МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**  
**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

Направление подготовки: Прикладная математика и информатика

**ОТЧЕТ**

по проектной работе

Тема:

**«Сегментация пожаров на спутниковых снимках с использованием моделей глубокого обучения»**

Выполнили:

студенты группы 3821Б1ПМ1

Иванченко Алексей Михайлович,

Коновалов Игорь Алексеевич.

Нижний Новгород  
2023

# Содержание

[Содержание 2](#_Toc1)

[Введение 3](#_Toc2)

[1. Постановка задачи 4](#_Toc3)

[2. Подход к сегментации с использованием глубокого обучения 5](#_Toc4)

[2.1. Подготовка набора данных 5](#_Toc5)

[2.1.1. Подготовка спутниковых снимков 5](#_Toc6)

[2.1.2. Подготовка масок 5](#_Toc7)

[2.2. Разработка модели 6](#_Toc8)

[2.2.1. Использованные библиотеки 6](#_Toc9)

[2.2.2. Архитектура модели 6](#_Toc10)

[2.2.3. Метрики 7](#_Toc11)

[2.2.4. Функция потерь 7](#_Toc12)

[2.3. Результаты 8](#_Toc13)

[2.3.1. Результаты обучения 8](#_Toc14)

[2.3.2. Сравнение с существующими решениями 8](#_Toc15)

[3. Пользовательский интерфейс 11](#_Toc16)

[3.1. Программная реализация 11](#_Toc17)

[3.1.1. Загрузка спутниковых снимков 11](#_Toc18)

[3.1.2. Графический интерфейс 11](#_Toc19)

[3.2. Руководство пользователя 12](#_Toc20)

[Заключение 13](#_Toc21)

[Список литературы 14](#_Toc22)

[Приложение 16](#_Toc23)

[Приложение 1 16](#_Toc24)

[Приложение 2 16](#_Toc25)

[Приложение 3 19](#_Toc26)

[Приложение 4 25](#_Toc27)

# Введение

На сегодняшний день лесные пожары – одна из самых актуальных экологических проблем. Неконтролируемое распространение огня по лесным территориям наносит огромный ущерб экосистемам, зачастую полностью уничтожая их. От скорости обнаружения лесного пожара зависит эффективность его тушения и количество ресурсов, затраченное на пожарные мероприятия. Наиболее эффективным способом обнаружения лесных пожаров является спутниковый мониторинг. Данный метод позволяет предоставить наиболее полную информацию о всей лесной территории и значительно дешевле по сравнению с другими методами. Существует устоявшийся опыт обнаружения пожара с использованием алгоритмов, разработанных для различных датчиков, позволяющих использовать преимущества повышенной яркости огня на более длинных волнах. Однако, тепловые аномалии, обнаруживаемые спутниками, зачастую связаны с деятельностью человека – большие города, промышленные предприятия могут быть ошибочно приняты за область, охваченную пожаром. Поэтому поставленную задачу нельзя решить простыми алгоритмами.

Обнаружение пожаров на спутниковых снимках – задача семантической сегментации. Для решения таких задач стандартным решением является использование моделей глубокого обучения: в медицине для анализа рентгеновских снимков и данных компьютерной томографии [1], в анализе видео с видеорегистраторов [2], управлении роботизированными манипуляторами [3]. Развивающейся является тематика использования моделей глубокого обучения для сегментации спутниковых данных. Ba et al. (2019) [4] использовали классификационную архитектуру на основе ResNet, чтобы отличить дым от аналогичных объектов, таких как облака, пыль, дымка, суша и побережье, на спутниковых снимках. Gargiulo et al. (2019) [5] предложили архитектуру на основе CNN для улучшения сверхразрешения изображений в конкретном контексте лесных пожаров. Langford et al. (2018) [6] исследовали методы устранения проблемы несбалансированной классификации, обычно присутствующая при сегментации пожаров. Bermudez et al. (2019) [7] использовали генеративную состязательную сеть (GAN) для синтеза отсутствующих или поврежденных мультиспектральных оптических изображений с многовременными данными, причем одно приложение было сосредоточено на обнаружении лесных пожаров. Pinto et al. (2020) [8] использовали глубокий обучающий подход к картографированию и датированию подвергшихся пожару территорий.

Все эти исследования подводили к решению задачи обнаружения лесных пожаров с использованием сверточных нейронных сетей. Именно такое решение показали Preria et al. [9]. В нашей работе мы использовали похожий подход сегментации пожаров с использованием моделей глубокого обучения.

# 1. Постановка задачи

Целью данной работы является создание сервиса, позволяющего отследить на карте лесные пожары, обнаруженные в заданной области в конкретный момент времени.

Вопрос обнаружения пожаров важен в первую очередь для специальных спасательных служб, поэтому интерфейс программы будет выполнен в виде десктопного приложения, и основные вычисления будут производиться на стороне пользователя. Также будем считать, что пользователя интересует только карта пожаров Нижегородской области.

Задача обнаружения лесных пожаров сводится к задаче семантической сегментации спутниковых снимков. Иными словами, для данного снимка необходимо определить все пиксели изображения, на которых есть лесной пожар. Для решения этой задачи будет проведена тренировка модели глубокого обучения.

# 2. Подход к сегментации с использованием глубокого обучения

## 2.1. Подготовка набора данных

Для подготовки набора данных были использованы ресурсы из открытого git-репозитория[[1]](#footnote-2), подготовленного авторами статьи Preria et al. [9]. Был составлен набор из 6313 пар изображение-маска, 80% которого использовалось для обучения модели, 20% использовалось для валидации и одновременно в качестве тестового набора.

### 2.1.1. Подготовка спутниковых снимков

Для получения изображений был выбран спутник Landsat 8 [[2]](#footnote-3). Спутник ведет съемку в 11 каналах. Известно, что большая часть излучения при лесных пожарах приходится на диапазон длин волн 2,9 - 3,6 мкм. [10] Также известно, что температурные аномалии могут быть вызваны не только пожарами, но и деятельностью человека, например, промышленным производством или большим городом. Поэтому из предоставленных спутником каналов было выбрано пять: три канала видимого света и два канала короткого инфракрасного излучения (то есть каналы 2, 3, 4, 6, 7 из таблицы в приложении 1).

Проблема обработки данных, полученных непосредственно со спутника нетривиальна, однако нами были использованы уже обработанные изображения из ранее упомянутого репозитория. Они представляют собой изображения размером 256 x 256 в формате TIFF. Это растровый формат, позволяющий хранить изображения с большой глубиной цвета без сжатия или с минимальным сжатием – без потерь данных.

Такие изображения требуют большого объема памяти для хранения, поэтому из исходного датасета был сформирован уменьшенный размером 6313. В него вошли только изображения, содержащие достаточное количество пикселей, помеченных, как пожары (не менее 25). Датасет содержит изображения, сделанные в разных местах на разных континентах.

### 2.1.2. Подготовка масок

Маски к изображениям также были взяты из упомянутого репозитория. Из 6313 масок, 100 были созданы вручную, а остальные были сформированы алгоритмами Schroeder et al [11], Murphy et al. [12], Kumar-Roy[13], а также их комбинациями (подробнее в статье [9]).

Маски представляют собой черно-белые изображения 256 x 256, где белым отмечены пиксели с пожаром.

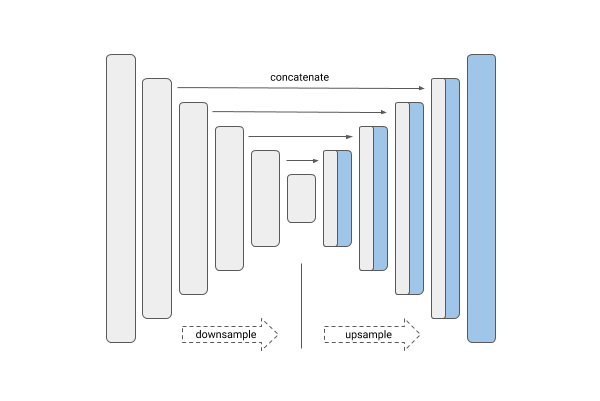
## 2.2. Разработка модели

### 2.2.1. Использованные библиотеки

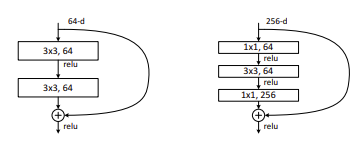
Обучение моделей производилось на языке Python с использованием библиотеки Keras (TensorFlow). Архетиктура моделей строилась с помощью библиотеки Segmentation Modes. Подробнее можно ознакомиться в нашем github-репозитории[[3]](#footnote-4). Некоторые файлы с кодом реализации обучения моделей представлены в приложении 3.

### 2.2.2. Архитектура модели

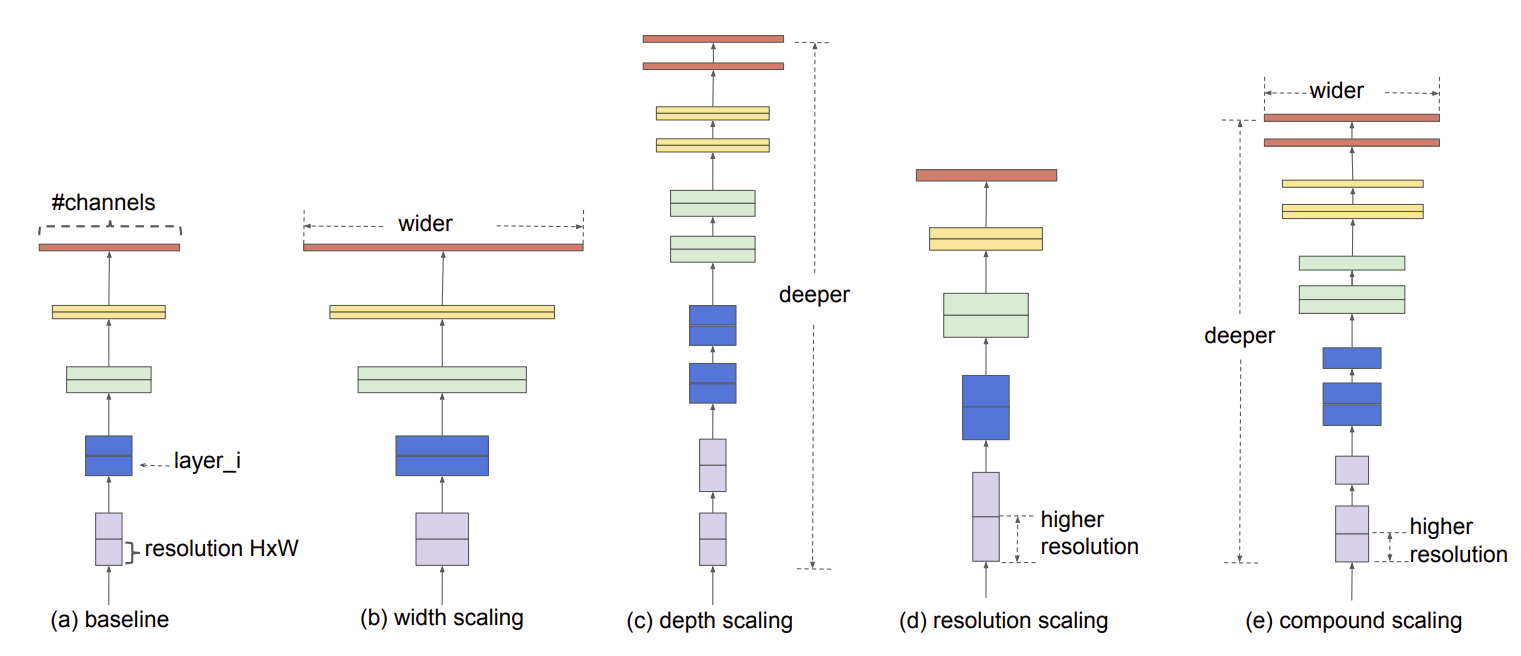
В качестве модели как стандартное в данной области решение был выбран U-Net [14] вместе со сверточными основами ResNet [15] и EfficientNet [16] с разным количеством сверточных слоев.



**Рисунок 1.** Архитектура U-Net состоит из стягивающего пути для захвата контекста и симметричного расширяющегося пути.



**Рисунок 2**. Остаточные блоки являются особенностью архитектуры ResNet и позволяют справиться с проблемой затухания градиентов.



**Рисунок 3**. Методы масштабирования глубины и ширины приводят к высокой эффективности моделей EfficientNet.

### 2.2.3. Метрики

В нашем наборе данных наблюдается сильная несбалансированность классов (в исходном наборе более 99% картинок составляли не горящие пиксели, описанный в 2.1.1 отбор незначительно улучшил этот показатель). В качестве метрики использовался индекс Жаккара (Intersection over Union), так как данная метрика хорошо работает в условиях сильной несбалансированности классов. Индекс Жаккарда измеряет сходство между конечными наборами выборок и определяется как размер пересечения, деленный на размер объединения наборов выборок:

### 2.2.4. Функция потерь

В качестве функции потерь была выбрана Jaccard loss, так как данная функция хорошо работает в условиях сильной несбалансированности классов:

## 2.3. Результаты

### 2.3.1. Результаты обучения

Используя описанные в пункте 2. данные и общие характеристики модели, мы производили запуск моделей с разными параметрами: архитектура сверточной основы, количество слоев, оптимизатор, темп обучения.

Приведем результаты оценки точности на тестовом наборе данных некоторых из опробованных моделей.

**Таблица 1.** Оценка точности некоторых из обученных нами моделей.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Backbone | Trainable parameters | Optimizer | Learning rate | Epochs | IoU score, % |
| Unet | EfficientNetb0 | 10,072,077 | Adam | 0.001 | 30 | 73.1 |
| Unet | EfficientNetb1 | 12,577,713 | Adam | 0.001 | 60 | 73.2 |
| Unet | EfficientNetb2 | 14,226,323 | Adam | 0.001 | 30 | 73.9 |
| Unet | EfficientNetb3 | 17,779,273 | Adam | 0.0007 | 30 | 72.7 |
| Unet | ResNet-34 | 24,445,078 | Adam | 0.0007 | 30 | 73.5 |
| Unet | EfficientNetb4 | 25,608,697 | Adam | 0.001 | 30 | 72.5 |
| Unet | ResNet50 | 32,519,830 | NovoGrad | 0.0007 | 30 | 66.4 |
| Unet | ResNet-50 | 32,519,830 | Adam | 0.001 | 60 | 75.3 |

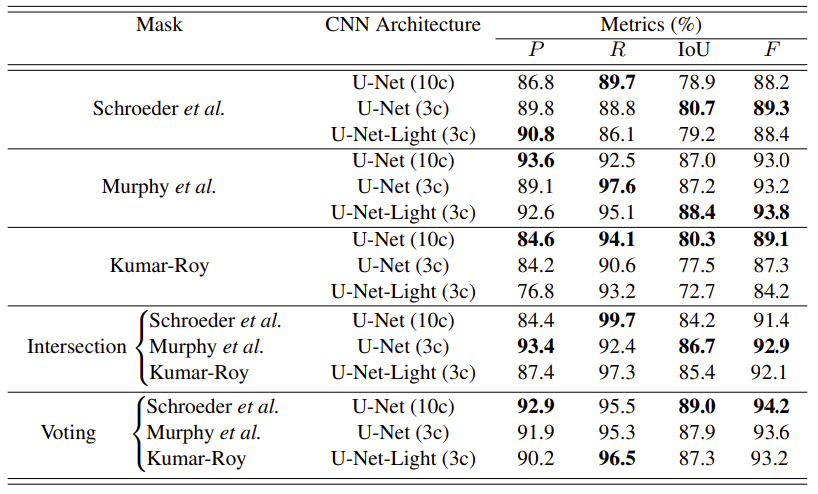
Полный список опробованных моделей и результаты их обучения представлены в приложении 2.

### 2.3.2. Сравнение с существующими решениями

Аналогичную задачу сегментации пожаров на спутниковых снимках с использованием алгоритмов глубокого обучения решали Preria et al. [9]. В данной работе использовался тот же самый набор данных, что и в нашей работе.

Ниже представленны данные, взятые из их статьи: лучшая из представленных моделей достигла 89.0% IoU score, что более чем на 15% лучше достигнутого нами значения 73.5 % IoU score.

**Таблица 2.** Оценка точности моделей, обученных Preria et al. , на тестовом наборе, подготовленном Preria et al.



Мы также сделали оценку точности моделей, обученных Preria et al., на сформированном нами тестовом наборе данных (оценка наших моделей была получена именно на этом наборе).

**Таблица 3.** Оценка точности моделей, обученных Preria et al., на тестовом наборе, подготовленном нами

|  |  |  |
| --- | --- | --- |
| Mask | CNN Architecture | IoU, % |
| Schroeder et al. | U-Net (10c) | 0.2 |
| U-Net (3c) | 0.7 |
| U-Net-Litght (3c) | 0.6 |
| Murphy et al. | U-Net (10c) | **40.4** |
| U-Net (3c) | 33.1 |
| U-Net-Litght (3c) | 7.01 |
| Kumar-Roy | U-Net (10c) | 20.2 |
| U-Net (3c) | 5.9 |
| U-Net-Litght (3c) | 11.6 |
| Intersection | U-Net (10c) | 19.1 |
| U-Net (3c) | 29.2 |
| U-Net-Litght (3c) | 25.1 |
| Voting | U-Net (10c) | 31.4 |
| U-Net (3c) | 9.8 |
| U-Net-Litght (3c) | 12.8 |

На таком тестовом наборе лучшим результатом оказался 40.4 % IoU, что значительно хуже как результатов тех же моделей на описанных в статье наборах данных, так и результатов обученных нами моделей.

Стоит отметить, что Preria et al. формировали тестовый набор данных, а также наборы для обучения и валидации по другой стратегии. Модели отдельно обучались на наборах, в каждом из которых маски были сгенерированы конкретным алгоритмом. Тогда как в нашей работе набор данных содержал маски, полученные с помощью разных алгоритмов.

# 3. Пользовательский интерфейс

## 3.1. Программная реализация

### 3.1.1. Загрузка спутниковых снимков

Для загрузки спутниковых снимков используется Processing API, предоставленный сервисом Sentinel Hub.

### 3.1.2. Графический интерфейс

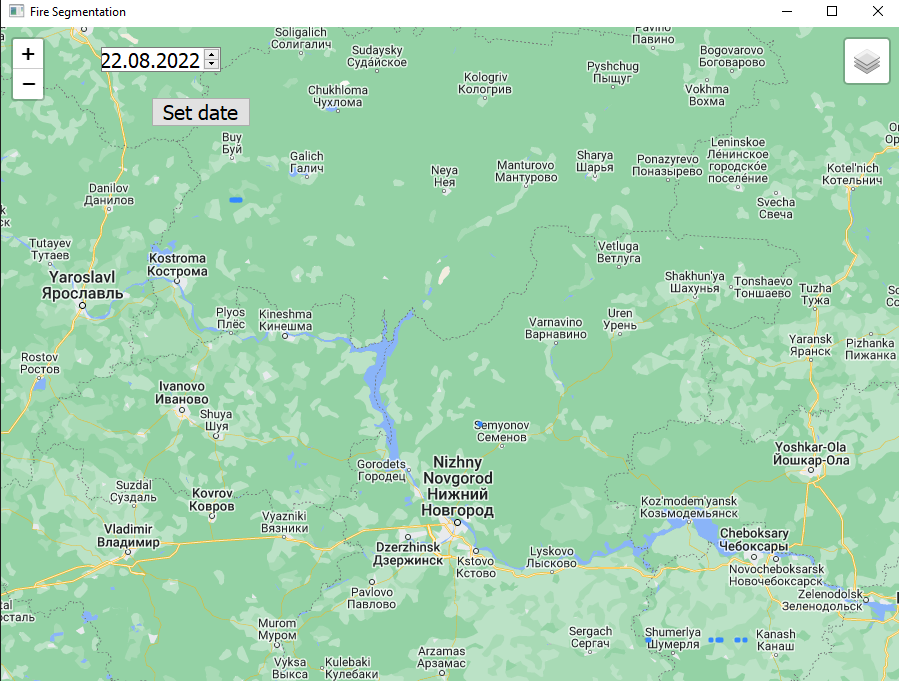
Интерфейс реализован на языке Python с помощью библиотек PyQt5 и folium. Folium используется для отображения интерактивной карты, а PyQt5 – для реализации интерфейса и логики его работы. Интерактивная карта сохраняется в формате html и отображается с помощью QWebEngineView – встроенного в PyQt5 средства отображения веб страниц.

Отрисовка элементов на карте работает с помощью запуска javascript кода на странице через QWebEngineView.

Подробнее можно ознакомиться в нашем github-репозитории[[4]](#footnote-5). Некоторые файлы с кодом реализации пользовательского интерфейса представлены в приложении 4.

## 3.2. Руководство пользователя

На экране приложения изначально будет интерактивная карта, которую можно двигать, приближать и отдалять, поле для ввода даты и кнопка ввода. При нажатии кнопки ввода программа запросит спутниковый снимок не старше указанной в поле даты, обработает его, и закрасит часть интерактивной карты, в которой наиболее вероятно нахождение пожара.



**Рисунок 4.** Скриншот интерфейса приложения.

# Заключение

В данной работе мы рассмотрели проблему обнаружения лесных пожаров с использованием методов глубокого обучения. Мы провели исследование, обучив модели разной архитектуры с различными параметрами, и получили модель глубокого обучения, точность которой близка к точности уже существующих решений (а на некоторых выборках даже значительно выше). Кроме того, мы создали десктопное приложение, позволяющее определять лесные пожары на карте Нижегородской области с использованием обученных нами моделей. Таким образом, мы создали сервис по обнаружению пожаров, пройдя полный путь от подготовки моделей глубокого обучения до автоматизации получения спутниковых снимков и создания пользовательского интерфейса.

# Список литературы

[1] Lachinov, D.A. Glioma Segmentation with Cascaded UNet / D.A. Lachinov, E.P. Vasilyev, V.E. Turlapov // Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. BrainLes 2018. // LNCS – 2019. – Vol. 11384. – P. 189–198.

[2] Rethinking atrous convolution for semantic image segmentation. https://arxiv.org/abs/1706.05587

[3] Sun, G. Robotic grasping using semantic segmentation and primitive geometric model based 3d pose estimation / G. Sun, H. Lin // IEEE/SICE International Symposium on System Integration – 2020. – P. 337–342.

[4] Ba R. et al. SmokeNet: Satellite smoke scene detection using convolutional neural network with spatial and channel-wise attention //Remote Sensing. – 2019. – Т. 11. – №. 14. – С. 1702.

[5] Gargiulo M. et al. A CNN-based super-resolution technique for active fire detection on Sentinel-2 data //2019 PhotonIcs & Electromagnetics Research Symposium-Spring (PIERS-Spring). – IEEE, 2019. – С. 418-426.

[6] Langford Z., Kumar J., Hoffman F. Wildfire mapping in Interior Alaska using deep neural networks on imbalanced datasets //2018 IEEE International Conference on Data Mining Workshops (ICDMW). – IEEE, 2018. – С. 770-778.

[7] Bermudez J. D. et al. Synthesis of multispectral optical images from SAR/optical multitemporal data using conditional generative adversarial networks //IEEE Geoscience and Remote Sensing Letters. – 2019. – Т. 16. – №. 8. – С. 1220-1224.

[8] Pinto M. M. et al. A deep learning approach for mapping and dating burned areas using temporal sequences of satellite images //ISPRS Journal of Photogrammetry and Remote Sensing. – 2020. – Т. 160. – С. 260-274.

[9] Pereira, G.H. Active fire detection in Landsat-8 imagery: A large-scale dataset and a deep-learning study / G.H. Pereira, A.M. Fusioka, N.B. Tomoyuki, R. Minetto // ISPRS Journal of Photogrammetry and Remote Sensing. – 2021. – Vol. 178. – P. 171–186.

[10] Карпов А. А., Алешко Р. А., Шошина К. В. Технологии определения природных пожаров с использованием данных спутниковой съемки //Молодой ученый. – 2015. – №. 13-1. – С. 17-19.

[11] Schroeder W. et al. Active fire detection using Landsat-8/OLI data //Remote sensing of environment. – 2016. – Т. 185. – С. 210-220.

[12] Murphy S. W. et al. HOTMAP: Global hot target detection at moderate spatial resolution //Remote Sensing of Environment. – 2016. – Т. 177. – С. 78-88.

[13] Kumar S. S., Roy D. P. Global operational land imager Landsat-8 reflectance-based active fire detection algorithm //International Journal of Digital Earth. – 2018. – Т. 11. – №. 2. – С. 154-178.

[14] Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation //Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. – Springer International Publishing, 2015. – С. 234-241.

[15] He K. et al. Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 770-778.

[16] Tan M., Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks //International conference on machine learning. – PMLR, 2019. – С. 6105-6114.

# Приложение

## Приложение 1

Полный список каналов, на которых ведется снимка спутником Landsat 8

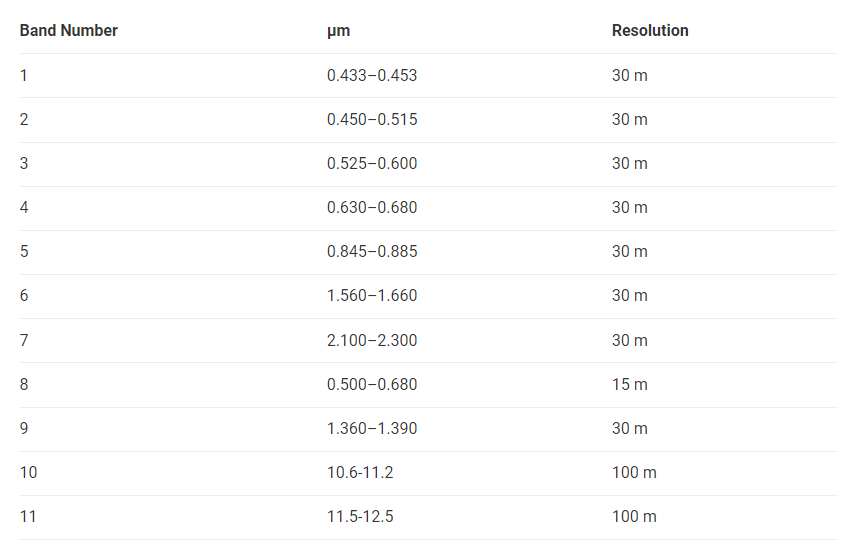


Таблица взята с сайта NASA (https://landsat.gsfc.nasa.gov/satellites/landsat-8/landsat-8-bands/).

## Приложение 2

Поиск лучших параметров моделей, описанных в пунке 2.2

|  |
| --- |
| Search space summary  Default search space size: 3  backbone (Choice)  {'default': 'resnet34', 'conditions': [], 'values': ['resnet34', 'resnet50', 'resnet101', 'efficientnetb3', 'efficientnetb4'], 'ordered': False}  learning rate (Float)  {'default': 0.0001, 'conditions': [], 'min\_value': 0.0001, 'max\_value': 0.001, 'step': 0.0003, 'sampling': 'linear'}  optimizer (Int)  {'default': None, 'conditions': [], 'min\_value': 0, 'max\_value': 1, 'step': 1, 'sampling': 'linear'}  Note: optimizer “0” – Adam, optimizer “1” - NovoGrad  Best val\_iou\_score So Far: 0.7352995872497559  Total elapsed time: 18h 36m 31s  Results summary  Results in ./untitled\_project  Showing 15 best trials  Objective(name="val\_iou\_score", direction="max")  Trial 03 summary  Hyperparameters:  backbone: resnet50  learning rate: 0.001  optimizer: 0  Score: 0.7352995872497559  Trial 09 summary  Hyperparameters:  backbone: resnet34  learning rate: 0.0007  optimizer: 0  Score: 0.7352643013000488  Trial 10 summary  Hyperparameters:  backbone: resnet34  learning rate: 0.001  optimizer: 0  Score: 0.7328929901123047  Trial 12 summary  Hyperparameters:  backbone: resnet34  learning rate: 0.0007  optimizer: 0  Score: 0.7328076362609863  Trial 13 summary  Hyperparameters:  backbone: resnet34  learning rate: 0.0007  optimizer: 0  Score: 0.7318616509437561  Trial 06 summary  Hyperparameters:  backbone: efficientnetb3  learning rate: 0.0007  optimizer: 0  Score: 0.7278991341590881  Trial 11 summary  Hyperparameters:  backbone: resnet50  learning rate: 0.0007  optimizer: 0  Score: 0.7273863554000854  Trial 01 summary  Hyperparameters:  backbone: efficientnetb4  learning rate: 0.001  optimizer: 0  Score: 0.7248330116271973  Trial 05 summary  Hyperparameters:  backbone: efficientnetb4  learning rate: 0.0007  optimizer: 0  Score: 0.7203198075294495  Trial 14 summary  Hyperparameters:  backbone: efficientnetb3  learning rate: 0.0007  optimizer: 0  Score: 0.7186773419380188  Trial 07 summary  Hyperparameters:  backbone: resnet34  learning rate: 0.0001  optimizer: 0  Score: 0.6813756227493286  Trial 00 summary  Hyperparameters:  backbone: resnet101  learning rate: 0.0001  optimizer: 0  Score: 0.6728431582450867  Trial 02 summary  Hyperparameters:  backbone: resnet50  learning rate: 0.0007  optimizer: 1  Score: 0.6635692715644836  Trial 08 summary  Hyperparameters:  backbone: resnet34  learning rate: 0.0001  optimizer: 1  Score: 0.5498355627059937  Trial 04 summary  Hyperparameters:  backbone: efficientnetb4  learning rate: 0.0001  optimizer: 1  Score: 0.5111038684844971 |

## Приложение 3

Пример кода работы с моделями глубокого обучения: поиск гиперпараметров модели.

|  |
| --- |
| import os  import tensorflow as tf  import keras  import segmentation\_models as sm  from keras.utils import load\_img  import numpy as np  import imageio  from glob import glob  import math  from random import randint  from skimage.transform import rotate, rescale, resize  from tensorflow import keras  from keras.optimizers import Adam  from tensorflow.keras.optimizers.experimental import AdamW  from tensorflow\_addons.optimizers import NovoGrad  import keras\_tuner  # available GPUs  print("Num GPUs Available: ", len(tf.config.list\_physical\_devices('GPU')))  gpus = tf.config.experimental.list\_physical\_devices('GPU')  for gpu in gpus:  tf.config.experimental.set\_memory\_growth(gpu, True)  # paths  # your project directory  project\_dir = 'fire/'  '''  project\_dir  |  | -- fire.py  |  | -- dataset  | | -- patches  | | -- folder1  | | -- ...  | | -- masks  | | -- folder1  | | -- ...  |  | -- predictions  | | -- hh-mm-ss  | | -- image.png  | | -- mask.png  | | -- predict.png  | | -- ...  '''  # project paths  save\_path = os.path.join(project\_dir, 'model.h5') # path for saving model  search\_path = os.path.join(project\_dir, 'hp\_search-25-04-2023-2.h5') # best during hp search  saved\_images = os.path.join(project\_dir, 'predictions') # save all images here  dataset\_dir = project\_dir # !  patches\_dir = os.path.join(dataset\_dir, 'patches')  masks\_dir = os.path.join(dataset\_dir, 'masks')  # Segmentation Models: using `keras` framework.  os.environ["SM\_FRAMEWORK"] = "tf.keras"  # get paths  patches = glob(patches\_dir + '/\*\*/\*.tif', recursive=True)  masks = glob(masks\_dir + '/\*\*/\*.tif', recursive=True)  patches.sort()  masks.sort()  print('{0} patches and {1} masks were found'.format(len(patches), len(masks)))  annotated = list()  annotation = list()  for patch in patches:  name = patch[-10:]  batch = patch[13:52]  if batch[1] == "a":  batch = batch[15:]  matched = filter(lambda x: batch in x and name in x, masks)  matched = list(matched)  try:  mask = matched[0]  annotation.append(mask)  annotated.append(patch)  except Exception as e:  print("Exception {0} while searching for matched masks for patch {1}".format(e, patch))  patches = annotated  masks = annotation  # open images  max\_ds\_size = 10000  ds\_size = min(max\_ds\_size, min(len(patches), len(masks)))  print('final dataset length: {0}'.format(ds\_size))  print('validation split = 0.2')  channels = 5  # normalization layer  normalization = keras.layers.Normalization(axis=1)  adapt\_data = np.empty(shape=(ds\_size, 256, 256, channels), dtype=np.float32)  for i in range(ds\_size):  x = imageio.imread(patches[i])  x = np.asarray(x).astype('float32')  x = x.transpose((2, 0, 1))  x = np.array([x[1], x[2], x[3], x[4], x[5]])  x = x.transpose((1, 2, 0))  adapt\_data[i] = x  normalization.adapt(adapt\_data, batch\_size=100, steps=ds\_size)  adapt\_data = 0  # open and normalize  train\_x = list()  train\_y = list()  val\_x = list()  val\_y = list()  for i in range(ds\_size):  x = imageio.imread(patches[i])  x = np.asarray(x).astype('float32')  x = x.transpose((2, 0, 1))  x = np.array([x[1], x[2], x[3], x[4], x[5]])  x = x.transpose((1, 2, 0))  x = normalization(x).numpy()[0]  y = imageio.imread(masks[i])  y = np.asarray(y).astype('float32')  if (i % 5 == 0):  val\_x.append(x)  val\_y.append(y)  else:  train\_x.append(x)  train\_y.append(y)  # Here we use \*\*\*channel last\*\*\* format  # image is a tensor with (H, W, C) shape, where C is a number of channels, H and W are image height and width.  #  # But rasterio opens image in \*\*\*channel first\*\*\* format, so you need to do `.transpose((1, 2, 0))`  # data generation and augmentation  # Here, `x\_set` is list of patches  # and `y\_set` is the list of masks  def augmentation(image, mask):  if randint(0, 1) == 1:  image = np.fliplr(image)  mask = np.fliplr(mask)  if randint(0, 1) == 1:  image = np.flipud(image)  mask = np.flipud(mask)  angle = randint(-45, 45)  image = rotate(image, angle, mode='symmetric')  mask = rotate(mask, angle, mode='symmetric')  return image, mask  class DataGenerator(tf.keras.utils.Sequence):  def \_\_init\_\_(self, x\_set, y\_set, batch\_size, augmentation=False, shuffle=True):  self.x, self.y = x\_set, y\_set  self.batch\_size = batch\_size  self.shuffle = shuffle  self.index = 0  self.on\_epoch\_end()  def on\_epoch\_end(self):  '''  # good shuffle  self.index = np.arange(len(self.indices))  if self.shuffle == True:  np.random.shuffle(self.index)  '''  # my shuffle  index = randint(0, len(self) - 1)  def \_\_len\_\_(self):  return math.ceil(len(self.x) / self.batch\_size)  def \_\_getitem\_\_(self, index):  batch\_x = np.array(self.x[index \* self.batch\_size:(index + 1) \*  self.batch\_size])  batch\_y = np.array(self.y[index \* self.batch\_size:(index + 1) \*  self.batch\_size])  if (augmentation):  for x, y in zip(batch\_x, batch\_y):  x, y = augmentation(x, y)  return (batch\_x, batch\_y)  def \_\_next\_\_(self):  self.index += 1  if (self.index >= len(self)):  self.index = 0  return self[self.index]  batch\_size = 80  val\_split = 0.2  train\_generator = DataGenerator(train\_x, train\_y, batch\_size, augmentation=True)  val\_generator = DataGenerator(val\_x, val\_y, batch\_size, augmentation=False)  # steps in model.fit()  steps = math.ceil(len(train\_generator))  val\_steps = math.ceil(len(val\_generator))  # hyperparameter search  def build\_model(hp):  BACKBONE = hp.Choice('backbone', values=['resnet34', 'resnet50', 'efficientnetb0', 'efficientnetb1', 'efficientnetb2'])  # модель - Unet  model = sm.Unet(BACKBONE,  classes=1,  encoder\_weights=None,  input\_shape=(None, None, 5),  activation='sigmoid')  lr = hp.Float('learning rate', min\_value=0.0002, max\_value=0.001, step = 0.0004)  opt\_list = [  Adam(learning\_rate=lr),  NovoGrad(learning\_rate=lr),  AdamW(learning\_rate=lr),  ]  opt\_number = hp.Int('optimizer', min\_value=0, max\_value=len(opt\_list) - 1)  model.compile(  opt\_list[opt\_number],  loss=sm.losses.JaccardLoss(),  metrics=[sm.metrics.IOUScore()],  )  return model  tuner = keras\_tuner.BayesianOptimization(  build\_model,  objective=keras\_tuner.Objective("val\_iou\_score", direction="max"),  overwrite=True,  max\_trials=12,  # max\_epochs=50,  )  tuner.search\_space\_summary()  tuner.search(  train\_generator,  epochs=15,  steps\_per\_epoch=steps,  validation\_data=val\_generator,  validation\_steps=val\_steps, )  tuner.results\_summary(num\_trials=12) |

## Приложение 4

Пример кода работы с пользовательским интерфейсом: графический интерфейс

|  |
| --- |
| import random  import io  import sys  from jinja2 import Template  import folium  from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QHBoxLayout, QVBoxLayout, QDateEdit, QPushButton  from PyQt5.QtWebEngineWidgets import QWebEngineView, QWebEnginePage  import utils.show  from utils.process import get\_mask, get\_image  import matplotlib.pyplot as plt  import numpy  class WebEnginePage(QWebEnginePage):  def javaScriptConsoleMessage(self, level, message, lineNumber, sourceID):  print("javaScriptConsoleMessage: ", level, message, lineNumber, sourceID)  def getLatLon(i, j):  #return (54.290882 + (58.263287 - 54.290882) / 256 \* i, 41.389207 + (48.151968 - 41.389207) / 256 \* j, 54.290882 + (58.263287 - 54.290882) / 256 \* (i + 1), 41.389207 + (48.151968 - 41.389207) / 256 \* (j + 1))  return (54.500526 + (55.263287 - 54.500526) / 256 \* i, 41.389207 + (42.151968 - 41.389207) / 256 \* j, 54.500526 + (55.263287 - 54.500526) / 256 \* (i + 1), 41.389207 + (42.151968 - 41.389207) / 256 \* (j + 1))  class MyApp(QMainWindow):  def onLoad(self):  js = Template("var f = true; var markerGroup = L.layerGroup().addTo({{map}});").render(map=self.map.get\_name())  self.webView.page().runJavaScript(js);    def \_\_init\_\_(self):  super().\_\_init\_\_()  self.setWindowTitle('Fire Segmentation')  self.window\_width, self.window\_height = 900, 1000  self.setMinimumSize(self.window\_width, self.window\_height)  layout = QVBoxLayout()  #self.setLayout(layout)  coordinate = (56.296505, 43.936058)  self.map = folium.Map(  zoom\_start=7, location=coordinate, control\_scale=True, tiles=None  )  folium.raster\_layers.TileLayer(  tiles="http://mt1.google.com/vt/lyrs=m&h1=p1Z&x={x}&y={y}&z={z}",  name="Standard Roadmap",  attr="Google Map",  ).add\_to(self.map)  folium.LayerControl().add\_to(self.map)  data = io.BytesIO()  self.border = [[54.290882, 41.389207], [54.290882, 48.151968], [58.263287, 48.151968], [58.263287, 41.389207]]  #folium.Polygon(locations = border, fill\_color="orange").add\_to(self.m)    self.map.save(data, close\_file=False)  self.webView = QWebEngineView()  self.webView.setPage(WebEnginePage(self.webView))  self.webView.setHtml(data.getvalue().decode())  layout.addWidget(self.webView)  self.setCentralWidget(self.webView)  self.webView.loadFinished.connect(self.onLoad)    self.date\_edit = QDateEdit(self)  self.date\_edit.setGeometry(100, 20, 120, 25)  self.button = QPushButton('Set date', self)  self.button.move(150,70)  self.button.clicked.connect(self.update)    def update(self):  value = self.date\_edit.date()    print("done")  print("{}T00:00:00Z".format(value.toString("yyyy-MM-dd")))  print("done")  time\_interval = ("{}T00:00:00Z".format(value.toString("yyyy-MM-dd")), "2023-01-18T00:00:00Z")  self.webView.page().runJavaScript("markerGroup.clearLayers();")  bbox = [41.389207, 54.500526, 42.151968, 55.263287]  y = get\_mask(bbox, time\_interval, "model-resnet50-novograd-0008.h5")  print(y)  k = numpy.size(y, 0)  print(numpy.size(y, 0))  print(numpy.size(y, 1))  print(numpy.size(y, 2))    for i in range(256):  for j in range(256):  for i1 in range(int(numpy.size(y, 0) / 256)):  for j1 in range(int(numpy.size(y, 1) / 256)):  #print(y[i \* int(numpy.size(y, 0) / 256) + i1][j \* int(numpy.size(y, 1) / 256) + j1])  if y[i \* int(numpy.size(y, 0) / 256) + i1][j \* int(numpy.size(y, 1) / 256) + j1][0] >= 0.00000000000001:    js = Template(  """  L.polygon(  [[{{firstLat}}, {{firstLon}}], [{{secondLat}}, {{secondLon}}], [{{thirdLat}}, {{thirdLon}}], [{{fourthLat}}, {{fourthLon}}]], fill\_color='#fc0000', {weight: 0}).addTo(markerGroup); """  #).render(map=self.map.get\_name(), firstLat = self.border[0][0], firstLon = self.border[0][1],  # secondLat = self.border[1][0], secondLon = self.border[1][1],  # thirdLat = self.border[2][0], thirdLon = self.border[2][1],  # fourthLat = self.border[3][0], fourthLon = self.border[3][1],  # c1 = i, c2 = j)  ).render(firstLat = getLatLon(i, j)[0], firstLon = getLatLon(i, j)[1],  secondLat = getLatLon(i, j)[2], secondLon = getLatLon(i, j)[1],  thirdLat = getLatLon(i, j)[2], thirdLon = getLatLon(i, j)[3],  fourthLat = getLatLon(i, j)[0], fourthLon = getLatLon(i, j)[3],  c1 = i, c2 = j)  self.webView.page().runJavaScript(js)  break  else:  continue  break        if \_\_name\_\_ == '\_\_main\_\_':  app = QApplication(sys.argv)  app.setStyleSheet(''' QWidget { font-size: 20px; } ''')  myApp = MyApp()  myApp.show()  try:  sys.exit(app.exec\_())  except SystemExit:  print('Closing Window...') |

1. <https://github.com/pereira-gha/activefire> [↑](#footnote-ref-2)
2. <https://landsat.gsfc.nasa.gov/satellites/landsat-8/> [↑](#footnote-ref-3)
3. https://github.com/IgorKonovalovAleks/fire-segmentation [↑](#footnote-ref-4)
4. https://github.com/IgorKonovalovAleks/fire-segmentation [↑](#footnote-ref-5)