Road Segmentation using Convolutional Neural Networks

Anna Vera Linnea Fristedt Andersson, Erik Agaton Sjöberg & Theodor Tveit Husefest
Machine Learning, CS-433, EPFL, Switzerland

***Abstract*—The aim of this project is to build a classifier that segments roads on satellite images. In this project, we show how a convolutional neural networks, particularly U-Nets, can solve this task with good results. By making use of U-Nets, data augmentation and finding the right hyperparameters, an F1-score of 0.902 was achieved on the test set.**

I. Introduction

Machine learning is used in many different areas and image segmentation is an area that is becoming increasingly popular. There are many ways of applying an image segmenting algorithm. It can be used in autonomous driving, face recognition or google searches. Since this project strives to build a model that can separate roads from its surroundings, it is chosen to implement a convolutional neural network (CNN) since this has shown to add great value to image segmentation tasks. In this project the CNN type U-Net is used in order to classify the given images.

A. Limitations

The steps executed in the project are in general either influenced or affected by limitations in computing power. The computations are first run locally on laptops and later upgraded to using multiple Graphics Processing Units (GPU's) in *Google Colab* and *Google Cloud*. This enables more opportunities in terms of data and complexity of the chosen model. However, it still limits the possibility of running the most complex models.

II. Data exploration

Given that the task of the project is to segment images and classify each pixel as road or background, 100 unique satellite images are supplied with their corresponding ground truth. The satellite images are of the size 400x400 pixels and each pixel has its own RGB value. The ground truths are gray scaled images of the same size as the original satellite image, where each pixel only has one value - black or white. The task consists of going from RGB and its three channels to a gray image with only one channel. Some examples of the data can be seen in Figure 1.

It is worth noting that while studying Figure 1, roads that are not made for cars are not considered as roads in this project, which can be seen in column one. The second column in Figure 1 also shows that roads in parking lots should not be segmented as roads in this project. In the third column, it is possible to see that some roads are covered with trees from above but the model should still be able to recognize it as a road.

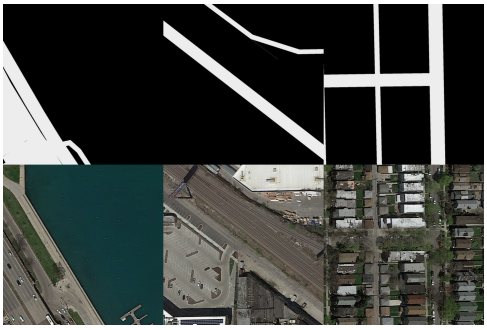

Fig. 1. Satellite and ground truth images from the data set. Row one consists of ground truth images while row two consists of their corresponding satellite image.

The test set that is being evaluated on, is without corresponding ground truth and the images are of the size 608x608 pixels. These images do not represent a more defined 400x400 image, but rather a satellite image consisting of a larger land surface. This means that during the development of the model, the model must be scalable to higher resolutions.

In other words, it is possible to assume that this task requires a complex model since the task consists of segmenting difficult images that even a human can falter on doing. This is why it is chosen to use a neural net as opposed to implementing a less complex model, such as logistic regression.

III. Preprocessing

In order to optimize the result of the image segmentation, preprocessing of the data set, consisting of 100 satellite images, is implemented.

A. Data augmentation

Since a large data set in machine learning models is highly valued, augmentation of the 100 initial images is implemented. In this project, two different ways of enriching a dataset made up of images are used, rotation and mirroring. Mirroring an image refers to flipping it along one of its two axes. Below, mirroring of a matrix among its columns are shown. One can think of the integers in the matrix as one of the pixels in the satellite images.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix}$$

A rotation of 45 degrees is shown in Figure 2. One of the model parameters are the number of angles all images are rotated. Since a rotation of an image results in a larger image with empty corners, the corners are filled in by mirroring the initial image. In order to end up with a 400x400 pixel image after rotation and filling of corners, the middle of the image is cropped as is shown in Figure 2. Rotations of the images also means that the model is fed with roads that travel in all various directions and not only parallel and perpendicular orientations.

Each of these augmentations add an additional 100 images to the training set. This means that for each degree an additional 300 images is added. Each image is first rotated, has its corners filled and cropped followed by mirroring among both axes. Each operation increases the size of the dataset by 100.

Given that more data should result in a better model, the number of angles are always tried to be maximized. The number of angles that are used are dependant of the computational power, since all of the images have to be stored in the memory.

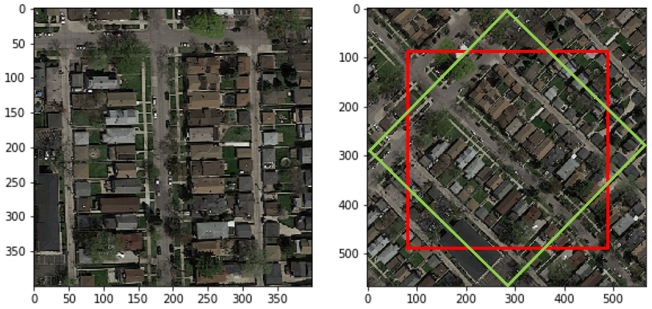

Fig. 2. Left image: original image. Right image: Green square; original image Red square; new image added to the data set. Everything outside the green square is mirrored.

IV. Model

To classify the roads, it is decided to go for a U-Net, originally proposed by Olaf Ronneberger, Philipp Fisher, Thomas Brox in the paper *U-Net: Convolutional Neural Nets for Biomedical Image Segmentation*. We decided to go for this architecture as it previously has shown promising results on relatively small data-sets. One of the main advantages with U-Net is the ability to retain localization of the pixels as well as classifying them, which is needed when segmenting roads.

Figure 3 shows the U-Net as proposed in the original paper. It consists of three parts: contraction, bottleneck and expansion. It it symmetrical, which means the contracting and expanding blocks are inverse of each other in order of number of features and image-size.

- **Contraction** - Consists of two convolutional layers each followed by an activation function. After that a down-sampling is performed by using max pooling with strides 2.
  Each of the contraction steps reduces the image-size by half, and doubles the number of features. Before max pooling we save the contracted image to use in expansion block.
- **Bottleneck** - Consists of two new convolutional layers each followed by an activation function. At this point have maximum number of features-channels.
- **Expansion** - First there is an up-sampling with strides 2, then this image is concatenated with the corresponding convolution. The result of this is again convoluted twice. Each expansion reduces the features by half, and doubles the size of the picture.

We slightly modified the model in Figure 3 to fit our needs. We added padding to the convolutional layers, so that input-size is equal to output-size. There are also no regularization layers in this model, which will be discussed later.

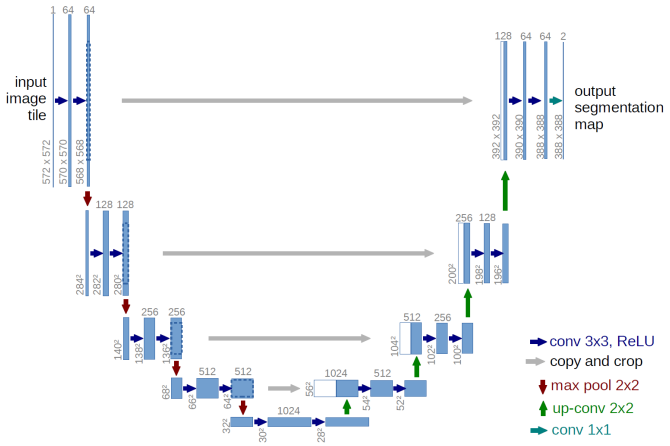

Fig. 3. U-Net as proposed in the original paper.

V. Choosing Method

Training the model turned out be more difficult than expected. Hardware-problems limited our training on (400x400) images, and also the test-images were (608x608), so we needed a way around this problem.

A. First Attempt - (40x40) Patches

Our first attempt was to feed the U-Net with patches of (40x40), and then reassemble the pieces after training. This solved our computational issues, but did not give great results in training. The patches retained too little contextual information which made the patterns in roads hard to distinguish.

B. Second Attempt - (128x128) Resolution

To keep the contextual information in our images we tried to lower the resolution of our images to (128x128). This way the images keeps the patterns and shapes of the roads (e.g. long straight lines) which the neural net is good at recognizing. However, this comes at a cost of losing information in the images. In our training set where the images are (400x400), the information loss is

$1 - \frac{(128\times128)}{(400\times400)} = 1 - 0.1024 \approx 90\%$. In the test-images (608x608) this gives $1 - \frac{(128\times128)}{(608\times608)} = 1 - 0.0044 \approx 96\%$ information loss.

This method provided very good results in training, which a F1-score of around 0.95 on our validation set. However, when predicting on our test-set and uploading to AICrowd we got an F1-score of 0.777. Since there was a wide gap between our validation-score and submitted score we concluded that our method included too much information loss.

C. Final Attempt - Patches with Padding

Our final solution to the problem was a combination of the two first attempts. Since the U-Net worked well on (128x128) images, we decided to divide our images into patches of (100x100) and (200x200) patches and add padding along the edges of the images. This kept the high resolution in the images as well as it keeps contextual information. It is also scaleable, and should work well on any image-size. This method provided coherent results on validation-set and AICrowd, so this is the method we optimized going forward.

Fig. 4. One patch and its padding. Given image size of 400x400, the gray square has size 256x256 and the black square has size 200x200. Padding in this example is set to 28 pixels.

Figure 4 shows the relationship between one image, its patch and padding. In reality the model is trained with more than one patch which means that the padding of each patch will have some overlap. As for the padding in the corner that is outside of the actual image we used a mirroring/reflection of the image in order to extend each side with the padding size. In the end, when the predicted patches with their padding were put back together to a full image, the padding was removed which made it possible to put the patches back together without any overlapping.

VI. Model Selection

When optimizing the model we had a lot of hyper-parameters we could tune. As the network takes a long time to train, we were not able to do normal cross-validation on all of the parameters. Below we have listed the hyper-parameters in the model, and how they where chosen.

Hyper-Parameter	Method
Epochs	Automatic stop if the model stops learning or is overfitting.
Batch Size	Experimental approach. Decided along with learning rate.
Learning Rate	Linearly decreasing if the model stops learning.
Depth of Network	Followed original U-Net paper
Filter Sizes	Followed original U-Net paper
Kernel Sizes	Used the standard (3x3) kernel.
Activation Function	Leaky Relu to avoid problems with vanishing gradient
Batch-normalization	Added as it increased performance
Dropout	Not added as it did not increase performance

Batch-size	Start LR	End LR	Batch Norm	Dropout
64	0.02	0.001	Yes	No

Final Values for hyper-parameters. LR - Learning Rate.

A. Batch-size and Learning Rate

Batch-size and learning rate closely connected when training a neural network. When starting training we spent a long time getting to a reasonable result, which may be explained by having a too small learning rate. In order to use a higher learning rate we decided to use a higher batch size, as a higher batch size gives a better approximation of the gradient. A high learning rate does however give a higher change of overshooting the minimum. Therefore we used a changeable learning rate which linearly decreases if the model stops learning (loss is non-decreasing). In Figure 5 we can see that we quickly get a high F1-score due to a high starting learning rate, and around epoch 13 we overshoot the minimum. This is corrected by lowering the learning rate.

B. Depth of Network and Filter-Sizes

These two parameters decides the complexity of our model. The depth of the network refers to how many contractions and expansions to have in the U-Net, and filter-sizes refers to how many feature-channels that should be made at each contraction.
At each attempt we started with low complexity to get some meaningful predictions and increased the complexity until the model started overfitting. In the end we ended up using the depth and filter-sizes as in Figure 3.

C. Regularization

To prevent overfitting we tried out two of the most common methods, dropout and batch normalization. Dropouts did not give any performance gains, but batch-normalization after the activation function significantly improved performance so this was used. Figure 5 shows the F1-metric during training, and as the two scores are very close to each other there is little sign of overfitting.

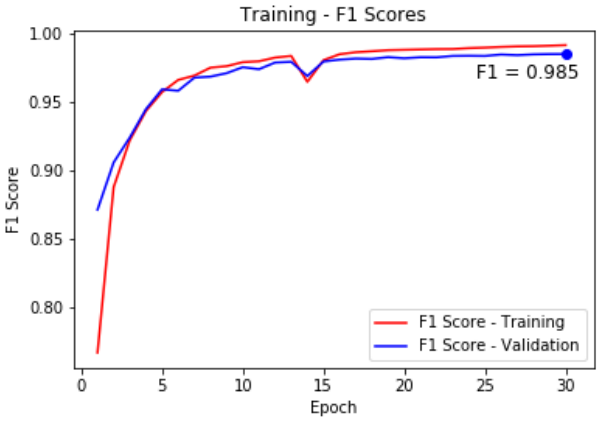

Fig. 5. F1-Score during training of neural net.

VII. Results

After iterating through all different hyper-parameters, patch sizes and computers we ended up with the best F1-score on AIcrowd being 0.902. This was achieved with patches of 200 pixels and a padding of 28 pixels. As can be seen in Figure 6 the model was predicting straight roads with ease while having problems as roads got too narrow or the surrounding of the roads were unusual and not only houses.

Fig. 6. A few of the submitted images with road prediction as overlay.

Figure 7 shows the F1-scores for the different techniques used in project. The worst result came from resizing the initial 400x400 train and 608x608 test images to 128x128, most likely due to information loss. By using the patching technique described in section V with (200x200) and 28 pixels in patches, we got the best result of 0.902.
We expected (400x400) to give better results than (200x200), but this may be explained by that we had to train on a smaller data-set due to memory issues.

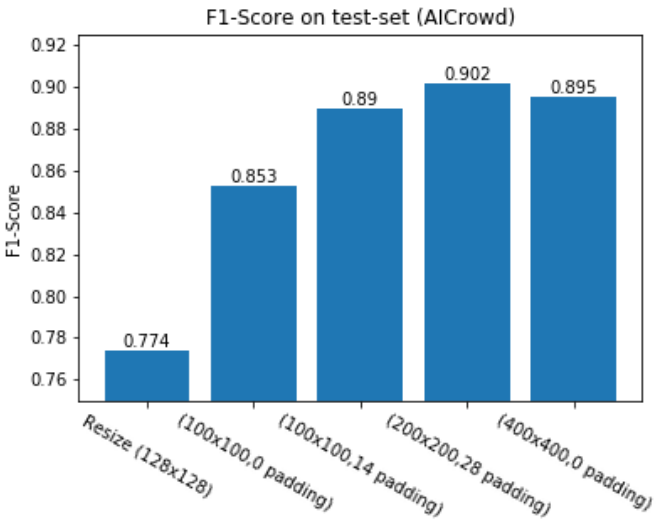

Fig. 7. Various U-Net input sizes and their corresponding F1-score on AICrowd.

VIII. Summary and Discussion

This report shows that with the right data augmentation and computational power a U-Net can segment satellite images with a satisfying result. However, computational power is crucial to when training as we need a lot of data, as well as complex models.
If one were to further improve upon our model we believe that greater computational power, as well as a richer data-set can improve the model. This would make it possible to train on larger image patches with enough data, and do proper cross-validation to tune the hyper-parameters perfectly. We also believe that post-processing of the images could improve the accuracy of the model.

References

[1] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597.