SEGNET ON DRONE IMAGES: IMAGE SEGMENTATION FOR SMART AGRICULTURE

Anders Henriksen, Asger Schultz, Oskar Wiese, Mads Andersen, Søren Winkel Holm

s183904, s183912, s183917, s173934, s183911

ABSTRACT

The abstract should appear at the top of the left-hand column of text, about 0.5 inch (12 mm) below the title area and no more than 3.125 inches (80 mm) in length. Leave a 0.5 inch (12 mm) space between the end of the abstract and the beginning of the main text. The abstract should contain about 100 to 150 words, and should be identical to the abstract text submitted electronically along with the paper cover sheet. All manuscripts must be in English, printed in black ink.

***Index Terms*—** One, two, three, four, five

1. INTRODUCTION

1.1. Dataset and Preprocessing

Our primary dataset consists of two large, high resolution orthomosaic RGB images of a sugar cane field[1]. The first image consists of several drone images stitched together, with the second image of the same size being the by an expert biologist manually labelled human ground truth. The three classes each have a corresponding colour – crop rows are green, weeds are yellow, and soil is red. Void pixels are black.

We also used second, smaller dataset from a corn field. **INSERT LINK** We did not use this for hyperparameter optimization or architechtural decisions like this primary dataset. Instead, only a train/test split was used to evaluate the network using exact same hyperparameters.

1.2. Motivation

Image segmentation and object classification have been a huge talking point in recent years. This is partly due to the wide array of possible applications and the recent interest in machine learning and, in particular, deep learning. One such important application is seperating weeds, crops and dirt in aerial drone images of a field. Applying the SegNet architecture to this field of crop segmentation could allow for smart agriculture that circumvents the many laborious man-hours of manual classification. Using SegNet for this classification could also prove useful in terms of the computational efficiency, as this allows for end-to-end training and makes embedded systems, e.g. in drones, feasible.

1.3. Goal and Application

In the following segments of this paper, the relevant methods, results and discussion will be covered; In section 2, methods such as regularization, data augmentation, metrics and chosen loss function are described. The results are tabulated in section 3. Section 4 is a discussion of the results and accuracy of the reconstruction, a comparison to other similar methods as well as a perspective on the problem. The relevant references are included in section 5.

2. METHODS

2.1. Preprocessing and Data Augmentation

In order to get the most out of the data, preprocessing is needed. First, the RGB values of the non-void pixels of the aerial image are standardized, and a matrix is used to represent the ground truth. Each entry corresponds to a pixel and contains a number 0-2 for the different classes or 3 for void. The images are then padded with black pixels and cropped into smaller 512×512 pixel images. Any of these images containing only black pixels are discarded, leaving a total of 108 pairs of aerial photo/ground truth images. These are then split into 69 training, 18 validation, and 25 test images.

In order to increase the effective size of the dataset, we perform aggressive data augmentation. When training the network, each pair of aerial photo/ground truth images is randomly cropped into 256×256 pixel images. Furthermore, we applied a 50 $\%$ chance of performing a top/down flip as well as a 50 $\%$ chance of left/right flip to each image pair. Even though data augmentation is not as a good as more independent data, it still allows the network to generalize better and overfit less.

The same preprocessing was applied to the secondary dataset with the only difference being that black pixels were a class, so these were not left out of the training and evaluation. We ended up with **INSERT NUMBER OF IMAGES**

[1] Aerial image: http://www.lapix.ufsc.br/wp-content/uploads/2019/05/sugarcane2.png
Ground truth: http://www.lapix.ufsc.br/wp-content/uploads/2019/05/crop6GT.png

2.2. Regularization

Because deep neural networks are such highly flexible models, regularization is necessary on top of the data augmentation to further reduce overfitting. This is done in two ways.

Dropout at $10\ \%$ is used after each blue block in the network (see Fig. **??**). This randomly shuts off nodes during training leading to node redundancy and variability, as the same input will vary somewhat in its output, which learns the network to generalize better. We experimented with higher dropout, but found that too much would significantly reduce learning.

Batch normalization is applied after each dropout. This standardizes the activations, keeping them close to zero. As a result, the weights and biases also stay close to zero, which reduces the flexibility of the model, leading to less overfitting. Batch normalization also has several other benefits, such as reducing the vanishing gradient problem and allowing for a higher learning rate and thus faster convergence time. [1]

2.3.

The encoder part of the network creates a rich feature map representing the image content. The more layers of max-pooling there are the more translation invariance for robust classification can be achieved. The boundary detail is very important when dealing with image segmentation. Hence, capturing boundary information in the feature maps of the encoder before upsampling is important. This can simply be done by storing the whole feature map, but due to memory constrains only the maxpooling indices are saved, which is a good approximation of the feature maps.

2.4. Loss Function: Quality over Quantity

Multi-class cross entropy because:

- Softmax Network: Minus log likelihood
- Can be seen as a classic multiclass classifier – just on a pixel-by-pixel basis.

Weighted cross entropy because:

- Unbalanced class distribution: Network has to learn to focus on important pixels: Don't classify everything as dirt.
- Initial tests made the network behave as the baseline: Simple features in early layers got were not penalized enough and learning was not stable.
- Resampling expensive

2.5. Metrics

Had to use different metrics because

- Not agreement in Image Segmentation papers.
- Want to get accuracy on a global scale and on a class scale.
- Different metrics important in different fields.

The metrics [2] [3]

- Global accuracy: Trivial and not very important because of class imbalance but is good for smoothness
- Mean class-wise accuracy: Takes class imbalance into account. Is what is being optimized for in the model.
- Mean Intersect over Union: "Jaccard Index". Found to be better correlated with human classification though still only ≈ 0.5. Favours region smoothness highly and not boundary accuracy.
- Harmonic mean of precision and recall. To compare to others with same project. Penalizes false positives and gives less credit to true negatives thus being better for unbalanced classes.

2.6. Experiments and Reconstruction

When training the network, we used the following hyperparameters: Batch size of 3, dropout of $10\ \%$, and a learning rate of $1.5 \cdot 10^{-4}$ with the ADAM optimization algorithm. For the convolutional layers, kernel size was set to 3×3 and stride to 1, as deeper network structures tend to learn to better and faster with smaller kernels. This is because the receptive field is the same as larger kernels in shallower networks, but the multiple layers allow for learning more complex structures. In the encoder part, the first convolutional layer increased the number of channels to 64. Thereafter, the first convolutional layer in each block doubled the number of channels, ending at 512. In the max pooling layers, we used a 2×2 with a stride of 2, cutting the size of the feature maps to a quarter each time. The decoder part exactly mirrored this structure.

The training can be recreated using the Jupyter Notebook in the `src` folder in the Github repository. Because the network takes close to 30 GB of VRAM to train, we have implemented a simpler version of SegNet with fewer layers that takes significantly less memory to train. This is controlled using the `use_simple` variable, which is set to `True` by default.

[2] https://hal.inria.fr/hal-01581525/document

[3] http://www.bmva.org/bmvc/2013/Papers/paper0032/paper0032.pdf

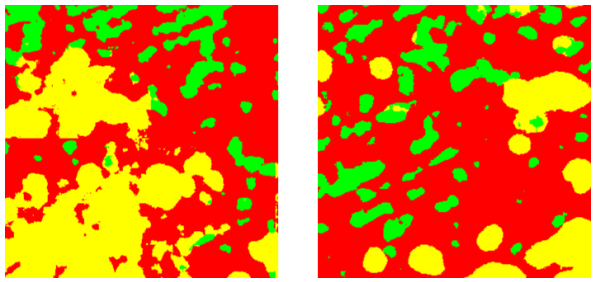

Fig. 1. **Left:** Smaller inferences put next to eachother without use of reconstruction techniques. **Right:** Reconstruction with padding and overlap between smaller inferences.

2.7. Unification of Cropped Image Predictions

In a real-world application of the field classification a farmer would want a complete and precise segmentation of his whole field at once, such that fertilizer and pesticides can be distributed accordingly. To accomplish this, a reconstruction of the smaller image inferences is neccesary. The most straight forward method of combining the smaller images, by simply lining them up next to each other results in a very rough transition between the smaller inferences. This can be seen in the left side of 1. The blocky nature of the field prediction is caused by a lack of information from neighbouring pixel when inference is performed near the borders of an image. To solve this problem, we have chosen to increase the size of the cropped images and add some overlap, and then infer on these enlarged pictured. In the procedure of joining the enlarged cropped pictures they are cropped again, to avoid the border areas. At the cost of computational efficiency more information is available near the borders and a smooth field prediction can be archieved. This can be seen in the right side of 1. A visualization of the reconstruction technique can be seen in appendix 2.

3. RESULTS

4. DISCUSSION

4.1. Comparison of Different Image Segmentation Neural Networks

- Several competing network structures with high performance in image segmentation. U-net, FCN, DeepLabv1, DeconvNet
- Purpose of SegNet, efficient
- 3 out of the 4 mentioned uses the encoder from the famous VGG16 paper, but differ in decoder.
- FCN, No decoder -¿ Blocky segmentation, but very efficient in inference time.
- DeconvNet, Deconvolution and fully connected layers.
- U-Net, (different purpose), skip connections.
- Main takeaway
- (Deeplabv-LargeFOV & FCN)

4.1.1. Draft for Comparison

Before we move on to a comparison of the leading network architectures in the field of image segmentation, it is important to emphasize that the purpose of SegNet is to be efficient in inference time and memory wise, such that it can be used in embedded systems, for instance in drones or in self-driving cars. Besides SegNet, the most acknowledged structures are named U-net, DeepLabv1, DeconvNet and FCN. They have very similar performance and the encoding architecture is almost identical. However, they differ in decoding structure and techniques, and as a common denominator the SegNet alternatives use a lot of memory during inference. Among the heavy users of memory are the fully connected layers of the DeconvNet and FCN, and the skip connections in the U-Net. SegNet uses about half the memory during inference as compared to the other structures and comes in second with regard to inference time. The fastest network is the FCN, and this is becaused it does not have a decoder structure, but makes a prediction directly from the output from the encoder, and this leads to blocky predictions. The speed of SegNet comes in part from the lack of fully connected layers. The encoder part of the network has only 14.7 million parameters compared to partially fully connected structures that tend to have well over 100 million. [2]

4.2. Potential in SegNet

The architecture of SegNet allows for an easy and fast end to end training of the network. As a consequence of this SegNet is easily modified into other use cases without much technical effort. Because of the low memory usage and the speed of which the inference takes place SegNet could prove to be a good choice for self-driving cars and drones. However, it can be challenging and costful to acquire labelled data as this typically involves professionals manually doing the labelling. In our project we only have a single field image, which is probably not representative of similar fields under different circumstances such as lighting, type of crop and season. (The robustness of our network has been tested by running inference on a different data set and with XXX result (EKSILD DATA?) The robustness of

the network can be enhances by different augmentation techniques that is problem dependant. There exits a vast amount of different techniques and some have proved useful in medical imaging, whereas other are preferred in other problem settings.

[2]

4.3. Extension of network

4.4. Conclusion

5. REFERENCES

[1] Jaron Collis, "Glossary of deep learning: Batch normalization," https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82, June 27th 2017, Accessed: 2019-12-21.

[2] Alex et al. Kendall, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," .

5.1. Summary

The purpose of this project has been build an efficient neural network architecture for semantic pixel-wise segmentation on a crop field. The neural network has to be efficient especially in terms of memory usage and time during inference, such that it can be used in embedded system such as drones are cars. The architecture implemented goes under the name SegNet **??**, and it consists of an encoder and a corresponding decoder network and ends with a pixelwise classification layer. The purpose of the encoder is to reduce the dimensionallity and extracts the useful information in the image thereby creating a dense feature maps that can be classified. This procedure is performed by using blocks of convolutional layers, each block ending with a max pooling layer. The decoder has to upsample the low-resolution feature map back into full input resolution, such that a precise pixelwise classification can take place. During the encoding procedure only the most characteristic (Replace with another term) information is extracted with max pooling layers, and local information is lost . This local information is neccesary for a precise classification and therefor SegNet uses max-pooling indices as skip connections to retain this local information. During the upsampling these max pooling indices are used to create a sparse feature map and afterwards convolutional layers are use to create more dense feature maps. The use of max pooling indices are a very efficient way of storing local information with only a slight loss of accuracy. Our results show that ——xx—-. Conclusion —-x—-.

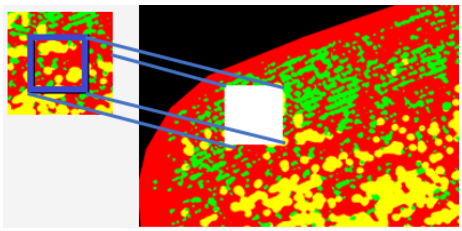

Fig. 2. Reconstruction of the smaller inferences into a unified field prediction.

5.2. Appendix