

# ОТЧЁТ

*задача Семантической Сегментации*

---



*Валеев Арслан*

*02.04.2023*

*ФАКУЛЬТЕТ ВМК, 214 ГРУППА*

## ПОСТАНОВКА ЗАДАЧИ

Требовалось решить задачу бинарной семантической сегментации: обучить модель, которая для спутниковых снимков предсказывает расположение всех строений (то есть строит маску для построек на фотографии). Исходные данные - два спутниковых снимка (города Санта-Роза и Вентура), которые нужно разбить на тренировочную и тестовую часть.

(исходное описание задачи: [building\\_segmentation.pdf](#))

## ПОДХОД К РЕШЕНИЮ

Я решил использовать снимок города Санта-Роза в качестве тренировочной выборки, а город Вентура разбить на валидационную и тестовую часть: при помощи валидационной я буду выбирать наилучшие веса модели, а при помощи тестовой вычислять финальные метрики, которые будут показывать состоятельность моей модели.

**Обработка данных:** для обработки сырых данных (спутниковых снимков) и разбиения их на сэмплы я реализовал класс `Houses_Dataset`, который принимает на вход спутниковый снимок, его разметку и размер сэмпла (стандартное значение - 128), и разбивает фотографию на сэмплы заданного размера. У данного класса есть метод `rebuild`, который заново разбивает снимок и для каждого сэмпла добавляет случайное смещение из нормального распределения (таким образом, при вызове `rebuild` мы получаем формально новый датасет). Также классу можно передать список из аугментаций, так что при выборе объекта из класса к объекту будет применена одна случайно выбранная трансформация. (Использованные мною аугментации: отражение по вертикали/горизонтали, поворот на 90/180/270 градусов)

**Модель:** в качестве модели я реализовал архитектуру U-net (изображение архитектуры с 4-мя слоями: [u-net-architecture.png](#)) с дропаутом и возможностью изменять число слоев и число свёрток в каждом слое. Также я взял предобученную модель (<https://github.com/milesial/Pytorch-UNet>), и попробовал решить ту же задачу.

Описание деталей модели:

*BasicBlock* - состоит из свёртки + batchnorm. Я реализовал этот класс, чтобы четче структурировать модель.

*Block* - слой архитектуры U-net, состоит из  $k$  (стандартное значение  $k = 2$ ) *BasicBlock*, дропаута и maxpool или ConvTranspose (зависит от флага dir).

*Encoder* - часть модели, переводящая изображение в скрытое состояние (у карты признаков она увеличивает число каналов, но уменьшает высоту и ширину, способствуя выявлению более глубоких зависимостей). Использует *Block* с maxpool.

*Decoder* - часть модели, переводящая скрытое состояние в карту признаков, по размерам совпадающую с исходным изображением. (использует *Block* с ConvTranspose).

*Segmentation\_model* - соединяет *Decoder* и *Encoder*, и осуществляет residual connection между ними.

**Метрики:** для валидации модели было решено использовать accuracy и f1-score или dice-metric, что в бинарном случае одно и то же.

**Loss-function:** в качестве лосс-функции я взял сумму кросс-энтропийного критерия (с весом 15 на класс строений) и dice-loss, функция которой оптимизирует dice метрику (f1-score). ([Dice Loss Explained | Papers With Code](#))

**Параметры обучения:** оптимизатор, использованный для этой задачи: SGD с моментом = 0.9 и weight\_decay = 1e-6 (при использовании Adam модель переучивается на трейне). Также был использован lr\_scheduler, чтобы более точно настроить модель. Обучение длилось 700 эпох, чтобы достичь максимально возможного результата. Во время обучения каждые 20 эпох для используемых данных вызывался метод rebuild, чтобы обновить датасет.

**Inference:** для инференса написал функцию, которая  $n$  раз строит предсказания для выборки из полученного на вход класса Houses\_Dataset и вызывает метод rebuild. Таким образом мы получаем  $n$  предсказаний для  $n$  “разных” выборок. После, для каждого пикселя функция выбирала наиболее часто предсказываемый класс, и полученную маску возвращала в качестве результата.

## РЕЗУЛЬТАТЫ

Пусть далее  $M(i)$  - моя реализация U-net с  $i$  - м числом слоев,  $M$  - предобученная модель из интернета.

Модель	f1-score на тестовом участке	ассигасу на тестовом участке	f1-score на всём снимке города Вентура
$M(5)$	0.715	0.925	0.73
$M(4)$	0.717	0.933	0.728
$M(3)$	0.670	0.918	0.679
$M$	0.685	0.917	0.694

Как видно из результатов,  $M(3)$  и  $M(5)$  работают хуже, чем  $M(4)$ . В случае  $M(3)$  понятно, что модели не хватает глубины для изучения сложных паттернов. В случае  $M(5)$  результаты не сильно отличаются от  $M(4)$ , но модель обучалась дольше.  $M$  показала не самый лучший результат, вероятно, из-за небольших различий в архитектуре (например, в  $M$  не хватало дропаута).

Я подбирал немало вариантов гиперпараметров (менял  $lr$ , оптимизатор,  $lr\_scheduler$ ,  $dropout\_rate$ ,  $loss\_function$ ), и как мне кажется, чтобы ощутимо улучшить результат (больше, чем на 2-3 процента), нужно либо поменять архитектуру, либо поменять подход к разбиению данных на тренировочную и тестовую часть (например, разбить каждый снимок на две части, и первые половины обеих фотографий использовать в качестве тренировочной выборки), либо поступить хитро, и обучить модель на спутниковых снимках из интернета, так как в правилах ничего не сказано об использовании сторонних ресурсов.

## ВЫВОД

Была решена задача бинарной семантической сегментации для спутниковых снимков с точностью 0.72 f1-score на тестовом участке.

Тестовый участок: [result\\_test.tif](#)

Весь город: [result\\_all.tif](#)