Компьютерное зрение в Wildberries: поиск товара по фотографии и детекция «главного» объекта

Евстифеев Степан, Wildberries
Васильев Григорий, Яндекс

wildberries

О спикерах

wildberries

Степан Евстифеев

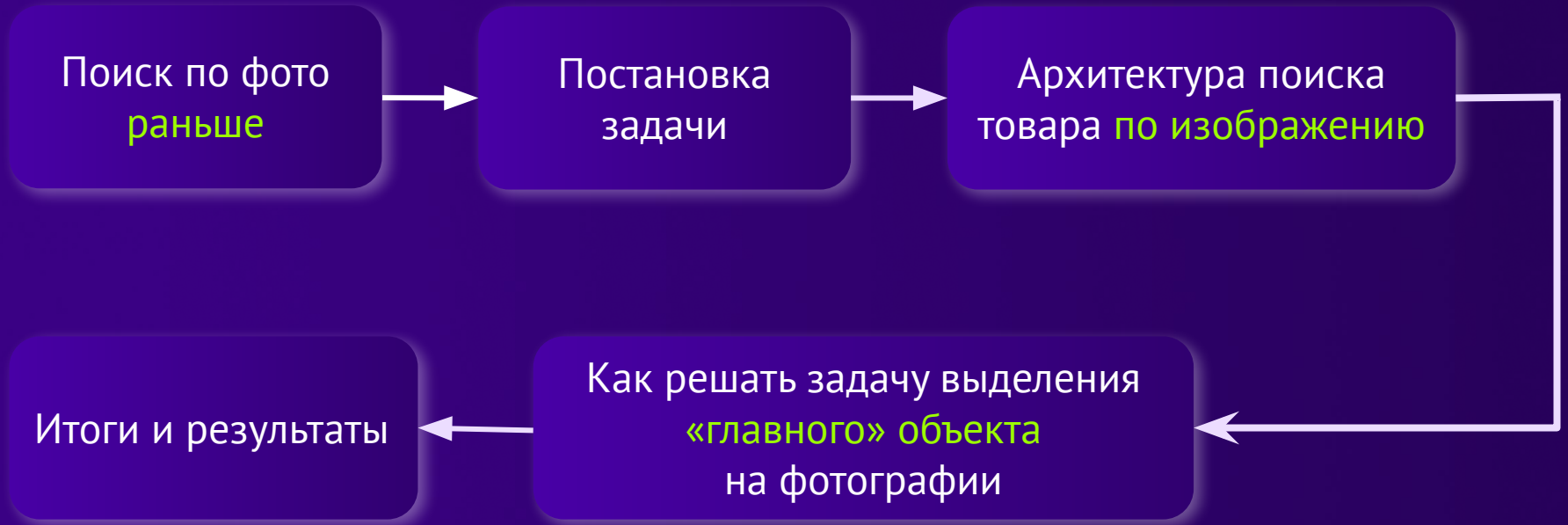
- Lead направления мультимодальных рекомендаций в Wildberries
- Магистр МГУ ВМК. Занимается Data Science с 2018 года
- До этого занимался компьютерными зрением в Сбере

Григорий Васильев

- Студент 4 курса МФТИ по направлению «Прикладная математика и информатика»
- ex CV-Engineer Wildberries
- Занимается ранжированием рекламы в Яндексе

План выступления

wildberries



wildberries

Поиск по фото раньше

Как работал поиск по фото

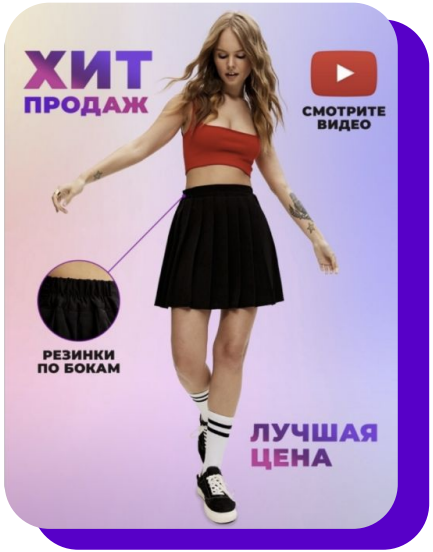
wildberries

Входное изображение.
Товар «юбка»

Сервис поиска визуально похожих

Выдача визуально похожих

Проблемы выдачи



Цвета одежды не совпадают

Несовпадение длины юбки

Общая композиция похожа, но товар не тот

Постановка задачи

Проблема и поиск решения

wildberries

Что есть?

В Wildberries реализованы алгоритмы поиска по фотографии. Но они учитывают контекст всей фотографии в карточке товара

Какая существует проблема?

Учитывается лишняя информация: поза, текст, композиция и фон изображения, что **влияет на качество** выдачи поиска

Что хотим получить?

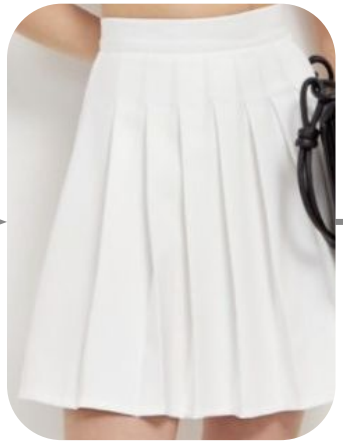
Выделение **главного товара** с фотографии, который и будем использовать для поиска

Поиск по фото и выделение главного объекта

wildberries

Алгоритм выделения главного объекта

Главный объект на фотографии — юбка

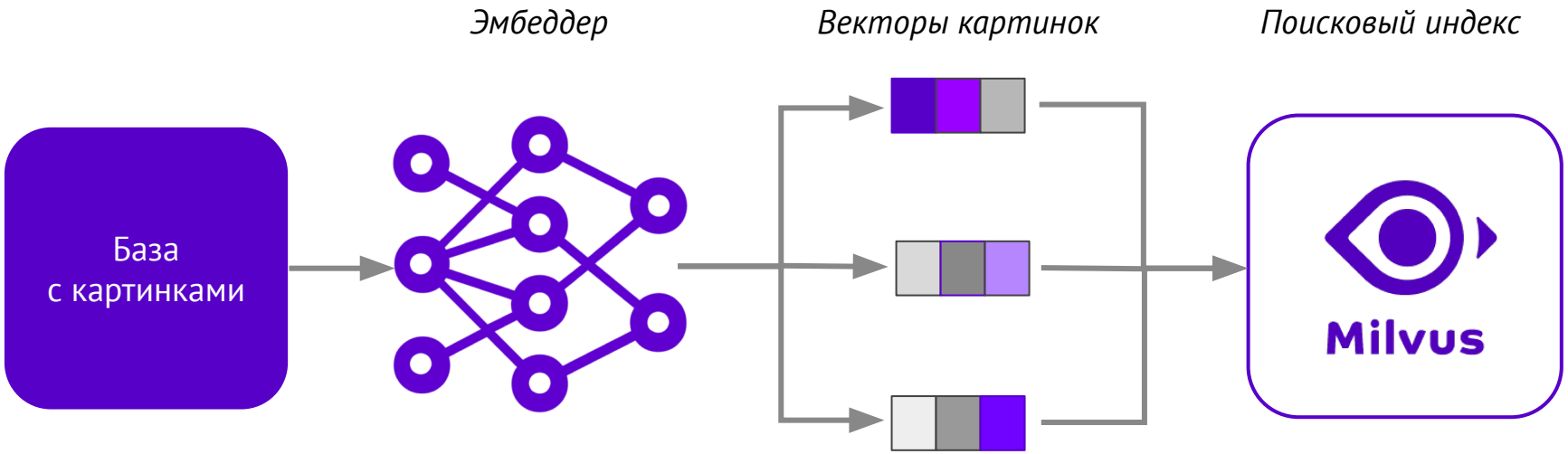


Сервис поиска визуально похожих

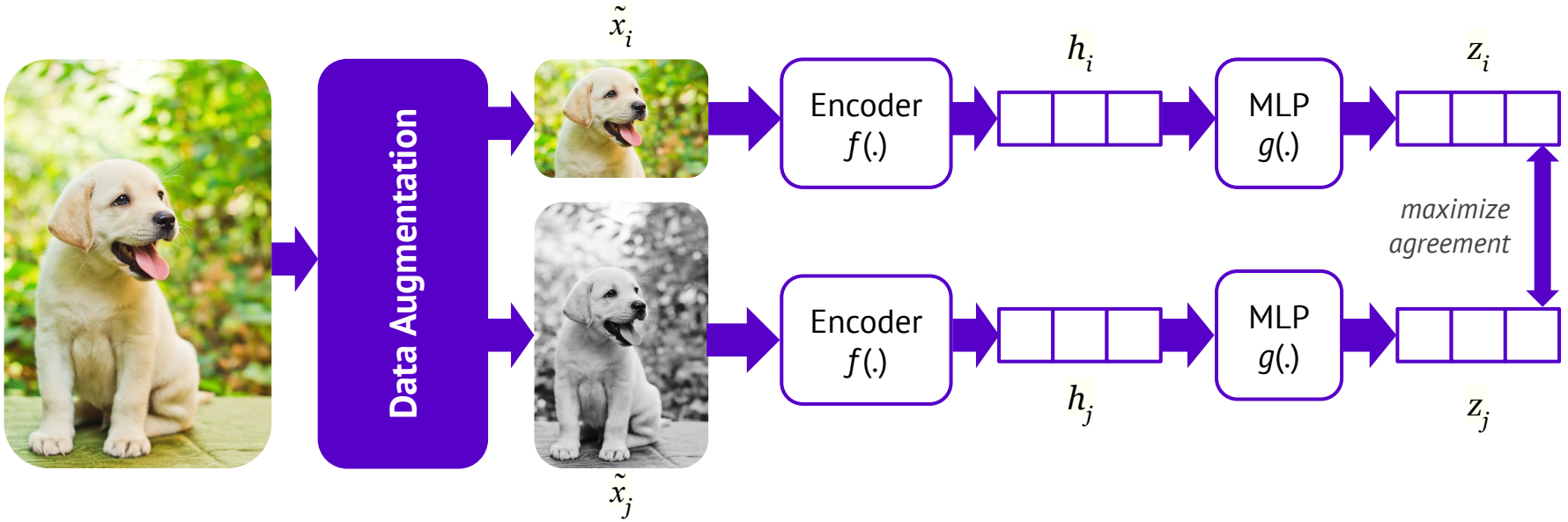
wildberries

Архитектура поиска товара по изображению

Архитектура поиска по фото



Архитектура поиска по фото. Эмбеддер



Процесс обучения визуально похожих товаров основан на процедуре из статьи MoCo v2

Архитектура поиска по фото. Обучение

wildberries

SwinV2

Визуальный энкодер

Вставка фона

Дополнительная аугментация

150 млн

Картинок в обучении

82 млн

Параметров

Архитектура поиска по фото.
Поисковый индекс

wildberries

Поисковый индекс:
Milvus (GPU)

Inverted File Index
для поиска

Квантуем векторы для
хранения

Количество картинок
в индексе: **160 млн**

Скорость поиска
похожих по 1 вектору:
200ms

wildberries

Выделение главного объекта

Выделение главного объекта через карточку товара

Карточка товара «юбка»

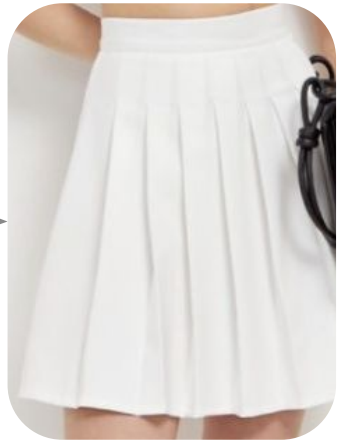
Детектор

Топ
Очки
Юбка
...
Обувь

Найденные классы

Категория карточки: **юбка**

Главный объект



Данные. Открытые датасеты

- Собрали и обработали почти все открытые датасеты с одеждой (самая продаваемая на WB категория товара)
- Классы датасетов мы отображаем в специально отобранные 13 классов, покрывающие 99% видов вещей, которые может надеть человек ;)

Открытые датасеты	Размер
DeepFashion (multimodal)	7к
DeepFashion (in-shop)	12к
Fashionpedia	50к
ModaNet	52к
Всего	121к

Данные. Label Studio

wildberries

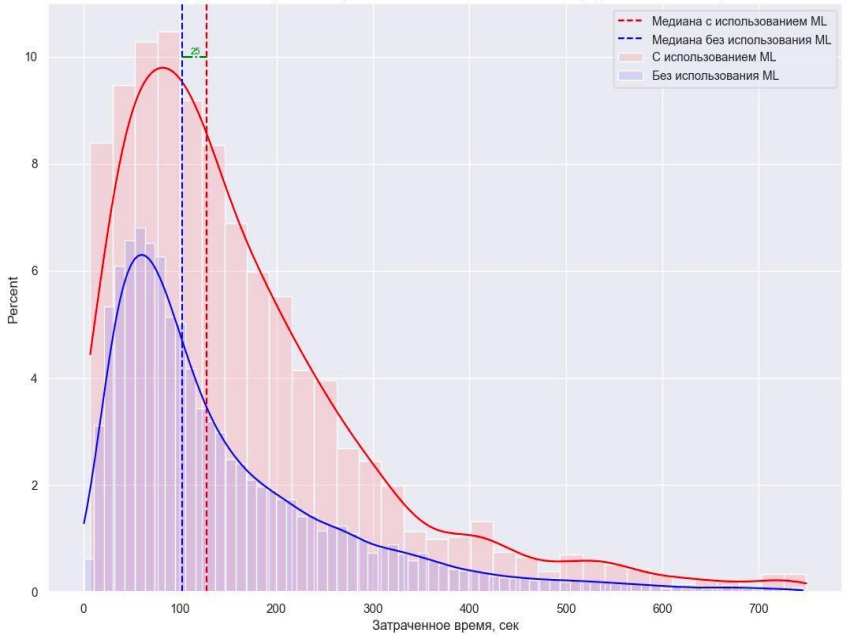
- В качестве платформы для разметки мы используем **Label Studio**
- Позволяет работать в разных доменах: текст, картинки, аудио
- Очень легко разобраться с UI асессорам
- Мы размечаем полигонами: так хоть и дольше, но зато качественнее!

Пример разметки в Label Studio

Данные. Авторазметка

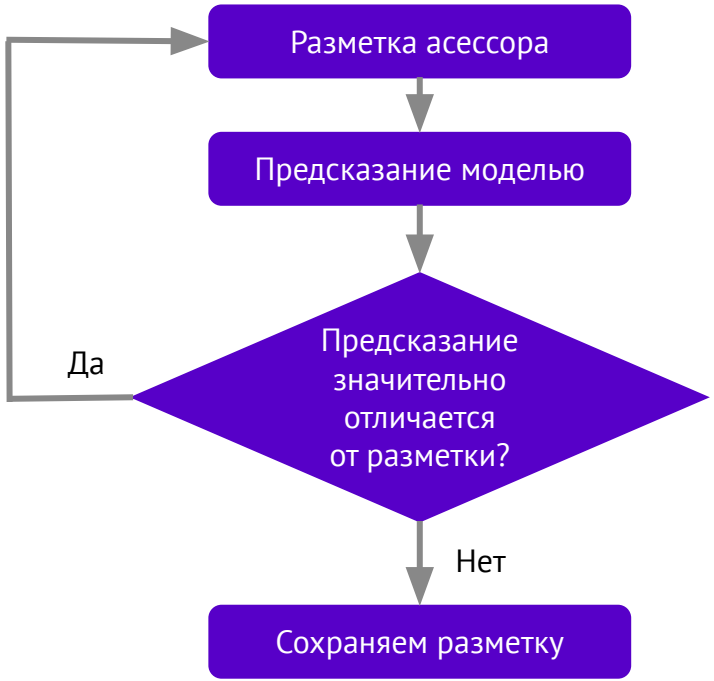
- Добавили кастомную разработку для **автогенерации разметки** с помощью ML
- Но в итоге это увеличило время разметки в среднем на 25 секунд!
- А дальше асессоры стали пользоваться авторазметкой все реже

Распределение затраченного времени в зависимости от количества предсказаний в аннотации



Данные. Active learning

- Используем своеобразный **active learning**
- Переразмечаем объекты, которые не совпадают с предсказаниями обученной модели
- В среднем **23%** из них оказываются действительно неправильно размеченными



Данные. Разметка. Инструкция

Важно писать очень подробную инструкцию!

Если не просить четко выделять границы объекта, то асессоры и будут размечать «на глаз»

Сначала было не очевидно, что одна часть товара может быть «разделена» другим

Здесь решили выделить лицо в качестве предмета одежды

Архитектура модели

wildberries

- В качестве основного детектора используется модель семейства **YOLO**
- Обученную модель конвертируем в **ONNX** — это единый формат для представления моделей для деплоя
- А уже ONNX конвертируем в **TensorRT** — это оптимизированный формат моделей на видеокартах Nvidia

Формат модели	Скорость работы*
ONNX (CPU)	2.679 s
TorchScript	0.00984 s
TensorRT	**0.00596 s**

**1 картинка 1024x1024*

Тестирование

wildberries

- Когда проект становится большим и в нем есть различные модули, то часто бывает так, что при изменении одного кусочка кода он может сломаться в самых неожиданных местах
- Для того, чтобы этого избежать мы в каждом проекте пишем тесты на **pytest**
- Удобно использовать **fixture** для создания сложных объектов
- А **mock** позволяют эмулировать поведение объектов (например, задать определенные логиты у модели)

Тестирование

wildberries

Юнит-тесты

Помогают отловить баги в реализации логики модулей

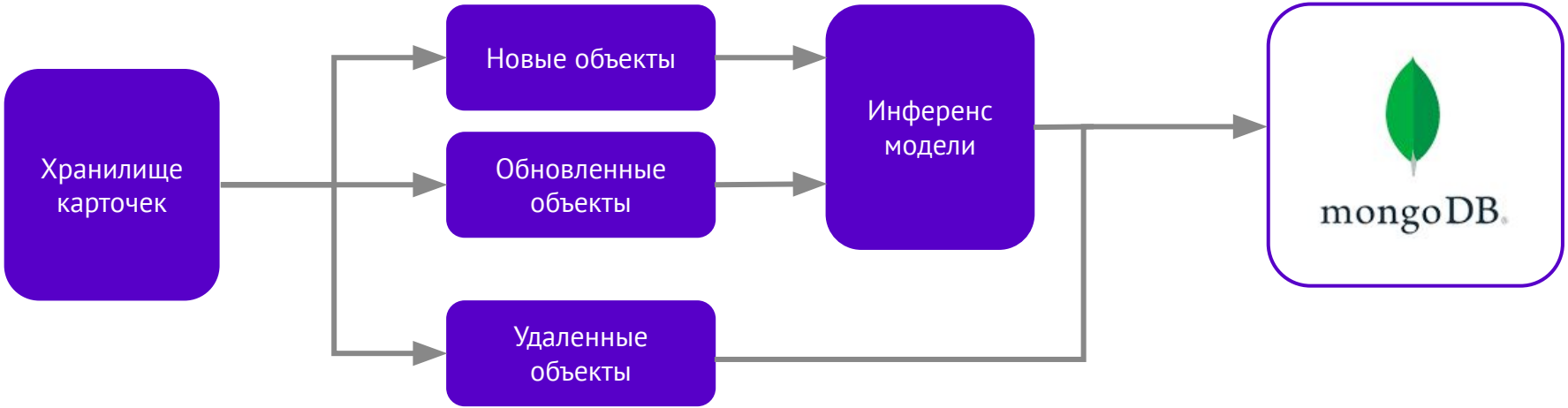
Как-то тест нам помог найти странный баг с TensorRT, который при большом батче не падает с ошибкой, а дублирует предсказание для первого объекта для всех картинок в батче

Интеграционные тесты

Запускаются для всех моделей, которые добавлены в проект — метрики на фиксированном датасете не должны быть слишком низкие

Интеграция с продом.
Оффлайн пайплайн

Для обучения модели поиска по фото нужно рассчитать предсказания детектора для всех товаров WB

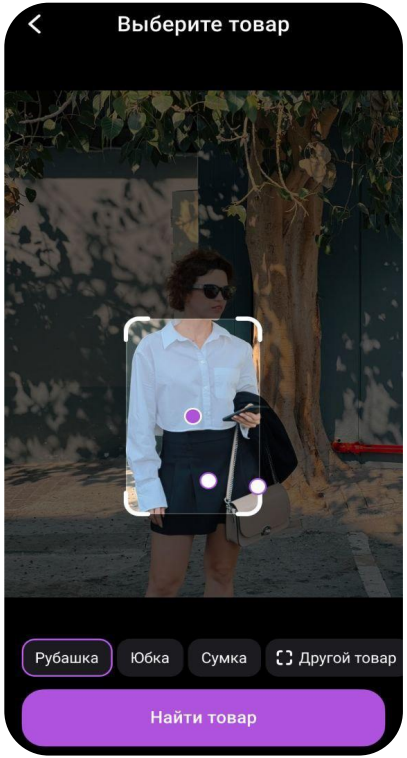


В качестве хранилища предсказаний мы используем MongoDB. Запускаем пайплайн раз в сутки

Интеграция с продом. API

wildberries

- Для помощи пользователю выбрать главный объект мы реализовали **API** и **UI**
- API реализован с помощью фреймворка **FastAPI**
- Для деплоя модели в онлайне мы используем веса модели TensorRT в инференс сервере **Triton**
- Он позволяет распределять нагрузку между GPU, а также эффективно делать инференс модели

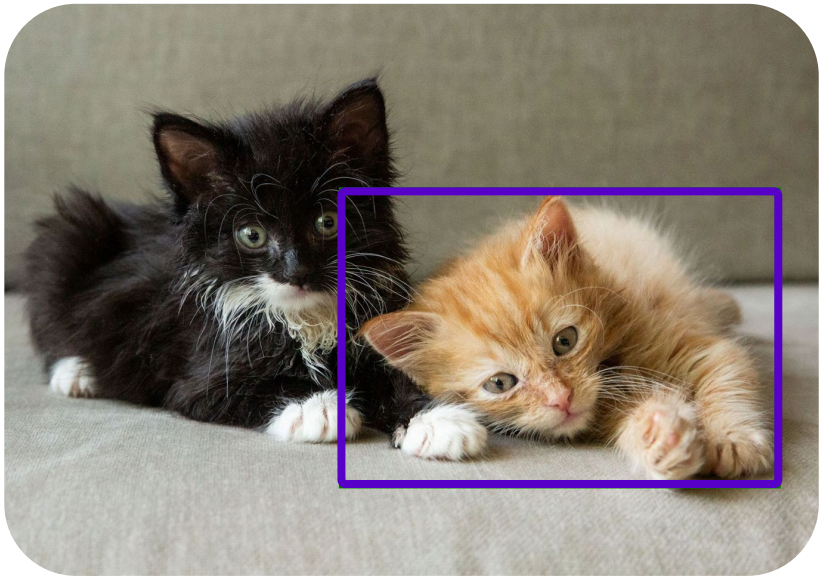


Дизайн интерфейса поиска по пользовательскому фото

Zero-shot detection

wildberries

- Модель работает только с одеждой, но хочется расширить её и на другие категории
- Один из самых простых способов это сделать — использовать **zero-shot детекторы**
- Они могут по текстовому запросу (промпту) найти интересующий объект на изображении
- Мы использовали 2 zero-shot детектора: **YOLO-World, OWLv2**



Пример работы open-set детектора.
В данном случае был промпт «right kitten»

Результаты

Оффлайн-метрики
Детектор

wildberries

Метрики моделей на тестовой части

Модель	MAP@0.5:0.95	F1 Weighted	FP ratio*
YOLOv5	0.71	0.789	0.62
YOLOv8	0.70	0.778	0.59
YOLOv8 + аксессуары	**0.74**	0.799	0.69
YOLOv8 + аксессуары + FP	0.73	**0.813**	**0.03**
YOLO-World	0.31	0.395	0.46
OWLv2	0.42	0.437	-

**посчитан на датасете, который не имеет предметов одежды*

Оффлайн-метрики
Поиск по фото

Модель	Precision@10	F1 score
SwinV2 (полные картинки)	0.05	0.02
SwinV2 (кропы)	**0.11**	**0.06**

A/B-тестирование

wildberries

Эксперимент	Revenue	Add to cart	View Item	Orders
Добавили возможность выбрать пользователю главный объект из найденных	+1.1%	+4.5%	+6.71%	-0.53%
Обучили энкодер на поиск кропов в кропах	+4.83%	+4.58%	+5.71%	+3.29%

Спасибо за внимание!

Telegram Евстифеева Степана: @minemile

Telegram Васильева Григория: @vasgreg

Блог WB TECH на Хабр

Telegram-канал WB Space

wildberries