W-Net: A Stacked U-Net for Image Segmentation.

CIL 2019 - Road Segmentation - It's raining in Como

Valentin Anklin 15-916-281 | Flurin Hidber 14-968-451 | Thomas Langerak 17-911-702 | Amray Schwabe 15-920-648

***Abstract*—Being able to determine the exact location of roads from satellite data is extremely useful in a wide variety of cases, for example in the creation of accurate maps or investigation of the impact of disasters. Pixel-wise road segmentation is therefore a important topic. Our suggestion combines U-Nets with the equally well-known hourglass CNNs. Since we combine two U-Nets, we call our approach W-Net. We hypothesize that the second part of the W-Net will learn to infill roads or remove artifacts. We compare our model against 3 baselines. The first baseline uses straightforward logistic regression, the second baseline is a 2-layer CNN. Both were provided for this class. The third baseline is our implementation of a U-Net. We train and evaluate the models on the dataset provided as well as the Chicago dataset. We find that both U-Net and W-Net heavily outperform the other two baselines. We also find that the W-Net is marginally more accurate than the U-Net.**

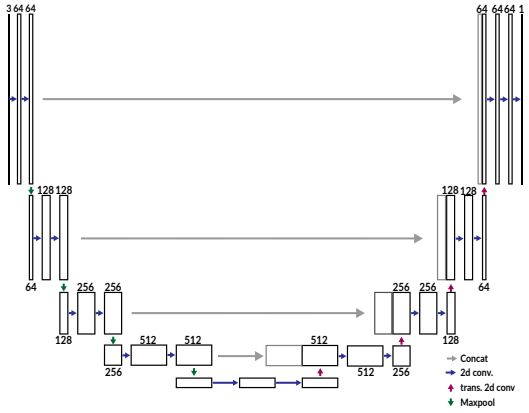


Figure 1: Our implementation of the U-Net architecture.

I. Introduction

In recent years, the availability of high resolution satellite images has drastically increased. With these aerial images, a whole range of real life challenges such as detecting illegal deforestation, distribution of resources after natural disasters or delivering products with drones can be tackled. This can be achieved through automated semantic segmentation of images. The goal of semantic image segmentation is to label every pixel of the input image with the corresponding class. An important thing to note is that here, we do not distinguish between different instances of a class (e.g. between different roads), but just want to decide whether it is a road or not. In the past, the challenge of image segmentation has been tackled by using classic computer vision methods such as edge detection or thresholding [1] [2] [3]. Because satellite images can look very different (e.g. city vs. rural area, different vegetation, different times of the year), it is nearly impossible to reliably segment them by using classical methods. This is why in recent years, different deep learning methods have been applied to image segmentation tasks [4] [5] [6]. In this work, we will focus on the detection of roads on satellite images with machine learning. We will try to achieve state of the art results by adapting and improving existing image segmentation architectures. What makes this task specifically challenging is the extremely limited number of training samples (only 100 images of size 400x400 pixels). Classically, the strength of neural nets is to learn features from immense amounts of data. This lack of training data calls for machine learning architectures that can learn very efficiently with only little data. We adapt the well-known U-Net architecture [7] to an hourglass-like structure. We call this "W-Net". In addition to choosing such a model, we will also augment our input data with the Chicago dataset freely available on the internet [8][1]. We made our implementation open-source available at: https://github.com/tlangerak/CILSubmission. Specifically, the main contributions of our work are.

- We *extend* the U-Net architecture to an hourglass-like structure.
- We *compare* a variety of datasets.
- We *improve* over three baselines.

II. Related Work

A. U-Net

U-Net was introduced in 2015 by Ronneberger, Fischer and Brox as a novel method to train a deep neural network end-to-end with a low amount of training images. Originally, the proposed method was used for biomedical image segmentation. The network is based on a traditional fully convolutional neural network for image segmentation as proposed by Long, Shelhamer Darrell [5]. The authors adjust this network with several important augmentations. Arguably, the most important one being concatenating the output of the contracting part of the architecture with its upsampling counterpart (as illustrated in Figure 1). Since

[1]Based on visual inspection it did not overlap with the test set.

the decoder part has the same dimension as the encoder side of the architecture, the network takes a U-shape, hence the name. Another important one being the increase of the number of feature channels in the decoder part of the network. This allows the network to propagate more contextual information.

B. Stacked Hourglass

Stacked hourglass networks were first introduced in 2016 by Newell [9], specifically for human pose estimation. The philosophy behind this is that "this allows for repeated bottom-up, top-down inference across scales" [9]. The name hourglass is derived from its striking symmetrical shape. The "stacked" originates from multiple hour glasses placed in succession. It is critical that the network has intermediate supervision between the hourglasses.

Since its introduction, hourglasses and their stacked versions have become the defacto choice for many computer vision tasks such as facial landmark localization [10] and action recognition [11].

Only recently has it been used in conjunction with a U-Net-like architecture [12]. This work is also most similar to ours. Indeed, it is even used on satellite image data. Our network makes of use of more channels and we have larger input images. Furthermore, Kahalel et al. use Conditional Random Fields [13] as a post-processing technique which we do not, since that was out of the scope of this project. However, it should be relatively easy to extend the current implementation.

III. Method

A. Architecture

Our architecture is depicted in Figure 2 and is heavily based on the "U-Net". To create our architecture, we implement two U-Nets in sequence, in which the output of the first U-Net is directly fed into the second. Here, we also implement an intermediate loss to ensure that both parts of the network learn. The intuition behind this architecture is that the first part will be relatively good at the task already, however by adding an additional U-Net, we can learn some form of post-processing (e.g. removing isolated labels and straightening road segments).

The input is a 400x400 RGB (3 channels) image. After this, there are two convolutions, which we call a double convolution, with kernel size 3. The activation function is always ReLu. We apply dropout (p=0.1) after every convolution. After this, we perform four "downward" convolutions. Each downward convolution consists of a max pooling operation with stride 2 and a double convolution. After the four downward convolutions, we have four "upward" convolutions. Each upward convolution consists of a bilinear upsampling with factor 2 (to counter the maxpooling in downward convolutions). We concatenate the output of the corresponding downward convolution to this and do a double convolution. After the last upward convolution, we add a single convolutional layer with one out channel. We repeat this sequence for the second U. The exception to this is that the output of the last upconvolution in the first U is concatenated with the second downconvolution in the second U. Next to this, the input to the second half is of size 400x400x1 rather than 400x400x3.

B. Loss

We use a binary cross-entropy loss augmented with a weighted Jaccard loss, as inspired by [14]. The binary cross-entropy loss (BCE) is generally regarded as most suitable for pixel-level image segmentation tasks. The Jaccard loss is defined as:

$$L_{jac} = -\frac{1}{N}\sum \frac{y * y'}{y + y' - y * y' + \epsilon} \tag{1}$$

where N is the number of labels in an image. y and y' are the target and prediction respectively. ϵ is a small scalar to prevent division by 0. The Jaccard loss helps we the unequal distribution of True and False labels (80% and 20% respectively). It is also known as the intersection over union. The binary cross entropy (BCE)[2] is defined as:

$$L_{BCE} = -\frac{1}{N}\sum y \log(y') + (1-y)\log(1-y') \tag{2}$$

This results in the final loss used:

$$\mathcal{L}_u = L_{BCE} + w * L_{jac} \tag{3}$$

in which w is a positive scalar weight.

For the W-Net we added an intermediate loss term. This loss term is identical to L_{BCE} and is applied on the output of the first "U". The intuition behind it is that we want the first U of the network to learn as much as possible about the ground truth and the second U to post-process the prediction. It also helps with learning as there are gradients that directly go to the first U, enabling similar learning than as in the vanilla U-Net. The resulting loss is:

$$\mathcal{L} + w = L^{(1)}_{BCE} + L^{(2)}_{BCE} + w * L_{jac} \tag{4}$$

C. Data Augmentation & Supplementation

Satellite images have the property that they do not have any given "correct" rotation. We observed that the training set contains only a very small number of training images with diagonal streets (12/100), while the test set contains a higher percentage of images with diagonal streets (32/94). Based on this, we rotated all images with steps of $45°$. Then we flipped the images and repeated the same procedure. To reshape these images back to the original size of 400x400 pixels, we cropped the rotated images and then rescaled them by using bicubic interpolation.

[2]We specifically use the BCEwithLogits from Pytorch.

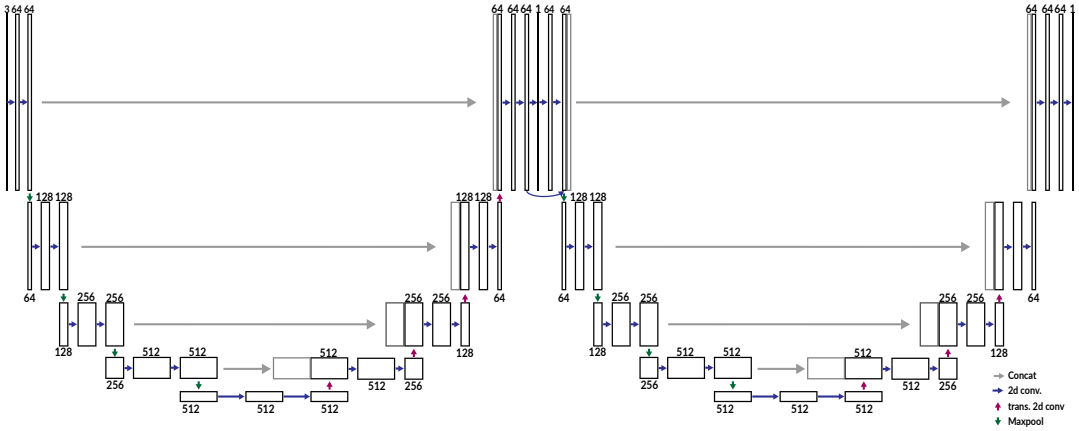


Figure 2: Our implementation of the W-Net architecture.

In extension to enhancing our given data, we used additional satellite data from the "Chicago" dataset[3]. Despite having found multiple datasets online, this one, based on visual inspection, most closely resembled our original data. Because the images were significantly larger, we split them into four disjoint patches and cropped them to 400x400 pixels. We applied the same augmentation described above.

D. Implementation

We implemented our architecture in Python 3.6 using the PyTorch library. We train on a single Nvidia GTX 1080 Ti on the Leonhard cluster. The provided baselines are implemented in Tensorflow. On the U-Net and W-Net we employ the ADAM optimizer [15], that is natively implemented in PyTorch[16], and did hyper-parameter tuning. We found that a learning rate of 0.00025, with a batch size of 4 (inc. batch normalization), a dropout rate of 0.1 and no dilations works best. Furthermore, we set the Jaccard-weight to 0.2. We train for 100 epochs and test on the best model generated during this time period.

We always validate on the validation set based on the supplementary data for two reasons. First of all, the amount of data in the original dataset was limited (100 images in total) which would hardly give an accurate representation of the model learned. Second, the supplementary dataset was on visual inspection most similar to the test data.

IV. Results

A. Baselines

We compare our model with the two baseline models given in the project statement. The first very simple model classifies image patches of 16x16 pixels by extracting 6-dimensional features (mean and variance of RGB colors) and then applying logistic regression to the data. This simple approach achieves a score of 0.59082 on the public part (49%) of the test set. The second baseline consists of a convolutional neural network with 2 convolutional layers with max pooling in between and 2 fully connected layers. The network uses ReLU activation functions and predicts batches of 16x16 size. After training this network[4] for 30 epochs with a batch size of 1 and learning rate of 0.01 (and an exponential decay of 0.95), we got a score of 0.63773 on the public part of the test set.

B. Quantitatively

Model	# of Parameters	Dataset	Accuracy
Logistic Regression		Original	0.59082
2-Layer CNN	580K	Original	0.63773
U-Net	13.4M	Original	0.87972
		Augmented	0.88791
		Sup. + Aug.	0.91459
W-Net (*Ours*)	26.9M	Original	0.88486
		Augmented	0.89807
		Sup. + Aug.	**0.92165**

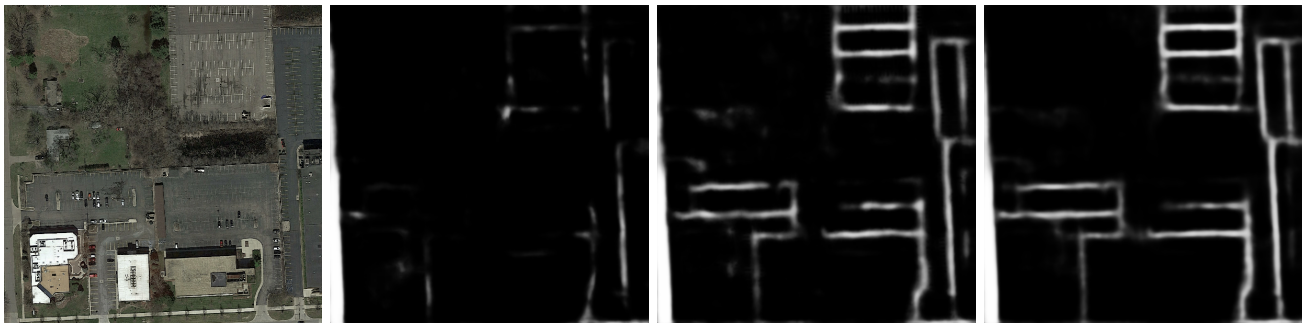
Table I: Results for the different datasets and models. The accuracy reported is from the public dataset on the kaggle submission system.

Table I shows a summary of our results. These numbers are based on the public score on Kaggle. The W-Net with the supplementary and augmented data outperforms the others with an accuracy of 0.92165. The W-Net always outperforms the U-Net on the same dataset. The two other baselines are heavily outperformed by both the U-Net and W-Net. For both the U-Net and W-Net, we see that more data, unsurprisingly, improves the result.

Figure 4 tells a similar story, except less pronounced. Again, we see that just the original dataset gets outperformed by the other options. However, we cannot make a clear

[3]found at https://zenodo.org/record/1154821#.XOAmvS9XZQI (last checked 18-05-2019)

[4]except for the number of epochs we did not finetune these parameters, but used the default ones.



(a) Input (b) Final Output U-Net (c) Intermediate Output W-Net (d) Final output W-Net

Figure 3: A qualitative comparison between the W-Net and the U-Net. The input is taken from the test set, and therefore there is no groundtruth available.

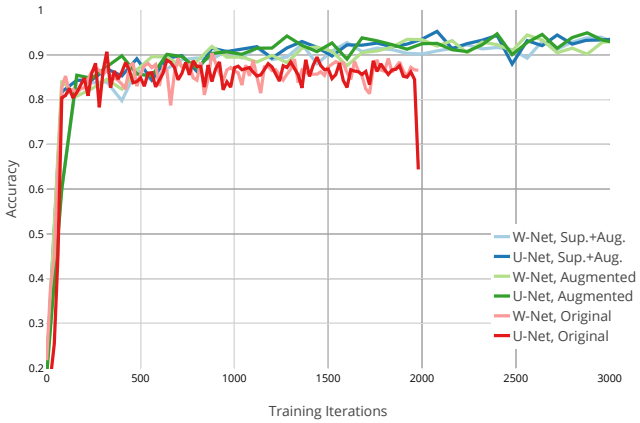


Figure 4: The validation accuracy over iterations for U-Net and W-Net and all datasets.

distinction, in terms of final accuracy and learning rate, between U-Net and W-Net nor between supplementary and just augmented data.

C. *Qualitatively*

In Figure 3 we give a sample of the results. We compare the W-Net against the U-Net for a hard task (a parking lot). Notice how the final output of the W-Net (3d) is more pronounced than the final output of the U-net (3b. Furthermore, it is interesting to compare to the intermediate output of the W-Net (3c). We can see that artefacts are removed (in the middle on the left side) and some infilling is happening (e.g. top road of the block on the bottom left). Finally, we observe that the roads seemed to be straightened after the second U-Net (e.g. the road on the bottom right).

V. Discussion

One of the striking findings is that the W-Net only outperforms the U-Net on the test data. It is likely that this is due to the overfitting of the U-Net on the validation set. This is not unlikely since we used 1-fold cross validation, due to long training times.

Qualitatively, we see that the W-Net has important post-processing capabilities. Although it was not to the extent we expected, we can still see a significant difference between the two parts of the U-Net. Perhaps increasing the kernel size will help in making the second U-Net take a larger context into account and thereby improving results more. In a similar fashion, we have tried different variations of dilations, to no avail.

Other areas of improvement could be to use a more advanced version of the U-Net (Such as a Recurrent residual U-Net), as proposed in [17]. We have done preliminary experiments towards this direction, however the results did not improve significantly over the U-Net and quadrupled the number of parameters. Further improvements can be made by adding additional concatenations between the two Us (e.g. every ”layer). Finally, the low difference between the U-Net and W-net on the validation set but present difference on the test makes us question how much we would have gained by using more folds in our cross-validation. Yet, this difference also shows improved generalization.

VI. Conclusion

In this project we introduced a stacked variation of the U-Net called ”W-Net”. We found that it outperforms the U-Net and a variety of different sized datasets. Overall, it reached more than 92% accuracy on the public scoreboard of the Kaggle competition. The biggest advantage seems to be its capability to generalize better. We also find that increasing the size of the dataset is the largest contributing factor in improved performance.

Despite the mentioned suggestions for future directions, we are content with the work presented in this project. Our third place ranking in the competition is something we think that is fair for this work. Furthermore, we believe that our approach of combining two popular techniques is relatively novel and creative.

REFERENCES

[1] S. S. Al-Amri, N. Kalyankar, and S. Khamitkar, "Image segmentation by using edge detection," *International Journal on computer science and engineering*, vol. 2, no. 3, pp. 804–807, 2010.

[2] A. K. Bhandari, V. K. Singh, A. Kumar, and G. K. Singh, "Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using kapur's entropy," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3538–3560, 2014.

[3] S. Saha and S. Bandyopadhyay, "Application of a new symmetry-based cluster validity index for satellite image segmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 2, pp. 166–170, 2008.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[5] 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.

[6] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[7] 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.

[8] K. Pascal, W. J. Dirk, L. Aurelien, J. Martin, H. Thomas, and S. Konrad, "Learning aerial image segmentation from online maps," Jul 2017. [Online]. Available: https://zenodo.org/record/1154821#.XOAC1i9XZQI

[9] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 483–499.

[10] J. Yang, Q. Liu, and K. Zhang, "Stacked hourglass network for robust facial landmark localisation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 79–87.

[11] M. Fani, H. Neher, D. A. Clausi, A. Wong, and J. Zelek, "Hockey action recognition via integrated stacked hourglass network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 29–37.
object labeling for aerial imagery using stacked u-nets," *arXiv preprint arXiv:1803.04953*, 2018.

[13] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[14] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*. Springer, 2016, pp. 234–244.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[17] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, "Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation," *arXiv preprint arXiv:1802.06955*, 2018.

[12] A. Khalel and M. El-Saban, "Automatic pixelwise