



# Сервис по поиску домашних животных

22.10.2021

— Богомолов Кирилл, Данилов Максим, Скороходов Всеволод

МФТИ

## Содержание

1. Как использовать сервис
2. Объем задач, решаемый сервисом
3. Детекция животного
4. Классификация животного
5. Классификация породы собаки и длины ее хвоста
6. Определение цвета
7. Определение наличие хозяина
8. Улучшение качества вырезанной фото с помощью NN  
SuperResolution
9. Взаимодействие сервера с клиентом. React.js + Flask.py
10. Количественные результаты
11. Технические характеристики системы, на которой обучали



## Как использовать сервис

С сервисом можно ознакомиться по ссылке: <http://194.87.237.22:5000/>

1. Загрузить серию из фотографий
2. Нажать “Получить метки”
3. Подождать, пока нейросеть их обработает. Для ~10 фото обработка до 60 секунд

CSV файл с метками для тестового датасета лежит на гитхабе:

<https://github.com/kirrya95/leaders2021>



Лидеры цифровой трансформации  
Система распознавания животных  
Команда *cucumber*

Загрузите снимки с камер наблюдения  
После нажатия на кнопку вы увидите  
результаты по каждому снимку

Перетащите сюда файлы, либо нажмите  
для выбора

**Получить метки**

### Результаты распознавания

[Скачать csv](#)



open\_moscow\_cam\_531

Собака, цвет чёрный, с длинным хвостом, без хозяина

Топ-3 породы: Чихуахуа, Бульдог, Овчарка

Адрес камеры: Москва, ул.Первомайская д30к7

Время съёмки: 10.10.2021 в 13:15



open\_moscow\_cam\_531

Черная собачка, цвет чёрный, с длинным хвостом, без хозяина

Топ-3 породы: Чихуахуа, Бульдог, Овчарка

Адрес камеры: Москва, ул.Первомайская д30к7

Время съёмки: 10.10.2021 в 13:15

## Объём задач, решаемый сервисом

Мы решали наиболее общую задачу: по серии снимков с камер видеонаблюдения находится собака, её цвет, порода, длина хвоста, с хозяином/без хозяина.

Наш сервис умеет:

1. По серии снимков распознавать собаку
2. Определять:
  - а) цвет
  - б) топ-3 похожие породы
  - в) с хозяином / без хозяина
  - г) хвост: (длинный/короткий)
  - д) время съёмки, адрес камеры

## Технические подробности

### 1. Детекция животного

Для детекции мы попробовали несколько моделей: YOLOv5, Mask R-CNN ResNet-50 FPN, Faster R-CNN ResNet-50 FPN, Faster R-CNN MobileNetV3-Large FPN, RetinaNet ResNet-50 FPN. Но мы остановились на Faster R-CNN MobileNetV3-Large FPN ввиду ее скорости, качество детекции на всех моделях не сильно отличалось.

Мы заметили, что детектор не всегда корректно распознает объекты, но хорошо выделяют их, поэтому мы решили обрабатывать фрагменты с изображениями животных и скамейки.

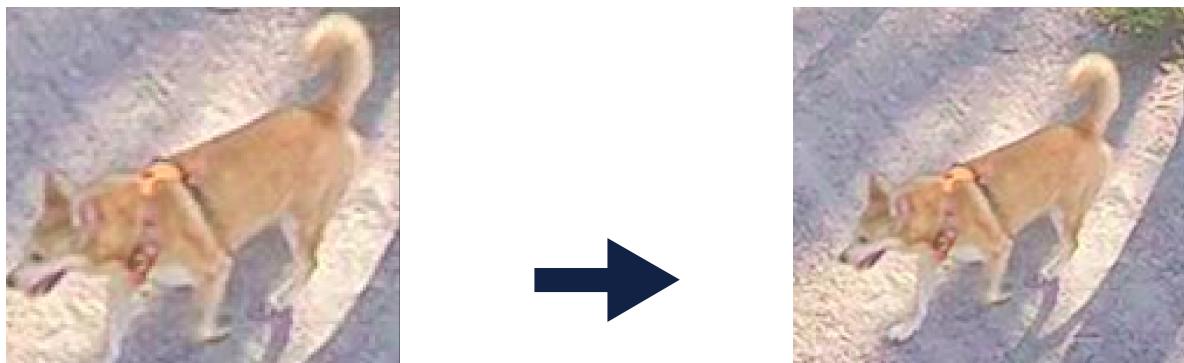
Одной из проблем стала то, что детектор несколько раз выделяет одну и ту же собаку. Для ее решения мы отсеивали слишком маленькие фрагменты и выбирали среди пересекающихся объект с максимальной площадью или с меткой "собака".



## 2. Классификация

2. В ходе проверки детектора на картинках из выданного датасета было замечено, что он детектирует собак не только к собакам, но к кошкам и (???) скамейкам и людям. Так что мы решили установить классификатор для проверки фрагментов изображения. В будущем возможно добавление автоэнкодера для более точно определения собаки.

Следующая проблема была в том, что детектор дает сильное ограничение на размеры детектируемых фрагментов, так что мы реализовали функцию, которая получает фрагменты изображения от детектора и увеличивает ширину окна в некоторое количество раз.



Далее мы поставили классификатор на основе ResNet101, использовали Transfer Learning (обучали классификатор на выходах обученной модели)([https://drive.google.com/file/d/10rwCjC8EXjTE3\\_AXWOsLSOG1YR053OQe/view?usp=sharing](https://drive.google.com/file/d/10rwCjC8EXjTE3_AXWOsLSOG1YR053OQe/view?usp=sharing)). Так как на картинках собаки, как правило, занимают небольшую часть, то вырезанные фрагменты имеют плохое качество, так что мы “подпортили” качество картинок обучающего датасета.

Для обучения мы сжали все картинки до 70x70. В качестве метрики выбрали accuracy. Качество модели на валидации - 0.9042. Датасет для обучения - Kaggle “DoGs vs. Cats” (<https://www.kaggle.com/c/dogs-vs-cats/data>)

## 3. Классификация породы

3. Далее необходимо классифицировать породы собак. Данные для обучения были взяты с Kaggle - Stanford Dogs Dataset (<https://www.kaggle.com/jessicali9530/stanford-dogs-dataset>). В этом датасете было 120 пород. Так что мы просмотрели породы и выкинули редкие и нечасто встречающиеся в России. В итоге осталось около 44 пород. Далее мы построили классификатор на основе ResNet50, обучили его последний слой и добавили несколько линейных

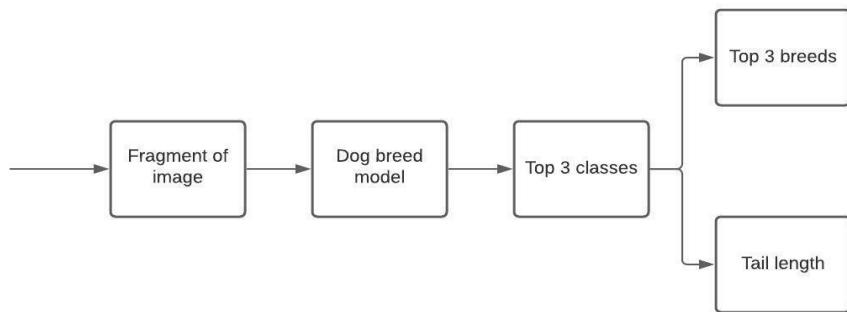
слев.(<https://drive.google.com/file/d/13N4ArAV3TRLIzhq5lcZZDq0oYIKCj3M/view?usp=sharing>)

Аналогично предыдущему пункту мы подпортили качество изображений. Так как классов достаточно много и качество картинок не очень высокое, мы решили брать топ-3 класса, которые выдает модель. В качестве метрики качества модели решили опять взять accuracy. На валидации качество модели - 0.7932.

Следующей проблемой было определение длины хвоста собаки. Мешали 2 фактора:

- 1) Субъективность определения длины
- 2) Плохое качество картинки и, как следствие, сложности в обнаружении хвоста
- 3) Отсутствие размеченного датасета

Для решения второго пункта мы решили определять длину хвоста по породе собаки(в дальнейшем возможно улучшение или отказ от этого метода). Что касается первого пункта, от субъективности при определении длины хвоста мы избавиться не могли, так что сами разметили данные по длине хвоста в зависимости от породы.



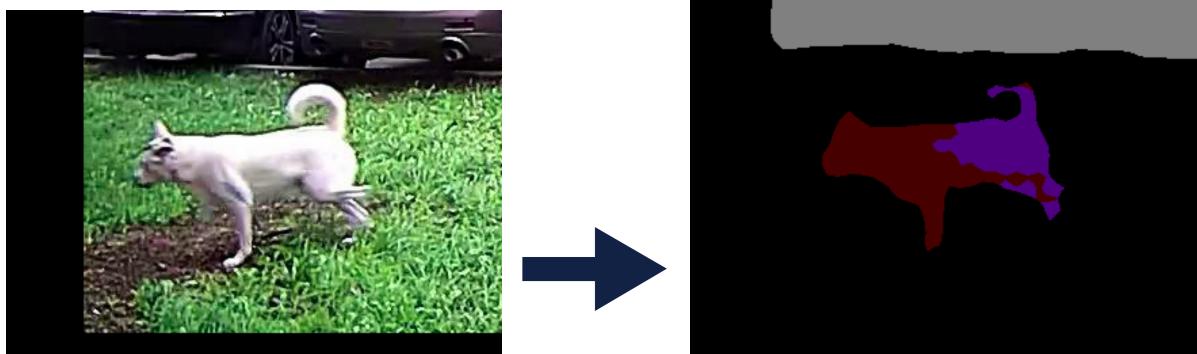
### 3. Определение цвета

Для определения цвета у нас было несколько идей.

- a) Мы хотели выделить границы собаки на изображении с помощью Gradient Border и просто посчитать средний цвет выделенного контура. Однако пока от такой идеи пришлось отказаться. Во-первых, качество картинок довольно низкое, поэтому Gradient Border плохо находил границы, либо вообще не находил. Во-вторых, опять таки из-за плохого качества Gradient Border мог выделять границы других объектов(скорее всего, потому что эти объекты имели более простую форму и, соответственно, их можно было выделить), например, заборов, что тоже приводило к неправильному определению цвета.

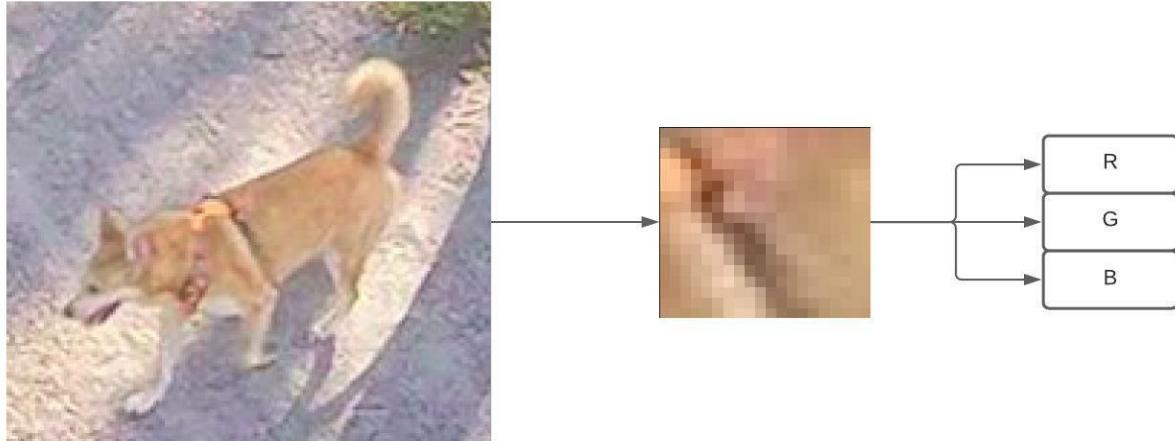


6) Мы пробовали решать это через семантическую сегментацию. Посмотреть на выделенную собаку и пройтись по этому фрагменту и уже там посчитать цвета.



Однако тут опять появились проблемы: из - за низкого качества ему проблематично распознавать собаку, так что он одну и ту же собаку мог на несколько разных выделить. К тому же мы не учитываем таким образом участки с тенями, которые сильно влияют на среднее значение. Использовали сегментационные модели: DeepLabV3 MobileNetV3-Large, DeepLabV3 ResNet101, FCN ResNet50.

в) Третий вариант был более простой, но и надежнее. Мы решили его оставить. Так как детектор выдавал нам фрагменты изображения с собаками, то всего скорее, если выделить небольшой квадрат в центре, то он будет полностью принадлежать собаке. Конечно, возможно такие выбросы, как тень или то, что собака пятнистая или разноцветная, поэтому возможно в дальнейшем придется эту модель улучшать. После получения этого квадрата, усредняем каждый из каналов RGB.



Далее нужно было по RGB описанию вернуть цвет. Мы взяли самые популярные и известные цвета и определяли ближайший к полученному RGB цвет. Как оказалось использовать евклидово расстояние не очень хорошо в силу различного восприятия человеком различных цветов. Поэтому для подсчета расстояний мы пользовались библиотекой colormath.

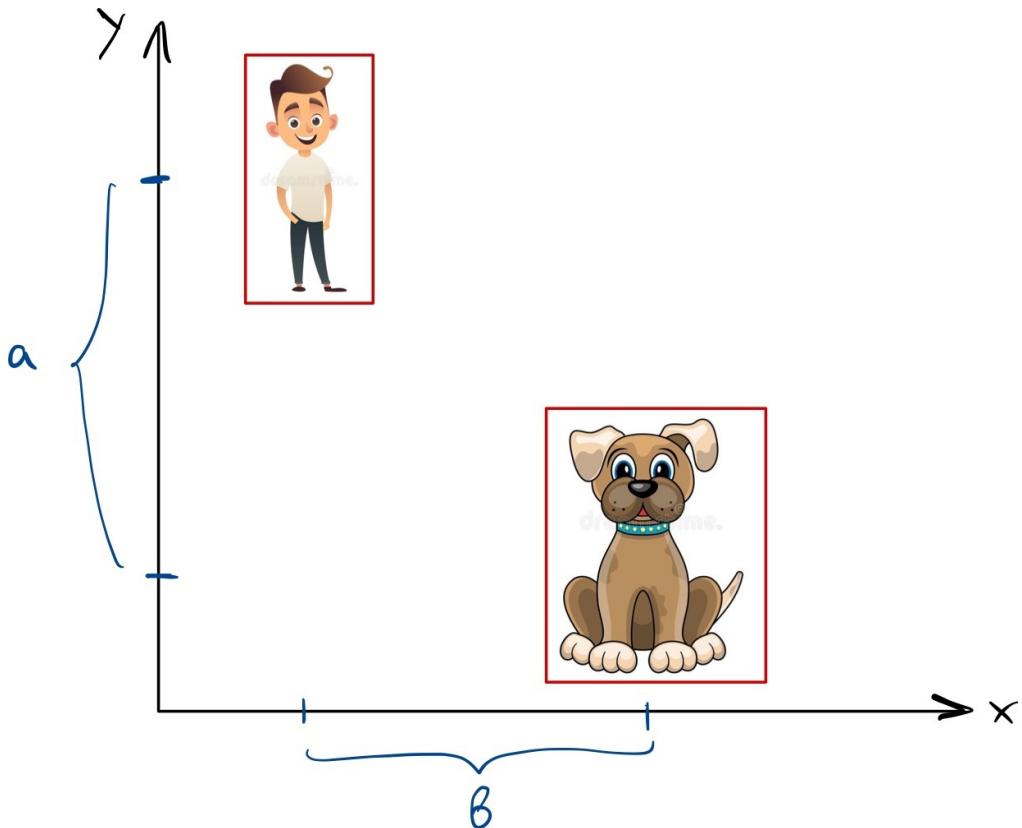
## 5. Наличие хозяина

Для определения наличия хозяина на фотографии необходимо было определить есть ли на фото человек. С этой задачей хорошо справлялся детектор, так как человек достаточно большой, чтобы успешно его детектировать и классифицировать. За неимением лучшего мы пытались измерять расстояния между людьми и собаками. Из-за перспективы проблематично вычислять его точно, поэтому здесь мы ограничились лишь качественными оценками.

Помимо непосредственного вычисления расстояния между центрами прямоугольников (назовём расстояние  $a$ ), в которых находятся собака и человек, мы это делаем на нормировочный коэффициент равный высоте человека, так как в среднем человек в несколько раз по размеру превышает собаку. Значит если он оказался маленьким относительно собаки, то он находится далеко и, вероятно, не является хозяином данной собаки.

Теперь разберёмся с горизонталью. Мы заметили, что в среднем собаки "шире" человека. Поэтому в данном случае в качестве нормировочного коэффициента выступит ширина собаки (это связано ещё и с тем, что, как правило, маленькие собаки отходят от хозяина на меньшее расстояние). (Расстояние по горизонтали назовём  $b$ )

Из двух полученных величин берём максимальную и сравниваем с какой-то предельной. Мы решили взять предельную равную 3,0.



####добавить в main.py, потом допишу

## 6. Улучшение качества фрагмента изображения

Мы решили, что на сайте было бы хорошо выводить фрагмент изображения, на котором обнаружена собака. Однако, как было видно выше, картинки имеют достаточно плохое качество. Чтобы визуально смотрелось лучше, мы решили улучшить качество картинки с помощью NN Super Resolution ([https://github.com/Saafke/FSRCNN\\_Tensorflow/blob/master/models/FSRCNN\\_x4.pb](https://github.com/Saafke/FSRCNN_Tensorflow/blob/master/models/FSRCNN_x4.pb)).

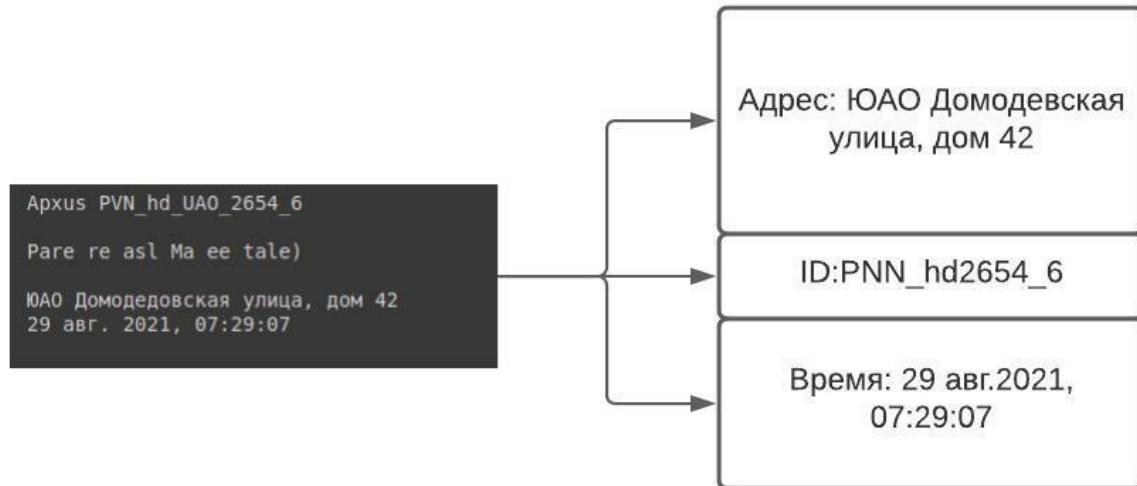


## 7. Определение времени съемки и адреса камеры

Для определения времени и адреса мы решили считывать информацию непосредственно с фотографии, для этого была необходима модель для выделения текста с картинки. Мы решили воспользоваться OCR из opencv, однако он работает достаточно медленно (~30 сек на одной картинке), поэтому мы решили отказаться от этой модели.

Далее мы решили воспользоваться функциями библиотеки pytesseract. Они хоть и используют более простые методы, работают быстро и достаточно хорошо для решения нашей задачи. Так как камеры имеют примерно одинаковый формат, для извлечения необходимых нам меток (время, адрес и ID камер) из детектированного текста, мы воспользовались регулярными выражениями.





## 9. Взаимодействие сайта с сервером и моделью

Front-end написан на React.js. Файлы загружаются в форму и отправляются на сервер.

Сервер, работающий на Flask, принимает этот пост-запрос, запускает модель (обученную с весами) и отправляет в качестве ответа JSON на сайт. Далее этот JSON парсится на сайте и выводятся результаты пользователю

## 10. Результаты

Точность распознавания собаки – **80%**

Определение цвета: **90%**

Хвосты: точность **80%**

Определение хозяина: **80%**

## 11. Технические характеристики системы

Обучали на Гугл колабе в основном, поэтому довольно сложно определить истинную конфигурацию машины.