

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Южный федеральный университет»

Институт математики, механики  
и компьютерных наук им. И. И. Воровича

Кафедра информатики и вычислительного эксперимента

**Дроздов Дмитрий Сергеевич**

**РЕШЕНИЕ ЗАДАЧИ СЕГМЕНТАЦИИ НЕКОТОРЫХ ТИПОВ  
ОБЛАКОВ ПО ИХ СПУТНИКОВЫМ СНИМКАМ**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
по направлению подготовки  
02.03.02 – Фундаментальная информатика и информационные  
технологии

**Научный руководитель –**  
доцент, к. ф.-м. н. Абрамян Анна Владимировна

Допущено к защите:  
заведующий кафедрой \_\_\_\_\_ Михалкович С. С.

Ростов-на-Дону – 2021

# Содержание

Введение . . . . .	4
1. Постановка задачи . . . . .	6
2. Обзор существующих решений . . . . .	7
3. Обзор данных . . . . .	9
4. Используемые инструменты . . . . .	13
4.1. Google Colab . . . . .	13
4.2. Библиотека для сегментации . . . . .	13
5. Предобработка изображений . . . . .	15
6. Описание использованных моделей . . . . .	20
6.1. Сеть U-Net . . . . .	20
6.2. Сеть Feature Pyramid Network (FPN) . . . . .	22
6.3. Кодировщик ResNet-101 . . . . .	23
6.4. Кодировщик Inception-v3 . . . . .	24
7. Настройка моделей . . . . .	28
7.1. Callback-функции . . . . .	31
7.2. Эпохи . . . . .	32
7.3. Аугментации . . . . .	33
8. Обучение модели и предсказания . . . . .	40

9. Описание полученных результатов . . . . .	43
10. Постобработка . . . . .	46
11. Сравнение моделей . . . . .	49
11.1. Влияние постобработки . . . . .	50
11.2. Сравнение кодировщиков . . . . .	51
11.3. Сравнение архитектур сети . . . . .	52
11.4. Влияние предобработки . . . . .	53
11.5. Сравнение аугментаций . . . . .	54
12. Объединение предсказаний . . . . .	57
13. Восстановление маски до прямоугольника . . . . .	59
14. Результаты проведенного исследования . . . . .	61
Заключение . . . . .	63
Список литературы . . . . .	64

# Введение

Институтом метеорологии Макса Планка совместно с Kaggle предложена задача распознавания структур облаков на спутниковых снимках [1]. Для решения задачи необходимо построить алгоритм, производящий сегментацию снимков. Полученные результаты могут помочь создать новые климатические модели и увеличить точность прогнозирования погоды.

Для сегментации спутниковых снимков облаков в работе используются сверточные сети, такие как U-Net и FPN (Feature Pyramid Network) с кодировщиками ResNet и Inception.

Сверточные нейронные сети (Convolutional Neural Network — CNN) появились в результате изучения зрительной коры головного мозга и применяются с 1980-х годов. Последние несколько лет проводятся соревнования по распознаванию визуальных образов ImageNet Large Scale Visual Recognition Challenge (ILSVRC), на которых представляются новые архитектуры сетей.

В данной работе был реализован следующий план решения задачи распознавания типов облаков на снимках из космоса:

1. Предобработка изображений.
2. Настройка моделей.
3. Обучение различных моделей.
4. Постобработка полученных масок.

5. Сравнение результатов работы моделей.

6. Объединение предсказаний.

В работе дан сравнительный анализ использованных моделей, оценено влияние примененных способов обработки изображений на качество сегментации и указана наиболее подходящая, с точки зрения автора, последовательность действий для решения данной задачи.

## **1. Постановка задачи**

Исследовать возможность и эффективность использования методов машинного обучения для решения задачи сегментации типов облаков на спутниковых снимках.

Для решения поставленной задачи выполнить следующие этапы исследования:

1. обучить несколько моделей и сравнить результаты их предсказаний;
2. исследовать и оценить эффективность возможных вариантов обработки полученных предсказаний;
3. проанализировать возможность и целесообразность использования ансамблей.

## 2. Обзор существующих решений

Существует несколько методов решения задачи сегментации снимков облаков. Например, в статье [2] описан метод обнаружения облаков на снимках Landsat, основанный на физических правилах. Данный метод распознает облака по их физическим свойствам: они белые и яркие. Однако алгоритм заточен под распознавание облаков в принципе, а не выделение облаков заданного типа. Кроме того, для предложенного авторами метода требуются такие данные, как температура и высота, отсутствующие в данной задаче.

В то же время в статье [3] для решения задачи используется нейронная сеть с глубокой пирамидальной архитектурой и показывается, что сверточные сети в данной задаче работают лучше, чем методы, основанные на правилах. В работе также экспериментально доказано, что использование предобученной сети увеличивает качество модели.

В статье [4] использованы сети U-Net и Feature Pyramid Network (FPN), а также кодировщики группы ResNet, EfficientNet и DenseNet. Кроме того, в работе выполнен экспериментальный подбор порога бинаризации предсказаний сети и порога отсечения небольших масок. Для достижения наилучшего результата авторы использовали ансамбли моделей.

В статье [5] задача сегментации решалась с помощью сети U-Net и кодировщиков группы EfficientNet, объединенных в ансамбли. В работе авторы отслеживали PR-кривую, демонстрирующую

связь между точностью (precision) и полнотой (recall). Модели настраивались экспериментально так, чтобы максимизировать площадь под кривой и улучшить тем самым качество предсказания. Для того чтобы разнообразить обучающее множество и увеличить обобщающие способности нейросети, авторы использовали такие преобразования изображений, как отражения, вращения и искажения по сетке.

### 3. Обзор данных

В данной работе используется обучающая выборка, состоящая из 5546 изображений, и тестовая выборка, в которой 3698 изображений (см. рис. 1) [1]. Соотношение обучающей выборки и тестовой составляет 3 : 2. Все спутниковые снимки имеют размер  $2100 \times 1400$  пикселей и глубину цвета 24 бита на пиксель. Объем обучающей выборки составляет 3,45 Гб, тестовой — 2,3 Гб.

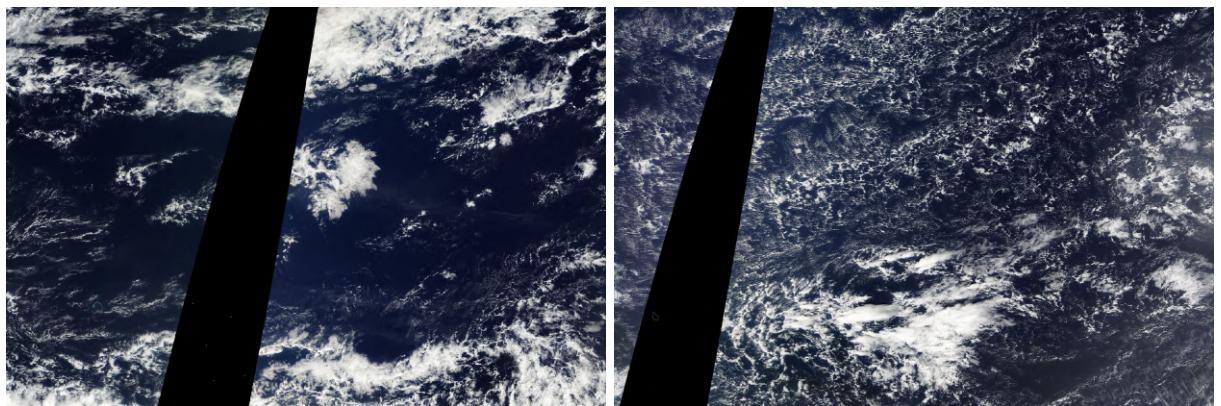


Рисунок 1. Изображения из тестовой выборки

Исходные данные, взятые с сервисов НАСА Worldview, получены с помощью двух орбитальных спутников. Ввиду небольшого размера сканера изображение склеивалось из двух кадров, при этом непокрытая область заполнялась черным цветом. Кроме того, на некоторых снимках присутствуют блики. На рисунке 2 приведен пример изображения с указанными особенностями.

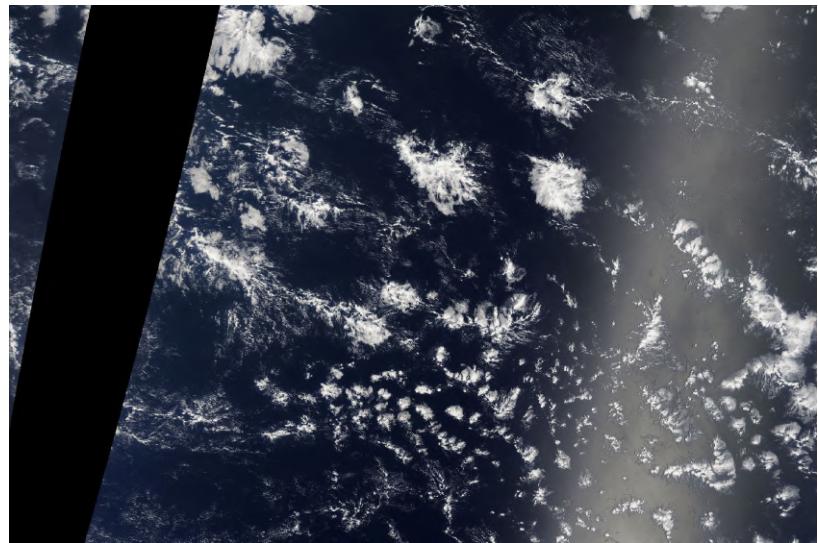
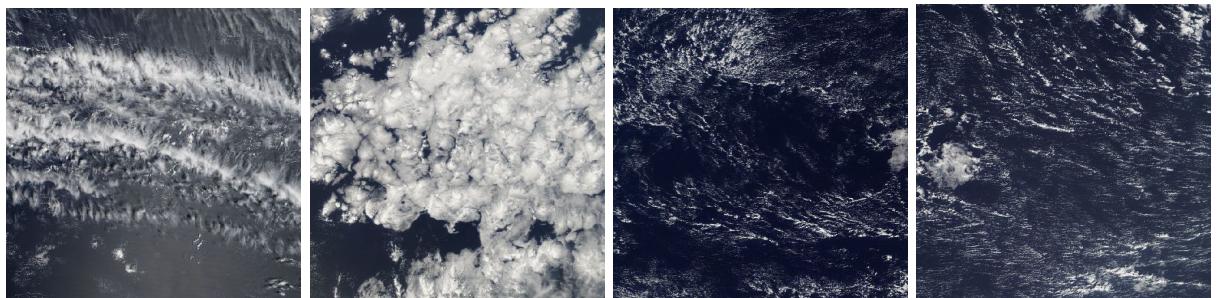


Рисунок 2. Пример спутникового снимка

Авторами задачи выделено 4 типа облаков: Fish (рыба), Flower (цветок), Gravel (гравий) и Sugar (сахар). Примеры облаков каждого типа приведены на рисунке 3.



Fish

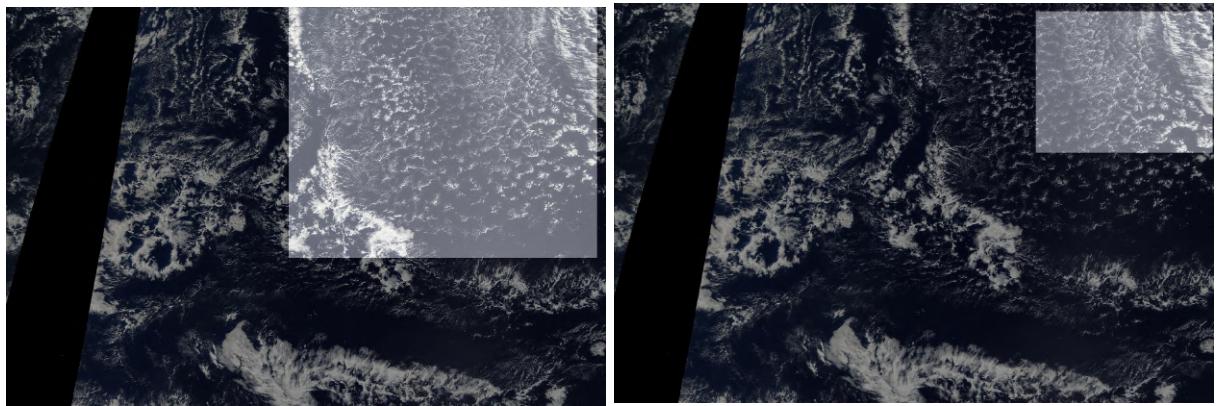
Flower

Gravel

Sugar

Рисунок 3. Четыре типа облаков

Видно, что изображения Gravel и Sugar очень похожи. Действительно, в обучающей выборке пересечение областей этих двух классов составляет около 22 % (см. рис. 4), в то время как, например, маски для Fish и Flower совпадают всего на 9 %.



Gravel

Sugar

Рисунок 4. Пересечение двух масок

Для изображений из обучающего множества дана разметка в виде таблицы, в которой каждый снимок представлен четыре раза: в отдельных строках указывается имя файла с меткой и маска. Область пикселей (маска) закодирована с помощью RLE. Одна строка, представляющая маску, состоит из пар вида  $(s, l)$ , где  $s$  — номер пикселя,  $l$  — протяженность; изображение обходится по столбцам сверху вниз. Ниже (см. таблицу 1) приведен фрагмент разметки.

Таблица 1. Фрагмент размеченных данных

Метка изображения	Закодированные пиксели
0011165.jpg_Fish	264918 937 266318 937 267718 937 269118 937 ...
0011165.jpg_Flower	1355565 1002 1356965 1002 1358365 1002 ...
0011165.jpg_Gravel	NaN
0011165.jpg_Sugar	NaN
002be4f.jpg_Fish	233813 878 235213 878 236613 878 238010 ...

Например, на изображении 0011165.jpg выделены две области: Fish и Flower (см. рис. 5). Другие типы облаков на этом изображении отсутствуют (отметка NaN).



Область Fish

Область Flower

Рисунок 5. Области, выделенные на снимке 0011165.jpg

В обучающем множестве классы хорошо сбалансиированы, что облегчает процесс обучения модели. Ниже приведена диаграмма распределения долей типов облаков.

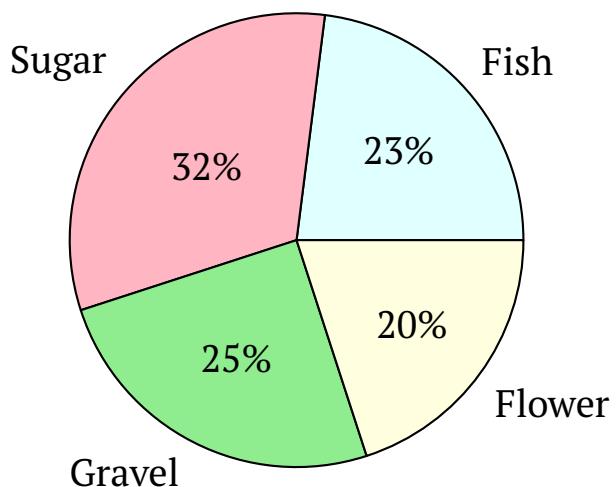


Рисунок 6. Диаграмма распределения долей типов облаков

## **4. Используемые инструменты**

### **4.1. Google Colab**

Для ускорения вычислений в работе был использован облачный сервис Google Colab [6]. Предоставляемая виртуальная машина имеет объем ОЗУ 25 Гб и объем диска 174 Гб. Кроме того, сервис позволяет использовать мощную видеокарту Tesla P100. В данной работе все вычисления переложены на GPU, что позволяет сократить время работы более чем в 50 раз по сравнению с вычислениями на CPU.

Единственным недостатком Colab является то, что при отключении от машины сбрасываются все данные. Однако данный сервис связан с Google Диском, что позволяет оперативно загружать файлы обучающего и тестового наборов. Наиболее быстрый способ заключается в копировании архива с Google Диска и его разархивации ресурсами Colab. Процессор AMD EPYC 7H12 позволяет это сделать за несколько минут, благодаря наличию 64 ядер и 128 потоков.

Точные характеристики предоставляемых видеокарт и процессоров могут отличаться от запуска к запуску, однако преимущество над домашним ПК в любом случае очевидно.

### **4.2. Библиотека для сегментации**

Работа выполнена на языке Python 3.6.9, в качестве основной библиотеки выступает Segmentation Models 1.0 [7], работающая

на основе фреймворка Keras. Библиотека предоставляет следующие возможности:

- высокоуровневое API для создания нейронной сети;
- 4 архитектуры сети (включая U-Net и FPN);
- 25 предобученных кодировщиков (включая ResNet и Inception);
- вычисления на видеокартах.

## 5. Предобработка изображений

Исходные снимки являются цветными, и некоторые из них содержат блики. Изображения были обработаны по следующему алгоритму [8]. Алгоритм переводит снимок в оттенки серого и устраняет блики. Некоторые параметры ( $T$  и  $\sigma$ ) в данной работе были экспериментально подобраны так, чтобы получить более точный фон снимков.

Шаг 1. Изображение переводится в оттенки серого путем усреднения значений трех каналов R, G, B (см. рис. 8). Значения пикселей после преобразования принадлежат отрезку [0; 255]. Исходный снимок представлен на рисунке 7.



Рисунок 7. Исходный снимок

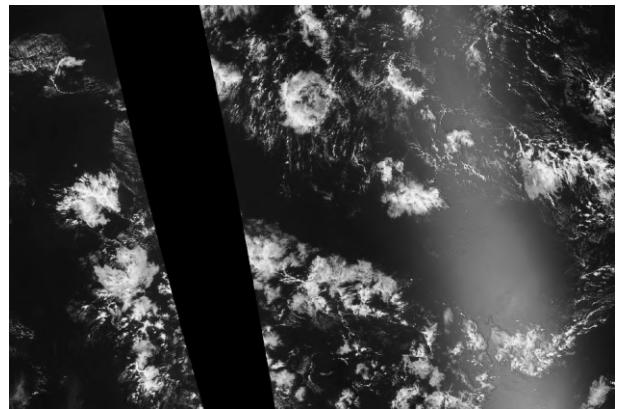


Рисунок 8. Изображение в оттенках серого

Шаг 2. Вычисляется маска, определяющая фон снимка. Для этого изображение разбивается на квадраты размером  $8 \times 8$  пикселей и считается, что квадрат принадлежит фону, если стандартное

отклонение интенсивности пикселей ниже порога  $T$  (опытным путем установлено, что  $T = 4$ ).

Если квадрат принадлежит фону, то все пиксели в нем полагаются равными 1, если не принадлежит, то 0. На рисунке 9 приведена маска, на которой белым цветом обозначен фон.

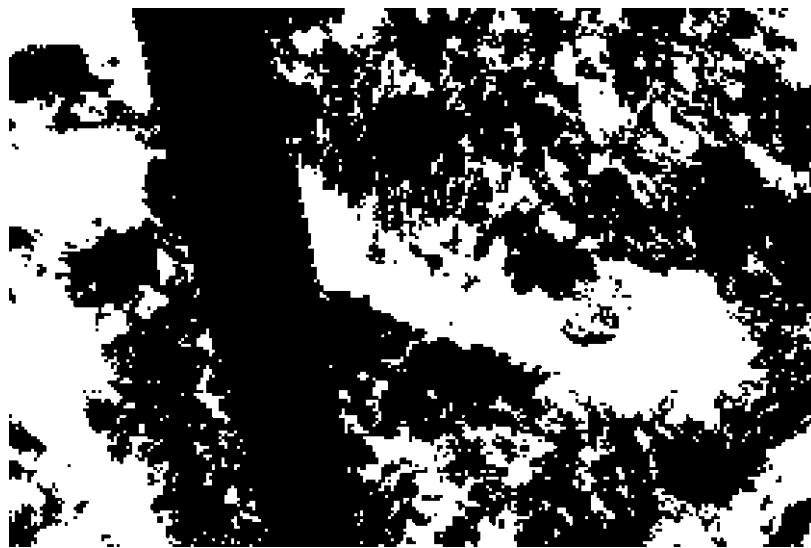


Рисунок 9. Выделение фона

Шаг 3. Изображение разбивается по сетке, состоящей из квадратов размером  $50 \times 50$ . Далее вычисляется средний цвет в каждом квадрате по пикселям, относящимся к фону. Принадлежность пикселя фону определяется по маске, полученной на прошлом шаге. При этом если все пиксели квадрата не относятся к фону, то он заливается черным цветом (см. рис. 10).

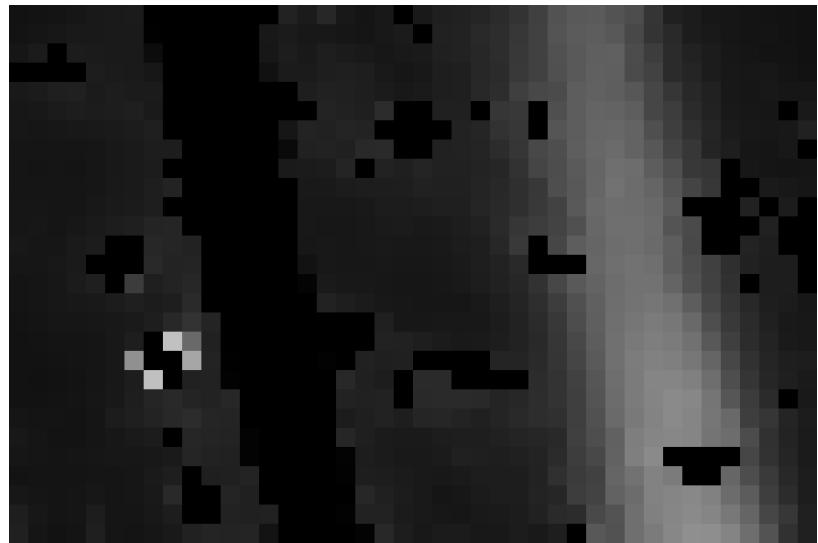


Рисунок 10. Средний цвет по квадратам  $50 \times 50$

Шаг 4. Полученные усредненные цвета сглаживаются с помощью фильтра размытия по Гауссу из библиотеки `skimage` с экспериментально подобранным параметром  $\sigma = 40$  (см. рис. 11).

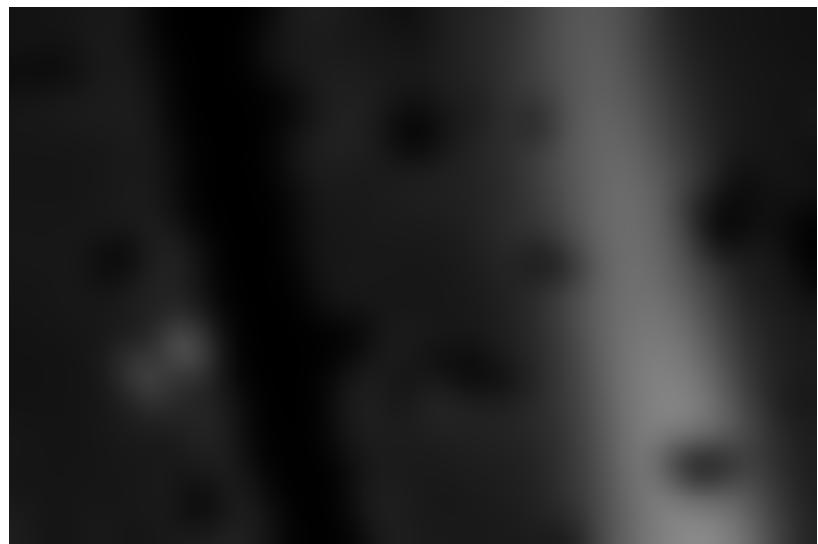


Рисунок 11. Размытие средних цветов фильтром Гаусса

Шаг 5. Изображение, полученное на предыдущем шаге, почти полностью черное, за исключением области блика. В таком слу-

чае при вычитании данного изображения из исходного блики будут удалены.

Обозначим через  $p_{orig}$  пиксели исходного изображения в оттенках серого, а  $p_{back}$  — пиксели изображения, полученного на прошлом шаге. Тогда разность изображений описывается выражением  $p_{orig} - p_{back}$ .

После вычитания производится нормирование значений пикселей. Для этого разность делится на  $255 - p_{back}$  (отнимается  $p_{back}$ , так как значение исходных пикселей также было уменьшено на эту величину).

На месте, где был черный цвет, в результате преобразований иногда получаются отрицательные значения. Для устранения этой проблемы берется максимум между вычисленным значением пикселя и нулем.

Таким образом, пиксели результирующего изображения вычисляются по формуле (1):

$$p_{res} = \max \left( \frac{p_{orig} - p_{back}}{255 - p_{back}}, 0 \right) \quad (1)$$

Исходное изображение и результат обработки представлены на рисунке 12.



Рисунок 12. Исходное изображение и результат обработки

## 6. Описание использованных моделей

Приведем краткое описание использованных в работе нейронных сетей.

### 6.1. Сеть U-Net

Сверточная сеть U-Net создана в 2015 году для сегментации биомедицинских изображений. Подробное описание структуры и применения сети дается в статье разработчиков [9].

Сеть состоит из сужающегося и расширяющегося путей. Первый путь содержит два подряд расположенных сверточных слоя  $3 \times 3$ , после которых идет слой ReLU и операция объединения по максимуму (max pooling)  $2 \times 2$  с шагом 2.

Если на входе имеется двумерное изображение  $I$  и ядро  $K$ , то операция свертки определяется по формуле (2).

$$s(i, j) = \sum_m \sum_n K(m, n)I(i - m, j - n) \quad (2)$$

Подробные теоретические сведения и примеры вычисления свертки приведены в статье [10].

Функция активации ReLU [11] определяется по формуле (3).

$$f(x) = \begin{cases} x, & \text{если } x \geq 0 \\ 0, & \text{если } x < 0 \end{cases} \quad (3)$$

Операция объединения по максимуму подробно описана в книге [12]. Ниже на рисунке 13 приведен пример применения операции max pooling. В результате такого объединения размер изображения уменьшается в два раза по каждому направлению.

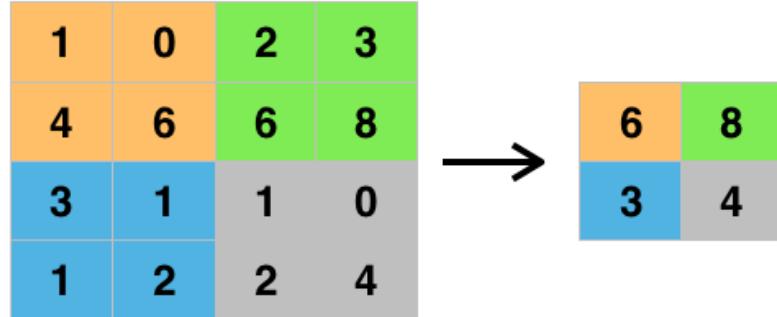


Рисунок 13. max pooling

Один шаг расширяющего пути содержит слой, обратный объединяющему слою, который расширяет карту признаков. Затем следует свертка  $2 \times 2$ , уменьшающая количество каналов признаков. После этого применяется конкатенация с обрезанной картой признаков из сжимающего пути и две свертки  $3 \times 3$  (после каждой применяется ReLU). На последнем слое свертка  $1 \times 1$  используется для приведения каждого 64-компонентного выходного вектора до требуемого количества классов.

Полная архитектура сети для изображения с разрешением  $32 \times 32$  приведена на рисунке 14 [9]. Синий прямоугольник обозначает многоканальную карту свойств (количество каналов указано в верхней части прямоугольника). Размер изображения приведен в левом нижнем углу прямоугольника. Обрезка, выполняемая перед конкатенацией, обозначается пунктиром.

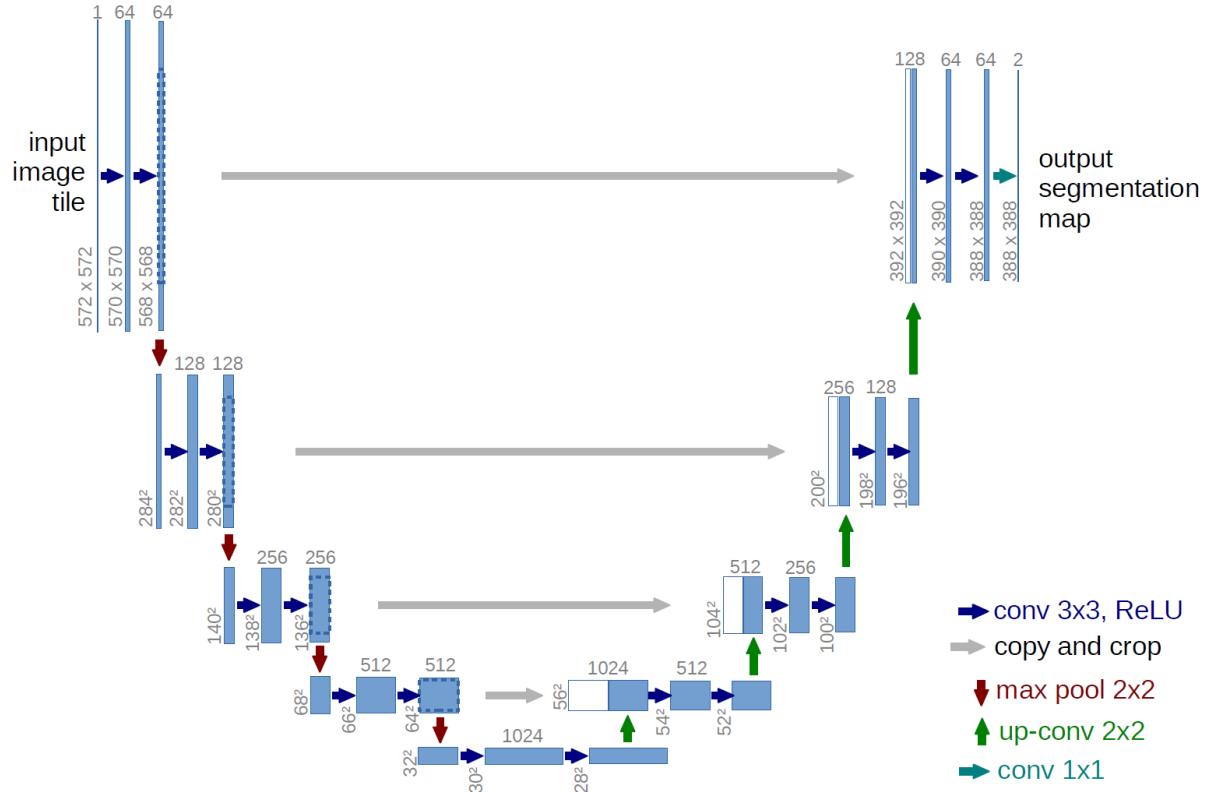


Рисунок 14. Архитектура сети U-Net

## 6.2. Сеть Feature Pyramid Network (FPN)

В Feature Pyramid Network извлеченные последовательными слоями сети с уменьшающейся размерностью карты признаков рассматриваются как иерархическая «пирамида» [13]. При этом карты признаков как нижних, так и верхних уровней обладают своими преимуществами и недостатками: первые имеют высокое разрешение, но низкую обобщающую семантическую способность, а вторые — наоборот.

В целом архитектура FPN похожа на U-Net: она содержит сжимающий путь и расширяющий, имеет «проброс» признаков с пер-

вого пути на второй. Однако в FPN для предсказания используются карты всех уровней [14]. Для этого сначала применяется две свертки  $3 \times 3$ , затем на каждом слое происходит увеличение размера так, чтобы все уровни стали одного размера (верхний уровень увеличивается в 8 раз, следующий — в 4 раза и т.д.). На рисунке 15 приведена схема архитектуры сети FPN.

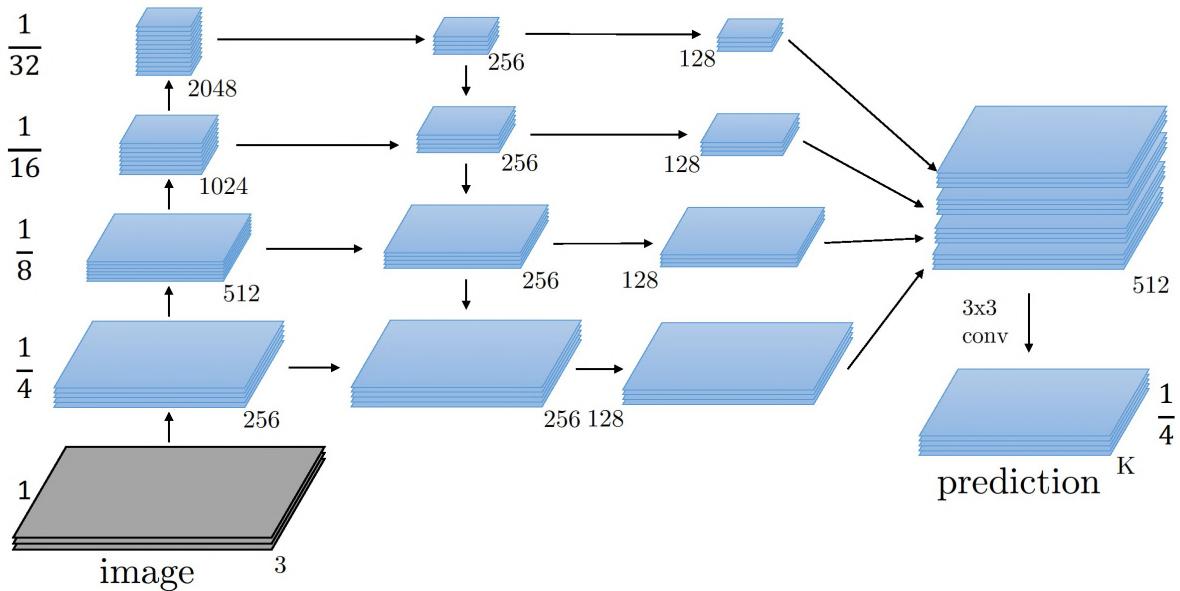


Рисунок 15. Архитектура сети FPN

### 6.3. Кодировщик ResNet-101

Архитектуры, описанные выше, содержат сужающий (сжимающий) путь, который извлекает признаки из изображения. Часто этот путь заменяется на различные кодировщики. Например, в данной работе используется остаточная сеть ResNet-101, которая стала победителем в решении задачи ILSVRC в 2015 году [15]. Ключом к победе стало использование так называемых сокращенных (или об-

ходящих) связей: сигнал, передаваемый в слой, также добавляется к выходу слоя, расположенного выше [12].

При обучении нейросети решается задача моделирования целевой функции  $h(x)$ . Если добавить вход  $x$  к выходу сети (т.е. обходящую связь), то сеть будет моделировать  $f(x) = h(x) + x$ . Такой процесс называется остаточным обучением [16].

В обычновенной нейронной сети ее веса после инициализации, как правило, близки к нулю, поэтому она просто выдает значения, также близкие к нулю. Но если добавить обходящую связь, то сеть выдаст копию своих входов, то есть первоначально будет моделировать тождественную функцию. Известно, что целевая функция, близкая к тождественной, значительно ускоряет обучения [12].

Группа сетей ResNet имеет несколько вариаций в зависимости от числа слоев: 34, 50, 101, 152. Чем глубже сеть, тем выше точность, но и процесс обучения длится дольше.

В данной работе была использована ResNet-101, так как она точнее, чем ResNet-50 (см. раздел 11.2) и обладает приемлемым временем работы (около 10 минут на одну эпоху) в отличие от сети со 152 слоями, скорость обучения которой составляет около 20 минут на эпоху.

## 6.4. Кодировщик Inception-v3

В 2014 году победителем ILSVRC (ImageNet Large Scale Visual Recognition Challenge) стала сеть GoogLeNet, ставшая намного более

глубокой и, следовательно, производительной, чем предшествующие сверточные сети. Такие результаты стали возможны благодаря использованию модуля начала (inception module) [12]. Его структура подробно описана в статье [17]. На рисунке 16 приведена схема модуля начала.

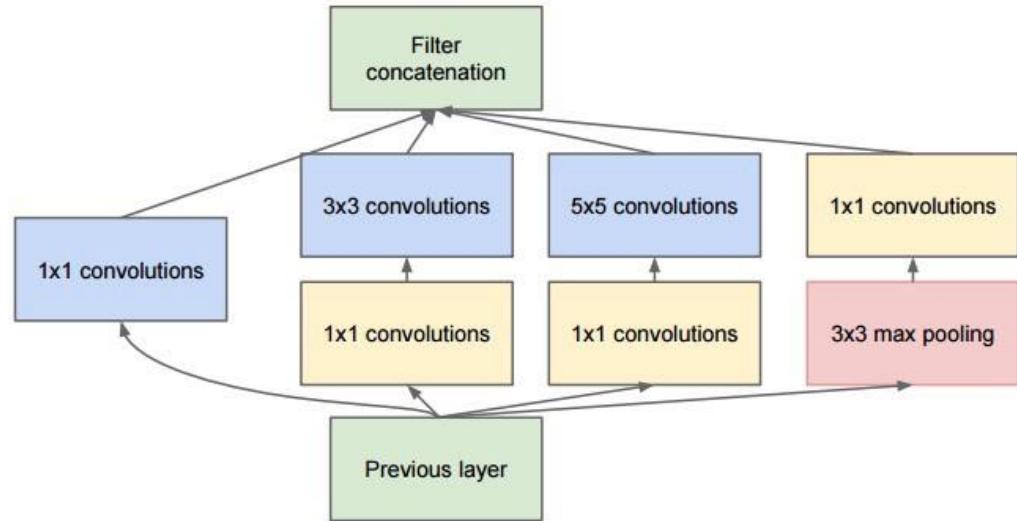


Рисунок 16. Модуль inception

Отличительная особенность модуля состоит в использовании сверточных блоков  $1 \times 1$  для уменьшения количества свойств (и вместе с тем операций) перед подачей в «дорогие» параллельные блоки. В результате обеспечивается более высокая скорость получения результата [18].

В случае многоканального изображения свертка  $1 \times 1$  позволяет объединить все каналы в один. На рисунке 17 к изображению размера  $W \times H \times D$  применяется свертка  $1 \times 1 \times D$ , в результате чего получается изображение размера  $W \times H$ .

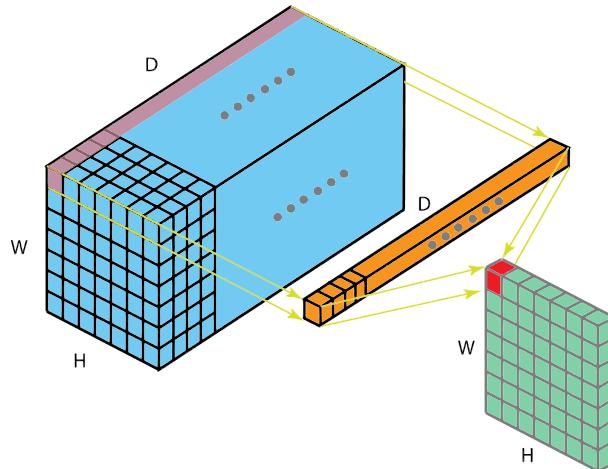


Рисунок 17. Свертка  $1 \times 1$

В 2015 году вышла новая версия модуля и соответствующей архитектуры — Inception-v3. Подробное описание изменений дается в статье разработчиков [19].

Новый inception-модуль представлен на рисунке 18.

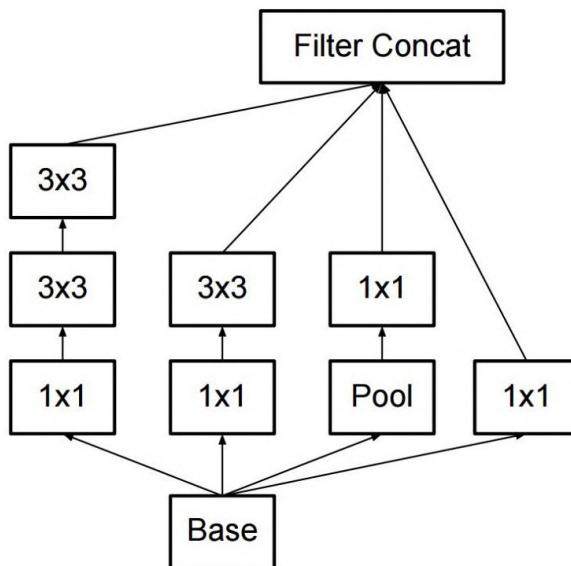


Рисунок 18. Модуль inception

Существует также 4-я версия сети inception, однако в данной работе она не используется, так как отсутствует в библиотеке Segmentation Models.

## 7. Настройка моделей

Преобразуем исходный массив данных, разделив метку изображения на название файла и наименование класса облаков. Из файла `sample.csv` получим список имен файлов со спутниковыми снимками для предсказания. Код, выполняющий описанные действия, приведен на листинге 7.1.

---

### Листинг 7.1 Подготовка данных

---

```
train = pd.read_csv('/content/drive/My Drive/clouds/train.csv')
subm = pd.read_csv('/content/drive/My Drive/clouds/sample.csv')
train['image'] = train['Image_Label'].\
    apply(lambda x: x.split('_')[0])
train['label'] = train['Image_Label'].\
    apply(lambda x: x.split('_')[1])
subm['image'] = subm['Image_Label'].\
    apply(lambda x: x.split('_')[0])
test = pd.DataFrame(subm['image'].unique(), columns=['image'])
```

---

В результате получим следующий массив `train` (см. таблицу 2).

Таблица 2. Массив размеченных данных

Image_Label	EncodedPixels	image	label
0011165.jpg_Fish	264918 937 266318 937...	0011165.jpg	Fish
0011165.jpg_Flower	1355565 1002 1356965 1002...	0011165.jpg	Flower
0011165.jpg_Gravel	Nan	0011165.jpg	Gravel
002be4f.jpg_Fish	233813 878 235213 878...	002be4f.jpg	Fish

После этого разделим данные на тренировочные (80 %) и валидационные (20 %).

В данной работе в качестве модели выступает сеть U-Net, либо FPN с одним из кодировщиков (ResNet, Inception). То есть для каждой сети в качестве сжимающего пути подставляется один из кодировщиков. В разделе 11 сравниваются разные комбинации сети и кодировщика.

Для настройки модели необходимо задать следующие параметры:

1. Размеры изображения.
2. Функция потерь и оптимизатор.
3. Скорость обучения.
4. Callback-функции.
5. Количество эпох.
6. Аугментации.

По требованию архитектур каждая размерность изображения должна делиться на 32, при этом размер  $2100 \times 1400$  слишком большой, поэтому было принято решение уменьшить размер входного изображения до  $480 \times 320$  пикселей с помощью метода `resize` библиотеки cv2. Этот размер близок к тому, который требуется на Kaggle —  $525 \times 350$ . Предсказания, сделанные сетью, будут увеличиваться до размера, требуемого платформой. Это необходимо для

того, чтобы проверять качество моделей на тестовой выборке, доступной только на Kaggle.

В качестве оптимизатора используется Adam, который «в среднем является наилучшим выбором на данный момент» [20]. В качестве функции потерь в данной работе выбрана бинарная кросс-энтропия [21], определяемая по формуле (4)

$$BCE = -\frac{1}{N} \sum_{i=1}^N (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) \quad (4)$$

где  $y_i$  — принадлежность пикселя маске ( $y_i \in \{0, 1\}$ ),  $\hat{y}_i$  — предсказанная вероятность принадлежности пикселя к маске ( $0 \leq \hat{y}_i \leq 1$ ),  $N$  — число пикселей.

При этом функция  $f(x) = x \log x$  доопределена в нуле формулой  $f(0) = 0$ . Бинарная кросс-энтропия используется потому, что она показала хорошие результаты при решении аналогичной задачи сегментации [4].

В качестве скорости обучения обычно берется небольшое число порядка  $10^{-4}$  [22]. Положим скорость равной  $3 \times 10^{-4}$ . В процессе обучения этот параметр будет меняться с помощью callback-функций.

## 7.1. Callback-функции

В работе определены следующие callback-функции, вызываемые после каждой эпохи и направленные на предотвращение переобучения:

1. Снизить в 2 раза скорость обучения, если функция потерь на валидационном множестве не убывает 3 эпохи подряд. Предельное минимальное значение этого параметра —  $1 \times 10^{-6}$ .
2. Остановить обучение, если функция потерь на валидационном множестве не убывает 5 эпох подряд. После остановки происходит восстановление весов из наилучшей модели. Такая callback-функция называется Early stopping и подробно описана в книге [22].

На листинге 7.2 приведен код определения списка callback-функций, который затем будет передан в функцию обучения модели.

---

### Листинг 7.2 Задание callback-функций

---

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
erlstop = EarlyStopping(monitor='val_loss', mode='min', \
                       patience=5, restore_best_weights=True, verbose=1)
redlr = ReduceLROnPlateau(monitor='val_loss', mode='min', \
                          patience=3, factor=0.5, min_lr=1e-6, verbose=1)
callback_list = [erlstop, redlr]
```

---

На рисунке 19 приведен график изменения функции потерь на обучающем и валидационном множествах. Вертикальными линиями

ми отмечены эпохи 11 и 15, когда была снижена скорость обучения. Кроме того, на 17 эпохе произошла досрочная остановка обучения. Действительно, за 5 эпох не произошло уменьшения функции потерь на валидационном множестве: эпоха 12 показывала 0,2693, а эпоха 17 — 0,2713.

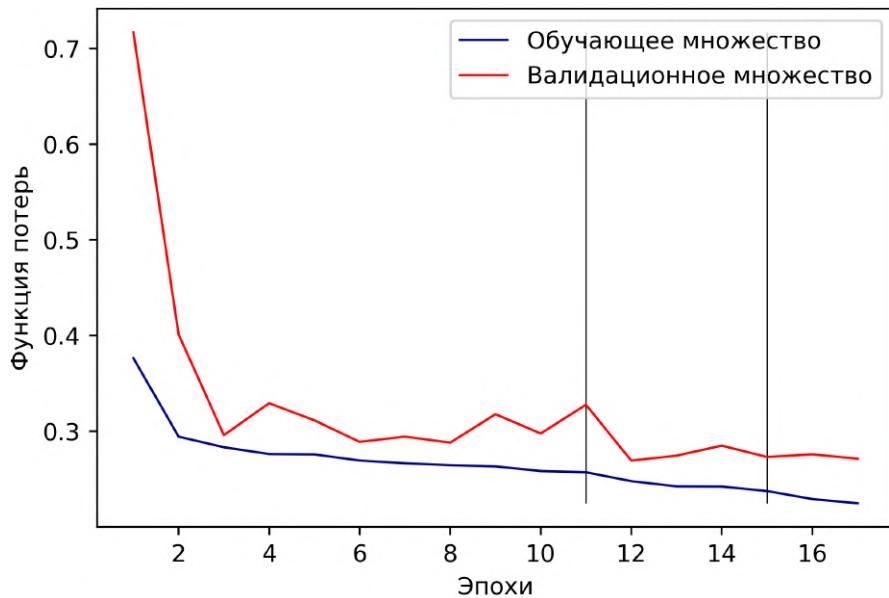


Рисунок 19. График изменения функции потерь

## 7.2. Эпохи

Несмотря на используемые callback-функции, иногда происходит переобучение сети. Например, для кодировщика ResNet-101 (FPN) достаточно 15 эпох. При увеличении их числа остановка не происходит, но качество предсказания, измеряемое с помощью коэффициента Дайса (см. раздел 11), уменьшается (см. рис. 20).

Исходя из этого, необходимо контролировать количество эпох, чтобы избежать переобучения. В работе тренировка модели

происходила на 10, 15, 20 и 25 эпохах. При обучении меньше чем на 10 эпохах результаты ниже, чем на большем количестве эпох, а при тренировке более чем на 25 циклах наблюдается переобучение модели. Промежуточные значения проверяются потому, что обучение разных кодировщиков и моделей идет по-разному.

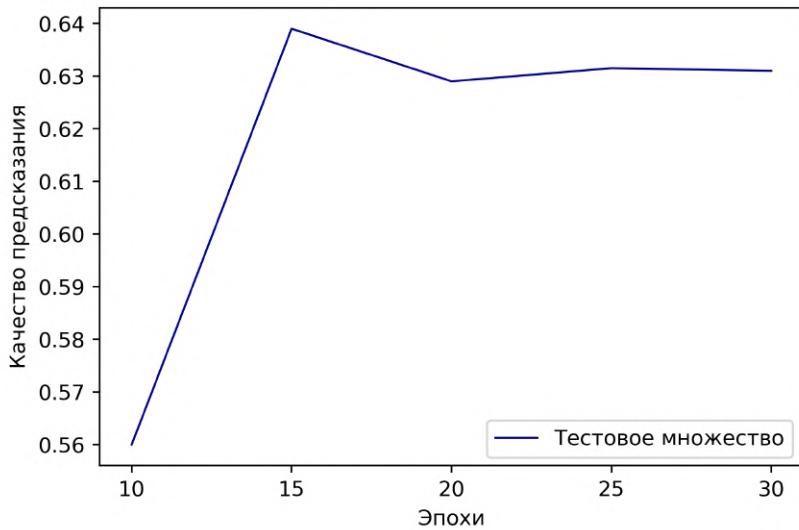


Рисунок 20. Зависимость качества предсказания от числа эпох

### 7.3. Аугментации

Для того чтобы повысить качество обучения сети, применяются различные аугментации — преобразования изображений для увеличения обучающей выборки. В работе аугментации выполняются с помощью библиотеки Albumentations [23].

К каждому изображению применяется несколько преобразований с разной вероятностью. Листинг 7.3 содержит функцию, кото-

ряя применяет композицию преобразований к изображению и его маске.

---

### Листинг 7.3 Композиция аугментаций

---

```
import albumentations as albu
def __random_transform(img, masks):
    composition = albu.Compose([albu.HorizontalFlip(p=0.5),
        albu.GridDistortion(p=0.2), albu.ElasticTransform(p=0.2)])
    composed = composition(image=img, mask=masks)
    img_aug = composed['image']
    masks_aug = composed['mask']
    return img_aug, masks_aug
```

---

Ниже приведено несколько используемых в работе композиций аугментаций, для краткости называемых блоками.

#### Блок № 1. Отражения

Данный блок содержит 2 преобразования:

- Горизонтальное отражение (вероятность 0,7).
- Вертикальное отражение (вероятность 0,7).

Комбинация этих аугментаций является довольно простой и не изменяет структуру изображения. На рисунке 21 приведен пример изображения и маски, выделяющей область Flower, до применения преобразований. Рисунки 22–23 демонстрируют результат применения соответственно горизонтального и вертикального отражений.

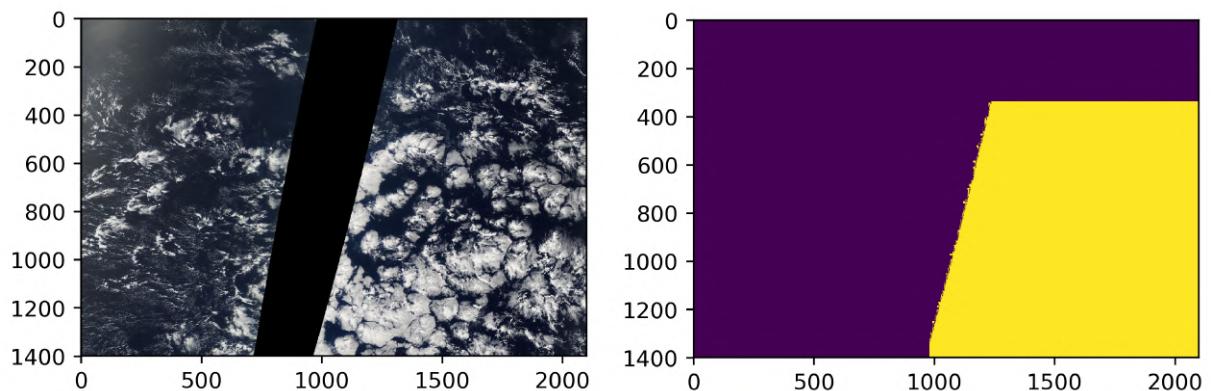


Рисунок 21. Исходное изображение и маска Flower

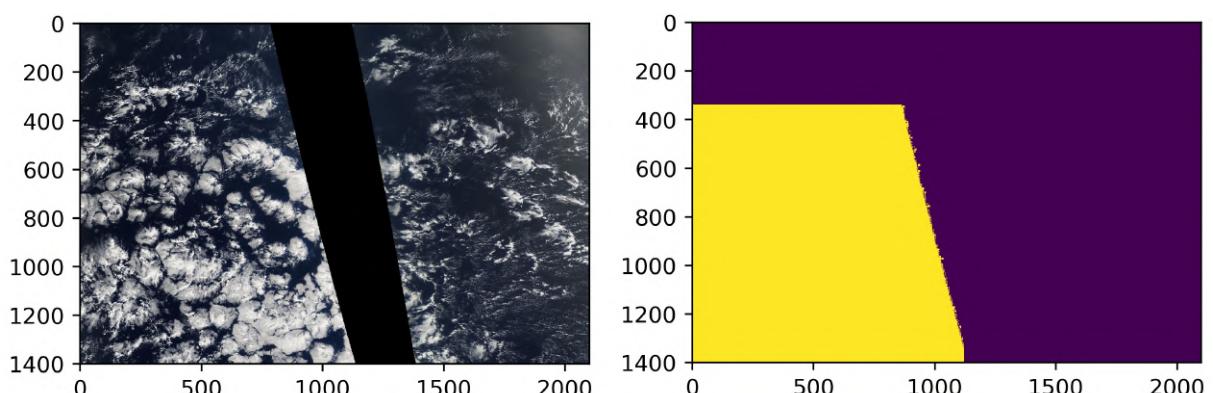


Рисунок 22. Горизонтальное отражение

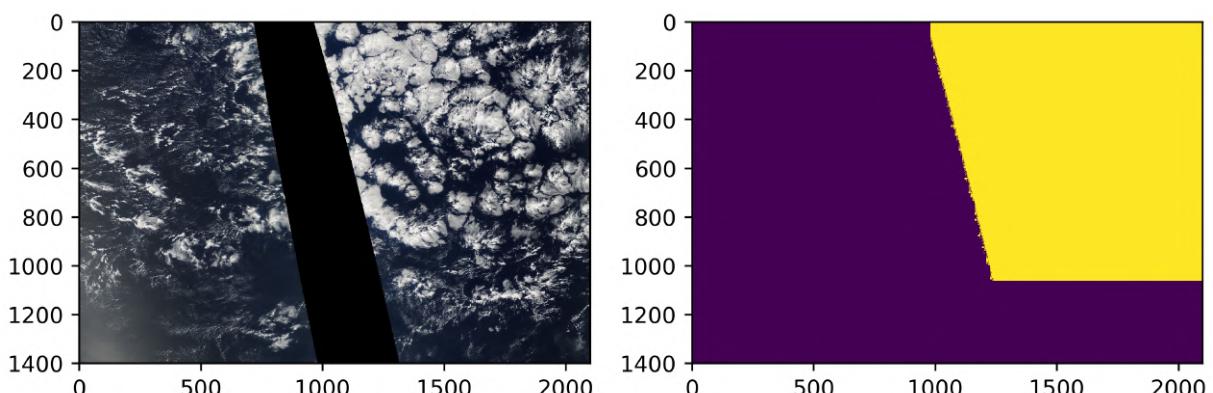


Рисунок 23. Вертикальное отражение

## **Блок № 2. Отражение, смещение и масштабирование, искривления**

В данном блоке используются 4 преобразования:

- Горизонтальное отражение (вероятность 0,5).
- Смещение вместе с масштабированием (вероятность 0,5).
- Искривление по сетке (вероятность 0,5).
- Оптическое искривление (вероятность 0,5).

Такие аугментации позволяют существенно разнообразить обучающее множество, в основном за счет искривлений.

Смещение осуществляется либо по вертикали, либо по горизонтали не более чем на 10 %. Масштабирование выполняется в диапазоне от уменьшения в два раза до увеличения в два раза. На рисунке 24 приведен результат применения смещения и масштабирования.

Искривление по сетке растягивает или отражает некоторые области изображения (см. рис. 25).

Оптическое искривление моделирует накладывание изображения на выпуклую или вогнутую поверхность (см. рис. 26).

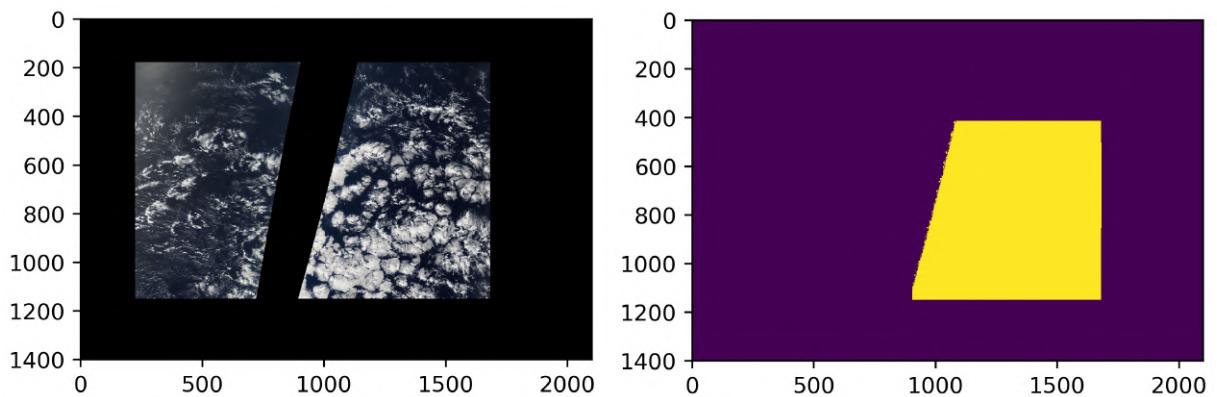


Рисунок 24. Смещение и масштабирование

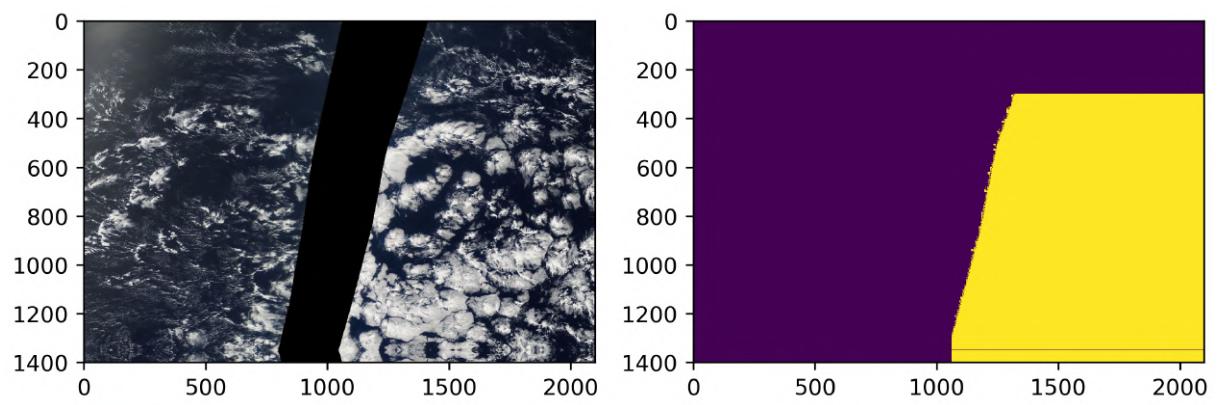


Рисунок 25. Искривление по сетке

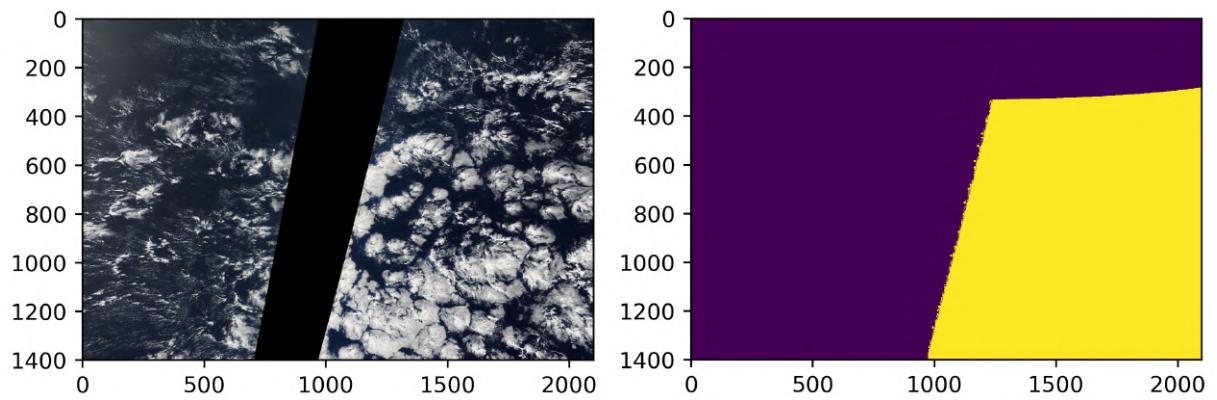


Рисунок 26. Оптическое искривление

### **Блок № 3. Отражения, искривление по сетке, упругая деформация**

Последний блок содержит 4 преобразования:

- Горизонтальное отражение (вероятность 0,5).
- Вертикальное отражение (вероятность 0,5).
- Искривление по сетке (вероятность 0,2).
- Упругая деформация (растяжение, сжатие; вероятность 0,2).

В данном наборе аугментаций намеренно снижена вероятность искривлений и деформаций, чтобы не получить чересчур разнообразное множество. Ниже (см. рис. 28) приведен пример упругой деформации и исходное изображение (см. рис. 27).

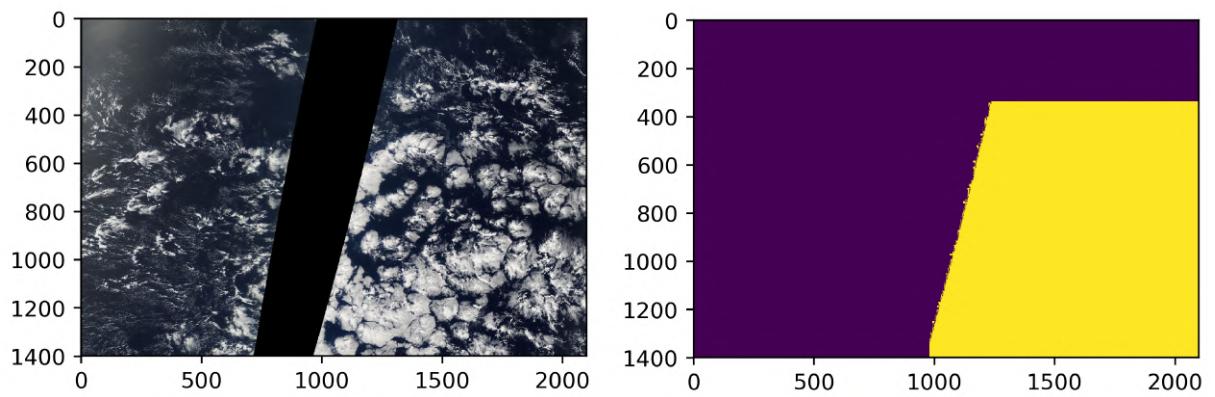


Рисунок 27. Исходное изображение и маска Flower

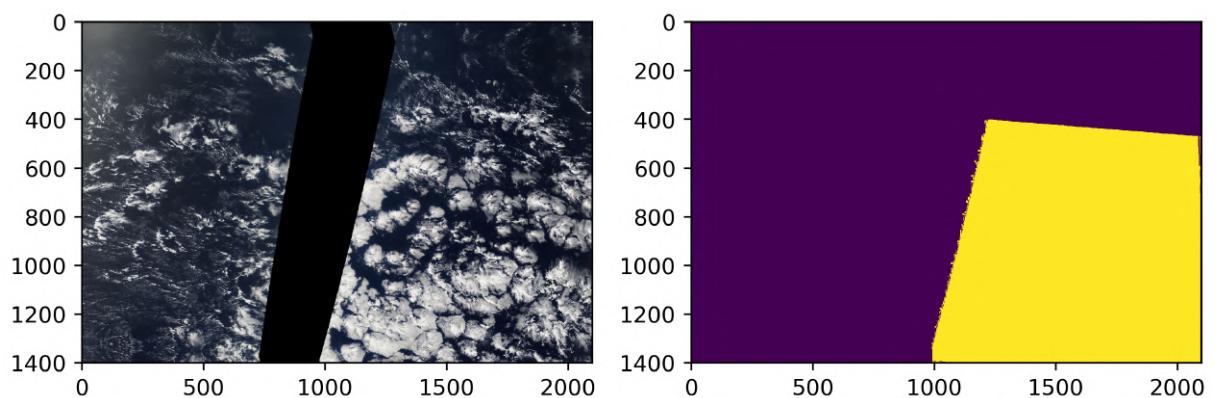


Рисунок 28. Упругая деформация

## 8. Обучение модели и предсказания

После того как определены все параметры, можно загрузить модель с весами, предобученными на датасете ImageNet [24]. Для этого удобно определить функцию (см. листинг 8.1), принимающую в качестве параметра название кодировщика (BACKBONE\_NAME).

---

### Листинг 8.1 Загрузка модели

---

```
import segmentation_models as sm
from keras.losses import binary_crossentropy
from keras import optimizers
def download_model(BACKBONE_NAME):
    model = sm.FPN(backbone_name=BACKBONE_NAME, classes=4,
                    activation='sigmoid', input_shape=(320, 480, 3))
    optimizer = optimizers.Adam(lr=3e-4)
    model.compile(optimizer=optimizer, loss=binary_crossentropy)
    return model
```

---

Для данной задачи потребовалось определить собственный генератор данных, так как имеющийся в библиотеке Keras не работает с масками, заданными в формате RLE [25]. Написанный генератор унаследован от встроенного в Keras.

Так как объем памяти видеокарты ограничен, то изображения предъявляются сети пакетами (batch), размер пакета обычно полагают равным  $2^n$  [26]. Для архитектуры U-Net один пакет состоит из 16 изображений, для FPN — из 8. В FPN для предсказания используются карты всех уровней, поэтому расход памяти больше, чем у U-Net, следовательно, размер пакета должен быть меньше.

Далее создаются два экземпляра класса генератора — для обучающего множества и валидационного, отличающиеся входными данными. В них указывается датасет, размер пакета, разрешение изображения, число каналов и классов. Кроме того, задается зерно для перемешивания обучающего множества. Это делается с целью воспроизводимости результатов.

После этого загруженную модель можно обучить, указав полученные ранее параметры (см. листинг 8.2).

---

### Листинг 8.2 Обучение модели

---

```
def fit_model(EPOCH_COUNT, model):
    history = model.fit_generator(generator=train_generator,
                                    validation_data=valid_generator, epochs=EPOCH_COUNT,
                                    callbacks=callback_list, verbose=2).history
    return history
```

---

Параметр `verbose`, равный 2, означает, что после каждой эпохи будет выводиться информация о значении функции потерь и о времени работы. Для того чтобы дополнительно видеть прогресс-бар процесса обучения, следует установить значение 1.

Далее обученной сети по пакетам подаются изображения из тестового множества, для каждого из которых она выдает маску по каждому из типов облаков, указывающую вероятность  $p$  ( $0 \leq p \leq 1$ ) отнесения пикселя к определенному классу. Затем проводится бинаризация с порогом  $t$  и пиксели со значением 1 включаются в результирующую маску. Таким образом формируются выделенные области для каждого из четырех типов облаков.

Для определения порога  $t$  проведено сравнение качества работы нескольких моделей при различных  $t$  из отрезка  $[0,3; 0,7]$  с шагом 0,1. Из графика зависимости метрики от порога (см. рис. 29) следует, что наилучшие результаты достигаются при  $t = 0,4$  и  $t = 0,5$ . Исходя из этого, для всех моделей порог подбирается экспериментально путем проб разных  $t$  на отрезке  $[0,35; 0,55]$  с шагом 0,05.

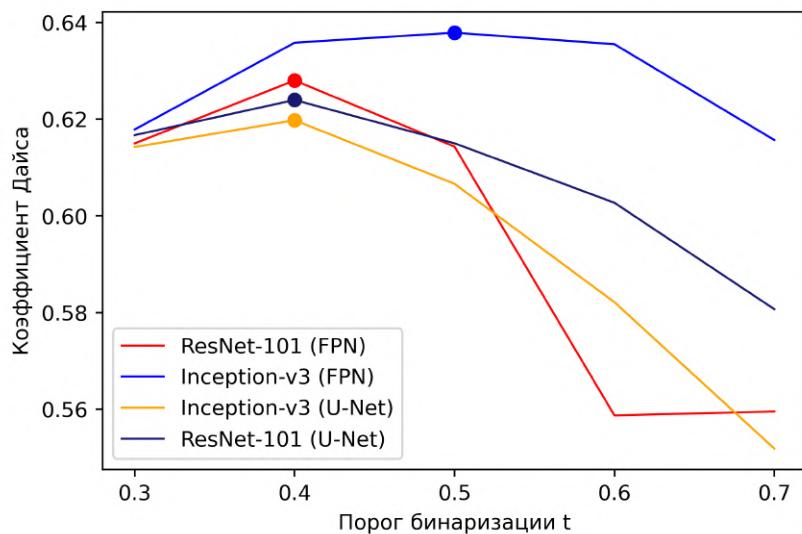


Рисунок 29. Зависимость метрики от порога бинаризации

## 9. Описание полученных результатов

Во всех предсказаниях на одном снимке чаще всего выделяются 2 типа облаков (48 %) или 1 тип (42 %), намного реже выделяется 3 типа (9 %). На снимки со всеми 4 классами приходится всего 1 %.

Ниже (см. рис. 30) приведен пример сегментации снимка с помощью U-Net, на котором выделены классы Gravel и Sugar.

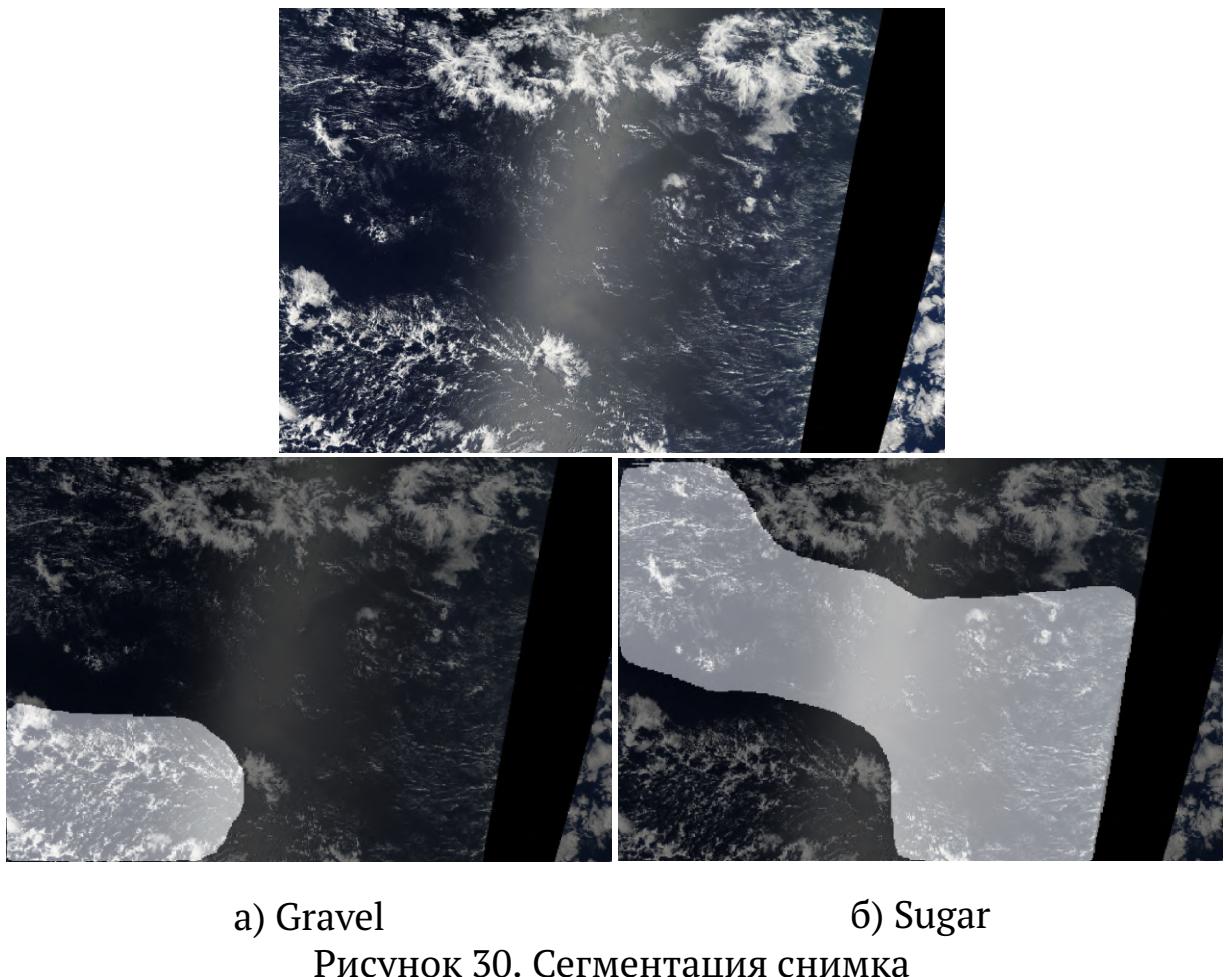
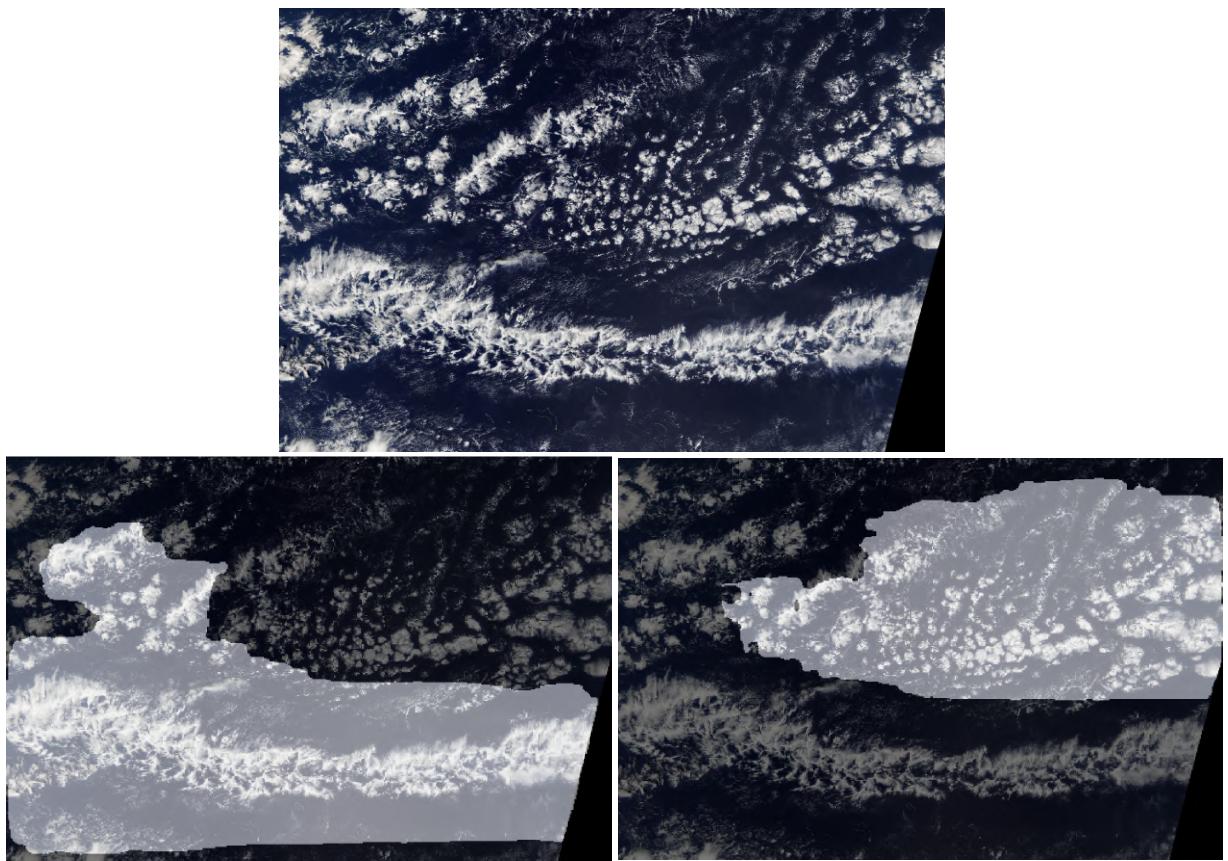


Рисунок 30. Сегментация снимка

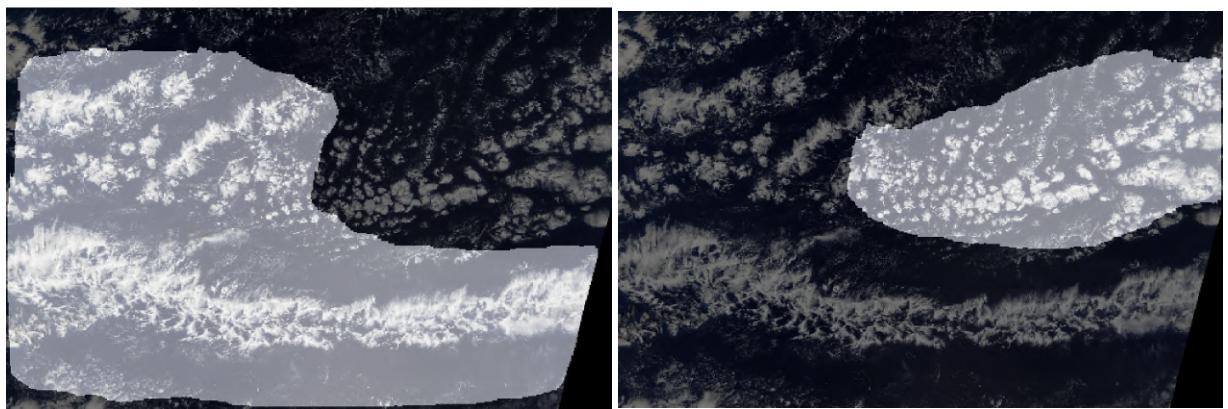
На рисунках 31-32 представлены результаты работы сетей U-Net и FPN. На снимке выделены области Fish и Flower.



a) Fish

б) Flower

Рисунок 31. Результат сегментации (FPN)

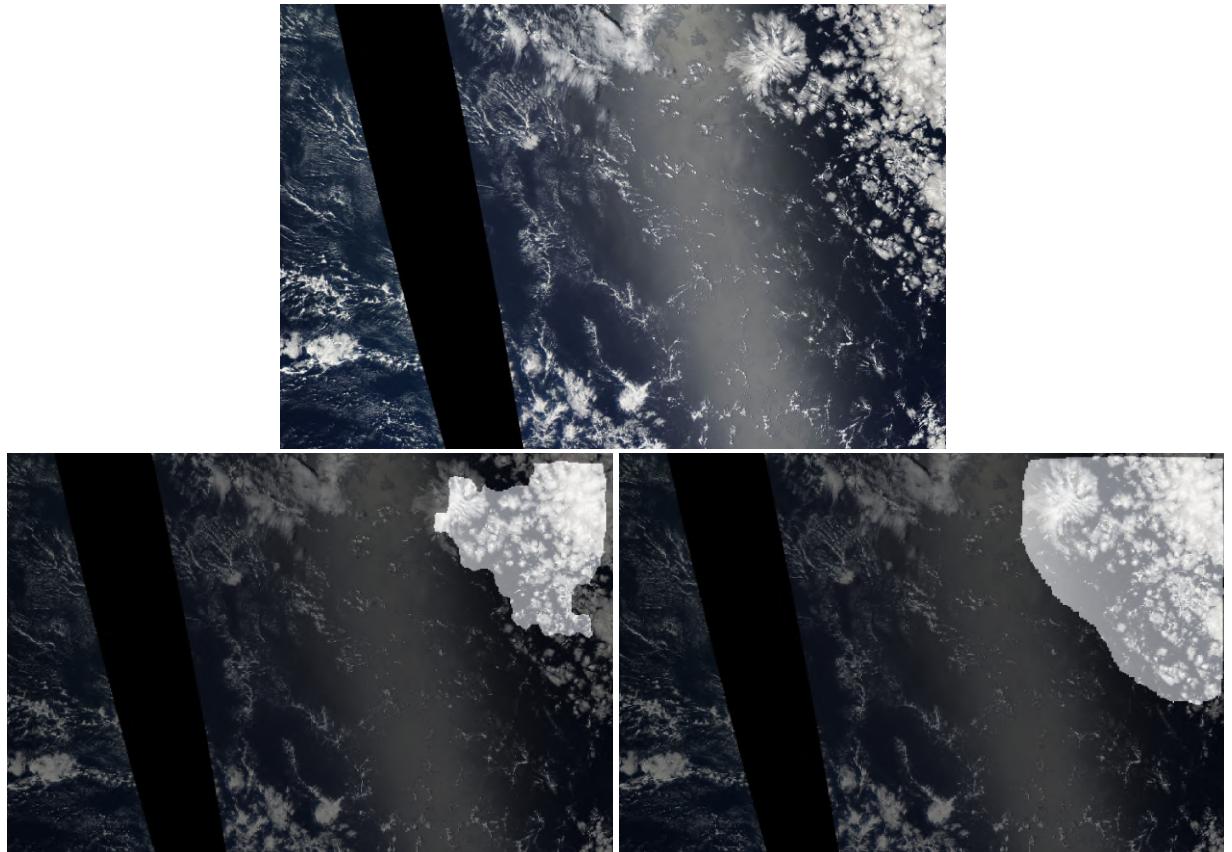


a) Fish

б) Flower

Рисунок 32. Результат сегментации (U-Net)

Маска, предсказанная с помощью FPN, имеет менее гладкие края, чем результат сегментации посредством U-Net. Для сравнения представлены оба варианта предсказания (см. рис. 33).



a) FPN

б) U-Net

Рисунок 33. Сравнение FPN и U-Net

## 10. Постобработка

Результат сегментации для одного типа облаков может содержать несколько областей. В работе сделано предположение, что небольшие области являются ошибочными, так как не могут показать всю структуру облаков.

Для проверки этой гипотезы реализована функция, удаляющая все компоненты маски, размер которых меньше 11 000 пикселей. Такой порог выбран исходя из гистограммы, показывающей распределение площадей областей в одном из предсказаний сети FPN (см. рис. 34).

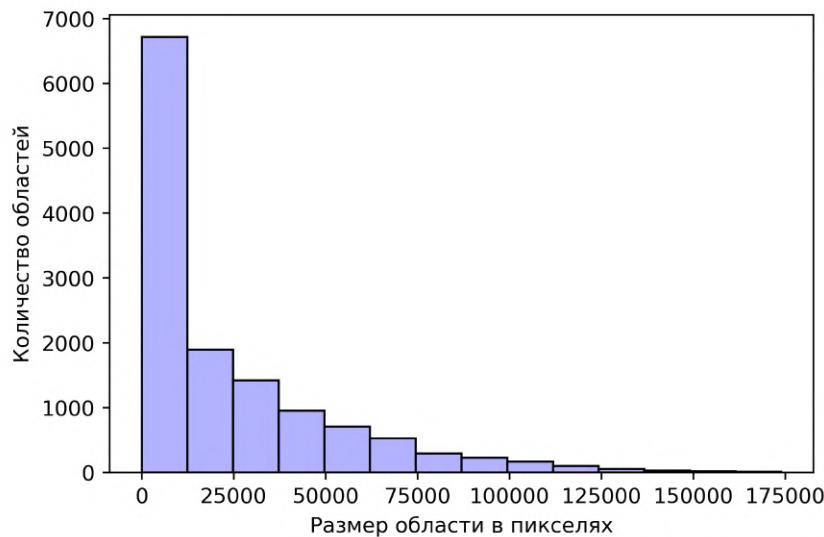


Рисунок 34. Гистограмма площадей областей

В реализованной функции связные области выделяются с помощью метода `connectedComponents`, встроенного в библиотеку cv2. Затем организуется цикл по числу связных компонент, каждая

из которых включается в результирующую маску только если число пикселей в ней больше `min_size = 11 000`.

Код, реализующий постобработку, представлен на листинге 10.1.

---

### Листинг 10.1 Постобработка

---

```
import cv2
def post_process(mask, min_size=11000):
    cnt_component, component = cv2.connectedComponents(
                                mask.astype(np.uint8))
    predictions = np.zeros(mask.shape, np.float32)
    for c in range(1, cnt_component):
        p = (component == c)
        if p.sum() > min_size:
            predictions[p] = 1
    return predictions
```

---

На рисунке 35 приведен пример работы алгоритма.

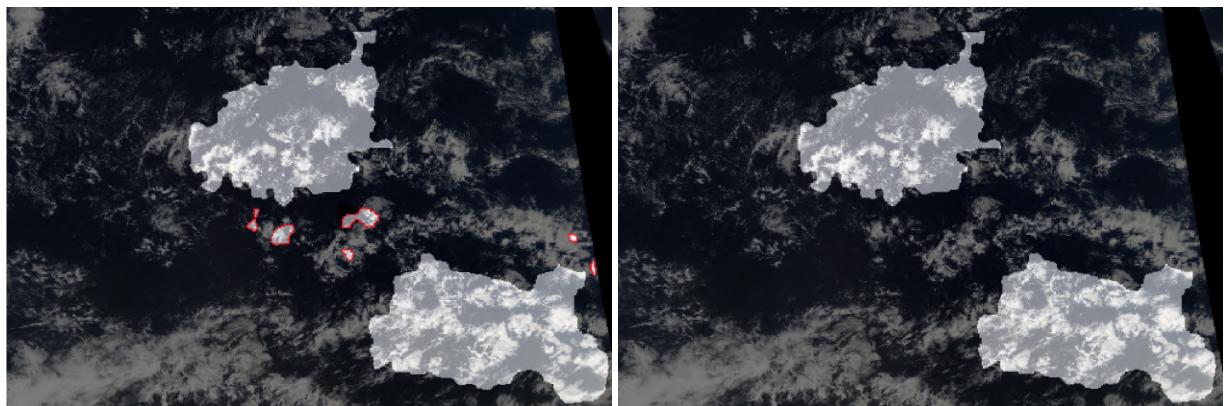
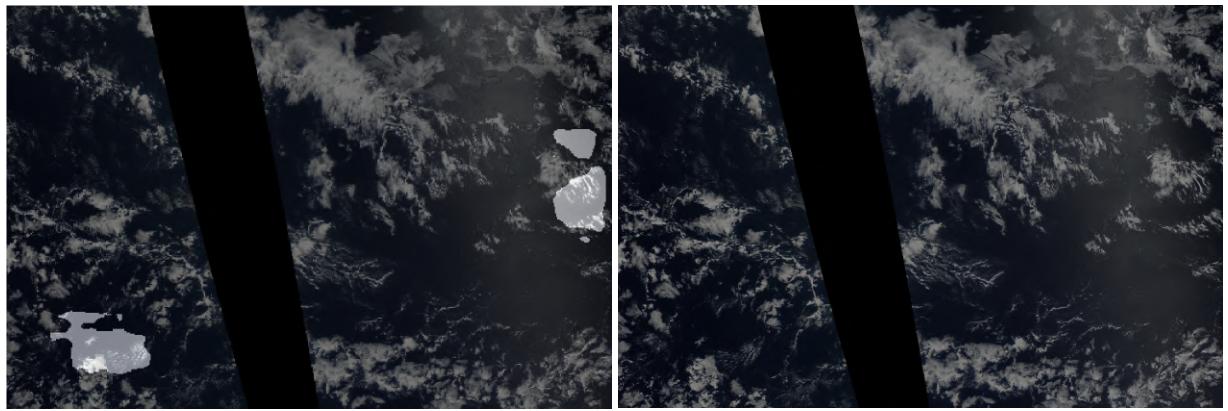


Рисунок 35. Результат удаления небольших областей маски Flower

В случае если маска состоит из небольших областей, постобработка может полностью удалить ее (см. рис. 36).



а) Предсказание

б) Постобработка

Рисунок 36. Полное удаление маски Flower

Влияние постобработки на качество предсказания описано в подразделе 11.1. В результате применения данной обработки средняя величина метрики увеличилась с 0,5919 до 0,6235.

## 11. Сравнение моделей

Для того чтобы сравнить модели, для них были вычислены значения метрики на тестовом множестве. Так как данное множество в задаче скрыто, то полученные решения отправлялись на Kaggle, где и вычислялась метрика. Для автоматизации процесса получен персональный токен и настроена среда для работы с API (см. листинг 11.1).

---

### Листинг 11.1 Настройка Kaggle API

---

```
!mkdir ~/.kaggle
!touch ~/.kaggle/kaggle.json
api_token = {"username": "student", "key": "abcdef0123456789"}
import json
with open('/root/.kaggle/kaggle.json', 'w') as file:
    json.dump(api_token, file)
!chmod 600 ~/.kaggle/kaggle.json
```

---

Метрика, вычисляемая на Kaggle, называется коэффициентом Соренсена-Дайса [27] и определяется по формуле (5)

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \quad (5)$$

где  $X$  — предсказанная область,  $Y$  — область из эталонной разметки.

Для случая  $X = Y = \emptyset$  полагается, что  $DSC = 1$ . Из формулы следует, что  $0 \leq DSC \leq 1$ .

Эталонная разметка содержит прямоугольные области, а предсказания сети имеют неровные границы (т.е. предсказание никогда не совпадет точно с эталоном). Кроме того, некоторые типы облаков трудноразличимы даже для экспертов, размечавших снимки. Все это сказывается на том, что  $DSC$  для лучших решений не превышает 0,7.

## 11.1. Влияние постобработки

Таблица 3 показывает влияние удаления небольших областей в предсказании на значение метрики. Результаты приведены для сети U-Net на 15 эпохах обучения на трех разных блоках аугментаций.

Таблица 3. Сравнение работы с постобработкой и без нее

Аугментации	ResNet-101		Inception-v3	
	Без постобр.	С постобр.	Без постобр.	С постобр.
Блок № 1	0,5764	0,6177	0,5714	0,6124
Блок № 2	0,5974	0,6222	0,5882	0,6217
Блок № 3	0,6047	0,6267	0,5972	0,6403
Среднее	0,5983	0,6222	0,5856	0,6248

Из этих данных видно, что постобработка добавляет несколько процентов к точности. Улучшение результатов объясняется тем, что метрика чувствительна к False positive: если эталонная сегмен-

тация не содержит маску, а предсказание включает в себя хотя бы один пиксель, то получается ошибка.

Далее все результаты будут приводиться с выполненной постобработкой.

## 11.2. Сравнение кодировщиков

В таблице 4 приведено сравнение трех кодировщиков на примере архитектуры U-Net (15 эпох).

Таблица 4. Сравнение кодировщиков

Аугментации	ResNet-50	ResNet-101	Inception-v3
Блок № 1	0,6217	0,6177	0,6296
Блок № 2	0,6011	0,6222	0,6217
Блок № 3	0,6248	0,6267	<b>0,6403</b>
Среднее	0,6159	0,6222	0,6305

Как видно, результаты кодировщика Inception-v3 в среднем лучше других, и при этом наилучший результат фиксируется именно на Inception-v3.

На рисунке 37 дается сравнение кодировщиков, приведенное в статье [28]. Диаграмма подтверждает, что Inception-v3 обладает большей точностью, чем сети группы ResNet.

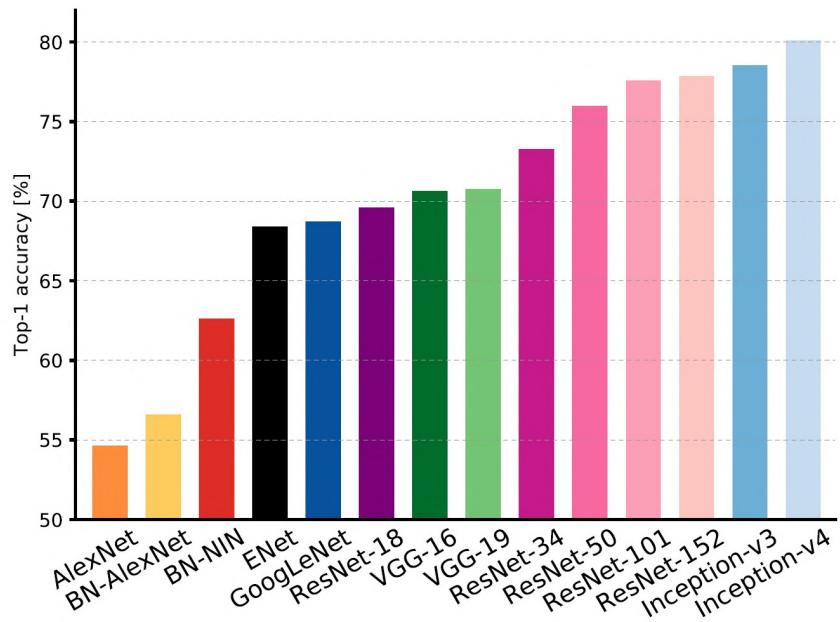


Рисунок 37. Сравнение кодировщиков

### 11.3. Сравнение архитектур сети

Результаты работы сетей U-Net и FPN отличаются незначительно (см. таблицу 5). Следовательно, между ними нет существенной разницы для решения данной задачи.

Таблица 5. Сравнение архитектур

Аугментации	ResNet-101		Inception-v3	
	U-Net	FPN	U-Net	FPN
Блок № 1	0,6177	0,6071	0,6296	0,6249
Блок № 2	0,6222	0,6365	0,6217	0,6293
Блок № 3	0,6267	0,6365	0,6403	0,6371
Среднее	0,6222	0,6267	0,6305	0,6304

## 11.4. Влияние предобработки

Сравним (см. таблицу 6) работу моделей на цветных снимках и на изображениях, прошедших предобработку. Обучение модели длилось 20 эпох, использовалась сеть U-Net.

Таблица 6. Влияние предобработки

Аугментации	ResNet-101		Inception-v3	
	Цветное	Предобработ.	Цветное	Предобработ.
Блок № 1	0,6189	0,6117 ↓	0,6293	0,6232 ↓
Блок № 2	0,6272	<b>0,6413 ↑</b>	0,6284	0,6312 ↑
Блок № 3	0,6318	0,6273 ↓	0,6297	0,6275 ↓

Как видно, для блоков № 1 и 3 предобработка только уменьшила значение метрики. Однако для блока № 2 произошло увеличение коэффициента Дайса, а лучший результат (выделен полужирным) получен именно на обработанных снимках.

Более того, предобработка уменьшает время обучения модели. Одна эпоха на обработанных изображениях длится на 20 % меньше, чем на исходных (см. рис. 38). Результаты приведены для Inception-v3.

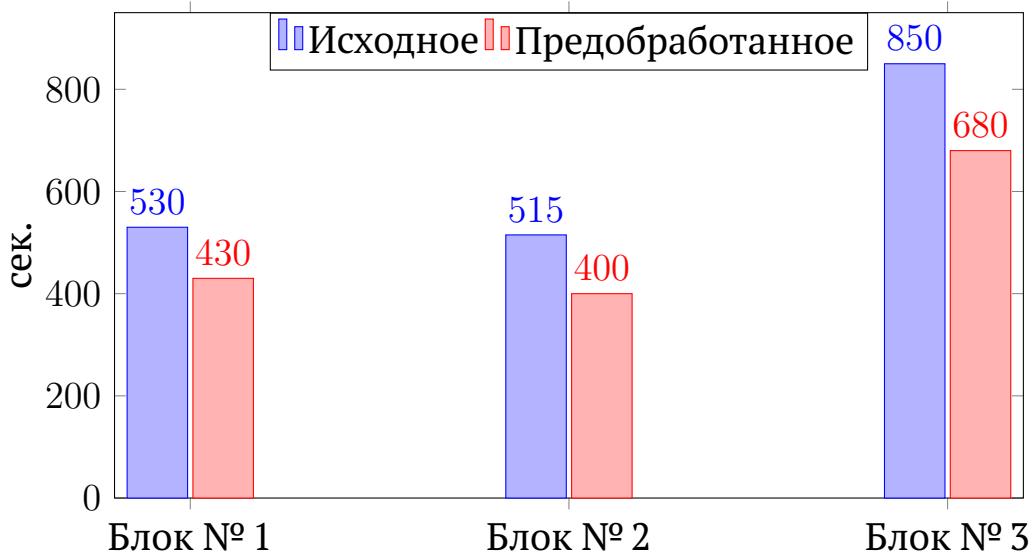


Рисунок 38. Время работы одной эпохи

## 11.5. Сравнение аугментаций

Для начала отметим, что без применения аугментаций функция потерь на валидационном множестве начинает не убывать, а расти (см. рис. 39), из-за чего уже после седьмой эпохи происходит остановка обучения. Очевидно, что без аугментаций нет смысла тренировать модель.

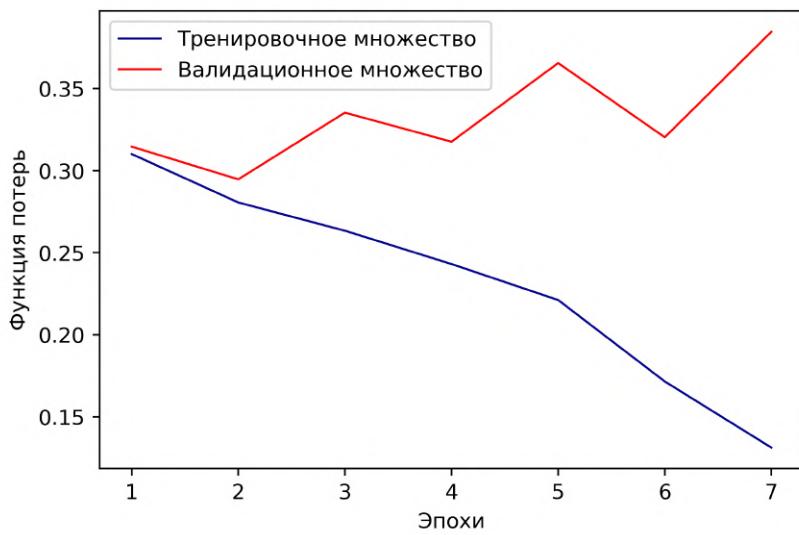


Рисунок 39. График изменения функции потерь

В таблице 7 приведено сравнение трех блоков аугментаций для разных сетей на 20 эпохах обучения ( $U\text{-Net}^{Pr}$  означает  $U\text{-Net}$ , на вход которой подается предобработанное изображение). В каждой строке полужирным шрифтом выделен наилучший результат для данного блока. Как видно, в среднем, блок № 3 показывает результаты лучше других.

Таблица 7. Сравнение аугментаций

Аугмент.	ResNet-101			Inception-v3			Среднее
	U-net	$U\text{-Net}^{Pr}$	FPN	U-Net	$U\text{-Net}^{Pr}$	FPN	
Блок № 1	0,6189	0,6117	0,6241	0,6293	0,6232	<b>0,6346</b>	0,6236
Блок № 2	0,6272	<b>0,6413</b>	0,6287	0,6284	0,6312	0,6201	0,6294
Блок № 3	0,6318	0,6273	0,6287	0,6297	0,6275	<b>0,6347</b>	0,6300

Однако наилучший результат среди приведенных зафиксирован на блоке № 2, что говорит о необходимости сравнения всех блоков, даже если какой-то из них в среднем хуже других.

Кроме того, модель обучается дольше на блоке № 3: обучение одной эпохи занимает на 310–320 секунд дольше (см. рис. 40), что на 20 итерациях даст разницу уже в 1,5–2 часа. Это может быть существенно при вычислении на облачных ресурсах, где время работы на GPU ограничено.

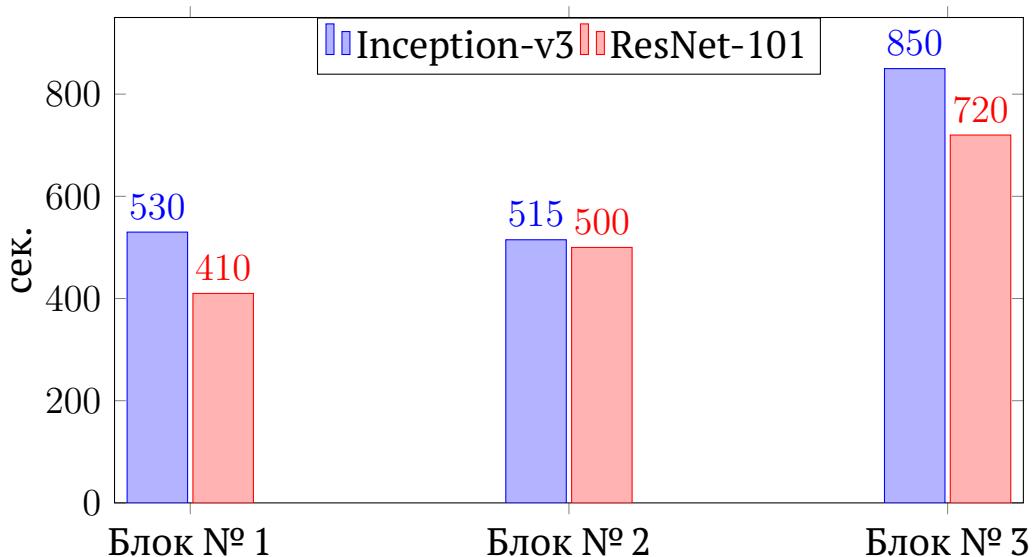


Рисунок 40. Время работы одной эпохи

Значительной разницы во времени работы между Inception-v3 и ResNet-101 нет.

## 12. Объединение предсказаний

Маски, полученные с помощью разных моделей, можно объединить. Пример для класса Flower представлен на рисунке 41: объединяются предсказания, сделанные с помощью кодировщика Inception-v3.

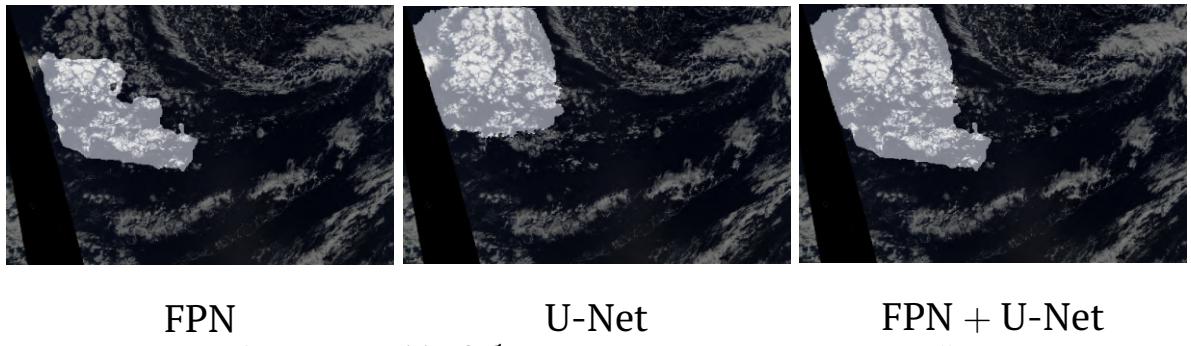


Рисунок 41. Объединение предсказаний

Объединение может быть полезно и в тех случаях, когда одна из двух моделей вообще не выдала маску. На примере рассмотренных решений таких ситуаций немало — 22 % от общего числа предсказаний.

Для того чтобы оценить влияние объединения на метрику, выберем из каждого блока аугментаций и кодировщика по одному лучшему результату — таких решений всего шесть. Затем объединим каждое предсказание с другими. Всего будет 15 различных вариантов.

В таблице 8 приведены результаты анализа. Для блока № 2 использовались изображения, предварительно обработанные по алгоритму устранения бликов (отмечено в таблице индексом  $Pr$ ).

Таблица 8. Объединение предсказаний

ResNet-101			Inception-v3				
	Блок № 1 FPN	Блок № 2 U-Net <sup>Pr</sup>	Блок № 3 FPN		Блок № 1 FPN	Блок № 2 U-net <sup>Pr</sup>	Блок № 3 U-net
Nº	1	2	3		4	5	6
1	0,6241	0,6398	0,6352		0,6379	0,6362	0,6423
2		0,6413	0,6451		0,6421	0,6436	<b>0,6484</b>
3			0,6365		0,6405	0,6429	0,6457
4					0,6345	0,6416	0,6443
5						0,6312	0,6465
6							0,6442

Из полученных данных следует, что первое решение вносит минимальный вклад в объединение, а шестое — максимальный. Наилучший результат получен из решений 2 и 6 с максимальными значениями метрик.

## 13. Восстановление маски до прямоугольника

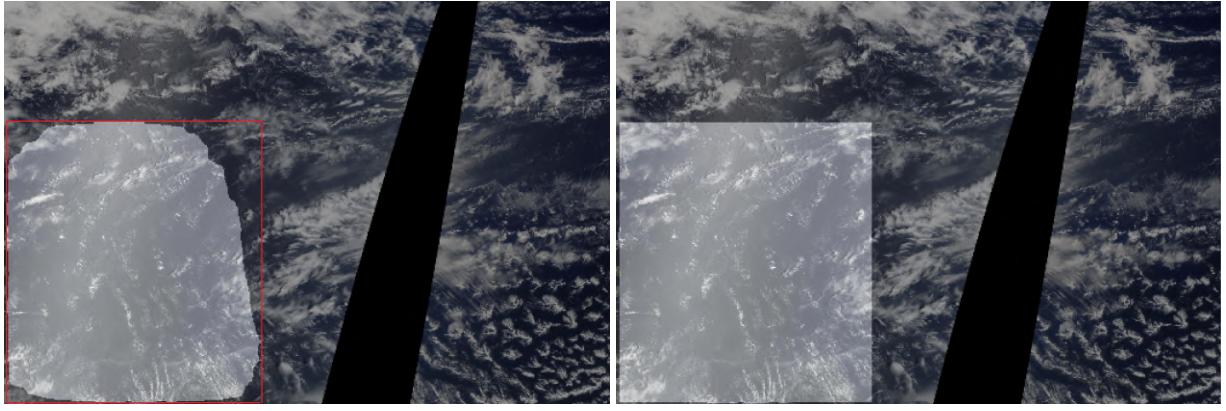
Эталонная разметка выполнена экспертами, и области, выделенные в ней, являются прямоугольниками, в отличие от предсказаний моделей. Вероятно, произвольная форма масок снижает значение метрики.

Восстановим каждую предсказанную область до минимального покрывающего прямоугольника. Для этого каждое предсказание разделяется на связные области с помощью метода `connectedComponents` из библиотеки `cv2`. В каждой области находится 4 значения:

- минимальный и максимальный номер строки, содержащей пиксель маски;
- минимальный и максимальный номер столбца, содержащего пиксель маски.

Затем все пиксели, лежащие между найденными строками и столбцами, полагаются принадлежащим результату предсказания.

На рисунке 42 приведен пример маски Sugar и ее восстановления до прямоугольной формы.



а) Предсказанная маска                                        б) Восстановленная маска  
Рисунок 42. Результат дополнения маски до прямоугольника

Для отобранных в прошлом разделе решений выполним дополнение масок. В результате для всех моделей произошло небольшое увеличение коэффициента Дайса (см. таблицу 9).

Таблица 9. Объединение предсказаний

Блок	ResNet-101			Inception-v3		
	№ 1	№ 2	№ 3	№ 1	№ 2	№ 3
Кодировщик	FPN	U-Net <sup>Pr</sup>	FPN	FPN	U-net <sup>Pr</sup>	U-net
До	0,6241	0,6413	0,6365	0,6345	0,6312	0,6442
После	0,6273	0,6432	0,6394	0,6355	0,6319	0,6448

Однако если снова объединить полученные предсказания, то результаты будут хуже. Средняя величина коэффициента Дайса до дополнения составляла 0,6421, а после — 0,6330. Кроме того, лучший результат с восстановленными масками составил 0,6404 против 0,6484 без такой обработки.

## **14. Результаты проведенного исследования**

В результате проведенных исследований были сделаны следующие выводы:

1. Для данной задачи хорошо подходит кодировщик Inception-v3 и третий блок аугментаций (см. стр. 38).
2. В предсказанных масках следует обязательно удалять небольшие области.
3. Существенных различий между использованием сетей U-Net и FPN для решения данной задачи нет.
4. Объединение решений увеличивает метрику.
5. Предобработка может дать прирост точности.
6. Дополнение маски до прямоугольника увеличивает качество работы одной конкретной модели, однако ухудшает результат объединения предсказаний.

Наилучший результат достигнут путем объединения двух постобработанных предсказаний, полученных с помощью следующих моделей:

- Сеть U-Net с кодировщиком ResNet-101, обученная на преоб-работанных изображениях и со вторым блоком аугментаций. Порог бинаризации равен 0,4.

- Сеть U-Net с кодировщиком Inception-v3, обученная на исходных снимках и с третьим блоком аугментаций. Порог бинаризации равен 0,45.

## **Заключение**

Таким образом, в работе обучено несколько моделей и проведено сравнение полученных предсказаний. Выполнен анализ влияния предобработки снимков на обучение и качество модели.

Кроме того, показана эффективность рассмотренных вариантов обработки предсказанных масок. В завершении работы произведено объединение лучших моделей, что позволило улучшить итоговый результат.

С кодом, созданным в процессе решения данной задачи, можно ознакомиться по ссылке: <https://git.io/Js5Pj>

## **Список литературы**

1. Kaggle: Understanding Cloud Organization. — URL: [https://www.kaggle.com/c/understanding\\_cloud\\_organization](https://www.kaggle.com/c/understanding-cloud-organization) (дата обр. 01.05.2021).
2. Cloud and cloud shadow detection for Landsat images: The fundamental basis for analyzing Landsat time series / Z. Zhu [и др.] // Remote Sensing Time Series Image Processing; CRC Press: Boca Raton, FL, USA. — 2018. — C. 3—23.
3. Ozkan S., Efendioglu M., Demirpolat C. Cloud detection from RGB color remote sensing images with deep pyramid networks // IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium. — IEEE. 2018. — C. 6939—6942.
4. Tymchenko B., Marchenko P., Spodarets D. Segmentation of cloud organization patterns from satellite images using deep neural networks // Herald of Advanced Information Technology. — 2020. — T. 3, № 1. — C. 352—361.
5. Ahmed T., Sabab N. H. N. Classification and understanding of cloud structures via satellite images with EfficientUNet // arXiv preprint arXiv:2009.12931. — 2020.
6. Google Colaboratory. — URL: <https://colab.research.google.com> (дата обр. 01.05.2020).

7. Репозиторий Segmentation Models на GitHub. — URL: [https://github.com/qubvel/segmentation\\_models](https://github.com/qubvel/segmentation_models) (дата обр. 01.05.2021).
8. Алгоритм устранения бликов на спутниковых снимках. — URL: <https://git.io/J3qHE> (дата обр. 01.05.2021).
9. Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation // International Conference on Medical image computing and computer-assisted intervention. — Springer. 2015. — С. 234—241.
10. Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning. — 2016.
11. Соснин А. С., Суслова И. А. Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLU, tanh // Издательство РГППУ. — 2019. — С. 237—246.
12. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем // М.:«Диалектика». — 2018.
13. Mask R-CNN: архитектура современной нейронной сети для сегментации объектов на изображениях. — URL: <https://habr.com/ru/post/421299/> (дата обр. 01.05.2021).

14. Feature pyramid networks for object detection / T.-Y. Lin [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2017. — С. 2117–2125.
15. Review: ResNet — Winner of ILSVRC 2015 (Image Classification, Localization, Detection). — URL: <https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8> (дата обр. 01.05.2021).
16. Deep residual learning for image recognition / K. He [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — С. 770–778.
17. Going deeper with convolutions / C. Szegedy [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — С. 1—9.
18. Архитектуры нейросетей. — URL: <https://habr.com/ru/company/nix/blog/430524/> (дата обр. 01.05.2021).
19. Rethinking the inception architecture for computer vision / C. Szegedy [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — С. 2818–2826.
20. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — Издательский дом «Питер», 2017.
21. Understanding binary cross-entropy / log loss: a visual explanation. — URL: <https://towardsdatascience.com/>.

- com / understanding - binary - cross - entropy - log - loss - a - visual - explanation - a3ac6025181a (дата обр. 01.05.2021).
22. *Goodfellow I., Bengio Y., Courville A.* Deep Learning. — USA : MIT Press, 2016.
  23. Сайт библиотеки Albumentations. — URL: [https : / / albumentations.ai/](https://albumentations.ai/) (дата обр. 01.05.2021).
  24. Сайт датасета ImageNet. — URL: [http : / / image - net . org/](http://image-net.org/) (дата обр. 01.05.2021).
  25. Keras data generators and how to use them. — URL: [https : / / towardsdatascience . com / keras - data - generators - and - how - to - use - them - b69129ed779c](https://towardsdatascience.com/keras-data-generators-and-how-to-use-them-b69129ed779c) (дата обр. 01.05.2021).
  26. *Bengio Y.* Practical recommendations for gradient-based training of deep architectures // Neural networks: Tricks of the trade. — Springer, 2012. — C. 437—478.
  27. Metrics to Evaluate your Semantic Segmentation Model. — URL: [https : / / towardsdatascience . com / metrics - to - evaluate - your - semantic - segmentation - model - 6bcb99639aa2](https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2) (дата обр. 01.05.2021).
  28. *Canziani A., Paszke A., Culurciello E.* An analysis of deep neural network models for practical applications // arXiv preprint arXiv:1605.07678. — 2016.