# Iter-UNet: Iterative Road Segmentation from Aerial Images

Nicolas Dutly, Janik Lobsiger, Fabijan Dokic, **Group:** PixelSurfers
Department of Computer Science, ETH Zurich, Switzerland

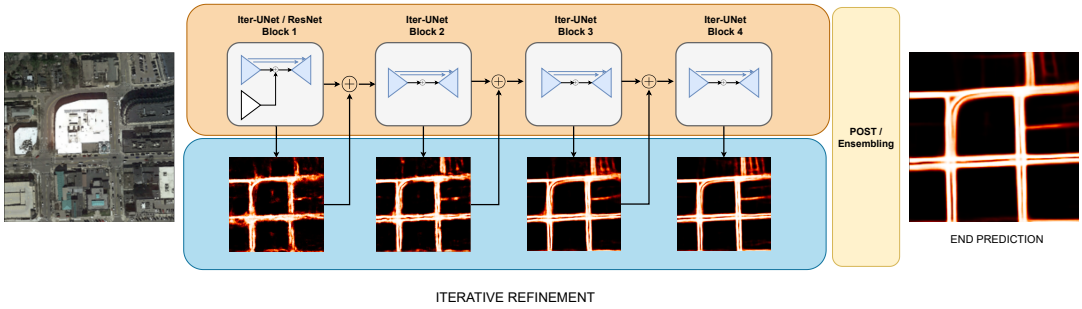


Figure 1. Overview of our Iter-Unet architecture, featuring stacked modules and auxiliary losses.

**Abstract**

**Road segmentation in areal imagery is primordial in areas such as digital mapping, navigation systems and others. We present and compare several variations of a novel segmentation model we developed to tackle this problem. We quantitatively and qualitatively evaluate our novel architecture and compare it to state of the art segmentation models. We combine the best-performing variant with test-time augmentation and ensembling to achieve a public kaggle score of 91.5%. In a broader context, this paper studies how deep segmentation models can be stacked and combined to achieve more powerful, self-refining models.**

## I. Introduction

This paper was written in the context of the Compuational Intelligence Lab (CIL) at ETH Zürich (FS21). It describes our approaches at solving the **project 3**, entitled *road segmentation*. The goal of the road segmentation project is to develop a method which is able to label pixels in an areal image, assigning one of two classes: Roads and everything else (non-roads). The goal of this paper is to give reviewers a comprehensive exposition of our ideas, contributions and thought processes.

### A. Motivation

Satellite and areal imaging are powerful surveying tools and have been critical to the development of various fields ranging from geophysics, mineral exploration, urban planing and others. Road detection in particular, is of importance in fields such as digital mapping, intelligent transportation systems, GPS and many others. In many cases the accuracy of road detection systems directly influences the quality of the above aforementioned systems. The sheer amount of data has pushed the research community to develop automated methods and models for detecting roads in areal images. Though there are a number of non-vision based techniques which enable road detection, such as Google's road API [1] (leveraging GPS user data), these methods are unfortunately not universally applicable (for example in rural areas).

### B. Contributions

We present a novel architecture which is based on stacking the popular UNet model in different ways. We quantitatively and extensively evaluate these different variants of our novel model using a cross-validated approach, supported by imbalance resistant evaluation metrics. We discuss the impact of these variations and compare our result metrics to state of the art segmentation baselines. Besides contributing a (hopefully) high kaggle score, this paper aims to quantitatively evaluate the effect of stacking segmentation models in different ways, and provide possible interpretations for the performance differentials.

## II. Related Work

Early methods tackled road segmentation by using conventional machine learning methods. For example in [2], the authors combined Markovian random fields with random forest classifiers to extract road segmentation masks. In [3], the authors leveraged support vector machines to solve the same task. These traditional statistical machine learning methods were surpassed by neural networks. One family of models have proven particularly effective for pixel-level segmentation tasks: Fully convolutional networks. First introduced by [4], they provide an end-to-end, pixel-to-pixel method of solving semantic segmentation tasks. The

architecture has since been developed by many researchers. For example [5] introduced the UNet model, which provided state of the art results on biomedical image segmentation tasks. Since then, many fully convolutional architectures have emerged, notable ones include the original DeepLab model [6] as well as subsequent improvements. Fully convolutional models have enjoyed great success in the task of road segmentation. For example [7] showed that, when paired with a large amount of data, they can effectively learn from soft-labels, i.e automatically generated labels that may be sub-optimal in terms of accuracy. The success of fully convolutional approaches has been surveyed in [8], where the authors compared fully convolutional models in the context of areal road detection. Domain specific knowledge can also improve predictions. For example [9] noted that a topology-aware loss function which penalizes non curvi-linear segmentation maps improved road segmentation predictions. The idea of using domain-specific structural assumptions predates the usage of neural networks for road detection, as shown by [10], which leveraged higher-order conditional random fields to solve the road detection problem.

## III. Models and Methods

### A. Baselines

For our baselines, we selected three state of the art image segmentation networks, namely the Deeplabv3 [11] architecture, the FCN architecture [12] using the ResNet101 [13] model as backbone and finally the classic UNet [5] architecture. Our choice is motivated by the fact that our novel architecture is an extension of the UNet model and choosing state of the art models in the same family allows for optimal quantitative and qualitative comparisons. Moreover, the first two models are available as part of the *Torchvision*[1] package, enabling effortless reproducibility.

### B. Our Iter-UNet architecture

We named our novel model *Iter-UNet*. In this section we discuss and compare two variations of our model. Both variations are based on stacking the original UNet model multiple times. This idea is motivated by the intuition that early stacks should come up with a rough road segmentation, while consecutive stacks then iteratively refine the prediction of preceding stacks. To force a meaningful representation at every stack, we apply a segmentation loss on every stack's prediction. In summary, our novel model combines stacking approaches which have been traditionally used on CNNs [14], together with the usage of auxiliary losses, whose usage can be traced back to deep convolutional networks such as the Inception model [15].

[1]https://pytorch.org/vision/stable/index.html


Figure 2. Illustration of the stacking variant I (hourglass). The output of one block is directly used as the input of the next block. The predictions and the input of the first block are up-sampled to 64 channels using 1x1 convolution, such that element-wise addition is possible.

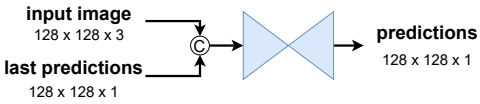


Figure 3. Illustration of the variant II (simple). The symbol c represent concatenation along the channel dimension.

*1) Variant I (hourglass)*: In the first approach, the stacking method is similar to [14]. The initial block receives the RGB image as an input. The input of subsequent blocks can be described as the elementwise addition of features, probabilities and input of the previous block. This procedure is schematically illustrated in Fig. 2.

*2) Variant II (simple)*: The second variant is a simplification of variant I described above. Instead of combining features, probabilities and the previous input, we restrict the input of intermediate stacks to only include the original image and the probability map produced by the previous stack. A schematic representation this variant is illustrated in Fig. 3.

Our proposed variants differ solely in the way outputs and inputs are reconstructed between stacks. The remainder of the architecture is identical: Generally, we distinguish between the first stack (also called the *init* or initial block) and consecutive stacks (*refinement blocks*). Each block's architecture is based on the classical UNet [5] model. The initial block can additionally extract features from the input image using the first three blocks of a ResNet50 [13] model, pre-trained on ImageNet. For the subsequent (refinement) blocks, we use the vanilla UNet, only changing the first convolution layer to match the number of input channels depending on which variation we use, as shown in fig. 2 and fig. 3. The choice to not add the ResNet backbone to every block was mainly made to avoid an explosion in model parameters and subsequently, training time. Both blocks are visualized in fig. 4.

To force a meaningful representation between the different stacks, we decided to compute and combine intermediate losses. As noted by [15], these auxiliary losses can also combat vanishing gradients in deep models. The overall model loss is defined as

$$\mathcal{L}_w(y) = (1-\alpha)\cdot l_n(o_n, y) + \alpha \sum_{i=1}^{n} l_i(o_i, y) \qquad (1)$$

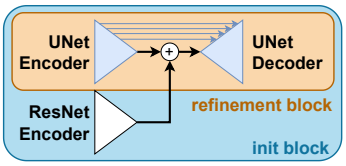


Figure 4. Illustration of the UNet blocks. The blue arrows represent skip connections.

Where $n$ is the number of stacks, $\alpha$ is a weighting factor, $o_i$ is the output of the $i$-th stack and $l_i$ is a loss function applied after the $i$-th stack. After some experimentation with imbalance-resistant losses such as the dice loss, various versions of the Tversky loss and even a topology preserving loss [16], we empirically made the observation that our model performed best when using a classical binary cross entropy error function. This observation is plausible, given the depth of our model and the fact that the gradients of a cross entropy function tend to "behave" better than those of more exotic losses.

### C. Pre-processing and Augmentations

All input data was normalized to have mean zero and variance one. Due to the limited set of data we had, we used augmentations based on the dihedral group $D_4$, i.e all transformations which preserve square symmetries (reflections, rotations), which represent optimal transforms for areal imagery according to [17]. In addition, we added a small color-jitter and random crop transform to increase robustness against differences in lightning, contrast and scale.

### D. Post-Processing

We evaluated several post-processing methods, including conditional random fields (CRFs), morphological operations and simple / adaptive threshholding. Due to the nature of our novel architecture, we did not invest too much time into post-processing, as our model qualitatively performs iterative improvements between blocks (Fig. 1), reducing the importance of overall post-processing.

### E. Test-Time Augmentations, Ensembling and additional data (Kaggle submission only)

All methods in this section were solely used to improve our competitive Kaggle score but are not really interesting from a research perspective, which is why we did not apply any of these techniques when running our comparative experiments. We investigated the benefits of test time augmentations [18], additional training data and ensembling. Test time augmentations make use of invertible functions to reduce variance on a given prediction. In our case, for a given input $x$, we feed $\phi_i(x)$ through the network where we define $\phi_0$ as being a horizontal flip operation, $\phi_1$ as being a vertical flip an $\phi_2$ as a vertical followed by a horizontal flip. We then average the given inverted results to get a final prediction ($\psi$ being the function implemented by our model):

$$\hat{y} = \frac{1}{2}\sum_{i=0}^{2} \phi_i^{-1}(\psi(\phi_i(x))) \tag{2}$$

Furthermore we implemented simple model ensembling by training the model 5 times using different seeds and during prediction averaging the 5 model outputs. Both ensembling and test time augmentations aim to reduce the variance of a given prediction. We also generated an additional 20K of input / groundtruth pairs by cropping out random 400 times 400 patches from the DeepGlobe [19] dataset[2].

### F. Evalutation Methodology

All comparative results found in table I where gathered using the same evaluation protocol: We used two-fold cross-validation on the provided CIL training dataset, applying the same pre-processing and data augmentations across all models. We did not apply any ensembling, test time augmentations, nor did we use the extended dataset, and trained every model for 300 epochs. We decided to use the intersection over union (IoU) metric alongside with the pixel accuracy to get more representative results due to the unbalanced nature of our problem. We deliberately do not report the public Kaggle score for our comparative experiments, as our cross-validated approach is likely to be a better indicators of performance discrepancies given the limited public Kaggle test set we have access to, alongside the fact that the Kaggle score only reflects the patch accuracy, which is biased due to the imbalanced nature of our task.

## IV. Results

The quantitative comparisons of our Iter-UNet model, its variants and our baselines are summarized in table I. We re-trained our best performing model on an extended dataset and applied test time augmentations and ensembling to achieve a public Kaggle score of $91.5\%$. For a qualitative evaluation, we refer to Fig. 5. Using only CIL-data, we achieved a score of $91\%$. Using Deepglobe data, but omitting ensembling $90.3\%$ and further omitting test-time augmentation $90.1\%$.

## V. Discussion

### A. Comparative Baseline Remarks

We note that the vanilla UNet model outperforms our other two baselines by a significant margin. We presume that this is the case due to the set of initial parameters chosen working particularly well with UNet class models. We point out that we did not perform hyper-parameter optimization of any kind, sticking with the default parameters wherever possible, in an effort to reduce accidentally over-fitting the

[2]The resulting dataset can be downloaded here: https://polybox.ethz.ch/index.php/s/KsWqVBk2ppukgSa

Table I
QUANTITATIVE EVALUATION OF THE ITERUNET VARIANTS AND BASELINES

| Architecture | # stacks | Variant | Backbone | IoU | Pixel Accuracy |
|---|---|---|---|---|---|
| IterUNet | 2 | hourglass | no | $0.7264 \pm 0.0027$ | $0.9345 \pm 0.0035$ |
| IterUNet | 2 | hourglass | yes | $0.7337 \pm 0.0052$ | $0.9398 \pm 0.0004$ |
| IterUNet | 2 | simple | no | $0.7312 \pm 0.0031$ | $0.9342 \pm 0.0016$ |
| IterUNet | 2 | simple | yes | $0.7373 \pm 0.0079$ | $0.9414 \pm 0.0011$ |
| IterUNet | 4 | hourglass | no | $0.7393 \pm 0.0048$ | $0.9375 \pm 0.0006$ |
| **IterUNet** | **4** | **hourglass** | **yes** | $\mathbf{0.7408 \pm 0.0024}$ | $\mathbf{0.9418 \pm 0.0001}$ |
| IterUNet | 4 | simple | no | $0.7393 \pm 0.0048$ | $0.9375 \pm 0.0006$ |
| IterUNet | 4 | simple | yes | $0.7387 \pm 0.0031$ | $0.9381 \pm 0.0024$ |
| IterUNet | 4 | hourglass, no aux. losses | yes | $0.7326 \pm 0.0.0057$ | $0.9377 \pm 0.00002$ |
| Unet | n/a | n/a | n/a | $0.7204 \pm 0.0009$ | $0.9355 \pm 0.0006$ |
| Deeplabv3-101 | n/a | n/a | n/a | $0.5899 \pm 0.0221$ | $0.9055 \pm 0.0052$ |
| FCN-ResNet101 | n/a | n/a | n/a | $0.6067 \pm 0.0348$ | $0.9069 \pm 0.0050$ |

training data and thus compromizing the quality of our cross-validated results. In general, we note that every variant of our Iter-UNet architecture outperforms all three baselines.

### *B. Iter-UNet Variant Evaluation: How to Stack*

From the results (table I), we can deduce that model performance is positively correlated with the number of stacks used, regardless of the stacking strategy. We note that stacking models might yield diminishing returns after a certain point, as the time and memory required to train models increases linearly with the number of stacks. For our application, we concluded that four stacks was a reasonable upper-bound, any experimentation with more stacks would have turned out to be very time consuming. As for the two stacking variants, we can see that the hourglass option slightly outperforms the simple stacking method. A possible explanation would be that by using the output of the previous block we actually create a recurrent information flow, which intuitively would be better suited for performing iterative refinement. The simple approach of just feeding the original input image might not carry over the work done by previous models well enough, thus achieving a slightly lower performance. As can also be seen from the table, not applying any auxiliary losses, i.e. only a single loss at the end of the network, leads to worse performance. This highlights the advantage of iterative refinement over simply making the model deeper. We also investigated the effect of weight-sharing (using the same weight for all stacks), but we could not find any conclusive predictive performance advantages, besides the lowered resource requirements.

### *C. Iter-UNet Variant Evaluation: The Backbone Effect*

The pretrained ResNet blocks also contributed to slight increase in model performance. We hypothesize that this can be attributed to the fact that the model does not have to learn any high-level feature extractor from scratch, cutting down the time for convergence and possibly being able to focus on lower level representations.

## VI. CONCLUSION: TO STACK OR NOT TO STACK ?

After having carefully cross-validated our experiments, we conclude that our novel architecture does in fact improve on both our baselines, regardless of which variant we consider. We qualitatively confirm that stacking segmentation models with auxiliary losses can be used to perform implicit post-processing. We have shown that combining features in a recurrent manner (hourglass stacking), alongside with using a pre-trained feature extractor provides the best results. Nonetheless, it is important to note that the resulting model (4 stacks, with backbone) consists of around 130 million trainable parameters, compared to the standard UNet model which has only 30 million trainable parameters. Taking into account the relatively modest impact (though statistically significant) on performance metrics, we believe that ML practitioners should evaluate on a case-by-case basis whether the increased resource usage is a price they are willing to pay. For practical and performance critical uses, we postulate that standard segmentation models might be better suited, even while providing less accurate results.

Figure 5. Selection of test set results using our best performing architecture. Most mistakes happen when classifying parking spaces.

## REFERENCES


[1] Google, *Google Roads API*, (accessed Mai 2021). [Online]. Available: https://developers.google.com/maps/documentation/roads/overview

[2] I. Grinias, C. Panagiotakis, and G. Tziritas, "Mrf-based segmentation and unsupervised classification for building and road detection in peri-urban areas of high-resolution satellite images," *ISPRS journal of photogrammetry and remote sensing*, vol. 122, pp. 145–166, 2016.

[3] M. Song and D. Civco, "Road extraction using svm and image segmentation," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 12, pp. 1365–1371, 2004.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018. [Online]. Available: https://doi.org/10.1109/TPAMI.2017.2699184

[7] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning aerial image segmentation from online maps," *IEEE Trans. Geosci. Remote. Sens.*, vol. 55, no. 11, pp. 6054–6068, 2017. [Online]. Available: https://doi.org/10.1109/TGRS.2017.2719738

[8] R. Wang, F. Pan, Q. An, Q. Diao, and X. Feng, "Aerial unstructured road segmentation based on deep convolution neural network," in *2019 Chinese Control Conference (CCC)*. IEEE, 2019, pp. 8494–8500.

[9] A. Mosinska, P. Márquez-Neila, M. Kozinski, and P. Fua, "Beyond the pixel-wise loss for topology-aware delineation," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 3136–3145. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Mosinska_Beyond_the_Pixel-Wise_CVPR_2018_paper.html

[10] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler, "A higher-order CRF model for road network extraction," in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. IEEE Computer Society, 2013, pp. 1698–1705. [Online]. Available: https://doi.org/10.1109/CVPR.2013.222

[11] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[12] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1605.06211, 2016. [Online]. Available: http://arxiv.org/abs/1605.06211

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[14] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," *CoRR*, vol. abs/1603.06937, 2016. [Online]. Available: http://arxiv.org/abs/1603.06937

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[16] S. Shit, J. C. Paetzold, A. Sekuboyina, A. Zhylka, I. Ezhov, A. Unger, J. P. W. Pluim, G. Tetteh, and B. H. Menze, "cldice - a topology-preserving loss function for tubular structure segmentation," *CoRR*, vol. abs/2003.07311, 2020. [Online]. Available: https://arxiv.org/abs/2003.07311

[17] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, 2020.

[18] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "When and why test-time augmentation works," *arXiv preprint arXiv:2011.11156*, 2020.

[19] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 172–181.