**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: Selim Seferbekov
* Handle: selim\_sef
* Placement you achieved in the MM: 2nd
* About you: I’ve 10 years’ experience in software development, mostly using Java as the main programming language. Recently I became interested in Deep Learning and computer vision.
* Why you participated in the MM: The topic is quite interesting for me and I wanted to apply the knowledge in image segmentation that I got during Carvana Image Masking Challenge on Kaggle

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?

* I solved the problem as a semantic segmentation task. My model is based on encoder-decoder architectures similar to U-Net  [[Olaf et al, 2015]](https://arxiv.org/abs/1505.04597) and Linknet [[Chaurasia et al]](https://arxiv.org/abs/1707.03718).
* (Signed) Distance Transform did not give any improvements due to poor masks quality
* DSM/DTM channel fusion: In early stages I used only RGB channels and quickly reached the upper limit when models stopped improving. It turned out that normalized DSM helps to solve complex cases when there are structures on the ground, that look similar to buildings.
* As an alternative approach, it would make sense to try Mask-RCNN [[He et al]](https://arxiv.org/abs/1703.06870). As I did not have much experience in tuning region proposal based object detectors at the time, I did not try this approach.
* Ground truth masks cleaning. I spotted some issues with ground truth masks by looking at the out of fold evaluation results. I removed some false positive buildings from masks. Even though it decreased provisional score a bit (as there were similar mistakes for the same cities), I decided to go with cleaned masks.

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:

* I used 4 channels: RGB + normalized DMS (DSM - DTM) which was critical to get high provisional score
* For U-Net architecture I used VGG16 encoder with batch normalization and zero initialization for 4th channel as was done in Deep Image Matting paper [[Xu et al]](https://arxiv.org/abs/1703.03872). This model has the best performance.
* SpatialDropout = 0.5 was used reduce overfitting
* For Linknet architecture I used residual network, similar two Resnet-18 but without an additional pooling layer. It was trained from scratch.
* For training, I used random 256x256 crops with batch size = 16
* As an optimizer RMSProp was used with Cyclic Learning Rate [[Smith]](https://arxiv.org/abs/1506.01186)
* During my experiments, I found that networks converged at the same epochs due to CLR usage and instead of bagging it is possible to train one network and receive several models. The approach is similar to Snapshot Ensembles [[Huang et al]](https://arxiv.org/abs/1704.00109)
* Loss function: pixelwise binary cross entropy + (1 - soft dice)
* Additional loss function to separate buildings: I used dilation and erosion to find pixels between the buildings and put more weights for that regions. Models trained with that loss separated buildings even if there is only 1 pixel margin between them. On the other hand, these models split complex buildings with thin structures.
* Overall ensemble of U-Net VGG and Linknet models trained with different loss functions was used to get the final prediction.
* Postprocessing: watershed transform with two thresholds

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:

* Docker, <https://www.docker.com> (Apache License 2.0)
* Tensorflow, <https://www.tensorflow.org>/ (Apache License 2.0)
* Nvidia-docker, <https://github.com/NVIDIA/nvidia-docker>, ( BSD 3-clause)
* Python 3, <https://www.python.org/>, ( PSFL (Python Software Foundation License))
* Scikti-image, <http://scikit-image.org/>, ( BSD 3-clause)
* Scikit-learn, <http://scikit-learn.org/stable/>, (BSD 3-clause)
* Numpy, <http://www.numpy.org/>, (BSD)
* Scipy, <https://www.scipy.org/>, (BSD)
* Tqdm, <https://github.com/noamraph/tqdm>, ( The MIT License)
* Keras, <https://keras.io/>, ( The MIT License)
* Anaconda, <https://www.continuum.io/Anaconda-Overview>,( New BSD License)
* OpenCV, <https://opencv.org/> (BSD)

1. **Potential Algorithm Improvements**

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

* Signed Distance Transform – if masks were more precise it could give better results
* It is possible to improve custom loss function to put more weight on thin building structures.
* Usage of more powerful encoder in the network, like InceptionResnetV2, but it would require more data
* Instance Segmentation: Mask-RCNN

1. **Algorithm Limitations**

Please specify any potential limitations with the algorithm:

* The model often splits complex buildings. This could be addressed with optimized loss function. Current state-of-the-art instance segmentation architectures like Mask-RCNN with additional DSM channel most likely will solve this issue.

1. **Deployment Guide**

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

1. In this contest, a Dockerized version of the solution was required, which should run out of the box
2. **Final Verification**

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

1. train.sh <train\_data\_path> will train the models. The training data in the provided directory should look like this:  
         JAX\_Tile\_004\_DSM.tif  
         JAX\_Tile\_004\_DTM.tif  
         JAX\_Tile\_004\_GTC.tif  
         ...  
         JAX\_Tile\_005\_DSM.tif
2. test.sh <train\_data\_path> <test\_data\_path> <output\_file> should run prediction code for a given set of test images.
   1. <train\_data\_path> is as defined above for train.sh
   2. <test\_data\_path> points to a folder that contains test data files.
   3. <output\_file>.txt is the name of a file code generates.
3. **Feedback**

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

* Problem Statement – problem formulation and the description was quite good to start implementing a solution
* Data – the amount of data and quality was not good enough to fairly evaluate different solutions
* Contest – the competition was fun, especially when a lot of people reached the limit
* Scoring – the chosen metric perfectly fits the task

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