

Московский Государственный Университет им. Ломоносова

Факультет Вычислительной Математики и Кибернетики



Сегментация изображений с помощью нейронной сети U-Net

Юдина Диана

217 группа

Москва

2025

Содержание

Введение	3
Почему именно U-Net ?	3
Архитектура U-Net	4
Encoder (Contracting Path)	4
Decoder (Expanding Path)	4
Проект	5
Программная реализация	6
Заключение	8
Литература	9

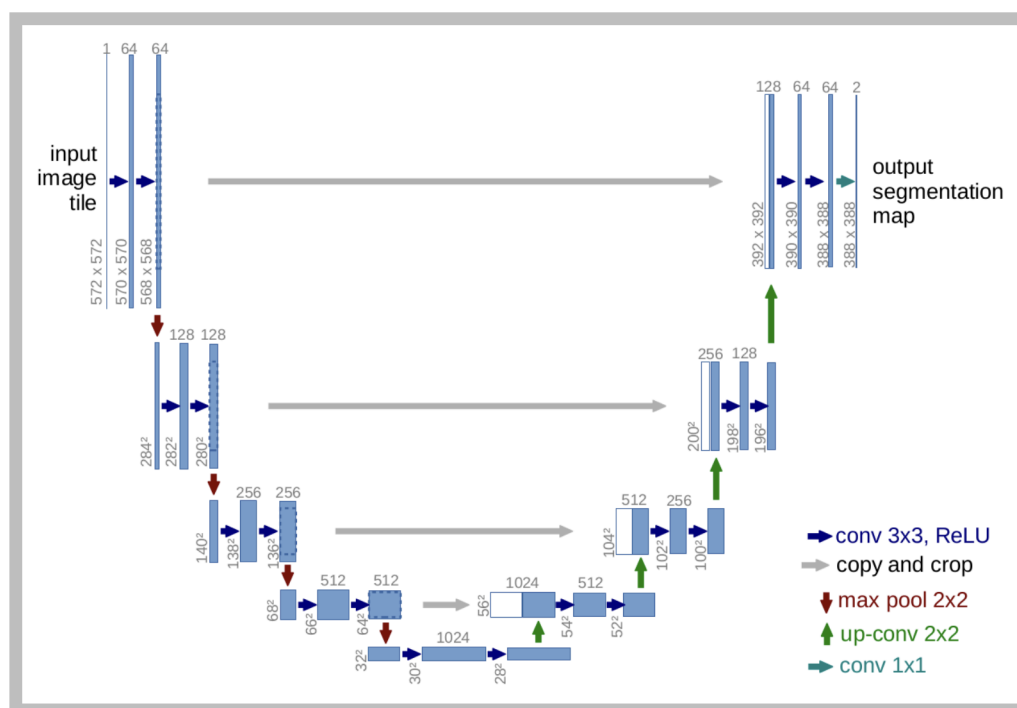
Введение

Сегментация изображений - разбиение изображения на множество покрывающих его областей. Сегментация изображений используется во многих областях: медицина, геология, астрономия и т.д. В этом проекте мы рассматриваем сегментацию рудных металлов с помощью сверточной нейросети U-Net.

Почему именно U-Net ?

Изначально для сегментации изображений выбор пал на кластеризацию (KMeans) , так как это классический метод, так как это простой, точный и быстрый метод для обработки, но к сожалению, не давал больших показателей. Поэтому было принято решение перейти более к сложной модели - сверточной нейронной сети U-Net, которая показала высокие показатели оценки mean iou, точность, но заняла долгое обучение модели и сложность в написании.

Архитектура сверточной нейронной сети U-Net



U-Net architecture as proposed by [Ronneberger et al. \(2015\)](#).

Encoder (Contracting Path)

Основной задачей Encoder (энкодера, сжимающего пути) является уменьшение размера фотографий с выделением важных признаков, которое позволяет уменьшить пространственное разрешение. В свою очередь сжимающий путь тоже состоит из нескольких частей:

- **Convolutional Layers** (сверточные слои)

Каждый блок содержит два последовательных 3×3 сверточных слоя с функцией активации ReLU

- **Pooling Layers** (субдискретизация)

Pooling уменьшает размер изображений в 2 раза и помогает конкретизировать информацию и уменьшить вычислительную сложность работы модели.

- **Increasing the number of filter features** (увеличение числа фильтров)

По мере движения модели, количество фильтров удваивается (64 -> 128 -> 256 и т.д.). Это помогает еще больше конкретизировать признаки и работать с более сложными данными

Decoder (Expanding Path)

Основной задачей Decoder (восстанавливающей частью, декодером) является восстановление исходного размера, используя информацию из сжимающего пути, т.е. те самые признаки, которые он выделял на каждом этапе, а также точно определить границы объектов. В свою очередь decoder тоже состоит из нескольких частей:

- **Upsampling, Transposed Convolution**

Увеличивает размер изображений в 2 раза

- **Skip Connections**

Skip Connections (прямые соединения) является одной из ключевых этапов в использовании нейросети U-Net. Это этап

соединения Encoder и Decoder, который позволяет восстановить мелкие детали, которые могли быть утеряны в процессе работы модели.

- **Convolutional Layers**

После Skip Connections, выполняются два сверточных слоя с ReLU, это помогает применить признаки, которые мы выделяли ранее с пространственному расширению.

Проект

Проект выполнен в поддержку спецкурса “Практическое введение анализ изображений”.

Основные атрибуты:

- Язык программирования Python
- Импорт библиотеки petroscope
- DataSet:
https://drive.google.com/drive/folders/1eYvegX54xDZkHWwmG6Jrr57sGAUYaR4p?usp=share_link
- Метод, используемый для сегментации изображений: нейронная сеть U-Net

В репозитории <https://github.com/DianaY23/Image-analysis.git> :

1. Файл project.pynb - from Google Collab
2. Файл project.py - python file
3. Файл project.ipynb - Colab.pdf
4. Файл outputs.txt - выходные данные для выборки test

Ссылка на google collab

https://colab.research.google.com/drive/1A_Fh04m-F-vEu6mBQDIkwk2PFmJAP1tC?usp=sharing

Программная реализация

1. GeoSegmModel

Библиотека `petroscope`, предоставляет удобные инструменты для геологической сегментации, включая *GeoSegmModel*, который можно расширить для создания кастомной модели сегментации.

2. Модель U-Net

```
def _build_model(self):
    inputs = Input(self.input_size)

    conv1 = Conv2D(64, 3, activation='relu',
padding='same')(inputs)
    conv1 = Conv2D(64, 3, activation='relu',
padding='same')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation='relu',
padding='same')(pool1)
    conv2 = Conv2D(128, 3, activation='relu',
padding='same')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation='relu',
padding='same')(pool2)
    conv3 = Conv2D(256, 3, activation='relu',
padding='same')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation='relu',
padding='same')(pool3)
    conv4 = Conv2D(512, 3, activation='relu',
padding='same')(conv4)

    up5 = concatenate([UpSampling2D(size=(2, 2))(conv4), conv3],
axis=-1)
    conv5 = Conv2D(256, 3, activation='relu',
padding='same')(up5)
    conv5 = Conv2D(256, 3, activation='relu',
padding='same')(conv5)
    up6 = concatenate([UpSampling2D(size=(2, 2))(conv5), conv2],
axis=-1)
    conv6 = Conv2D(128, 3, activation='relu',
padding='same')(up6)
```

```

        conv6 = Conv2D(128, 3, activation='relu',
padding='same')(conv6)
        up7 = concatenate([UpSampling2D(size=(2, 2))(conv6), conv1],
axis=-1)
        conv7 = Conv2D(64, 3, activation='relu', padding='same')(up7)
        conv7 = Conv2D(64, 3, activation='relu',
padding='same')(conv7)

        outputs = Conv2D(len(self.classes), 1,
activation='softmax')(conv7)

        model = Model(inputs=[inputs], outputs=[outputs])
                                model.compile(optimizer=Adam(),
loss='categorical_crossentropy', metrics=['accuracy'])
        return model

```

Алгоритм реализации функции def _build_model

1. Создаем входной слой модели U-Net, которая принимает изображения размером (256, 256, 3) - где 3 - количество каналов
2. Далее выполняются действия Encoder. В начале работаем с двумя слоями свертки, которые выполняют извлечение признаков. У каждого слоя 64 фильтра, ядро 3x3.
3. Выполняем Pooling, уменьшение размера изображения происходит в два раза. Это поможет уменьшить вычислительную нагрузку, меньше загрузить ОЗУ и , самое главное, извлечь более признаки.
4. На следующем этапе снова выполняется свертка, аналогичная в п.2, но уже с 128 фильтрами (согласно архитектуре нейросети U-Net).
5. Совершаем Pooling, аналогичный п.3, но уже для новых слоев свертки.
6. П.2 с 256, 512 фильтрами

7. Pooling для новых слоев с еще большим уменьшением размера изображений и выявлением новых признаков.
8. Далее выполняется работа уже Decoder. Выполняем увеличение разрешения изображения.
9. Выполняем Skip Connection , т.е. объединяем результаты слоя Encoder и Decoder. Так мы сможем сохранить информацию о признаках, которые мы вывели ранее
10. Далее , согласно архитектуре U-Net, выполняем Upsampling. И конкатенируем результаты с Encoder, чтобы запомнить детали, которые могли быть потеряны во время сжатия. И применяем два слоя свертки с 256 фильтрами, аналогичные действия п.8 выполняем для 128, 64 фильтров.
11. Выполняем свертку 1x1 чтобы получить результат сегментации.
12. Далее “собираем” нашу модель

Заключение

В ходе данного проекта был изучен метод сегментации изображений - нейронная сеть U-Net, а также была реализована задача сегментации рудных минералов.

Архитектура U-Net является одной из наиболее популярных и новых моделей на сегодняшний день. Данная модель позволяет добиться высокой точности показателя mean iou, а также предоставляет работу с большим количеством данных.

Литература

1. Исследование применимости сверточной нейронной сети U-Net к задаче сегментации изображений авиационной техники, *Д.А. Гаврилов*
2. U-Net: Convolutional Networks for Biomedical Image Segmentation, https://boreal17.github.io/paper_summaries/u_net/
3. Multi-Encoder U-Net for Automatic Kidney Tumor Segmentation, Xueying Chen¹ and Chao Xu, https://kits-challenge.org/public/site_media/manuscripts/kits19/cxyjy_3.pdf
4. Simonyan K., Zisserman A. *Very Deep Convolutional Networks for Large-Scale Image Recognition* <https://arxiv.org/pdf/1409.1556>
5. Goodfellow I., Bengio Y., Courville A. *Deep Learning* // MIT Press. – 2016. <https://bjpcjp.github.io/pdfs/dl-goodfellow/toc.pdf>