ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 37 за 2022 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

И.А. Белозеров, В.А. Судаков

Исследование моделей машинного обучения для сегментации медицинских изображений

Рекомендуемая форма библиографической ссылки: Белозеров И.А., Судаков В.А. Исследование моделей машинного обучения для сегментации медицинских изображений // Препринты ИПМ им. М.В.Келдыша. 2022. № 37. 15 с. https://doi.org/10.20948/prepr-2022-37
https://library.keldysh.ru/preprint.asp?id=2022-37

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

И.А. Белозеров, В.А. Судаков

Исследование моделей машинного обучения для сегментации медицинских изображений

Москва — 2022

Белозеров И.А., Судаков В.А.

Исследование моделей машинного обучения для сегментации медицинских изображений

На примере рентгеновских снимков легких человека проводятся анализ и построение моделей семантической сегментации. В работе исследуются различные подходы к обработке медицинских изображений, сравнение способов реализации моделей глубинного обучения и их оценивания. Разработаны 5 моделей нейронных сетей для выполнения задачи сегментирования. Они реализованы с помощью библиотек TensorFlow и PyTorch. Модель с лучшими показателями может быть применена для построения системы автоматической сегментации различных снимков пациентов и расчета характеристик их органов.

Ключевые слова: сегментация, компьютерное зрение, глубинное обучение, нейронные сети, TensorFlow, PyTorch.

Ilya Andreevich Belozerov, Vladimir Anatolyevich Sudakov.

Investigation of machine learning models for medical image segmentation

On the example of X-ray images of human lungs, the analysis and construction of models of semantic segmentation of computer vision is carried out. The paper explores various approaches to medical image processing, comparing methods for implementing deep learning models and evaluating them. 5 models of neural networks have been developed to perform the segmentation task, implemented using such well-known libraries as: TensorFlow and PyTorch. The model with the best performance can be used to build a system for automatic segmentation of various images of patients and calculate the characteristics of their organs.

Key words: segmentation, computer vision, deep learning, neural networks, TensorFlow, PyTorch.

Работа выполнена при финансовой поддержке ФГБОУ ВО "РЭУ им. Г.В. Плеханова".

Введение

Современное компьютерное зрение охватывает огромный спектр различных прикладных задач, начиная с классификации изображений и видео нейронными сетями и заканчивая генерацией жестов, мимики и эмоций человека. Не менее важным блоком задач в обработке и анализе изображений является задача семантической сегментации. Для медицинской сферы очень важными аспектами являются определенное заключение врача и быстрое принятие соответствующих решений, так как от этого зависит здоровье пациента. Для оптимизации работы и помощи специалистам области медицины в формировании заключений разрабатываются разнообразные математические модели, в том числе алгоритмы сегментации. Результаты работы необходимы для вычисления необходимых доктору характеристик органов человека.

В данной работе исследуются подходы к решению задачи сегментации нейронными сетями, проводится тестирование различных архитектур, комбинаций слоев, функций потерь и метрик оценки качества. Цель работы – разработать модель для сегментации рентгеновских снимков легких человека на основе набора изображений и их масок сегментаций. Задача может быть спроецирована на обработку любых других наборов снимков: МРТ сердца, головного мозга и т.д, поэтому она актуальна и имеет широкий диапазон применений с некоторыми изменениями. В работе проанализированы современные методы решения вышеупомянутой задачи, а именно, реализация нейронной сети с архитектурой Unet и LinkNet, которые в большей степени используются для обработки и анализа медицинских изображений. Одной из основных проблем разработки оптимальной модели является поиск соответствующих данных, так как такая разновидность уникальна в своем роде. Другой не менее важной проблемой является предобработка такого вида набора – высокие требования к разрешениям снимков, их четкость необходимы при обучении модели для запоминания самых мельчайших паттернов. В качестве основного набора данных для анализа были взяты снимки легких человека с платформы Kaggle.com.

Как уже отмечалось выше, медицинские снимки имеют специфику большого количества пикселов, структуры органов и их форм; всевозможные необычные изгибы и другие паттерны (например форма легких, структур головного мозга). Основной трудностью в работе с изображениями, от которых требуется высокое разрешение и четкость картинки, является выбор метода загрузки данных в память облака и подачи их нейронной сети на вход, чтобы избежать потери данных. В таком случае можно прибегнуть к использованию некоторых техник: выгрузка снимков несколькими экземплярами (размерами по batch-size), случайное разбиение полного изображения на части меньшего размера (crop) или выгрузка всех экземпляров в облако, при достаточном количестве памяти и сравнительно небольшом количестве снимков в наборе с предварительной трансформацией размеров.

Существует множество других интересных подходов для построения модели сегментации каких-либо органов медицинских снимков: решение задач instance segmentation [1], где для каждого класса на изображении существует своя маска и своя рамка детектирования (bound box), GAN – способны восстанавливать рамку детектирования на объекте и могут также выступать аналогом для выделения и анализа необходимого органа на снимке.

Обработка и анализ исходных данных

Основные методы подготовки данных для сегментирования объектов схож с методами обработки для задач классификации. Во-первых, масштабирование значений пикселов изображений в диапазон (0, 1) посредством деления на 255, поскольку изначально значения находятся в шкале от 0 до 255, так как алгоритму будет легче осуществлять вычисление весов на данных из одного распределения; как вариант, можно отмасштабировать целевые переменные. Кроме того, следует выбрать необходимое число каналов при преобразовании картинки в тензор. Сам снимок можно сделать с тремя каналами или с одним, а вот маску, в нашем случае, необходимо преобразовать с 1 каналом, так как количество предсказываемых классов равно 2 (фон и сам объект). Во-вторых, трансформация изображения к необходимому размеру (256 x 256, 512 x 512), чаще всего выбор зависит от дальнейшей стратегии обучения и имеющихся ресурсов памяти. Как вариант, исходное изображение может быть разделено на отдельные части заданного размера (обучение по crop), что позволит не потерять данные при изменении ширины и высоты. В-третьих, применение различных техник аугментации, которые могут быть встроены непосредственно в библиотеки PyTorch или TensorFlow, или же из сторонних модулей – albumentations. Рассмотрим некоторые примеры из техник, которые были использованы при разработке:

- Горизонтальное вращение. С заданной вероятностью метод изменяет значения пикселов случайного изображения из набора так, что оно становится повернутым в относительно горизонтали. Существуют также аналогичные техники поворота картинки вертикально или на заданное число градусов.
- Случайное приближение. Исходные пикселы изображения преобразуются таким образом, что снимок становится визуально больше или меньше в зависимости от параметров.
- Случайное применение аффинного преобразования [2] к данным. Это метод позволяет изменить изображение различными линейными операциями (увеличение, уменьшение, поворот, интерполяция участка) посредством использования матричных операций.
- Добавление пикселов на изображение (есть сходства с padding в сверточных слоях нейронных сетей), если размер стороны меньше заданного значения.

- Точечное преобразование перспективы. Данные метод осуществляет проецирование изображения на новую плоскость просмотра, то есть может переводить изображение из формы трапеции (по 4 координатам) в прямоугольное или наоборот.

В данной работе были смоделированы 4 основных алгоритма сегментации и один, сочетающий в себе элементы из некоторых остальных. В каждом из вариантов были использованы те или иные подходы загрузки и предобработки входных данных. Рассмотрим их.

1. В самом первом подходе набор изображений и их масок в объеме 704 экземпляра был полностью загружен в облако, трансформирован к размеру (256, 256) с помощью библиотеки OpenCv так, что первоначальный входной тензор имел размерность (704, 256, 256, 1) для самих признаков и их целевых значений. Обращаем внимание, что сами снимки были представлены 1 каналом. Используя функцию flip библиотеки numpy, размер тензора увеличили в 2 раза по первой оси. Далее значение изображений и целевых переменных были отмасштабированы, также был выбран порог (0.5) для интерпретируемости и лучшего предсказания меток классов. Данные делились на mini batch-size размером 16. Обучение осуществлялось с помощью библиотеки TensorFlow.
2. В следующем варианте была использована загрузка и подача данных алгоритму с помощью batch-генератора, что позволяет эффективно использовать ресурсы ОЗУ. Изображение были представлено в виде тензора размерностью (batch-size, 256, 256, число каналов). Для самих снимков непосредственно было выбрано 3 канала, а для масок – 1 канал. Аугментация в этом случае не применялась, код был написан библиотеке PyTorch. Стоит отметить удобство выгрузки и подачи данных на вход пакетным способом, так как для этого нужно определить собственный класс, наследованный от базового класса Dataset и передать встроенному в библиотеку генератору mini-batch.
3. В последних образцах, отличающихся только типом архитектуры алгоритма, данные были предварительно разделены на тренировочную и валидационную выборки; созданы соответствующие классы на TensorFlow для выгрузки пакетным способом, и применена аугментация: случайное горизонтальное вращение, аффинное преобразование, добавление пикселов и преобразование перспективы, а также целое изображение было разделено на части размера (512 x 512) – crop и отмасштабировано соответственно. Число каналов у снимков равно 3.

Анализ архитектур и методов их реализации

Прежде чем переходить к обзору использованных архитектур, необходимо дать небольшое представление о способе решения задачи сегментации. Сама идея выходит непосредственно из задачи классификации изображения. Как мы помним, задачу классификации можно решать при помощи сверточных нейронных сетей, используя сверточные слои [3] со слоями пулинга и подавая

их на вход функции активации. Блоки такой архитектуры (Convolution – Activation – Pooling) с соответствующим понижением ширины и высоты изображения и увеличением числа каналов называются feature extractor. Для задач классификации этот блок подается на вход полносвязной части с линейными слоями, а затем слою с сигмоидной или softmax [4] функцией активации. В задаче сегментации feature extractor увеличивают в размерах, путем пропускания его через слои UpSampling или Transposed Convolution [5]. Результат подается на вход сигмоидной или softmax функции активации, чтобы получить карту вероятностей отнесения того или иного пиксела к определенному классу. Стоит отметить, что блоки, которые уменьшают изображения, называются Encoder, а блоки, увеличивающие его, соответственно, Decoder.

Существует несколько архитектур для решения задачи сегментации. Лучшей и наиболее популярной является Unet, принципиальная архитектура которой изображена на рисунке 1, и именно она предназначена для работы с различными медицинскими изображениями. Рассмотрим ее особенности.

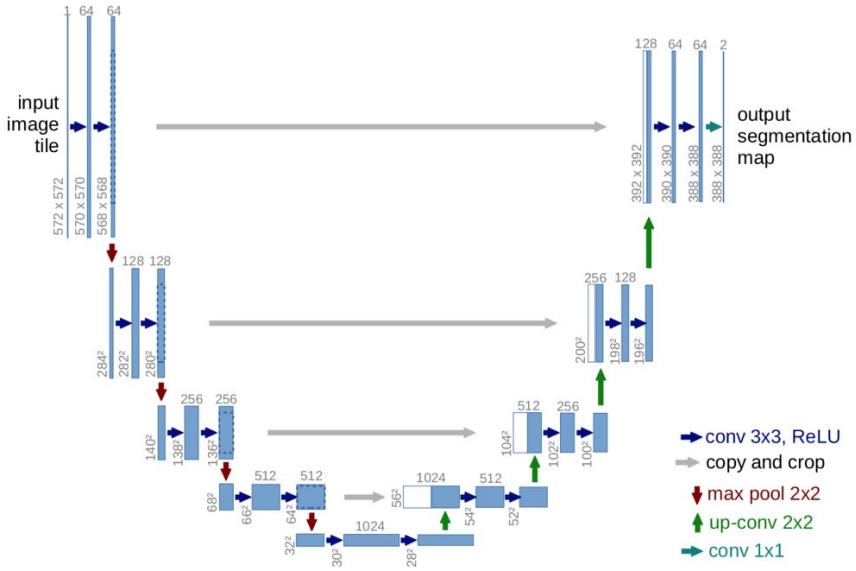

Рис.1. Архитектура Unet

Как можно заметить, что основное преимущество Unet в том, что мы используем такую технику, как skip connection, то есть верхние блоки слоев, содержащие низкоуровневые паттерны (линии, изгибы), мы соединяем с глубокими слоями той же размерности, но которые содержат высокоуровневую информацию о снимке (контуры, силуэты). Важным моментом является то, что увеличение размера путем Transposed Convolution осуществляется до применения блоков сверток к соединенным частям. Также возможно использование слоев BatchNorm и Dropout в комбинации со сверточными слоями, но не вместе.

Описав основную идею и архитектуру для решения задачи сегментации, перейдем к обзору разработанных моделей.

В самом первом варианте, написанном на TensorFlow, использовалось 4 блока энкодера со следующими слоями:

$$Convolution2d \rightarrow ReLU \rightarrow Dropout \rightarrow Convolution2d \rightarrow ReLU \rightarrow MaxPooling$$

Число каналов увеличивается следующим образом: 1 – 64 – 128 – 256 – 512. После добавляется блок из сверточных слоев с функцией активации ReLU и слой Dropout; итоговое число каналов равно 1024.

Далее по схеме Unet производится увеличение размеров карты энкодера и уменьшение числа каналов до первоначального значения. Важный момент: размерность входного тензора должна быть равна размерности итогового тензора.

В конечном итоге применяем сверточный слой с размером фильтров 1 x 1 и сигмоидную функцию активации. Обучение такой архитектуры заняло 1 час 11 минут на 10 эпохах.

В следующем подходе была использована библиотека PyTorch, и вся архитектура была реализована с помощью встроенных базовых классов. Структура и концепция сети одинаковые, отличия состоят в только в том, что для обработки данных перед входом использовался блок:

$$Convolution2d \rightarrow BatchNorm2d \rightarrow ReLU \rightarrow Convolution2d + Convolution2d \rightarrow BatchNorm2d$$

Также отсутствуют слои MaxPooling, и вместо Dropout используется BatchNormalization.

Циклы обучения, валидации были написаны вручную, а также функции подсчета необходимых метрик и функций потерь; был применен метод масштабирования шага сходимости (обучения) посредством контроля за метрикой качества. Сам процесс обучения длился чуть более 7 часов в течение 10 эпох.

Такой подход написания сети на более низком уровне абстракции позволяет осуществлять больше контроля за процессом обучения, а следовательно, улучшить результаты.

Последние два варианта отличаются лишь самой архитектурой сети: в одном использовалась Unet, а в другом LinkNet [6]. Идея этих подходов заключалась в использовании известной техники, применяемой часто на практике – transfer learning. В чем ее смысл? В пакетах Keras и PyTorch уже существуют предобученные сети на соревновании Imagenet. Таким образом, в качестве энкодера необходимо выбрать одну из существующих архитектур. Что же касается части декодера, то его можно написать самостоятельно исходя выбранной архитектуры, однако существуют библиотеки, в которых достаточно вызвать базовый класс (Unet или LinkNet) и передать название предобученной модели. В качестве такой сети была выбрана EfficientNetb3 [7]. Обучение продолжалось в течение 40 итераций, и суммарное время составило 3 часа 38 минут.

Метрики качества и функции потерь моделей

Самым важным аспектом при построении и обучении модели является выбор функции потерь [8], так как именно от нее зависит, насколько хорошо алгоритм оптимизирует веса. Не менее важным пунктом является выбор метрики качества, которая показывает, как хорошо модель решает задачу. Самым важным свойством метрики, пожалуй, я бы выделил ее интерпретируемость – функция должна быть понятна абсолютно любому человеку, в первую очередь заказчику. Следует упомянуть, что в большинстве случаев сначала выбирается метрика оценки, а затем сама функция потерь.

Рассмотрим функционалы качества и обучаемые функции, которые использовались в задаче.

В первом алгоритме в качестве функции потерь использовалась бинарная перекрестная энтропия:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \, log\big(p(y_i)\big) + (1 - y_i) log\big(1 - p(y_i)\big), \quad (1)$$

где $p(y_i)$ – предсказанная вероятность того, что пиксел относится к ground truth маске объекта.

Соответственно, метрика качества доля правильных ответов (accuracy):

$$accuracy = \frac{1}{m}\sum_{i=1}^{m} I[a_i = y_i], \quad (2)$$

где y_i – истинное значение i-го объекта, a_i – ответ алгоритма на данном объекте, m – число объектов.

Стоит отметить, что выбор accuracy в качестве метрики не совсем подходит для данной задачи, так как непонятно, какой числовой показатель охарактеризует задачу решенной достаточно хорошо. Кроме того, в нашей задаче присутствует дисбаланс классов – количество пикселов фоновой части больше, чем число пикселов маски.

Прежде чем охарактеризовать функционал качества второй модели, рассмотрим, с помощью каких метрик она сравнивалась:

$$IoU = \frac{Intersection}{Union}, \quad (3)$$

где intersection – количество тех пикселов, которые алгоритм предсказывает верно и относит к положительному классу – TP, а Union – FP + TP + FN.

В данной модели к знаменателю прибавляется параметр smooth, равный 10^{-8} для предотвращения деления на 0. Также в задаче было вычислено accuracy и dice.

$$Dice = \frac{2 * Intersection}{Union + Intersection}. \quad (4)$$

Эта метрика очень похожа на IoU, и в задачах лучше всего использовать что-нибудь одно, но в данном примере использовались два этих показателя. Также рассчитывается и показатель accuracy.

Рассмотрим функции потерь, которые использовались в этом алгоритме:

$$Loss = Jaccard_{loss} + dice_{loss} + bce, \quad (5)$$

$$Jaccard_{loss} = 1 - \frac{intersection}{Union + smooth}, \quad (6)$$

$$Dice_{loss} = 1 - \frac{2 * intersection}{Union + intersection + smooth}. \quad (7)$$

В варианте с использованием transfer learning оптимизировалась сумма бинарной перекрестной энтропии, jaccard_loss и dice_loss.

Что касается метрик, то основной упор был сделан на IoU и F1_score:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}, \quad (8)$$

$$Precision = \frac{TP}{TP + FP}, \quad (9)$$

$$Recall = \frac{TP}{TP + FN}. \quad (10)$$

Для более детального понимания происходящего рассмотрим графики зависимости функции потерь и метрики от количества эпох обучения для модели с архитектурой Unet и LinkNet.

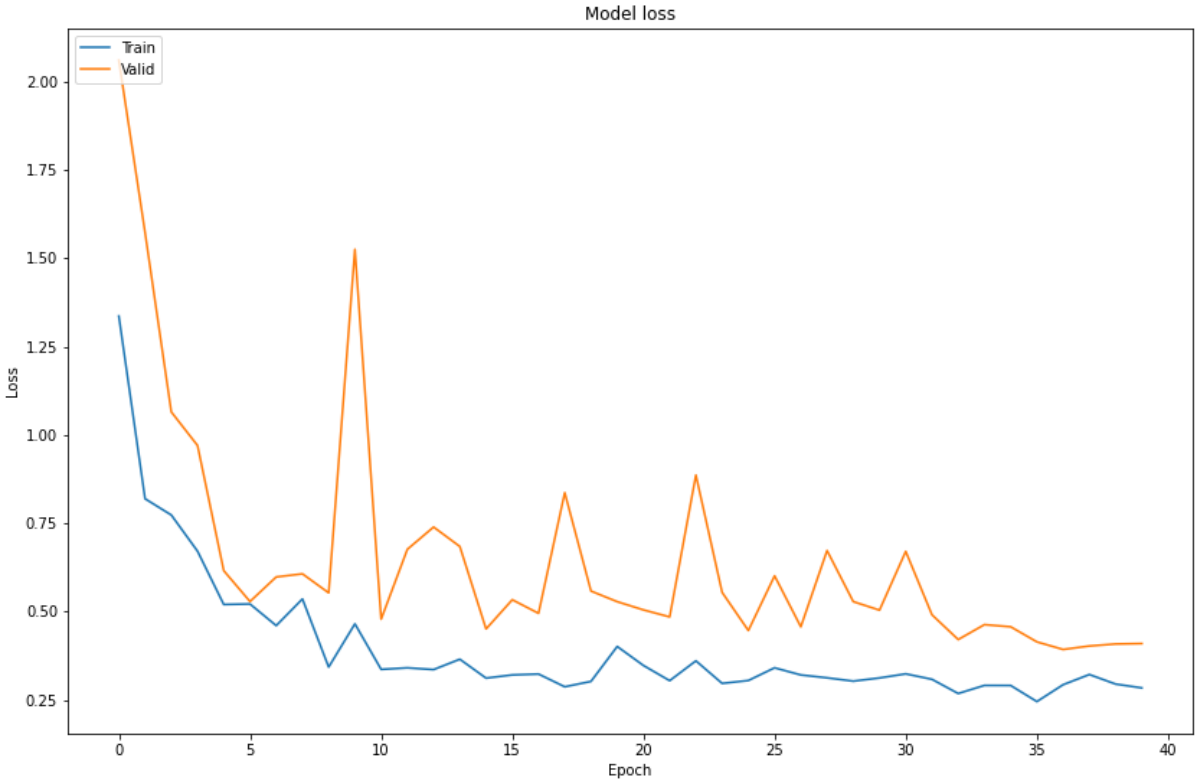

Рис. 2. Функция потерь, Unet

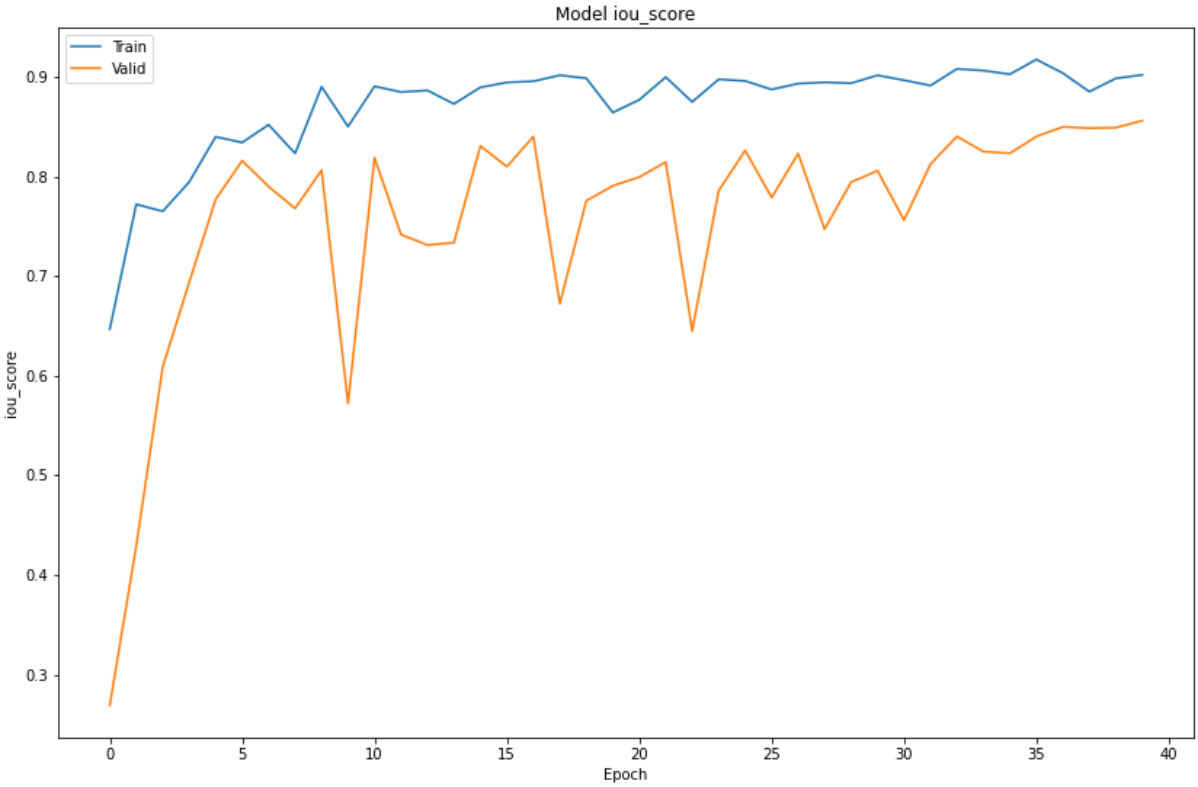

Рис. 3. IoU score, Unet

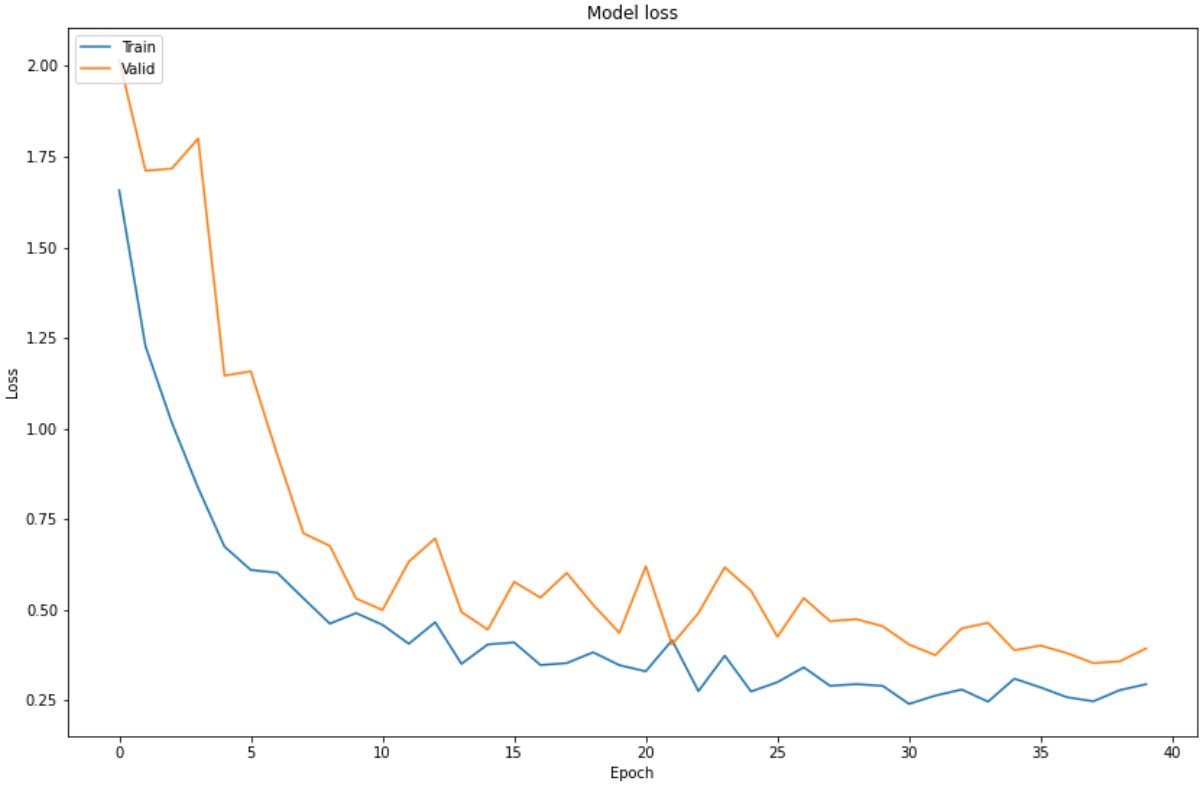

Рис. 4. Функция потерь LinkNet

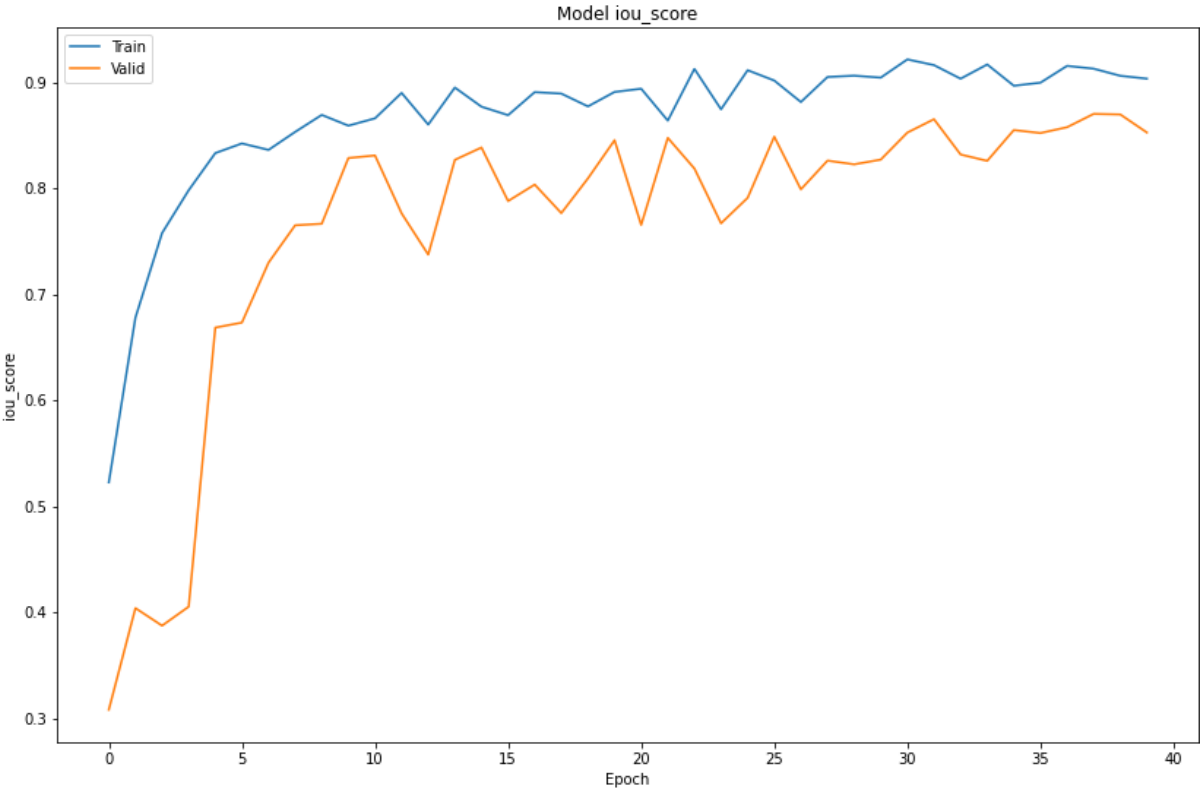

Рис. 5. IoU score, LinkNet

Как мы можем наблюдать на обоих графиках, изображенных на рисунках 2, 3, 4, присутствуют сильные колебания и скачки в сторону увеличения и уменьшения. Это связано с тем, что алгоритм обучается не на всей выборке, а лишь только на частях размеров batch_size.

Финальный вариант, результаты которого представлены на рисунке 5 , заключался в следующем: способ загрузки, подача данных, библиотека для вычислений, архитектура использовались такие же, как в первом варианте, метрику оценивания выбираем IoU и F1, функцию потерь – сумму бинарной перекрестной энтропии и jaccard loss.

Результаты работы моделей и их сравнение

В данной работе для написания кода алгоритмов сегментации в основном использовалась библиотека TensorFlow, в одном варианте использовались ресурсы PyTorch, для предварительной обработки данных – albumentations. В большинстве случаев обучения нейронных сетей используется фреймворк TensorFlow из-за своей скорости вычислений, основанной на графах. PyTorch же применяется в случаях, когда задача не является очевидной для решения. Также вычисления с использованием данной библиотеки проводятся достаточно долго. Следовательно, TensorFlow лучше подходит для решения таких задач, как в данной работе.

Как итог, приведем результаты работы моделей: оценки метрик, значения функций потерь и примеры работы на тестовых данных.

Таблица 1

Результаты работы

Модель	Название и значение метрики на обучающей выборке	Название и значение функции потерь на обучающей выборке	Название и значение метрики на проверочной выборке	Название и значение функции потерь на проверочной выборке	Время выполнения и число эпох
Unet №1	Accuracy 97.51%	BCE 0.0635	Accuracy 97.7%	BCE 0.0635	1 час 11 минут, 10
Unet №2 PyTorch	Accuracy 98.27% IoU 92.72% Dice 96.21%	BCE + Jaccard loss + Dice loss 0.1724	Accuracy 98.04% IoU 92.15% Dice 95.89%	BCE + Jaccard loss + Dice loss 0.1925	7 часов, 10

Модель	Название и значение метрики на обучающей выборке	Название и значение функции потерь на обучающей выборке	Название и значение метрики на проверочной выборке	Название и значение функции потерь на проверочной выборке	Время выполнения и число эпох
Unet №3 Transfer Learning	IoU 90.4% F1 94.78%	BCE + Jaccard loss + Dice loss 0.293	IoU 85.02% F1 91.77%	BCE + Jaccard loss + Dice loss 0.3927	3 часа 38 минут, 40
LinkNet Transfer Learning	IoU 91.3% F1 95.27%	BCE + Jaccard loss + Dice loss 0.2475	IoU 87.04% F1 93.05%	BCE + Jaccard loss + Dice loss 0.3528	3 часа 38 минут, 40
Unet №4	IoU 91.01% F1 95.29%	BCE + Jaccard loss 0.1504	IoU 88.94% F1 94.14%	BCE + Jaccard loss 0.1792	55 минут, 100

Исходя из данных таблицы заметим, что последняя модель показывает наилучшие результаты по всем параметрам на проверочной выборке, имея IoU=88.94%, F1=94.14% и значение функции потерь 0.1792. Поэтому для дальнейшей доработки и настройки алгоритма для решения других задач сегментации из области медицины следует взять за основу данный вариант.

В качестве демонстрации работы данной модели рассмотрим ее результаты работы, представленные на рисунках 6 и 7 на примере изображений, которых алгоритм никогда не видел.

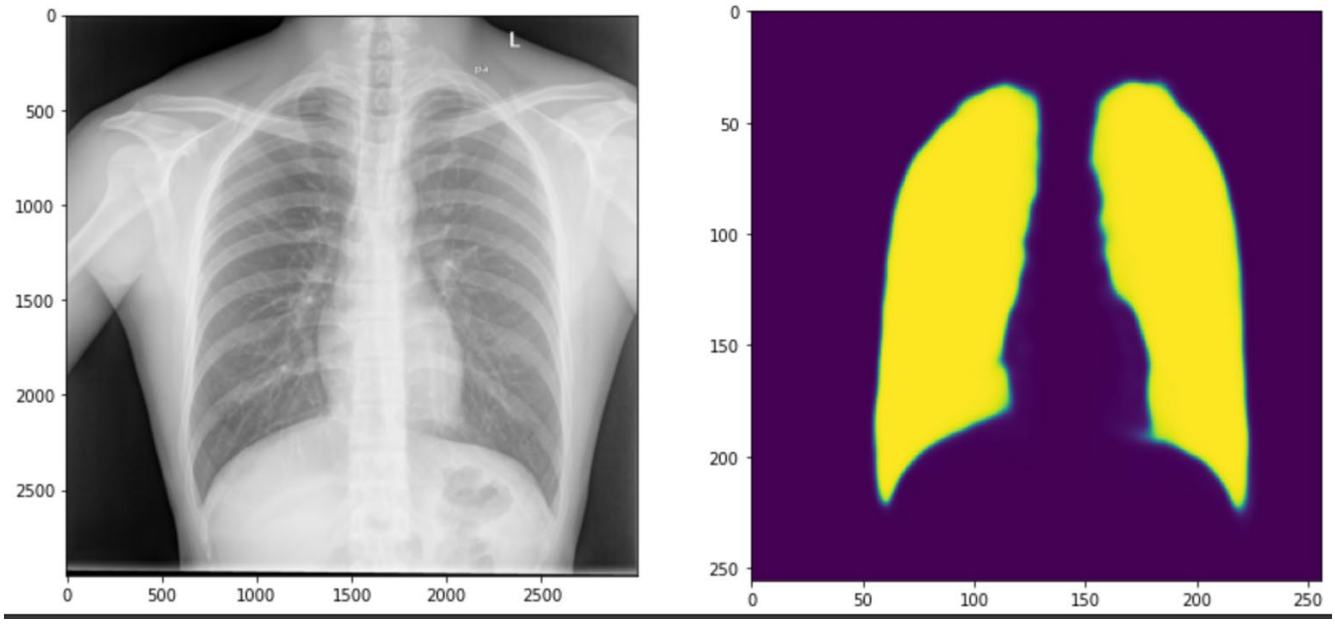

Рис. 6. Результат № 1 работы алгоритма на тестовой выборке

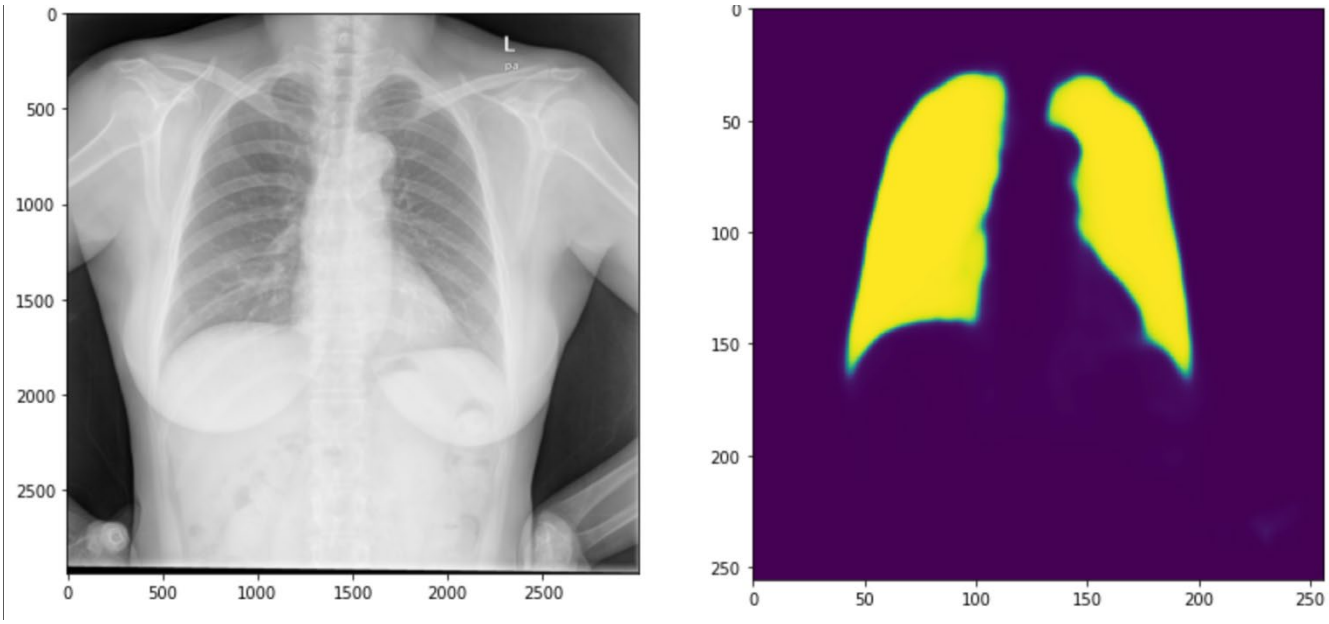

Рис. 7. Результат № 2 работы алгоритма на тестовой выборке

Заключение

Создана модель для сегментации медицинских изображений. Выявлены несколько важных моментов, на которые необходимо обращать внимание при исследованиях в данной области.

Во-первых, большинство изображений данного типа сделаны в тональностях серого цвета, поэтому проще всего преобразовывать снимок к одноканальной палитре. Как и для любых изображений, для лучшего обучения нейронной сети следует производить трансформацию, которая в общем случае зависит от задачи. Скажем, для работы со снимками клеток тела организма можно попробовать использовать гауссовский шум. Лучшим инструментом для построения данных нейронных сетей является TensorFlow благодаря скорости вычислений. Кроме того, фреймворк позволяет переходить на более низкий уровень обучения, где возможностей контроля процесса становится гораздо больше. Наилучшей архитектурой сети в данной задаче является Unet. И самыми важными моментами в разработке модели является выбор метрик качества и функции потерь. В данной работе было показано, как выбор того или иного функционала влияет на итоговый результат. Функции потерь, использованные в данной работе, не являются единственными для решения подобных задач. Существуют целые классы таких функций, которые включают большое количество параметров, рассчитанных эвристически. Принятие того или иного решения о выборе соответствующего функционала или использовании их всевозможных комбинаций зависит от целей проекта, сроков его выполнения и дальнейших действий.

Библиографический список

1. Iyer N. Low Memory Instance Segmentation. https://towardsdatascience.com/low-memory-instance-segmentation-b9c425b67db1

2. McQuistan A. Аффинные преобразования изображений в Python с помощью Numpy, Pillow и OpenCV. https://pythobyte.com/affine-image-transformations-in-python-with-numpy-pillow-and-opencv-ed73c238/

3. Шолле Ф. Глубокое обучение на Python. — СПб.: Питер, 2018. — 480 с.

4. Капаца Е. Софтмакс (Softmax). https://www.helenkapatsa.ru/softmaks/

5. Naoki. Up – sampling with Transposed Convolution. https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

6. Chaurasia A., Culurciello E. LinkNet. https://codeac29.github.io/projects/linknet/

7. Tan M., Le Q.V. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

8. Баданина Н.Д., Судаков В.А. Модели машинного обучения для классификации отзывов о банках // Препринты ИМП им. М.В. Келдыша. 2021, № 50. – С. 1 – 14.

Оглавление