

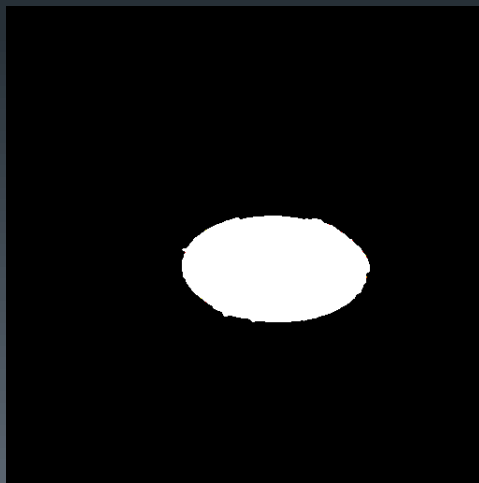
Отчет по практике на тему «Детектирование люков на дорожном полотне»


Подготовил студент группы БВТ1901
Перевозчиков Сергей

Поставленная задача и ее применение

Цель работы: обучить модель нейронной сети находить и выделять на изображении или видео люк (люки) или выдавать черно-белую маску с его (их) расположением на ней.

Возможное применение: распознавание люков, как и других дефектов дорожного полотна может применяться в различных дорожных службах, которые следят за состоянием дорог.



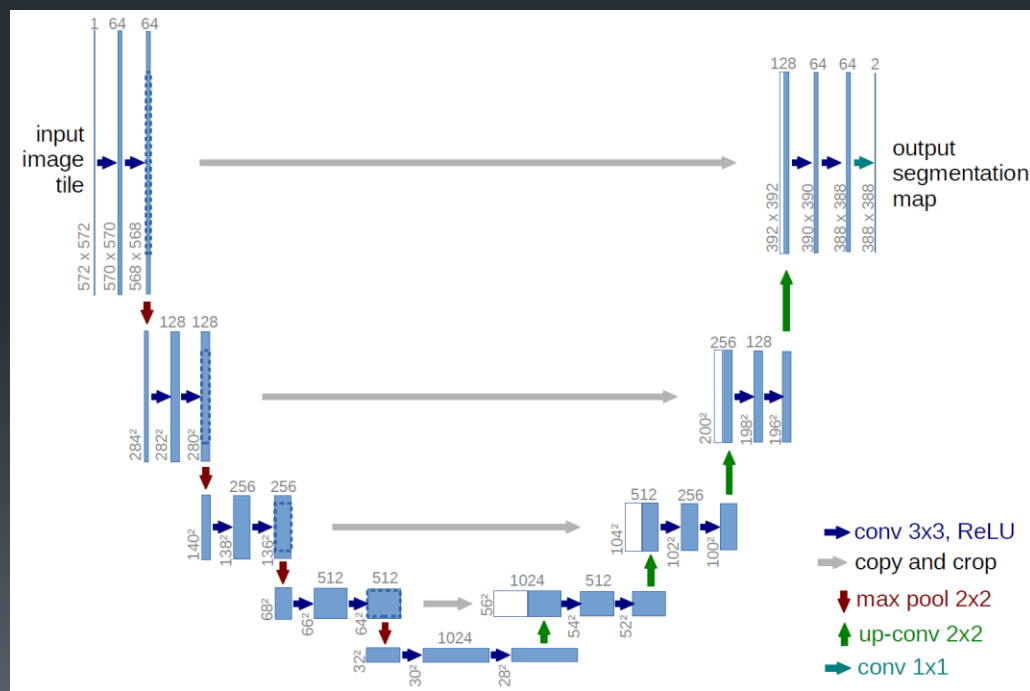


Использованные инструменты для разработки и библиотеки

- Python 3.7.5
- Модули для Python: TensorFlow 2.3.0, Keras 2.4.3, opencv-python 4.4.0, NumPy 1.18.5
- Visual Studio 2019
- Jupyter Notebook
- VIA

Использованные архитектуры

Для задачи распознавания изображений применяются сверточные нейронные сети (CNN) с глубоким обучением, мною была произведена попытка обучения модели Mask-RCNN и обучена модель U-Net



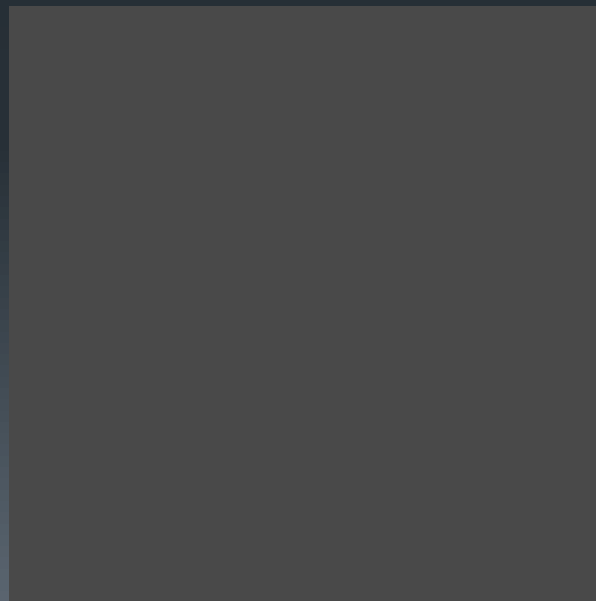
Разметка данных для обучения и тестирования

С помощью методы библиотеки OpenCV для Python и инструментария программы для аннотации изображений VIA (VGG Image Annotator), был подготовлен датасет из 30 различных изображений и соответствующих им масок, приведенных к одному размеру, с 1 или несколькими люками, и с помощью методов аугментации в Keras увеличено число изображений и аннотаций для них до 20 тысяч



Обучение и тестирование модели

По неопределенным техническим причинам (возможно, недостаток видеопамяти или требование определенных версий библиотек для Python), обучить модель Mask-RCNN не вышло. Модель U-Net обучилась с точностью 87.6% и потерями 0.4644, но при тестировании оказалась не способна выдать маску с расположением люка. Причиной может быть как недостаток самих изображений при обучении, так и не достаточная точность разметки.



Результаты обучения и тестирования

Found 30 images belonging to 1 classes.

Found 30 images belonging to 1 classes.

Epoch 1/5

2/2000 [.....] - ETA: 6:57 - loss: 1.2839 - accuracy: 0.1484WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time (batch time: 0.1199s vs `on_train_batch_end` time: 0.2880s). Check your callbacks.

2000/2000 [=====] - ETA: 0s - loss: 0.6588 - accuracy: 0.8751

Epoch 0001: loss improved from inf to 0.65881, saving model to unet_trapdoors.hdf5

2000/2000 [=====] - 930s 465ms/step - loss: 0.6588 - accuracy: 0.8751

Epoch 2/5

2000/2000 [=====] - ETA: 0s - loss: 0.5958 - accuracy: 0.8762

Epoch 0002: loss improved from 0.65881 to 0.59578, saving model to unet_trapdoors.hdf5

2000/2000 [=====] - 971s 486ms/step - loss: 0.5958 - accuracy: 0.8762

Epoch 3/5

2000/2000 [=====] - ETA: 0s - loss: 0.5435 - accuracy: 0.8762

Epoch 0003: loss improved from 0.59578 to 0.54349, saving model to unet_trapdoors.hdf5

2000/2000 [=====] - 981s 491ms/step - loss: 0.5435 - accuracy: 0.8762

Epoch 4/5

2000/2000 [=====] - ETA: 0s - loss: 0.4999 - accuracy: 0.8763

Epoch 0004: loss improved from 0.54349 to 0.49991, saving model to unet_trapdoors.hdf5

2000/2000 [=====] - 1010s 505ms/step - loss: 0.4999 - accuracy: 0.8763

Epoch 5/5

2000/2000 [=====] - ETA: 0s - loss: 0.4644 - accuracy: 0.8762

Epoch 0005: loss improved from 0.49991 to 0.46443, saving model to unet_trapdoors.hdf5

2000/2000 [=====] - 1034s 517ms/step - loss: 0.4644 - accuracy: 0.8762

Out[2]: <tensorflow.python.keras.callbacks.History at 0x1b206c87808>

2/30 [=>.....] - ETA: 1sWARNING:tensorflow:Callbacks method `on_predict_batch_end` is slow compared to the batch time (batch time: 0.0094s vs `on_predict_batch_end` time: 0.0895s). Check your callbacks.

30/30 [=====] - 4s 139ms/step



Спасибо за внимание!