**SpaceNet 7 - Solution Description**

**Overview**

Congrats on winning this marathon match. As part of your final submission and in order to receive payment for this marathon match, please complete the following document.

1. **Introduction**

Tell us a bit about yourself, and why you have decided to participate in the contest.

* Name: Victor Durnov
* Handle: cannab
* Placement you achieved in the MM:
* About you: I’m independent Software Developer/Data Scientist interested in hard algorithmic challenges and machine learning (<https://www.linkedin.com/in/victordurnov>)
* Why you participated in the MM: I liked previous SpaceNet challenges a lot

1. **Solution Development**

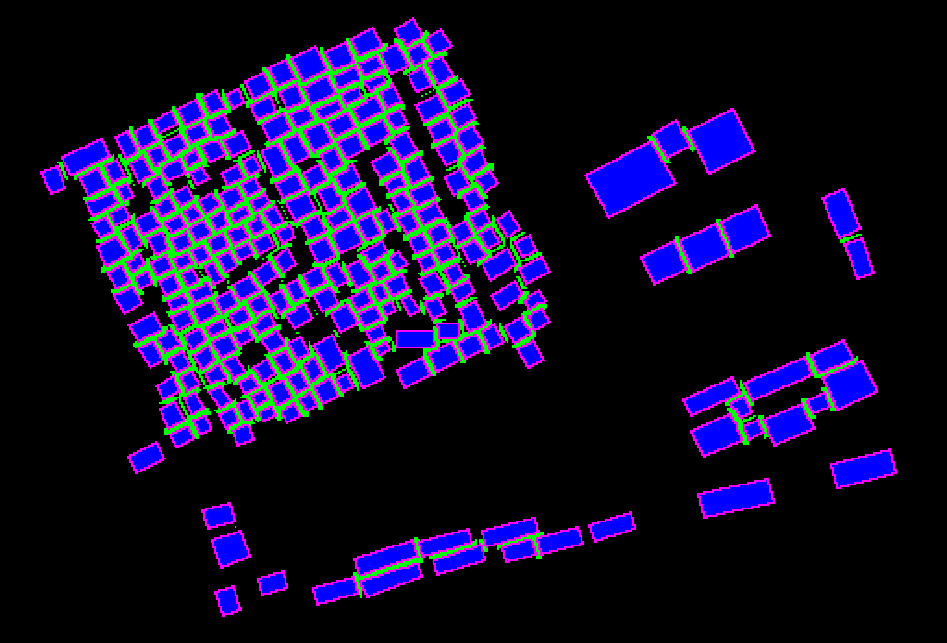
How did you solve the problem? What approaches did you try and what choices did you make, and why? Also, what alternative approaches did you consider?

* My approach is heavily based on my solutions from SpaceNet4 (<https://github.com/SpaceNetChallenge/SpaceNet_Off_Nadir_Solutions/tree/master/cannab>) and xView2 (<https://github.com/vdurnov/xview2_1st_place_solution>) challenges.
* The main part is the Neural Network (NN) for segmentation with encoder-decoder architecture similar to UNet ( https://arxiv.org/pdf/1505.04597.pdf ). Used pretrained efficientnet-b6 and efficientnet-b7 encoders (Transfer Learning).
* Models trained on four times increased images (4096\*4096). Lower resolution worked worse. Higher – harder to work with. For 4096\*4096 input inference goes with 3 overlapping parts to fit in memory.
* Trained segmentation model converted to Siamese Neural Net to be able to do predictions for neighbor pairs of input images. Since weights taken from segmentation models, training goes very fast.

1. **Final Approach**

Please provide a bulleted description of your final approach. What ideas/decisions/features have been found to be the most important for your solution performance:

* Because of low pixel resolution (4 meter) and small objects input images resized to 4096\*4096. Better masks can be created with better separation of small buildings.
* Main targets for segmentation (masks): building bodies, building borders (for better shapes learning and separation), separation area (pixels between neighbor buildings):



* Heavy encoders used worked better with larger crops during training (864\*864 for efficientnet-b7 and 928\*928 for efficientnet-b6)
* Loss function used for training NN is combination of Dice and Focal losses: (Dice + 2 \* Focal)
* When segmentation models trained, they converted to Siamese Neural Net architecture – features of last Convolution layer in Decoder concatenated for two input images (same place) and predicted 2 masks for corresponding input images + 1 channel mask with only new building to force models detect the difference.
* Siamese models tuned on random pairs of months. Places with new buildings used more often during training.
* 2 checkpoints used for final predictions (Siamese models based on efficientnet-b6 and efficientnet-b7) in following way: 1+2, 2+3, 3+4,.. etc.(image from first month together with second, etc.)
* Predicted Buildings mask separated to instances using predicted borders and separation lines, then watershed algorithms used to extend seeds to original size:

Seeds = Building\_Body \* (1 – 0.5 \* Separation\_Line) \* (1 - 0.5 \* Building\_Border)

* Optimization of score - remove FP for F\_change part of metric. Simple thresholds used to remove false “new” building if not enough detections in further months.

1. **Open Source Resources, Frameworks and Libraries**

Please specify the name of the open source resource along with a URL to where it’s housed and it’s license type:

* Anaconda as base Python 3 environment, [www.anaconda.com](http://www.anaconda.com/)
* Pytorch (1.6.0), [https://pytorch.org](https://pytorch.org/)
* EfficientNet PyTorch (0.6.3), https://github.com/lukemelas/EfficientNet-PyTorch
* OpenCV, [https://opencv.org](https://opencv.org/) BSD License
* Shapely, <https://github.com/Toblerity/Shapely>

1. **Potential Algorithm Improvements**

Please specify any potential improvements that can be made to the algorithm:

* Predict images all with all pair, not only neighbors (increase inference time)
* Larger ensemble
* Better resolution of images

1. **Algorithm Limitations**

Please specify any potential limitations with the algorithm:

* Siamese Network works with image pair at a time and cannot see all images at once. (requires more GPU memory and training time).
* Data quality/resolutiuon.

1. **Deployment Guide**

Please provide the exact steps required to build and deploy the code:

Dockerized version prepared as requested.

1. **Final Verification**

Please provide instructions that explain how to train the algorithm and have it execute against sample data:

train.sh and test.sh scripts meet required specification.

1. **Feedback**

Please provide feedback on the following - what worked, and what could have been done better or differently?

* Problem Statement – Very clear, interesting task.
* Data – Ok. Not problems with it.
* Contest – Good as always. Will be waiting for the next one!
* Scoring - Ok.

**NOTE**: Please save a copy of this template in word format. Please do not submit a .pdf