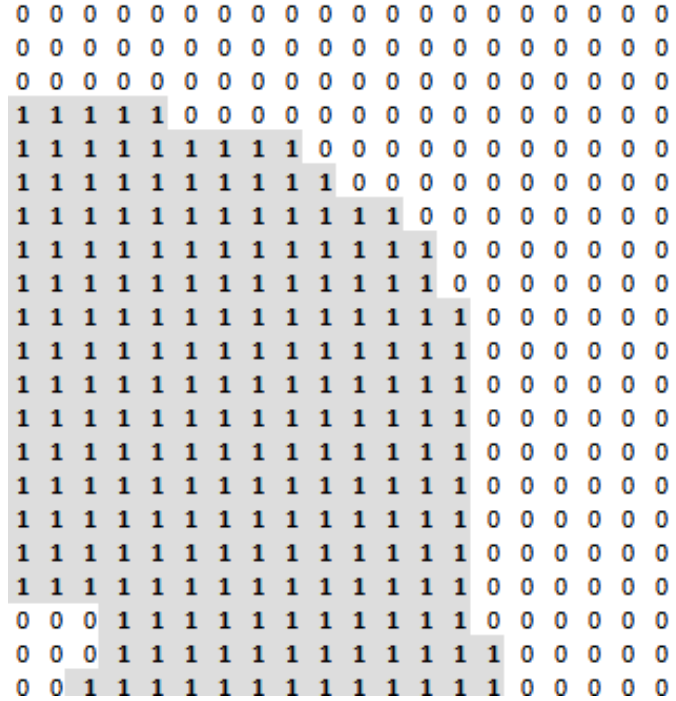


Набор данных сформирован

Построение нейросетевой модели

Преобразование масок в бинарные многоканальные изображения, размерность которых равна количеству классов объектов



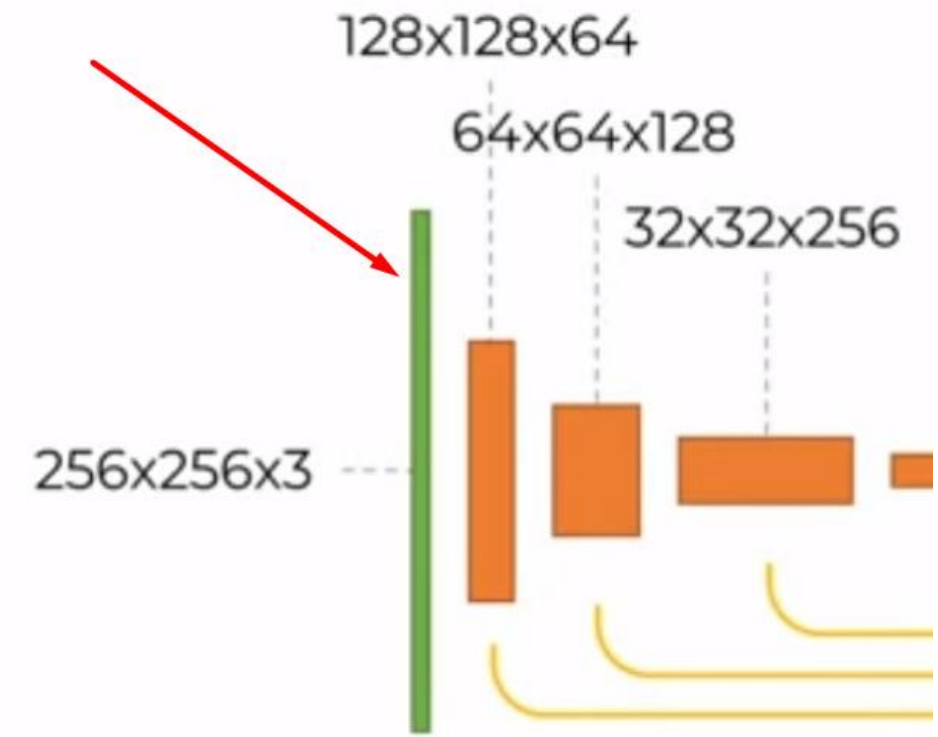


Построение нейросетевой модели

Входной слой – тензор 256x256x3,

где 256x256 – размер изображения,

3 – каналы R G B

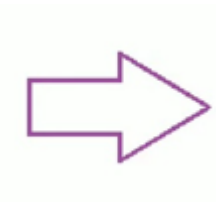


Входной слой:

- просто определяет структуру входных данных
- не выполняет никаких вычислений
- передает данные в первый слой свертки

Построение нейросетевой модели

Преобразование изображений 1920 x 1080 в 256 x 256:



Построение нейросетевой модели

Преобразование изображений 1920 x 1080 в 256 x 256:

1. Вычисление коэффициентов масштабирования $s_w = \frac{256}{1920}$ $s_h = \frac{256}{1080}$
2. Вычисление координат каждого пикселя (i, j) в новом изображении координат (x, y) в исходном изображении: $x = i \times s_w$ $y = j \times s_h$
3. Определение соседних пикселей (x_1, y_1) и (x_2, y_2) в исходном изображении:

$$x_1 = \lfloor x \rfloor, \quad x_2 = \lceil x \rceil$$

$$y_1 = \lfloor y \rfloor, \quad y_2 = \lceil y \rceil$$

4. Вычисление весов для билинейной интерполяции:

$$w_{x_1} = x_2 - x, \quad w_{x_2} = x - x_1$$

$$w_{y_1} = y_2 - y, \quad w_{y_2} = y - y_1$$

Построение нейросетевой модели

5. Вычисление значения пикселей:

Пусть $I(x_1, y_1), I(x_2, y_1), I(x_1, y_2)$, и $I(x_2, y_2)$ будут значениями пикселей в исходном изображении.

Промежуточные значения пикселей вычисляются как:

$$I_{x_1}(y) = I(x_1, y_1) \cdot w_{y_1} + I(x_1, y_2) \cdot w_{y_2}$$

$$I_{x_2}(y) = I(x_2, y_1) \cdot w_{y_1} + I(x_2, y_2) \cdot w_{y_2}$$

Итоговое значение пикселя $I(i, j)$ в новом изображении:

$$I(i, j) = I_{x_1}(y) \cdot w_{x_1} + I_{x_2}(y) \cdot w_{x_2}$$

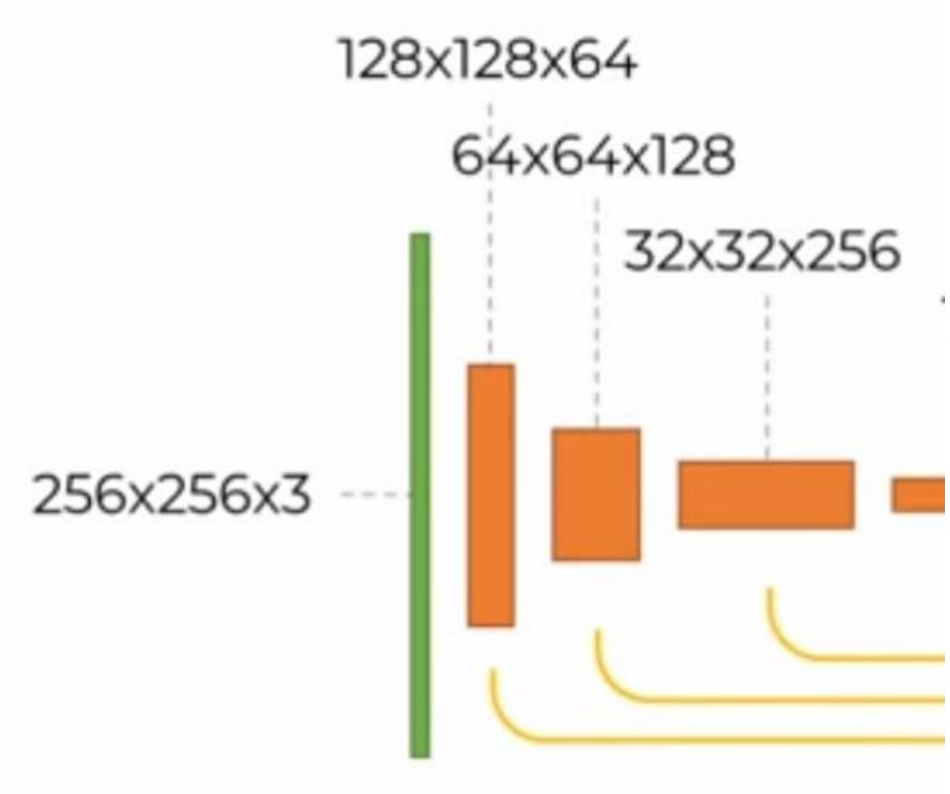
Построение нейросетевой модели

Энкодер:

- уменьшает пространственное разрешение изображения
- увеличивает количество каналов для представления более высокоуровневых абстракций.

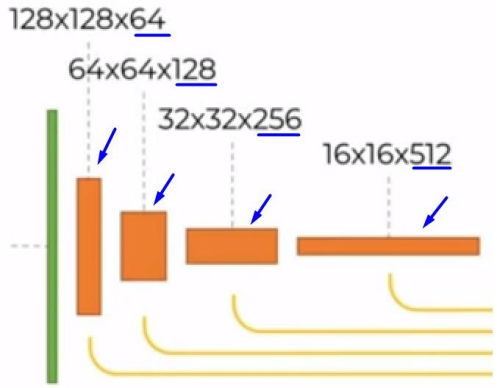
Состоит из:

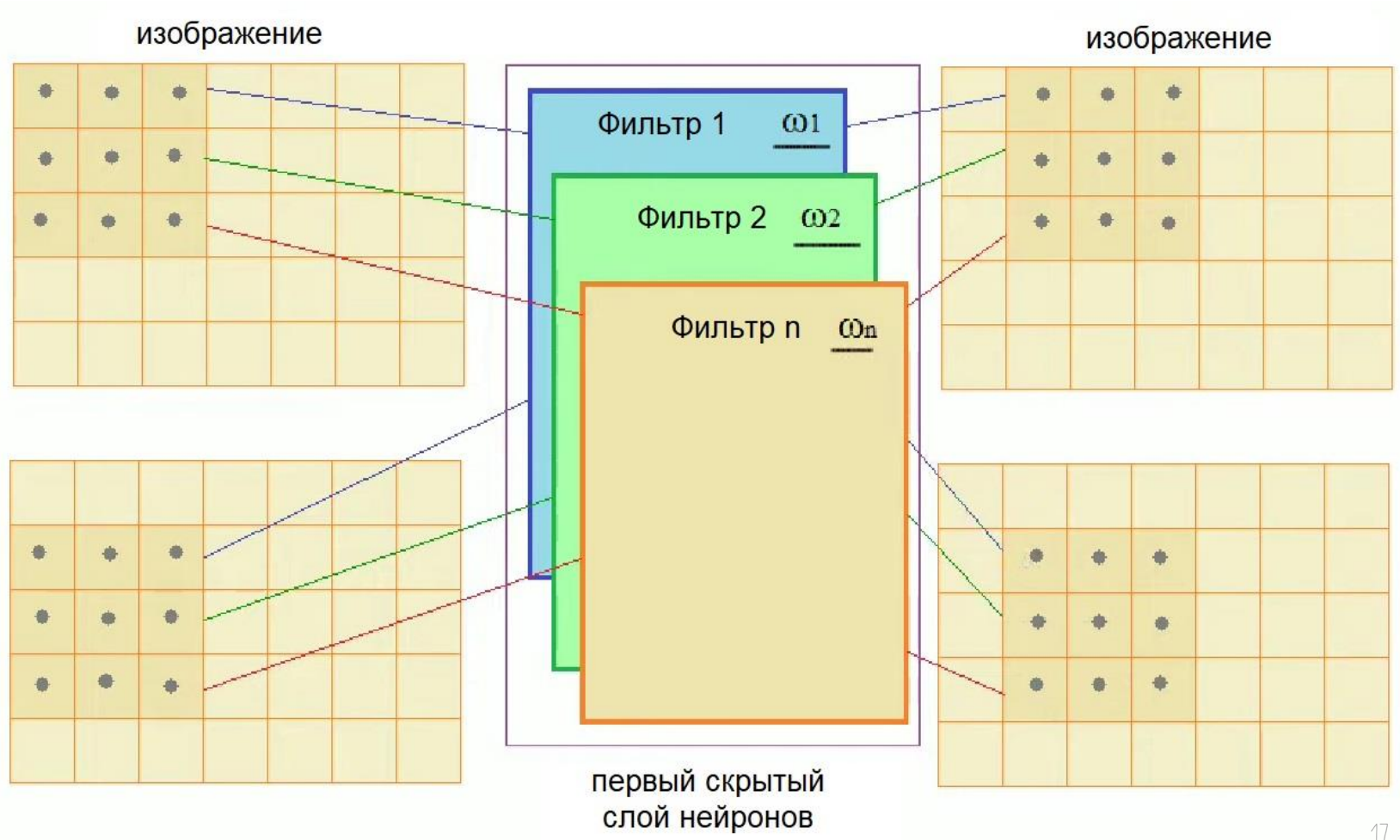
- свёрточных слоёв с фильтрами и с шагом (замена пуллинга)
- слоёв пакетной нормализации
- активационной функции



Построение нейросетевой модели

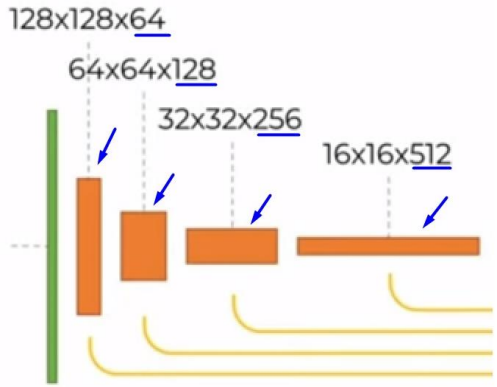
Ядро (фильтр)

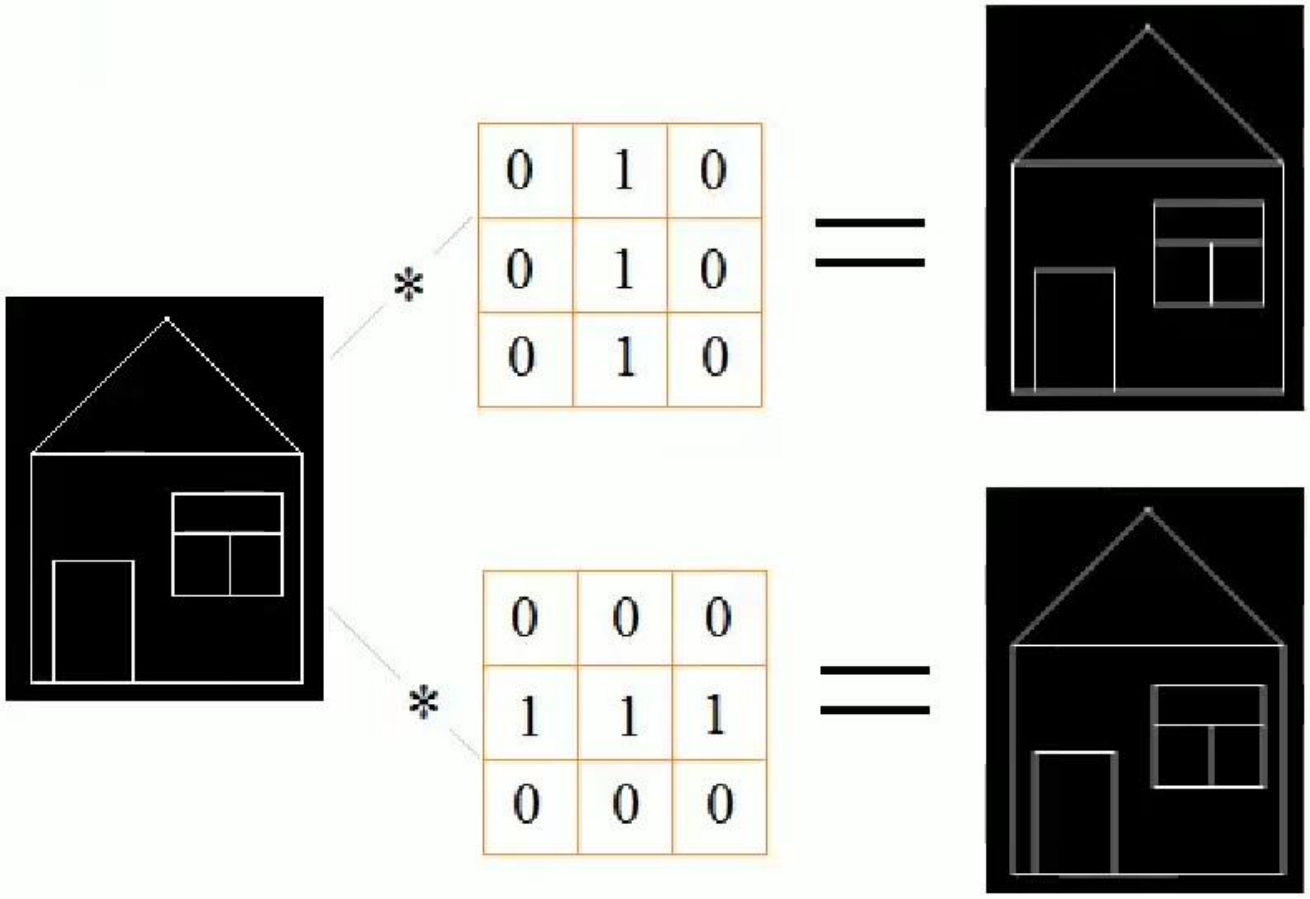




Построение нейросетевой модели

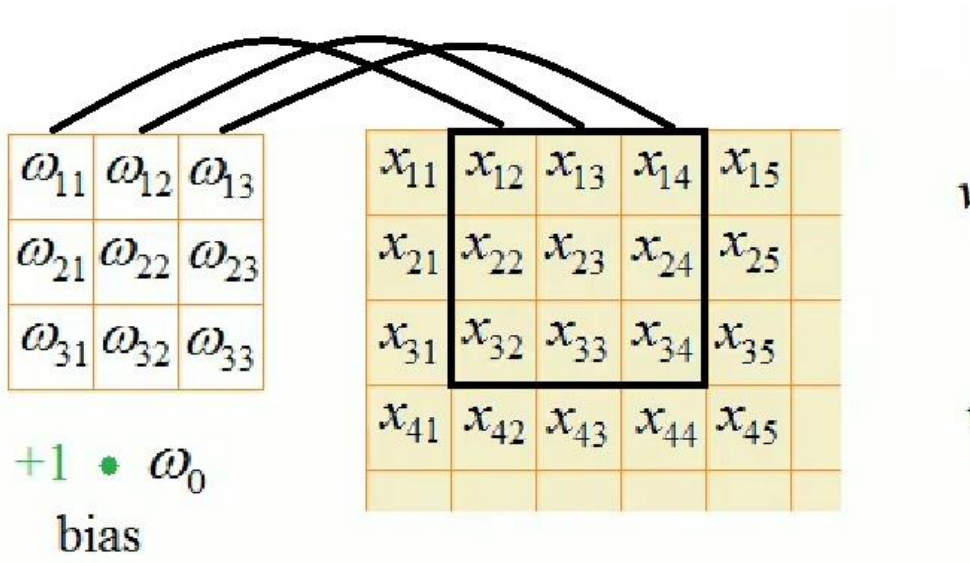
Ядро (фильтр)





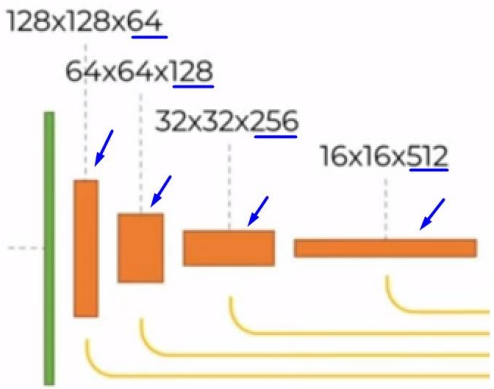
Построение нейросетевой модели

Ядро (фильтр)



$$v_{0,0} = \sum_{i=1}^{3}\sum_{j=1}^{3} x_{i,j} \cdot \omega_{ij} + \omega_0$$

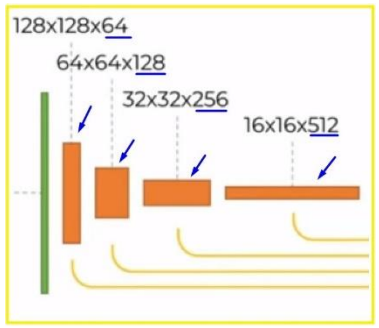
$$v_{0,1} = \sum_{i=1}^{3}\sum_{j=1}^{3} x_{i,j+1} \cdot \omega_{ij} + \omega_0$$

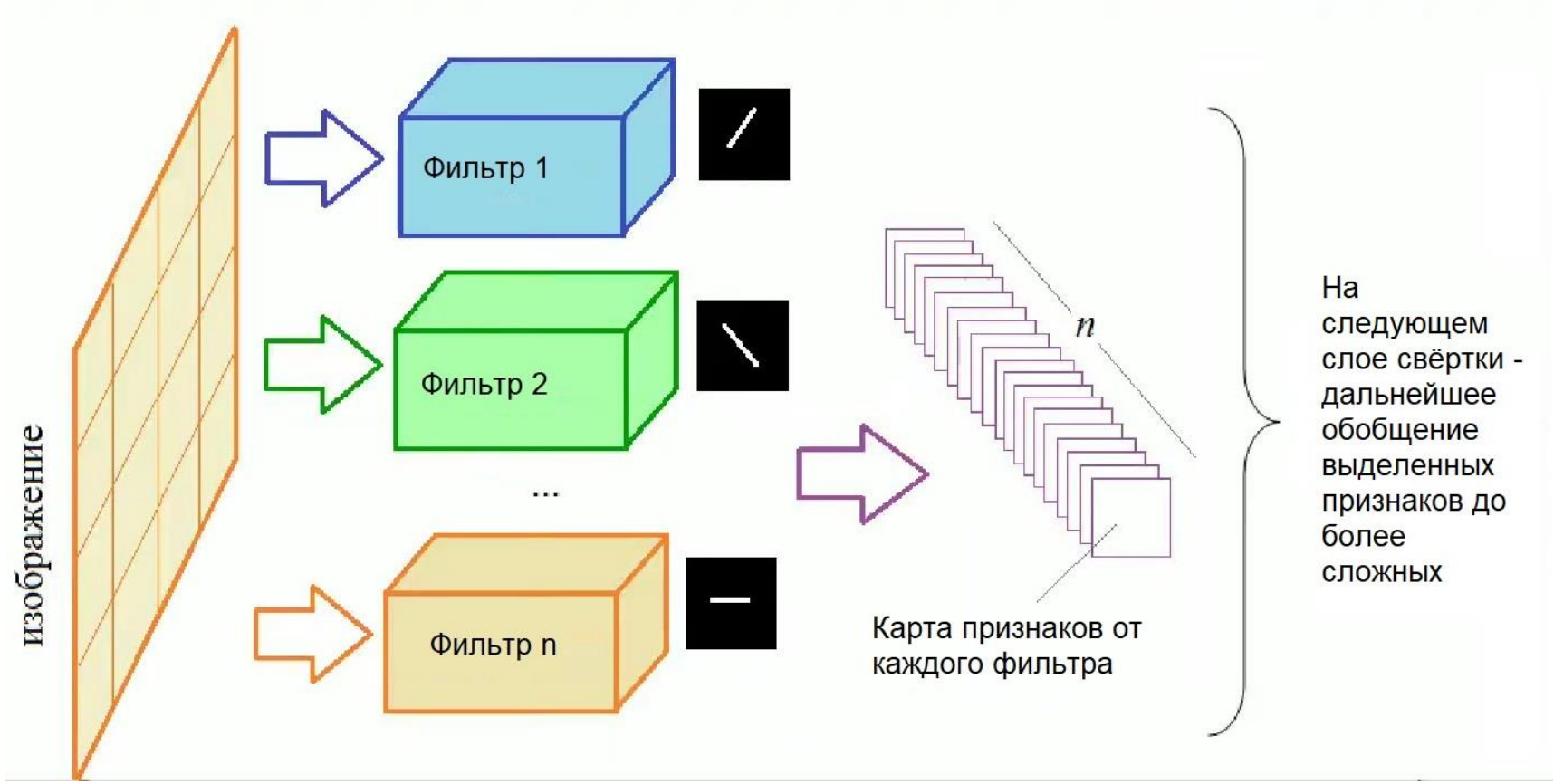


$$v_{k,m} = \sum_{i=1}^{3}\sum_{j=1}^{3} x_{i+k,j+m} \cdot \omega_{ij} + \omega_0, \qquad k,m = 0,1,2,\ldots$$

Построение нейросетевой модели

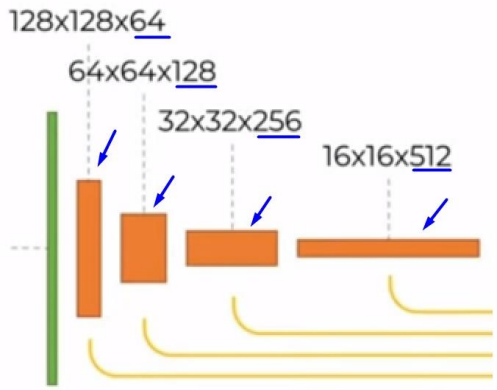
Ядро (фильтр)

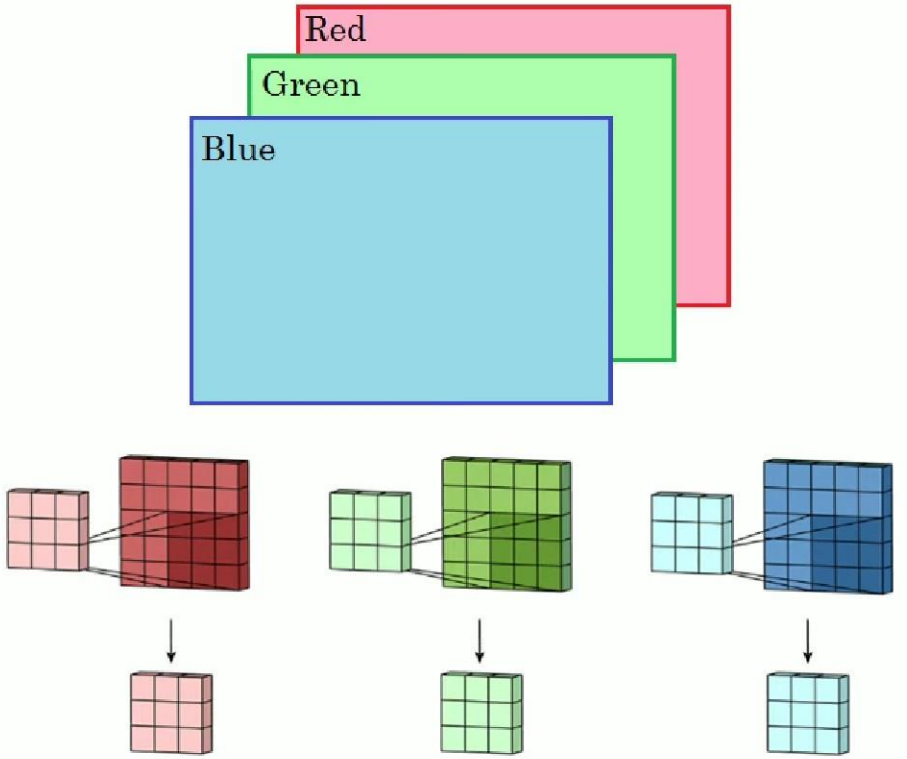


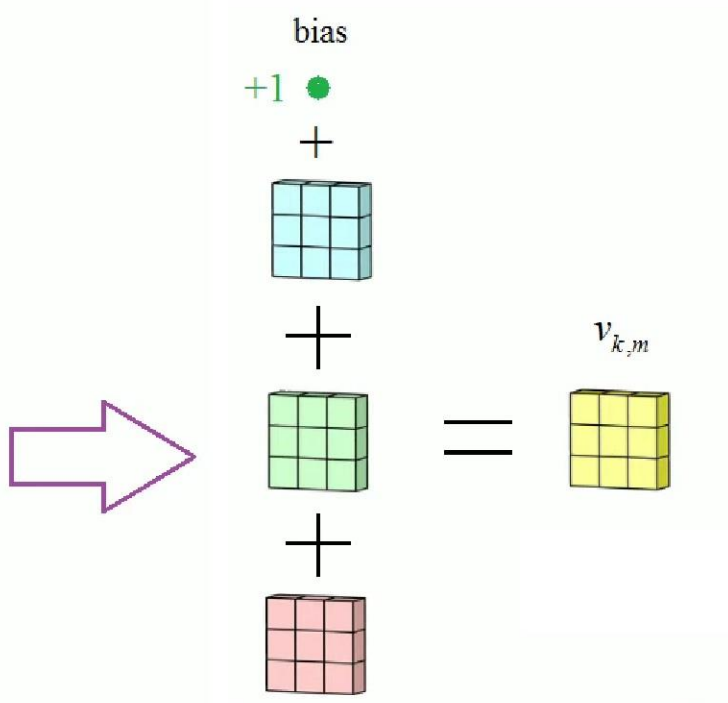


Построение нейросетевой модели

Ядро (фильтр)





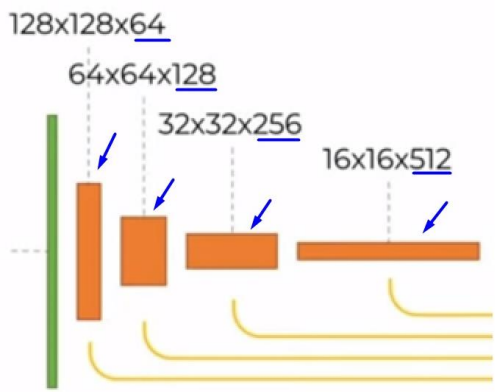


Построение нейросетевой модели

Свёрточные слои: извлечение признаков с помощью фильтра (ядра) свёртки

Входной тензор H x W x C (высота, ширина, кол-во каналов), K – фильтр размером k_h x k_w x C, где k_h и k_w – высота и ширина фильтра, C – кол-во каналов фильтра. Пусть S – шаг свёртки (stride), P – паддинг (добавление нулевых элементов вокруг существующих данных). Формула значения в позиции (i, j) выходного тензора O размером H_0 x W_0 x F (F – кол-во фильтров):

$$O[i,j,f] = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} \sum_{c=0}^{C-1} I[i \cdot S + m - P, j \cdot S + n - P, c] \cdot K[m,n,c,f] + b_f$$



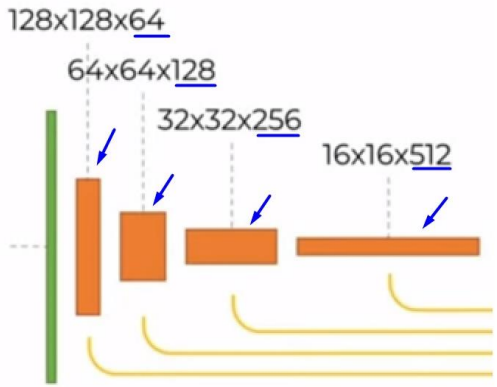
где:

- $O[i,j,f]$ — значение выходного тензора в позиции (i,j) для фильтра f
- I — входной тензор.
- K — фильтр.
- b_f — смещение (bias) для фильтра f.
- S — шаг свертки.
- P — паддинг.
- H_o и W_o — высота и ширина выходного тензора, вычисляемые как:

$$H_o = \left\lfloor \frac{H+2P-k_h}{S} \right\rfloor + 1$$

$$W_o = \left\lfloor \frac{W+2P-k_w}{S} \right\rfloor + 1$$

Построение нейросетевой модели

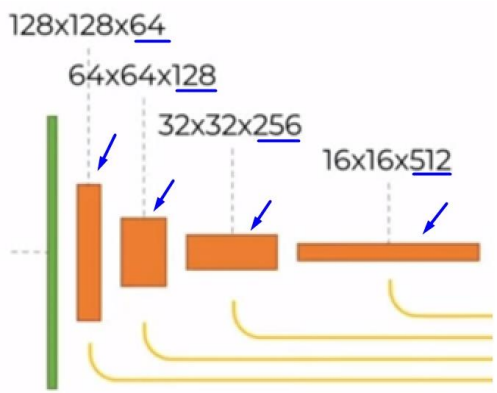


Слой пакетной нормализации:

- нормализует выход свёрточного слоя, приводя значения к среднему 0 и дисперсии 1
- стабилизирует обучение и ускоряет сходимость, уменьшая проблему исчезающих и взрывающихся градиентов
- делает модель более устойчивой к изменениям в параметрах, таких как начальная скорость обучения

Построение нейросетевой модели

Процесс пакетной нормализации:



Для данного мини-батча входных данных $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ с размером батча m:

1. **Вычисление среднего значения и стандартного отклонения:**

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

2. **Нормализация:**

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

где ϵ — маленькое значение для избежания деления на ноль (обычно 10^{-5} или 10^{-6}).

3. **Скалирование и смещение:**

$$y_i = \gamma\hat{x}_i + \beta$$

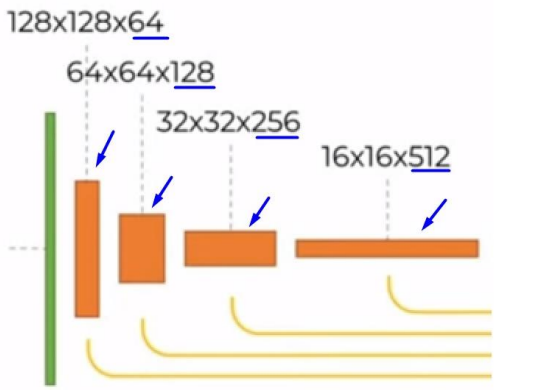
где γ и β — параметры, обучаемые в процессе обучения, которые позволяют модели восстанавливать распределение данных.

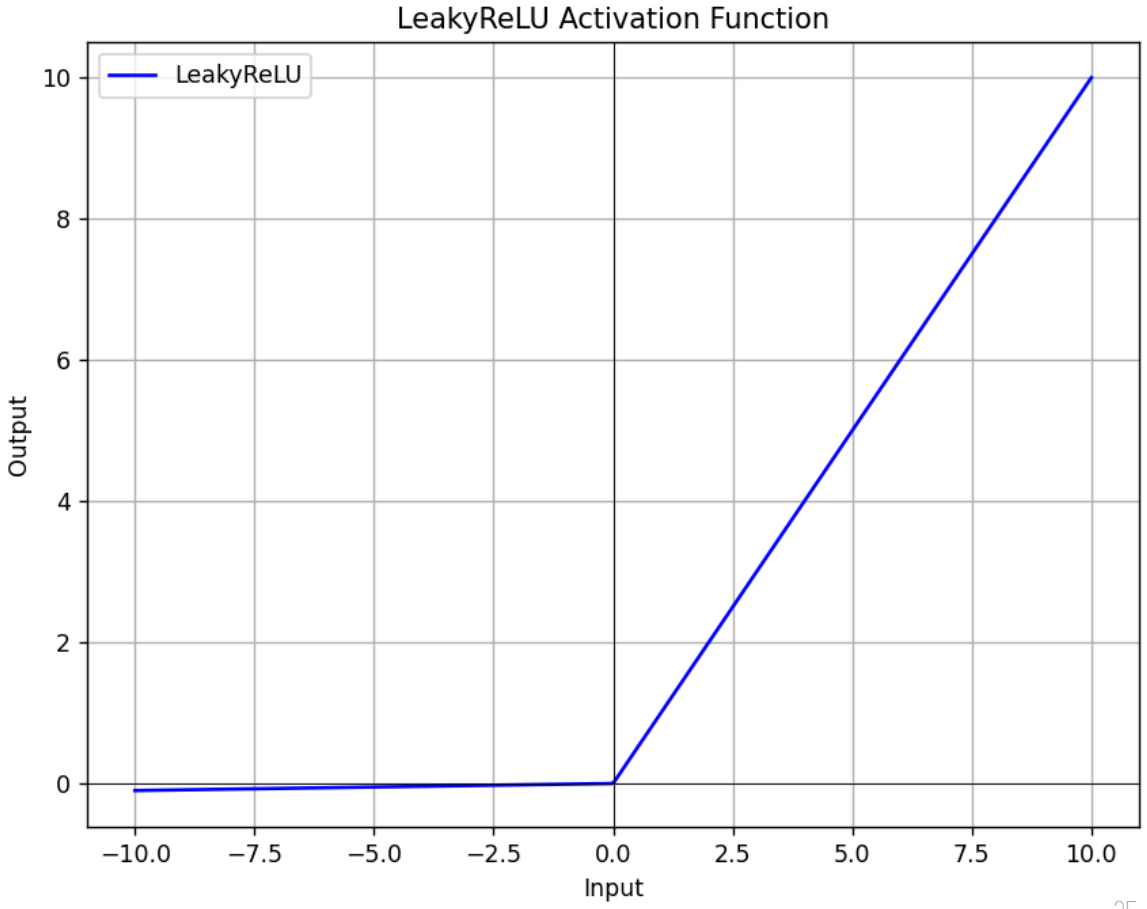
Построение нейросетевой модели

Активационная функция Leaky ReLU (Rectified Linear Unit):

$$f(x) = \begin{cases} x & \text{если } x > 0 \\ \alpha x & \text{если } x \leq 0 \end{cases}$$

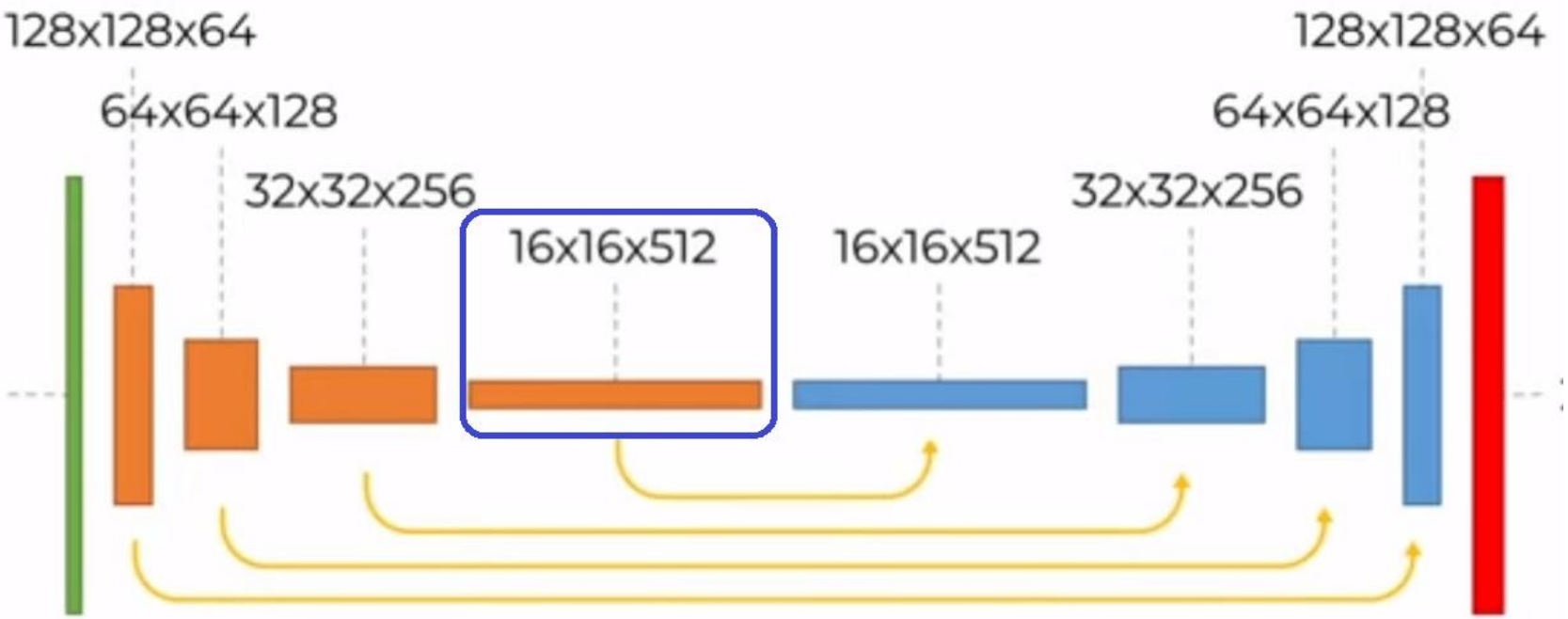
Где α — небольшой коэффициент (например, 0.01), позволяющий небольшое отрицательное значение вместо нуля, чтобы избежать проблемы умирающих нейронов.





Построение нейросетевой модели

На выходе из энкодера: тензор признаков, который является сжатием исходного изображения. Этот тензор содержит высокоуровневые абстрактные признаки, извлеченные из исходного изображения, и имеет уменьшенные пространственные размеры по сравнению с входным изображением, но увеличенное количество каналов (или глубину).



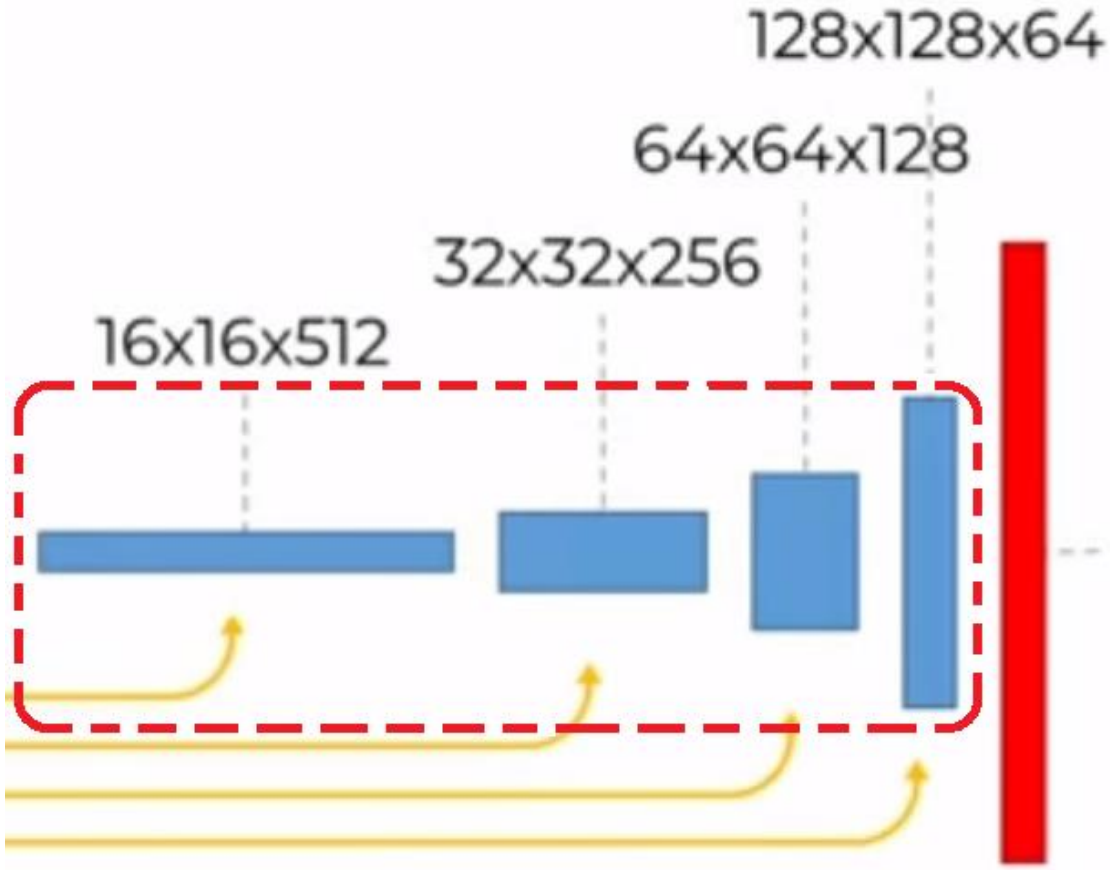
Построение нейросетевой модели

Декодер:

- поэтапно восстанавливает оригинальные размеры изображения
- использует признаки, извлечённые энкодером и детализирует их, чтобы получить конечный результат сегментации

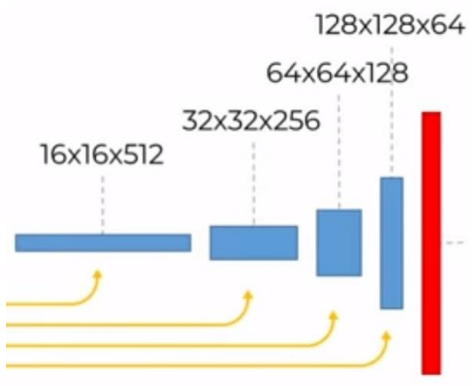
Этапы:

- развёртка
- пакетная нормализация (как в энкодере)
- функции активации



Построение нейросетевой модели

Развёртка: операция транспонированной свёртки или обратная свёртка, увеличивает пространственные размеры тензора за счёт применения фильтров и шага (strides), обратных обычной свёртке. Формула выходного тензора:



$$H_{out} = (H_{in} - 1) \times S - 2 \times P + K$$
$$W_{out} = (W_{in} - 1) \times S - 2 \times P + K$$

где:

- H_{out} и W_{out} — высота и ширина выходного тензора соответственно.
- H_{in} и W_{in} — высота и ширина входного тензора соответственно.
- S — шаг (stride).
- P — паддинг (padding).
- K — размер фильтра (kernel size).

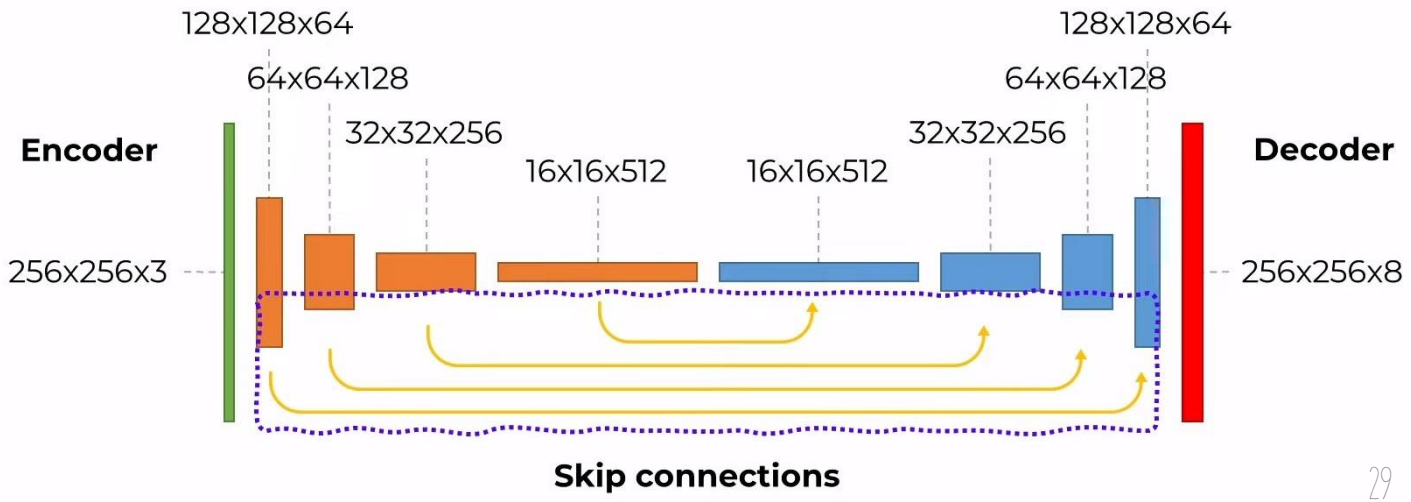
Построение нейросетевой модели

Skip connections - механизм, который позволяет передавать информацию о пространственных деталях из энкодера в декодер

После каждого слоя энкодера добавляется соответствующий слой декодера.

Информация с этого слоя энкодера объединяется с данными декодера, чтобы передать детальную пространственную информацию.

Это позволяет декодеру использовать контекст из энкодера, одновременно сохраняя информацию о пространственной структуре.

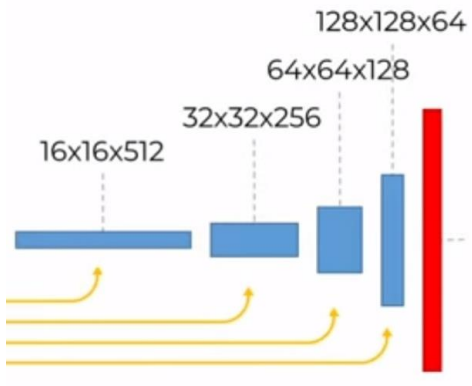


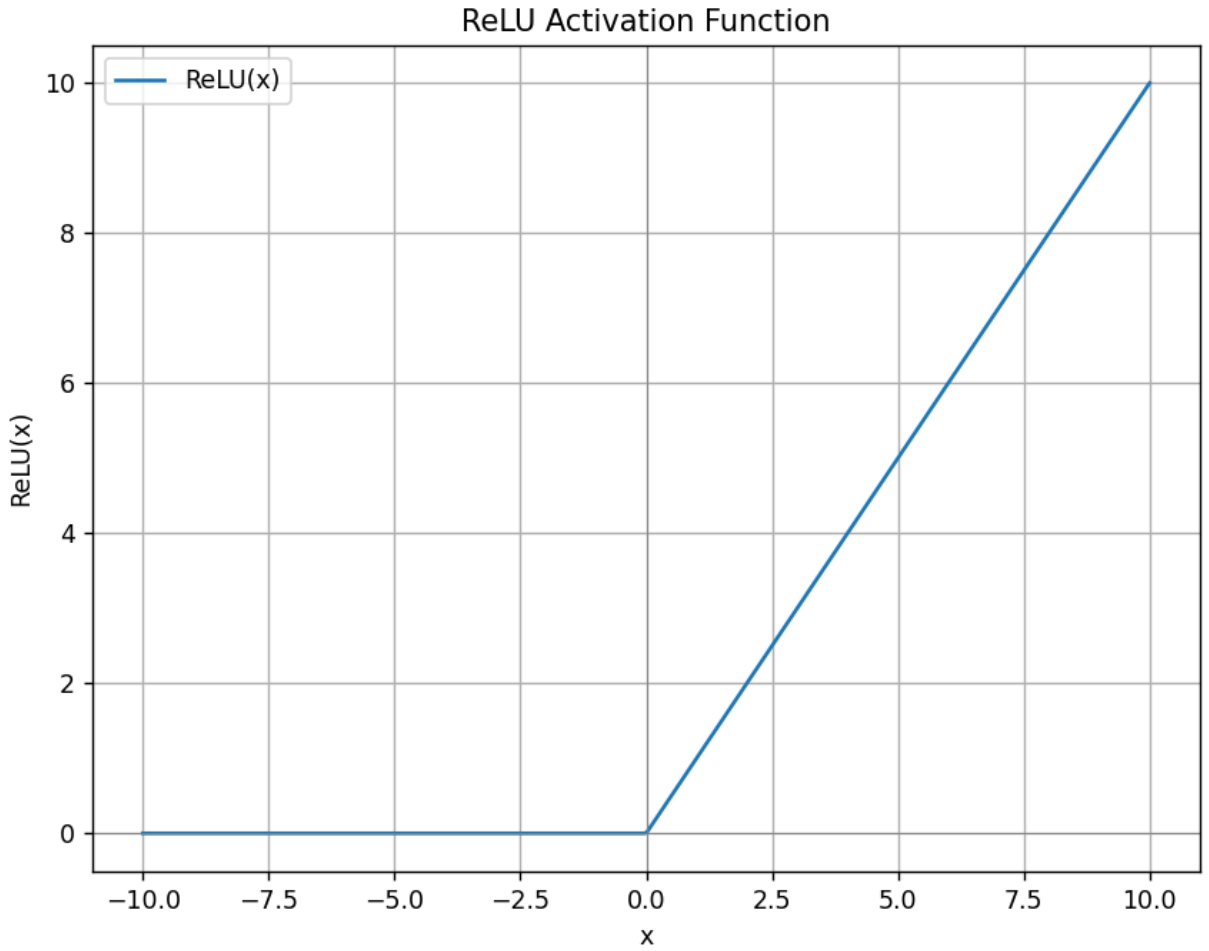
Построение нейросетевой модели

Активационная функция ReLU

Активирует только положительные значения, что помогает модели лучше схватывать основные структуры и формы на изображениях:

$$\mathrm{ReLU}(x) = \max(0, x)$$



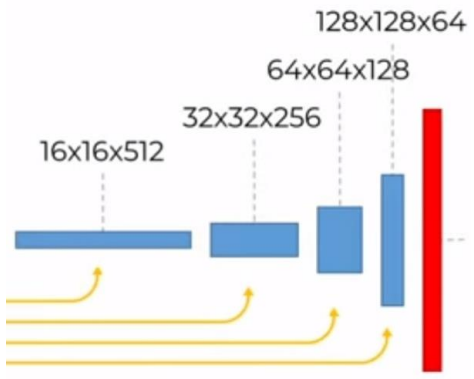


Построение нейросетевой модели

В последнем слое декодера в качестве активационной функции используется sigmoid

Необходимо для генерации выходного тензора, который представляет собой вероятности принадлежности каждого пикселя к одному из классов

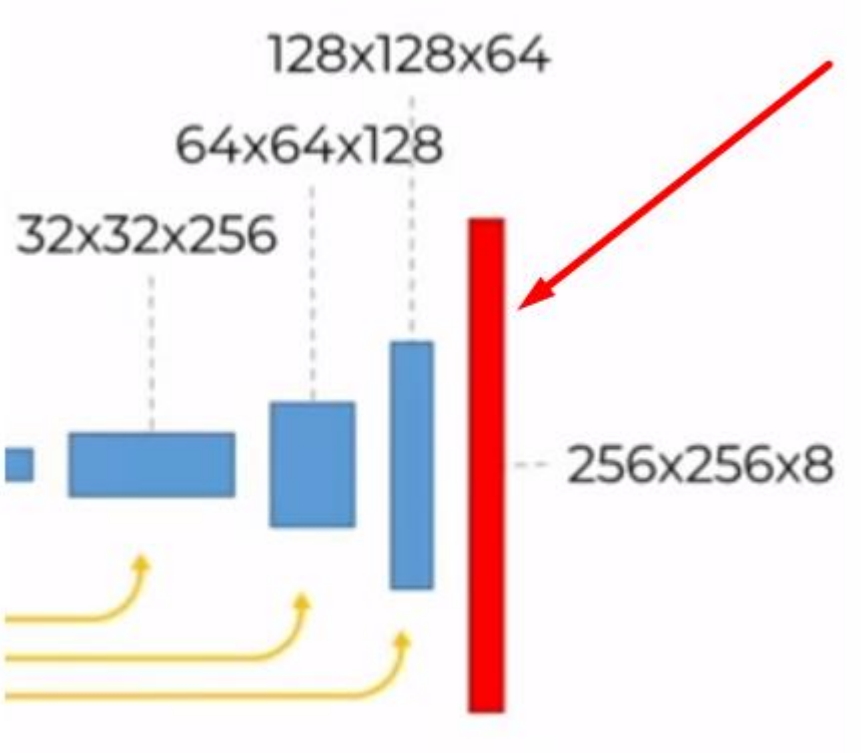
Эта функция преобразует любое входное значение в диапазон от 0 до 1, что удобно для задач, где нужно предсказать вероятность принадлежности к какому-либо классу.



$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

где x - входное значение.

Построение нейросетевой модели



Выходной слой - транспонированный свёрточный слой.

- Производит карту предсказаний, которая соответствует каждому пикселю входного изображения.
- Выходной слой U-Net генерирует тензор с теми же пространственными размерами, что и входное изображение, но с числом каналов, равным количеству классов в задаче сегментации. Например, для задачи сегментации с 8 классами выходной тензор будет иметь размер $(H,W,8)$, где H и W - высота и ширина входного изображения.
- Увеличивает размеры пространственных признаков до размеров исходного изображения.

Построение нейросетевой модели

Модель построена.

Перед обучением определимся с метриками оценки качества работы модели

Тестирование нейросетевой модели

Метрики качества работы нейросети при сегментации изображений:

Бинарная кросс-энтропия (BCE): стандартная функция потерь. Она вычисляет расхождение между истинными метками и предсказанными вероятностями. В контексте сегментации масок каждому пикселю в маске присваивается вероятность принадлежности к классу. BCE минимизирует разницу между этими вероятностями и фактическими метками.

$$BCE(y, \hat{y}) = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}))$$

Коэффициент Сёренсена (Dice coefficient): вычисляет сходство между истинными масками A и предсказанными масками B. Это метрика сходства между двумя выборками. В задачах сегментации она измеряет сходство между предсказанными масками и истинными масками. Он вычисляется как удвоенное отношение площади пересечения к общей площади двух масок.

$$Dice(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}$$

Здесь $|A|$ и $|B|$ обозначают суммарное количество пикселей в масках A и B, а $|A \cap B|$ - количество пикселей, которые одновременно принадлежат обеим маскам.

Тестирование нейросетевой модели

Комбинация BCE и Dice

Позволяет балансировать две важные характеристики:

- Острота границ и точность пикселей: BCE обычно лучше работает для точного совпадения пикселей, особенно на границах объектов.
- Сегментационная полнота и чувствительность к малым объектам: Dice coefficient хорошо подходит для оценки полноты сегментации и часто более чувствителен к малым объектам или фрагментам.

$$loss = BCE + w \cdot Dice$$, где w - вес функции потерь Dice

В данной работе использовались метрики:

- **Dice**
- **$loss = BCE + 0.3 \cdot Dice$**

Обучение нейросетевой модели

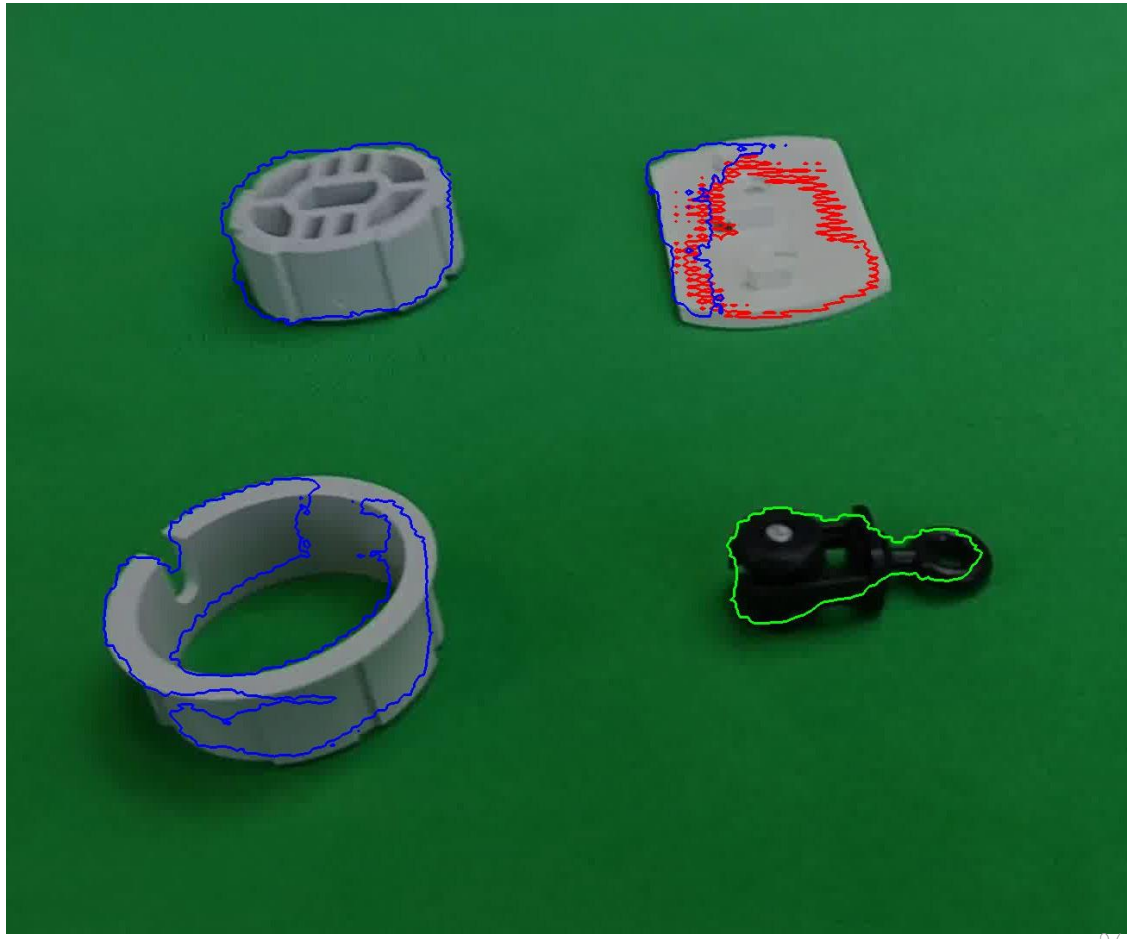
Первоначальное количество итераций обучения – 25.

Метрики отслеживались по мере обучения для оперативного принятия решения об остановке обучения.

Результат обучения после 1-й итерации:

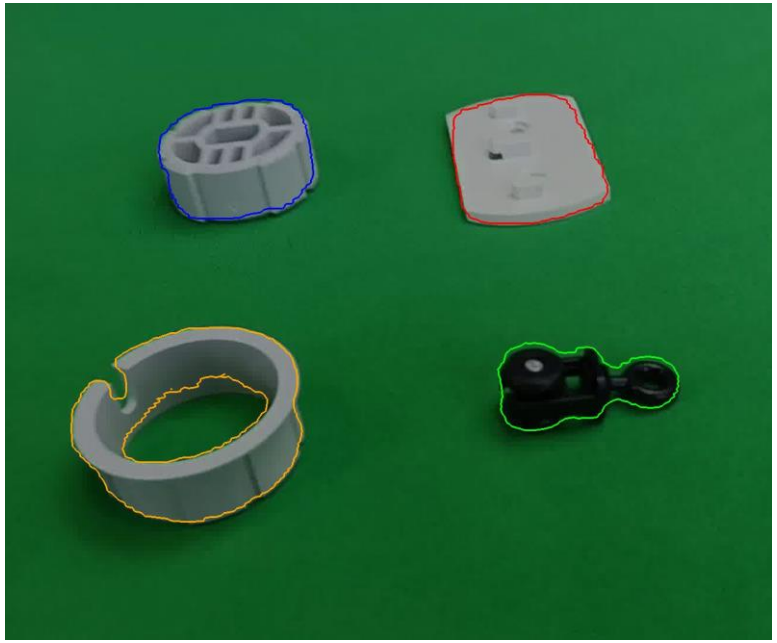
Dice = 0.3022

Loss = 0.3382



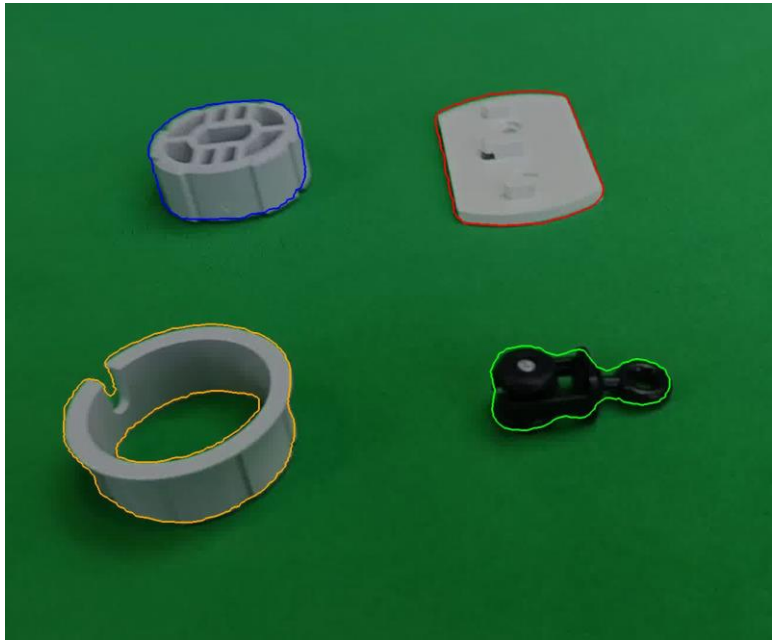
Обучение нейросетевой модели

Результаты обучения после 2 и 3 итераций



Dice = 0.7329

Loss = 0.0875



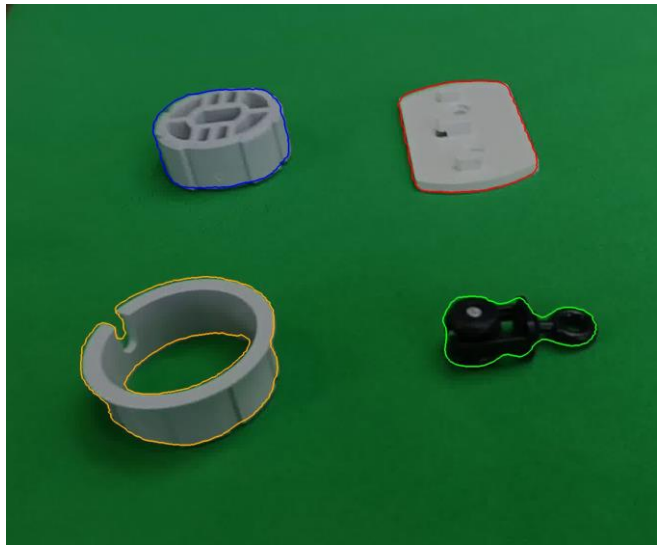
Dice = 0.9773

Loss = 0.0111

Обучение нейросетевой модели

Результаты обучения после 4 - 15 итераций

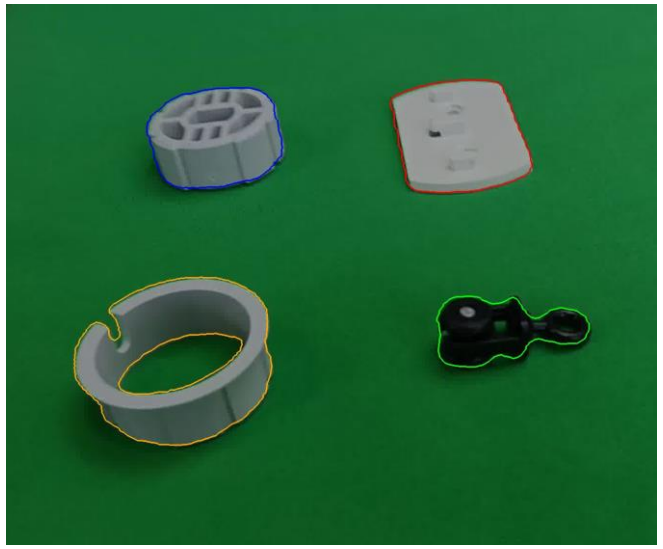
Итерация 4



Dice = 0.9845

Loss = 0.0075

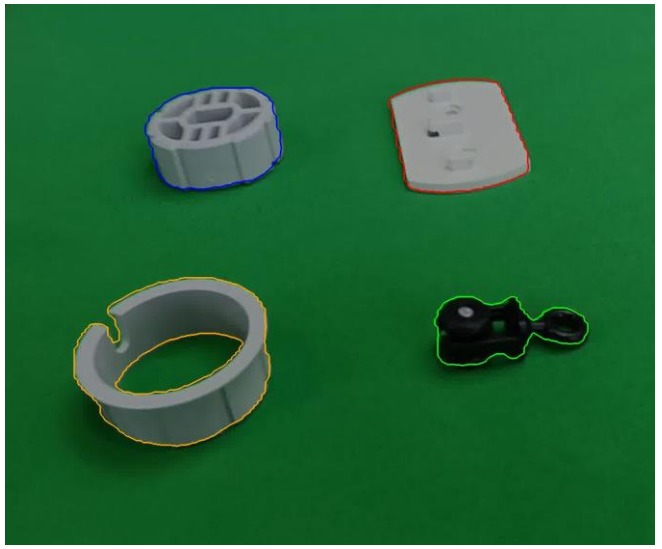
Итерация 10



Dice = 0.9912

Loss = 0.0044

Итерация 15



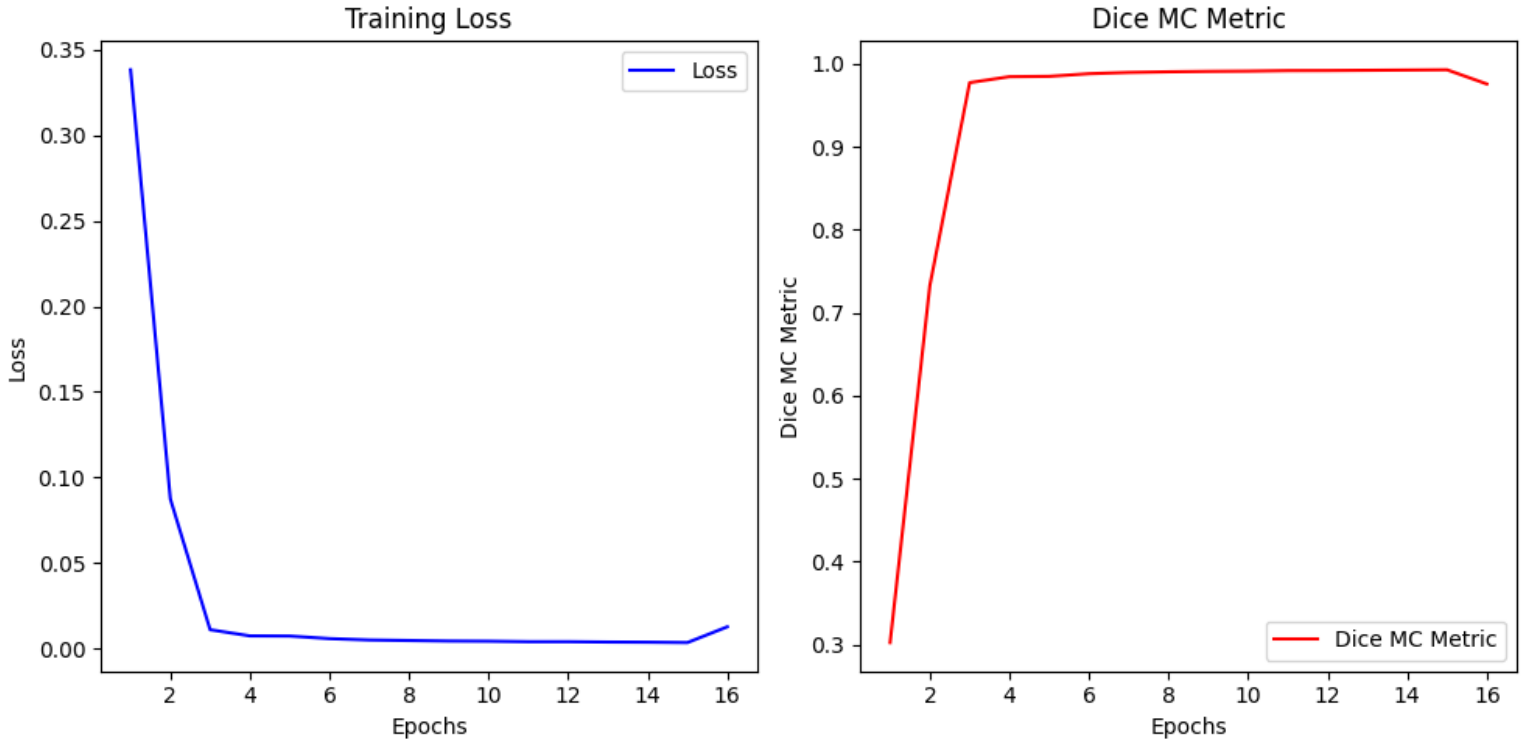
Dice = 0.9928

Loss = 0.0036

Обучение нейросетевой модели

После быстрого улучшения метрик в районе 3 - 4 итерации - перегиб графиков.

В районе 15 итерации переобученность модели. Обучение остановлено.



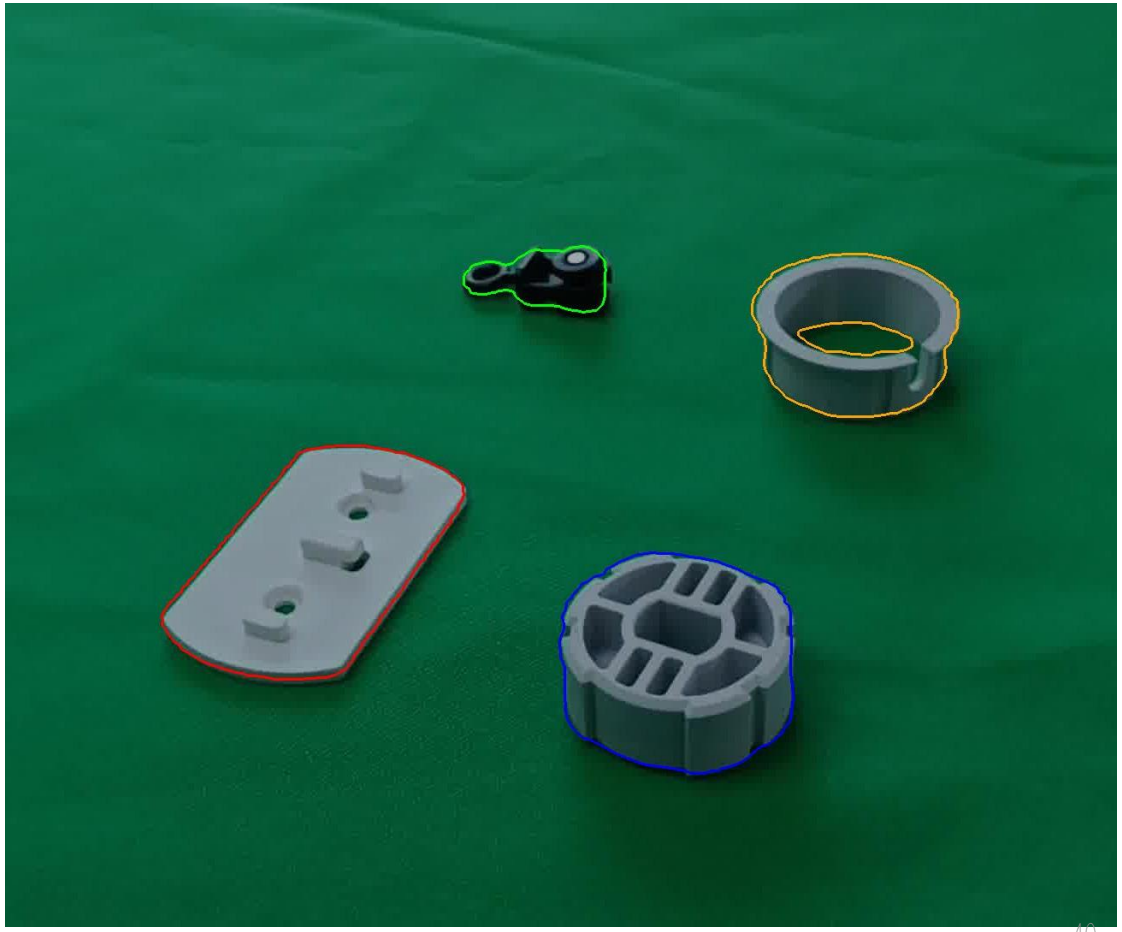
Тестирование нейросетевой модели

Тестирование модели на кадрах видео, не вошедших в обучающий набор.

Целевые значения метрик:

Dice ≥ 0,95

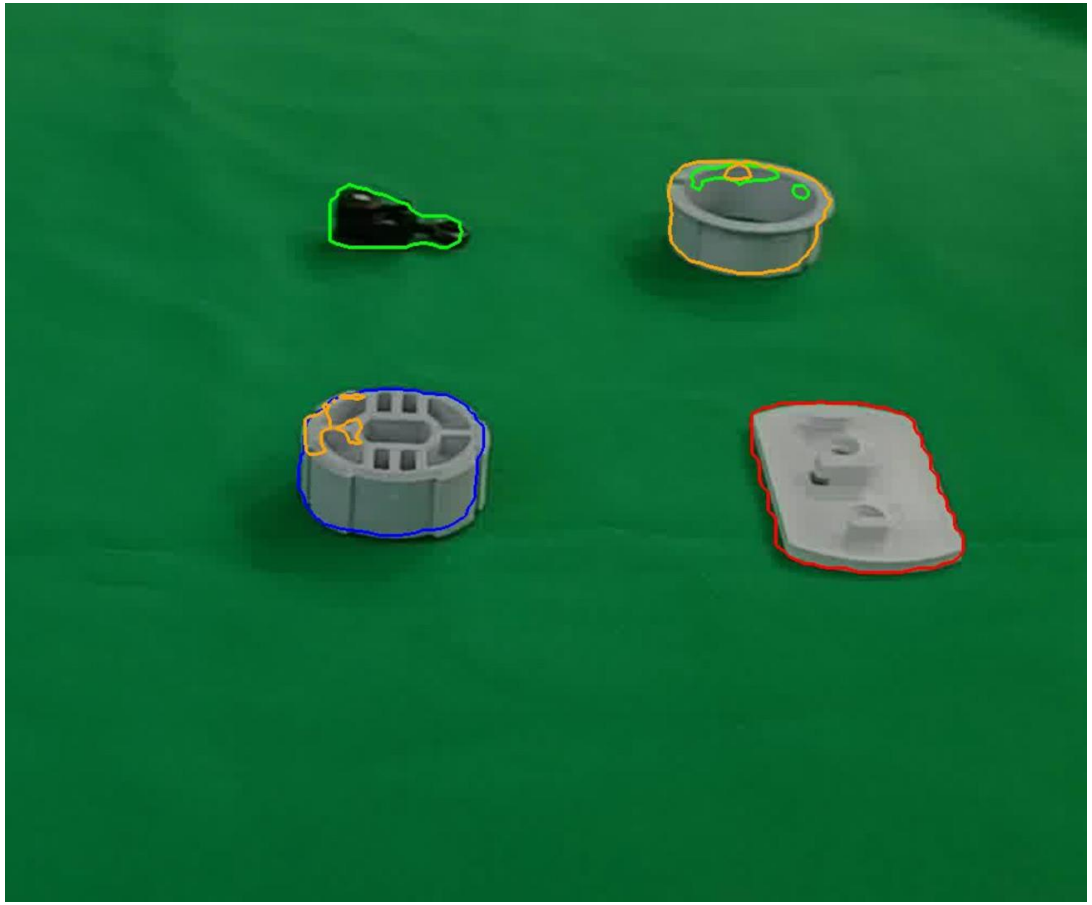
Loss ≤ 0,05



Тестирование нейросетевой модели

Тестирование модели на новом видео, которое модель ещё «не видела».

В обучающий набор добавлены несколько размеченных вручную изображений из нового видео. При этом нейронная сеть обучается не с нуля, достаточно продолжить обучение ранее сохраненной модели.

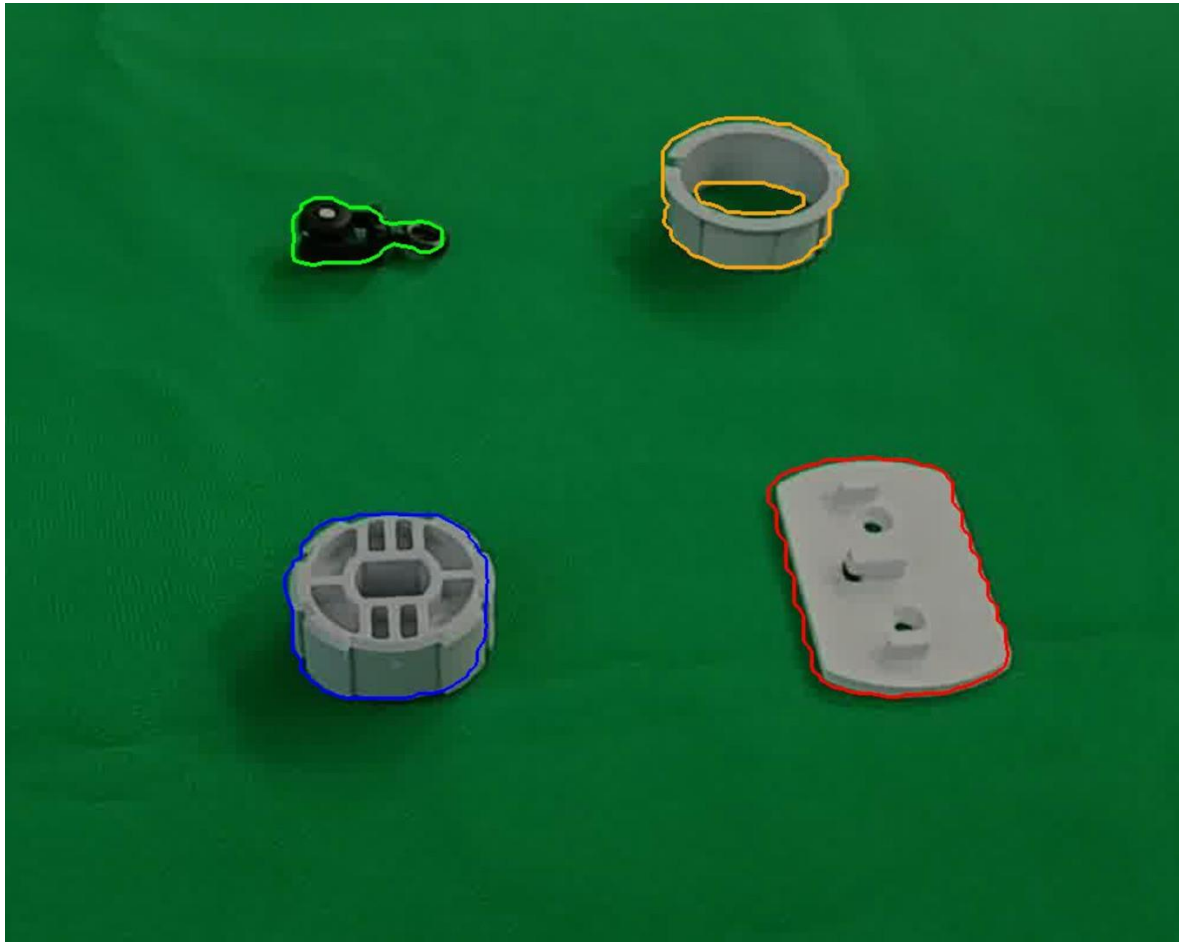


Тестирование нейросетевой модели

Тестирование модели на новом видео после дообучения.

Dice = 0.9758

Loss = 0.0119



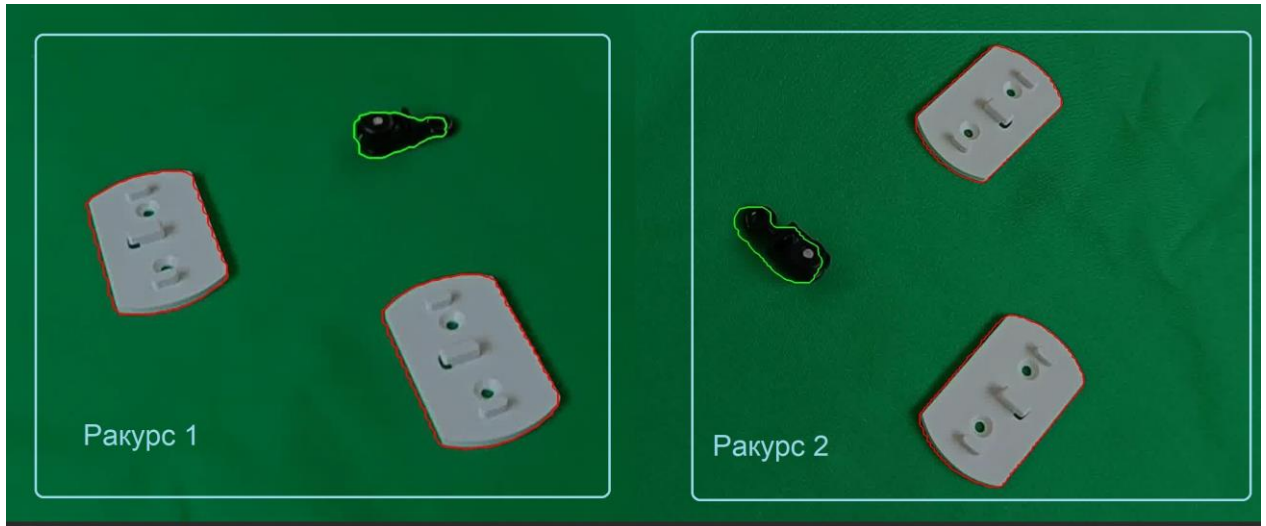
Тестирование нейросетевой модели

Тестирование модели на новом видео с меньшим количеством предметов.

Dice = 0.9864

Loss = 0.0066

Итоги сверки с ложной спецификацией



Наименование	Цвет контура	Кол-во в спец.	Кол-во факт.	Добавить	Убавить
Адаптер	голубой	1	0	1	0
Бегунок	зелёный	2	1	1	0
Кольцо	оранжевый	0	0	0	0
Пластина	красный	1	2	0	1

РЕЗУЛЬТАТ: есть ошибки

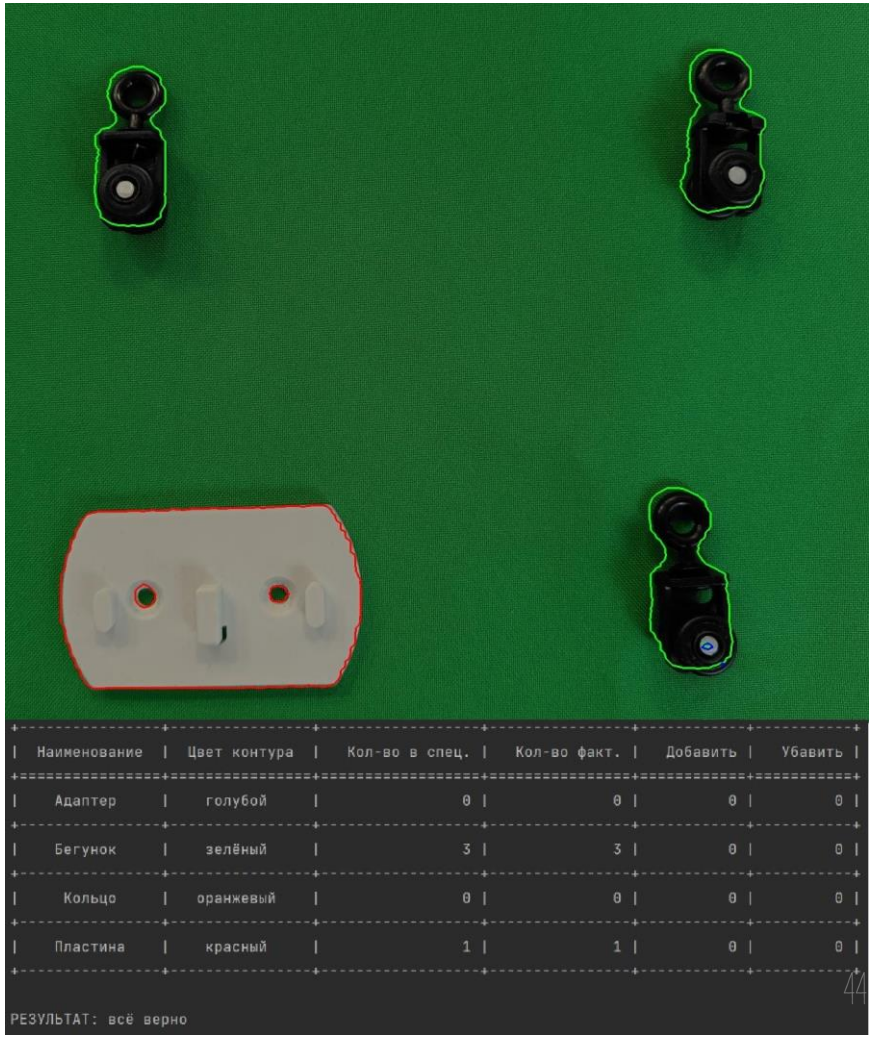
Тестирование нейросетевой модели

Тестирование модели на новом видео с таким же количеством предметов, но в другом составе.

Dice = 0.9883

Loss = 0.0058

Итоги сверки с истинной спецификацией



Наименование	Цвет контура	Кол-во в спец.	Кол-во факт.	Добавить	Убавить
Адаптер	голубой	0	0	0	0
Бегунок	зелёный	3	3	0	0
Кольцо	оранжевый	0	0	0	0
Пластина	красный	1	1	0	0

РЕЗУЛЬТАТ: всё верно

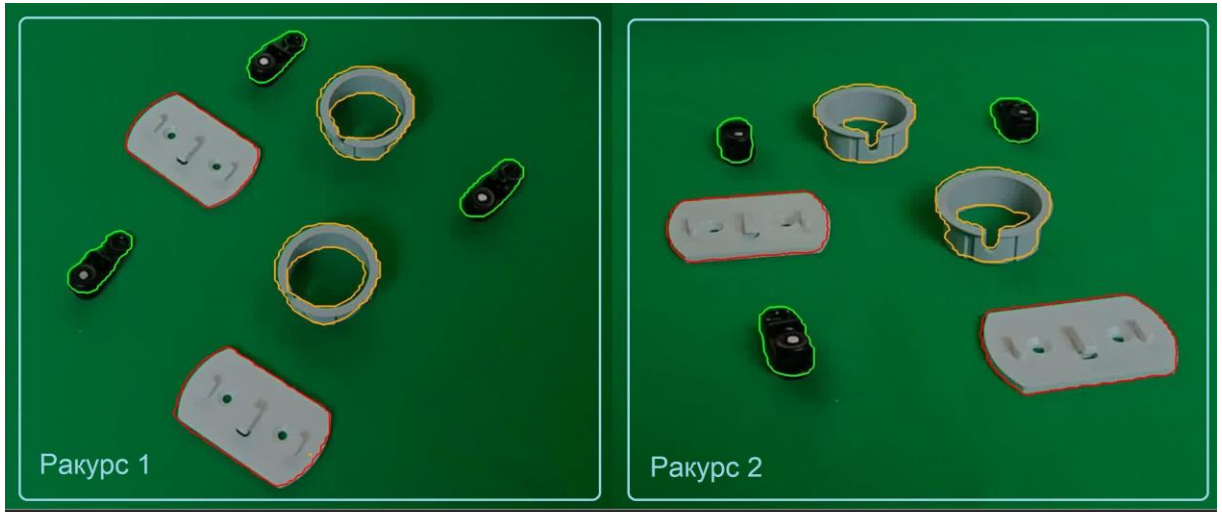
Тестирование нейросетевой модели

Тестирование модели на новом видео с б**о**льшим количеством предметов.

Dice = 0.9799

Loss = 0.0098

Итоги сверки с истинной спецификацией



Наименование	Цвет контура	Кол-во в спец.	Кол-во факт.	Добавить	Убавить
Адаптер	голубой	0	0	0	0
Бегунок	зелёный	3	3	0	0
Кольцо	оранжевый	2	2	0	0
Пластина	красный	2	2	0	0

РЕЗУЛЬТАТ: всё верно

Заключение

В процессе работы были выполнены задачи:

- Разработана нейросетевая модель для распознавания объектов в архитектуре U-Net.
- Определены метрики качества работы модели и целевые значения.
- Создан программный модуль для сверки результатов распознавания со спецификациями изделий.
- Нейросетевая модель успешно обучена и применена на практике, получены результаты выше целевых.

Заключение

Возможные ограничения и пути решения:

Недостаточное качество распознавание мелких предметов.

Решение: использование более мелких фильтров, многошаговой сегментации. Требуется больше ресурсов.

Качество работы модели существенно зависит от качества набора данных.

Решение: регламентирование процесса разметки, например – съёмку проводить в условиях, идентичных «боевым».