

Оглавление

Введение	3
Постановка задачи	5
Задача классификации	5
Задача семантической сегментации	6
Исходные данные	7
Выбор метрики	9
Описание экспериментальной установки	12
Построение базового решения	13
Решение с использованием человеческих ресурсов	14
Исследование гипотез	15
Опорная гипотеза	19
Гипотеза о применимости алгоритма оптимизации Adam	23
Гипотеза об увеличении ядра свертки	27
Гипотеза об уменьшении свертки	31
Гипотеза об увеличении количества эпох при уменьшенном ядре свертки	33
Гипотеза о достижении заложенной асимптоты	35
Исследование гипотез задачи семантической сегментации	37
Первые результаты	37
Опорная гипотеза	39
Исследование модели на корректных данных	42
Исследование модели SegNet	44
Гипотеза об увеличении числа фильтров SegNet	46
Исследование модели FCN	48
Исследование алгоритма расчета метрики DICE в модели U – Net	50
Исследование модели на большем объеме данных	53
Заключение	56
Список литературы	59
Приложения	60

Введение

Морской грузопоток быстро растет. Большое количество судов увеличивает вероятность возникновения происшествий в море, таких как экологически разрушительные аварии, катастрофы на судах, пиратство, незаконный лов рыбы, незаконный оборот наркотиков и незаконные перевозки грузов. Это вынудило многие организации, от природоохранных учреждений до страховых компаний и национальных государственных органов, более внимательно следить за обстановкой в открытом море.

В результате чего была поставлена задача нахождения кораблей на спутниковых снимках. С задачей распознавания объектов на изображении прекрасно справляются свёрточные нейронные сети. Сверточная нейронная сеть - специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году¹ и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения. Использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток. Таким образом, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв и субдискретизирующих. Структура сети — однонаправленная, принципиально многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения

¹ Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998.

умножается на матрицу - ядро свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

Работа свёрточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное.

Для актуальности темы можно добавить, что данный тип нейронной сети сейчас активно используется компанией Google, тот же Google – переводчик, в котором присутствует функция перевода текста с изображения. Та же компания ввела в свой поисковый сервис функцию поиска изображений, аналогичных нашему. Ну и если перейти к более серьезным вещам, то сверточные сети используются для автопилотов в современных автомобилях, например, как в бортовом компьютере автомобилей компании Tesla.

Помимо абстрактных примеров, есть и конкретные приложения сверточных нейронных сетей в медицине. В статье ¹ применяется сверточная нейронная сеть для локализации пневмоторакса в легких на рентгеновских снимках, если он присутствует. Реализовано с помощью архитектуры нейронной сети U-Net², которая на данный момент стала стандартной для решения задач сегментации изображений. Аналогично можно использовать

U-Net и для обработки изображений для нахождения раковых опухолей, что и применяется в работе³.

² the 2st-unet for pneumothorax segmentation in chest x-rays using resnet34 as a backbone for u-net. arXiv:2009.02805v1[eess.IV] 6 Sep 2020

³ U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597v1 [cs.CV] 18 May 2015

⁴ Colorectal Cancer Segmentation using AtrousConvolution and Residual Enhanced UNet. arXiv:2103.09289v1 [eess.IV] 16 Mar 2021

Постановка задачи

Задача классификации

Как было описано выше, у человечества появилась потребность в отслеживании движений морских судов в море для тех или иных целей. Для успешного отслеживания судов было принято использовать спутниковые снимки, но ведь вряд ли найдется человек, который будет скрупулезно рассматривать спутниковые снимки в поисках корабля на изображении, а если даже и будет, то компания не станет тратить свои ресурсы для оплаты труда такого работника. Из этой ситуации существует только одно рациональное решение – доверить монотонную работу автоматике. Вычислительной машине подается изображение, а далее следует ответ, присутствует ли морское судно на снимке.

Формализуя, задача будет классификации выглядеть следующим образом. Дана выборка изображений $X = \{x_i | i \in \{1, 2, \dots, n\}\}$, которую мы разделим на обучающую подвыборку $X' = \{x'_i | i \in \{1, 2, \dots, m\}\}$ и тестовую подвыборку $X'' = \{x''_i | i \in \{1, 2, \dots, j\}\}$ так, что $X = X' \cup X''$ и $X' \cap X'' = \emptyset$. Так же мы делим множество правильных ответов $Y = \{y_i | i \in \{1, 2, \dots, n\}\}$ на Y' и Y'' так, что $Y = Y' \cup Y''$ и $Y' \cap Y'' = \emptyset$. Итак, есть выборка изображений X и выборка правильных ответов Y . Пусть $\xi: \Omega \rightarrow x$ – случайная величина, представляющая собой случайное изображение из X . И пусть $\eta: \Omega \rightarrow y$ – случайная величина, представляющая собой случайный правильный ответ из Y . Тогда определим случайную величину $(\xi, \eta): \Omega \rightarrow (X, Y)$ с распределением $p(x|y)$, которое является совместным распределением объектов и их классов. Тогда размеченная выборка – это элементы из распределения $(x_i, y_i) \sim p(x|y)$. Определим, что все элементы независимо и одинаково распределены. Тогда задача классификации будет сведена

К задаче нахождения $p(x|y)$ и заданном наборе элементов $D = \{(x_i, y_i) \sim p(x|y), i = \overline{1, N}\}$.

С помощью обучающей выборки X' и правильных ответов Y' будем находить распределение $p(x|y)$, а уже на тестовой выборке X'' и наборе правильных ответов Y'' для нее, будем смотреть, как хорошо сверточная нейронная сеть может распознавать тестовые изображения, которые никогда не видела, натренированная на обучающей выборки.

Задача семантической сегментации

Казалось бы, автоматизированная система для нахождения изображений с кораблями – это уже полезное нововведение. Осталось прикрепить к изображениям географические метки и уже будет получена окрестность, внутри которой будет находиться морское судно. Но разве нет способа получить точные координаты? Конечно есть, но для этого нужно определить точное расположение корабля на изображении.

Данная задача является задачей семантической сегментации изображения. Суть задачи в нашем случае в том, что нужно выделить одним цветом каждый пиксель изображения, где находится корабль и другим цветом выделить каждый пиксель того, что не является кораблем. Формальная постановка задачи будет аналогичная с постановкой нашей задачи для классификации, только лишь с тем различием, что каждый элемент $y_i \in Y$ не будет правильным ответом, есть ли корабль на изображении $x_i \in X$, а будет являться матрицей, содержащая в себе правильный ответ для каждого пикселя элемента x_i .

Исходные данные

Исходные данные представляет из себя набор изображений $w \times h$, где $w = 768, h = 768$ пример на рисунке 1.



Рисунок 1

Далее предоставляется таблица, содержащее имя изображения и перечисляются пиксели, где находится корабль, если он есть на изображении, пример на рисунке 2.

▲ ImageId	▲ EncodedPi...
00003e153.jpg	
0001124c7.jpg	
000155de5.jpg	264661 17 265429 33 266197 33 266965 33 267733 33 268501 33 269269 33 270037 33 270805 33 271573 33 ...
000194a2d.jpg	360486 1 361252 4 362019 5 362785 8 363552 10 364321 10 365090 9 365858 10 366627 10 367396 9 368165...
000194a2d.jpg	51834 9 52602 9 53370 9 54138 9 54906 9 55674 7 56442 7 57210 7 57978 7 58746 7 59514 7 60282 7 6105...
000194a2d.jpg	198320 10 199088 10 199856 10 200624 10 201392 10 202160 10 202928 10 203696 10 204464 10 205232 10 ...

Рисунок 2

На основе этих данных будет производится обучение сверточной нейронной сети. Перед обучением производится операция перехода от

изображения к его матричной форме. Пример матричной формы первого столбца для одного изображения в таблице 1.

(w = 0, i)	Red	Green	Blue
i = 0	126	141	146
i = 1	126	141	146
i = 2	126	141	146
.....			
i = h - 3	117	134	138
i = h - 2	115	133	137
i = h - 1	116	134	138

Табл.1 – пример матричного представления изображения для обучения модели.

Для бинарной классификации, столбец пикселей корабля преобразуется в вектор – строку размерности два, пример для одного изображения на рисунке 3.

[0., 1.]

Рисунок 3

Для семантической сегментации столбец пикселей корабля заменяется на полную карту изображения, хранящая бинарное значение для каждого пикселя, являющееся признаком принадлежности к первому или второму классу. Пример карты для одного изображения на рисунке 4.

```
[0][0][0][0][0][0][0][0][0][0][0]...
[0][0][0][0][0][0][0][0][0][0][0]...
[0][0][0][0][0][0][0][0][0][0][0]...
[0][0][0][0][0][0][0][0][0][0][0]...
[0][0][0][0][0][0][0][0][0][0][0]...
[0][0][0][0][0][0][0][0][0][0][0]...
[0][0][0][0][0][0][1][1][1][0][0]...
[0][0][0][0][0][0][1][1][1][1][0]...
[0][0][0][0][0][0][1][1][1][1][1]...
[0][0][0][0][0][0][1][1][1][1][1]...
[0][0][0][0][0][0][0][1][1][1][1]...
...
```

Рисунок 4

Выбор метрики

Итак, определив наши задачи нужно понять, как же оценивать полученные результаты. Наиболее интуитивной для задачи бинарной классификации является такая метрика точность (accuracy). Чтобы определить ее и последующие метрики, введем карту обозначений, приведенную в таблице 1.

		Прогноз	
		+	-
Правильный ответ	+	TP (True – positives)	FN (False – negatives)
	-	FP (False – positives)	TN (True – negatives)

Табл.2 – карта обозначений.

Определим обозначения, в нашем случае, как TP (True-positives) – число раз, когда корабль действительно опознан на изображении, FP (False – positives) – когда корабль ошибочно опознан и FN (False – negatives) – когда корабль присутствие корабля ошибочно отрицается.

Тогда точность можно определить, как:

$$accuracy = \frac{TP+TN}{TP+FP+FN+TN}.$$

Как можно заметить по выражению точности, данная метрика не позволит корректно оценить качество прогноза, если в выборка есть перевес классов ту или иную сторону. Например, если выборка из 1000 элементов будет иметь 990 экземпляров с отрицательным ответом и 10 с положительным, то данная метрика не позволит оценить, как хорошо мы прогнозируем истинно положительный ответ, ведь точность будет представлять собой долю правильных ответов в 99%. Получается, что доля положительных ответов не представляет из себя никакой информации о качестве прогнозирования. Однако, если есть возможность сбалансировать выборку, то данная метрика будет очень информативной.

Но все же, если нет возможности сбалансировать выборку, лучше перейти к другим метрикам, таким как точность (precision) и полнота (recall):

$$precision = \frac{TP}{TP+FP},$$
$$recall = \frac{TP}{TP+FN}.$$

Точность (precision) показывает, какая доля положительных прогнозов является истинно положительными. Полнота (recall) же показывает, какая доля положительных ответов была определена.

На основании двух вышеописанных метрик можно построить график зависимости точности (precision) от полноты (recall), называемый PR – кривой. Данный график будет давать представление о качестве прогнозов и для несбалансированной выборки. Далее находим площадь под этой кривой и получаем количественное значение, представляющее собой меру качества прогнозов в сравнении с истинными ответами.

Далее нужно определить метрику для задачи семантической сегментации. Так как нам априори известно, что на изображениях будет преимущественно преобладать площадь морской глади над площадью морского судна, то требуется мера, которая будет противостоять данному, достаточно внушительному, перевесу. С данной проблемой отлично справится метрика, называемая коэффициентом Соренсена – Дайса (DICE коэффициент). Определяется она как:

$$DICE = \frac{2|A \cap B|}{|A \cup B|},$$

где А – прогнозируемое множество, а В – множество истинных ответов. Но для нашего, бинарного, случая метрика принимает более удобный вид:

$$DICE = \frac{2TP}{2TP+FN+FP}.$$

Данная метрика демонстрирует успешную борьбу с преобладанием одного класса над другим на одном изображении, что доказывается в очередной медицинской статье по семантической сегментации⁴.

⁵ V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. arXiv:1606.04797v1 [cs.CV] 15 Jun 2016

Описание экспериментальной установки

Сверточная нейронная сеть реализована на высокоуровневом языке программирования Python, с использованием встроенных библиотек: keras, scipy, numpy, tensorflow. Данные программные библиотеки позволяют строить нейронные сети, проверять те или иные гипотезы, не затрачивая время на программную реализацию самого механизма проектирования модели и на реализацию обучения модели. Запуск программы совершен на бесплатной облачной платформе Google Colab, которая позволяет совершать неподъемные для домашней машины вычисления.

Несмотря на преимущества облачных вычислений, бесплатная платформа имеет ограничения по объему файлового хранилища и времени работы программы. Время работы не должно превышать восьми часов, а объем файлового пространства не больше 15 Гб. Итак, фактор времени является наиболее весомым, так как время работы программы тратится не только на обучение модели сверточной нейронной сети, но и на предобработку данных. Поэтому решено задать количество изображений для исследования гипотез равное 20000, из которых 16000 непосредственно для обучения модели, а 4000 для проверки качества прогнозов на изображениях, которые модель никогда не обрабатывала. Выборка данного объема сбалансирована для бинарного классификатора и все элементы подобраны независимо друг от друга, что дает уверенность, что при исследовании, результаты для тех или иных гипотез будут релевантны относительно друг друга.

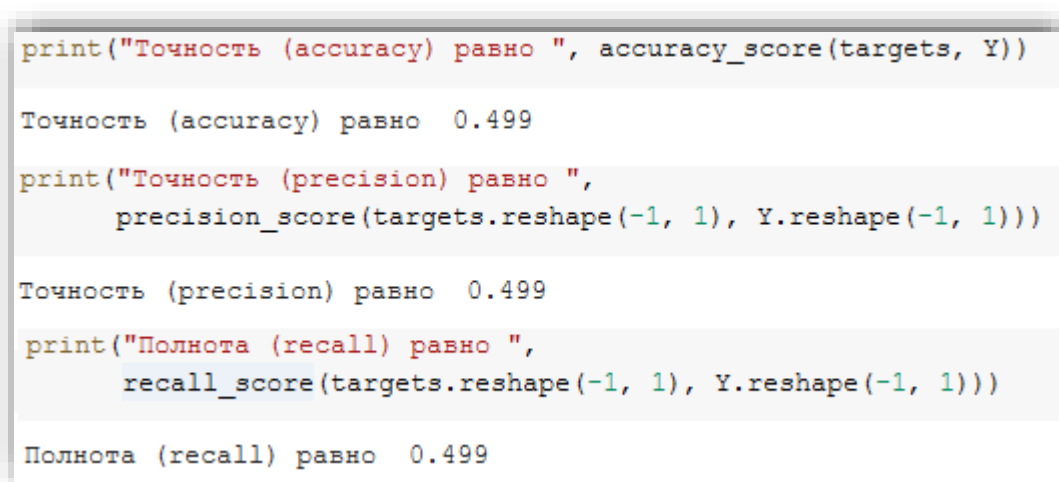
Обучение модели будет проходить на протяжении 50 эпох с объемом пакета одновременной обработки в 256 изображений.

Построение базового решения

Для исследования результатов работы сверточной нейронной сети требуется определить асимптоты. Так как для решения задачи семантической сегментации морских судов нужно полностью освоить упрощенную задачу в лице бинарной классификации изображений, следует задать граничное значение метрик, результаты гипотез ниже которых сразу будут отклонены.

Для задания минимального значения метрик будем использовать генератор случайного подбрасывания монеты, где результатом будет равновероятный исход орел или решка, интерпретируемые как нуль и единица. Проведем сравнение результатов генерации и истинных ответов на прежде описанной выборке в 20000 изображений и найдем для этого такие метрики, как: точность (accuracy), точность (precision), полнота (recall).

Результаты эксперимента приведены на рисунке 5.



```
print("Точность (accuracy) равно ", accuracy_score(targets, Y))

Точность (accuracy) равно  0.499

print("Точность (precision) равно ",
      precision_score(targets.reshape(-1, 1), Y.reshape(-1, 1)))

Точность (precision) равно  0.499

print("Полнота (recall) равно ",
      recall_score(targets.reshape(-1, 1), Y.reshape(-1, 1)))

Полнота (recall) равно  0.499
```

Рисунок 5

Итак, если округлить, то нижний порог для результатов гипотез составляет 0.5 для точности (accuracy), 0.5 для точности (precision), 0.5 для полноты (recall). Гипотезы с результатами ниже данного порога отклоняются.

Решение с использованием человеческих ресурсов

Для обоснования использования сверточной нейронной сети на практике нужно проверить, будет ли это решение более оптимальным, нежели человек сам будет помечать изображения с кораблями.

Для нахождения метрик качества определения класса изображения человеком создадим некий тестовый стенд. Для этого нам понадобится стационарный компьютер, клавиатура, монитор и программа, реализующая вывод изображения на экран и ввод ответа в бинарном виде.

Проведя данное исследование с использованием 100 изображений, за 3 минуты были получены ответы и следующие метрики качества ответов в сравнении с истинными. Результаты на рисунке 6.

```
print("Точность (accuracy) равно ", accuracy_score(Y_true, Y))  
  
Точность (accuracy) равно 0.78  
print("Точность (precision) равно ",  
      precision_score(Y_true.reshape(-1, 1), Y.reshape(-1, 1)))  
  
Точность (precision) равно 0.78  
print("Полнота (recall) равно ",  
      recall_score(Y_true.reshape(-1, 1), Y.reshape(-1, 1)))  
  
Полнота (recall) равно 0.78
```

Рисунок 6

Итак, метрики качества ответов человека на 100 изображениях за 3 минуты составляет 0.78 для точности (accuracy), 0.78 для точности (precision), 0.78 для полноты (recall). Гипотезы с результатами выше данного порога будут считаться оптимальными.

Исследование гипотез

Для исследований мы не будем составлять свою архитектуру сети, а позаимствуем уже готовую, под названием «VGG16».

«VGG16» — модель сверточной нейронной сети, предложенная К. Simonyan и А. Zisserman из Оксфордского университета в статье “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Модель достигает точности 92.7% в задаче распознавания объектов на 14 миллионов изображений, принадлежащих к 1000 классам. Для нашей модели мы сократим размеры входных данных из-за технических ограничений и для решения нашей задачи бинарной классификации мы сократим количество классов с тысячи до двух классов. Структура данной модели изображена на рисунке 7:

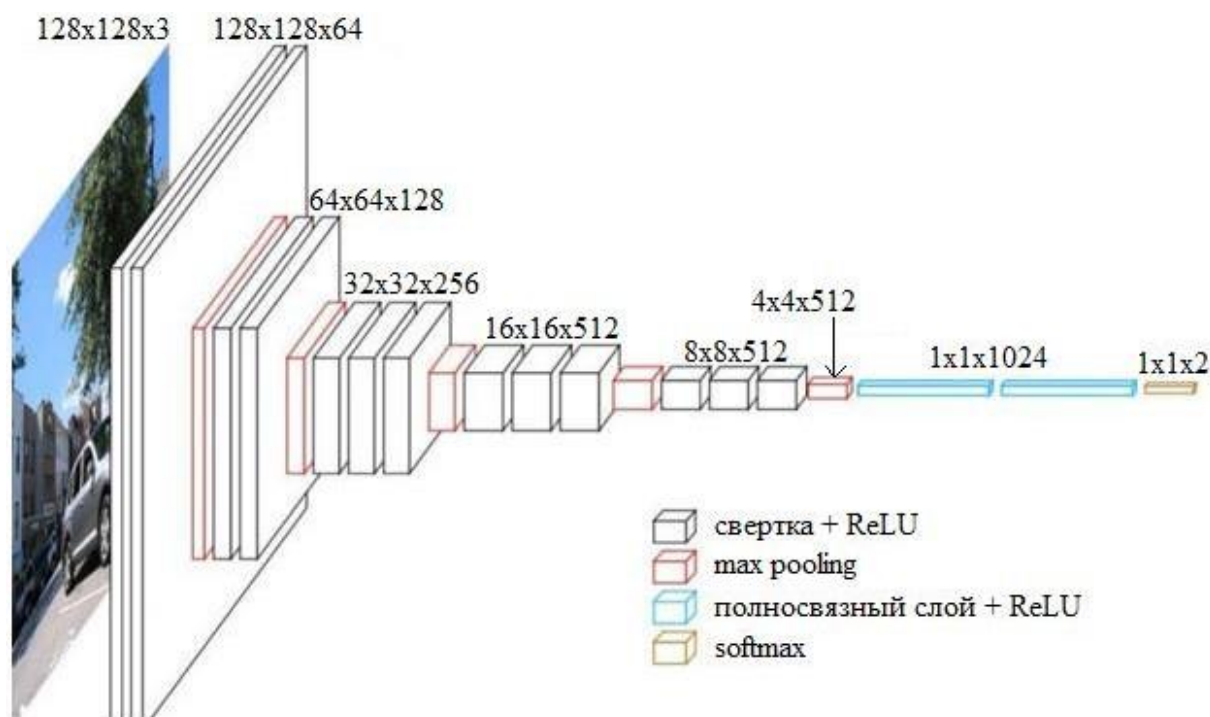


Рисунок 7

Так как используется изображение в формате Red-Green-Blue, каждый из трех каналов цвета обрабатывается отдельно, то сеть принимает три двумерной матрицы 128x128 с интенсивностями цвета. Если обратить внимание на легенду структуры на рисунке 7,

можно заметить, что помимо обозначения самого слоя мы также обозначаем и функцию активации, определенную на слое.

Начнем с функции активации ReLU (англ. Rectified linear unit) или иначе линейный выпрямитель. Функция определяется как

$$f(x) = \begin{cases} 0, & \text{при } x < 0 \\ x, & \text{при } x \geq 0 \end{cases}$$

и имеет график, изображенный на рисунке 8:



Рисунок 8

Сущность функции предельно понятна, если функция активации принимает отрицательное значение, она возвращает ноль и данное значение не искажает данные для дальнейшей обработки, если иначе, то значение следует в дальнейшие слои. Преимущество этой функции активации в том, что вычисление ReLU реализовано с помощью простого порогового преобразования матрицы активаций в нуле. Также к преимуществу можно отнести то, что ReLU не подвержен насыщению. Применение ReLU существенно повышает скорость сходимости стохастического градиентного спуска по сравнению с гиперболическим тангенсом. Считается, что это обусловлено линейным характером и отсутствием насыщения данной функции. К сожалению, ReLU не всегда достаточно надежны и в процессе обучения могут выходить из строя («умирать»).

Слишком большой градиент, проходящий через ReLU, приводит к такому обновлению весов, что данный нейрон никогда больше не активируется. Если это произойдет, то, начиная с данного момента, градиент, проходящий через этот нейрон, всегда будет равен нулю. Соответственно, данный нейрон будет необратимо выведен из строя. Например, при слишком большой скорости обучения может оказаться, что до 40% нейронов с ReLU «мертвы» или иначе говоря, больше не когда не активируются. Эта проблема решается посредством подбора приемлемой скорости обучения.

Далее разберем функцию активации softmax. Softmax – это обобщение логистической функции для многомерного случая. Функция преобразует z – вектор K – размерности в σ – вектор той же размерности, где каждая σ_i – координата полученного вектора представлена вещественным числом в интервале $[0,1]$ и сумма координат равна 1.

Координаты σ_i вычисляются следующим образом:

$$\sigma_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, \text{ где}$$

$$z = w^T x - \theta, \text{ где}$$

x – вектор – столбец признаков объекта размерности $M \times 1$, а M – количество признаков объектов. w^T – транспонированная матрица весовых коэффициентов признаков, имеющая размерность $K \times M$, а θ – вектор – столбец с пороговыми значениями размерностью $K \times 1$, где K – количество классов объектов, в нашем случае $K = 2$.

Рассмотрим пример работы функции softmax на выходном слое сети для нашей задачи. Пусть $i = 1$ – отсутствие корабля на изображении, а $i = 2$ – присутствие. Допустим, что нашли вектора z по формуле

$$z = w^T x - \theta$$

вида

$$z = \begin{pmatrix} -1 \\ 4 \end{pmatrix}.$$

Тогда имеем

$$e^z = \begin{pmatrix} e^{-1} \\ e^4 \end{pmatrix} = \begin{pmatrix} 0.36 \\ 54.98 \end{pmatrix}$$

и получаем

$$\sigma = \begin{pmatrix} 0.007 \\ 0.993 \end{pmatrix},$$

выбирая максимальное значение получаем, что с вероятностью в 99% корабль присутствует на фотографии, то есть результатом обработки изображения сетью будет 1 – корабль присутствует на фотографии.

Теперь перейдем к функции потерь. Так как на выходном слое используется softmax, для определения потерь с ней используют функцию перекрестной энтропии, которая имеет вид:

$$L(\sigma, y) = - \sum_i^K y_i \ln(\sigma_i).$$

Свойства данной функции:

1. $L(\sigma, y) \geq 0$
2. Минимум функции достигается при $\sigma_i = y_i$

Так как мы будем обучать нашу сеть не по одной фотографии за раз, а сразу на нескольких, то функция потерь для всей выборки находится как:

$$L = - \frac{1}{|V|} \sum_{(x,y) \in V} \sum_i^K y_i \ln(\sigma_i).$$

Опорная гипотеза

Так как задача машинного обучения – это прежде всего задача оптимизации, то и результаты обучения во многом зависят от методов нахождения оптимального решения.

За опорное решение возьмем результаты прошлых исследований, где использовался метод оптимизации RMSProp.

RMSProp (от англ. Root Mean Square Propagation) среднеквадратичное распространение — это метод, в котором скорость обучения настраивается для каждого параметра. Идея заключается в делении скорости обучения для весов на сгруппированные средние значения градиентов для этого веса. Таким образом, первое сгруппированное среднее вычисляется в терминах среднеквадратичного. Как и все градиентные методы, этот имеет общий вид:

$$w = w + h \odot \Delta w,$$

где w – матрица весов нейронной сети, а Δw – направление для минимизации целевой функции, а h – шаг или в нашем случае это скорость обучения. Вся суть метода в том, как мы находим наше направление.

Возьмем нашу скорость h и скорость затухания ρ (параметр сглаживания для экстраполяции). Определим наше начальное значение весов w как нули. И определим малую константу $\varepsilon = 10^{-8}$ чтобы избежать деления на нуль в будущем. Задаем параметр для агрегирования градиента $r = 0$. И пока условие остановки не выполнено, повторяем следующие шаги:

1. Выбираем часть экземпляров количества m

$$\{x^1, \dots, x^m\}$$

из нашей выборки и соответствующие им y^i

2. Находим градиент

$$g = \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial w} L(f(x^i, w), y^i),$$

Где $L(f, y)$ – целевая функция, а $f(x)$ – функция активации.

3. Агрегируем квадраты градиента

$$r = r\rho + (1 - \rho)g \odot g,$$

где $g \odot g$ – это поэлементное произведение матриц градиента.

4. Вычисляем направление

$$\Delta w = -\frac{1}{\sqrt{(\epsilon+r)}} \odot g,$$

где операция $\frac{1}{\sqrt{(\epsilon+r)}}$ применяется к каждому элементу матрицы.

5. Находим новое значение

$$w = w + h \odot \Delta w$$

Условием остановки алгоритма может быть оптимальное значение целевой функции или ситуация, когда веса практически не меняются.

Эмпирически показано, что RMSprop - эффективный и практичный алгоритм оптимизации глубоких нейронных сетей. В настоящее время он считается одним из лучших методов оптимизации⁶.

Проведя обучение модели на протяжении 50 эпох, с размером пакета одновременной обработки изображений в 256 элементов были получены следующие результаты, описанные ниже.

⁶ Глубокое обучение / Ян Гудфеллоу, Иошуа Бенджио, Аарон Курвилль // ДМК Пресс, 2018г., второе цветное издание, исправленное.

Как можно судить по графику показаний функции потерь на рисунке 9, к 30 – й эпохе модель содержит уже достаточно оптимальные значения параметров и дальше происходит уже топтание вокруг оптимума, что иногда приводит к аномальному росту значения ошибки.

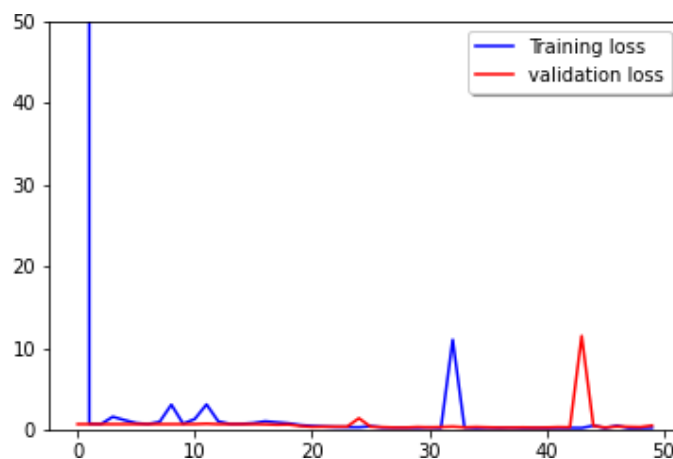


Рисунок 9

Аналогичная картина наблюдается и с графиком метрик, в частности точности (accuracy), максимальное значение которой достигается еще до 30 – эпохи и дальше происходит уже брожение вокруг оптимума с последующими аномальными скачками. Но очень положительным моментом является то, что значение метрики как для тренировочной выборки, так и для тестовой подобны, что говорит об отсутствии переобучения модели. Также стоит отметить, что во время обучения модели точность достигла своей заложенной асимптоты.

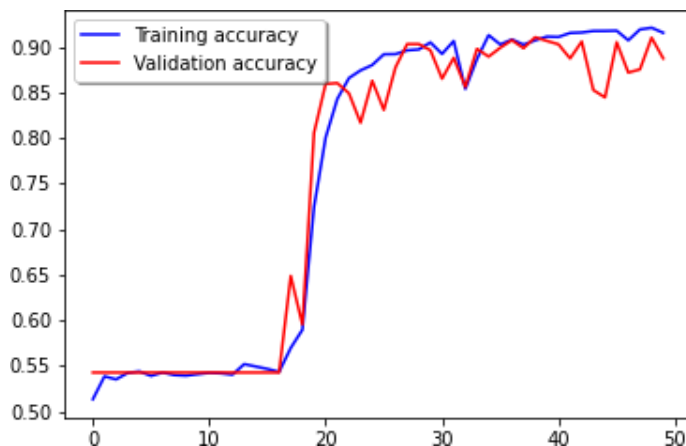


Рисунок 10

Площадь под кривой PR так же достигает своего максимума к 30 – й эпохе, а график метрики на тренировочной выборке так же подобен графику метрики на тестовой выборке, что еще раз доказывает отсутствие переобучения.

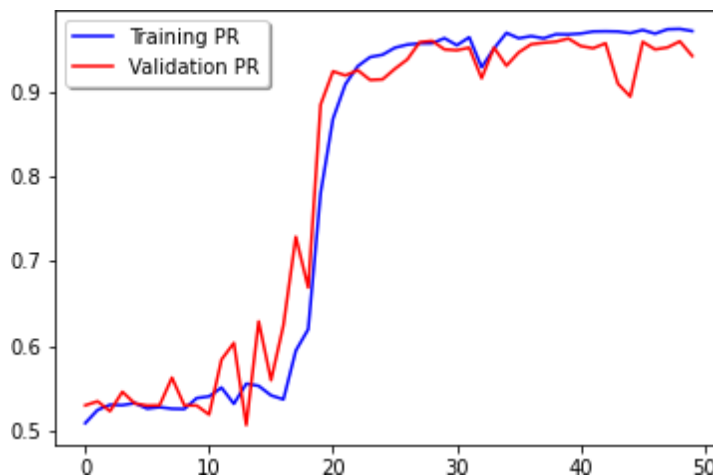


Рисунок 11

На основании данных результатов данная гипотеза принимается и считается одним из оптимальных решений.

Гипотеза о применимости алгоритма оптимизации Adam

Продолжим исследование влияния алгоритма оптимизации на обучение сверточной нейронной сети для бинарной классификации. Алгоритм Adam, переводится как адаптивные моменты (adaptive moments), является комбинацией вышеописанного алгоритма RMSProp и импульсного метода, суть которого заключается в том, что найденное направление и определяемый шаг будет затухать при приближении к оптимуму. Пример движение импульсного алгоритме оптимизации на рисунке 12, где красной линией отображена траектория импульсного метода.

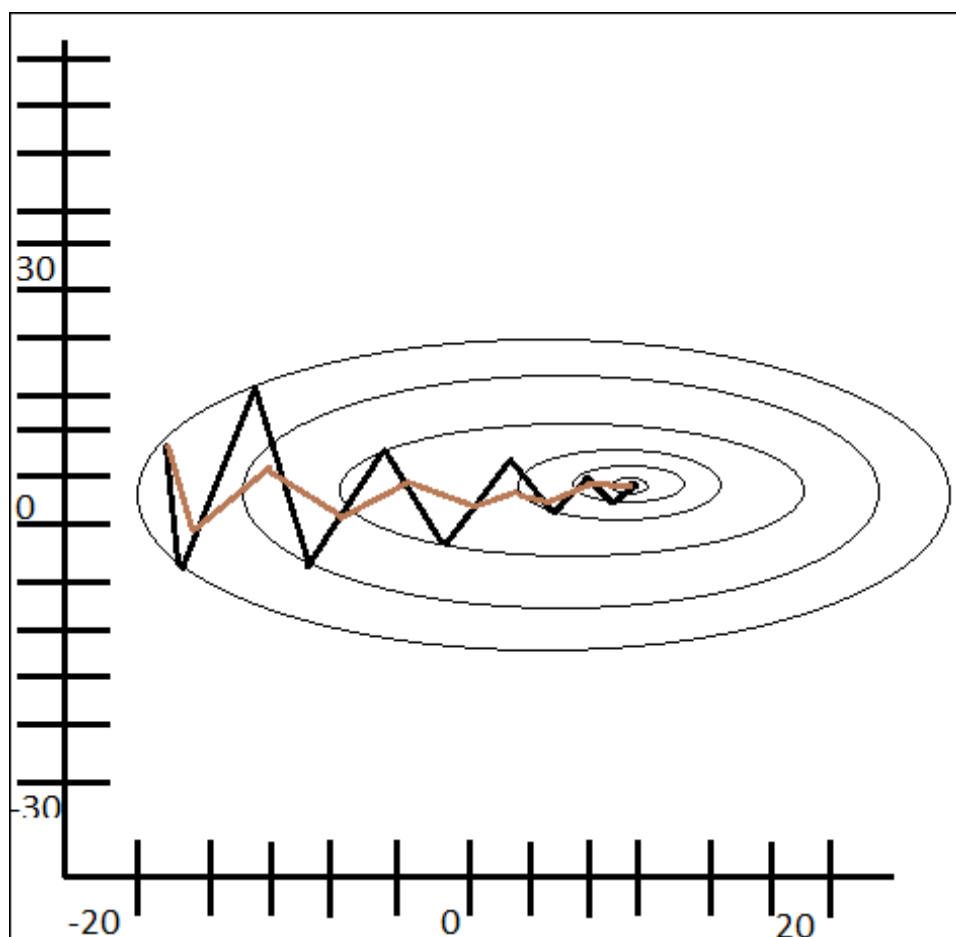


Рисунок 12

Метод Adam имеет следующий алгоритм:

Как и метод RMSProp, этот имеет общий вид:

$$w = w + h \odot \Delta w,$$

где w – матрица весов нейронной сети, а Δw – направление для минимизации целевой функции, а h – шаг или в нашем случае это скорость обучения. Вся суть метода в том, как мы находим наше направление.

Возьмем нашу скорость h и скорость затухания ρ_1 и ρ_2 для оценок моментов. Определим наше начальное значение весов w . И определим малую константу $\varepsilon = 10^{-8}$ чтобы избежать деления на нуль в будущем. Задаем параметр для первого и второго момента $s = 0$, $r = 0$. Задаем шаг по времени $t = 0$. И пока условие остановки не выполнено, повторяем следующие шаги:

1. Выбираем часть экземпляров количества m
 $\{x^1, \dots, x^m\}$

из нашей выборки и соответствующие им y^i

2. Находим градиент

$$g = \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial w} L(f(x^i, w), y^i),$$

Где $L(f, y)$ – целевая функция, а $f(x)$ – функция активации.

$$t = t + 1$$

3. Обновляем смещенную оценку первого момента

$$s = s\rho_1 + (1 - \rho_1)g$$

4. Обновляем смещенную оценку второго момента

$$r = r\rho_2 + (1 - \rho_2)g \odot g,$$

где $g \odot g$ – это поэлементное произведение матриц градиента.

5. Скорректировать смещение первого момента

$$\hat{s} = \frac{s}{(1 - \rho_1^t)}$$

6. Скорректировать смещение второго момента

$$\hat{r} = \frac{r}{(1 - \rho_2^t)},$$

7. Вычисляем направление

$$\Delta w = - \frac{\hat{s}}{\sqrt{\hat{r}} + \varepsilon}$$

8. Находим новое значение

$$w = w + h\Delta w$$

Обучив нейронную сеть на протяжении 50 – ти эпох, были получены разочаровывающие результаты, представленные на рисунках ниже.

На рисунке 13 отображено поведение функции потерь. По данному графику можно сказать, что данный метод справился со своей задачей, поэтому перейдем к показанию метрик точности (ассигасу) на рисунке 14, площади под кривой PR на рисунке 15.

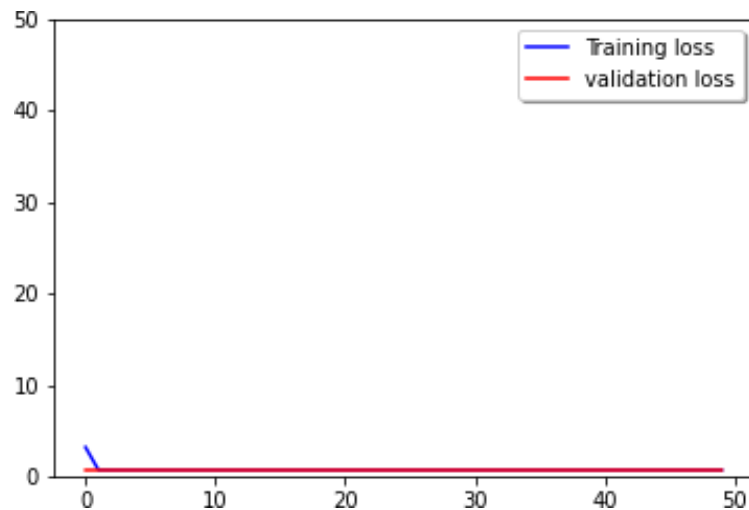


Рисунок 13

Показатели точности на рисунке 14 лишь на несколько сотых превышают значение базового решения, что является отрицательным результатом с учетом разницы вычислительных и временных затрат. Как можно судить по графику, можно предположить, что данный градиентный метод достиг локального минимума, из которого не удастся выбраться.

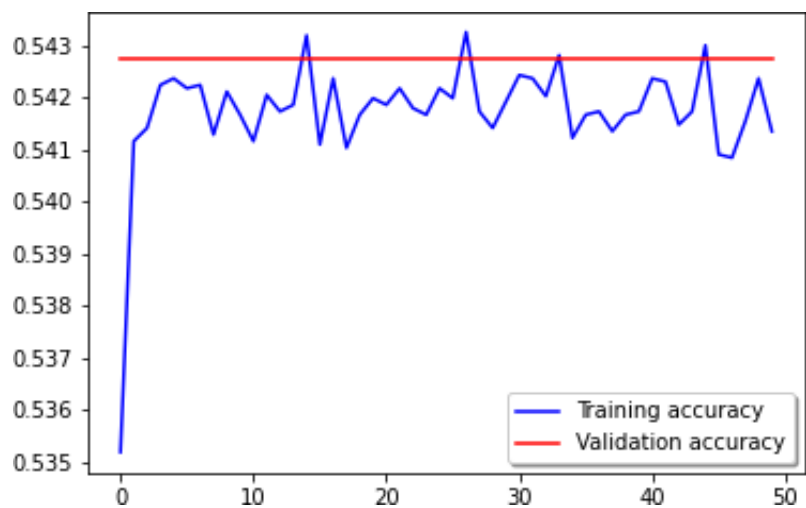


Рисунок 14

Значение площади под кривой PR на рисунке 15 может только увеличивает вероятность попадания в локальный минимум.

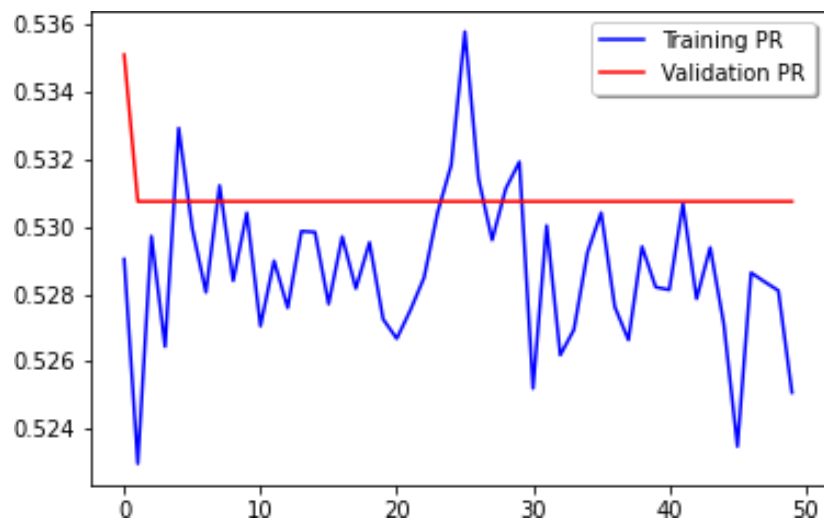


Рисунок 15

На основании вышеописанных результатов гипотеза о рациональности использования метода Adam в данной задаче отвергается.

Гипотеза об увеличении ядра свертки

Все эксперименты до этого проводились с ядром свертки размерности 3×3 . Ядро свертки – это матрица, в нашем случае квадратная, участвующая в линейной операции свертки, где второй операнд – наше изображение.

Рассмотрим операцию свертки на примере матричного вида изображения, только элемент будет в черно – белом варианте, то есть будет задана только одна интенсивность серого. Это будет иметь вид таблицы 1.

(0, i)	Greyscale	...	(w, i)	Greyscale
i = 0	126		i = 0	43
i = 1	126		i = 1	47
i = 2	126		i = 2	49
.....			
i = h - 3	117		i = h - 3	200
i = h - 2	115		i = h - 2	204
i = h - 1	116		i = h - 1	214

Табл.1 – пример матричного отображения черно-белого изображения.

Как видим, в данном случае мы имеем дело с двумерной матрицей. Далее скажем, что наше ядро свертки будет иметь размерность 3×3 , которая заполнена весами, как в таблице 2:

w_{00}	w_{01}	w_{02}
w_{10}	w_{11}	w_{12}
w_{20}	w_{21}	w_{22}

Табл.2 – ядро свертки с значениями весов.

Далее формально представим подматрицу в таблице 3, взятой из матрицы таблицы 1:

(w = j, h = i)	Greyscale		
	j = 0	j = 1	j = 3
i = 0	a_{00}	a_{01}	a_{02}
i = 1	a_{10}	a_{11}	a_{12}
i = 2	a_{20}	a_{21}	a_{22}

Табл.3 – подматрица матрицы интенсивностей серого цвета изображения.

Далее проведем операцию свертки:

$$\sum_{j=0}^2 \sum_{i=0}^2 (a_{ji} * w_{ji}) = a_{00} * w_{00} + a_{01} * w_{01} + a_{02} * w_{02} + \dots + a_{22} * w_{22} = c_{kl}.$$

Так как ядро свертки проходит над матрицей изображения каждый участок, например, с шагом в одну клетку, то образуется следующая матрица в таблице 4, результат свертки:

c_{00}	...	c_{0n}
...		...
c_{m0}		c_{mn}

Табл.4 – результат свертки.

В цикле процедура свертки будет выглядеть как на рисунке 16:

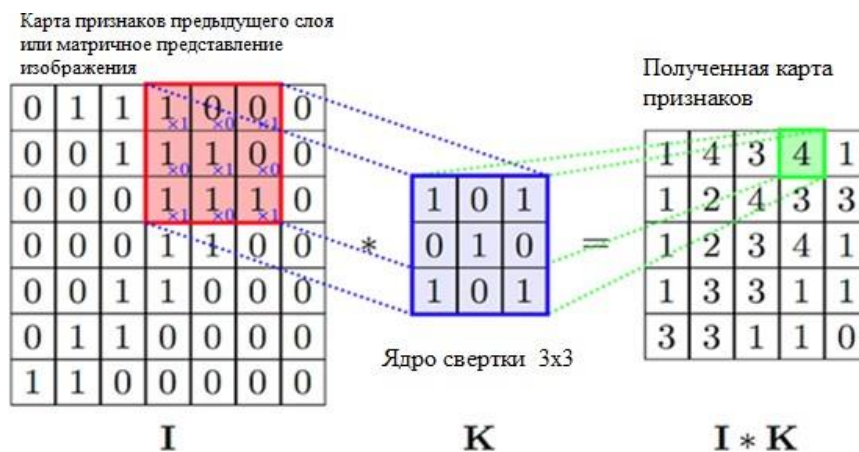


Рисунок 16

Далее результат свертки C передается в функцию активации $F(C)$, которая образует выходной сигнал нейрона, множество которых образует новую матрицу для дальнейшей обработки в следующих слоях.

Итак, ядро свертки напрямую влияет на качество обучения нашей модели. Во – первых, от размерности ядра свертки и от количества ядер в одном слое будет зависеть количество параметров нашей сети. Во – вторых, размерность ядра влияет на емкость модели, то есть на размерность получаемой матрицы после операции свертки, которая уже передается в следующие слои и обрабатывается дальше.

Далее исследуем поведение модели, если задать матрицу свертки размерности 5×5 .

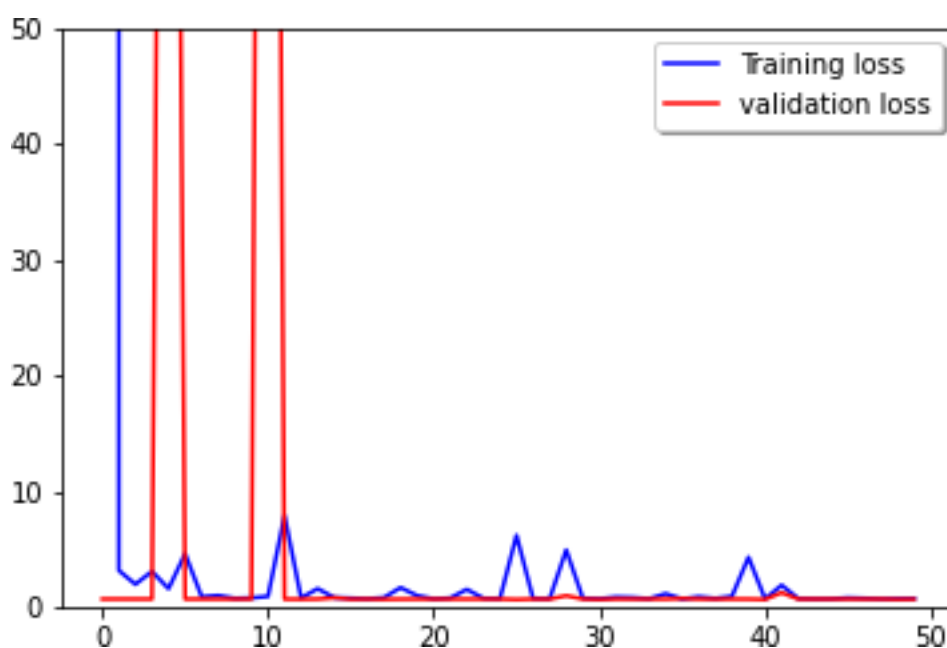


Рисунок 17

Как можно наблюдать на рисунке 17, функция ошибки стремится к нулю, хоть и присутствуют anomальные скачки. Из чего можно сделать вывод, что положение весов близко к оптимальному состоянию.

Однако, показатели точности (accuracy), на рисунке 18 говорят об отрицательном результате, ведь увеличение ядра свертки вдвое увеличило количество параметров в сети, а максимальное значение метрики снизилось почти на 30%, не говоря уже о том, что результаты меньше, чем у решения с помощью человеческих ресурсов.

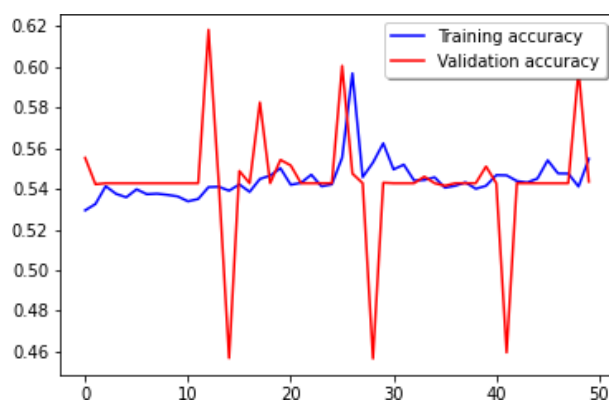


Рисунок 18

Подобные результаты и у метрики площади под кривой PR на рисунке 19. Максимальное значение метрики уменьшилось на 30%.

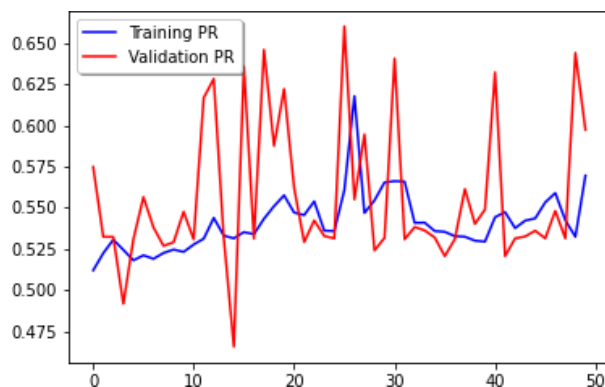


Рисунок 19

Итак, по количественным результатам, гипотеза об увеличении свертки отклоняется. Причем, судя по графикам метрик, модель попала в локальный минимум на обучающей выборке, так как показатели тестовой выборки в среднем выше тренировочной.

Гипотеза об уменьшении свертки

Продолжая эксперименты с размерностью ядра, уменьшим его, что позволит уменьшить количество обучаемых параметров модели и обрабатывать больше данных на последующих слоях.

Показатели функции потерь на рисунке 20 говорят о равномерной оптимизации параметров модели, что говорит об оптимальном состоянии сверточной нейронной сети.

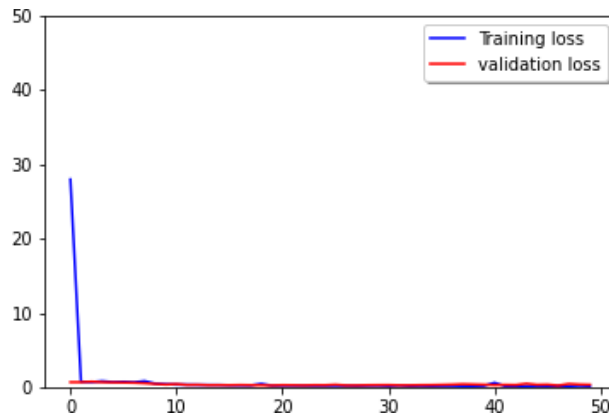


Рисунок 20

Точность (ассигасу) на рисунке 21 имеет уже знакомый вид, причем максимальное значение метрики еще больше приблизилось к заложенной асимптоте. Но увеличивающийся, с номером эпохи, разрыв между показателем тренировочной и тестовой выборки может говорить о переобучении, что требует дополнительного исследования.

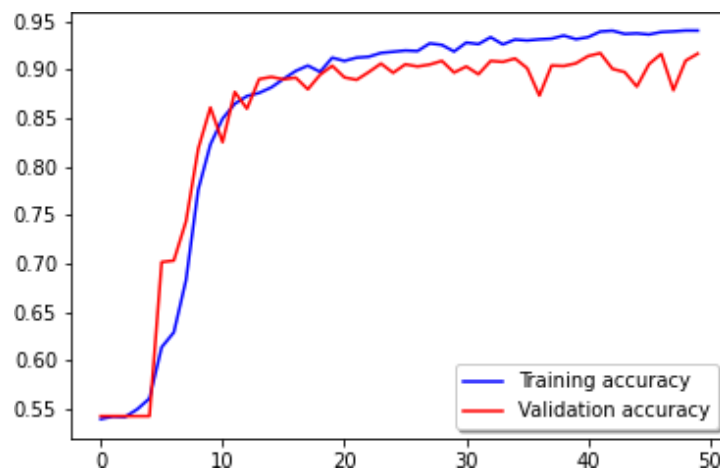


Рисунок 21

На графике площади под кривой PR на рисунке 22 так же присутствует увеличивающееся отклонение между метрикой на тренировочной выборке от тестовой, что так же может говорить о переобучении.

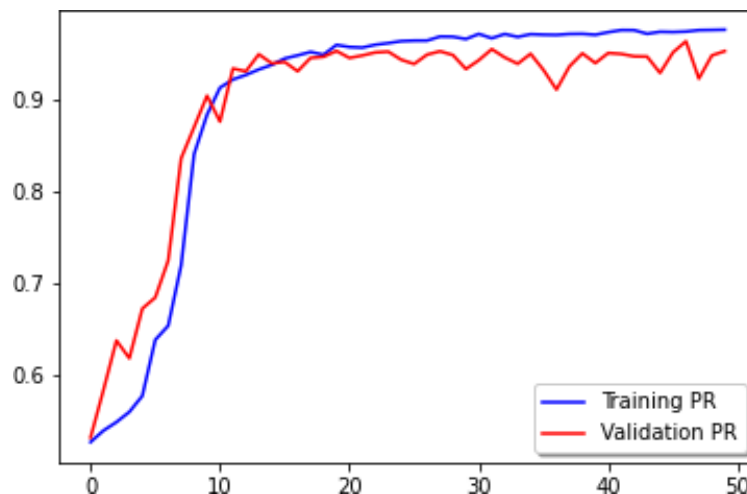


Рисунок 22

По результат метрик гипотеза принимается и будет исследоваться дальше.

Гипотеза об увеличении количества эпох при уменьшенном ядре свертки

Так как результаты предыдущего исследования говорят о возможном переобучении, нужно проверить этот аспект путем увеличения количества эпох.

Обучив модель на протяжении 100 эпох, были получены следующие результаты.

На графике значений функции ошибки на рисунке 23 неожиданно появились аномальные скачки, даже там, где до этого они не появлялись, что можно обусловить случайным шумом.

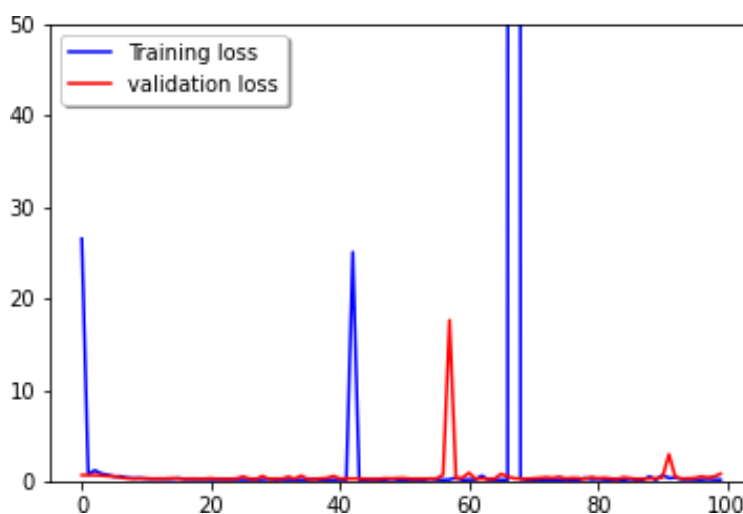


Рисунок 23

На графике метрики точности (ассигасу) на рисунке 24 можно наблюдать, что переобучение отсутствует, а максимальное значение точности еще больше приблизилось к асимптотике в 92.7%

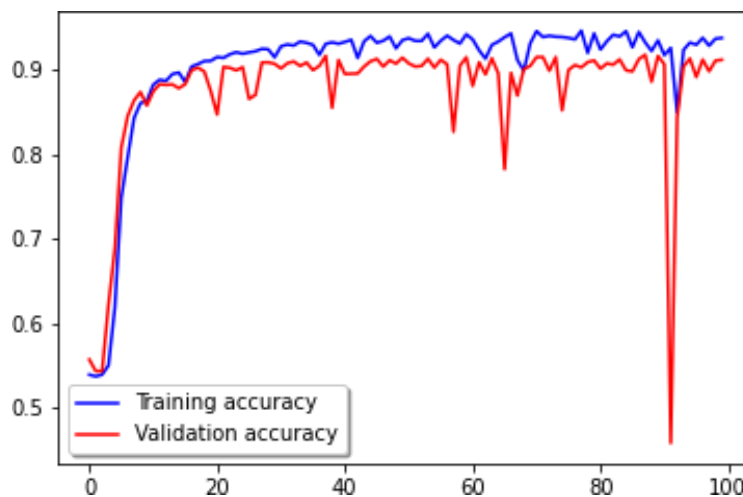


Рисунок 24

По графику значения площади под кривой PR на рисунке 25 так же наблюдается отсутствие переобучения.

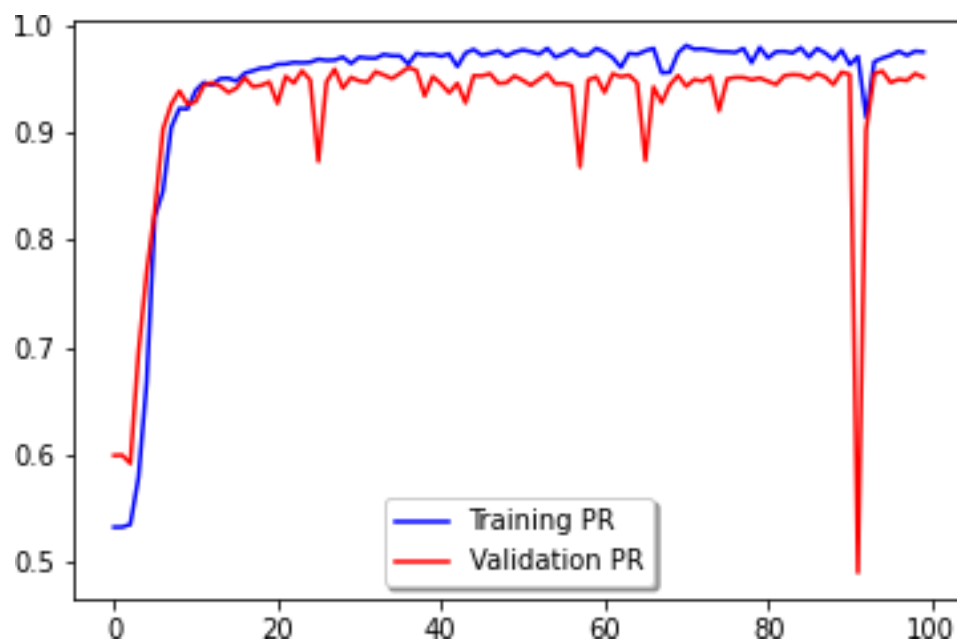


Рисунок 25

Увеличение эпох опровергло подозрение о переобучении, а также доказало, что предел значений метрик все еще не достигнут. Причем по кривым метрик для обучающей и тестовой выборке видна некоторая зависимость, что является положительным аспектом.

Гипотеза о достижении заложенной асимптоты

Проведя ряд исследований, мы определили оптимальные параметры модели и приблизились к заложенной в структуру данной сверточной нейронной сети асимптоте по точности (ассурасу) и появилась идея совершить масштабное исследование и обучить модель на протяжении 1000 эпох.

Были получены следующие результаты. Показатели функции потерь на рисунке 26 говорят о достижении оптимального устойчивого состояния весов к 200 – й эпохе, после этого порога возрастает количество аномальных скачков и вид кривой принимает колебательный характер, а после достижения 700 – й эпохи, судя по графику, модель попадает в локальный минимум, откуда ей не удастся выбраться все оставшееся время обучения, из – за чего ожидается и ухудшение показателей метрик после 700 – й эпохи.

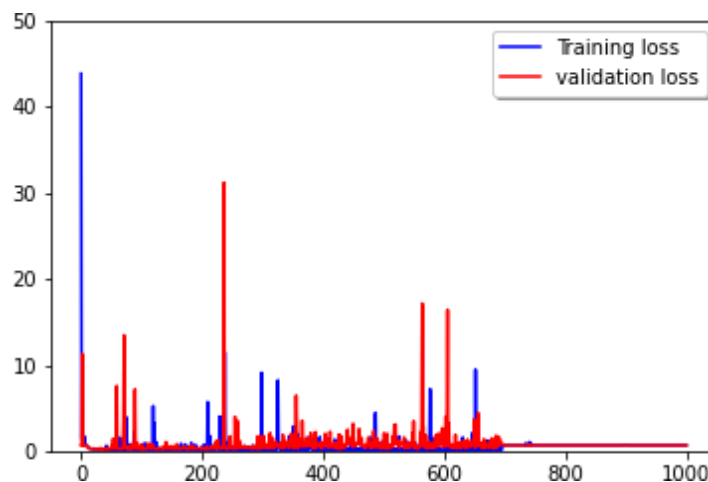


Рисунок 26

Как и ожидалось, график метрики точности (ассурасу) на рисунке 27 подтверждает состояние локального минимума у параметров модели и на графике все же отчетливо видно расхождение кривой обучающей и тестовой выборке, что говорит все же о наличие переобучения, но нельзя не отметить обновленное максимальное значение метрики в 92.5%, что является максимум по

итогах исследований. В данном случае, бороться с переобучением может помочь увеличение объема обучающей выборки.

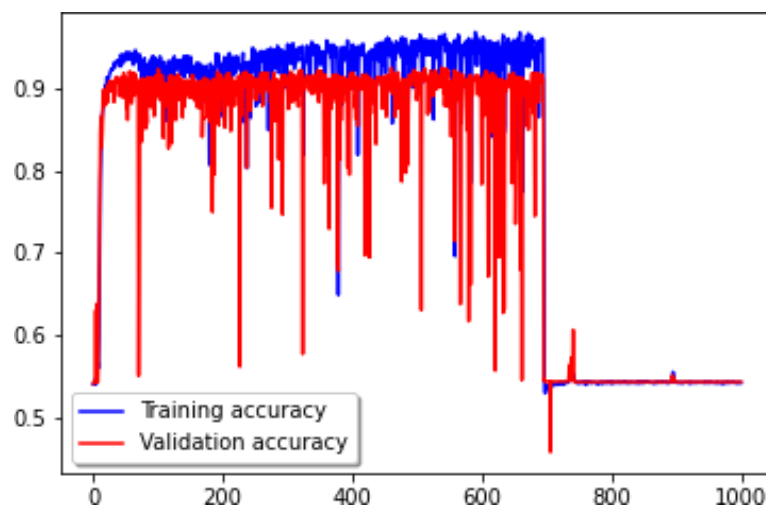


Рисунок 27

График площади под кривой PR на рисунке 28 так же отчетливо отражает попадание в локальный минимум и присутствие переобучения, причем оно здесь более ярко выражено.

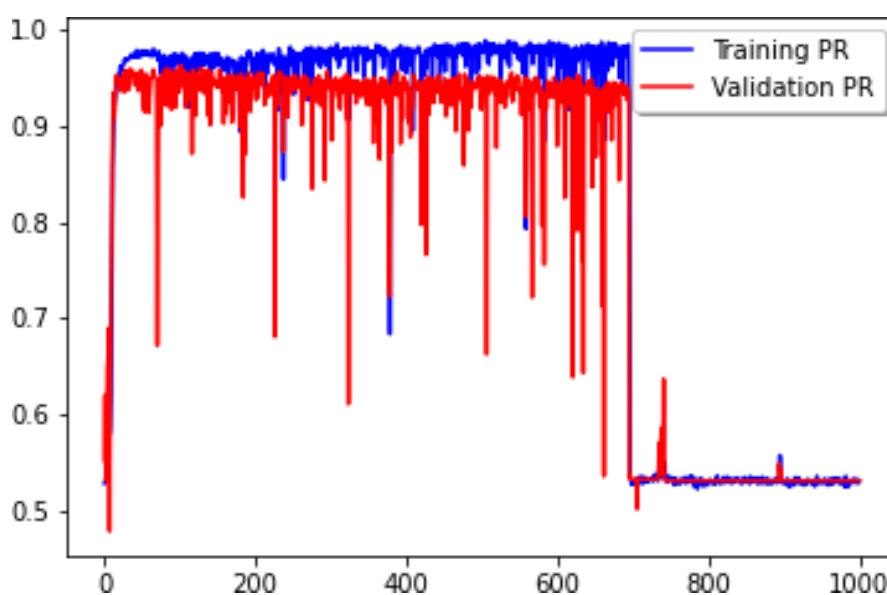


Рисунок 28

Хоть данное исследование и позволило достичь нового предела метрики для тестовой выборки, но результат вряд ли нивелирует временные затраты, так что обучение на протяжении 1000 эпох можно считать избыточным, и гипотеза отклоняется.

Исследование гипотез задачи семантической сегментации

Первые результаты

Исследовав задачу бинарной классификации, стоит так же упомянуть и результаты первых результатов решения задачи семантической сегментации.

В данном решение используется уже упомянутая раньше модель сверточной нейронной сети U – Net, структуру которой мы отобразим в приложении. Обучение сети проходит на том же наборе данных, что и для бинарной классификации, на протяжении 50 – ти эпох, а параметры модели оптимизировали уже знакомым методом RMSProp.

Так как модель только предстоит исследовать в дипломной работе, не будем обращать внимание на метрики, а продемонстрируем ее работу.

На рисунке 29 отображен результат работы данной сети. В первом столбце находится исходное изображение. Во втором столбце отображен результат обработки исходного изображения моделью, где каждому пикселю изображения ставится в соответствие вероятность того, что этот пиксель принадлежит кораблю, чем краснее – тем больше вероятность. В третьем столбце предыдущий результат проходит через фильтр, оставляя лишь пиксели, превышающие порог. В последнем столбце отображен истинный ответ.

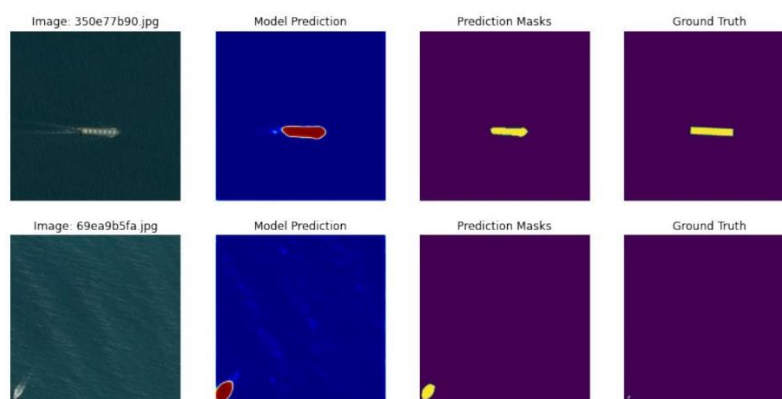


Рисунок 29

Как можно судить по рисунку 29, модель научилась распознавать изображения на эталонных изображениях – когда корабль четко отличим от фона.

На рисунке 30 видно, что модель хорошо распознает корабли даже на тех изображениях, которые имеют некорректный истинный ответ.

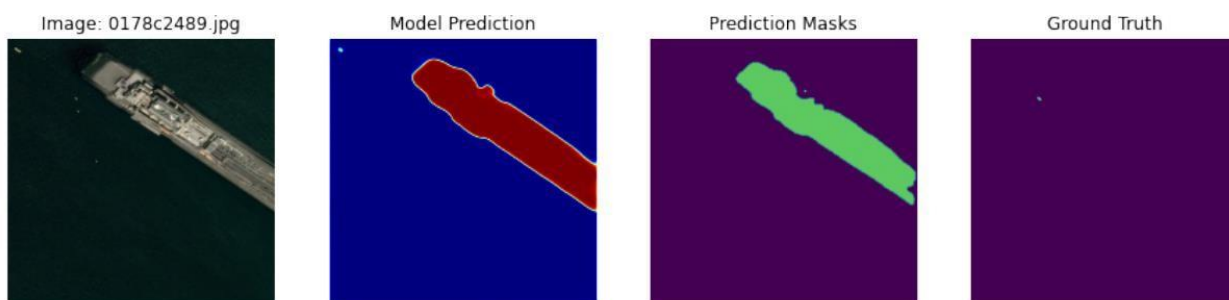


Рисунок 30

Однако, на зашумленных изображениях, которое присутствует на рисунке 31, модель ведет себя некорректно и не может даже близко пометить истинное местоположение морского судна. Борьба с такой проблемой нужна «обогащением» обучающей выборки, то есть добавлением в нее зашумленных изображений и удалением изображений, на которых корабль отсутствует.

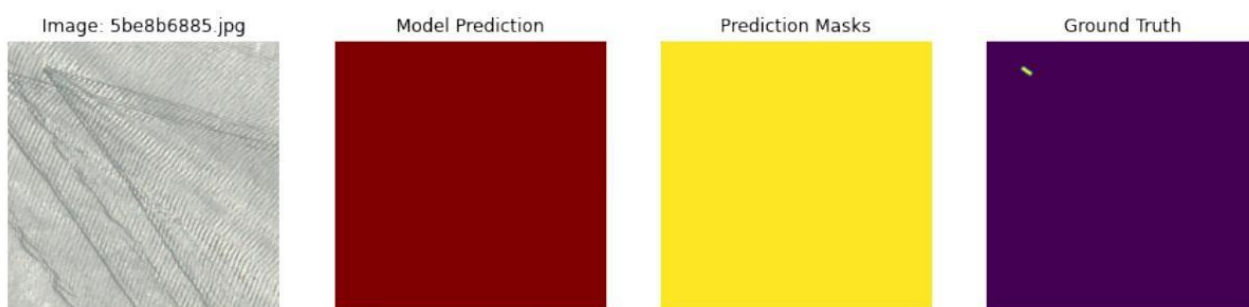


Рисунок 31

Опорная гипотеза

Перед исследованием определим первоначальные результаты и как они были получены. Для оптимизации весов нашей сверточной нейронной сети U-Net будет использоваться метрика DICE, уже описанная выше. Но метрика принимает значения от 0 до 1, где 1 – полное соответствие двух множеств, что и является целью. Тогда для минимизации будет использоваться функция потерь вида:

$$DICE_LOSS = 1 - DICE$$

Оригинальная архитектура сети изображена на рисунке 1.

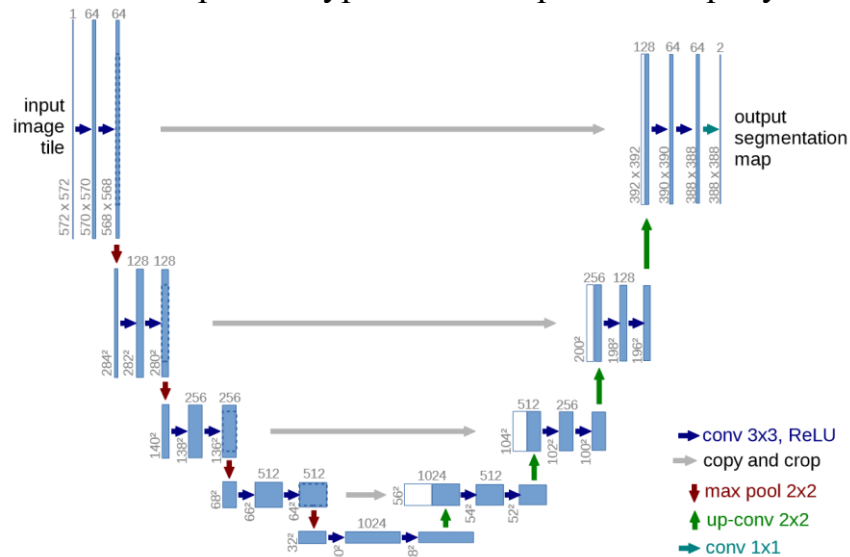


Рисунок 32

Но из – за технических ограничений, мы сократим количество сверточных ядер на 8 на каждом слое. Тогда вместо [64, 128, 256, 512, 1024] множество количества ядер сверток будет содержать [8, 16, 32, 64, 128]. К сожалению, это напрямую отразится на качестве обучения.

Проведя обучение на протяжении 100 – а эпох были получены следующие метрики: график функции ошибок на рисунке 32 и график самой метрики DICE на рисунке 33.

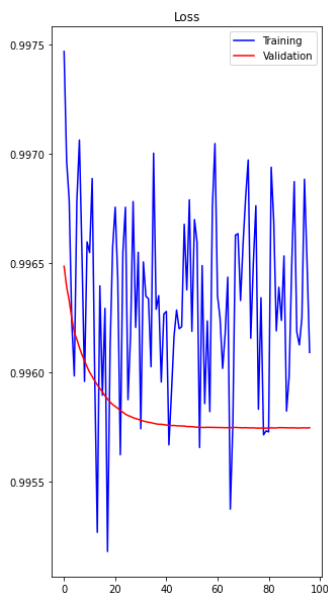


Рисунок 32

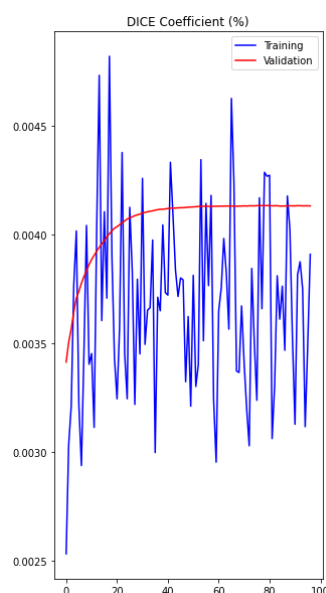


Рисунок 33

Как видно на рисунке 32 и 33, функция потерь не достигла оптимальных значений, но во время обучения была достигнута ассимптота. Так как функция потерь зависит от метрики DICE, то исключим ее из рассмотрения и будем ориентироваться только по метрике.

После подачи изображений модели для получения предсказания местоположения морского судна на изображении были получены следующие результаты, часть из которых изображена на рисунке 34.

По двум из четырех примеров можно понять, что модель действительно распознает корабли на изображении, где корабль явно выделяется на фоне морской глади. Однако имеются проблемы с менее контрастными изображениями.

На рисунке 34 в последнем примере можно заметить несовпадение по количеству отмеченных кораблей на изображении – «Ground Truth» и определяемых визуально на исходном изображении.

После данной заметки пришлось изучить исходные данные. Было выявлено, что набор табличных записей содержал в каждой строчке координаты пикселей каждого корабля для каждого соответствующего изображения. Но для обучения модели в задаче

классификации были отсеены строки с одинаковыми именами изображений, что было вполне логично для задачи классификации, где модель училась давать лишь бинарный ответ по всему изображению. Однако уже в задаче сегментации это нанесло большой ущерб, ведь для модели один набор пикселей, где находится корабль, таковым и является. А другой корабль для модели будет помечен как фон.

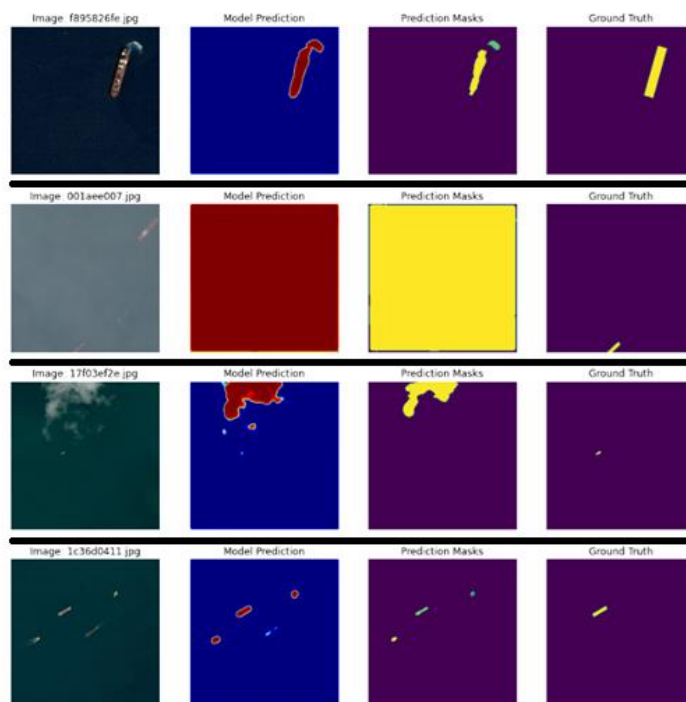


Рисунок 31

Исследование модели на корректных данных

По результатам предыдущей гипотезы была выявлена некорректность исходных данных и данные были дополнены. На гистограмме с рисунка 35 изображено распределение кораблей по изображениям.

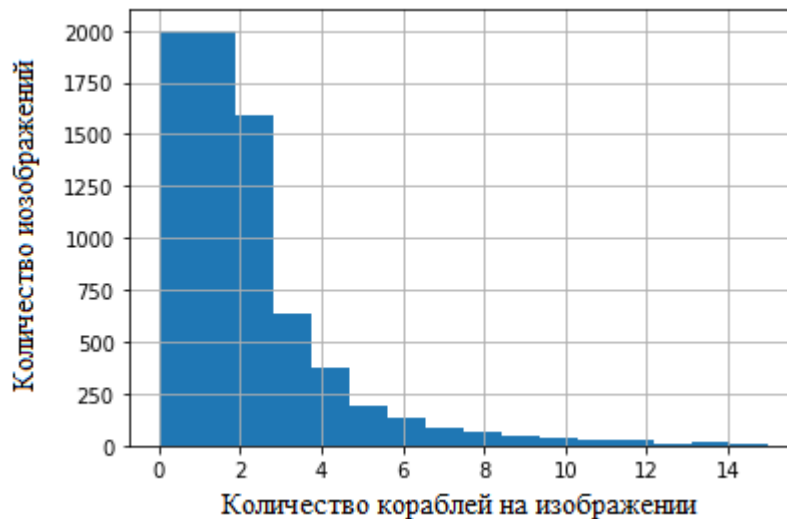


Рисунок 35

Так как объемы для обработки возрасли, дальнейшее обучение моделей проходило на протяжении 50 – ти эпох и для прошлой гипотезы с корректными исходными данными были получены следующие результаты метрики на рисунке 36.

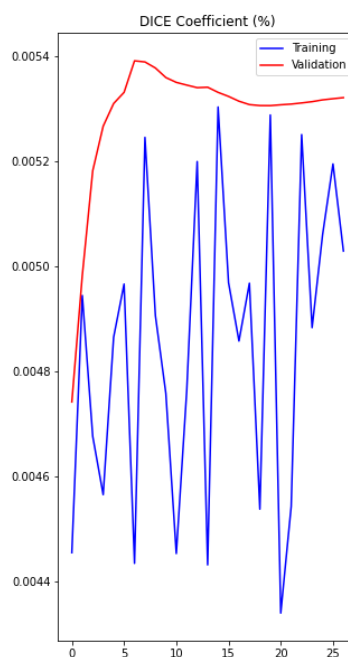


Рисунок 32

На графике отображены данные лишь для 25 эпох, так как метод оптимизации перестал давать значительные изменения метрики на последних эпохах, из — за чего произошла автоматическая остановка.

Но даже по этим данным видно, что произошел качественный прирост, по сравнению с предыдущими результатами.

Результат работы модели с изображениями на рисунке 37. Теперь модель обучалась корректно и она распознает несколько кораблей на одном изображении, но теперь отсутствуют четкие контуры морских судов, что говорит о недообучении модели. Перед решением данной проблемы, попробуем обучить другие модели и сравним результаты.

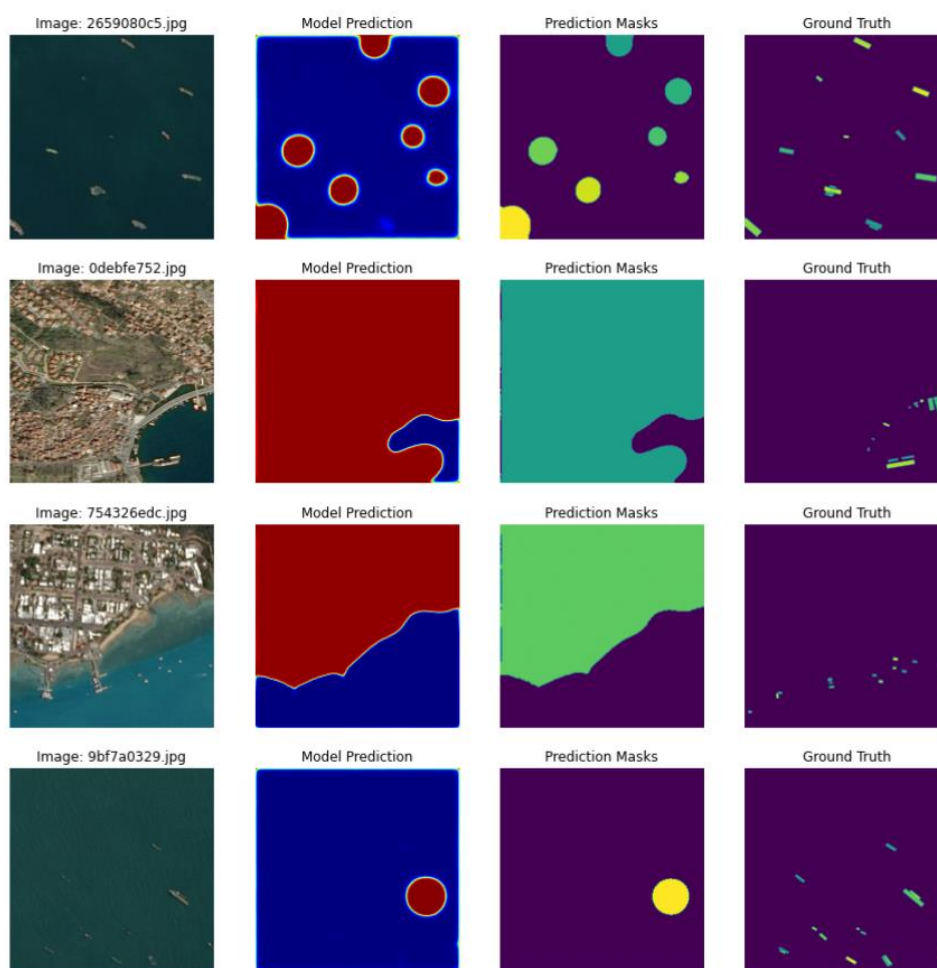


Рисунок 33

Исследование модели SegNet

Хоть архитектура U-Net появилась и недавно, но спустя год появилась новая архитектура для задачи семантической сегментации SegNet

⁷. Архитектура данной сети изображена на рисунке 38.

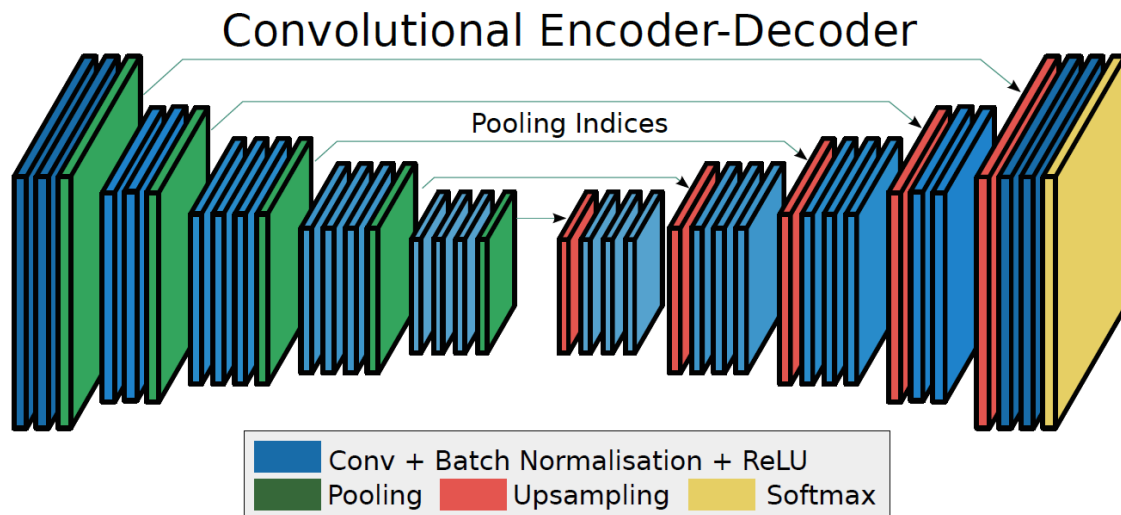


Рисунок 34

В отличие от предыдущей архитектуры U – Net, у данной отсутствует некоторый промежуточный слой посередине между набором слоев с понижающей размерностью и набором слоев с повышающей размерность, но при этом у SegNet 10 слоев, против 9 у U – Net, что должно положительно сказаться на результатах. Также данная сеть содержит нормализацию пакета данных после каждого слоя свертки, что должно ускорить обучение модели, по сравнению с U-Net, которая нормализацию не содержит.

Данная модель, как и наша опорная, будет начинаться с 8 – ми сверточных ядер, удваивая количество на каждом понижающем слое.

После обучения на протяжении 50 – ти эпох, были получены следующие результаты.

На рисунке 39 изображены данные лишь за 20 эпох, так как

⁷ SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, Senior Member, IEEE. arXiv:1511.00561v3 [cs.CV] 10 Oct 2016

значение метрики для тестовой выборки имела лишь незначительные изменения и произошла автоматическая остановка. По результатам можно сказать, что модель имеет сильное недообучение.

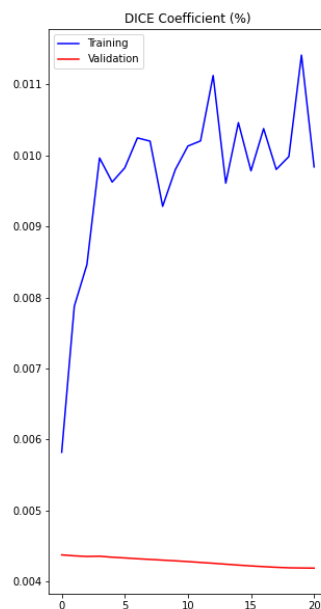


Рисунок 35

Посмотрим результат работы сети после обучения на рисунке 40.

Как и ожидалось, сеть имеет сильное недообучение и не может определить ни корабль, ни море.

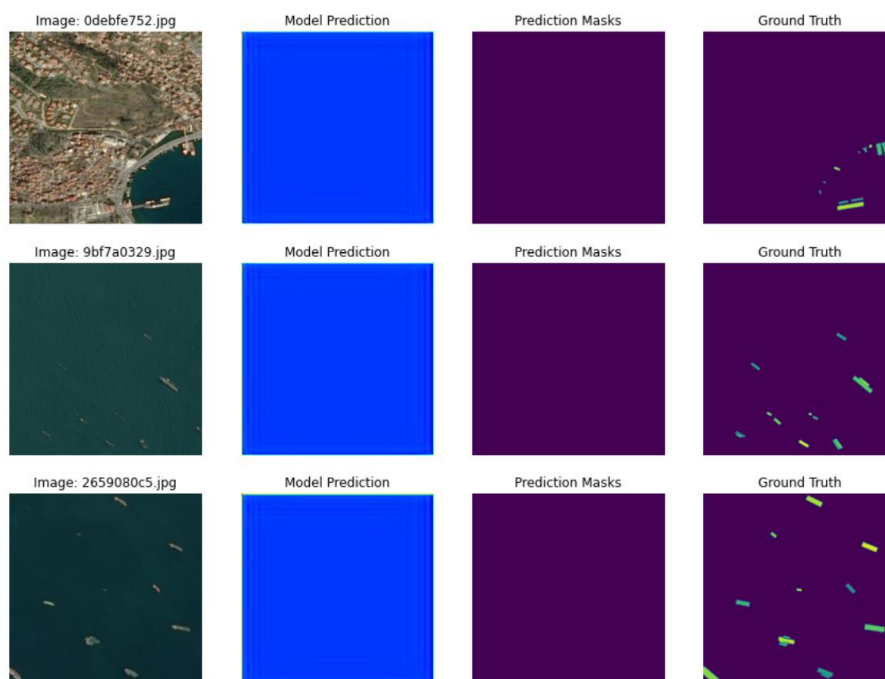


Рисунок 6

Гипотеза об увеличении числа фильтров SegNet

После отрицательного результата предыдущей гипотезы было решено увеличить число фильтров, но из — за технических ограничений, удалось увеличить число ядер свертки на каждом слое лишь в 1.5 раза.

После обучения на протяжении 50 — ти эпох были получены следующие результаты.

На рисунке 41 отчетливо видно, что произошло улучшение метрики для тестовой выборки и что ассимптота даже близк оне достигнута, так что было проведено повторное обучение уже на протяжении 100 эпох. Результаты приведены на рисунке 42.

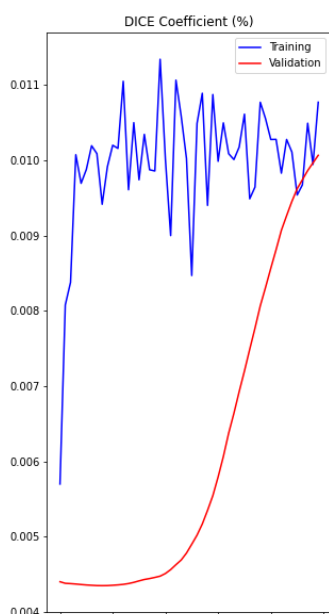


Рисунок 41

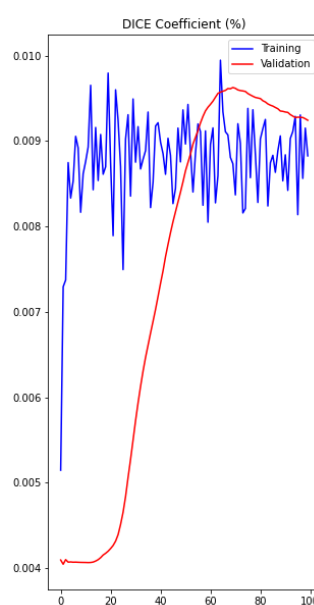


Рисунок 42

К сожалению, ожидания не оправдались и к 70 — ой эпохе метрика начала убывать.

Посмотрим на результаты обучения на рисунке 43. Как видно по рисунку 43, модель все же научилась распознавать корабли, хоть и с достаточно малой точностью, но также стоит учесть, что модель смогла распознать корабли даже на фотографиях с малой контрастностью, чего не удавалось предыдущим моделям.

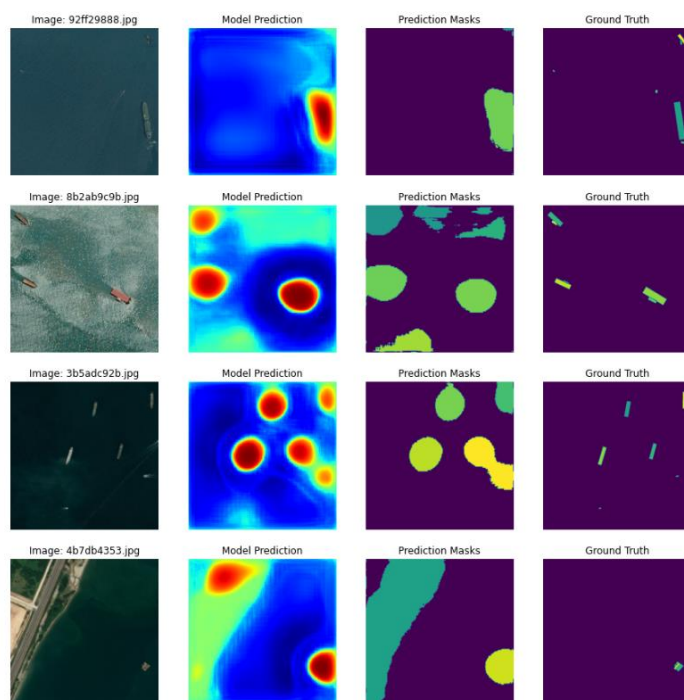


Рисунок 43

Возможно, результаты данной модели можно улучшить, увеличив количество фильтров, но это повлечет за собой и увеличение количество параметров, и время обучения модели, затрачиваемое на одну эпоху. Из – за чего данная модель не будет рассматриваться далее, так как при увеличении количества фильтров у U – Net, ее результаты так же улучшатся, что в сравнении не уменьшит отставание SegNet.

Исследование модели FCN

Следующей архитектурой сверточной нейронной сети для задачи семантической сегментации стала архитектура FCN[8] - Fully Convolutional Network (Полностью сверточная сеть). Суть данной архитектуры в том, что перед использованием берется уже обученная модель для классификации, в нашем случае это будет уже исследованная ранее модель VGG16. Концептуальная идея изображена на рисунке 44.

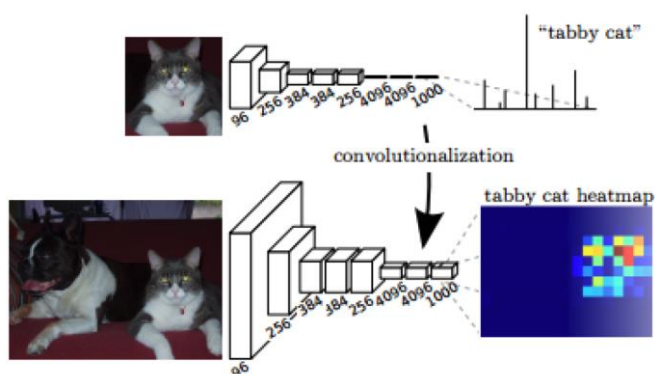


Рисунок 44

Для оценивания все еще будем использовать DICE коэффициент и посмотрим на его динамику во время обучения, на рисунке 45.

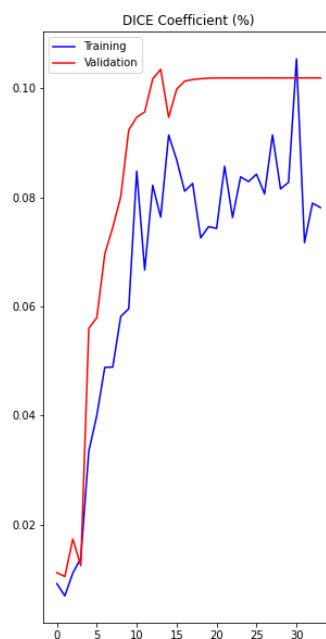


Рисунок 45

⁸ Fully Convolutional Networks for Semantic Segmentation. arXiv:1605.06211v1 [cs.CV] 20 May 2016

Как видно, модель чуть превзошла значение максимальной метрики у модели SegNet и значительно превзошла U – Net, но посмотрим на фактическую работу архитектуры с изображениями на рисунке 46.

По результатам обработки можно сказать, что модель научилась улавливать какие – то очертания морских судов, но даже так, модель имеет сильное недообучение, хотя и судя по графику метрики для тестовой выборки, итерации не давали никаких результатов. В попытках преодолеть данную проблему можно увеличить некоторые гиперпараметры модели, такие как размер батча или количество ядер свертки, но данный шаг увеличит потребляемые вычислительные мощности и превысит уровень имеющихся.

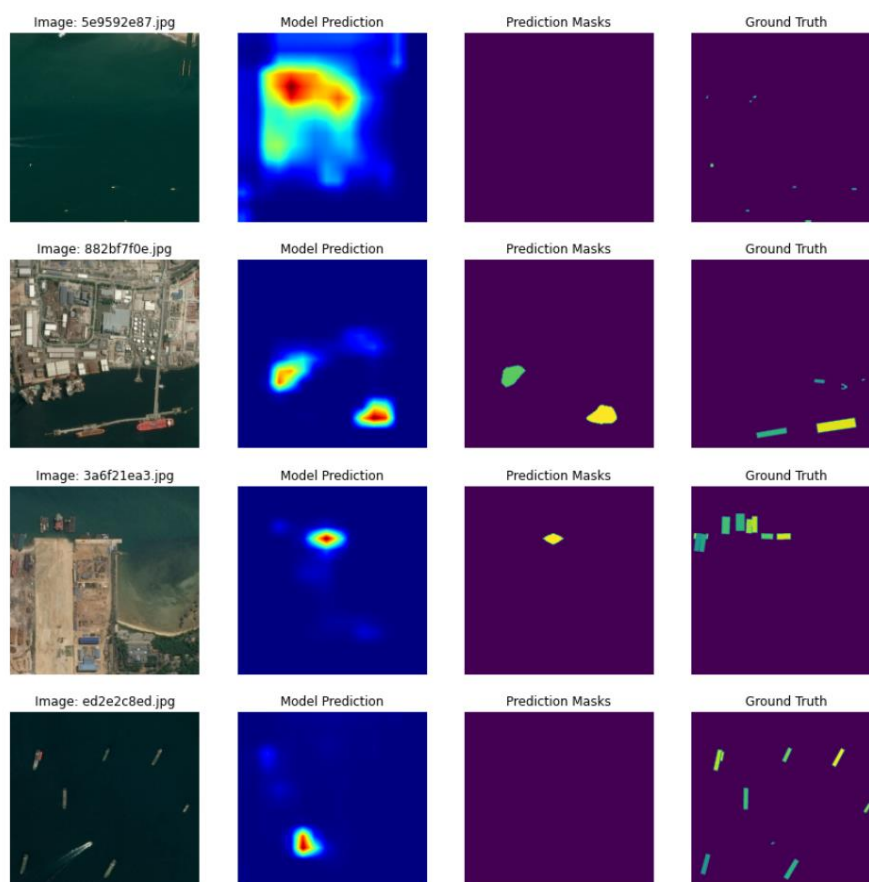


Рисунок 46

Исследование алгоритма расчета метрики DICE в модели U – Net

По результатам исследований трех моделей, хоть и являющейся не самой лучшей по значению метрики, но по фактической работе, только сверточная нейронная сеть для семантической сегментации U – Net будет исследоваться дальше и рассматриваться для полномасштабного обучения, если не на всем, то на внушительном объеме предоставленных данных.

До этого момента при расчете метрики DICE, полученные вероятности о принадлежности каждого пикселя к морскому судну, проходили через классификатор и уже преобразовывались в однозначный ответ о принадлежности каждого пикселя к кораблю.

Но возникло сомнение, что некое утаивание информации о фактической величине ошибки может ухудшить результаты оптимизации, а вследствие и обучения модели, поэтому было решено провести обучения без преобразования выходных данных.

Также была принята во внимание особенность нашего метода оптимизации RMSProp, который чувствителен к опорной точке, с которой мы начинаем оптимизацию. До этого начальные значения задавались случайным образом по нормальному распределению. Но после ознакомления с рекомендациями по начальной инициализации весов[9], для начальных значений параметров был выбран подход инициализации Ксавье. Он предложил подход, который учитывает количество входных и выходных нейронов при случайной инициализации. Данный подход позволяет избежать больших значений взвешенной суммы входных данных на скрытом слое и уменьшает вероятность попасть на плато.

Итак, после 50 – ти эпох был получен следующий результат метрики, отображенный на рисунке 47.

⁹ Why better weight initialization is important in neural networks? <https://towardsdatascience.com/why-better-weight-initialization-is-important-in-neural-networks-ff9acf01026d>

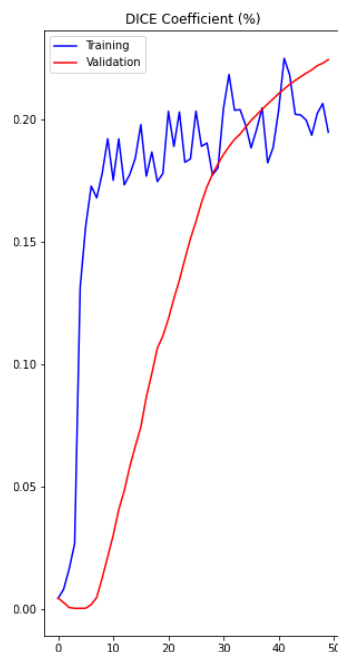


Рисунок 47

По графику можно увидеть, что максимальное значение метрики на тестовой выборке сильно превосходит значение на первых исследованиях U – Net и вдвое превосходит максимальное значение у остальных рассматриваемых архитектур.

Далее посмотрим фактическую работу по рисунку 48.

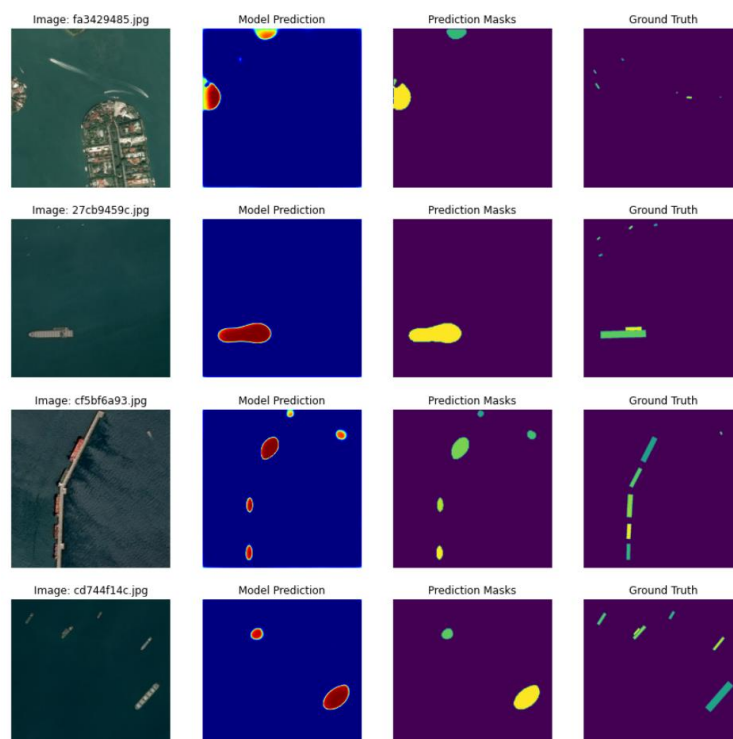


Рисунок 48

Как можно заметить, область местонахождения корабля,

определяема моделью, сильно снизилась, хоть и достаточно малые морские суда все еще не улавливаются сеть, зато модель преодолевает достаточно трудные участки, как изображение в третьем ряду и отделяет паром от корабля.

Результаты данного исследования можно назвать впечатляющими и следующим шагом будет обучение на гораздо большей выборке.

Исследование модели на большом объеме данных

Так как одним из важных факторов машинного обучения являются данные, потому что чем больше разнообразных экземпляров для обучения, тем больше вероятность, что модель найдет неявные зависимости, которые предположительно должны быть в обучающей выборке, самым главным и завершающим шагом - будет обучение сверточной нейронной сети U – Net для задачи семантической сегментации на гораздо большем объеме изображений, что использовалось во время сравнительного исследования. Теперь обучение будет проходить на выборке и 70000 изображений, где фотоснимки с морскими судами и без распределены равномерно, что уже было отображено в раннем исследовании U – Net.

Проведя обучение на протяжении 35 – ти эпох, произошла автоматическая остановка, были получены результаты метрики DICE, изображенные на рисунке 49.

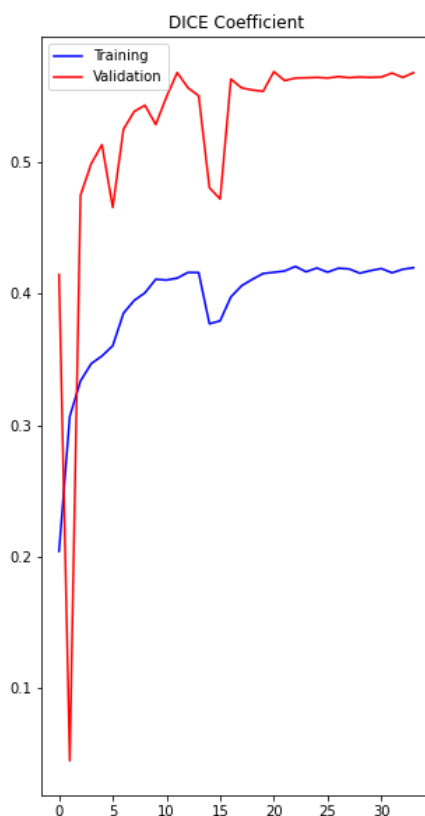


Рисунок 49

По рисунку 49, максимальное значение метрики имеет почти трехкратный прирост, только лишь после увеличения обучающей выборки.

Далее рассмотрим фактическую работу модели с изображениями на рисунке 50.

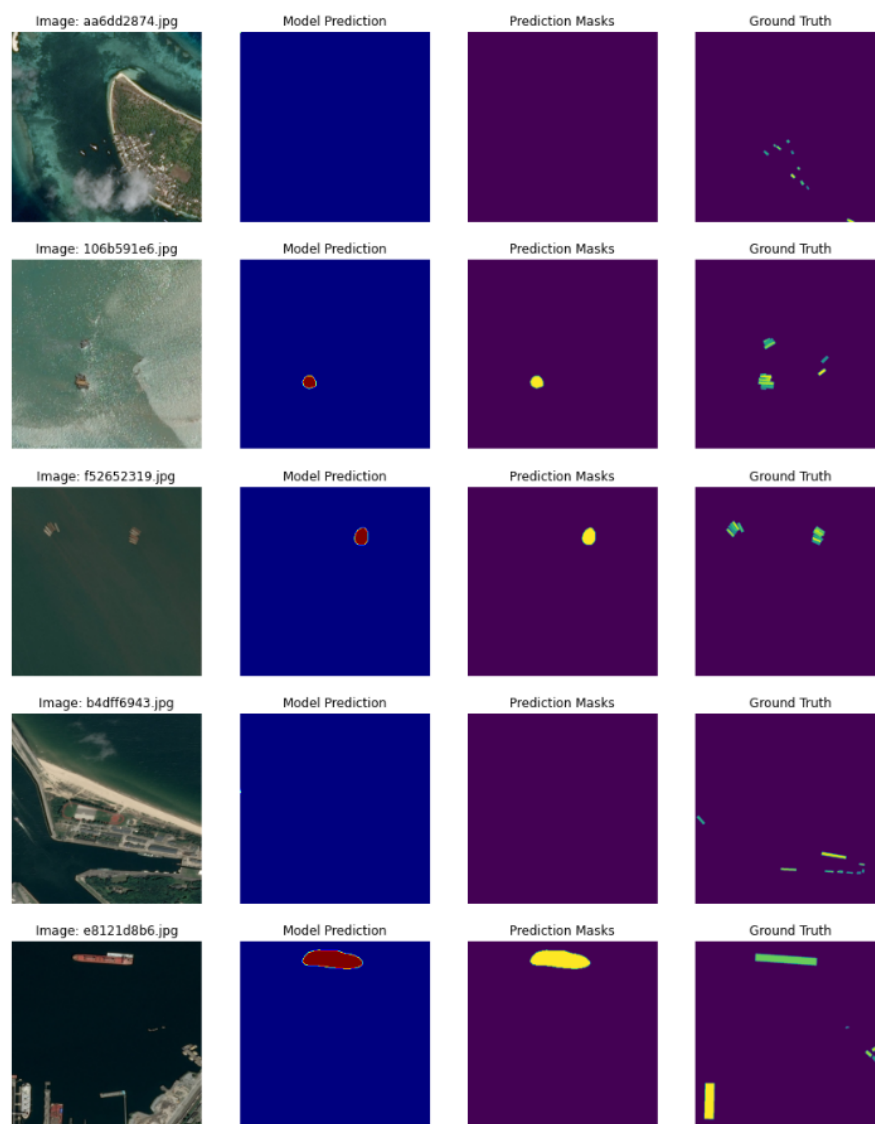


Рисунок 50

Как можно судить по рисунку 50, модель научилась достаточно хорошо распознавать контуры морских судов. Причем, даже на слабозашумленных изображениях, как например в третьем ряду. Область обнаружения корабля значительно сократилась, но малые морские суда все еще не определяются, но скорее всего, происходит это из — за уменьшения разрешения входных

изображений, что проводится для уменьшения потребляемой памяти из — за технических ограничений. Следовательно, при повышении разрешения входных изображений и при увеличении количества ядер свертки модель может достичь еще более лучших результатов, определяя корабли различных размеров на фотографиях почти любой зашумленности. Также стоит отметить почти полное отсутствие участков с ложно положительной детекцией морских судов на водной глади. Особенно критично это было с фотографиями облаков или сильно светлой морской гладью, где модель детектировала корабли. Но на рисунке 51 продемонстрируем доказательство отсутствия ложноположительного выделения корабля.

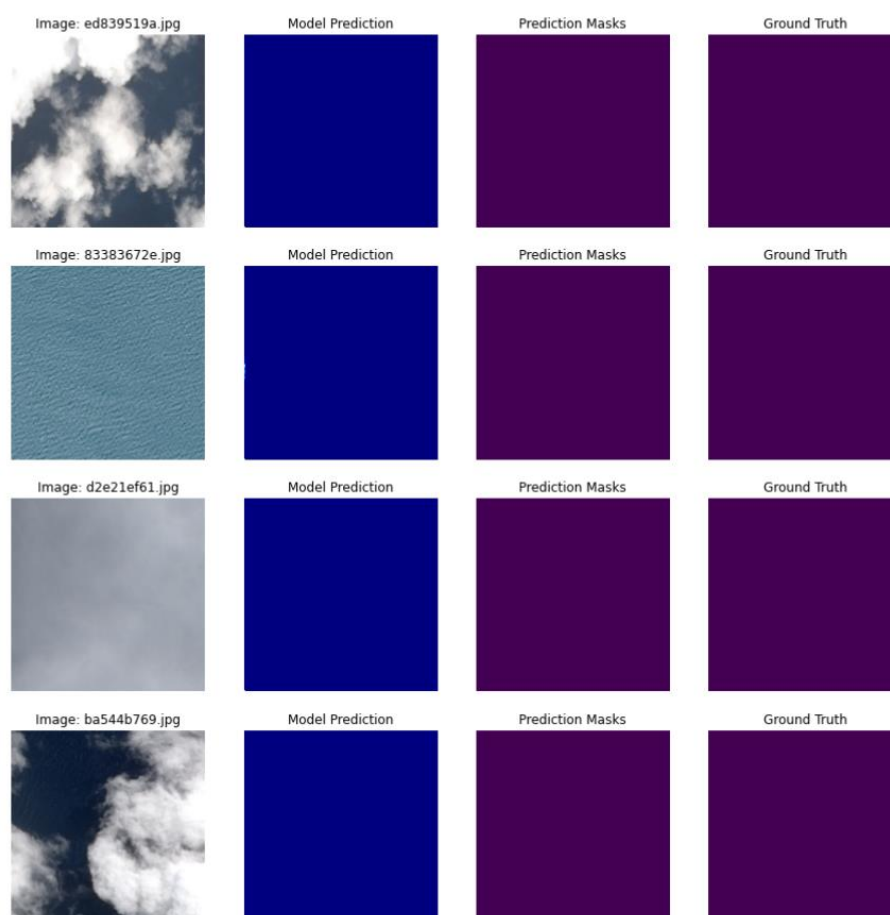


Рисунок 51

Заключение

В данной работе поэтапно решается задача распознавания морских судов на аэрофотоснимках посредством компьютерного зрения. В качестве исходных данных был использован набор изображений и табулицы с размеченными ответами для каждого изображения, предоставленный в открытом доступе компанией Airbus[10].

Сначала исследовался подход классификации изображений, где 0 – корабль отсутствует и 1 – корабль присутствует. Так как исследования увенчались успехом, дальше была поставлена задача точного определения местоположения корабля на изображении посредством семантической сегментации. Метриками качества для решения данной задачи были точность (accuracy) и площадь под кривой PR (precision – recall), а для минимизации была выбрана целевая функция в виде перекрестной энтропии.

Также в данной работе были описаны используемые архитектуры сверточной нейронной сети для решения поставленной задачи, такие как модель для классификации VGG16[7], модель для семантической сегментации U – Net[2], полностью сверточная нейронная сеть на основе нашей предобученной модели VGG16 – FCN VGG16[8][5] и сверточная нейронная сеть для семантической сегментации SegNet[7]. Метрикой качества была выбрана метрика подобия двух множеств DICE, которая позволяет работать с данными, где истинно положительных ответов на порядок меньше, чем истинно отрицательных, что и имеется на деле в используемой выборке. Она же и выступала в роли целевой функции для оптимизации моделей.

Во время исследования задачи классификации изображений были найдены оптимальные гиперпараметры модели, такие как:

¹⁰ <https://www.kaggle.com/c/airbus-ship-detection/data>

размер ядра свертки - (2×2) , метод оптимизации – *RMSProp* и продолжительность обучения – 100 эпох. При обучении модели с использованием данных параметров достигается релевантное значение метрик при затраченном времени обучения.

Во время исследования модели U-Net для решения задачи семантической сегментации был определен лучший метод оптимизации для данной задачи – *RMSProp*. В отличие от также рассматриваемого метода оптимизации Adam, который является надстройкой над алгоритмом *RMSProp*, он достигает более высоких значений метрики и за меньшее время.

Также сравнивались и сами архитектуры сверточных нейронных сетей для задачи семантической сегментации в лице трех моделей: U – Net, SegNet и в нашем случае для архитектуры FCN рассматривалась предобученная и модернизированная модель VGG16. Из сравнительного исследования выяснилось, что лучшей архитектурой для данной задачи является модель U – Net, хоть и не по значениям метрики, но по фактической работе с изображениями. После доработки алгоритма расчета метрики качества DICE и подбора начальных значений параметров модели, было проведено более масштабное обучение сверточной сети на большем наборе данных, что позволило модели достичь больших значений метрики и показать хоть и не отличную, но хорошую работу при фактической обработке изображений.

Для улучшения качества нейронной сети предлагается в дальнейшем провести ряд экспериментов с моделью U – Net, основанных на увеличении гиперпараметров, таких как размер батча, количество ядер свертки и разрешение входных изображений. Данный шаг должен обязательно увеличить значение метрики, хоть и увеличит время обучения и требуемые вычислительные мощности. Также, в условиях эскалации

гиперпараметров, нашу модель можно будет сравнить с иными моделями, которые не удалось включить в сравнение из — за высоких системных требований самих архитектур во время их обучения.

Список литературы

1. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998.
2. the 2st-unet for pneumothorax segmentation in chest x-rays using resnet34 as a backbone for u-net. arXiv:2009.02805v1 [eess.IV] 6Sep 2020
3. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597v1 [cs.CV] 18 May 2015
4. Colorectal Cancer Segmentation using AtrousConvolution and Residual Enhanced UNet. arXiv:2103.09289v1 [eess.IV] 16 Mar 2021
5. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. arXiv:1606.04797v1 [cs.CV] 15 Jun 2016
6. Глубокое обучение / Ян Гудфеллоу, Йошуа Бенджио, Аарон Курвилль // ДМК Пресс, 2018г., второе цветное издание, исправленное.
7. SegNet: A Deep ConvolutionalEncoder-Decoder Architecture for ImageSegmentationVijay Badrinarayanan, Alex Kendall, Roberto Cipolla,Senior Member, IEEE. arXiv:1511.00561v3 [cs.CV] 10 Oct 2016
8. Fully Convolutional Networksfor Semantic Segmentation. arXiv:1605.06211v1 [cs.CV] 20 May 2016
9. Why better weight initialization is important in neural networks? <https://towardsdatascience.com/why-better-weight-initialization-is-important-in-neural-networks-ff9acf01026d>
10. Kaggle Airbus Ship Detection Challenge - <https://www.kaggle.com/c/airbus-ship-detection/data>
11. Глубокое обучение. / Николенко С., Кадури А., Архангельская Е. // СПб: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).
12. Very deep convolutional networks for large-scale image recognition / Karen Simonyan, Andrew Zisserman

Приложения

Приложение 1

Программная реализация исследования гипотез для задачи бинарной классификации

Оптимизация RMSProp

```
from google.colab import drive
drive.mount('/content/drive')

import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42

data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu

print("array sizes of data array: ", data.shape)
print("array sizes of target array: ", data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])

#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets

#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data, targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape

#Data augmentation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()

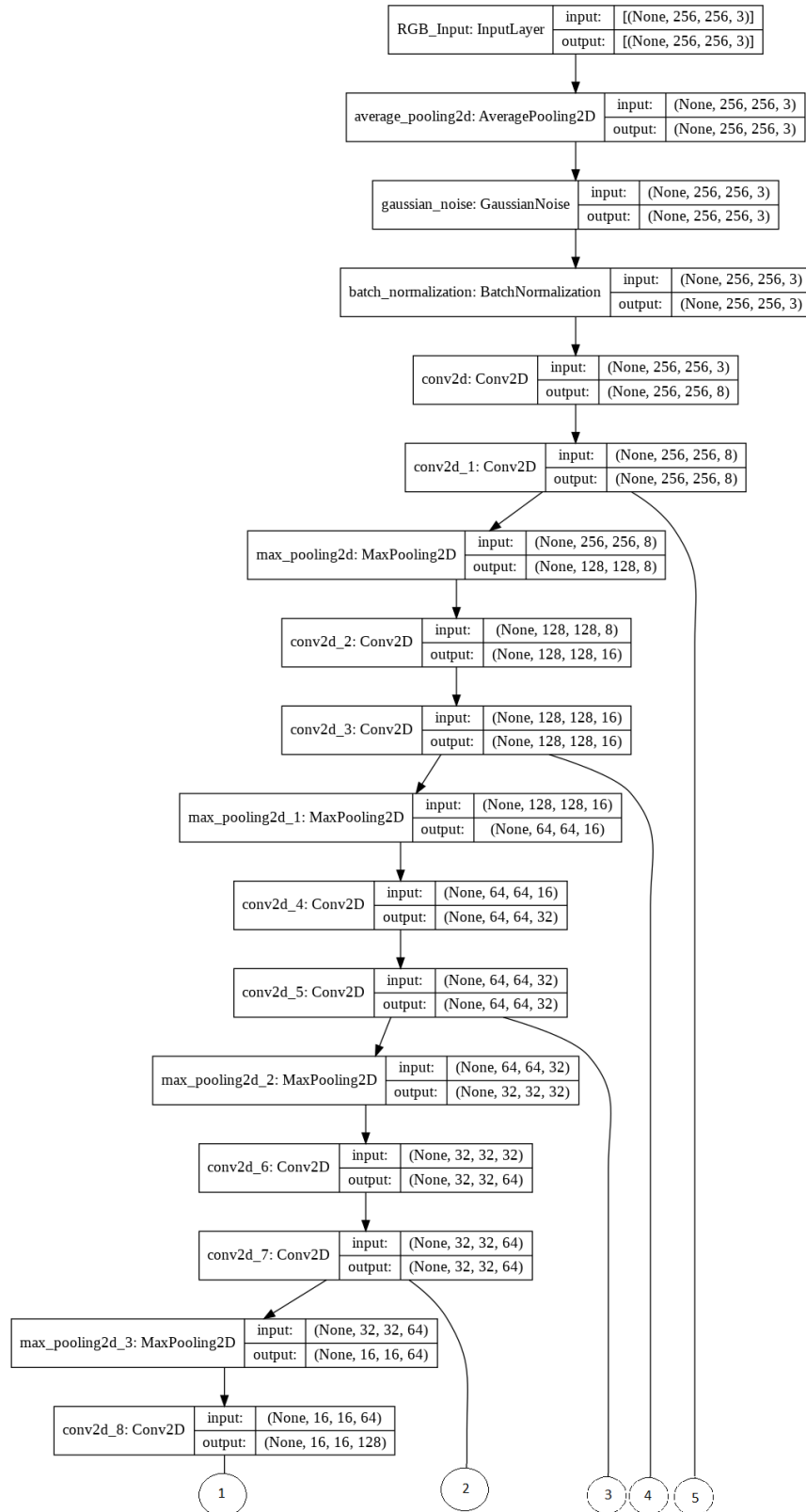
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproo
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape

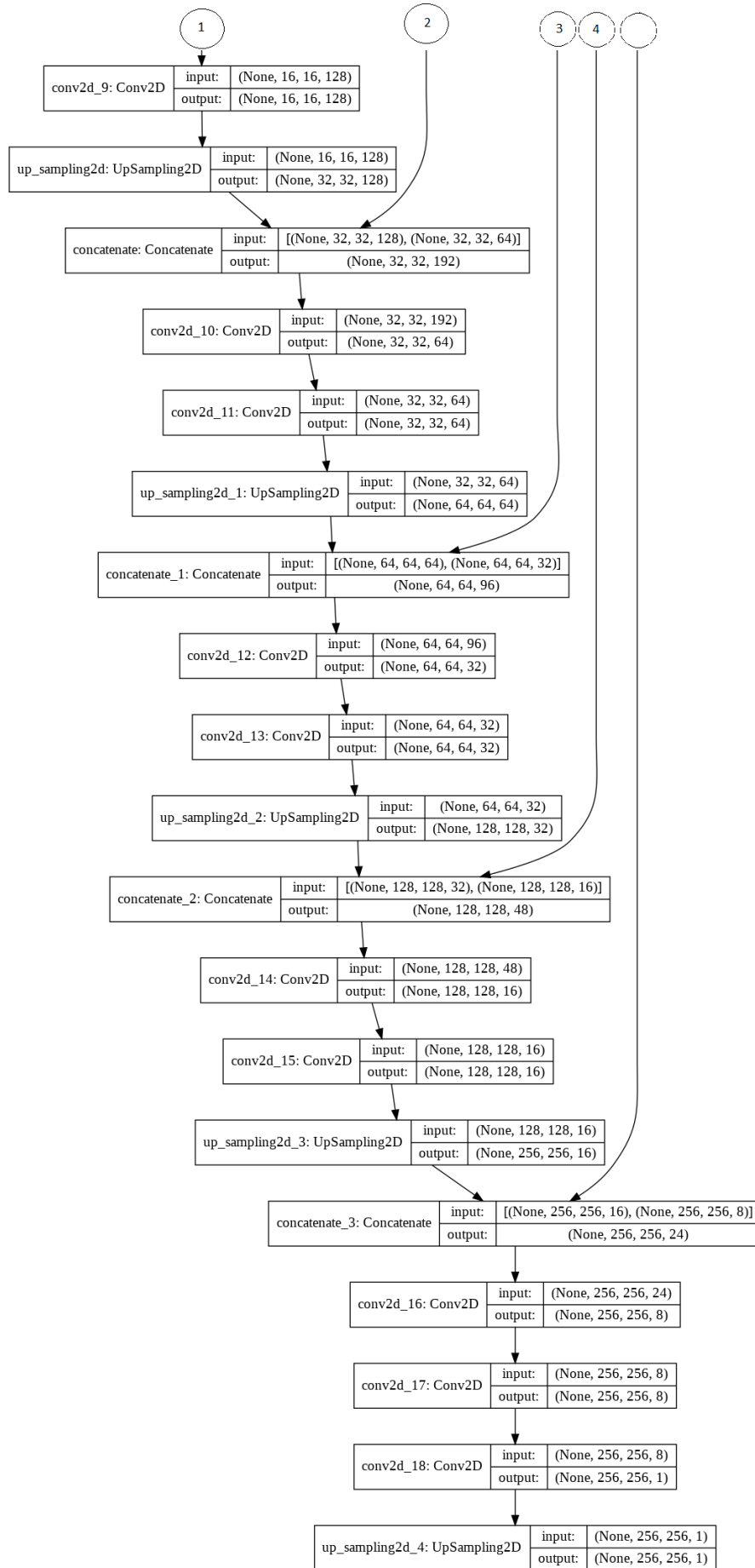
gc.collect()

#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint
```

Приложение 2

Изображение структуры сверточной нейронной сети для задачи семантической сегментации





Приложение 3

Программная реализация задачи семантической сегментации

Параметры модели

```
BATCH_SIZE = 64
EDGE_CROP = 16
GAUSSIAN_NOISE = 0.1
UPSAMPLE_MODE = "SIMPLE"

#Понижение дискретизации внутри сети
NET_SCALING = (1, 1)

#Понижение размерности в предобработке
IMG_SCALING = (3, 3)

#Число изображений для валидации
VALID_IMG_COUNT = 900

#Максимальное число шагов за эпоху при обучении
MAX_TRAIN_STEPS = 9
MAX_TRAIN_EPOCHS = 50
AUGMENT_BRIGHTNESS = False

SEED = 42
```

```
from skimage.util import montage
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from skimage.io import imread
import matplotlib.pyplot as plt
from matplotlib.cm import get_cmap
from skimage.segmentation import mark_boundaries
from sklearn.model_selection import train_test_split
import keras.backend as K

from keras.preprocessing.image import ImageDataGenerator
from keras import models, layers
import keras.backend as K
from keras.optimizers import Adam
from keras.losses import binary_crossentropy
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping
from tensorflow.keras.optimizers import RMSprop, Adam

from skimage.morphology import binary_opening, disk, label
import gc; gc.enable()

from PIL import Image

montage_rgb = lambda x: np.stack([montage(x[:, :, :, i]) for i in range(3)])
ship_dir = "/content/drive/MyDrive/Диплом/Ship_detection/Input/"
train_image_dir = os.path.join(ship_dir, 'train')
```

Определим вспомогательные процедуры для декодирования, кодирования и вывода изображения и маски корабля