«Математические модели и численные методы в биологии и медицине» XV

Валетов Дмитрий Кириллович
Сеченовский Университет &
Институт Вычислительной Математики

Москва
3.11.2022

Портирование и применение нейросетей в браузерах

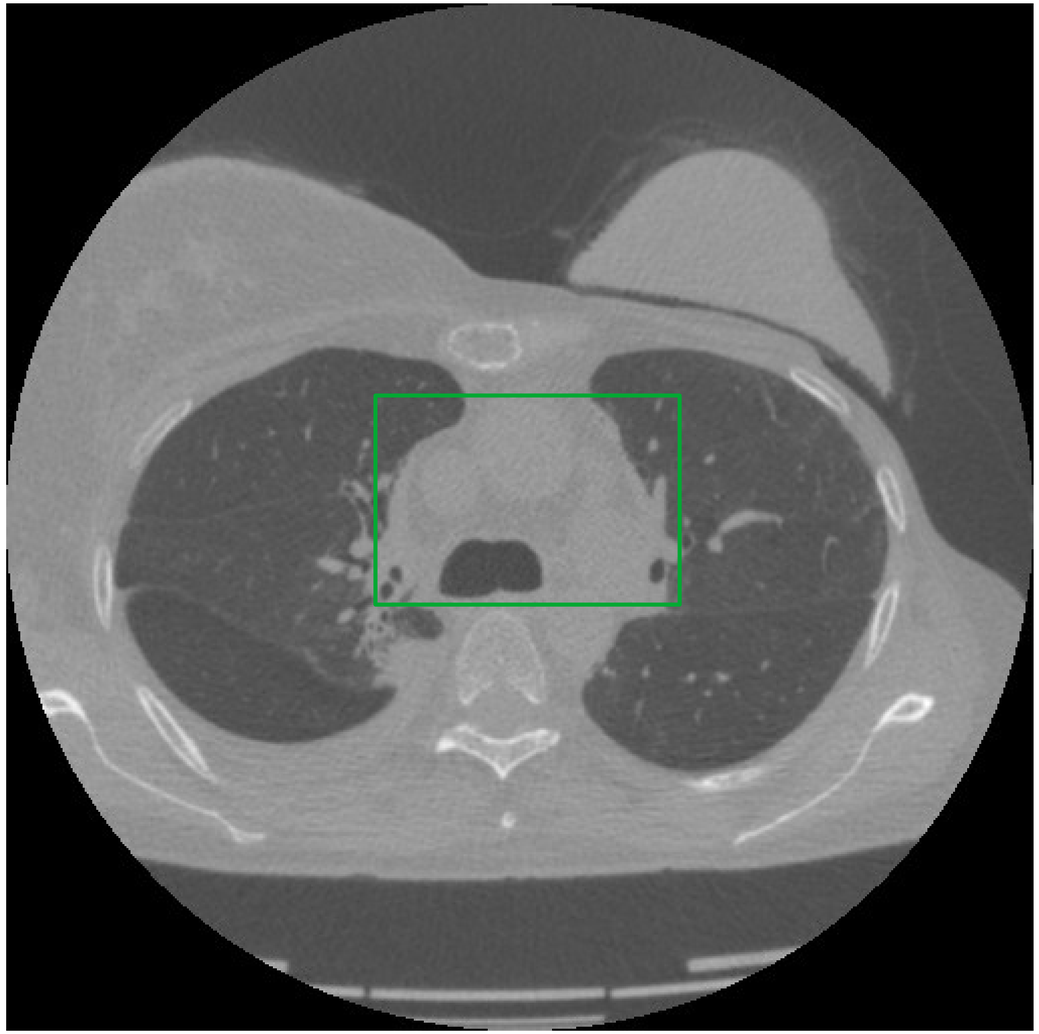
Нейросети, Webassembly

Происхождение темы доклада

Задача

- Имеется программа, сегментирующая аорту и анализирующая ветвление сосудов по КТ
- Задача сегментации аорты и её анализа - решена
- Но есть проблема с детектированием именно восходящей части аорты
- Это можно поправить заданием region of interest (ROI)

- Плюсы задания ROI:
- Восходящая аорта детектируется и сегментируется корректно
- Будет уменьшена исследуемая область — вычислительные затраты на сегментацию ветвления сосудов снизятся



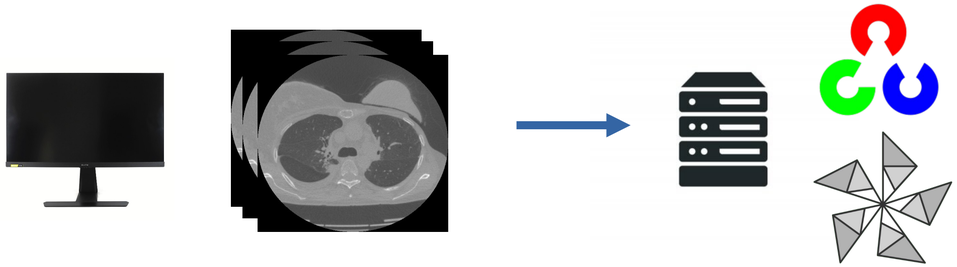
Как получить ROI?

Задавать руками в интерфейсе программы

- Это базовое решение проблемы

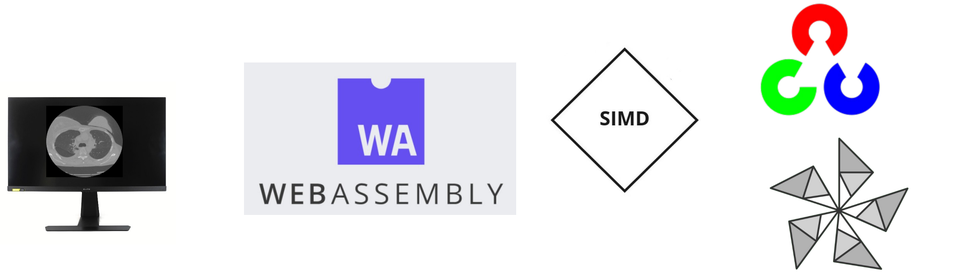
Сегментировать при помощи нейросети на сервере

- Долго слать данные на сервер

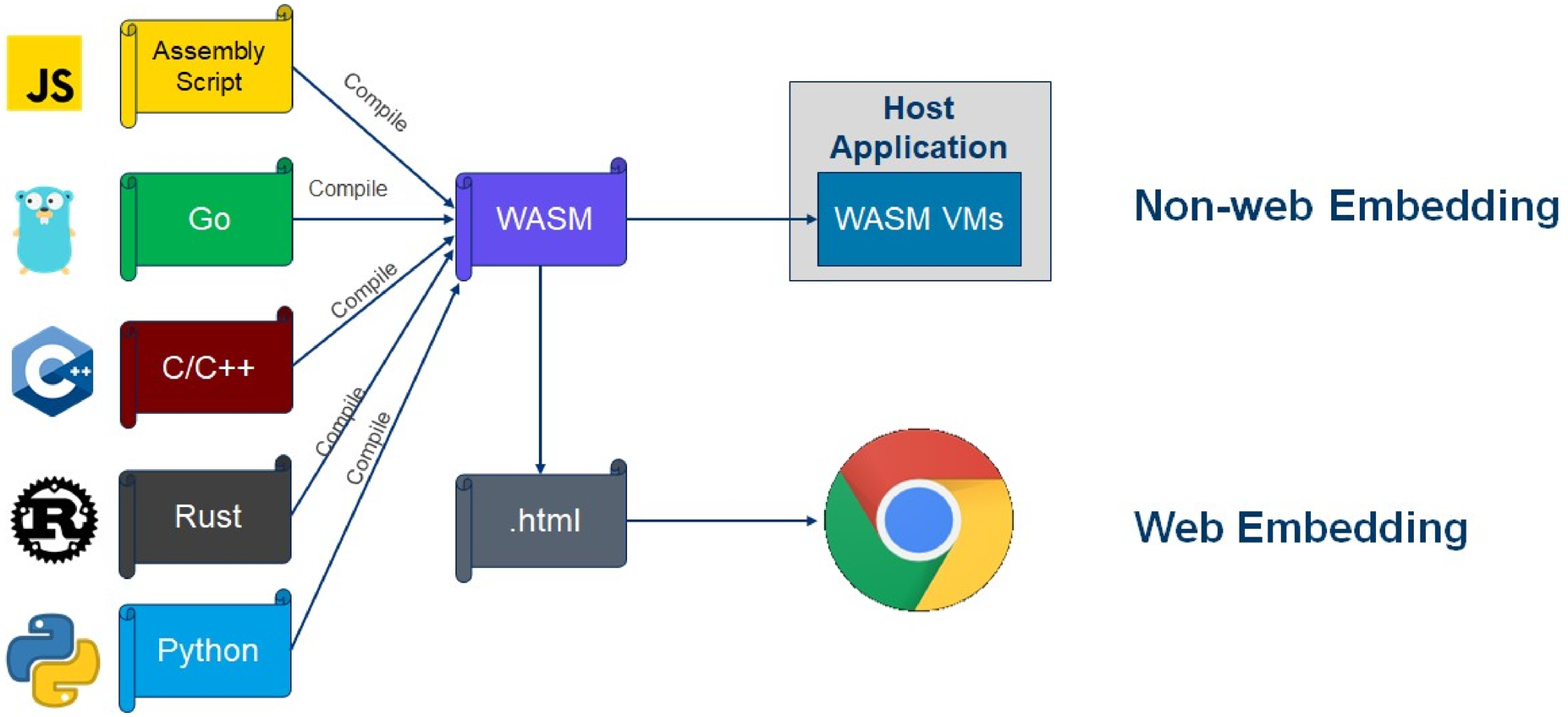


Сегментировать при помощи нейросети на устройстве

- У нас клиент-серверное приложение, переписывать на чистый десктоп затруднительно
- Webassembly — это что-то новое



Как это сделать на устройстве не переписывая программу по десктопную архитектуру?



Как такое собрать?

1) Имеем обычный cmake проект
2) Убеждаемся в том, что все библиотеки и зависимости подключаются статически. Иначе так и настраиваем
3) Скачиваем cmake toolchains emsdk и инициализируем его
4) Собираем все зависимости по отдельности при помощи emsdk
5)В финальный таргет добавляем bindings классов и функций, которые мы хотим использовать в js
6) При сборке делаем финального проекта так же используем emsdk и собранные им же статики
7) На выходе получаем *js, *.wasm, *.data

В стандартном рабочем случае, где у нас cmake, дописываем emcmake: emcmake cmake ..

А для make — emmake: emmake make

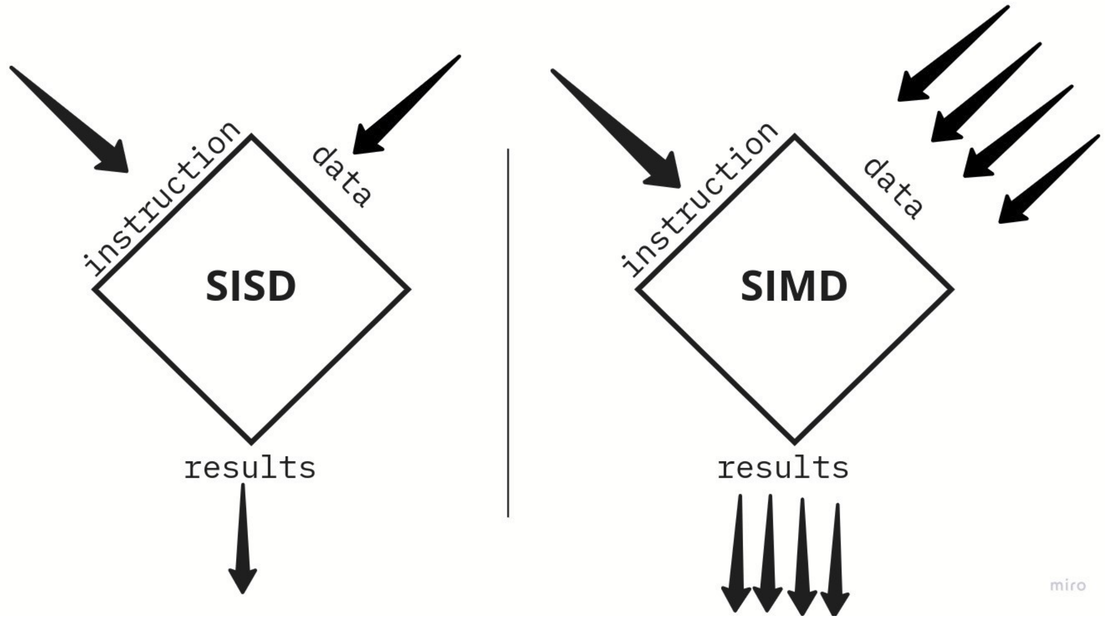
В итоге получаем примерно следующее:

```
source /emsdk/emsdk_env.sh && cd build && emcmake cmake ../src && emmake make
```

А будет ли это достаточно быстро?

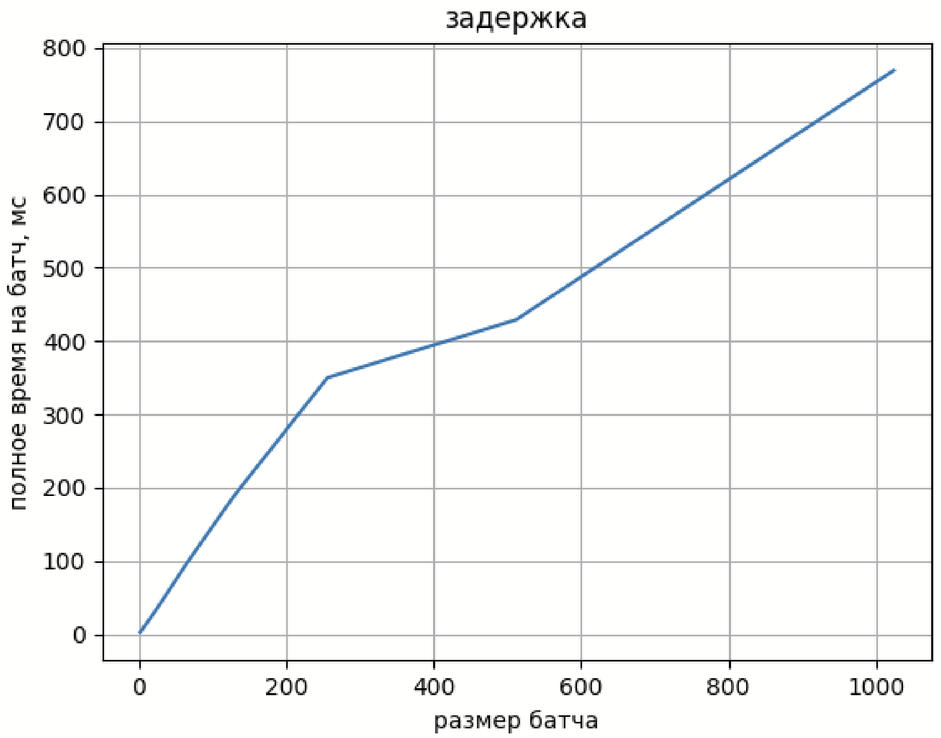
Для ускорения инференса пришлось:

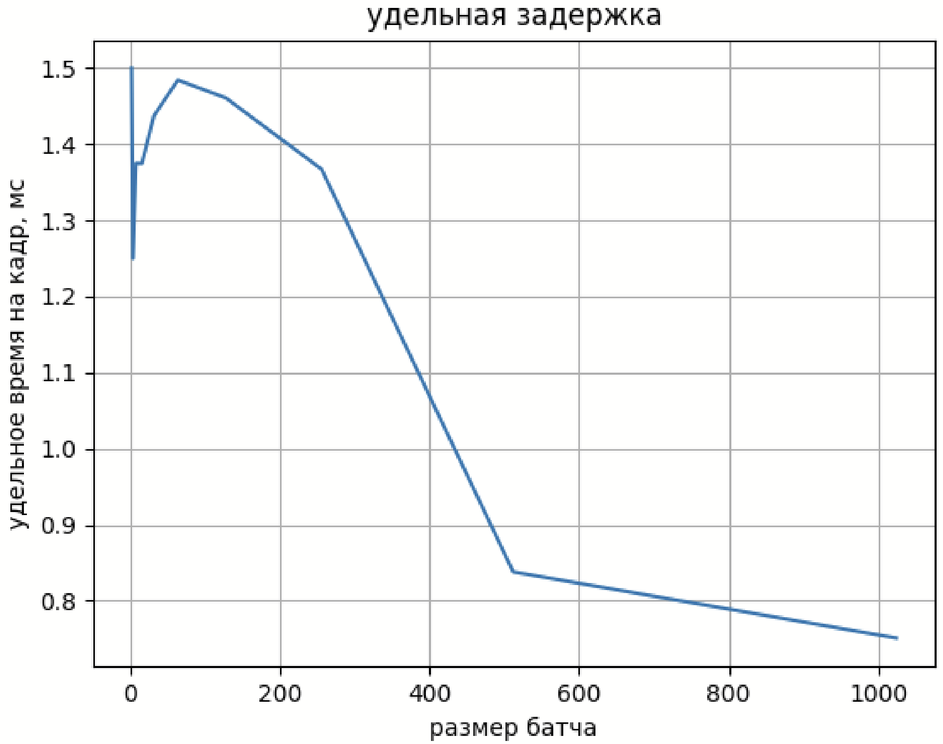
- Снизить разрешение картинки с 512*512 до 64*64
- Обрабатывать фотографии как можно большим батчем за раз
- Использовать урезанную сеть Mobilenet-v3
- Размер нейросети составил всего 210 КБ
- Порыться и найти флаги оптимизации и SIMD опций сборки



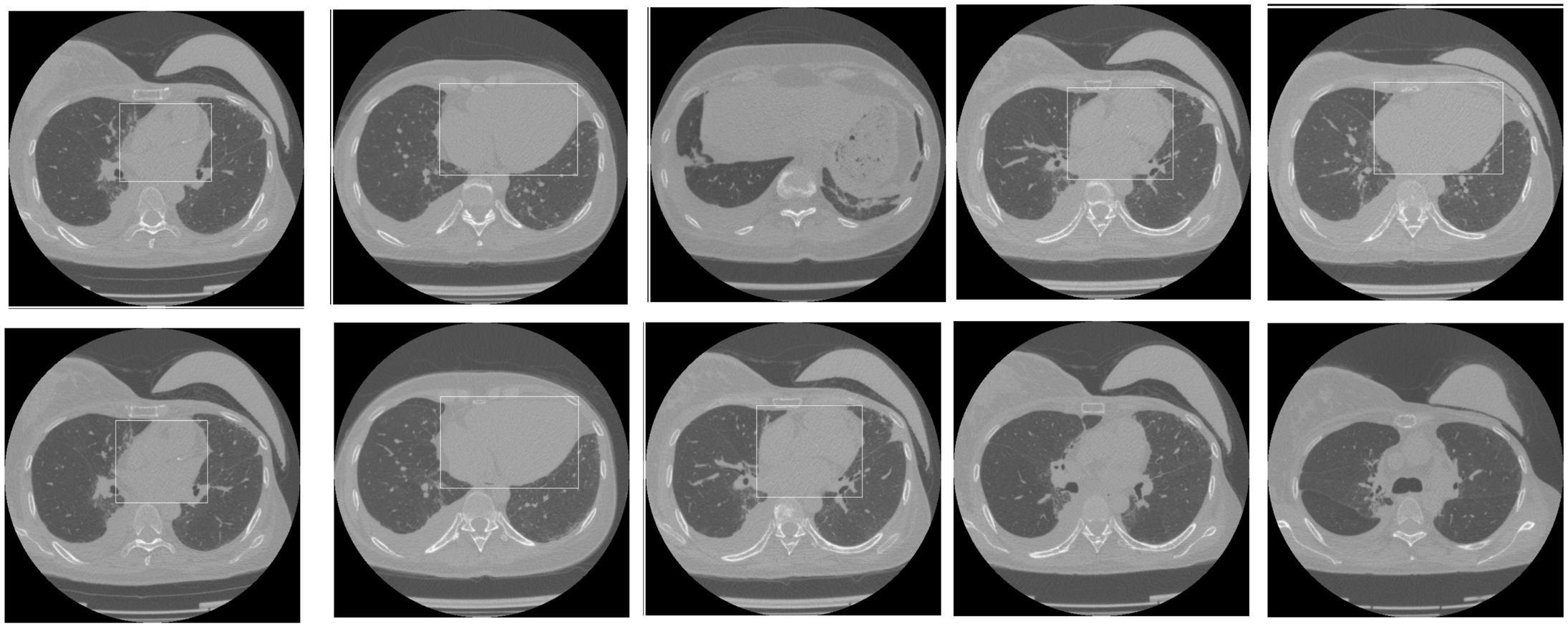
В итоге получили: 600-800 кадров за 2 секунды в сумме!

Замеры


задержка
полное время на батч, мс
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
размер батча


удельная задержка
удельное время на кадр, мс
0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
0
200
400
600
800
1000
размер батча

Примеры инференса



Материалы & контакты

Техническая статья на хабре «Как мы нейросеть в браузер тащили»: https://habr.com/ru/articles/723286/

Репозиторий с кодом: https://github.com/DmitriyValetov/onnx_wasm_example

Контакты:

- valetovdk@yandex.ru
- https://t.me/DmitriyValetov