

Пояснительная записка

к дипломной работе на тему:

# **"Детекция и классификация повреждений**

# **дорожного полотна по фотографиям"**

Автор: Алексеев Дмитрий

Группа: DLL-18, DSU-5

# ОГЛАВЛЕНИЕ

1. Постановка задачи …………………………..…………………………………………….3
2. Анализ данных………………………………………………….………………………….4
3. Методика реализации………………………………………………………………..…….9
4. Итоги обучения……………………………………………………………………………12
5. Выводы …………………………………………………………………………………….15

«В России две беды, и одна постоянно чинит другую» (Михаил Задорнов)

# ПОСТАНОВКА ЗАДАЧИ

По состоянию на 2020 год, Россия занимала 5-ое место в мире по протяженности автомобильных дорог (1 553 663 км)[[1]](#footnote-1). Согласно официальной статистике ГИБДД за 2022 год[[2]](#footnote-2), по статье «*ДТП, в местах совершения которых выявлены нарушения обязательных требований к эксплуатационному состоянию и обустройству автомобильных дорог общего пользования, улиц и дорог городов и сельских поселений, железнодорожных переездов*» (включающей в себя ДТП по причине плохого состояния дорог) произошло 38 647 ДТП (31% от общего количества), в которых погибло 4 089 чел., ранено 48 241 чел. И хотя статистика показывает уменьшение количества ДТП по этой статье на 15% к 2021г. (и пропорциональное уменьшение количества погибших/раненых), контроль состояния дорожного полотна представляется важной социально-экономической задачей. Улучшение состояния дорог приведет не только к уменьшению числа погибших и пострадавших в ДТП, но и снизит экономический ущерб от повреждений транспортных средств и грузов, уменьшит количество пробок, улучшит логистические показатели (товарооборот), приведет к развитию отдаленных территорий, повысит общее качество жизни в стране.

В настоящее время контроль состояния дорожного полотна, как правило, осуществляется в неавтоматическом режиме (визуальный осмотр работниками дорожных и контролирующих служб с фотографированием и видеосъемкой, с последующим анализом информации «вручную»). Такой формат контроля предполагает существенные трудо- и материальные затраты, а также неизбежно связан с малым процентом охвата дорожной сети (густонаселенные районы РФ контролируются лучше, чем небольшие дороги на периферии).

С целью повышения эффективности процесса контроля в данном проекте реализована автоматизация анализа состояния дорожного полотна с помощью нейронных сетей (далее - НС) по следующим направлениям:

* детекция (выявление наличия повреждения дорожного покрытия и его местоположения на источнике данных – фотоснимке или видеокадре);
* классификация (отнесение выявленного повреждения к одной из категорий).

Задача в целом относится к классу «Object Detection» и успешно реализуется средствами сверточных НС (CNN). Поэтому в качестве основной архитектуры НС была выбрана YOLOv5 («You Only Look Once» v5 от проекта Ultralytics[[3]](#footnote-3)), в качестве дополнительной для сравнения – SSD300 (Single-Shot-Detection @images 300х300)[[4]](#footnote-4) на базе архитектуры VGG16. Используемый фреймворк для реализации задачи – PyTorch.

Основной метрикой оценки качества обучения НС и сравнения результатов послужила mAP(mean Average Precision)[[5]](#footnote-5) с IoU=0.5 (далее - mAP@0.5), т.е. среднее значение Precision при различных Recall, с условием перекрытия (Intersection-over-Union) не менее 0.5[[6]](#footnote-6).

# АНАЛИЗ ДАННЫХ

В качестве исходных данных для решения задачи был использован размеченный датасет с фотографиями, общедоступный в рамках глобального соревнования (челленджа) «Crowdsensing-based Road Damage Detection Challenge» (CRDDC2022)[[7]](#footnote-7), проведенного в 2022 году с участием более 60-и команд из 19-и стран.[[8]](#footnote-8) Соревнование направлено на поиск наиболее эффективных методов автоматического обнаружения повреждений дорожного полотна по фотографиям и/или видеороликам, сделанным «обычными» средствами (смартфоны, видеорегистраторы) или беспилотными летательными аппаратами (дронами).

В наилучшей модели, предложенной победителями челленджа, использовалось ансамблевое обучение, основанное на моделях архитектур YOLO и Faster-RCNN, что дало значение метрики F1 = 76% для тестовых (неразмеченных) фотографий, включенных в состав датасета.

Автором дипломного проекта был предварительно проведен поиск общедоступных вариантов решения задачи на таких ресурсах, как Kaggle, Github, Paperswithcode и Huggingface. Были обнаружены варианты решения задачи детекции (Pothole Detection, т.е. только детекция повреждений без их классификации), а также варианты использования предобученных моделей с применением фреймворка Tensorflow[[9]](#footnote-9), что не соответствует условиям поставленной задачи. Варианты «свежих» решений (на датасете за 2022 год) не были обнаружены в принципе. Можно сделать вывод, что участники глобального соревнования не размещали готовые решения на общедоступных ресурсах, что логично в свете продолжения соревнований в 2023 году и далее.

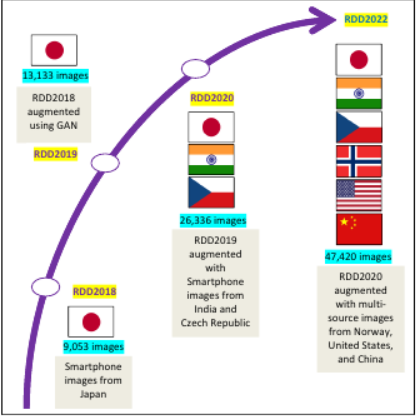


Рис.1 «Динамика развития глобального соревнования по обнаружению повреждений дорог»

Набор данных содержит 47420 цветных фотографий, в т.ч. 38385 (81%) размеченных (Train/Val) и 9035 (19%) - без разметки (Test, для проверки организаторами), по 6-и странам:

* Япония: фото 600х600 с видеорегистратора/смартфона, закрепленного в автомобиле;
* Китай: фото 512х512 с видеорегистратора/смартфона, закрепленного на мотоцикле (China\_M), а также фото 512х512, сделанные с дрона (China\_D);
* Индия: фото 720х720 с видеорегистратора/смартфона, закрепленного в автомобиле;
* Чехия: фото 600х600 с видеорегистратора/смартфона, закрепленного в автомобиле;
* Норвегия: фото 4040х2035 со специализированной камеры, закрепленной на крыше автомобиля (фото «склеены» из двух половинок);
* США: фото 640х640 с сервиса Google Maps (панорамы улиц).

Фотографии сделаны в хороших погодных условиях, при отсутствии осадков (дождя, снега) как в процессе фотосъемки, так и на дороге (нет ям, залитых водой).

Фотографии были размечены организаторами соревнования при помощи инструментов LabelImg и CVAT (Computer Vision Annotation Tool) в формате PASCAL VOC (к каждой фотографии из train/val-датасета есть файл разметки повреждений в формате xml). Ко многим фотографиям применена аугментация, т.е. фото могут повторяться с небольшими вариациями.



Рис.2 «Пример xml-файла разметки в формате PASCAL VOC»

Классификация повреждений соответствует «Руководству по дорожному ремонту и обслуживанию» Japan Road Association (2013 г., с дополнениями за 2019г) и приведена на рис.3 (основные типы повреждений выделены красной рамкой).

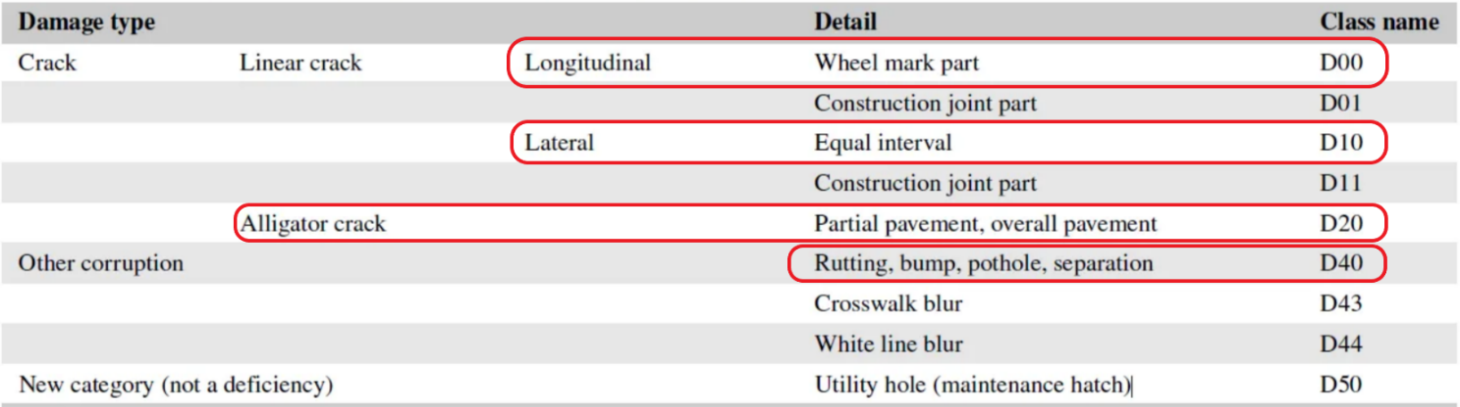


Рис.3 «Классификация повреждений дорожного полотна».

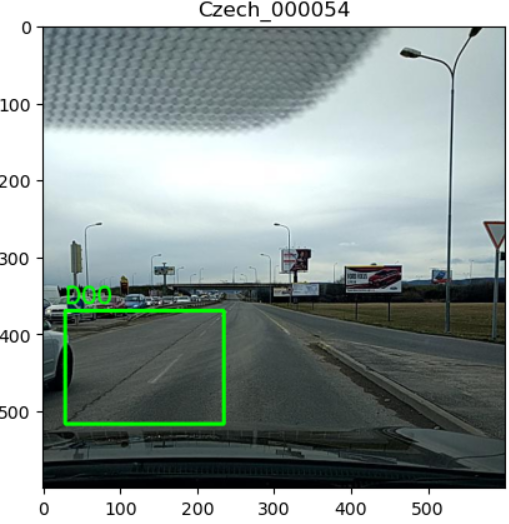


Рис.4 «Пример продольной трещины (D00, «Longitudinal Linear Crack»).

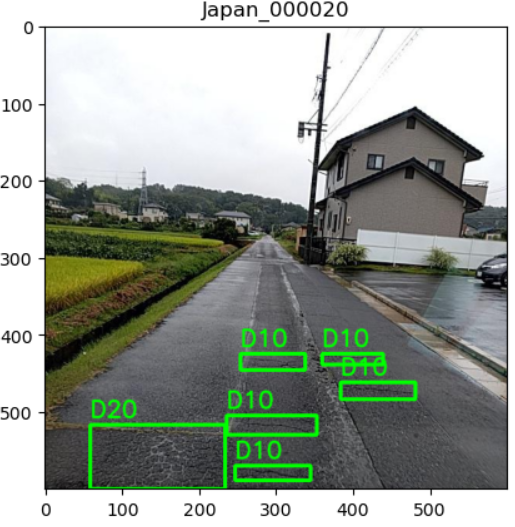


Рис.5 «Примеры поперечных трещин (D10, «Lateral Linear Crack»).

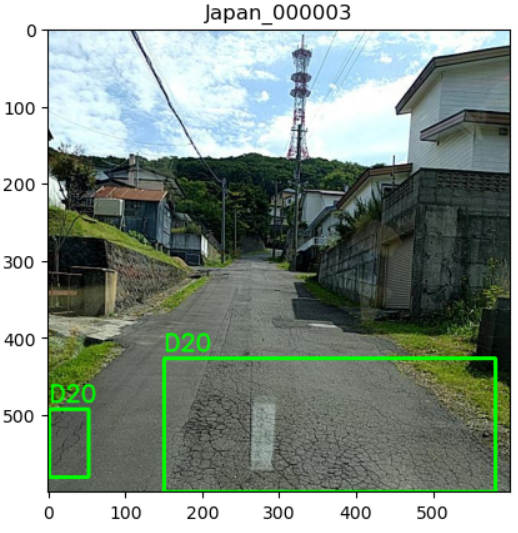


Рис.6 «Примеры «крокодилового» растрескивания (D20, «Alligator Crack»).



Рис.7 «Пример ямы/выбоины (D40, «Pothole»).

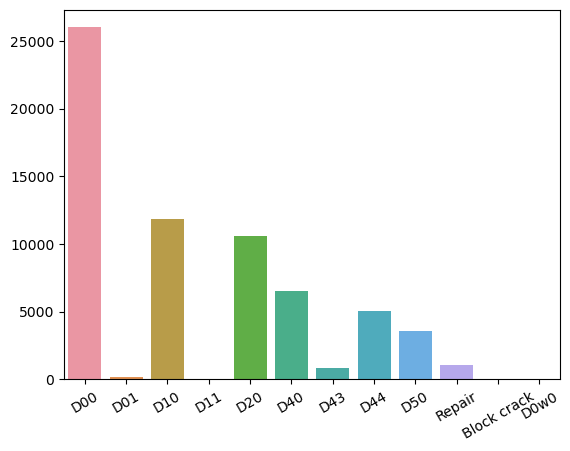


Рис.8 «Распределение повреждений по классам в размеченном (train/val) датасете».



Рис.9 «Распределение повреждений по типам и странам в размеченном (train/val) датасете».

На основании анализа данных были сделаны следующие выводы:

1. Классы не сбалансированы между собой. В частности, по Индии и Японии сделана более тщательная разметка (выделены элементы дорожной разметки, канализационные люки), а по Китаю размечены элементы отремонтированного дорожного полотна. В сумме такие «неосновные» элементы составляют порядка 16% train/val-датасета.

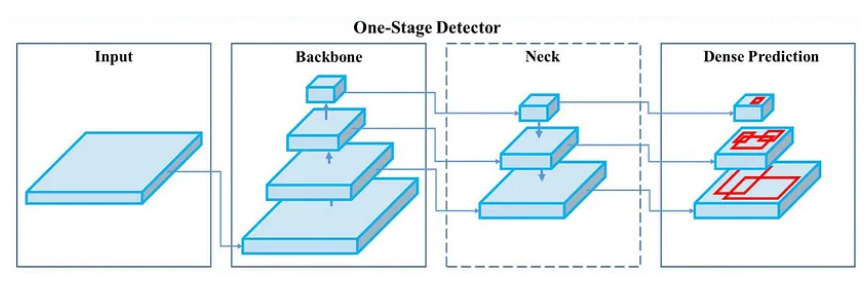
* В рамках глобального соревнования «неосновные» элементы предлагалось не учитывать в обучении моделей.
* В рамках дипломного проекта, учитывая существенную долю в общем количестве объектов, а также с целью эксперимента (анализа устойчивости модели к выбросам), такие элементы принято оставить в обучающей и валидационной выборке для основного и менее ресурсозатратного варианта (YOLOv5).
* Для обучения модели в дополнительном варианте (SSD300@VGG16) было принято решение ограничить количество классов 4-мя основными (D00, D10, D20, D40) – по причине значительно большей ресурсозатратности обучения VGG16.

1. В данных по Индии преобладают «критические» повреждения дорожного полотна (ямы и выбоины). По Норвегии и США – наоборот, что может говорить о значительно лучшем состоянии дорожного полотна в этих странах (неожиданно)), либо о меньшей добросовестности разметчиков фотографий.
2. По Японии предлагается датасет, наиболее сбалансированный между классами. Следовательно, данные по Японии могут быть использованы в качестве «пробного» датасета (ограниченной выборки) для расчета трудоемкого варианта - SSD300@VGG16, с целью подбора оптимальных гиперпараметров обучения на полной выборке (по всем 6-и странам).
3. Данные по тестовой выборке, по которым разметка объектов не была включена в датасет организаторами, в рамках дипломного проекта могут быть использованы только для визуальной оценки качества обучения моделей. Разбивка на трейн- и тест-выборки будет производиться в соотношении 80/20 на основании разделения размеченного (train/val) датасета.

# МЕТОДИКА РЕАЛИЗАЦИИ

# Вариант №1 (основной): YOLOv5

YOLOv5 является одной из State-of-the-Art архитектур, используемых в детекции объектов, и относится к архитектуре One-Stage detector: подход, который предсказывает координаты определённого количества *bounding box*'ов с результатами классификации и вероятности нахождения объекта, и в дальнейшем корректируя их местоположение. В целом такую архитектуру можно представить в следующем виде[[10]](#footnote-10):



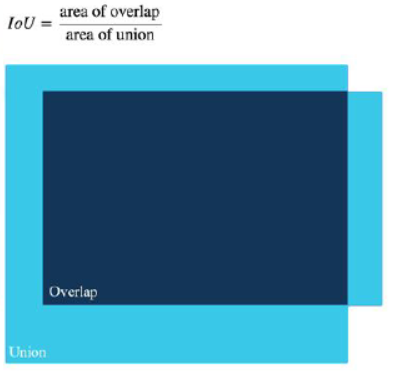
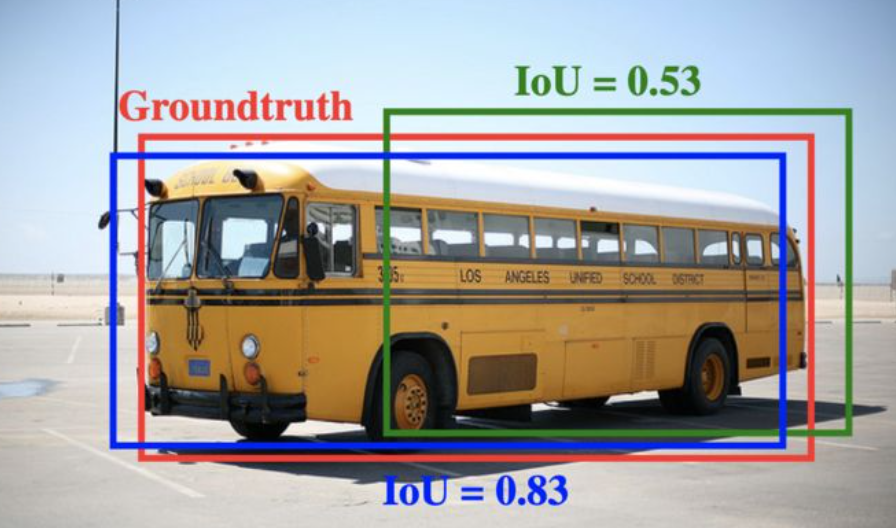
Модель YOLOv5 была реализована на PyTorch, предобучена разработчиками на датасете MS COCO (80 классов), и в полноразмерных вариантах показывала [mAP@0.5](mailto:mAP@0.5) на уровне выше 50%, а также показывала достаточно высокую сходимость за приемлемое время, что и предопределило выбор данной модели в качестве основной для решения задачи.

В ходе подготовки данных и обучения модели на YOLOv5 были проделаны следующие шаги:

1. Размеченный (train/val) датасет вручную разбит на трейн-(80%) и тест-(20%) выборки с сохранением %-ного соотношения между странами; все классы объектов сохранены;
2. Разметка файлов сконвертирована из формата PASCAL VOC (XML) в формат YOLO (TXT, координаты bounding boxes отнормированы по центру изображения, все классы объектов сохранены);
3. Склонирован с github репозиторий Ultralytics, содержащий pretrained-модели YOLOv5 разной степени сложности (от Nano до eXtraLarge), с целью дальнейшего обучения на нашем датасете;
4. Сформирован настроечный YAML-файл (пути к данным + перечисление классов);
5. Проведено предварительное обучение на малом датасете (данные только по Чехии) с целью выбора подходящей модели и гиперпараметров для всего датасета. Для контроля процесса обучения использовался Tensorboard, а также файлы с метриками, формируемые непосредственно моделью YOLOv5 в ходе обучения;
6. После подбора подходящей модели и гиперпараметров проведено обучение на всем датасете (по всем странам);
7. Проведен референс модели на тестовом датасете (фото без разметки), а также на общедоступных (любительских) видеороликах с YouTube.

Для оценки результатов обучения были использованы следующие метрики[[11]](#footnote-11):

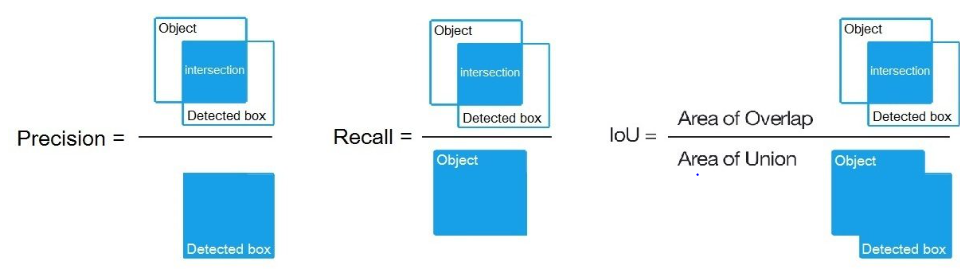
1. Intersection-over-Union (если IoU > 0.5, то предсказание можно считать «хорошим»)



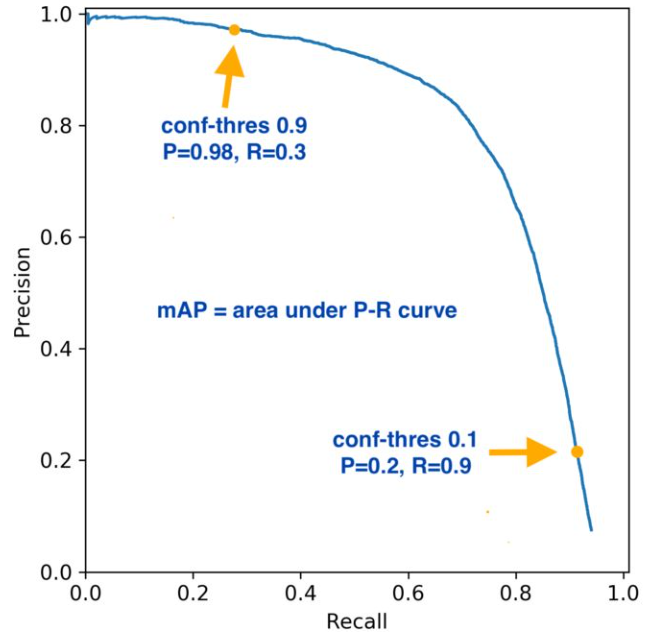
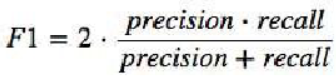
1. Precision (насколько точно предсказанный объект попал в истинный Groundtruth), Recall (насколько полно мы нашли все совпадения предсказаний с истинным Groundtruth):

Precision = True positive / (True positive + False positive)

Recall = True positive / (True positive + False negative)

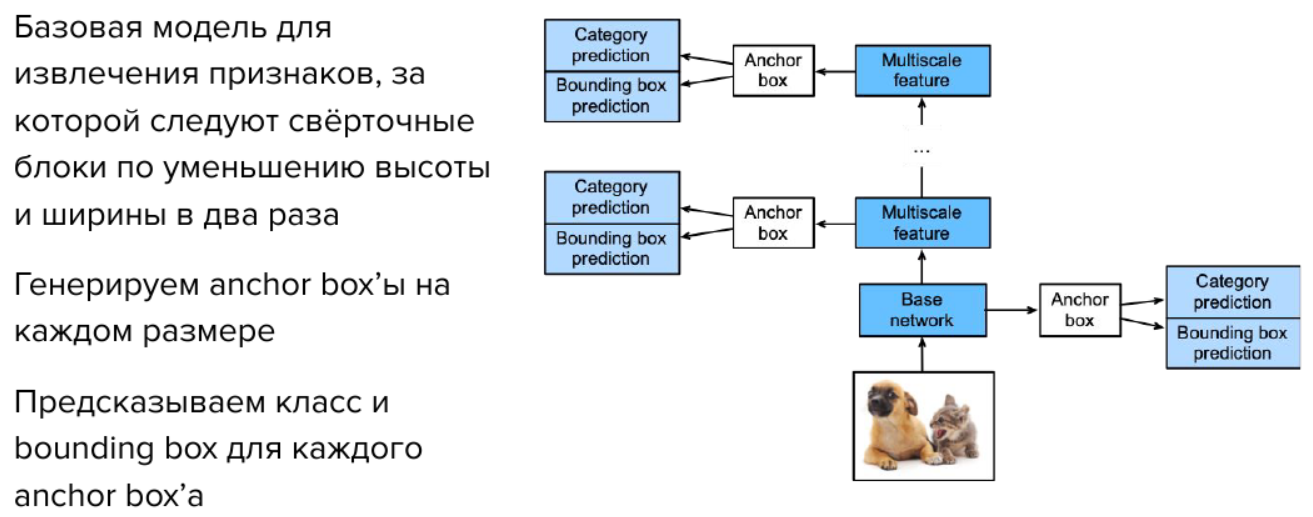


1. F1-score (гармоническое среднее между Precision и Recall) и mean Average Precision (среднее значение Precision при различных Recall, усредненное по всем классам). mAP@0.5 означает, что mAP рассчитан для порога IoU>0.5 (средняя доля правильно предсказанных объектов составляет выше 50%):



# Вариант №2 (дополнительный): SSD300 @VGG16

Данный вариант был выбран для сравнения с YOLOv5 и обучения модели НС «с нуля» (были использованы как default pretrained-веса VGG16\_Weights.IMAGENET1K\_V1, так и не использованы, - разница нивелировалась в течение нескольких эпох обучения). Обобщенное описание модели SSD приведено ниже[[12]](#footnote-12):



В ходе подготовки данных и обучения модели на SSD300 @VGG16 были проделаны шаги:

1. На основании размеченного трейн/вал-датасета, использованного ранее для YOLOv5, были сформированы файлы разметки в формате JSON (список классов, список трейн-файлов и трейн-объектов в них, список валидационных файлов и вал-объектов в них). С целью повышения скорости обучения оставлены только основные классы (D00, D10, D20, D40);
2. Проведена аугментация загружаемых данных - random crop, переворот, изменение размера и фотометрических параметров (яркости, насыщенности, контраста);
3. Описана структура SSD300 на архитектуре VGG16;
4. На малом датасете (только Япония) подобраны гиперпараметры обучения;
5. Проведено обучение на всём датасете;
6. Рассчитаны значения метрики mAP@0.45[[13]](#footnote-13) для сравнения c YOLOv5;
7. Проведен референс модели на выборочных снимках из тестового датасета.

# ИТОГИ ОБУЧЕНИЯ

Обучение сетей проводилось с использованием GPU на домашнем ПК в конфигурации Intel

Xeon E2670v3, RAM 32Gb, Nvidia RTX 2060 12Gb (Python-3.9.12, torch-1.13.0)

Облачные ресурсы (Colab, Kaggle) было решено не использовать в силу ограниченности

бесплатных лимитов.

В серии экспериментов были достигнуты следующие результаты:

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Показатель** | **YOLOv5** | **SSD300@VGG16** |
| 1 | mAP@0.45 по всем классам (%%), в т.ч.: | **62.4**[[14]](#footnote-14) | **32.1** |
|  | - D00 (продольные) | 56.2 | 32.8 |
|  | - D10 (поперечные) | 53.0 | 24.0 |
|  | - D20 (растрескивание) | 65.8 | 47.5 |
|  | - D40 (ямы/выбоины) | 55.5 | 24.3 |
| 2 | Оптимальная pretrained- модель | YOLOv5m (291 layers, 20915769 parameters, 20915769 gradients, 48.4 GFLOPs) | N/A |
| 3 | Batch Size | 16 | 8[[15]](#footnote-15) |
| 4 | Image Size | All (rescale to 600x600) | All (rescale to 300x300) |
| 5 | Learning Rate | 0.01 | 0.0001[[16]](#footnote-16) |
| 6 | Метод оптимизации | SGD | AdamW[[17]](#footnote-17) |
| 7 | Количество итераций / эпох обучения | 70 эпох[[18]](#footnote-18) | 48000 итераций (20 эпох) |
| 8 | Общее время обучения для достижения mAP (п.1) | ~14 часов (~5 эпох/час) | ~38 часов (~0.5 эпох/час) |
| 9 | Итоговый (суммарный) Loss | N/A[[19]](#footnote-19) | 3.34 |

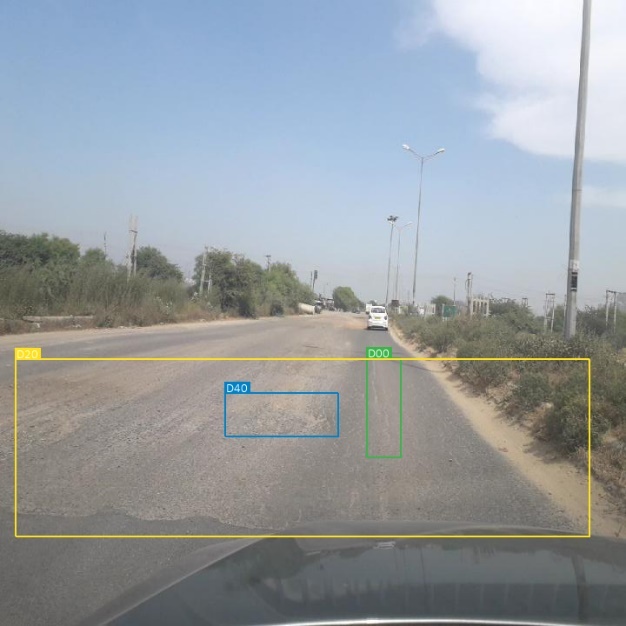
Для визуального сравнения результатов были сделаны несколько инференсных предсказаний:

1. На тестовом (неразмеченном) датасете, который не участвовал в обучении моделей:

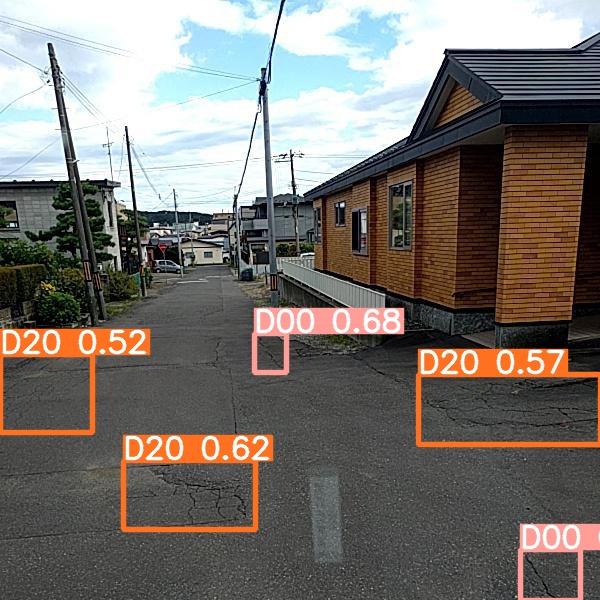
a) результаты предсказаний: China\_MotorBike\_001977.jpg (слева – YOLO, справа - SSD):



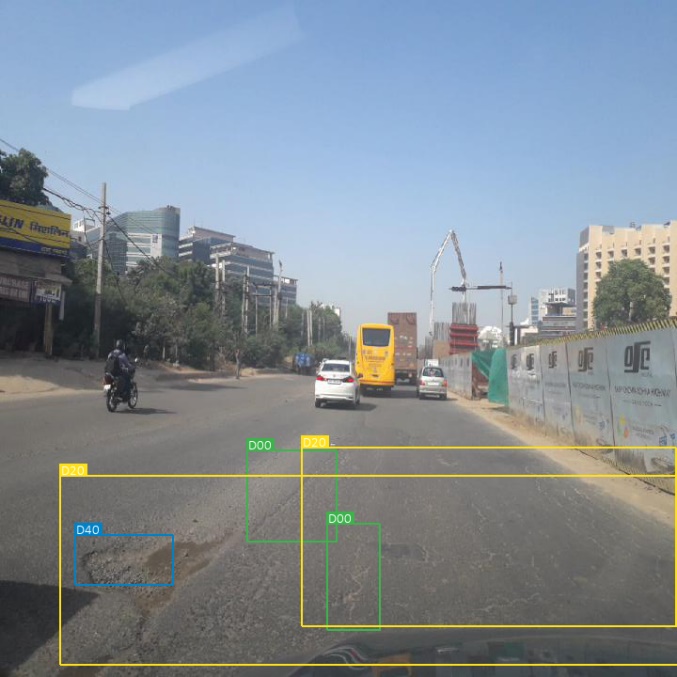
b) результаты предсказаний: India\_000807.jpg (слева – YOLO, справа - SSD):

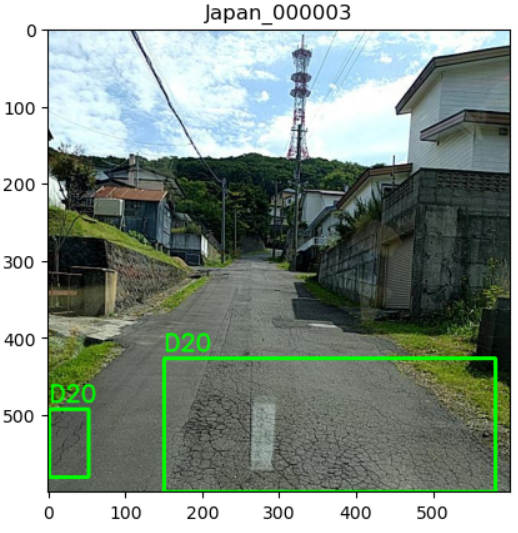
1. результаты предсказаний: Japan\_000195.jpg (слева – YOLO, справа - SSD):

1. Дополнительно убедимся в достаточно высоком (визуально) качестве предсказаний SSD300, сравнив предсказания с разметкой (Ground Truth):
2. результаты предсказаний: India\_000317.jpg (слева – GroundTruth, справа - SSD):



1. результаты предсказаний: Japan\_000003.jpg (слева – GroundTruth, справа - SSD):



# ВЫВОДЫ

Задача детекции и классификации повреждений дорожного полотна с применением методов глубокого обучения была успешно реализована. Наилучшие результаты показала архитектура YOLOv5, позволив приблизиться к достижениям, полученным в ходе глобального соревнования. Вместе с высокой скоростью и качеством на обучении, устойчивостью к выбросам (для обучения YOLOv5 был оставлен несбалансированный полный датасет), эта архитектура также показывает малое время инференса, что позволяет использовать её для решения задач реал-тайм детекции и классификации.

Тем не менее, реализация на базе SSD300@VGG16 также показала достойные результаты: несмотря на более низкие метрики, визуально качество детекции находится на высоком уровне.

В ходе работы над проектом были выявлены направления для дальнейшего повышения качества решения задачи:

* Дообучение сети на снимках со сложными/скрытыми повреждениями дорожного полотна (вспучиваниями, а также ямами, залитыми водой или грязью);
* Использование вычислений в формате FP16 (вместо FP32) для обучения SSD300@VGG16, что позволит существенно сократить время на обучение модели с сохранением высокой точности предсказаний. При этом YOLOv5 уже по умолчанию обучается в режиме AMP (Automatic Mixed Precision);
* Разметка данных и дообучение сети на датасете из России, в т.ч. с использованием открытых источников (панорамы улиц на Гугл- и Яндекс-картах, использование информации из проекта ОНФ «Убитые дороги» и т.д.);
* Подсчет выявленных повреждений (по каждому типу) с целью выявления наиболее «убитых» и опасных участков дорог, определения приоритетности ремонтов;
* Реализация привязки выявленных повреждений к точным координатам (геолокации) и дате/времени места съемки – с целью создания связки «Повреждение дороги – Геолокация - Время» и последующего анализа процессов в динамике, в т.ч. с помощью анализа временных рядов (Time Series).

1. https://ru.wikipedia.org/wiki/ Список\_стран\_по\_длине\_сети\_автомобильных\_дорог [↑](#footnote-ref-1)
2. <http://stat.gibdd.ru/> Раздел 5. Дорожно-транспортные происшествия и пострадавшие по видам ДТП, январь - декабрь 2022 [↑](#footnote-ref-2)
3. https://github.com/ultralytics/yolov5 [↑](#footnote-ref-3)
4. https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection [↑](#footnote-ref-4)
5. Формулы метрик приведены ниже по тексту [↑](#footnote-ref-5)
6. Весь ноутбук (анализ/обработка данных, обучение моделей, референсы на фото и роликах) выложен на Github [↑](#footnote-ref-6)
7. <https://crddc2022.sekilab.global/> (датасет можно скачать без регистрации на челлендже), см. также научные публикации https://arxiv.org/abs/2209.08538 [↑](#footnote-ref-7)
8. Результаты за 2022 год: https://www.researchgate.net/publication/365634205\_Crowdsensing-based\_Road\_Damage\_Detection\_Challenge\_CRDDC-2022 [↑](#footnote-ref-8)
9. <https://github.com/sekilab/RoadDamageDetector> (Tutorial @TF с использованием предобученной SSD\_Mobilenet) [↑](#footnote-ref-9)
10. https://jonathan-hui.medium.com/yolov4-c9901eaa8e61 [↑](#footnote-ref-10)
11. https://www.kdnuggets.com/2020/08/metrics-evaluate-deep-learning-object-detectors.html [↑](#footnote-ref-11)
12. Лекция Нетологии по курсу Deep Learning на тему «Компьютерное зрение (#1)» [↑](#footnote-ref-12)
13. Использовано значение IoU=0.45, а не IoU=0.5, т.к. в настройках YOLOv5 по умолчанию задан этот порог, оценка производилась непосредственно в ходе обучения, не переделывалась после обучения. Погрешность 5% признана несущественной. [↑](#footnote-ref-13)
14. Наилучший результат глобального соревнования за 2022год: 76%. [↑](#footnote-ref-14)
15. Были перепробованы разные варианты, начиная от 2 до 64 (пока помещались в память GPU), но оптимальным по производительности оказался размер батча 8 [↑](#footnote-ref-15)
16. При LR=0.01 и 0.001 градиенты вели себя крайне неустойчиво и в итоге взрывались, несмотря на clip\_gradients [↑](#footnote-ref-16)
17. SGD показал гораздо худшие результаты по времени сходимости (разница в несколько раз с AdamW), а также mAP на уровне 30%, лосс не опускался ниже 4.0. [SGD+Nesterov momentum также не дал хороших результатов]. [↑](#footnote-ref-17)
18. Уже после 10-и эпох (2 часа) был достигнут mAP@0.5 = 48.6%. Также было проведено дообучение сети до 150-и эпох (итого ~30 часов), но mAP@0.5 снизился с 62.4% до 60.7% (по всей видимости, НС начала переобучаться…) [↑](#footnote-ref-18)
19. В метриках такие итоговые значения по валидации: val/box\_loss = 0.045, val/obj\_loss = 0.01, val/cls\_loss = 0.01. [↑](#footnote-ref-19)