Road Segmentation of aerial images using ResidualAttentionDuckNet

Group: Noccos

Stefan Kramer
Department of Computer Science, ETH Zurich, Switzerland
stkramer@ethz.ch

Yannick Wattenberg
Department of Computer Science, ETH Zurich, Switzerland
ywattenberg@ethz.ch

Simon Wachter
Department of Computer Science, ETH Zurich, Switzerland
siwachte@ethz.ch

***Abstract*—In this paper, we introduce the RA-DUCKNet, a novel architecture for road segmentation in satellite images, which combines the Residual Attention UNet and parts of the DUCK-Net Model. We further studied the performance of multiple existing encoder and decoder models on the task of road segmentation. In order to train the models, we created a dataset consisting of various satellite images of urban areas. Finally, the best encoder-decoder combinations as well as the RA-DUCKNet Model were combined in an ensemble model and expanded with a test time augmentation module achieving a public score of 0.94497 on the ETHZ CIL Road Segmentation 2023 Kaggle Competition.**

I. Introduction

The worldwide road network has a length of around 33 million km and is ever-changing. New roads are built, the course of existing roads is changed or roads vanish completely on a daily basis. As a result, keeping maps up-to-date can be quite expensive when done manually. The development of automatic methods is therefore crucial. A novel approach to solving this challenge is the usage of road segmentation models on satellite images. Despite the high resolution of satellite images, it remains a difficult task as there are multiple objects in satellite images, such as rivers, parking areas, or railways, with high similarity to roads. Further, roads may be obscured by trees or buildings. As of today, deep convolutional neural networks (DCNN) have been quite successful in segmentation tasks. Inspired by two recently published DCNN models, namely Residual Attention UNet (ResAttUNet) [1] and Deep Understanding Convolutional Kernel UNet (DUCK-Net) [2], we developed a new architecture that combines the main building blocks of these two models.

Additionally, we build models using the common approach of combining pre-trained feature extraction models such as a ResNet [3] or EfficientNet [4] as an encoder with a CNN-based decoder like UNet++ [5]. We measure their performance on a road segmentation task and combine the best models in an ensemble together with the RA-DUCKNet for our final model. Finally, we created a new dataset containing some 90'000 satellite images with a ground truth mask from urban areas in the United States. We train all our models primarily on this dataset. Our code is available on Github [6].

II. Models and Methods

A. ResAttUNet with DUCK-blocks

1) ResAttUNet: The ResAttUNet [1] is a further development of the UNet Model by Ronneberger et al. [7]. The UNet Model consists of multiple Downsampling layers used as an encoder and Upsampling layers acting together as a Decoder. Between the last downsampling layer and the first upsampling layer, multiple residual blocks are placed for deep feature extraction. We omit a more detailed explanation of the different layers and refer to the original UNet paper. ResAttUNet extends the UNet architecture by appending a new attention module called Convolutional Attention Module (CBAM) to each downsampling, upsampling, and residual layer. CBAM itself consists of two sub-modules, a channel attention module (CAM) and a spatial attention module (SAM), both generating an attention map. The first one leverages the inter-channel relationship of features by calculating the maximum and mean value for each channel as can be seen in Figure II-A1.1. While the latter focuses on the inter-spatial relationship of the features computing maximum and average across the channels as shown in Figure II-A1.2. In both cases, the extracted features are then further processed with some additional layers and a sigmoid activation function at the end.

The two modules are executed sequentially. The input of CAM is the output of the layer CBAM is appended to. For an upsampling layer, this would be the output of the second convolutional layer. CAM's input and output combined via element-wise multiplication serve as input for SAM. An elementwise computation of SAM's input and output produces the final output of the whole module.

2) DUCK-blocks: DUCK-Net [2] is another example of modifying an existing encoder-decoder structure by adding additional modules. As before with ResAttUNet, the underlying base model is the well-known UNet. The authors' novel convolutional block, DUCK block, should enable more in-depth feature selection to predict the borders of the

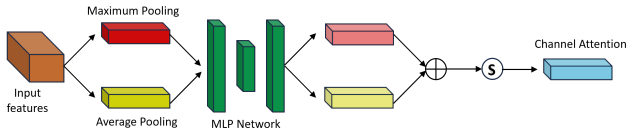

Figure II-A1.1. Architecture of the CAM module Figure 2, Mohammed et al. [1]

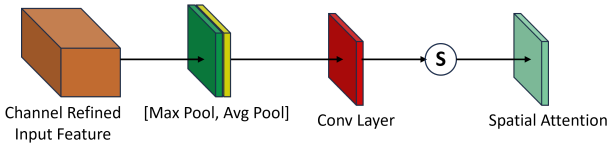

Figure II-A1.2. Architecture of the SAM module. Fig. 2, Mohammed et al. [1]

detected objects better.

A DUCK block can be split into multiple different types of blocks. The Residual block, not to confuse with the residual layer of the UNet, applies multiple small convolutions (3x3 kernel) to extract the small details that make a road. In the DUCK block, residual blocks are repeated multiple times to simulate slightly bigger kernel sizes like 5x5 or 9x9.

The Midscope and the Widescope block are used to gather spatial correlations on a bigger scale. Both apply dilated convolution. The dilation spreads the information of the pixels within a kernel to a larger area, while losing information of smaller details. Since the Widescope block uses one dilated convolution more, the effect is even stronger than in the Midscope one. Hence, Widescope blocks simulate larger kernels focusing on correlations on larger spatial areas of the image than Midscope blocks.

Lastly, a Separable block simulates NxN kernels for large N by combining a Nx1 with a 1xN kernel.

A DUCK block combines the aforementioned blocks in a parallel way using 6 pipelines as demonstrated in Figure II-A2. In the end, a simple addition of the pipeline outputs is applied.

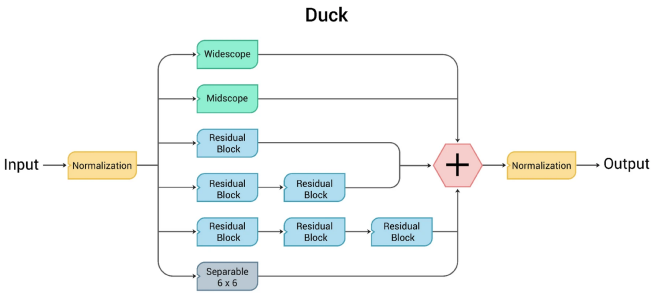

Figure II-A2.1. Architecture DUCK-block. Figure 6, Dumitru et al. [2]

3) Combined Model: We now combine the two ideas explained above into a new model named ResidualAttentionDuckNet (RA-DUCKNet). Our model inserts DUCK blocks into the ResAttUNet architecture. A visual depiction of our model is represented in Figure II-A3. The model consists of 5 downsampling and upsampling layers as well as 3 residual layers between the encoder and the decoder. Specifically at each downsampling layer, we first apply a Downsample block with CBAM. The Downsample block with CBAM is followed by a DUCK block using the output of the CBAM as its input. The outputs of the DUCK-block and the CBAM are finally combined by addition and forwarded to the layer below combined with features from an additional feature extraction network. The upsampling process is quite similar. Each layer consists of an upsample block with CBAM attached to it at the beginning and a DUCK block. Besides the output from the layer below, the CBAM of the upsample block also takes the output of the DUCK-block at the downsampling layer on the same level as input, which is marked by the skip connections in Figure II-A3. Between the residual blocks, there are no DUCK blocks.

B. SMP Models

In addition to our own model, we analyze multiple established models. For all of these models, we utilize the implementations from the well-known ”Segmentation models pytorch” (SMP) library [8]. These models work in the same encoder-decoder way as UNet however we leverage a bigger encoder network like ResNet [3] or EfficientNet [4]. The decoder is then based on segmentation-specific models such as UNet [7], UNet++ [5], or FPN [9]. Specifically, we use the ResNet and EfficientNet encoders both pre-trained on ImageNet [10]. We then train the encoder-decoder combination on the Wachterberg dataset. As decoders, we evaluated a suite of different models and landed on the aforementioned ones.

C. Ensemble Technique

Using our own model and the additional models we then build one ensemble model. This helps reduce variance and noise to give us more robust and better final predictions. Our Ensemble is a simple weighted mean over the raw output of the different models. This approach allows us to weight the predictions of models with especially good validation scores more. We decided to use a mean over a voting ensemble as we don’t lose the information on the model’s confidence in its prediction.

D. Loss function

A quite prevalent loss in image segmentation is the soft Dice-Loss, which is a differentiable reformulation of the Dice Similarity Score. This score is calculated as

$$\frac{2 * TP}{2 * TP + FP + FN}$$

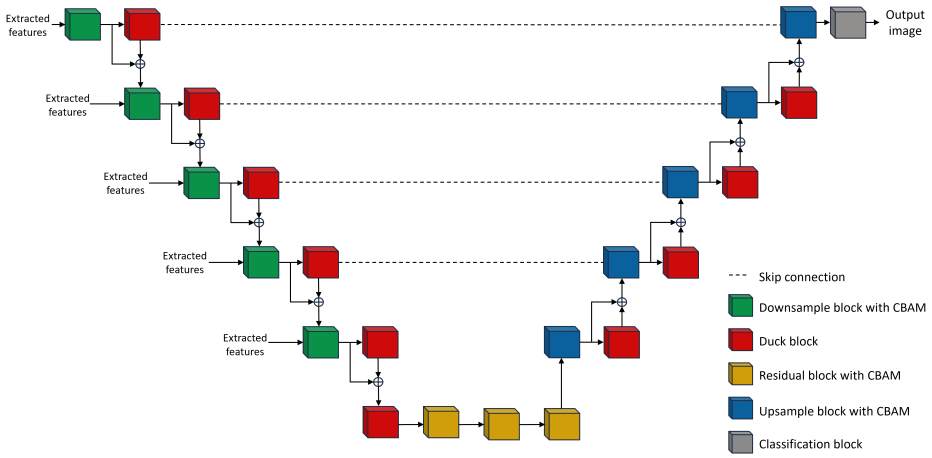

Figure II-A3.1. Architecture of our RA-DUCKNet model which combines ResAttUNet with DUCK-blocks and a feature extraction model.

The Dice-Loss does not directly factor in the topology of the road network. The centerline Dice-Loss (short clDice-Loss) by Shit et al. [11] on the other side encourages the preservation of the topology by depending not only on the two masks but also on their skeletons.
Our loss function linearly combines the Dice-Loss and the clDice-Loss to exploit the properties of both losses.

$$\mathcal{L} = \alpha L_{\text{Dice}} + (1-\alpha) L_{\text{clDice}} \qquad (\alpha \in [0,1]) \qquad (1)$$

A detailed explaination of the two losses can be found in the Appendix A.

E. Additional Dataset

For the training, we mainly use an additional dataset of 90 thousand aerial images of the greater area around US cities, namely Los Angeles, Boston, Houston, Chicago, Phoenix, Philadelphia, and San Francisco which have an especially clear street and highway network. We call this dataset the Wachterberg dataset. It is built by using the google maps static API to pull the images and masks. The dataset consists of 400 by 400 Images. After some simple postprocessing, we get pretty good results which are mostly comparable to the dataset provided through the Kaggle competition [12]. Some example images with masks of our dataset are presented in Appendix C.

III. Results

A. Analysis of SMP Models

In the first stage, we investigated the effectiveness of the models provided by the SMP library. Since testing all possible combinations of encoder and decoder network was not possible in the given amount of time, we fixed the encoder network on the ResNet-Architecture by He et al. [3] to compare the available decoder networks. The best performance was achieved by the DeepLabV3+ [13] decoder and the PAN [14] decoder. Next, we compared the different encoders whilst fixing the decoder on the UNet++ model [5]. The experiment showed that the two EfficientNet [4] variants perform the best with b5 slightly outperforming the b7 variant. All other models performed significantly worse. All results of the encoder and the decoder comparison can be found in Appendix D. Lastly, we ran some different encoder-decoder combinations to see if the previous results hold for different encoders and decoders respectively. The results of

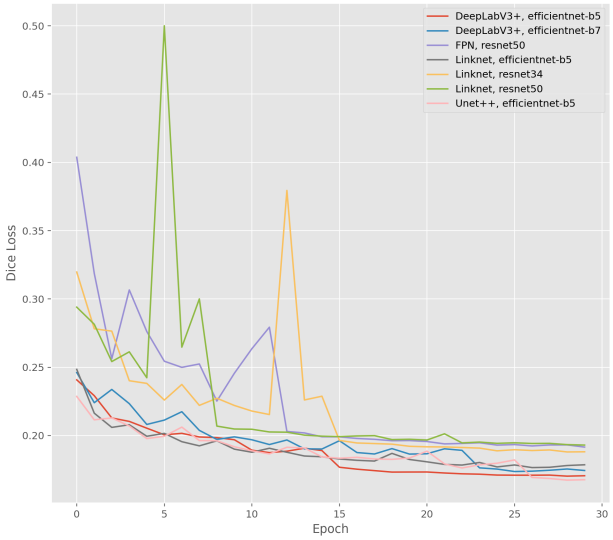

Figure III-A0.1. Comparison of different encoder and decoder combinations.

our runs are visually presented in Figure III-A. As one can see the results hold for the most part with the exception of UNet++ performing significantly better with an EfficientNet encoder instead of a ResNet encoder. In light of these results, we settled on a DeepLabV3+ and UNet++ decoder both with

an EfficientNetB5 encoder.

B. Model comparison

As Baselines models we use the basic UNet by Ronneberger et al. [7] and the RessidualAttentionUNet as proposed by Mohammed et al. [1]. We compare them to our own model RA-DUCKNet as well as the two best-performing models from our study in Section III-A, namely DeepLabV3+ and UNet++ both with an EfficientNetB5 encoder, on Intersection-over-Union (IoU), Accuracy, Recall, F1- and F2-score. For the comparison, we trained all models exclusively on our Wachterberg dataset for up to 30 Epochs. The mentioned scores were then evaluated across the dataset provided through the Kaggle competition [12]. Due to the computational effort of training the models, we did not train models multiple times. Further, we did not run any hyperparameter optimizations. We can see that our

Table I
MODEL METRICS ON THE DATASET PROVIDED THROUGH THE KAGGLE COMPETITION.

Model	IoU	F1	F2	Accuracy	Recall
UNet	0.58	0.73	0.71	0.95	0.68
ResAttUNet	0.60	0.75	0.72	0.95	0.68
RA-DUCKNet	0.68	0.81	0.78	0.97	0.74
DeepLabV3+	0.71	0.83	0.79	0.97	0.75
UNet++	**0.73**	**0.84**	**0.80**	**0.97**	**0.76**

architecture significantly outperformed the UNet and the ResAttUNet baseline. However, our other models which leverage bigger encoder networks still outperformed RA-DUCKNet. Specifically, the EfficientNet encoders seem to have performed the best out of the ones we tested.

For our final submission, we opted for an ensemble made up of all the models in Table I except for the baselines. Further, after training on the Wachterberg Dataset, we finetune the models using the training set of the Kaggle Competition. In the final prediction, we also apply test time augmentation. This technique creates multiple different modifications of the image, which all are sent through the network. In the end, the final mask is computed as an average of the outputs of the modified images. As mentioned above we weighted the predictions of the models by their validation score. This ensemble achieves a public score of 0.94497 on the Kaggle competition [12].

IV. DISCUSSION

We think that our model lacks behind the other comparable models due to multiple reasons. The first reason is that the other models likely utilize the encoder networks better. At the moment we just take the encoder features and concatenate them with the other network features, which we then compress into a more manageable number of channels. This might cause some of the extracted features to get lost. A better way to include the features might be to compress the channels from the previous level of the UNet so that we can use the raw encoder features. Further, it might be beneficial to replace the skip connection around the DUCK blocks and add an additional convolutional layer as in the original paper by Dumitru et al. [2]. This would allow the network to only propagate important details that might get lost because of the DUCK block. Another reason might be that our networks seem to be more unstable in training, especially on the ClDice loss. During training, we observed that the model sometimes breaks in evaluation mode and produces Nan-values. This might be due to suboptimal hyperparameter choices. However, due to resource constraints, we were not able to test training the model with for example a bigger batch size or significantly lower learning rate. What is surprising as well, is the poor performance of the ResAttUnet. This might point to the network being poorly suited for this specific task. It would be interesting to study in future work whether the performance of the observed models is different for other segmentation tasks such as the more common task of polyp segmentation.

We find that the EfficientNet models perform the best out of the ones we tested. We reason that this network strikes the right balance in terms of the size of the network (number of parameters) to the size of our dataset. This also makes sense as it has one of the best Top-1 Accuracy scores on the ImageNet task of the networks that we tested.

One can further notice that the encoder choice affects the performance of the decoder. This means that some decoder networks seem to be significantly better at incorporating features from some feature encoders than others. Specifically, UNet++ [5] jumps from one of the worst models in our decoder comparison with a ResNet encoder to one of the best models with the EfficientNet encoder.

V. SUMMARY

We presented a new architecture for image segmentation by expanding the ResAttUNet-Architecture with DUCK-blocks. On the specific road segmentation task of the Kaggle Competition, our new architecture clearly outperforms the unmodified ResAttUNet-model on multiple segmentation. scores. For better training, an additional dataset was created containing 90'000 satellite images of urban areas. We further studied the performance of multiple combinations of established encoder and decoder architectures. Finally, we created an ensemble model consisting of our own model RA-DUCKNet and the best encoder-decoder combinations including test-time augmentation and finetuning achieving a public competition score of 0.94497.

REFERENCES

[1] A. Mohammed, "Resattunet: Detecting marine debris using an attention activated residual unet," 2022.

[2] R.-G. Dumitru and D. Peteleaza, "Using duck-net for polyp image segmentation," *Scientific Reports*, vol. 13, 06 2023.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[4] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[5] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," 2018.

[6] Y. Wattenberg, S. Wachter, and S. Kramer, "Road segmentation of aerial images using residualattentionducknet," https://github.com/ywattenberg/road_segmentation, 2023.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[8] P. Iakubovskii, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2019.

[9] R. Bodur, B. Bhattarai, and T.-K. Kim, "A unified architecture of semantic segmentation and hierarchical generative adversarial networks for expression manipulation," 2021.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[11] S. Shit, J. C. Paetzold, A. Sekuboyina, I. Ezhov, A. Unger, A. Zhylka, J. P. W. Pluim, U. Bauer, and B. H. Menze, "clDice - a novel topology-preserving loss function for tubular structure segmentation," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021. [Online]. Available: https://doi.org/10.1109%2Fcvpr46437.2021.01629

[12] C. I. L. Team, "Ethz cil road segmentation 2023," https://www.kaggle.com/competitions/ethz-cil-road-segmentation-2023/, 2023.

[13] 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.

[14] H. Li, P. Xiong, J. An, and L. Wang, "Pyramid attention network for semantic segmentation," 2018.

[15] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, "Symbolic discovery of optimization algorithms," 2023. [Online]. Available: https://arxiv.org/abs/2302.06675

[16] P. Tillet, H. Kung, and D. Cox, "Triton: an intermediate language and compiler for tiled neural network computations," *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019.

APPENDIX

A. Loss Function

1) Dice Loss: Since the Dice Similarity Score is not differentiable, it cannot be used as a loss function. However, there exists a reformulation of the score, called soft Dice-Loss, which is indeed differentiable. The soft Dice-Loss replaces TP with $\sum_i y_i * \hat{y}_i$ and expression $2*TP+FP+FN$ with $\sum_i y_i * y_i + \sum_i \hat{y}_i * \hat{y}_i$. Notice, that in a binary setting where both the pixels of the masks and the ground truth are labeled either 0 or 1 the two expressions are actually equal.

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{\sum_i y_i * \hat{y}_i}{\sum_i y_i * y_i + \sum_i \hat{y}_i * \hat{y}_i} = 1 - \frac{\langle y, \hat{y} \rangle}{\langle y, y \rangle + \langle \hat{y}, \hat{y} \rangle}$$

2) Centerline Dice Loss: The clDice loss considers two scores, the Topology Precision (T_{prec}) and Topology Sensitivity (T_{sens}). T_{prec} is defined as the fraction of the ground truth skeleton that lies within the prediction mask. The higher T_{prec} the better the coverage of the road topology. Hence, a high T_{prec} is desirable for a good model. However, a high T_{prec} score does not necessarily translates into a good prediction. A complete white image for example would always cover the whole skeleton and therefore result in a score of 1. Consequently, we need a second measurement to prevent predictions from over-predicting the positive label. The aforementioned problem can be solved by adding Topology Sensitivity to the loss function. T_{sens} is computed identically to T_{prec}, but this time using the skeleton of the prediction and the mask of the ground truth. The T_{sens} score punishes over-prediction. The skeleton of a completely white image is also a completely white image. Most of this skeleton would not be covered by the ground truth mask. In this case, the T_{sens} score is low and mitigates the high T_{prec} score.

In order to maximize both, the precision and the sensitivity, the harmonic mean of the two values is taken.

$$\mathcal{L}_{\text{clDice}} = 1 - 2 * \frac{T_{prec} * T_{sens}}{T_{prec} + T_{sens}}$$

B. Training

We trained all models on our Wachterberg dataset for around 40 Epochs with the clDice loss at an alpha value of 0.5, further, we used the LION optimizer [15], [16] with an lr of 1^{-4} and L2-regularization 1^{-2}, all models where trained with a batch size of 32. For the submission to the competition, we then finetuned the models on the dataset provided through Kaggle for a further 30 Epochs with a reduced lr of 1^{-5}. For the fine-tuning, we switched from the clDice loss to only using the normal Dice loss.

C. Dataset

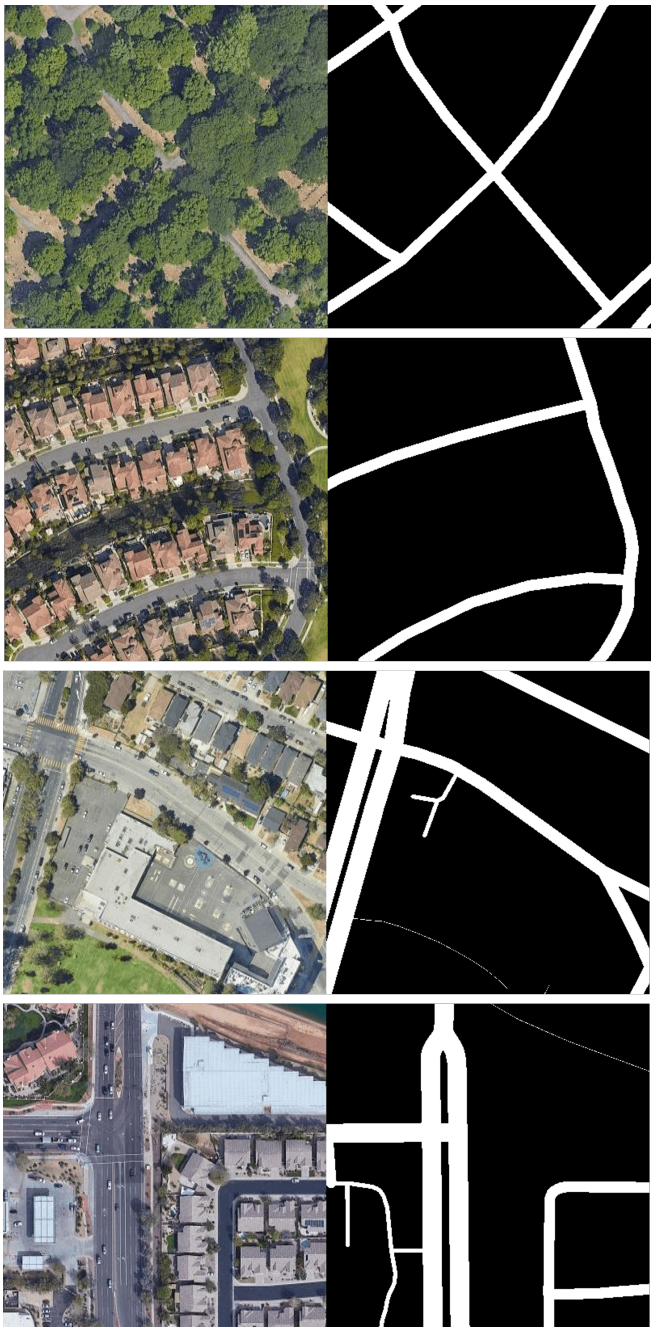

Figure C0.1. Four example images of our dataset taken from Google Maps.

D. SMP Decoder and Encoder comparrision

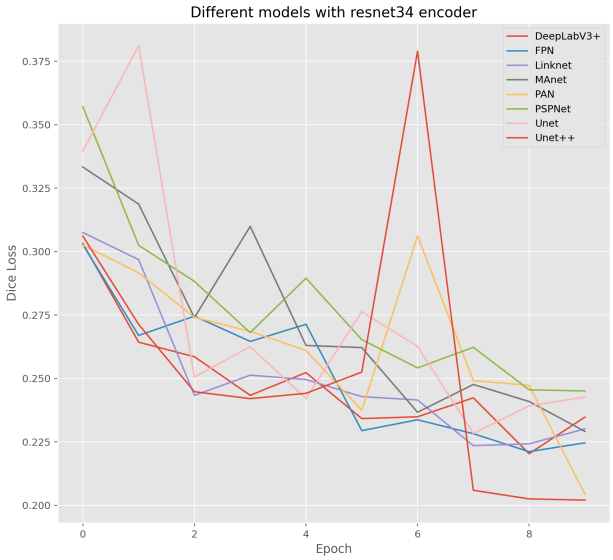

Figure D0.1. Comparison of decoders with ResNet34 as encoder network.

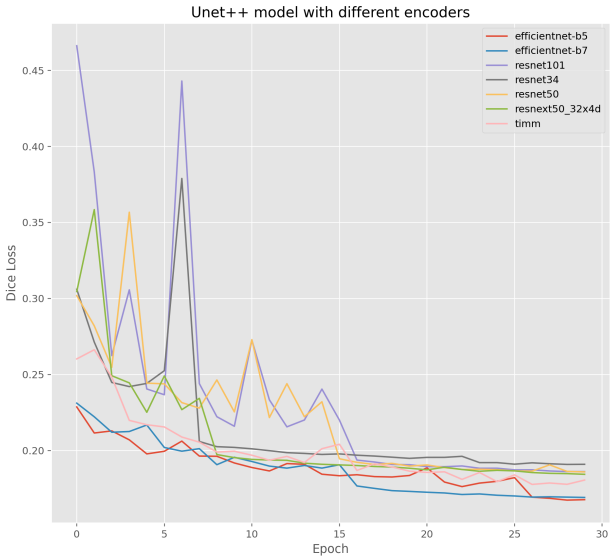

Figure D0.2. Comparison of encoders with UNet++ as decoder network.