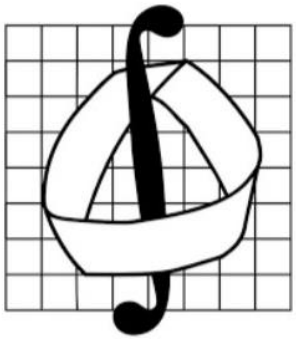
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М.В. ЛОМОНОСОВА

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ТЕОРИИ УПРУГОСТИ



КУРСОВАЯ РАБОТА

“Сегментация биомедицинских изображений рисунка капилляров в сетчатке глаза с использованием нейронных сетей и решение задачи о достаточном наборе данных для обучения”

“Biomedical image segmentation of an eye retinal vessel structure via Neural Networks and solution of statistical problem about sufficient amount of data for model training”

Выполнил студент

325 группы

Ушаков Иван Владимирович

Научный руководитель:

Москва

2020

План курсовой работы:

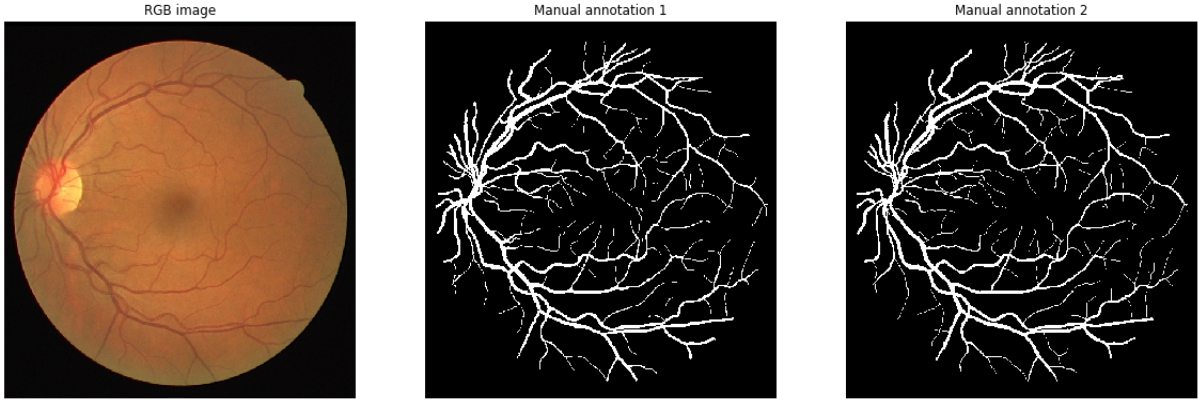
Цели и задачи работы

Исследуются возможности искусственный нейронных сетей третьего поколения в задачах биомедицинской сегментации изображений – определение структуры капилляров в сетчатке глаза с помощью нейронных сетей. Разработка новых методов сегментации изображений является актуальной задачей в медицине, она позволит диагностировать многие заболевания ещё на ранних стадиях.
Были достигнуты существенные продвижение в задаче о необходимом размере данных для достаточного обучения нейронной сети. Предложен и реализован на языке программирования Python 3 сегментатор сосудов – многослойная нейронная сеть и проведено сравнение результатов совместно с моим коллегой Леонидом Костюшко.

Актуальность проекта

Сегментация сосудов в сетчатке глаза и морфологическое описание атрибутов кровеносной системы в сетчатке, таких как длина отдельно взятого капилляра, его толщина, кривизна, витиеватость узора и углы между ними используются для диагностики, исследования, лечения и оценивания состоянии пациентов при различных сердечно сосудистых и офтальмологических заболеваниях. К ним относятся, сахарный диабет, повышенное кровяное давление, артериосклероз, неоваскуляризации. Автоматические определение и анализ сосудистой системы может помочь при внедрении скриниговых программ для выявления заболевания

диабетической ретинопатии на ранних стадиях, может помочь в исследовании зависимости между степенью кривизны капилляров и гипертонической ретинопатией, зависимости диаметра капилляров от наличия гипертонии у пациентов. Более того сегментация капилляров в сетчатке может быть использована в качестве помощника при проведении лазерных микрохирургический операций на глаза. Рассказывая о древовидной системе капилляров в сетчатке глаза, нельзя не упомянуть об её уникальности для каждого индивидуума. Как следствие, сегментация капилляров может быть использования для биометрической идентификации личности человека.
На рисунке приведена фотография сетчатки глаза и выделенный в чёрно-белом цвете рисунок капилляров.



Source: https://drive.grand-challenge.org/DRIVE/

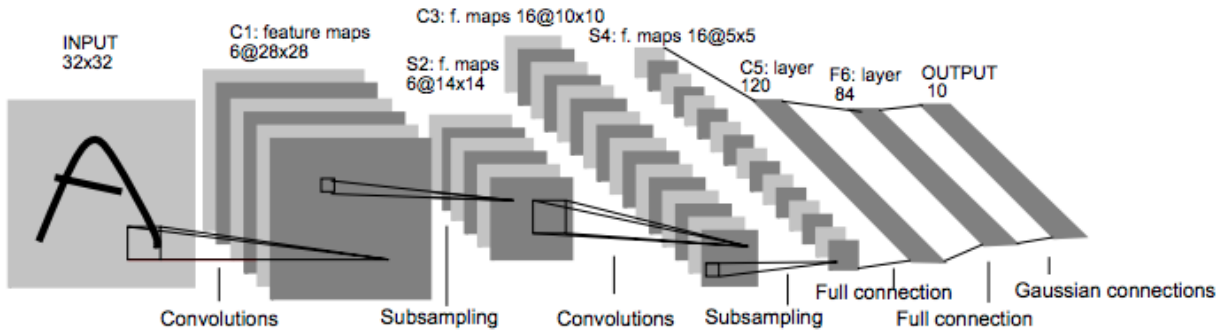
Обзор мирового опыта:

Задача сегментации изображений заключается в разбиение изображения на области, соответствующие различными объектами. Это необходимо для более глубокого, полного понимания изображения. С данной проблемой часто сталкиваются в робототехнике, медицине. Так, выявление инородного объекта на медицинском изображении способно предотвратить развитие серьёзного заболевания ещё на ранней стадии. Для сегментации изображений применяется не только подходы компьютерного зрения (Computer Vision), но и подходы связанные с Искусственными Нейронными Сетями о которых пойдёт разговор далее.
Выделяют три основных подхода используемые в Нейронных Сетях для :

- Patch-based approach для свёрточных нейронных сетей
- Fully Convolutional Networks (FCN) [1]
- Расширения FCN [6,7]

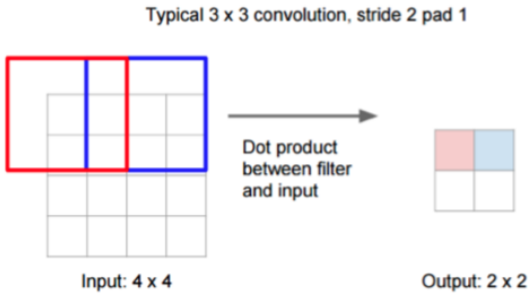
Так при использовании patch based подхода выбирается произвольный участок изображения называемый patch. Далее выбранный patch используется как входной слой для Свёрточной Нейронной сети. Пример

такой нейронной сети приведён ниже. Input 32x32 – часть исходного изображения размером 32 на 32 пикселя.

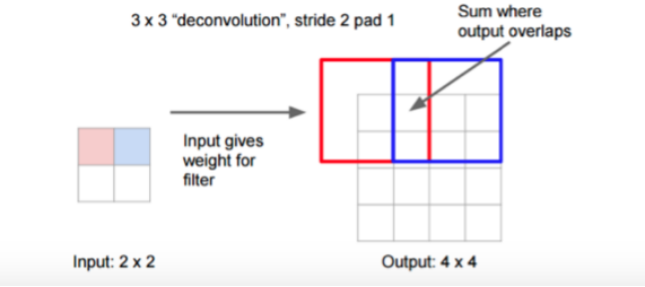


Source: Y. LeCun et al. "Gradient-Based Learning Applied to Document Recognition", Intelligent Signal Processing, 306-351, IEEE Press, 2001

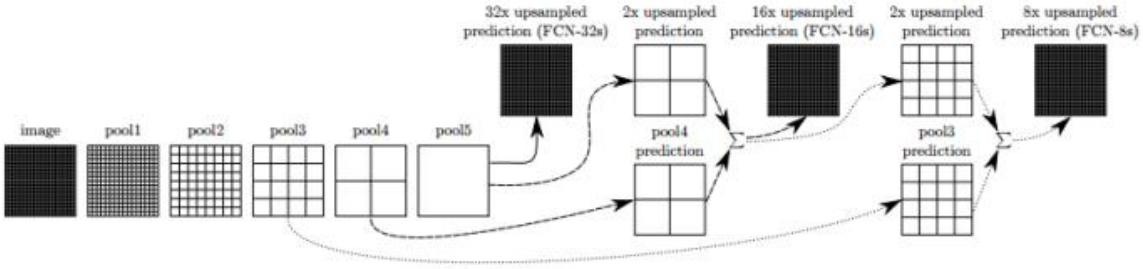
Fully convolutional networks подробно рассматриваются в работе [1], где подробно освещаются все преимущества и недостатки данного подхода. FCN – это так называемые полносвёрточные нейронные сети, которые подразумевают, что все их внутренние слои состоят исключительно из операции свёртки. К их характерным особенностям можно отнести получение сегментации изображения того же разрешения, что и исходное изображение. Очень часто для FCN используют так называемый transfer learning (адаптируют VGG, GoogLeNet и др.). Это актуальная научная проблема в области машинного обучения, заключающаяся в использовании накопленного опыта при решение одной задачи для решении совершенно другой задачи. Например, научившись классифицировать изображения машин на фотографиях используют накопленный опыт для классификации тракторов. Так, предобученную нейронную сеть под задачи классификации можно адаптировать для сегментации изображений. После встраивания слоёв можно произвести fine-tuning. К сожалению, основной проблемой таких манипуляций является низкое разрешение на выходном слое.



Для повышения разрешения (upsampling) часто используются как простые, так и обучаемые методы. Так convolution/ deconvolution layer показаны на рисунке справа. При этом, обучения на картинках более эффективно, чем обучение на patch данных, однако можно получить очень много patch с одного изображения, что является особенно эффективном при нехватке данных для обучения.



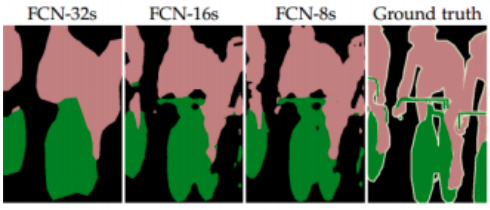
Это так называемые, базовые модели FCN, но существуют некоторые расширения идеи полносвёрточных нейронный сетей, в которых добавляются, например, skip connections. Смысл skip connections заключается в ансамблировании предсказаний по различным разрешениям. Схематически это может быть представлено так:



Source: G. Papandreou, L. C. Chen, K. P. Murphy and A. L. Yuille. Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1742-1750

На картинке справа приводятся сравнения результатов различных ансамблей с точки зрения метрик, а на картинке ниже – визуализация результатов.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**



Source: S. Zheng et al. Conditional Random Fields as Recurrent Neural Networks. 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1529-1537

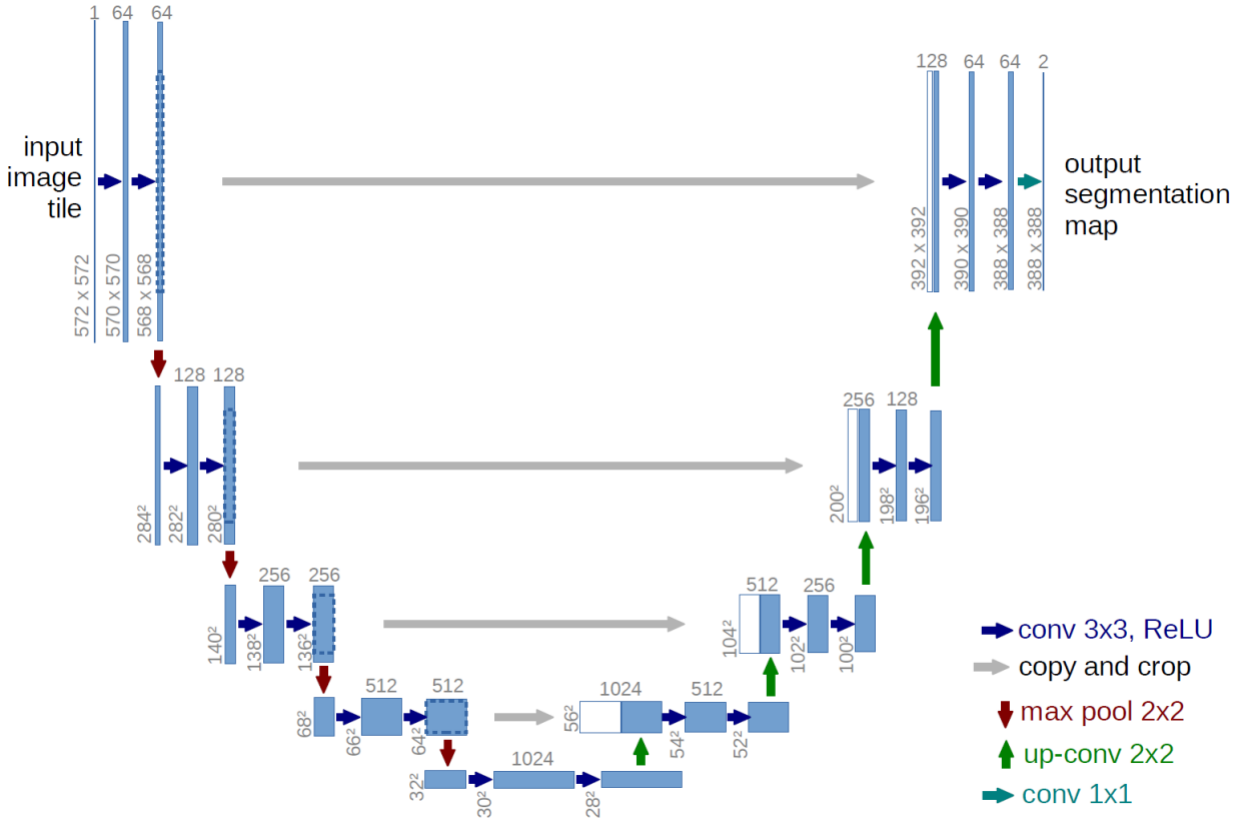
Более того, автору известны работы [6,7] в которых создаются более серьёзные расширения FCN, сети получили названия CRF-RNN и DeepLab соответственно. Хотя это модели показывают весьма высокие результаты, они не подходят для решения поставленной задачи в силу своей сложности реализации и существенном объёмом данных необходимых для обучения.

К сожалению, все приведённые выше алгоритмы страдают одним существенным недостатком, а именно, для обучения вышеперечисленных нейронных сетей требуется колоссальное количество данных. Многие тысячи фотографий для того, чтобы модель начала показывать достойные результаты. В этом и состоит главная сложность задачи связанной с

биомедицинской сегментацией изображений. Несмотря на то, что зачастую биомедицинские изображения проще, чем визуальные сцены, тем не менее исходных данных очень мало (десятки или сотни снимков).

U-Net

В 2015 году в университете Фрайбурга, Германия была создана глубокая свёрточная нейронная сеть (DCNN) специально для сегментации биомедицинских изображений, названная U-net [0]. В данной сети существенно используется идея skip connections.



Source: U-Net: Convolutional Networks for Biomedical Image Segmentation Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Сравнение сетей и выбор метода распознавания:

За основу для нашей сети мы планируем использовать свёрточную нейронную сеть такую как **U-Net** для решения задачи определения и сегментации рисунка капилляров в сетчатке глаза. Свёрточные нейронные сети (Convolutional neural network (CNN)) - это особая архитектура нейронной сети, нацеленная на эффективное распознавание изображений,

о который я рассказывал выше. Так, в работе [5] упоминается исследования сверхглубокой сети, в которой имеется более 1000 слоев. Большие успехи глубоких сетей в задачах классификации [3,5] и сегментации [2,4] широко известны и на научных изысканиях авторов этих работ строится наша идея решения. Основная идея CNN заключается в том, что при обучении CNN на основе исследуемых изображений одного класса, она самостоятельно формирует наборы признаков (карты признаков), которые в полной мере характеризуют этот самый класс, и в то же время, отделяют их от других изображений. Всё это реализуется в сети, с помощью так называемых свёрточных слоёв, которые составляют карты признаков изображения, и слоёв подвыборки, которые уменьшают масштаб изображения. Чередование данных слоёв позволяет составить карты признаков, с помощью которых изображения сегментируются. Данная технология была выбрана не случайно, так как исходя из вышеизложенного, U-net лучше всех подпадает под условия нашей задачи. Более того, нам известен ряд соревнований в которых выбранный метод хорошо себя зарекомендовал.

Так, U-Net показало лучший результат в соревновании Cell tracking Challenge 2015 [8]. В PhC-U373: было предоставлено 35 изображений для обучения. DIC-HeLa: было предоставлено 20 изображений для обучения.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

Source: U-Net: Convolutional Networks for Biomedical Image Segmentation Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Статистическая задача о необходимом размере данных для обучения

В процессе исследования и обучения нашей нейронной сети возник вопрос о размере данных необходимых для обучения CNN. Архитектура предложенной нейронной сети предусматривает обучение на случайной (участке) исходного изображения. Так как выбор этого участка осуществляется случайным образом, а нашей целью является покрыть такими участками всё исходное изображение, появляется логичный вопрос о среднем числе таких выборок (выборов участков изображения фиксированного размера) необходимых для покрытия всего исходного. Автору не известна ни одна научная публикация, решающая поставленную задачу по теории вероятности. Поэтому нижеизложенная задача и результаты в ней достигнутые имеют высокую степень актуальности в машинном обучении, так как дают оценку на необходимый размер данных для обучения в случае patch approach обучения, для любых свёрточных нейронных сетей. Для определения размера обучающей и валидационной

выборки для нашей с Леонидом Костюшко модели мы использовали мы использовали результаты полученные ниже мной.

Формулировка поставленной задачи

Дана $n \times n$ матрица (изображение) и есть подматрица(участок) $m \times m, \quad m < n$

1)Подматрица mxm прикладывается к матрице nxn случайным образом, не выходя за границы изображения и не нарушая клеточек (сетки матрицы), то есть матрицу mxm нельзя приложить под углом и т.д.

2)Место к которому была приложена матрица $m \times m$ сразу закрашивается

3)Последующее приложение матрицы $m \times \mathrm{m}$ продолжается так же случайным образом – она может быть приложена в то же самое место или как-то пересекать закрашенную область и полностью лежать в ней.
Вопрос: посчитать математическое ожидание T числа таких приложений подматрицы $m \times m$, чтобы $n \times n$ была целиком закрашена.

Решение:

Пусть T – интересующая нас величина – время до заполнения всей матрицы $n \times n$.

Теорема: при фиксированном m и $n \to \infty$

$$lim\ sup \frac{T}{2n^2 \ln n} \leq 1,\ lim\ inf \frac{T}{2n^2 \ln n} \geq \frac{1}{m^4},$$

где пределы рассматриваются по вероятности.

Доказательство:

Часть 1. Оценка сверху.

Рассмотрим "длинную" модель. В ней каждый раз, когда мы будем бросать квадрат, накрывающий несколько новых клеток, мы будем закрашивать только одну из них наугад. Ясно, что в длинной модели время T_1 до заполнения матрицы будет больше чем T.

Заметим, что $T_1 = 1 + T_{n^2-1,1} + \cdots + T_{1,1}$, где$T_{i,1}$ – время, когда количество клеток на доске ровно i. Заметим, что $T_{i,1}$ – время, пока матрицы,

бросаемые на доску с i не закрашенными клетками, попадают только на закрашенные клетки. Из всех $(n-m+1)^2$ возможных матриц минимум $\max(1, i-4m^2)$ матриц (при $i < n^2/2$, а иначе не больше $i - 4nm$ – что у клеток не лежащих в приграничной полосе таких матриц навалом легко доказывается) накрывает хотя бы одну из этих клеток. Значит, можно выбрать такое пространство на котором п.н.

$$T_{i,1} \le \widetilde{T_{i,1}},$$

где $\widetilde{T_{i,1}} \sim Geom\big((i-4m^2)/(n-m+1)^2\big)$при $i > 4m^2$,
$T_{i,1} \sim Geom(1/(n-m+1)^2)$при $i \le 4m$, – независимые величины.

Тогда $T \le T_1 \le \widetilde{T_1} = \sum_{i=1}^{n^2} \widetilde{T_{i,1}}$
Покажем, что

$$\frac{\widetilde{T_1}}{2n^2 \ln n} \to 1. \text{по вероятности}$$

Это легко сделать, например, по неравенству Маркова:

$$\boldsymbol{P}\left(\left|\widetilde{T_1} - 2n^2 \ln n\right| > 2\varepsilon n^2 \ln n\right) \le \frac{D\widetilde{T_1}}{4n^4 \ln^2 n} + \frac{\left(E\widetilde{T_1} - 2n^2 \ln n\right)^2}{4n^4 \ln^2 n} (1)$$

Но

$$E\widetilde{T_1} = 4m^2(n-m+1)^2 + \sum_{i=4m^2+1}^{n^2/2} \frac{(n-m+1)^2}{i-4m^2} + \sum_{i=n^2/2+1}^{n^2} \frac{(n-m+1)^2}{i-4nm}$$
$$= O(n^2) + n^2 \sum_{i=1}^{n^2/2-4m^2-1} \frac{1}{i} = 2n^2 \ln n + O(n^2).$$

При этом из независимости

$$D\widetilde{T_1} = 4m^2(n-m+1)^4 + \sum_{i=4m^2+1}^{n^2/2} \frac{(n-m+1)^4}{(i-4m^2)^2} + \sum_{i=n^2/2+1}^{n^2} \frac{(n-m+1)^4}{(i-4nm)^2} = O(n^4).$$

Отсюда правая часть (1) стремится к нулю. Оценки грубые, их можно улучшать, тогда, думаю, для всех $m = o(n)$ тоже нормально получится.

Часть 2. Оценка снизу.

Рассмотрим "короткую" модель. В ней каждый раз, когда мы будем бросать квадрат, накрывающий несколько новых клеток, мы будем заштриховывать m^2 клеток на нашей доске, включая те, которые были новые в этом квадрате (или все оставшиеся клетки на последнем ходу). Ясно, что в короткой модели время T_2 до заполнения матрицы меньше чем T .

Заметим, что $T_2 = 1 + T_{k,2} + \cdots + T_{1,2}$, где $k = [n^2/m^2] + 1$, $T_{i,1}$ – время, когда количество клеток на доске лежит в интервале $\left((i-1)m^2, im^2\right]$.
При этом из всех $(n-m+1)^2$ возможных матриц не более im^2 содержат хотя бы одну клетку из наших i , поэтому на некотором вероятностном пространстве $T_{i,2} \geq \widetilde{T_{i,2}}$, где $\widetilde{T_{i,2}} \sim Geom(im^4)$.

Опять же покажем, что $\widetilde{T_2} = \sum_{i=1}^{k} \widetilde{T_{i,2}}$ удовлетворяет требуемой сходимости. Тогда

$$P\left(\left|\widetilde{T_2} - 2m^{-4}n^2 \ln n\right| > 2\varepsilon n^2 \ln n\right) \leq \frac{D\widetilde{T_2}}{4n^4 \ln^2 n} + \frac{\left(E\widetilde{T_2} - 2n^2 m^{-4} \ln n\right)^2}{4n^4 \ln^2 n} \qquad (2)$$

$$E\widetilde{T_2} = \sum_{i=1}^{k} \frac{(n-m+1)^2}{im^4} = \frac{2n^2 \ln n}{m^4} + O(n^2),$$

$$D\widetilde{T_2} = \sum_{i=1}^{k} \frac{(n-m+1)^4}{i^2 m^8} = O(n^4).$$

Опять же из неравенства Маркова получится требуемая оценка.
Если действовать потоньше, то по неравенству Колмогорову и лемме Бореля-Кантелли, вероятно, можно и п.н. сходимость получить, да и явные оценки на отклонение T от искомой величины можно выудить.

QED

Результаты и новые перспективы

Обучение проходило 120 эпох и заняло около 3 суток. Финальное значение коэффициента dice – 0.8141 на тесте. С этим результатом удалось занять 45 место в мировом рейтинге.

В будущем планируется улучшить оценки на объём необходимых для обучения и валидации данных с использованием неравенства Колмогорова и лемме Бореля-Кантелли. Так же планируется изучить новые архитектуры,

основные идеи которых в улучшении блоков текущей сети: добавление рекуррентных зависимостей [6], что поможет сети лучше запоминать информацию, полученную на предыдущих шагах.

Ссылки на источники

Data: https://drive.grand-challenge.org/DRIVE/

[0]U-Net: Convolutional Networks for Biomedical Image Segmentation Olaf Ronneberger, Philipp Fischer, and Thomas Brox
https://arxiv.org/pdf/1505.04597.pdf
[1]Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. CVPR, 2015.
https://arxiv.org/pdf/1502.02734.pdf
[2] Sitthichok Chaichulee, Mauricio Villarroel, João Jorge, Carlos Arteta, Gabrielle Green, Kenny McCormick, Andrew Zisserman, Lionel Tarassenko, Multi-Task Convolutional Neural Network for Patient Detection and Skin Segmentation in Continuous Non-Contact Vital Sign Monitoring // Automatic Face & Gesture Recognition (FG 2017) 2017 12th IEEE International Conference on, 2017. pp. 266-272
[3] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D, Erhan D., Vanhoucke V., Rabinovich A. Going deeper with convolutions // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. P.1-9
[4] Jonathan Long, Evan Shelhamer, Trevor Darrell Fully convolutional networks for semantic segmentation // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015. pp. 3431-40
[5] K. He, X. Zhang, S. Ren, J. Sun Deep Residual Learning for Image Recognition// Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) — DC, USA: IEEE Computer Society Washington, 2016 — C. 2342—2354.
[6]S. Zheng et al. Conditional Random Fields as Recurrent Neural Networks. 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1529-1537
[7]G. Papandreou, L. C. Chen, K. P. Murphy and A. L. Yuille. Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1742-1750
[8]International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)
URL: http://biomedicalimaging.org/2015/program/isbi-challenges/

Аппендикс

Код программы:

U-Net

```
from torch import nn
import torch.nn.functional as F


class double_conv(nn.Module):
    def __init__(self, in_ch, out_ch):
        super(double_conv, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_ch, out_ch, 3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True),
         #   nn.Dropout(p=0.2),
            nn.Conv2d(out_ch, out_ch, 3, padding=1),
            nn.BatchNorm2d(out_ch),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        x = self.conv(x)
        return x


class inconv(nn.Module):
    def __init__(self, in_ch, out_ch):
        super(inconv, self).__init__()
        self.conv = double_conv(in_ch, out_ch)

    def forward(self, x):
        x = self.conv(x)
        return x


class down(nn.Module):
    def __init__(self, in_ch, out_ch):
        super(down, self).__init__()
        self.mpconv = nn.Sequential(
            nn.MaxPool2d(2),
            double_conv(in_ch, out_ch)
        )

    def forward(self, x):
        x = self.mpconv(x)
        return x


class up(nn.Module):
    def __init__(self, in_ch, out_ch, bilinear=False):
        super(up, self).__init__()
```

```
        #  would be a nice idea if the upsampling could be learned too,
        #  but my machine do not have enough memory to handle all those we
ights
        if bilinear:
            self.up = nn.Upsample(scale_factor=2, mode='bilinear', align_c
orners=True)
        else:
            self.up = nn.ConvTranspose2d(in_ch//2, in_ch//2, 2, stride=2)

        self.conv = double_conv(in_ch, out_ch)

    def forward(self, x1, x2):
        x1 = self.up(x1)


        diffY = x2.size()[2] - x1.size()[2]
        diffX = x2.size()[3] - x1.size()[3]

        x1 = t.nn.functional.pad(x1, (diffX // 2, diffX - diffX//2,
                        diffY // 2, diffY - diffY//2))

        # for padding issues, see
        # https://github.com/HaiyongJiang/U-Net-Pytorch-Unstructured-Buggy
/commit/0e854509c2cea854e247a9c615f175f76fbb2e3a
        # https://github.com/xiaopeng-liao/Pytorch-UNet/commit/8ebac70e633
bac59fc22bb5195e513d5832fb3bd

        x = t.cat([x2, x1], dim=1)
        x = self.conv(x)
        return x


class outconv(nn.Module):
    def __init__(self, in_ch, out_ch):
        super(outconv, self).__init__()
        self.conv = nn.Conv2d(in_ch, out_ch, 1)

    def forward(self, x):
        x = self.conv(x)
        return x


class UNet(nn.Module):
    def __init__(self, n_channels, n_classes):
        super(UNet, self).__init__()
        self.inc = inconv(n_channels, 64)
        self.down1 = down(64, 128)
        self.down2 = down(128, 256)
        self.down3 = down(256, 512)
        self.down4 = down(512, 512)
        self.up1 = up(1024, 256)
        self.up2 = up(512, 128)
        self.up3 = up(256, 64)
        self.up4 = up(128, 64)
```

```
        self.outc = outconv(64, n_classes)

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)

        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        x = self.outc(x)
        x = t.sigmoid(x)
        return x
```

Значения функции ошибок в процессе обучения модели в зависимости от номера эпохи: