Задание 1. Про здания и автомобили

Исходный код можно найти в репозитории по ссылке.
Веса для моделей можно получить самостоятельно, обучив модели или воспользоватся готовыми весами Ссылка на Я.диск

Препроцессинг и подготовка данных

На основе данных из XML файлов сделал бинарные маски для каждого класса отдельно в формате tif. На одном из снимков как здание была отмечена часть дороги, удалил этот полигон из маски.

Верхний диапазон значений пикселей у изображений был 2047, а нижний отличался и варьировался в пределах от 159 до 0 для тренировочных картинок, и 160-0 для тестовых.

Позже было замечено что если привести все значения в одинаковый интервал качество предсказания улучшится. Я приводил к интервалу 2009 - 159. (Возможно верхнюю границу изменять не стоило)

Для каждой пары изображение-маска из обучающей выборки, из одинаковых для изображения и маски рандомных мест я вырезал 50 кропов размером под вход и выход сети (256 x 256 пикселей), формируя тем самым очередь для обучения.

1/5 из этого набора отбиралась для валидации. На остальных производилось обучение сети.

Ближе к середине соревнований добавил аугментацию - отражение по вертикали и горизонтали. (Еще нужно было добавить повороты на углы кратные 90 градусам).

и усреднял значения сети, обученной на изображениях без отражений и сети обученной на изображениях с отражением. Визуально результат предсказаний казался хуже (появлялись разные артефакты), но на лидерборде это повышало скор, поэтому было решено оставить это решение.

Обучение нейросети

В качестве архитектуры нейросети я выбрал U-net.
Обучал сеть на кропах размером 256 на 256. Для каждого класса обучалась своя сеть.

Предсказания сети

“При обучении на кропах сильнее проявляется краевой эффект — сеть менее точно предсказывает на краях изображения, чем в областях более близких к центру (чем ближе к границе точка предсказания, тем меньше у сети информации о том, что находится дальше).” [1]

Я предсказывал маску на фрагментах с перекрытиями и отбрасывал области на границе. У каждой предсказанной маски по краям обрезалось по 50 пикселей.

Исходная картинка размером 1500 на 1500 расширяется с краев полосками изображения (нужно было их отзеркалить, но вспомнил я об этом поздно. Также можно было использовать np.pad, а не изобретать велосипед)

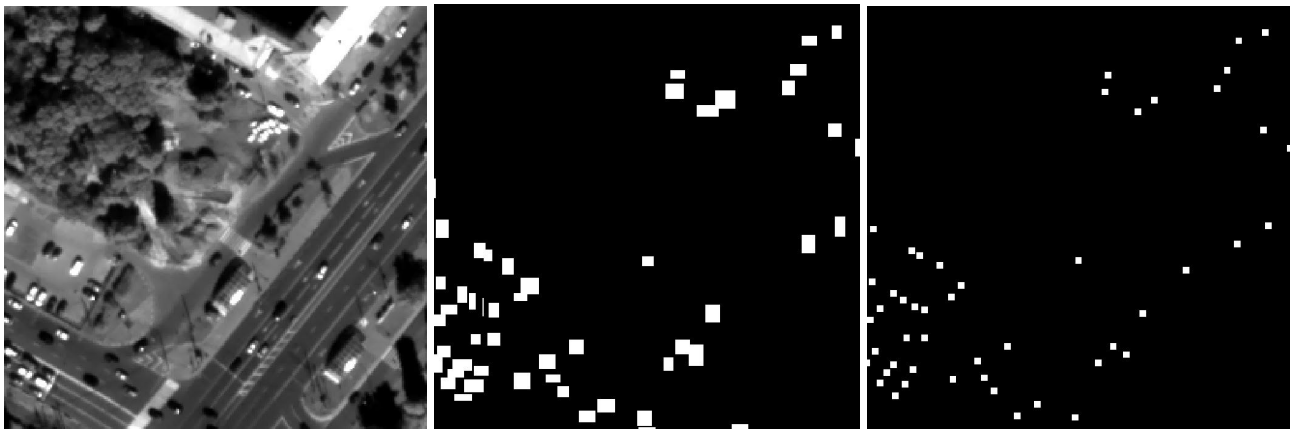
Эта картинка режется на пересекающиеся куски 256x256, с 50 пиксельным перехлестом, каждый кусок предсказывается. У каждой предсказанной маски по краям обрезалось по 50 пикселей. и добавляется обратно в большую картинку. Тут стоило оптимизировать код, чтобы нарезанные патчи собирались в один большой батч, предсказывались и потом разбирались обратно в большую картинку. А не дергать сеть заставляя предсказывать по одному патчу.

Нейросеть выдает вероятности того, что в конкретном пикселе находится каждый класс. Соответственно, чтобы получить бинарную маску, нужно подобрать порог, я подбирал визуально оценивая результаты бинаризации с разными порогами.

Предсказанные бинарные маски в формате tif для каждого класса в отдельности, собирались в один png файл. (Машины в нулевой канал, здания во второй, третий канал оставляем пустым)

Подсчет числа автомобилей.

Для подсчета числа автомобилей были сгенерированы маски, где в центр каждой машины ставился квадрат 4 на 4 пикселя. В большинстве случаев это позволило разделить плотно стоящие автомобили.
Далее, так же как и при сегментации, обучалась сеть. Делали предсказания для каждого снимка. В итоге получены маски с предсказанными центрами автомобилей
Подсчет выполнялся с помощью **skimage.feature.blob_log**



Фрагмент исходного изображения | Исходная маска | Преобразованная маска

Была попытка улучшить предсказания используя эти данные обучив на них sklearn.ensemble.ExtraTreesRegressor.
В качестве фичей были взяты предсказания blob-detection, сумма вероятностей, сумма пикселей выше определенной вероятности, целевая переменная - известное нам количество машин. Это не принесло повышение скора на публичном лидерборде (что странно, возможно плохо подобранные параметры регрессора), поэтому от этого решения я отказался.

Как запустить код

Исходный код можно найти в репозитории по ссылке.

Примечание:

Скачайте исходные данные для каждой задачи с сайта и разместите их в соответствующих каталогах.

Все папки, в которые производится запись и обращение, должны быть созданы заранее, нет проверок на существование папок.

Не все пути объявляются через переменные, где-то путь прописан прямо в коде.

1. Получим маски для каждого класса отдельно в формате tiff.
   Создать в masks папки masks_build_train и masks_car_train
   Запустить draw_xml.py TYPE='car' TYPE='building'
   Получим маски для каждого класса отдельно в формате tiff.

2. Привести границы интенсивности пикселей для каждого изображения к одинаковым значениям.
   В папке tif создать папки tif_train_percentile и tif_test_percentile. Запустить percentile.py
3. запустить create_crop_img для пар изображение + маска машин и изображение + маска зданий. Если хотим добавить аугментации отражением, запустить create_crop_img_flip

4. Запустить обучение train.py. Внимательно указать пути к ранее созданным npy-файлам и папкам, где будут сохраняться логи обучения и веса
5. Запустить предсказание predict.py или (predict_flip.py). Указать путь до весов, полученных ранее при обучении
6. Перевести в png. 1band_tif_to_3band_png.py. Изменить TYPE='car' или TYPE='building', в зависимости от того, с каким классом работаем.
6. Собрать два класса в один png. merge_pred.py
7. Подсчет количества автомобилей
Создать маски с точками в центре автомобилей. create_car_masks_with_point_4x4.py
8. Создать npy-файлы, обучить сеть, сделать предсказания аналогично как в задаче сегментации.
9. Запустить count_car_point.py. Сгенерируем csv со столбцами id и count.

1. https://habrahabr.ru/company/tinkoff/blog/342028/