**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: Motoki Kimura
* Handle: motokimura
* Placement you achieved in the MM:
* About you: A research engineer working on computer vision and machine learning. I’m especially interested in autonomous driving, robotics, geospatial data, and satellite images.
* Why you participated in the MM: The uniqueness of SpaceNet6 dataset made me to participate this competition. I’ve been curious about how practical SAR imagery is particularly for urban analysis e.g., building extraction. SpaceNet6 challenge is the first competition that provides large amount of high resolution SAR imagery and building labels so I decided to participate immediately after the opening announcement.

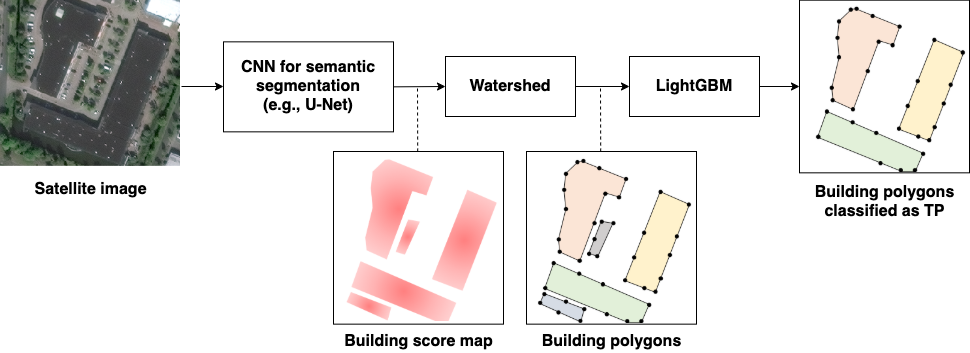
1. **Solution Development**

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 started from the approach most of the winners in previous SpaceNet challenges had taken. The pipeline of the approach consists of 3 parts:

1. Compute building footprint score map using deep convolutional neural networks for semantic segmentation, e.g., U-Net.
2. Extract building footprints as polygons from the score map using traditional image processing algorithms, e.g., watershed.
3. Remove false positives (wrongly detected buildings) with LightGBM models which are trained on the footprint morphological features.

The figure below shows this pipeline:



As for the first part, I tried following to come up with the final solution:

* Network input and output: input to all of my segmentation networks is SAR image with 4 channels. The networks output score maps for 2 classes: building body and building border. The final score map for building footprint, which is input to the second part of the pipeline, is computed as following: *score\_building\_body \* (1 – 0.2 \* score\_building\_border)*. Learning building border helps the network to learn better building shapes and to separate neighboring buildings. This approach was often used by the winners in previous SpaceNet challenges so I just followed it.
* Network architecture: I started from U-Net as it is one of the most popular networks in various competitions for image segmentation tasks including previous SpaceNet challenges, and found it the best for this challenge among the architectures I tried. I tried FPN, PSPNet, PAN, and DeepLab-v3, but none of them outperformed U-Net in the local validation.
* Encoders: as for the encoder of U-Net, I tried ResNet-50/101/152, ResNeXt-50/101, SE-ResNeXt-50/101, SE-ResNet-50/101/152, SENet-154, DenseNet-161, DPN-92/131, and EfficientNet-b5/b6/b7/b8. EfficientNet-b7/b8 outperformed other networks with large margin in the local validation so I selected EfficientNet-b7/b8 as the encoders.
* Encoder pre-training: pre-trained encoder helps the network to converge faster and often improves the accuracy. I found this was also the case for this challenge. I used 3 types of encoders pre-trained in different settings: (A) EfficientNet-b7 trained on ImageNet, (B) EfficientNet-b7 trained with Noisy Student setting, and (C) EfficientNet-b8 trained with AdvProp setting. I found pre-training on SpaceNet6 optical images slightly improved the local validation score in case of the U-Net with encoder-A. In contrast, no such improvement could be found in case of the U-Net with encoder-B/C.
* Attention module: I used scSE module in each block of the U-Net decoder. It slightly improved the score both in the local validation and in the public leaderboard.
* Loss function: I used simple sum of BCE loss (binary cross entropy loss) and dice loss. I found combining these losses helped the network to converge much faster and also improved the accuracy. I also tried focal loss instead of BCE loss, with which the network can put more focus on learning hard examples, but no score gain could be seen in the local validation.
* Optimizer: I started from Adam optimizer and found it the best. I also tried AdamW which was often used by the winners of the previous challenges, but could not find improvement on the local validation score.
* Data augmentation: I tried random flipping, rotation, brightness, salt and pepper noise but they just made the score worse or the convergence slower.
* Input image size: although all the images have the size of 900×900 pixels, I found that the smaller crop size mitigated overfitting of the model. I randomly cropped 256×256 regions and fed them to the network in training.
* Aligning image orientation: all SAR images were rotated before input to the networks, so that the direction from which the data was collected is the same in every case. This moderately improved the score both in the local validation and in the public leaderboard.
* Cross validation: I applied 5-fold cross validation for each U-Net architecture. It was crucial to separate folds by spatial location of the image to avoid leakage because most of the images are spatially overlapped. I split the dataset into 5 folds by longitude extracted from metadata of GeoTIFF.
* Validation metric: IoU score of building body class was evaluated every epoch and was used to choose the best model of each fold. Computing F-score of SpaceNet building detection task takes time to compute so I used the IoU score for the local validation instead.
* Model ensemble: as I used 3 U-Net architectures and applied 5-fold cross validation for each, 15 models could be ensembled to get the better building score map. I simply computed the average of outputs from 15 models. More sophisticated ensemble method may improve the result, but I had no time to try it.

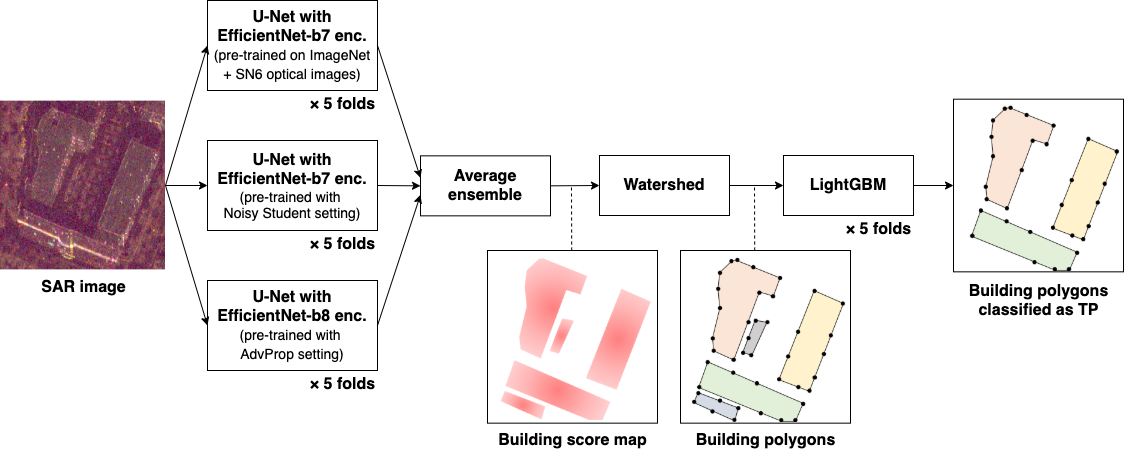
As for the second part, I just applied watershed algorithm implemented in scikit-image to extract building footprints as polygons from the building score map. I also tried an alternative approach: binarize the score map with a threshold and then extract isolated contours as building polygons. However, this approach did not work well in the case the buildings are densely located: neighboring buildings are detected as one building. Watershed helped to separate such buildings. I found that watershed with correctly tuned parameters improved the F-score a lot (by 2-3 points) in the public leaderboard compared to the alternative approach.

As for the third part, LightGBM was trained on the validation set of each fold. The input to the LightGBM models is morphological features of the predicted footprints. In training phase, the models learned IoU score between the input footprint and the best matched ground-truth building. In inference phase, by thresholding the score predicted by LightGBM models, each footprint was classified into true positive and false positive. The footprints classified as false positive were removed from the final prediction in order to improve the precision score. LightGBM models moderately improved the F-score in the public leaderboard. As for the morphological features input to the LightGBM models, I just used the ones found in the winners’ solutions of the previous challenges. I had no time to try, but additional input features might further improve the result.

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:

The figure below summarizes my final approach:



* I trained U-Net whose input is SAR imagery with 4 intensity channels and which outputs score maps for 2 classes: building body and building border. The building score map is computed as following: *score\_building\_body \* (1 – 0.2 \* score\_building\_border)*.
* I used 3 U-Net architectures each of which has different pre-trained encoder-A, B, and C:
  + Encoder-A: EfficientNet-b7 trained on ImageNet
  + Encoder-B: EfficientNet-b7 trained with Noisy Student setting
  + Encoder-C: EfficientNet-b8 trained with AdvProp setting
* I believe the choice of the encoders is one of the keys: use of EfficientNet-b7/b8 improved the score a lot.
* Each U-Net has scSE attention module in decoder part.
* I defined loss function as the sum of BCE and dice loss, and used Adam as the optimizer.
* The U-Net with encoder-A was firstly trained for 140 epochs on pan-sharpened optical images with learning rate of 1e-4. Then it was fine-tuned on SAR images for 260 epochs. In this fine-tuning step, learning rate started from 1e-4 and was decayed by the factor of 0.1 at the epoch of 230.
* The U-Net with encoder-B/C were directly trained on SAR images for 260 epochs. The learning rate started from 1e-4 and was decayed by the factor of 0.1 at the epoch of 230.
* Batch size was set to 16 for the U-Net with encoder-A/B, 14 for the U-Net with encoder-C. The batch size was set to the maximum that could be fit into a Tesla V100 GPU.
* As for input data in training, I randomly cropped 256×256 regions from SAR images. The images were rotated so that the direction from which the data was collected was the same in every image. I did not use data augmentations except for the random cropping.
* Each of 3 U-Net architectures was trained on 5 folds. IoU score of building body class is evaluated every epoch and is used to choose the best model of each fold. 3×5=15 models in total were used for average ensemble to get more accurate building score map.
* Then, watershed algorithm was applied to the score map computed by average ensemble in order to extract building footprints as polygons. Watershed helped to separate neighboring buildings and improved the score a lot. I could not rank in top 5 without using watershed probably.
* Finally, wrongly detected footprints (false positives) were removed with LightGBM models which were trained on the footprint morphological features.
* The input morphological features include polygon area, shape features of smallest external rectangle, major/minor axis length, mean/std values of SAR intensity and predicted building score, neighbor candidate counts in some distance ranges, etc.
* In training, the models learned IoU score between the input footprint and the best matched ground-truth building. In inference, the input footprint was removed from the final prediction if the score predicted by LightGBM is below a threshold.

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:

* NVIDIA Docker, <https://github.com/NVIDIA/nvidia-docker>, Apache License 2.0
* Anaconda, <https://www.anaconda.com/>, BSD 3-Clause License
* PyTorch, <https://pytorch.org/>, BSD 3-Clause License
* segmentation\_models.pytorch, <https://github.com/qubvel/segmentation_models.pytorch>, MIT License
* pytorch-image-models, <https://github.com/rwightman/pytorch-image-models>, Apache License 2.0
* torchvision, <https://github.com/pytorch/vision>, BSD 3-Clause License
* albumentations, <https://github.com/albumentations-team/albumentations>, MIT License
* OpenCV, <https://opencv.org/>, BSD 3-Clause License
* scikit-image, <https://scikit-image.org/>, BSD 3-Clause License
* NumPy, <https://numpy.org/>, BSD 3-Clause License
* pandas, <https://pandas.pydata.org/>, BSD 3-Clause License
* LightGBM, <https://github.com/microsoft/LightGBM>, MIT License
* Solaris, <https://github.com/CosmiQ/solaris>, Apache License 2.0
* GeoMet, <https://github.com/geomet/geomet>, Apache License 2.0
* Rasterio, <https://github.com/mapbox/rasterio>, BSD 3-Clause License
* Shapely, <https://github.com/Toblerity/Shapely>, BSD 3-Clause License
* YACS, <https://github.com/rbgirshick/yacs>, Apache License 2.0
* tqdm, <https://github.com/tqdm/tqdm>, MIT License
* tensorboard, <https://www.tensorflow.org/tensorboard>, Apache License 2.0
* tensorboardX, <https://github.com/lanpa/tensorboardX>, MIT License
* Jupyter, <https://jupyter.org/>, BSD 3-Clause License

1. **Potential Algorithm Improvements**

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

* Pre-training U-Net models on larger datasets in satellite image domain (e.g., datasets used in previous SpaceNet challenges) may improve the results.
* Make use of optical images more effectively, e.g., applying domain adaptation techniques to the model trained on optical images so that it can work on SAR images etc.

1. **Algorithm Limitations**

Please specify any potential limitations with the algorithm:

* The algorithm does not work perfectly where the buildings are densely located. Some neighboring buildings tend to be detected as one larger building.
* The algorithm fails to localize the buildings in some images (as shown in the figure below). I guess these SAR images were taken with a large look angle relative to the nadir and this causes the difficulty for the model to locate the footprints.



1. **Deployment Guide**

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

1. **Final Verification**

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

1. **Feedback**

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

* Problem Statement - The rules were clear and perfectly described on the website.
* Data - The dataset is excellent. I was really excited about working on the first public dataset with large amount of high resolution SAR and optical images. Series of the Medium blog posts about SpaceNet6 and SAR data helped me greatly.
* Contest - It was really nice that TopCoder team gave prompt actions to the questions posted to the forum. I could concentrate on the solution development thanks to them. Provided baseline approach and its implementation was also a big help for me.
* Scoring - The scoring server and the challenge website on TopCoder sometimes had troubles. It would be great if you could improve the stability of those systems in the next competition.

Thank you so much for such an exciting competition! I’m really looking forward to SpaceNet next challenge!

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