**Marathon Match - 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: Konstantin Maksimov (official Ukrainian transliteration is Kostiantyn Maksymov, I just prefer my own variant of transliteration)
* Handle: MaksimovKA
* Placement you achieved in the MM: TBA
* About you: At the moment I am Computer Vision / Machine Learning Engineer, previously worked as Data Scientist and Computer Vision Engineer for different companies in different domains (check my linkedin for more information <https://www.linkedin.com/in/konstantin-maksimov/>)
* Why you participated in the MM: I like to participate in different DS competitions and especially in geo oriented competitions. Also I got second place on spacenet 6 this year and decided to check my skills at new spacenet challenge.

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?

* First I tried standard Unet-like approach as in other spacenet competitions (for example <https://github.com/SpaceNetChallenge/SpaceNet_SAR_Buildings_Solutions/tree/master/2-MaksimovKA>). This approach gave me at the begging around 16.0 score on the leaderboard using ResNet34 as an encoder in Unet.
* I made a lot of experiments at this stage - the main goal was to find better cross validation split and augmentations and training schedule setup. As a result of this step I found a robust training schedule plus augmentations that worked the best (Random Crop, Random brightness / contrast and Horizontal / Vertical flips). Also as an encoder I used SEnet154 (I tried a lot of them and still do not understand why EffNets does not work for me). This approach gave me around a 20.5 score on the leaderboard.
* For a long time I was unable to improve my leaderboard score. So I decided to improve post processing - the idea was very simple - filter building seed on segmentation mask if count of the seed pixels is less than 4 - it means that if the building appears less than 4 times we just drop it from predictions. This approach gave me around a 26.6 score on leaderboard.
* And just 3 days before the competition ended I decided to try siamese Unet to improve my score. That gave me my final score (see training approach in section below). Also for some reference see great solutions from xview 2 competiton that used same aproach - <https://github.com/selimsef/xview2_solution> <https://github.com/vdurnov/xview2_1st_place_solution> )

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:

* Train siamese unet (two encoders and one shared decoder) using SEnet154 pretrained encoder.
* As input I used 2 random images from one AOI.
* As targets use masks (building instances + separation between buildings) for each image + third mask as a difference of 2 masks.
* On inference make next steps - for each predicted image in one AOI make prediction (with flips as test time augmentations) using this image and all others and after that make simple averaging of predicted probability masks
* On postprocessing use the same algorithm as was used in Spacenet 6 in my solution (<https://github.com/SpaceNetChallenge/SpaceNet_SAR_Buildings_Solutions/tree/master/2-MaksimovKA>)

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:

* segmentation\_models.pytorch,<https://github.com/qubvel/segmentation_models.pytorch>, MIT
* catalyst, <https://github.com/catalyst-team/catalyst>, Apache-2.0
* albumentations, <https://github.com/albumentations-team/albumentations>, MIT
* fire,<https://github.com/google/python-fire>,Apache-2.0
* rasterio,<https://github.com/mapbox/rasterio>, <https://github.com/mapbox/rasterio/blob/master/LICENSE.txt>
* pytorch-toolbelt, <https://github.com/BloodAxe/pytorch-toolbelt>, MIT

1. **Potential Algorithm Improvements**

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

* For tracking I used a solution from baseline code provided at the start of the competition. But this algorithm is very slow and needs some speed improvements
* It is better to use some additional post processing when we lose some building in predictions in the middle (for example if we at the moment t and we don't have building in some location but we have it in t -1 and t + 1 moment we need force to put this prediction in current t moment)
* Try other encoders because SEnet154 is too heavy for production.

1. **Algorithm Limitations**

Please specify any potential limitations with the algorithm:

* Slow speed of tracking
* Very heavy encoder

1. **Deployment Guide**

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

1. The same as described here <https://github.com/topcoder-platform-templates/marathon-data-and-code> and in official competiton rules.
2. **Final Verification**

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

1. The same as described here <https://github.com/topcoder-platform-templates/marathon-data-and-code> and in official competiton rules.
2. **Feedback**

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

* Problem Statement - Statement was very great with all needed details, the one thing could improve user interaction - If you can add bookmarks at the beginning, sometimes it is hard to scroll all the text.
* Data - Provide more data to ensure algorithms robustness.
* Contest - It will be great to see the next spacenet challenge.
* Scoring - I personally liked that now we have Python implementations that helped me a lot.