Road Segmentation Challenge - Project 2

Gianna Crovetto, Constance Gontier, Hendrik Hislberg
Machine Learning (CS-433), Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—In this project, we developed a binary classifier to identify roads in satellite images and distinguish them from the background. We compared various CNN and U-Net architectures to determine the best model for this task. Our approach also included image pre-processing steps to improve the performance of the classifier. Our results showed that the U-Net model achieved an F-1 score of 0.903. This project demonstrates the effectiveness of using U-Net model for road segmentation in satellite imagery.**

I. Introduction

In this project, we aim to perform image segmentation on satellite images to classify the pixels into two classes: roads and background. The images used in the project were taken from Google Maps and depict urban environments with various features, including roads, houses, and trees. However, accurately identifying roads in these images can be challenging due to factors such as trees, cars, shadows, and differences in road colors. To address this problem, we used machine learning techniques, specifically Neural Network techniques like CNN and U-Net, to classify the pixels in the images.

In addition to accurately identifying roads, we also want to avoid misclassifying background elements, such as buildings or parking lots, as roads. To achieve this goal, we used different technics of data augmentation. Through our classification, we were able to achieve a F-1 score of 0.903 using the U-Net model.

To begin our project, we experimented with a baseline model provided in the course. This allowed us to establish a baseline performance level and provided a foundation for further exploration.

II. Data exploration and analysis

A. Training set

Our training dataset consists of 100 satellite images, each of which is a 400x400 RGB image with corresponding ground truth images. These ground truth images are single channel, black and white images that have been manually annotated to identify roads (white) and background (black). For the purpose of this project, we are performing binary classification on patches of size 16x16. A patch is considered to be a road if at least 25% of its pixels are classified as such.

We observed that the satellite images in our dataset contain a range of challenges, including variations in road colors, shadows from trees, fluctuations in brightness, and the presence of railroads and parking lots. Additionally, a large number of roads in the dataset are oriented horizontally or vertically, which could lead to overfitting.

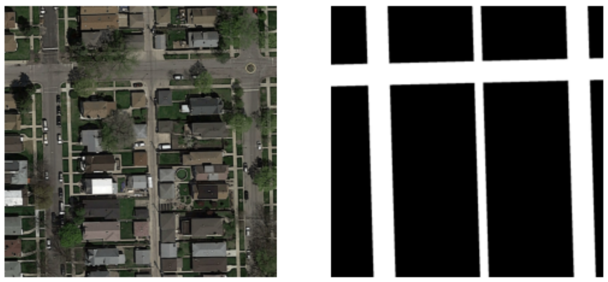

Fig. 1. Satellite Image and his corresponding Ground Truth

B. Score metric

While evaluating the performance of our classifier, we observed that the background class is more prevalent in the training dataset than the road class. As a result, relying only on accuracy as a measure of performance may not provide a reliable assessment of the classifier's effectiveness, as classifying all pixels as belonging to the background class would result in an accuracy of approximately 70%. To more precisely assess the classifier's performance, we rather consider the F1-score, which is a measure of the performance of a binary classifier that takes into account both precision and recall.

Precision is the ratio of true positive predictions to all positive predictions made by the classifier, and it reflects the accuracy with which the classifier is identifying the positive class (in our case, roads). Recall is the ratio of true positive predictions to all actual positive instances in the dataset, and it measures the classifier's ability to identify all instances of the positive class. The F1-score is the harmonic mean of precision and recall, and it provides a single metric that balances both factors.

As the F1 score is often used in cases where the classes are imbalanced, it is a particularly useful measure for our project, in which it is important to accurately identify roads while also avoiding misclassification of background elements as roads. By using the F1 score, we can ensure that the classifier is performing well in both of these areas.

III. Data augmentation

To increase the size and diversity of the training dataset, we implemented data augmentation techniques.

A. Transformations

1) Hsv: These techniques include random transformations of the hue, saturation, and value (HSV) of the images, which allow the model to learn to recognize roads and background elements under different lighting conditions, improving the robustness of the classifier.

2) Rotation: In our dataset, 85% of roads are oriented horizontally or vertically. We also apply random rotations by angles (-60, -45, -30, 30, 45, 60 degrees), allowing the model to learn to recognize roads and background elements from different perspectives. By rotating the images, we were able to introduce a greater variety of road orientations and reduce the risk of overfitting.

3) Flipped images: Horizontal and vertical flips allows the model to learn to recognize roads and background elements when they are flipped or mirrored, which can be useful in cases where the images are not always oriented in the same direction.

4) Blurring: Blurring the images helps the model to learn to recognize roads and background elements under different levels of image clarity, which can be useful in cases where the satellite images are not perfectly clear.

5) Edge detection: Finally, applying Canny edge detection to the images helps the model to learn to recognize the edges of roads and other objects in the images, improving the accuracy of image segmentation. Each of these techniques created a new version of the original image, allowing us to perform data augmentation. These techniques helped to improve the performance of the classifier by providing additional examples for the model to learn from and reducing overfitting.

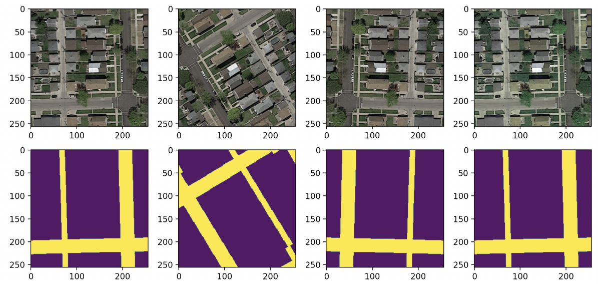

Fig. 2. Data augmentation - Satellite images with their corresponding ground truth below : (from the left) original image, tilted image, left/right flipped image and hsv transfromation

B. Balanced data set

During our analysis, we identified a deficiency in the number of images containing roads in our training dataset. To address this issue, we implemented additional data augmentation techniques specifically for images with a sufficient amount of road content, defined as images containing more than 28% of roads. By augmenting a larger number of these images, we were able to increase the overall proportion of images containing roads in the dataset and improve the performance of the classifier. We added additional image rotation and flipped on these images.

C. Image context

To prepare our training data, we tried several sizes of image patches, beginning with a size of 16x16. Eventually, we found that cropping the images to a size of 256x256 with an overlap of 56 pixels provided the best balance of computational efficiency and performance, we obtained an additional 24% performance. In addition to reducing computational costs and allowing for data augmentation, this cropping method helped to mitigate edge effects. By including an overlap, we ensured that the pixels at the edges of the cropped region were influenced by both the pixels within the cropped region and the pixels in the surrounding area, which helped to reduce the impact of edge effects and improve the overall performance of the classifier. We also normalized the RGB values between [0, 1] to standardize the data.

IV. Models and Methodology

A. CNN Architecture

In recent years, Convolutional Neural Networks (CNNs) have gained widespread popularity in the field of computer vision due to the availability of data and computational power. In this project, we used a simple CNN provided in the course as a baseline model and implemented it in PyTorch. We used this baseline model to compare the performance of other models and evaluate their effectiveness. The baseline architecture consists of 2 convolutional layers, 2 max pooling layers, 1 flattening layer, 2 fully connected layers, and a sigmoid activation function.

B. CNN Variation

1) Balancing: To address the imbalanced nature of our dataset, we implemented several approaches to balance the number of road and background images. Initially, we equalized the number of road and background images by selecting a similar number of images from each class. However, we found that this approach did not provide sufficient diversity in the road class, which was reflected in the performance of our model.

To improve the diversity of the road class, we implemented data augmentation techniques specifically for road images. This included applying random HSV transformations, random rotations, mirroring, blurring, and Canny edge detection to the road images. By augmenting the road images in this way, we were able to significantly increase the diversity of the road class and improve the performance of our model. Overall, these approaches helped us to achieve a more balanced dataset and improve the accuracy and robustness of our classifier.

2) Data augmentation: To train and test the baseline model, we used patches of size 16x16 and balanced the data. We also normalized the images by scaling them between 0 and 1. However, when tested on our dataset, the model achieved an accuracy of only 66%. We believe that the lack of context and a limited number of images contributed to the lower performance of this model.

As a result, we decided to try a different approach and predict pixel by pixel instead and to incorporate more context into the classification process. We modified the pre-processing step that involved cropping the images into 256x256 patches with an overlap of 56 pixels instead of 16x16 patches. In addition, we applied others data augmentation techniques as seen in chapter III.

C. U-Net Architecture

The U-Net model is a type of convolutional neural network that is particularly well-suited for tasks such as image segmentation, which require precise localization of features in an image. The U-Net model consists of an encoder, which extracts relevant features from the input image using convolutional and max pooling layers, and a decoder, which combines these extracted features using transposed convolutional layers. To help preserve spatial information, the U-Net model also includes skip connections between the encoder and decoder, which allow for the concatenation of the output of the encoder with the output of the decoder.

We used the Rectified Linear Unit (ReLU) activation function after each convolutional layer, and we also experimented with the use of batch normalization, although this did not improve the model's performance. In order to reduce the risk of overfitting, we also implemented dropout regularization. Overall, the U-Net model performed well in our experiments, achieving a F1 score of 0.903 on AIcrowd.

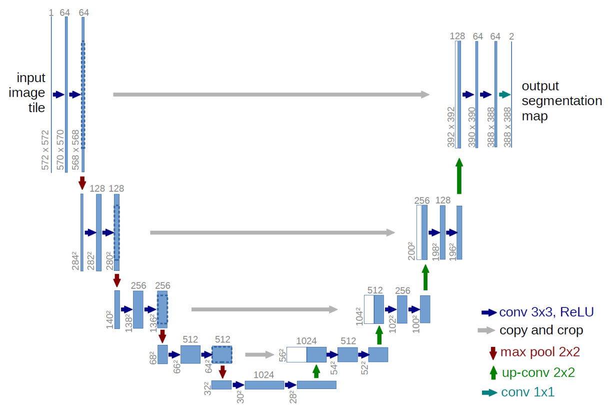

Fig. 3. Unet architecture

D. U-Net Variations

Once the architecture was established, we tried tuning it to get the best result out of it.

1) Input and output channels of convolutions: To further improve the performance of our model, we explored two main variations in the number of features at each convolution step. The first option was to increase the number of output filters from 16 to 256, while the second option was to increase the number of output filters from 32 to 512. Both variations used 3x3 kernels for the convolutions.

One key factor in the performance of a convolutional neural network is the number of features learned at each layer. Increasing the number of features can help the model learn more complex patterns and improve its ability to classify images accurately. However, increasing the number of features also increases the computational complexity of the model, which can make training and inference slower.

By comparing the two variations in the number of features, we were able to determine which option was the most effective in improving the performance of our model while still being efficient in terms of computational resources. This allowed us to strike a balance between accuracy and efficiency in our model design.

2) Padding and dilation: Based on our experimentation and analysis, we found that using padding and dilation set to 2 in the convolutional layers of the new U-Net architecture was the most effective approach. This method allowed the filters to have a larger receptive field, enabling them to capture more context and relevant features in the input images. As a result, this approach led to improved performance in terms of both accuracy and efficiency compared to using the default dilation and padding settings.

One of the main benefits of using a larger receptive field is that it allows the filters to capture more context and information about the surrounding pixels, which can be particularly important in image segmentation tasks where the spatial relationships between pixels are often important for correctly identifying objects or features in the image. In addition, using larger padding and dilation values can also help reduce overfitting by increasing the number of training samples and introducing more variability in the training data. Overall, our results show that carefully selecting the padding and dilation parameters can significantly impact the performance of a convolutional neural network, and it is important to carefully consider these factors when designing a model for a specific task.

3) Regularization: We implemented an Instance Normalization after every convolution except the last one. It worked better for us than the classic Batch Normalization. The difference is that Instance Normalization computes the mean and standard deviation for each object in the mini-batch. Normalizing is a good way to reduce the chance of overfitting. It also avoids the internal covariate to shift which could lead to huge losses. We also used dropout layers as a regularization method, which helped to reduce overfitting. Dropout, just as the name indicates, randomly drops a fraction of the nodes outputs. It allows the network to generalize and not to focus too much on some parts of an image, the network is then more robust.

4) Activation function: In our model, we used the Rectified Linear Unit (ReLU) activation function after every convolutional layer except the final layer. ReLU is a popular choice for activation functions in convolutional neural networks because it allows for efficient training and has been shown to improve the performance of many models. It is defined as the function $f(x) = max(0, x)$, which means that it sets all negative values to zero and leaves positive values unchanged. This allows the model to learn non-linear relationships in the data and improve its ability to classify images accurately. The final layer of our model uses the Sigmoid activation function, which maps the output of the model to a range between 0 and 1, representing the probability that a given pixel belongs to the road class.

V. Training Parameters

We encountered two main difficulties in this project. The first one is the lack of computational power. Thus, training the network was long. The second is the memory allocation. Augmenting the data renders a lot of images.

A. Hyperparameters

To optimize the performance of our model, we carefully selected the batch size and number of epochs to train on. After evaluating various combinations (Figure 4), we determined that a batch size of 16 was more appropriate and the best f1 score to choose our model over 100 epochs yielded the best results. This combination allowed the model to learn effectively while also not overfitting to the training data. We also monitored the training and validation loss to ensure that the model was generalizing well and not overfitting. By carefully selecting these hyperparameters, we were able to achieve good performance on the test set.

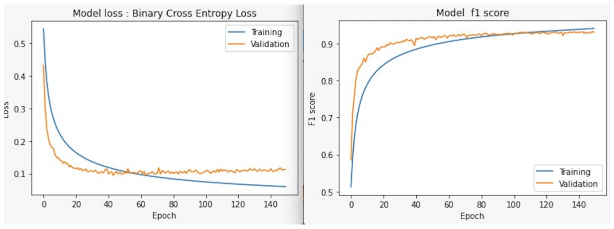

Fig. 4. Training and validation losses on the left, training and validation f1 score on the right, over 150 epoch

B. Optimizer and Loss function

To optimize our model, we used the Adam optimizer, with adaptive learn rate during epochs, with initial value at $lr = 1e - 3$, which is a popular choice for many deep learning tasks. The Adam optimizer is an adaptive learning rate optimization algorithm that adjusts the learning rate for each parameter based on the history of the gradients of that parameter. This helps to prevent oscillation and divergence of the optimization process, and allows the model to converge faster and more accurately. We found that using the Adam optimizer in combination with the Binary Cross Entropy loss function provided good results in terms of model performance and convergence speed. Here is the expression of the loss:
$loss_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(x_i) + (1 - y_i) \cdot log(1 - x_i)$
Here, x_i is our prediction and y_i is the ground truth label, and both $x_i, y_i \in \{0,1\}$. The prediction of our model represents the probability of the pixel to belong to class 0 (background) or 1 (road).

C. Evaluation

In the training process, we split our training data into train and validation sets to evaluate the model on unseen data during training. We used a 0.8 to 0.2 split ratio of the initial training set to create the validation set. As we discussed in chapter II-B, we used F1-score to evaluate our performances. The F1-score is a better metric for unbalanced datasets. Just using accuracy would be unreliable.

VI. Results

Here are the results of our predictions:

Model	**Training F1**	**Validation F1**	**AI score**
Baseline CNN	0.77	0.70	0.66
UNet: (2,2) padding and (2,2) stride	0.94	0.93	0.899
UNet: (1,1) padding and (1,1) stride	0.95	0.94	0.903

We have a slightly better score with the UNet with standard padding and dilation of 1. We thought we would have a better score using (2,2) dilation because that would allow the convolution to have more spatial information.

Here is an example of our prediction:

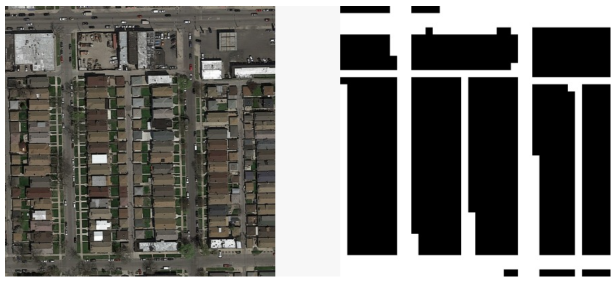

Fig. 5. Prediction image with his corresponding satellite image

VII. Discussion

All in all, we are satisfied with the results we achieved at the end of the project that is to say an f1 score of 0.903. This demonstrates that our model was able to accurately classify road pixels in the images. One key factor that contributed to this success was the augmentation of the patch size, which provided more context for the model to consider during the classification process. However, we did observe some limitations in the model's performance, such as difficulty in distinguishing roads from tree branches or shadows. In the future, we could consider implementing additional data augmentation techniques specifically designed to address these challenges. Overall, our results show that our model was able to effectively classify road pixels in images, and we believe that with further improvement, it has the potential to be a reliable tool for identifying roads in real-world applications.

VIII. Summary

Through this project, we learnt how to improve a Neural Network to adapt it to our application. We understood how our dataset was composed and how to augment it to achieve the wanted task correctly. Tuning the hyperparameters was also an important part of our work. You can see the UNet architecture we used on the appendix below.

Acknowledgements

The authors thank the Machine Learning teachers: Martin Jaggi and Nicolas Flammarion for their interesting classes as well as all the assistants for their help during all the project.

IX. Appendix

UNet architecture:

	Layers	Parameters	Output
2x	Convolution InstanceNorm ReLU	32 filters, (3,3) kernelsize -	C2
	MaxPool	(2,2) window size	
2x	Convolution InstanceNorm ReLU	64 filters, (3,3) kernelsize -	C4
	MaxPool	(2,2) window size	
2x	Convolution InstanceNorm ReLU	128 filters, (3,3) kernelsize -	C6
	MaxPool	(2,2) window size	
2x	Convolution InstanceNorm ReLU	256 filters, (3,3) kernelsize -	C8
	MaxPool	(2,2) window size	
2x	Convolution InstanceNorm ReLU	512 filters, (3,3) kernelsize -	
	Transposed convolution Skip connection	256 filters, (2,2) filters, (2,2) stride Concatenate(C8,TP8)	TP8
2x	Convolution InstanceNorm ReLU	256 filters, (3,3) kernelsize -	
	Transposed convolution Skip connection	128 filters, (2,2) filters, (2,2) stride Concatenate(C6,TP6)	TP6
2x	Convolution InstanceNorm ReLU	128 filters, (3,3) kernelsize -	
	Transposed convolution Skip connection	64 filters, (2,2) filters, (2,2) stride Concatenate(C4,TP4)	TP4
2x	Convolution InstanceNorm ReLU	64 filters, (3,3) kernelsize -	
	Transposed convolution Skip connection	32 filters, (2,2) filters, (2,2) stride Concatenate(C2,TP2)	TP2
2x	Convolution InstanceNorm ReLU	32 filters, (3,3) kernelsize -	
	Convolution Sigmoid	1 filter, (1,1) kernelsize	

References

[1] Olaf Ronneberger, Philipp Fischer, Thomas Brox (2015) *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, Vol.9351: 234–241. Link: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

[2] https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47