DOI: https://doi.org/10.30898/1684-1719.2025.6.2
УДК: 004.93:004.032.26

# ОЦЕНКА ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ НЕКОТОРЫХ SOTA МОДЕЛЕЙ НЕЙРОННЫХ СЕТЕЙ ПРИ РЕШЕНИИ ЗАДАЧ КЛАССИФИКАЦИИ КВАЗИСТАЦИОНАРНЫХ ОБЪЕКТОВ В ВИДИМОМ ДИАПАЗОНЕ ДЛИН ВОЛН

**М.Л. Белокопытов**

**Военно-космическая академия имени А.Ф. Можайского,**
**197198, Санкт-Петербург, ул. Ждановская, 13**

Статья поступила в редакцию 25 февраля 2025 г.

**Аннотация.** В работе представлен краткий обзор современных нейросетевых подходов к обнаружению квазистационарных объектов на оптических изображениях. Рассмотрены особенности построения, архитектура и работа пяти *SOTA* моделей нейронных сетей. Осуществлена подготовка набора данных (парсинг изображений и их разметка, разделение на обучающую, валидационную и тестовую выборки). На подготовленном наборе данных произведено обучение и тестирование таких моделей нейронных сетей, как *Faster R-CNN*, *DETR*, *RetinaNet*, *SSD*, *YOLOv8m*. Выполнен сравнительный анализ обученных нейросетей и оценена эффективность работы алгоритмов распознавания на основе точности работы моделей и их быстродействия. Сделан вывод о возможности использования подобных нейросетевых обнаружителей при решении задачи автоматизированной обработки фоноцелевой информации.

**Ключевые слова:** нейронная сеть, набор данных, выборка, обучение, метрика, эффективность.

**Автор для переписки:** Белокопытов Марк Львович, hommer1990@mail.ru

## Введение

Нейронные сети (НС) – мощный инструмент обработки и анализа данных, который позволяет решать задачи кластеризации и классификации, прогнозирования, генерации изображений и текста и многие другие. Существует множество моделей НС, которые созданы специально под определённый класс задач: свёрточные нейронные сети – для классификации изображений, рекуррентные – для задач с временной характеристикой, например, анализ непрерывного текста или звука.

В настоящей работе авторами предлагается к рассмотрению пять *SOTA* (*State-of-the-Art*) моделей нейронных сетей (*Faster R-CNN, DETR, RetinaNet, SSD, YOLOv8m*), работоспособность которых будет оценена при решении задачи классификации квазистационарных объектов в видимом диапазоне длин волн.

Обучение НС, а также оценка эффективности их работы будет осуществляться с использованием набора данных, который был сформирован авторами специально для этих целей.

## 1. Набор данных для обучения

Для того чтобы построить качественный классификатор (обнаружитель), необходимо иметь качественные данные. Никакой из методов построения классификаторов никогда не приведет к нужному качеству модели, если имеющийся набор данных не будет достаточно полным и представительным для той задачи, с которой придется работать этой модели в последующем.

При подготовке подобного рода наборов данных чаще всего оперируют следующими терминами, которые в последующем будут использоваться в настоящей работе:

1) ограничивающая рамка (*bounding box*) – координаты, ограничивающие определенную область изображения, – чаще всего в форме прямоугольника. Ограничивающая рамка может быть представлена четырмя координатами в двух форматах: центрированный $(c_x, c_y, w, h)$ и обычный $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$;

2) гипотеза (*Proposal*) – определенный регион изображения (заданный с помощью ограничивающей рамки), в котором предположительно находится объект интереса;

3) *end-to-end* обучение – обучение, при котором на вход нейронной сети поступают «сырые» изображения, а на выходе получаются готовые ответы.

Для подготовки качественного набора данных, который в последующем будет использоваться для разработки и тестирования моделей НС, авторами был осуществлен парсинг изображений видимого диапазона из открытых источников информации. В качестве квазистационарных объектов классификации использовались морские суда.

Набор данных состоит из 5628 размеченных изображений. Разметка осуществлялась силами авторов работы. Помимо самих изображений в формате «*.jpg*», в датасете содержится по два файла аннотации для каждого изображения в форматах «*.txt*» и «*.xml*».

Файлы в формате «*.txt*» необходимы для обучения НС *YOLO*, каждый из файлов содержит информацию о разметке одного изображения и представляет собой пять чисел. Первое число обозначает номер класса, остальные числа описывают ограничивающую рамку: *x_center* и *y_center* – нормализованные координаты центра ограничивающего прямоугольника, *width* и *height* – ширина и высота ограничивающего прямоугольника в нормализованном формате.

В файлах «*.xml*» содержится разметка изображения в формате *PascalVOC*. Этот файл содержит имя класса в теге «*name*», и координаты ограничивающей рамки в тегах «*x_min*», «*y_min*», «*x_max*», «*y_max*». Здесь «*x_min*» и «*y_min*» – координаты верхнего левого угла ограничивающего прямоугольника, а «*x_max*», «*y_max*» – координаты правого нижнего угла ограничивающего прямоугольника. Этот формат аннотации изображений нужен для обучения НС *Faster R-CNN, DETR, RetinaNet, MobileNet*.

На изображениях представлены 13 классов морских судов: *tanker* (танкер), *suhogruz* (сухогруз), *lng_tanker* (газовоз), *lainer* (лайнер), *containerovoz* (контейнеровоз), *parom* (паром), *maloe_grajdanskoe_sudno* (малое гражданское

судно), *rybolovetskoe* (рыболовецкое судно), *avianosets* (авианосец), *kreyser* (крейсер), *esminets* (эсминец), *storojevoi* (сторожевой корабль), *others* (другие).

Данный набор данных был разбит на три выборки: обучающая (4337 изображений), валидационная (682 изображений), тестовая (609 изображений). Пример изображений из набора данных представлен на рисунке 1.

Рис. 1. Пример изображений из набора данных.

## 2. Обучение моделей нейронных сетей

*SOTA* – это модели глубокого обучения с современной архитектурой, эффективность работы которых уже доказана [1]. Как правило, *SOTA*-модели являются самыми современными методами машинного обучения. Ряд готовых *SOTA*-моделей находятся в открытом доступе на платформе *Hugging Face*. На данной платформе можно найти модели для извлечения признаков и классификации изображений, обработки естественного языка и др.

В настоящей работе авторами были использованы пять *SOTA* моделей нейронных сетей: *Faster R-CNN*, *DETR*, *RetinaNet*, *SSD*, *YOLOv*8*m*. Эффективность этих моделей уже не раз была подтверждена на практике при решении задач классификации.

Для сравнения и последующего анализа работы НС будут в последующем использоваться следующие метрики: *Precision* (точность), *Recall* (полнота), *F*1-*score* (*F*-метрика), *mAP*50 (средние значения точности), *mAP*50-95 (средние значения точности на указанном интервале). Введём ряд важных определений для описания этих метрик в терминах ошибок классификации.

*Precision* – мера того, сколько из сделанных положительных прогнозов являются верными, т.е. точность системы в пределах класса – это доля объектов,

действительно принадлежащих к этому классу, относительно всех объектов, которые система отнесла к этому классу. Значение метрики *Precision* рассчитывается в соответствии с выражением (1):

$$precision = \frac{TP}{TP + FP}, \quad (1)$$

где *TP* (*True Positive*) – правильное определение класса, а *FP* (*False Positive*) – ошибка 1-го рода.

*Recall* – мера показывающая, долю найденных классификатором объектов, принадлежащих к классу, относительно всех объектов этого класса в выборке. Значение метрики *Recall* рассчитывается в соответствии с выражением (2):

$$recall = \frac{TP}{TP + FN}, \quad (2)$$

где *FN* (*False Negative*) – ошибка 2-го рода.

*F*1-*score* или *F*-метрика – метрика, позволяющая объединить *Precision* и *Recall* в агрегированный критерий качества, её можно интерпретировать как среднее гармоническое значение *Precision* и *Recall*. Она достигает максимума в случае, когда *Precision* и *Recall* равны единице, и близка к нулю, если один из аргументов близок к нулю. *F*-метрика рассчитывается в соответствии с выражением (3):

$$F_1 score = \frac{2\,precision * recall}{precision + recall}. \quad (3)$$

Для наглядности качества работы модели НС значения *Precision* и *Recall* при различных пороговых значениях изображают в виде кривой – *Precision-Recall curve*. При этом под порогом (*threshold*) понимается величина, обычно находящаяся в пределах от 0 до 1. Благодаря порогу определяется правильность работы НС. Если модель выдаёт некоторый ответ с вероятностью *P*, то этот ответ считается правильным, если ответ верный и $P \geq threshold$ [2].

Изображения *Precision-Recall curve* для каждого из классов приведены на рисунке 2.

*Average Precision* (*AP*) – значение площади под кривой *Precision-Recall curve*, которое предоставляет единственное значение, инкапсулирующее значения *Precision* и *Recall*.

*mAP* (*Mean Average Precision*) расширяет концепцию *AP*, вычисляя средние значения *AP* для нескольких классов объектов. Данный показатель применяется в сценариях обнаружения многоклассовых объектов с целью обеспечения всесторонней оценки производительности модели.

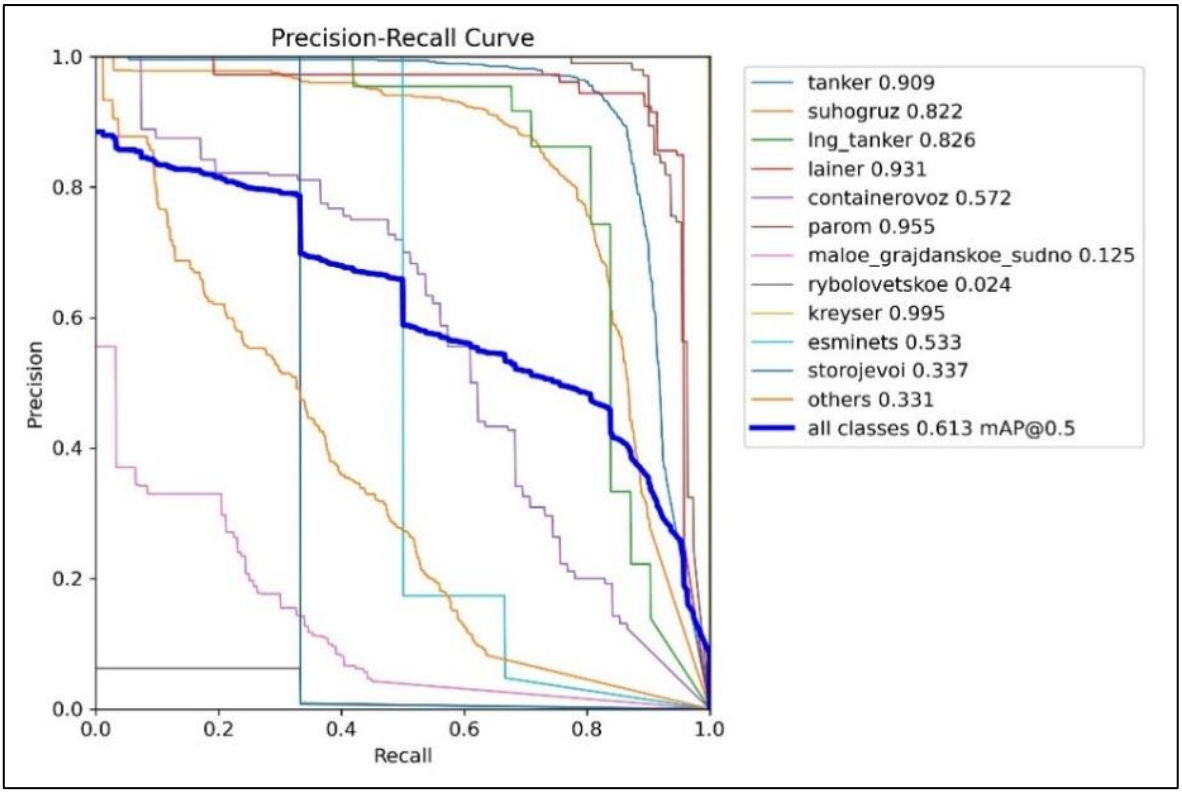


Рис. 2. *Precision-Recall curve* для каждого из классов.

*mAP*50 – значение *mAP*, вычисленное при пороге, равном 0,5. Это показатель точности модели, учитывающий только «простые» обнаружения.

*mAP*50-95 – среднее значений *mAP*, вычисленное при различных пороговых значениях, варьирующихся от 0,5 до 0,95 с шагом 0,05. Данный показатель даёт полное представление о работе модели при различных уровнях сложности обнаружения.

## 2.1 *Faster R-CNN*

*R-CNN* – свёрточная нейронная сеть, в которой используется предварительная информация о предполагаемом местоположении обнаруживаемых объектов. Возможные области обнаружения объектов обычно получают посредством алгоритма так называемого селективного поиска. Как правило, селективный поиск выдает около 2000 предложений об областях изображения, которые могут представлять интерес. Для селективного поиска перспективных областей, которые могут содержать объекты, обычно используют традиционные способы обработки изображений.

Вышеупомянутые области, представляющие интерес, передаются классифицирующей и локализующей сети для предсказания классов объектов и ассоциированных с ними ограничительных прямоугольников [3]. Классифицирующая сеть представляет собой свёрточную нейронную сеть, после которой применяется метод опорных векторов (*SVM*) для окончательной классификации. Высокоуровневая архитектура *R-CNN* представлена на рисунке 3.

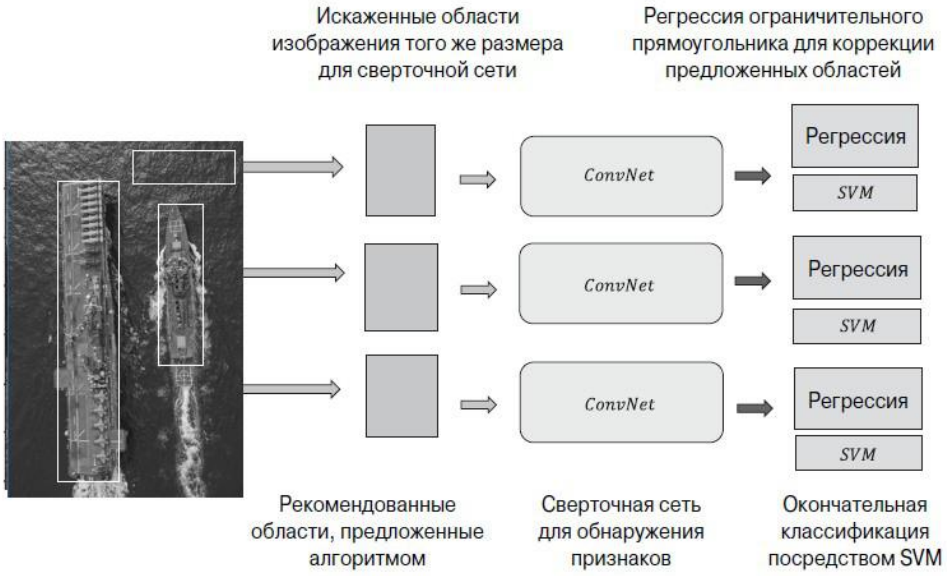


Рис. 3. Архитектура нейросети *R-CNN*.

По сравнению с *R-CNN*, *Faster R-CNN*, строит сеть, состоящую только из одного этапа. Данная НС разделяет вычисления на свёрточных слоях путём

использования нового слоя *ROI*-пулинга, который делает *Faster R-CNN* быстрее, чем *R-CNN*.

В программном виде модель НС *Faster R-CNN* была реализована на языке программирования *Python* 3.11 с помощью фреймворка для машинного обучения *PyTorch*.

В программный код архитектура модели была загружена с помощью пакета *torchvision.models.detection.faster_rcnn* и модуля *FastRCNNPredictor*. Далее была загружена предобученная модель *Faster R-CNN* и изменена головная часть детектора в соответствии с количеством классов из набора данных для обучения.

Параметры для обучения этой модели НС были выставлены следующим образом: размер батча – 8; число эпох – 100; изменение размера изображения для обучения и аугментаций – 512 пкс; алгоритм оптимизации – стохастический градиентный спуск; коэффициент скорости обучения – 0.001; импульс – 0.9; коэффициент увядания весов – 0.0005.

По окончании процесса обучения был построен график падения функции потерь, который показан на рисунке 4. Анализируя его, можно сделать вывод о том, что после 21 эпохи обучения НС начинает переобучаться, т.е. функция ошибки на обучающем наборе данных продолжает уменьшаться, а на валидационной выборке значения этой функции понемногу увеличиваются.

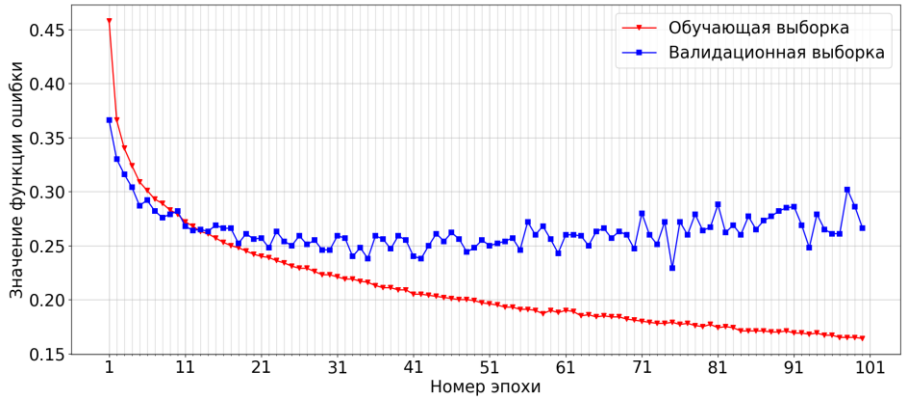


Рис. 4. График поведения функции потерь НС Faster R-CNN на обучающей и валидационной выборках.

На рисунке 5 показано как ведут себя метрики в зависимости от количества эпох обучения. Анализируя его, можно сделать вывод о том, что после примерно 40 эпох существенного увеличения показателей метрик не наблюдается, а значения *Precision* колеблются от 0,4 до 0,65 с самого начала обучения. Все остальные метрики плавно растут примерно до 40 эпохи, далее выходят на плато.

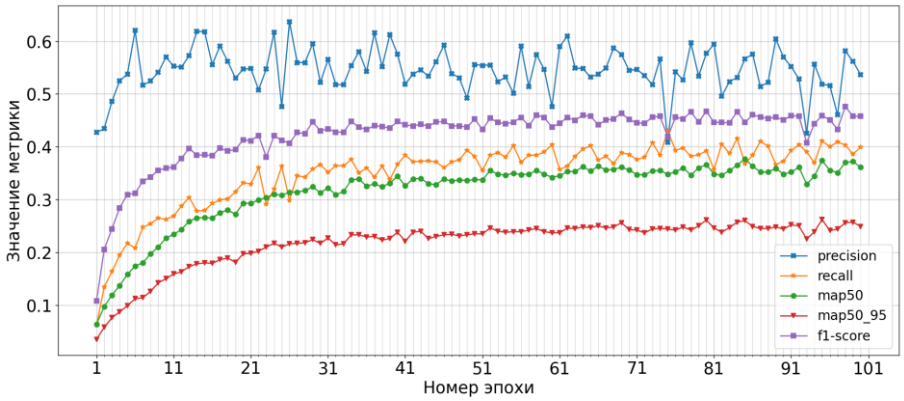


Рис. 5. Зависимость метрик от количества эпох обучения *Faster R-CNN*.

Пример работы нейронной сети *Faster R-CNN* на валидационном наборе данных представлен на рисунке 6.

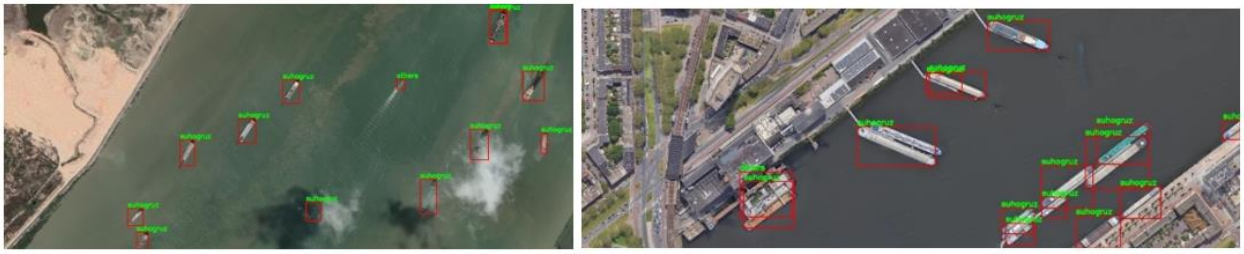

Рис. 6. Пример работы нейронной сети *Faster R-CNN* на валидационном наборе данных.

## 2.2 *DETR*

*DETR* или *Detection Transformer* – это модель глубокого обучения для обнаружения объектов, использующая архитектуру *Transformer* в качестве своего основного компонента [4]. Данная модель известна своим механизмом самоконтроля, который позволяет ей фиксировать сложные взаимосвязи между

элементами в последовательности или наборе данных. Общая архитектура *DETR* приведена на рисунке 7.

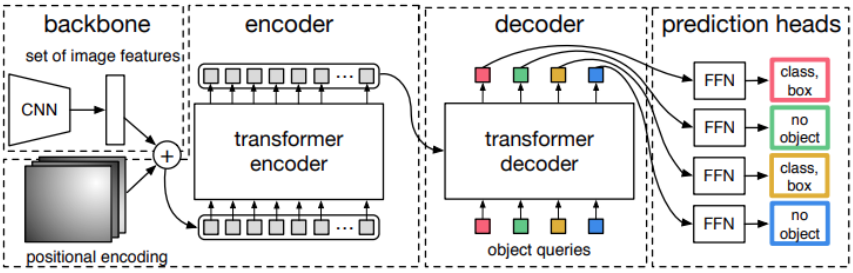


Рис. 7. Архитектура нейросети *DETR*.

В программном виде данная НС была также реализована с помощью библиотеки для машинного обучения *PyTorch*. В самом коде класс, описывающий *DETR*, наследует класс *nn.Module*. Для реализации алгоритма были загружены веса предобученной модели и модифицирован выходной слой в соответствии числу классов.

Параметры для обучения были выставлены следующим образом: размер батча – 16; число эпох – 100; вычислительное устройство для обучения – *cuda* 11.8; изменение размера изображения для обучения и аугментаций – 640 пкс; алгоритм оптимизации – *ADAMW*; коэффициент скорости обучения – 0.00005; коэффициент увядания весов – 0.0001.

На графике поведения функции потерь на протяжении всего процесса обучения (рисунок 8) видно, что значения функции потерь на валидационной и на обучающей выборках уменьшаются вплоть до 78 эпохи, далее на обучающей выборке значения этой функции продолжают уменьшаться, а на валидационной выборке перестают существенно меняться. Следовательно, модель начинает переобучаться.

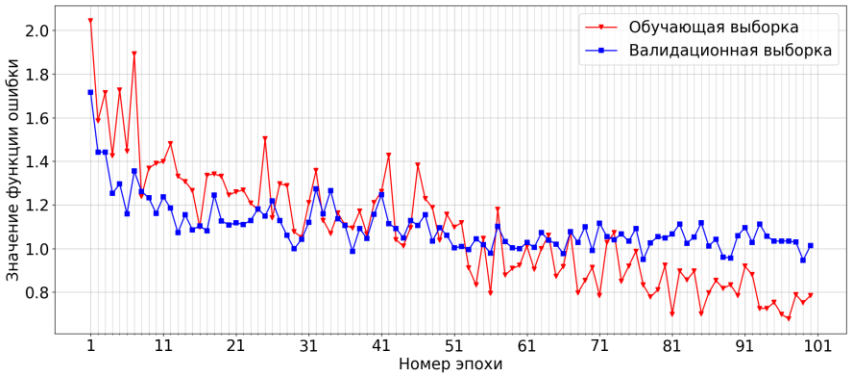


Рис. 8. График поведения функции потерь НС *DETR* на обучающей и валидационной выборках.

На рисунке 9 показано поведение метрик во время процесса обучения. Все, без исключения, метрики показывают стабильный рост значений с ростом числа эпох обучения.

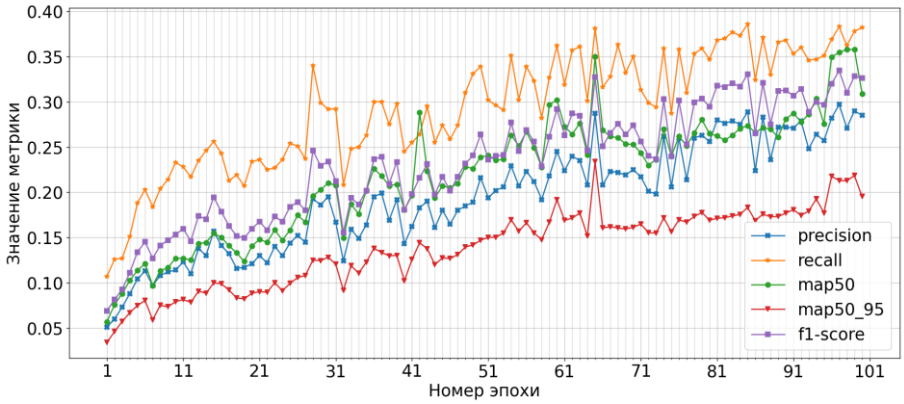


Рис. 9. Зависимость метрик от количества эпох обучения *DETR*.

Пример работы нейронной сети *DETR* на валидационном наборе данных представлен на рисунке 10.


Рис. 10. Пример работы нейронной сети *DETR* на валидационном наборе данных.

## 2.3 *RetinaNet*

Архитектура данной свёрточной нейронной сети (СНС) состоит из 4 отдельных сетей, каждая из которых имеет своё назначение.

*Backbone* – основная (базовая) сеть, служащая для извлечения признаков из поступающего на вход изображения. Она является вариативной и в её основу могут входить классификационные нейросети, такие как *ResNet, VGG, EfficientNet* и другие. В данной работе используется *ResNet*50. Архитектура *RetinaNet* с *backbone*-сетью *ResNet* показана на рисунке 11.

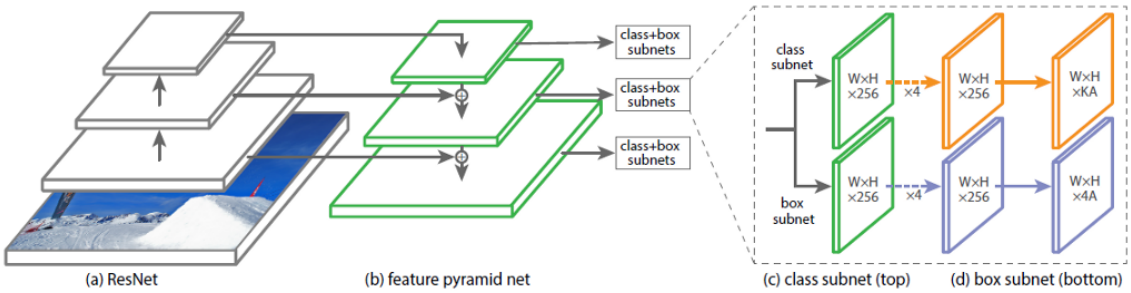


Рис. 11. Архитектура *RetinaNet* с *backbone*-сетью *ResNet*.

*Feature Pyramid Net* (*FPN*) – свёрточная нейронная сеть, построенная в виде пирамиды, служащая для объединения достоинств карт признаков нижних и верхних уровней сети [5].

Третей частью архитектуры *RetinaNet* являются две подсети: классификационная и регрессионная. Каждая из этих подсетей образует на выходе ответ о классе объекта и его расположении на изображении.

В последней, четвёртой сети каждая карта признаков преобразуется в набор векторов. Регрессионная модель на выходе имеет для каждой якорной рамки вектор из 4 значений, указывающих смещение целевой рамки (относительно якорной. Классификационная модель имеет на выходе для каждой якорной рамки *one-hot* вектор длиной *K*, в котором индекс со значением 1 соответствует номеру класса, который НС присвоила объекту.

В программном виде данная нейронная сеть была реализована с помощью библиотеки для машинного обучения *PyTorch*. Из пакета *torchvision.models.detection.retinanet_resnet*50*_fpn_v*2 была загружена предобученная

НС, в которой в дальнейшем был модифицирован выходной слой в соответствии с числом классов.

Параметры для обучения были выставлены следующим образом: размер батча – 8; число эпох – 100; изменение размера изображения для обучения и аугментаций – 512 пкс; алгоритм оптимизации – стохастический градиентный спуск; коэффициент скорости обучения – 0.001; импульс – 0.9; коэффициент увядания весов – 0.0005.

На рисунке 12 изображено поведение функции потерь на обучающей и валидационной выборке в зависимости от количества прошедших эпох обучения. Из рисунка видно, что после 21 эпохи модель начинает переобучаться, поскольку значения функции ошибок на обучающей и валидационной выборках начинают расходиться (на обучающей выборке значения функции падают, а на валидационной увеличиваются).

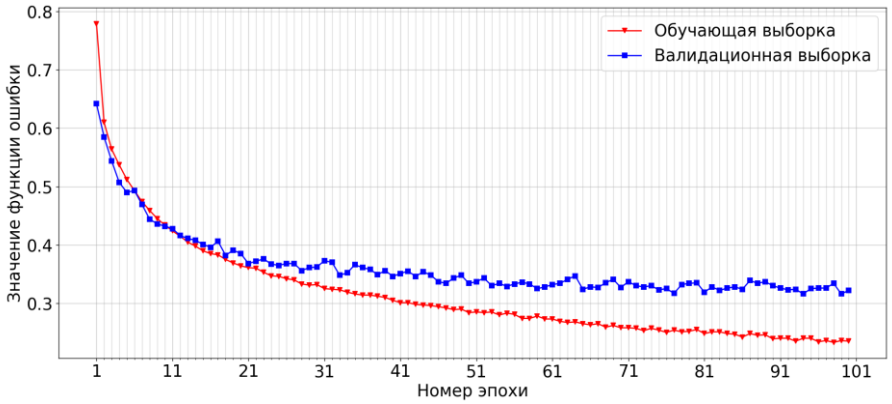


Рис. 12. График поведения функции потерь НС *RetinaNet* на обучающей и валидационной выборках.

Из рисунка 13, на котором изображено поведение значений метрик в зависимости от числа эпох обучения следует, что все метрики, кроме *precision* стабильно растут. Немедленная динамика этого роста очевидна вплоть до 30 эпохи, далее рост метрик замедляется, но не прекращается. Метрика *precision* заметно растёт до 20 эпохи, далее значимо не меняется.

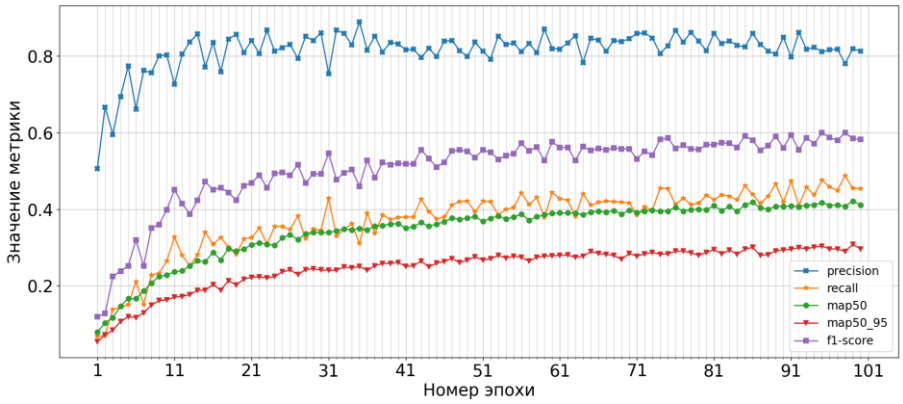


Рис. 13. Зависимость метрик от количества эпох обучения *RetinaNet*.

Пример работы нейронной сети *RetinaNet* на валидационном наборе данных представлен на рисунке 14.

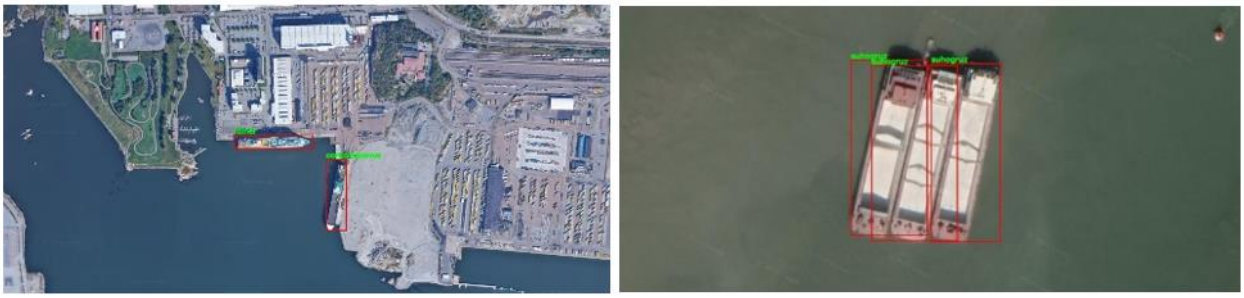

Рис. 14. Пример работы нейронной сети *RetinaNet* на валидационном наборе данных.

## 2.4 *SSD*

Работа *SSD* основана на свёрточной сети прямого распространения, которая создает конечный набор ограничивающих прямоугольных рамок и количественные оценки присутствия в этих рамках объектов различных классов, после чего производится подавление немаксимумов для получения окончательных предсказаний [6]. Архитектура *SSD* приведена на рисунке 15.

Отличительной особенностью данной НС является возможность дискретизировать множество различных форм результирующих ограничивающих рамок, что положительно сказывается на скорости работы сети.

Для программной реализации этой модели использовался пакет *torchvision*, из которого были извлечены модели *SSD*, *DefaultBoxGenerator*, *SSDHead*. Для последующей реализации НС объявляется функция, возвращающая

готовую модель в зависимости от переданного ей на вход числа классов. Внутри этой функции инициализируется базовая модель посредством нейронной сети *resnet*34 с весами по умолчанию из пакета *torchvision.models*, для того, чтобы создать слои базовой модели самой *SSD*. После чего с помощью *DefaultBoxGenerator* создаются якорные рамки. Создание головки нейронной сети *SSD* выполнено с помощью функции *SSDHead* и переданными ей аргументами. Таким образом была инициализирована конечная архитектура *SSD*.

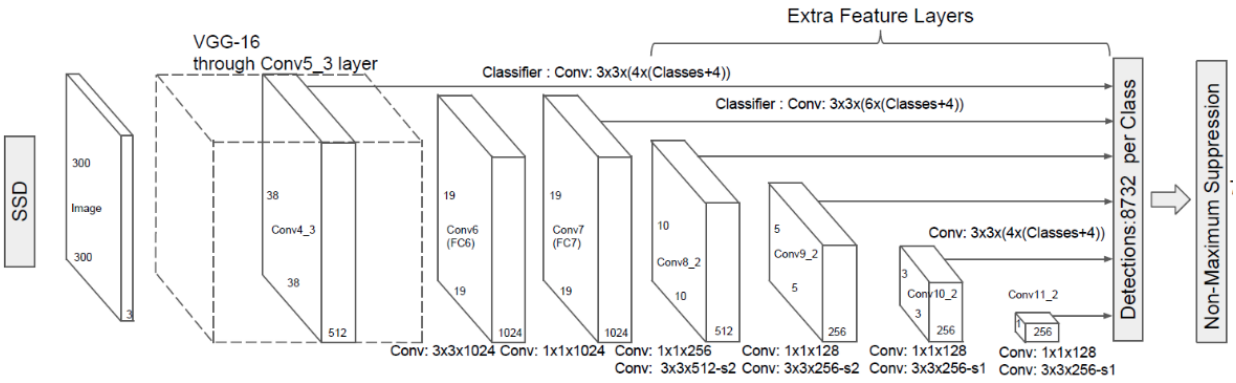


Рис. 15. Архитектура НС *SSD*.

Параметры для обучения аналогичны параметрам для НС *RetinaNet*, исключение составили размер батча – 16 и изменение размера изображения для обучения и аугментаций – 640 пкс.

На рисунке 16 приведено поведение функции ошибок на обучающей и валидационной выборках в зависимости от количества пройденных эпох обучения. Анализируя поведение функции потерь, можно сделать вывод, что данная НС обладает слабой способностью к обобщению данных, поскольку после 5 эпохи значения функции ошибок на обучающей и валидационной выборках начинают сильно расходиться.

На рисунке 17, показывающем зависимость значений метрик от количества пройденных эпох обучения наглядно демонстрируется плавный рост метрик: *recall*, *f*1-*score*, *mAP*50, *mAP*50-95 вплоть до 40 эпохи, далее рост метрик не значительный. Неочевидно здесь поведение метрики *precision* – её значения

уже на второй эпохе очень высоки, однако с увеличением числа эпох обучения значения этой метрики уменьшаются.

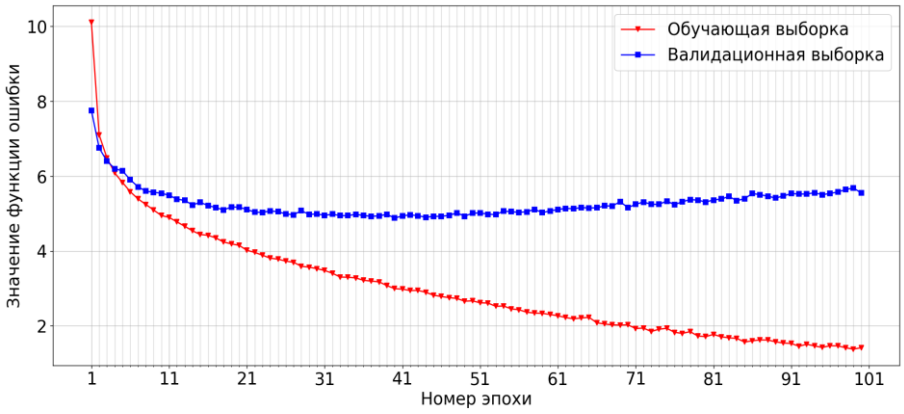


Рис. 16. График поведения функции потерь НС *SSD* на обучающей и валидационной выборках.

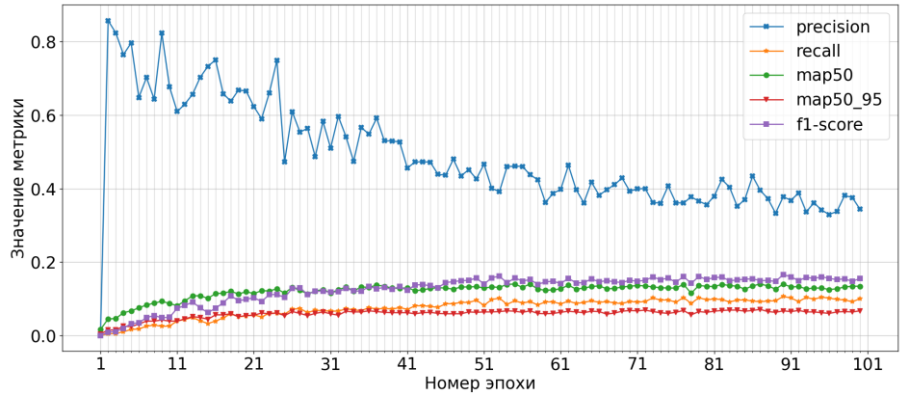


Рис. 17. Зависимость метрик от количества эпох обучения НС *SSD*.

Пример работы нейронной сети *SSD* на валидационном наборе данных представлен на рисунке 18.

Рис. 18. Пример работы нейронной сети *SSD* на валидационном наборе данных.

## 2.5 *YOLOv8m*

*YOLO* – архитектура СНС, предназначенная для детектирования объектов на изображениях. Метод *YOLO* делит изображение на *N* сеток, каждая из которых имеет сектор одинакового размера *S*x*S*. Каждая из этих *N* сеток отвечает за обнаружение и определение местоположения объекта, который она содержит (рисунок 19).

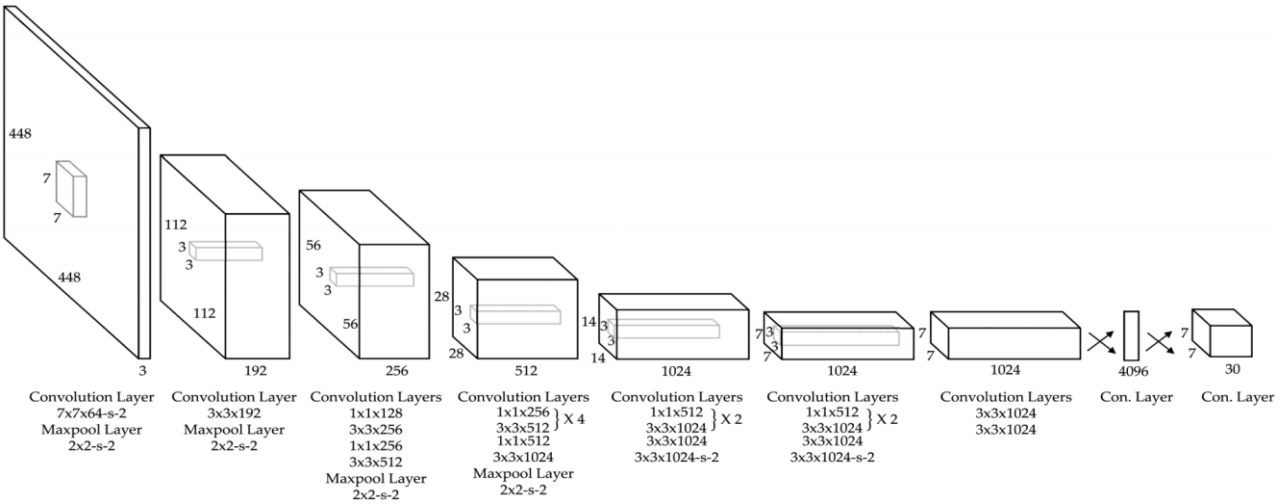


Рис. 19. Архитектура нейросети *YOLO*.

Исходное изображение сжимается таким образом, чтобы получить квадратную матрицу, в каждой клетке которой записана информация о наличии объекта и его классе на соответствующей части картинки. Таким образом, *YOLO* просматривает картинку один раз, что существенно увеличивает скорость обработки. Эти сети, в свою очередь, прогнозируют координаты ограничивающей рамки относительно координат ячейки, а также имя элемента и вероятность присутствия объекта в ячейке. Из-за того, что многие ячейки предсказывают один и тот же элемент с различными ограничивающими рамками, этот метод значительно сокращает вычисления, поскольку и обнаружение, и распознавание обрабатываются ячейками из изображения.

Основное отличие *YOLO* от других алгоритмов, используемых для обнаружения объектов, заключается в том, что она очень быстро опознает объекты в режиме реального времени.

*YOLOv*8 – это семейство моделей обнаружения объектов на базе *YOLO* от *Ultralytics* [7], обеспечивающих самые современные характеристики. По сравнению с предыдущими версиями *YOLO*, модель *YOLOv*8*m* работает быстрее и точнее, обеспечивая при этом единую структуру для обучения моделей и выполнения задач обнаружение, сегментация и классификации.

В программном виде данная нейронная сеть *YOLOv*8*m* была реализована на языке программирования *Python* 3.11 с помощью библиотеки *ultralytics*. Обучение производилось на протяжении 100 эпох, с размером батча равным 8, и изменением размеров входного изображения до 640 пикселей в длину и ширину.

На рисунке 20, который показывает поведение функции ошибок на обучающей и валидационной выборках в зависимости от числа эпох обучения, можно увидеть уменьшение значений этой функции на обоих выборках до 48 эпохи. Далее с увеличением числа эпох значение функции на обучающей выборке продолжает уменьшаться, а на валидационной выборке также уменьшается, но с более медленным темпом.

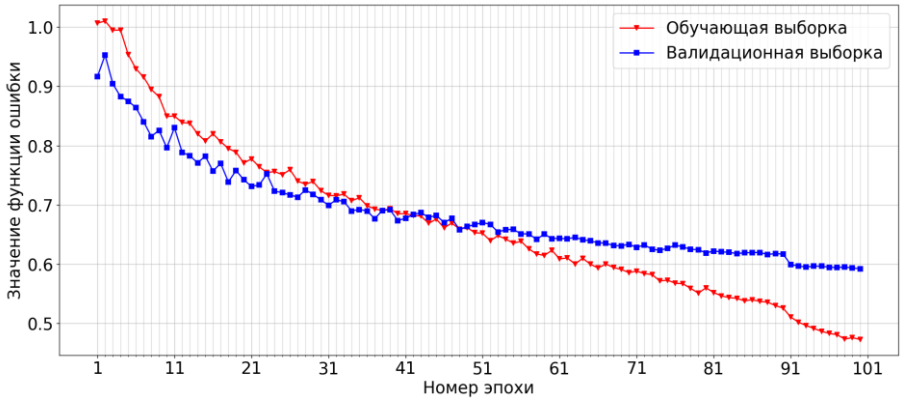


Рис. 20. График поведения функции потерь
на обучающей и валидационной выборках при обучении *YOLOv*8*m*.

На рисунке 21 показана зависимость значений метрик от числа пройденных эпох обучения. Все метрики показывают заметный рост своих значений вплоть до 48 эпохи. Далее, с увеличением числа эпох продолжается незначительный рост, после 90 эпохи значения метрик перестают значимо меняться.

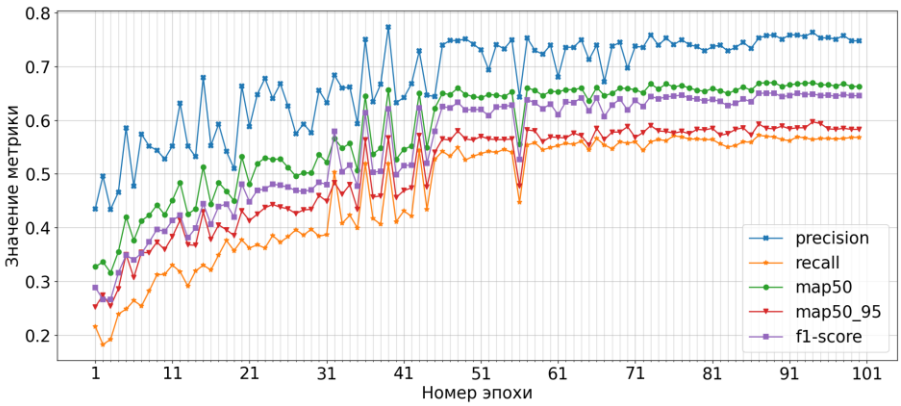


Рис. 21. Зависимость метрик от количества эпох обучения *YOLOv8m*.

Пример работы нейронной сети YOLOv8m на валидационном наборе данных представлен на рисунке 22.


Рис. 22. Пример работы нейронной сети *YOLOv8m* на валидационном наборе данных.

Обучение всех нейронных сетей, приведённых в этом разделе производилось на ЭВМ со следующими характеристиками:

- процессор: *Intel Core i*7-8700*K*;
- видеокарта: *NVIDIA Quadro P*5000;
- объём оперативной памяти: 64 ГБ;
- вычислительное устройство для обучения: *cuda* 11.8.

## 3. Оценка эффективности работы нейронных сетей

После обучения моделей для оценки эффективности работы каждой из рассмотренных нейронных сетей был произведён расчет ранее перечисленных в работе метрик (*Precision*, *Recall*, *F*1-*score*, *mAP*50, *mAP*50-95).

Для итогового сравнения эффективности нейронных сетей между собой были выбраны по одной модели каждой сети, которые имеют наибольшее

значение метрики *mAP*50-95 и обученные на определённом количестве эпох. Для *Faster R-CNN* это модель, обученная на 95 эпохах, для *DETR* – на 96 эпохах, для *Retina Net* – на 99 эпохах, для *SSD* – на 87 эпохах, для *YOLOv8m* – на 94 эпохах.

Выбор метрики *mAP*50-95 обусловлен тем, что она рассчитывается как средние значения *mAP*, вычисленные при различных пороговых значениях функции обнаружения потерь (*IoU*), варьирующихся от 0,50 до 0,95 с шагом 0,05. Благодаря чему эта метрика содержит в себе более полное представление о работе модели при различных уровнях сложности обнаружения [8].

Итоговое сравнение НС по каждой метрике приведено на рисунке 23. Основываясь на этой гистограмме нейронные сети в совокупности по всем метрикам в порядке от лучшей к худшей можно расположить в следующем образом: *YOLOv8m*, *Retina Net*, *Faster R-CNN*, *DETR*, *SSD*.

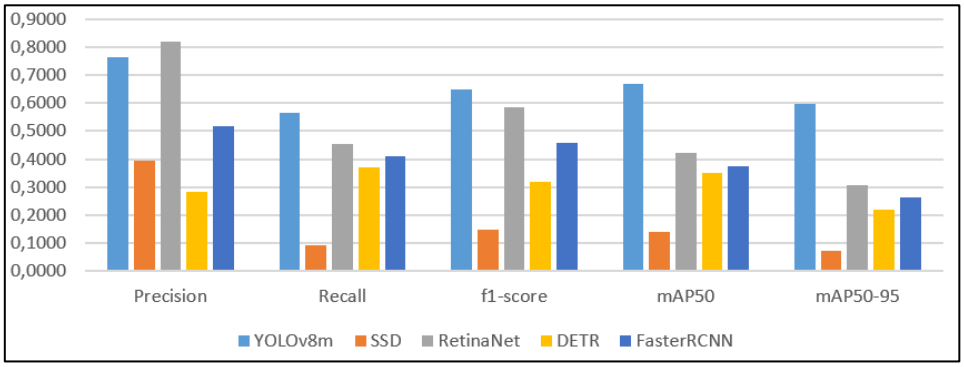


Рис. 23. Гистограмма метрик нейронных сетей.

На рисунке 24 изображена гистограмма быстродействия работы каждой нейронной сети. Самой медленной является *Faster R-CNN*, на обработку одного изображения ей требуется в среднем 2,4464 секунды. Немного быстрее работает *Retina Net*, эта нейронная сеть в среднем обрабатывает одно изображение за 1,9512 секунд. Остальные НС, в среднем, на обработку одного изображения тратят меньше 1 секунды. Так, *DETR* обрабатывает одно изображение за 0,8684 секунды, *YOLOv8m* – за 0,4877 секунды, а *SSD* – за 0,1628 секунды.

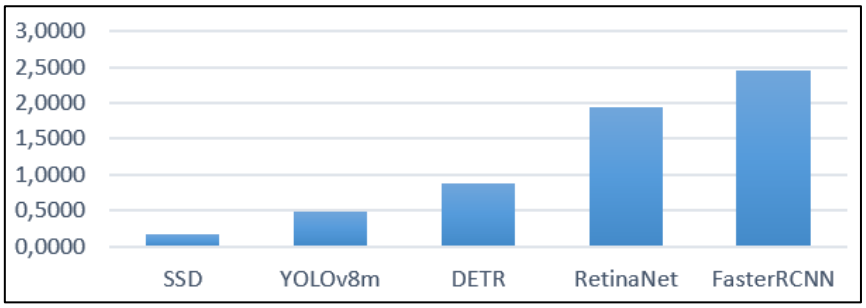


Рис. 24. Среднее время обработки НС одного изображения (в секундах).

Среднее время работы НС вычислялось на ЭВМ с процессором *Intel Core i*7-10510*U*.

Таким образом, модель *YOLOv8m* показала наибольшую среднюю точность, и, вместе с тем, она же и самая быстрая в вычислительном плане, поскольку в процессе своей работы она сжимает исходное изображение и обрабатывает его всего один раз. Эта модель имеет значительные преимущества перед рядом уже устаревших архитектур (например, *SSD*) и может быть использована в тех случаях, когда необходима быстрая автоматизированная обработка изображений не только в оптическом, но и в радиодиапазоне длин волн [9].

**Заключение**

В настоящее время нейросетевой подход при построении обнаружителей и классификаторов объектов на изображениях уже стал стандартом в индустрии обработки данных дистанционного зондирования Земли [10]. Выполненное авторами данной работы сравнительное тестирование нейросетевых обнаружителей подтвердило их высокую эффективность при решении задач обнаружения квазистационарных объектов (к которым, несомненно, относятся морские суда) в видимом диапазоне длин волн, в том числе и в условиях сложной фоноцелевой обстановки.

Появление новых наборов данных и архитектур нейросетевых обнаружителей (классификаторов) в перспективе позволит более эффективно решать задачу обнаружения любых объектов как в оптическом (видимом, инфракрасном), так и в радио диапазонах длин волн.

Так, например, использование набора данных с изображениями размером 800*800 пикселей сделает возможным обучение (и работу) модели *RetinaNet* без интерполяции входного изображения, а современные вычислительные платформы, в частности на основе тензорных процессоров, позволят сделать это быстро и обеспечат дополнительные преимущества, связанные с масштабируемостью и легкостью дальнейшего использования обученных моделей.

## Литература

1. Корчагин В.Д. Анализ современных SOTA-архитектур искусственных нейронных сетей для решения задач классификации изображений и детекции объектов // Программные системы и вычислительные методы. – 2023. – №. 4. – С. 73-87. https://doi.org/10.7256/2454-0714.2023.4.69306.
2. Траск Э. Грокаем глубокое обучение. – Питер, 2024.
3. Чорбаа Н.А., Ту Л.А., Толстой И.М. Сравнительный анализ методов детектирования объектов на радиолокационных изображениях при помощи нейронных сетей // Научный результат. Информационные технологии. – 2020. – Т. 5. – №. 4. – С. 15-25. https://doi.org/10.18413/2518-1092-2020-5-4-0-3.
4. Carion N. et al. End-to-end object detection with transformers // European conference on computer vision. – Cham: Springer International Publishing, 2020. – С. 213-229.
5. Болховитина Е.И. Исследование моделей свёрточных нейронных сетей YOLOV3 и RETINANET для задачи детектирования лица человека на изображении //StudNet. – 2022. – Т. 5. – №. 6. – С. 5439-5448.
6. Liu W. et al. Ssd: Single shot multibox detector // Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. – Springer International Publishing, 2016. – С. 21-37.
7. Брехт Э.А., Коншина В. Н. Применение нейронной сети YOLO для распознавания дефектов // Интеллектуальные технологии на транспорте. – 2022. – №. 2 (30). – С. 41-47. https://doi.org/10.24412/2413-2527-2022-230-41-47.

8. Тимошкин М.С., Миронов А.Н., Леонтьев А.С. Сравнение YOLO V5 и Faster R-CNN для обнаружения людей на изображении в потоковом режиме // Международный научно-исследовательский журнал. – 2022. – №. 6-1 (120). – С. 137-146. https://doi.org/10.23670/IRJ.2022.120.6.020.
9. Ситнова А.В., Валитов Э.Р., Светозарский С.Н. Применение алгоритмов глубокого машинного обучения на основе многослойной нейронной сети YOLOv8 для идентификации грибкового кератита // Современные технологии в медицине. – 2024. – Т. 16. – №. 4. – С. 5-14. https://doi.org/10.17691/stm2024.16.4.01.
10. Копенков В.Н. Совместный анализ оптических и радиолокационных данных дистанционного зондирования Земли в задачах анализа и мониторинга территории // Сборник трудов ИТНТ-2019. – 2019. – С. 21-24.

**Для цитирования:**

Белокопытов М.Л. Оценка эффективности применения некоторых SOTA моделей нейронных сетей при решении задач классификации квазистационарных объектов в видимом диапазоне длин волн. // Журнал радиоэлектроники. – 2025. – №6. https://doi.org/10.30898/1684-1719.2025.6.2