Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет ИТМО

Факультет Цифровых Трансформаций

Дисциплина: Инструментальные средства искусственного интеллекта

Отчет

по лабораторной работе № 2

«Применение инструментов для упрощения процессов разработки и исследований для обучения моделей ИИ»

Студенты:

Протопопов Артём Андреевич J4151,

Толмачев Сергей Евгеньевич J4140

Преподаватель: Проскурин Г.Е.

Санкт-Петербург

2024г.

Цель задания: выбрать задачу из области машинного обучения и выдвинуть гипотезы относительно способов её решения, используя при этом соответствующие фреймворки и системы трекинга.

Описание выбранной задачи: осуществить детектирование факта ношения людьми на картинке маски; всего есть 3 случая:

1. Маска надета некорректно (класс 0)
2. Маска надета (класс 1)
3. Маска не надета (класс 2)

Исходные данные: Face Mask Detection

Ссылка на код: Tendo1904/MLOps

Ход работы:

Гипотезы:

Для решения задачи были сформулированы следующие гипотезы:

1. Для экономии времени и моральных сил вместо разработки и экспериментирования с моделью собственной архитектуры, предлагается выбрать предобученную модель, которую можно дообучить на имеющемся размеченном датасете, что позволит получить в короткие сроки работоспособное решение.
2. Поскольку в данном случае предстоит работа с графическим материалом, что значительно повышает требования к вычислительным мощностям во время обучения, стоит рассмотреть возможность использования CUDA платформы для повышения скорости обучения.
3. Возможно, потребуется в дальнейшем корректировать или любым другим образом улучшать модель и наблюдать за процессом обучения в реальном времени, сравнивая между собой поведение тех или иных подходов. Для этого целесообразно подключить систему трекинга, чтобы облегчить процесс мониторинга обучения и визуализации моделей.

Проверка и реализация гипотез:

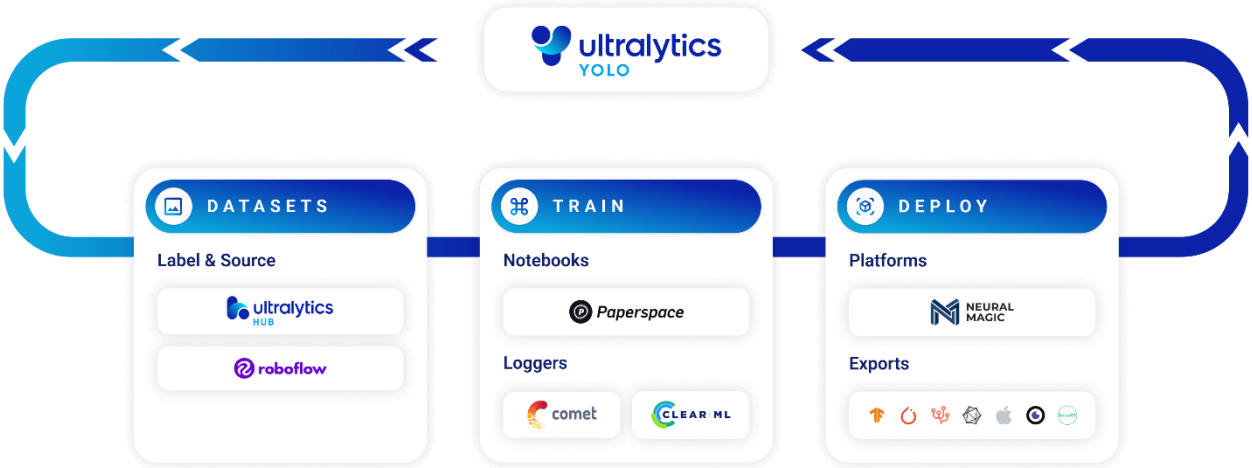
Выбор предобученной модели и препроцессинг данных:

Для решения поставленной задачи существуют различные модели и инструменты, однако требуемым функционалом, достаточно подробной документацией по использованию и широкой поддержкой обладают следующие модели: YOLO (You Only Look Once) и Detectron2 от Meta. В ходе проведенного разведывательного анализа была получена следующая сводная таблица характеристик:

Характеристика	YOLO	Detectron2
Скорость	Очень высокая, подходит для детекции в реальном времени	Ниже, особенно для сложных моделей
Точность	Достаточная для большинства приложений, может уступать Detectron2 на сложных задачах	Очень высокая, особенно для сегментации
Поддержка задач	Детекция объектов	Детекция объектов, сегментация, инстанс-сегментация
Простота настройки	Прост в настройке и быстр в обучении	Требует более сложной настройки и ресурсов
Применение	Мобильные приложения, камеры наблюдения, детекция в реальном времени	Научные исследования, промышленность, сложные задачи компьютерного зрения

По результатам проведенного исследования выбор был сделан в пользу использования моделей YOLO из-за простоты наладки, скорости обучения и потенциально приемлемой точности детектирования Use Cases of Ultralytics YOLOv8 Region Counting | Medium .

Также модель YOLOv8n поставляется совместно с фреймворком Ultralytics, что позволяет использовать «из коробки» MLOps инструменты



Для YOLO однако необходимо выстроить следующую организацию данных:

```
dataset/
├──── images/
│       ├──── train/
│       ├──── val/
└──── labels/
        ├──── train/
        └──── val/
```

Привести аннотации к .txt файлам следующего формата:

<class> <x_center> <y_center> <width> <height>, где все значения, кроме класса, нормализованы в диапазон [0, 1].

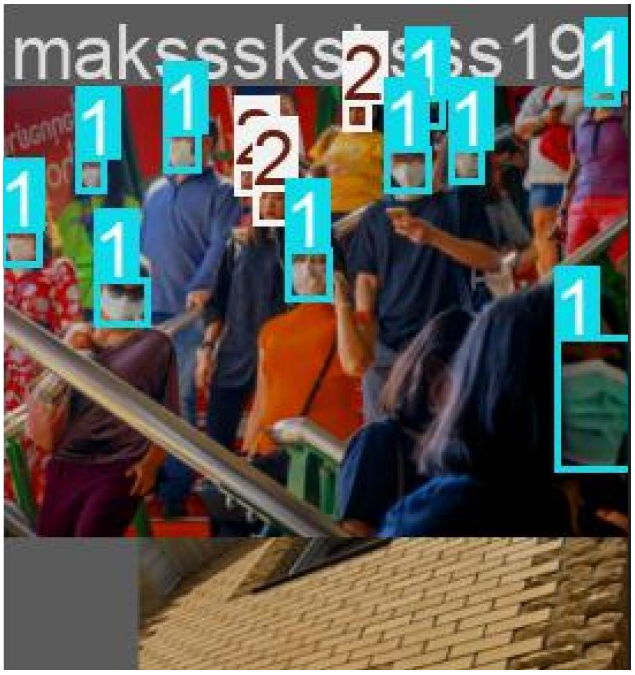
А также подготовить конфигурационный файл .yaml:

```
train: images/train

val: images/val

nc: 3  # количество классов

names: ['class1', 'class2', 'class3']
```

Для решения данной задачи был написан класс для выстраивания необходимой структуры данных, корректной аннотации и конфигурационного файла для обучения.

В результате реализации данной гипотезы можно отметить, что уже на этапе обучения (здесь вместо названий классов применяются их индексы) модель достаточно неплохо справляется с поставленной задачей:


1
maksssksksss428.png


maksssksksss191


maksssksksss293.png

Таким образом использование предобученной модели и удобного фреймворка, автоматизирующего работу с дообучением, позволяет существенно сократить время на разработку.

Использование CUDA:

В ходе обучения, однако, обнаружилась достаточно серьезная проблема, а именно: значительное время на обучение. Согласно логам, обучение на 50 эпох, используя лишь вычислительные мощности процессора Intel Core i5-6600K, заняло примерно 1 час 20 минут:

```
50 epochs completed in 1.372 hours.
```

В целом возможность дообучить модель, используя бюджетный процессор за 1.5 часа достаточна неплоха сама по себе и демонстрирует преимущество YOLO в скорости работы и вычислительной эффективности, но все же ускорить процесс обучения было бы предпочтительнее, особенно в условиях ограничения временных рамок на выпуск готового решения.

В связи с этим было решено использовать драйвера и библиотеки CUDA вычислений для глубокого обучения от NVIDIA, а именно CUDA v11.7.0 и cuDNN v8.9.7 (совместимы с pyTorch 2.0.0). Использование этих программных средств сделало возможным переложить дообучение с центрального процессора на дискретную видеокарту NVIDIA GTX 1070.

Данное решение позволило ускорить процесс обучения в 13 раз

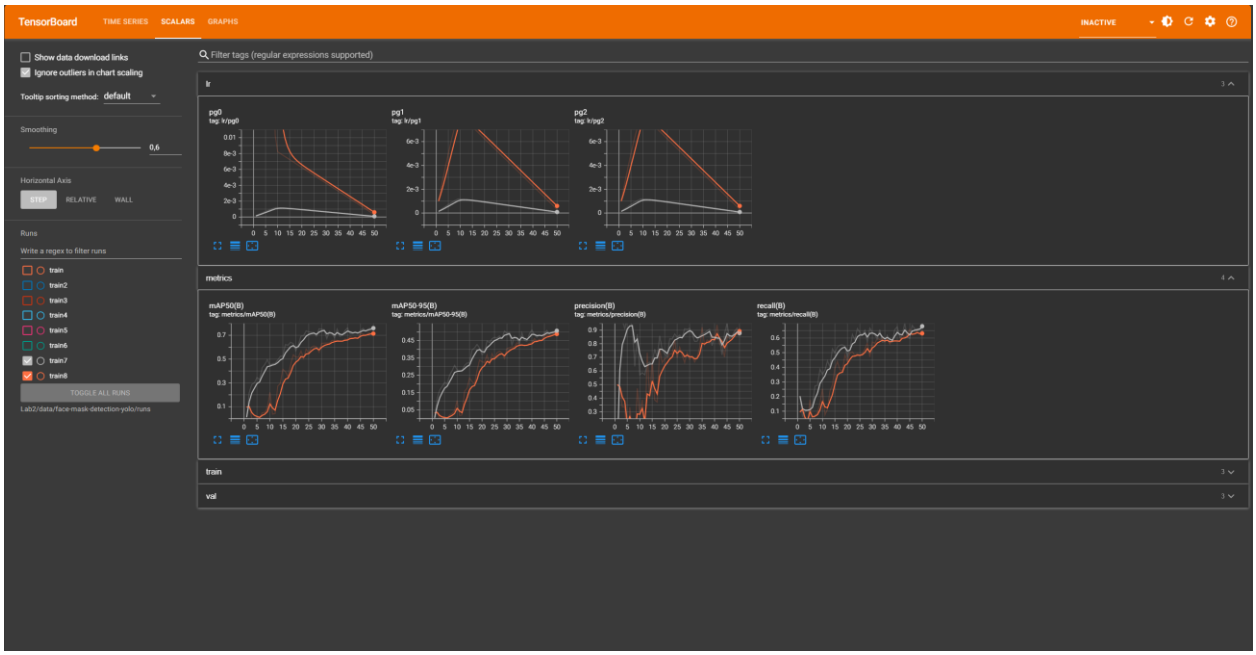
```
50 epochs completed in 0.105 hours.
```

Использование трекинг системы:

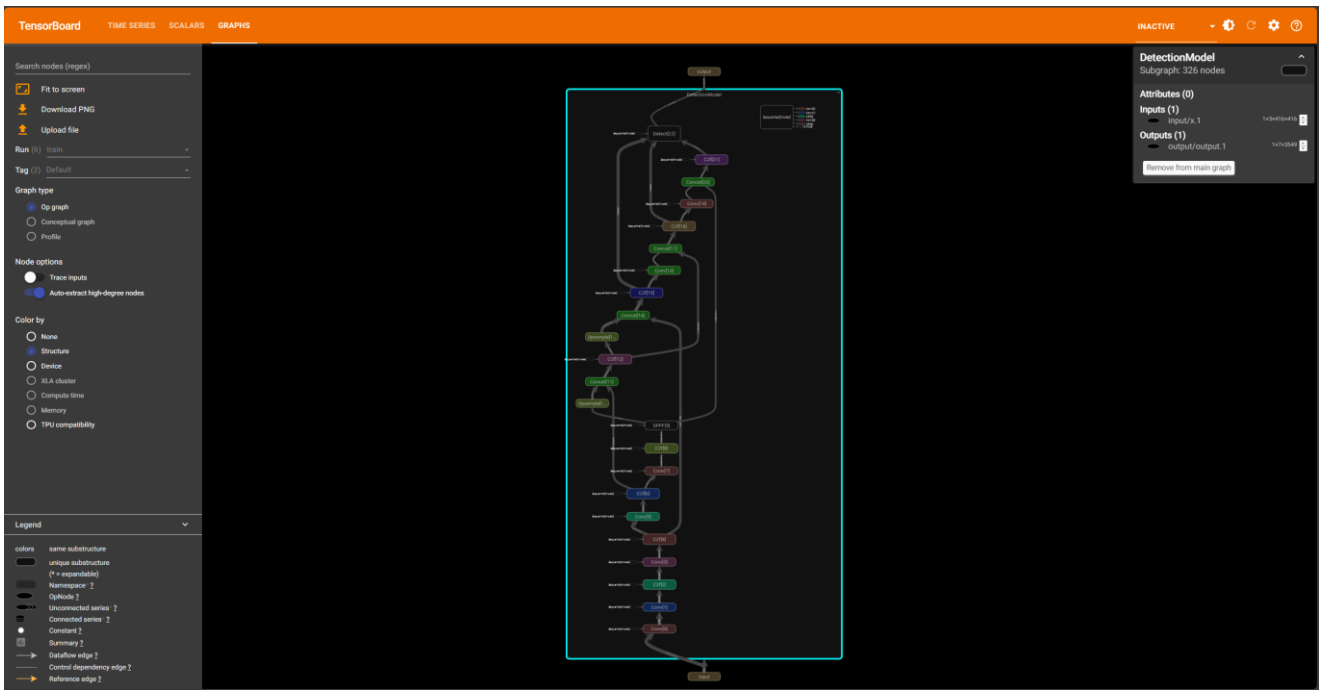
Для того чтобы, следить за обучением в реальном времени (период обновлений максимум раз в 30 секунд) был применен Tensorboard, который имеет поддержку и интеграцию с различными популярными фреймворками и библиотеками. Работает он, анализируя .tfevents. файлы, возникающие как callback-и в конце каждой эпохи с данными о результатах обучения.

Активируется он командой: tensorboard –logdir path/to/runs , где указывается путь с данными о результатах экспериментов и разворачивается локально на порту 6006.

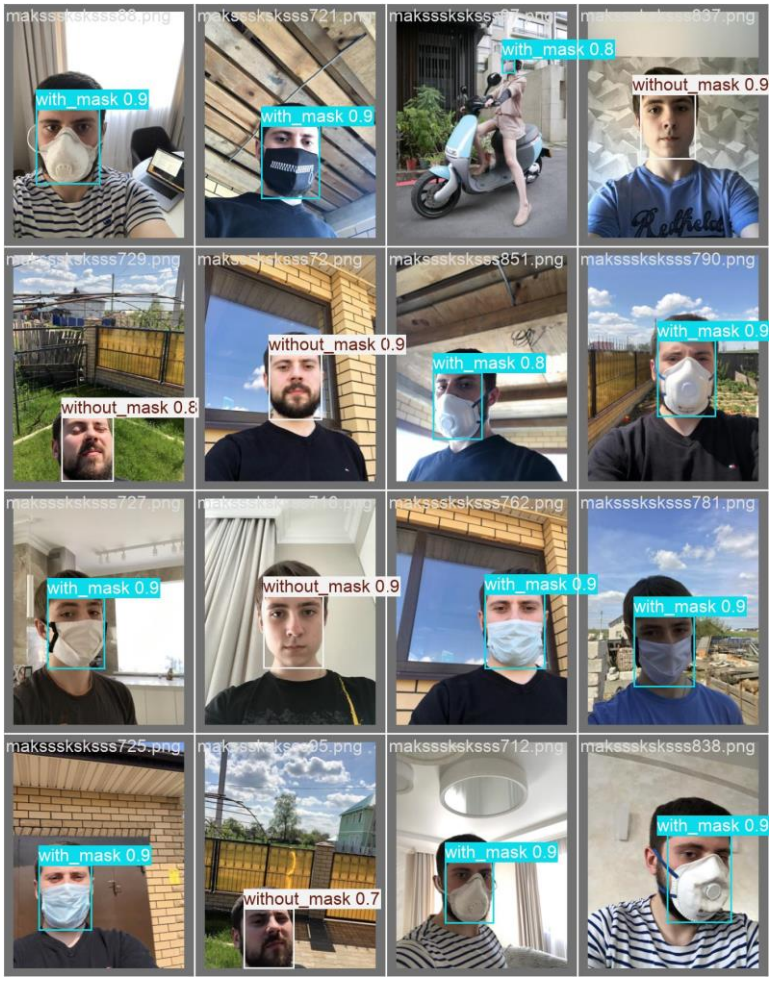
На рисунках представлены примеры метрик с запуском обучения на оптимизаторах AdamW (оранжевый) и Adam (синий).

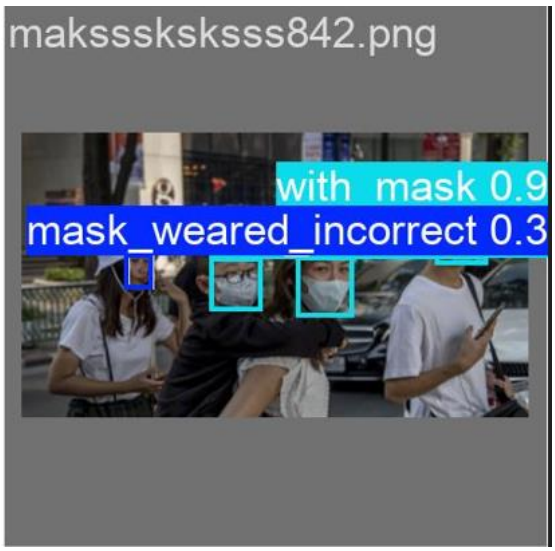


Также Tensorboard предоставляет возможность визуализации используемой модели, что позволяет лучше понять её устройство и при необходимости внести необходимые изменения.



В результате была получена модель (на основе train7), детектирующая факт ношения маски





Выводы:

В ходе выполнения лабораторной работы была решена задача детектирования 3х состояний на выбранном датасете с использованием модели YOLOv8n поставляемой фреймворком Ultralytics, платформой вычислений на графическом процессоре CUDA и системой трекинга Tensorboard. Все используемые инструменты позволяют значительно сократить время на разработку решения, а также его обучения и анализа результатов эксперимента. Например, в данном случае модель, использовавшая в качестве оптимизатора AdamW показывает лучшие результаты с точки зрения метрики mAP50, что позволяет говорить о более лучшем детектировании и локализации объектов при пороге точности в 50%. Также фреймворк Ultralytics позволяет экспортировать получившиеся модели в ONNX формат, т.е. дает возможность оптимизировать модель для повышения производительности инференса. Сложности были вызваны в организации требуемого паттерна организации данных и аннотаций для дообучения YOLO, так как исходный датасет организован произвольно, а также поиском и применением нужных версий CUDA и cuDNN, чтобы не возникало конфликтов с pyTorch, работающем под «капотом» Ultralytics.