Road Segmentation using U-Net and Y-Net

Sophie Selgrad
Zürich, Switzerland

***Abstract*—We compare and contrast the recently published Y-Net architecture, a model for segmentation using a large and a small arm, with a standard U-Net and demonstrate that unlike claimed by the Y-Net's authors there is no apparent advantage to Y-Net by providing a detailed ablation on the number of arms used. Furthermore, we propose so-called Large Upsampling (LU) which improves Y-Net's performance in image segmentation. Lastly, we provide a first open implementation of the Y-Net[1] architecture, as well as our improvements and proposed changes.**

I. Introduction

The task of image segmentation is foundational to visual machine learning as it has a plethora of applications ranging from medical imaging to GIS related tasks. Hence, a multitude of approaches have been developed. In particular, we work on the task of road segmentation of satellite images with the provided road segmentation dataset, in the following referred to as the provided training dataset.

While some decent models have been developed for segmentation tasks, the continuous research push for even better results means that there is still room for improvement. After reviewing a recent survey [2] the following models have been identified to be of interest: Adaptations of the original U-Net paper [3], a GAN-based approach in "Improving Road Semantic Segmentation Using Generative Adversarial Network" [4], and "A Y-Net deep learning method for road segmentation using high-resolution visible remote sensing images"[1]. A recurring theme is the U-Net-style arrangement of convolutions.

The GAN publication[4] and the Y-Net[1] seemingly fail to explain some of the respective design choices, encouraging further exploration of which model parts lead to the actual benefits over other models. For said reason, we further explore Y-Net's architecture and build our contribution to the task on top of it.

Our main contribution consists of an ablation on the usefulness of Y-Net's arms, as well as introducing Large Upsampling to Y-Net.

II. Models and Methods

A. Related Work

Street segmentation being a highly relevant problem sees a lot of research activity. Firstly, large advances have been made by Volodymyr Mnih by releasing the popular Massachusetts Road Dataset [5]. The U-Net as introduced by Ronneberger et al. [3] has laid the foundation for a lot of modern models, as evident due to its architecture being a recurrent theme in models until today. The Y-Net[1] as well as other designs such as FC-DenseNet [6] are two modern approaches that operate on the same downsample then upsample principle. In contrast to these traditionally trained models, there are also approaches trained in an adversarial fashion such as the above introduced [4] which claims to achieve competitive performance which we were unable to reproduce with our reimplementation. It can be assumed that this stems either from using different data, or is due to imprecise description by the paper's authors. An inquiry to get access to their code in order to confirm our results was left unanswered by the authors. Recent research on GAN approaches such as NVIDIA's semanticGAN [7] seems to achieve state-of-the-art results in image segmentation but come with a sharp increase in model complexity.

In the following subsections, we will take a detail look at the U-Net as well as Y-Net architectures.

1) Baselines:

U-Net: Our first baseline is a standard U-Net implementation [8] which was first introduced by Ronneberger et al.[3]. Simply put, the U-Net is a CNN that consists of a contracting and an expansive path with skip concatenation connections (U-Net style skip connections), that lead to a U shape.

The structure of the two paths are very regular and contains a total of four downsampling as well as upsampling steps, both with additional convolutions in between. In the contracting path the number of feature channels is doubled after every downsampling step while the intermediate representation has its dimensions halved. This leads to a maximum of 1024 feature channels in the middle. Almost analogously, the expansive path features four up convolutions after which the concatenation of the skip connection occurs, leading into the number of channels being halved in the following convolutions.

Since the model has a lot of feature channels for small input dimension in the middle, it detects concise small scale features which can then be combined with the higher resolution information provided by the skip connections after upsampling [3]. As the U-Net is highly successful is widely regarded as an off-the-shelf solution for image segmentation, we have chosen it as a reasonable baseline.

Y-Net: Our second baseline is the Y-Net[1], a CNN that features two arms which are merged by concatenation before being passed through a merge module consisting of

final convolution layers. One arm, referred to as the large arm in the following, is similar to a U-Net with a few distinct changes and the other one, referred to as the small arm in the following, is a simple stack of convolutions with dimensions remaining equal to the input dimensions.

Since no implementation for the Y-Net by Li et al.[1] was found online, a new implementation was completed. In the original publication, some parts are not sufficiently described, so certain assumptions had to be made to complete the implementation.

The differences between the large arm and the U-Net are as follows: 5 max-pooling layers compared to the 4 of the U-Net leading to their choice of the middle convolution having 4096 feature channels, only 2 U-Net style skip connections for the down and upsampling in the middle, and after the middle convolutions it abruptly reduces the number of feature channels from 4096 to 2, which is then used throughout the upsampling process which is similar, but missing the two convolutions after each deconvolution.

As the Y-Net paper is vague in some regards, we have introduced a convolution convolving the 512 channel skip-connection down to 2 channels before concatenating it with the equal size tensor consisting of 2 channels coming from the upsampling. We arrived at the assumption that concatenating two significantly different tensors, where one has 512 feature channels and the other has only 2, could lead to a bias towards the larger tensor, since it could theoretically contain significantly more information, when the number of channels is reduced from 514 to 2 in the next convolution.

In the original Y-Net publication, the authors claim that the large branch is focused on extracting semantic features while ignoring background disturbance, whereas the small branch is responsible for extracting detail features.

In their ablation studies, the authors compare the performance of the complete model with a version consisting of only the large arm with the merging module and another version consisting of only the small arm and the merging module. Results of their ablation studies using only one arm (either small or large) yield similar results, which seems surprising given the significant difference in the number of parameters. Nevertheless, the combination of both arms (large and small in tandem) outperforms the single armed versions essentially across the board. However, we note that when looking at some of their reported accuracies in table three of the Y-Net paper[1], the differences can be very minor and might not be statistically significant.

B. Our Contribution

The following section introduces our proposed changes which yielded the best results. Please refer to Appendix A for additional models and contributions which did not lead to substantial improvements in performance.

1) Number of Arms: One of our goals was to better understand the Y-Net architecture and which parts make it work. Hence, we perform ablation study style experiments such as using only the large arm combined with the final merge module, in comparison to the original 2 arm style Y-Net.

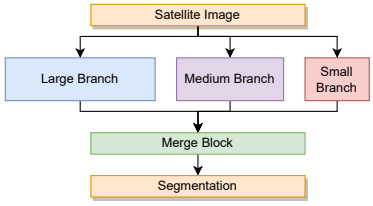

Figure 1. Three-Arm Y-Net.

The experiments immediately suggest adding a third arm. We decided on a medium arm, very much in the style of the large arm, that consists of only 3 downsampling and upsampling steps. It also has 2 skip concatenation connections for the two down and upsampling steps in the middle. Overall, it leads to a maximum of 256 feature channels that are expanded to 1024 and back to 2 analogous to the large branch.

The large branch can provide detailed information on very small patches. On the other hand, the medium branch can provide a mixed use of more details which can be useful after the merging step of all arms. Additionally, the medium branch might also capture sufficient information on medium-sized patches.

For all different arm configurations, branches are merged together, followed by the final convolution. The only differences here are the number of input channels in the merge module, as we have a different number of arms with 2 output channels to merge.

Moreover, we perform the 1 arm vs 2 arm ablation combined with the ”Large Upsampling” idea as explained in the next section.

2) Large Upsampling: The main change to the Y-Net which to a higher performing model, regarding both validation and test scores, is the following: Large Upsample (LU) is a change in terms of channels used during all upsampling (including concatenation) steps.

As stated before, the large arm in the original Y-Net drastically reduces the number of feature channels after the middle convolutions from 4096 to 2. From this point onwards, the number of channels is 2, or 4 if it follows a concatenation step. Furthermore, the skip connections are first passed through a convolution to reduce the number of channels to 2 to not concatenate two tensors with very unequal sizes of dimensions, such as 512 channels and 2 channels.

Hence, in our proposed change, which is illustrated in Figure 2, the upsampling part is altered to only reduce the number of feature channels in the deconvolution layers, namely ConvTranspose2d, by a factor of 2. The downsam-

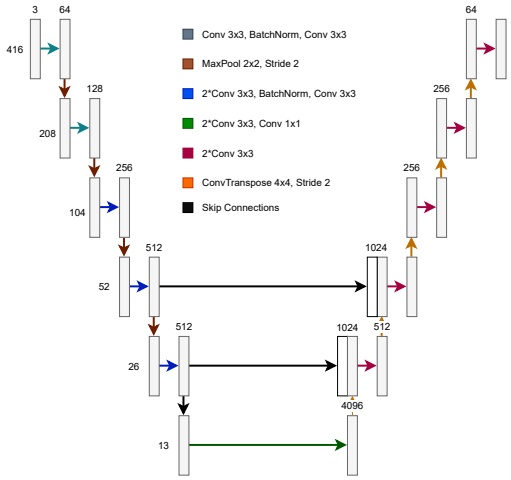

Figure 2. Large arm of Y-Net LU: The number on top of the box denotes the number of channels while the number to the left of the box corresponds to the x-y-size. The white boxes are the copied skip connections.

pling part remains unaltered. As a result, the number of channels is reduced at the same speed it was originally increased, as opposed to being abruptly reduced.

The change also makes the convolution before the skip concatenation obsolete, as it ensures that the channel dimension, for both tensors which are to be concatenated, is equivalent. Lastly, an additional 2 convolution layers are added after each upsampling step, which the original Y-Net does not contain. The Y-Net simply does deconvolution after deconvolution.

Since LU gives the large arm more parameters, we decided to experiment with a slightly changed small arm as well. In this work, small1 refers to the original small arm, and small2 is the altered one. Here, the number of channel features used for the convolutions is expanded more steadily and to a larger max of 256 as opposed to 64 channels.

Lastly, 16 channels each are used to concatenate before the merging part of the two arms in the LU case, compared to only 2 each for the original Y-Net.

C. Training Procedure

1) General Training: Training was performed on ETH's Euler cluster with different NVIDIA GPUs having 24GB of VRAM. We trained using the AdamW optimizer with the below-mentioned hyperparameters. We optimized for binary cross entropy and trained about twice as long as no significant improvement in validation accuracy was observed: For the U-Net model this translates to 1500 epochs and for all Y-Net variants 3000 epochs. Training time was capped at 24h due to computational constraints. For more information, view the results section.

2) Data Augmentation: Data augmentation has proven itself to be crucial in low data image problems using deep learning [9]. During the development phase, we have experimented with a plethora of different augmentations techniques: Using the open source library albumentations [10] we randomly rotate, flip, change brightness and contrast, perform grid distortions as well as normalize the images. Further details can be found in the code.

3) Hyperparameter Selection: We have optimized the hyperparameters of both baselines separately using a logarithmic map on the hyperparameters learning rate (LR) and weight decay (WD). This was done purely using validation. Besides using a batch size of 8 trading off training speed with model size, the following parameters were used:

- U-Net: LR: 1×10^{-4}, WD: 1×10^{-6}
- All Y-Nets: LR: 5×10^{-5}, WD: 1×10^{-4}

D. Performance Evaluation

Our models' performance is measured with the F1 score using a two-step procedure.

1) Validation: Prior to training, the training images were split such that 80% were used for training and the remaining 20% were used for validation. This split is random but with a fixed seed, allowing for reproducibility whilst reducing selection bias within the validation set. At every five epochs, segmentations for the validation set whose pixels range from 0 to 1 are generated, thresholded at 0.5, before being compared to the ground truth. We report top F1 scores for every model.

2) Testing: After hyperparameter tweaking and internal validation, our models are trained without validation. The models are trained on the entire training set. After every 20 epochs, all segmentations for the test images are generated. We did not selectively choose the epoch to submit but opted to select the latest test segmentations. For submission, the provided script was used to perform 16 by 16 patch classification. Again, we report F1 accuracy per model.

III. Results

In Table I we report the scores outlined in Section II-D. All F1 scores are given in percent and rounded to four significant digits. For every entry, the models are trained three times. The mean and standard deviation of all three runs are reported. Note that the LU models with 2 arms were only trained for approximately 2000 epochs compared to the others, as the time for training was capped at 24 hours.

A. Difference between Validation and Testing

There is a large discrepancy between the F1 scores reported in validation versus testing for multiple reasons. First, the test scores are calculated as mentioned above on patches of 16 by 16 pixels, which effectively allows for being more coarse during classification. Second, when internal validation is turned on, the model is trained on only 80% of the data. Given the already small training set, this reduction in data can limit performance. This hypothesis has been confirmed by varying the size of the validation set.

	Models	Validation	Test
Baselines	U-Net	$80.37_{\pm 6.333\times10^{-6}}$	$92.42_{\pm 1.304\times10^{-4}}$
	Y-Net 2 Arms	$79.76_{\pm 5.890\times10^{-4}}$	$92.40_{\pm 4.300\times10^{-7}}$
Y-Net Arms	Y-Net 1 Arm	$79.95_{\pm 1.444\times10^{-3}}$	$92.53_{\pm 2.312\times10^{-4}}$
	Y-Net 3 Arms	$80.03_{\pm 2.312\times10^{-3}}$	$91.99_{\pm 7.491\times10^{-3}}$
Y-Net LU	Y-Net 1 Arm LU	$81.12_{\pm 6.743\times10^{-4}}$	$92.75_{\pm 1.583\times10^{-4}}$
	Y-Net 2 Arms LU small 1	$81.01_{\pm 8.233\times10^{-5}}$	$92.84_{\pm 4.870\times10^{-5}}$
	Y-Net 2 Arms LU small 2	$81.14_{\pm 1.203\times10^{-4}}$	$92.59_{\pm 1.408\times10^{-4}}$

Table I
F1 SCORES (MEAN AND STD OVER THREE RUNS) GIVEN IN PERCENT FOR OUR PROPOSED BASELINES AND MODELS.

Comparison	Validation	Test
Y-Net 1 Arm (LU vs normal)	0.015	0.131
Y-Net 2 Arm (LU small1 vs normal)	0.006	0.007
Y-Net 2 Arm (LU small2 vs normal)	0.004	0.106
Y-Net 1 Arm LU vs U-Net	0.036	0.029
Y-Net 2 Arm LU small1 vs U-Net	0.004	0.009
Y-Net 2 Arm LU small2 vs U-Net	0.005	0.145

Table II
DETERMINED P-VALUES FOR NULL HYPOTHESIS THAT THE MODELS HAVE EQUAL F1 MEAN USING A WELCH'S T-TEST ASSUMING UNEQUAL VARIANCE.

B. Ablation on Number of Arms

Comparing our Y-Net models with different numbers of arms, it can be observed that the F1 scores presented in Table I do not confirm the hypothesis originally laid out by the Y-Net paper for road segmentation [1] that having two arms outperforms an equivalent model with solely one arm. It can also be observed that the three-armed model performs slightly worse during testing. We interpret this as statistical noise and assume that the null hypothesis, namely that the number of arms does not have a statistically measurable effect on performance, holds.

C. The Effect of Large Upsampling

Overall, it can be seen that the validation and test scores for our LU models are higher than their non-LU and baseline counterparts. To quantify the statistical significance of this, we perform a Welch's t-test test on equality of means assuming non-equal variance, the result of which can be seen in Table II. Note that even though for validation we have p values $p < 0.05$, a commonly used threshold for statistical significance, these values are not reached during testing.

D. Comparison to Baseline

From Table I we infer that there is no inherent benefit of using a Y-Net over a U-Net, as both scores overlap in validation and testing. Furthermore, adding or removing arms does not have a meaningful impact on performance compared to the 2-armed Y-Net baseline as scores overlap. Nevertheless, the LU variants of the Y-Net outperform both baselines.

IV. DISCUSSION

Given the aforementioned results, we conclude that the number of arms has a negligible impact on performance. On the other hand, we conclude that there is an indication that large upsampling improves the performance of Y-Nets, but more data is needed to either confirm or refute our hypothesis.

As for the comparison to the baseline for the LU models, performing Welch's t-test we find that there is a statistically significant difference on the validation set but the two-armed LU with small2 fails to statistically meaningfully outperform the U-Net on the test set. There is an indication that all three LU models outperform the U-Net baseline. However, more data is needed to either confirm or refute our hypothesis.

Furthermore, after manual data inspection of validation images and their corresponding segmentations labels which consistently looked very good, we ended up questioning some of the provided labels. We suspect that some models which are able to identify small roads or roads on parking lots end up being punished hard by inconsistent or missing labels.

V. SUMMARY

We have demonstrated that the Y-Net as used in [1] does not provide an inherent advantage over U-Net [3], [8]. Furthermore, we conclude that the claims in [1] that a second arm aids road segmentation cannot be confirmed, as our ablation has shown. Nevertheless, the large upsampling modification we propose for the Y-Net does have a positive effect on performance, although to say that this effect is definitely statistically significant $p < 0.05$, more investigation is needed.

REFERENCES

[1] Y. Li, L. Xu, J. Rao, L. Guo, Z. Yan, and S. Jin, "A y-net deep learning method for road segmentation using high-resolution visible remote sensing images," *Remote sensing letters*, vol. 10, no. 4, pp. 381–390, 2019.

[2] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020, pp. 1–7.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[4] A. Abdollahi, B. Pradhan, G. Sharma, K. N. A. Maulud, and A. Alamri, "Improving road semantic segmentation using generative adversarial network," *IEEE Access*, vol. 9, pp. 64 381–64 392, 2021.

[5] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[6] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 11–19.

[7] D. Li, J. Yang, K. Kreis, A. Torralba, and S. Fidler, "Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] milesial, "Pytorch-UNet," Jul. 2022, [Online; accessed 31. Jul. 2022]. [Online]. Available: https://github.com/milesial/Pytorch-UNet

[9] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[10] 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, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[11] P. Liu, Q. Wang, G. Yang, L. Li, and H. Zhang, "Survey of road extraction methods in remote sensing images based on deep learning," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 90, no. 2, pp. 135–159, 2022.

[12] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[13] P. Isola and J. Zhu, "Y, zhou t, efros aa. image-to-image translation with conditional adversarial networks," *ArXiv161107004 Cs*, 2016.

[14] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning aerial image segmentation from online maps," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 11, pp. 6054–6068, 2017.

[15] alpemek, "aerial-segmentation," Jul. 2022, [Online; accessed 31. Jul. 2022]. [Online]. Available: https://github.com/alpemek/aerial-segmentation

APPENDIX

1. OTHER APPROACHES

To complete the picture, we would like to briefly comment on other approaches we have tried and our intuition on why some of them were not fruitful.

A. Porting Y-Net to GAN Training

We reimplemented a GAN approach originally presented in [4] as GAN approaches seemed to be highly competitive [11]. Although we fully implemented multiple GAN approaches, including a Y-Net trained as the generator in an adversarial setting. As training was slow and did not provide an inherent advantage over a standard Y-Net trained with a binary cross entropy objective we abandoned this idea. In this process, we have adapted training procedure ideas from [12], [13] to stabilize GAN training.

B. Using Different Losses

Besides using the standard binary cross entropy objective, we have surveyed relevant literature [2] and experimented with Dice loss as it in theory more directly optimizes our desired goal of maximizing F1. Since the results were comparable to binary cross entropy, we kept the binary cross entropy objective.

C. Changing Y-Net Architecture

1) Skip Layers for Blocks of Convolutional Layers: Whenever we have two consecutive convolutions in our original model, we replace them by a newly introduced building block which contains two convolutional layers with kernel size 3 and two convolutional layers with kernel size 1. The behavior is as follows:

```
skip = Conv2D(in, k=1, ic, oc)
x = Conv2D(in, k=3, ic, oc)
x = Conv2D(x, k=3, oc, oc)
cat = concat([x, skip], dim=channels)
out = Conv2D(cat, kernel=1, 2*oc, oc)
```

Note that k stands for kernel, ic stands for input channels and oc stand for output channels.

After every convolutional layer, there is a Leaky ReLu as an activation function. Additionally, we also place a batch norm before the second convolutional layer with kernel size three.

This resulted in similar performance as our best performing model.

D. Additional Training Data

As we only had 144 images to train with, we checked if we could augment our training set. For this we had multiple approaches:

- *More General City Street Satellite Segmentations:* We quickly found additional data [14], [15] which also segmented buildings, hence we first wrote an adapter

script which generates the road masks. Furthermore, the data was of a higher resolution than the provided 144 training images, hence we downsampled the additional cities dataset by a constant factor which we determined by comparing the pixel lengths of cars in both datasets. Although we performed detailed investigations, this additional data did not aid performance. A close look at the image statistics revealed that the provided 144 CIL dataset had averaged histograms (i.e. the images were preprocessed in some way) whereas the additional cities dataset was not preprocessed. Unfortunately, despite large efforts we have not been able to correct for this shift in distribution, hence our final models are only trained on the provided CIL dataset.

- *More Boston Street Satellite Segmentations:* We identified that the provided data must come from an American city with trams, which quickly allowed to conclude the images come from Boston (we were even able to align some images to the map). Although the city of Boston provides allegedly annotated street segmentations as well as satellite images, we were not able to download and use them due to georestrictions.