Road Segmentation from Aerial Images

Colin Arnet, Julian Neff, Remo Kellenberger
Group: Colin and Friends, Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—This project deals with the segmentation of aerial images into "roads" and "non-roads". Today, deep convonlutional neural networks (CNNs) have evolved to be the standard for image segmentation as they make use of spatial locality and are efficient to train compared to other types of deep neural networks. These models usually require huge amounts of data to achieve meaningful results. Fortunately, there are already many large pre-trained models available, which have been shown to learn important features of images. This makes it less important to have a wide range of data available and speeds up the process of learning. In this project, we adapt state-of-the-art models for image segmentation to our specific task and compare them with each other. We further try to improve the model performance by enhancing our dataset with more samples. Our experiments have shown, that increasing a models complexity does in general not lead to better results for this specific task. Furthermore, using more samples from a similar dataset to train the model has only made a small improvement in performance.**

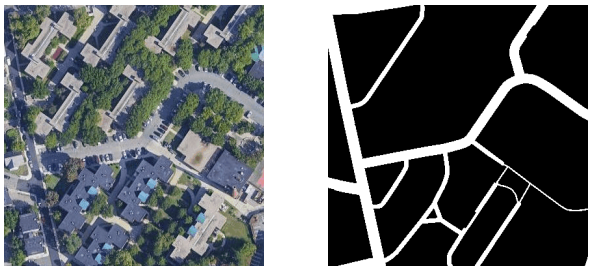
I. Introduction

Semantic segmentation of an image is the process of assigning each pixel to a specific class. In this project, we tackle the task of classifying aerial images into "roads" and "non-roads". This problem is of great importance, since there is a huge amount of data created each day and manually segmenting the images is a cumbersome task.

The most obvious solution to this task is to apply a supervised machine learning algorithm. Deep Convolutional Neural Networks (CNNs) have shown to fit this problem well, as they have already been applied for various different problems [1], [2], [3]. Compared to other deep learning algorithms, CNNs efficiently exploit spatial locality and are able to learn complex features from our (image) input data [4]. A big drawback however is that these algorithms require huge amounts of data. One way to circumvent this problem is to make use of a pre-trained model which are trained on more general tasks and can then be fine-tuned to a more specific one.

The provided dataset consists of only 144 aerial images acquired from Google Maps. As the quality and amount of data hugely impacts our models performance we made use of the Massachusetts Roads Dataset[1], especially because the roads can appear in all forms and noise is usually involved (e.g. cars & trucks on the road). This dataset consists of 1171 aerial images of the state of Massachusetts. Each image has a width and length of 1500 pixels. These images cover urban and suburban regions. We use these images for the pre-train process of our model. From our findings the data from the Massachusetts Roads Dataset is very similar to the data from Google Maps. There is only a mismatch in size and scale, which is easy to overcome, and after modifying the data, it is not possible to distinguish them from each other by looking at them.

[1]https://www.kaggle.com/datasets/balraj98/massachusetts-roads-dataset



(a) Aerial Image (b) Groundtruth mask

Figure 1: Example aerial image with its corresponding road segmentation mask

A. Related Work

Road detection is a task that is well covered by the scientific community. In 1987, Fischler et al. [4] introduced an approach that uses an image filter, graph search, and dynamic programming to extract the location of roads. This technique was introduced before the rise of neural networks. Stilla [5] makes use of maps to get information about objects and roads. Then the gathered information from the map is transferred to the aerial images.

CNNs perform well for semantic segmentation tasks. Initiallyw CNNs were used to classify the content of images, but by adapting the architecture they are also able to classify every pixel which enables the detection of the location of objects. Simonyan et al. [6] introduce a deep CNN architecture with convolutional layers followed by fully connected layers. This VGG-16 architecture was later also used by Kaiser et al. [7] in their paper. Kaiser et al. also use map information to automatically generate new training data. Long et al. [8] go even a step further and introduce the fully convolutional network (FCN) architecture which only contains convolutional layers.

Ronneberger et al. [1] introduce UNet as an FCN that contains a contracting path to capture context and an expanding path that enables precise localization. This architecture

has been used sucessfully for biomedical segmentation. It can also be used for road segmentation as we will use the architecture as one of our baselines.

The UNet architecture can be divided into two parts. An encoder that contracts the input and the decoder part that expands it back to the original size. This approach is being used more and more. For example, the DeepLabV3plus architecture introduced by Chen et al. [9] uses DeepLabV3 [2] as an encoder and adds a simple decoder to the network.

Since CNNs tend to become very large and expensive to train, more work has been done to make them more efficient. EfficientNet [10] always attempts to generate the most efficient network for a given size. These can also be used as a decoding component. Another method are Residual Nets [3] which insert skip connections to accelerate the learning process.

II. Models and Methods

A. Baseline model

We examine our contribution using two commonly used baseline models for image segmentation.

For our first baseline model, we constructed a UNet [1] architecture with ResNet18 [3] used as encoder. The UNet architecture is able to learn effectively even with few training samples. ResNets introduce shortcut connections to avoid the vanishing gradient problem. Both parts of the architecture are built to make learning more efficient. For the implementation of the models, we used Segmentation Models Pytorch [11] and adapted it for our usecase. Segmentation Models Pytorch is a collection of state-of-the-art deep neural networks for image segmentation.

Our second baseline model is DeepLabV3+ with MobilenetV2 used as encoder architecture. DeepLabv3+ is one of the most state-of-the-art model in image segmentation as of today.

Both models were trained with basic image preprocessing. The images were only padded and normalized, and no further augmentation was performed. This allows us to study the contribution of more advanced augmentation techniques such as image flipping and rotation.

B. Massachusetts Roads Dataset

One way to make more accurate predictions is to draw on more data. For this reason, we have generated additional data using the Massachusetts Roads Dataset. The dataset consists of 1171 aerial images of the state of Massachusetts with corresponding road segmentation masks. The images have a size of 1500x1500 pixels. We have processed the images with the following steps:

1) Resize the images to 3200x3200
2) Split each image in 64 tiles of 400X400
3) Filter out tiles with too less information

The first step is done to bring the images of the Massachusetts dataset to approximately the same scale as the images from google maps. Each image is then split into 64 tiles to obtain the dimensions that are needed for training. This results in 74944 tiles. Many image tiles did not contain any meaningful information because either there were no roads in the tile or there was a large black area in the tile. After filtering out those tiles, we ended up having 5332 additional images for training. We used these images to pretrain our models.

C. Image preprocessing

A powerful technique to improve model performance is to use an image augmentation pipeline. Our pipeline consists of three main stages. At the first stage, the images were automatically padded to have width and height to be compatible with the architecture used. This is important because all models perform down- and upsampling operations. For example, if the model uses five downsampling layers, the input size is required to be divisible by $2^{depth} = 2^5 = 32$.

The second stage consists of augmenting the images. This allows us to create new images from existing ones. As our dataset consists of aerial images, we only non-destructive transformations for augmenting the images. The reason behind this decision is, to not add or lose information. In partucular, we performed horizontal flip, vertical flip and rotation by a multiple of 90 degrees.

The third stage of image preprocessing was done while training. Each image was then normalized to fit the encoder used for the model.

D. Image splitting

Another way to get more data is to split the existing images into smaller images. In our case, we have divided each image into four smaller images of dimension 200x200. Thus, we were able to quadruple the number of images available for training. Since the images are now smaller but still contain all the information of the original dataset, we expect the training to be faster with the same precision and accuracy.

E. DeepLabV3+

DeepLabV3plus [9] is a simple decoder architecture developed to work with the DeepLabV3 encoder. The decoder concatenates low-level features with high-level features. A 3x3 convolutional layer refines the features of the concatenated results. As a last step, a simple bilinear upsampling by a factor of 4 brings it back to the original size. Compared to UNET the decoder is not symmetric to the encoder and is also much smaller. The decoder can also be combined with different encoder architectures.

For our experiment, we initialized DeepLabV3+ with three different encoder architectures:

1) Mobilenet V2 (2M parameters)
2) Efficientnet B3 (10M parameters)
3) ResNet101 (42M parameters)

F. Experimental setup

For the execution of our experiments, we tested out different hyperparameters. The following hyperparameters performed best and were finally used for our experiments. We trained our models for 200 epochs. For transfer learning with Massachusetts images, we always fine-tuned all layers of the model. The best scores were achieved by using a learning rate scheduler which decreases the learning rate when no further improvement was made. As a loss function, we used JaccardLoss and DiceLoss, with neither of them having a clear advantage in performance. All models were optimized using the Adam algorithm.

Hyperparameter	Value
Epochs	200
LR	0.001
Loss	JaccardLoss or DiceLoss
Optimizer	Adam
LR-Scheduler	ReduceLrOnPlateau
Batch size	8

Table I: Hyperparameters used for experiments

To evaluate the performance of our models, we tracked the F1 score. The F1 score is a combination of the accuracy and the precision of the predictions. We computed the F1 score for 16x16 pixel patches. A patch becomes 1 (= is road) if at least 25% of the pixels in a patch are 1.

III. Results

A. Using only Google Maps dataset

Model	F1	F1 (Kaggle)
UNet & ResNet 18	0.9481	0.9094
DeepLabV3+ & Mobilenet V2	0.9203	0.9097
DeepLabV3+ & ResNet101	0.9489	0.9129
DeepLabV3+ (timm) & EfficientNet B3	0.9275	0.9114
UNet & MobileNet V2	0.9181	0.9132
UNet (timm) & EfficientNet B3	**0.9628**	**0.9212**
UNet (timm) & EfficientNet B7	0.9155	0.9177

Table II: F1 Scores of models using only original data from Google Maps

Using the original data and applying just default augmentation to it, we were able to train various different configurations. The model we set as our baseline was able to achieve a F1 score of 0.9094 in the Kaggle competition (0.9481 locally). DeepLabV3+ with MobileNet V2 was only able to slightly improve the F1 score in the Kaggle Competition to 0.9097 (0.9203 locally). Using UNet with timm features and EfficientNet B3 achieving an F1 score of 0.9212 (0.9628 locally).

B. Using smaller dataset

Applying the image splitting technique as previously described in general caused worse results. Our baseline UNet and ResNet 18 model decreased its F1 score by 0.1392 compared to the previous local result. Similarly, the previous score from DeepLab V3+ with MobileNet V2 declined by 0.1554. Lastly, UNet with timm features and EfficientNet B3, which achieved the highest F1 score, previously declined by 0.177 to a new F1 score of 0.7858.

Model	F1
UNet & ResNet 18	**0.8089**
DeepLabV3+ & Mobilenet V2	0.7935
UNet (timm) & EfficientNet B3	0.7858

Table III: F1 (local) scores of models with small images (200x200)

C. Transfer Learning with the Massachusetts dataset

Pretraining the models with images from the Massachusetts dataset increases the F1 score from our Baseline UNet & ResNet 18 Model slightly by 0.0046 in Kaggle score (and decreased 0.0127 locally). Using DeepLab V3 + with MobileNet V2, the F1 score increased by 0.0026 (and decreased by 0.0298 locally). Lastly using UNet with timm features and EfficientNet B3 we see a decrease in the Kaggle score by 0.0031 (and decreasing by 0.0132 compared to the local score).

Model	F1	F1 (Kaggle)
UNet & ResNet 18	0.9354	0.9140
DeepLabV3+ & Mobilenet V2	0.8905	0.9123
UNet (timm) & EfficientNet B3	**0.9496**	**0.9181**

Table IV: F1 Scores of models pretrained on Massachusetts dataset and finetuned on google maps images

IV. Discussion

We compared three different models using different techniques. In the first stage of only working with the raw Google Maps data, we tried all combinations of encoders and decoders that made sense to us. We saw that UNet (with timm features) combined with EfficientNet B3 brought us the best result. In a first step of improving the results we tried to split each image to smaller tiles to get more data and increase model performance. However this led to a far lower F1 score as before on all models. We assume that in smaller images the visible features are not big enough to be distinguishable from each other as we lose spatial locality with this approach. Using the Massachusetts dataset, we hoped to achieve better scores because we believe that the given data is not enough and prone to overfitting. Overall we see a slight increase in the first two models (around +0.003) when training the model first on the new data and then fine-tune with the old one. However, for our best model UNet with timm features and EfficientNetB3, we achieved

a slightly lower score in the Kaggle competition. A logical reason for this would be that there is a mismatch in the domain of the pre-train images from the Massachusetts dataset and the downstream images. However looking at the data from a human perspective it is indistinguishable from the original one as we carefully chose the scale of the data and filtered unimportant samples so this seems not to be the reason to us. We assume this is because the model could converge to a local minimum and is then stuck for fine-tuning with the original Google Maps data. We believe that one could circumvent this problem by only training the model for a shorter period of time.

Looking at further improvements, we could apply sharpening filters to the model output, as it is highly unlikely that the roads in the images are not smooth. Similarly we could apply filters to the original data to increase the contrast such that roads would be easier to be detected. Additionally, one could also train the data with a larger dataset by synthetically augmenting the data from Google Maps during the fine-tuning process.

V. Summary

The problem we tackled in this project was to segment aerial images into two categories: ”roads” and ”non-roads”. From our reading we quickly saw that Convolutional Neural Networks fit this task well, as it efficiently makes use of spatial locality and is in general faster than other state-of-the-art neural networks. We made use of different combinations of architectures with pretrained encoders and decoders to segment the data and applied different techniques to it. Some of these models already achieved a good result (as well as our final best result). We believed that the provided data is the bottleneck, as we only have 144 images from Google Maps. In a first attempt, we tried to split the data into 4 patches and then train on this patches. However, because we also split up some features, we decreased the performance of the model significantly. In another approach, we made use of external data to increase the available data to us. This did increase the performance slightly for some models but, in the end, was not able to increase the performance of our best model, which did not make use of any additional data. We assume that this happens because the model is stuck in a local minimum.

Appendix

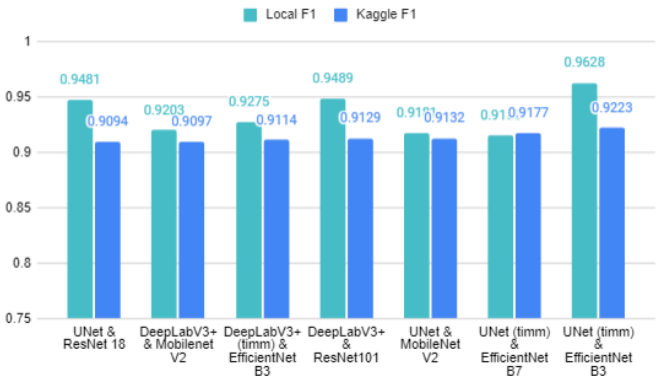


Figure 2: F1 Scores of models using only original data from Google Maps

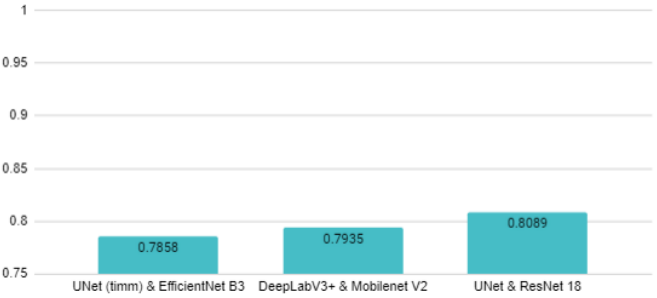


Figure 3: F1 Scores of models with small images (200x200)

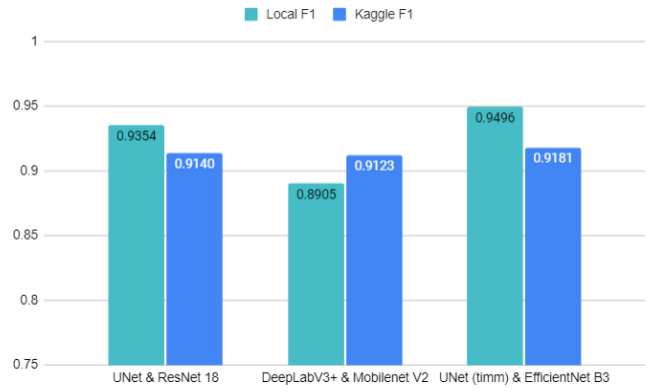


Figure 4: F1 Scores of models pretrained on Massachusetts dataset and finetuned on google maps images

REFERENCES

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

[2] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1706.05587

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

[4] M. FISCHLER, J. TENENBAUM, and H. WOLF, "Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique," pp. 741–752, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080515816500714

[5] U. Stilla, "Map-aided structural analysis of aerial images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 50, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092427169598232O

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[7] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning aerial image segmentation from online maps," *CoRR*, vol. abs/1707.06879, 2017. [Online]. Available: http://arxiv.org/abs/1707.06879

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2014. [Online]. Available: https://arxiv.org/abs/1411.4038

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1802.02611

[10] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1905.11946

[11] P. Yakubovskiy, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2020.