Synthetic and real data for human aid and disaster relief for residential area flooding scenarios

Final Project Report - Foundation of Deep Learning

Aiza Avula Canibe
DSBA M2
ESSEC Business School and Centrale Supélec, France
aiza.avila-canibe@student-cs.fr

Aline Helburg
DSBA M2
ESSEC Business School and Centrale Supélec, France
aline.helburg@student-cs.fr

ABSTRACT

Semantic aerial image segmentation is a process of identifying and classifying different features in aerial images, such as buildings, roads, and vegetation. This is a challenging task due to the high resolution and variability of aerial images. In recent years, deep learning techniques have been shown to be effective in semantic aerial image segmentation. In this paper, we propose a deep learning-based approach for semantic segmentation of aerial images. We evaluate their performance on a publicly available dataset - 374 aerial images of the Houston area following Hurricane Harvey in 2017. Our proposed method achieves relevant and accurate results on the benchmark dataset, demonstrating its effectiveness and potential for real-world applications - as it achieved an accuracy of 75%. The study highlights the relevance of utilizing deep learning methods for accurate and efficient semantic aerial image segmentation.

INTRODUCTION

The project presented in this research paper focuses on the use of deep learning models for aerial image segmentation in the context of a natural disaster. Our goal is to develop an accurate dense segmentation model for aerial post-flood aerial imagery.

More precisely, we are trying to identify property attributes in the Houston area following Hurricane Harvey in an effort to autonomously generate a post-flood map. Our final goal is to accurately identify and segment common large and small assets in residential Houston after Hurricane Harvey that may or may not be damaged by Hurricane Harvey.

Semantic aerial image segmentation offers several advantages over other techniques, making it a powerful tool for extracting valuable information from aerial images.

- *High accuracy*
- *Automation*: it can process large amounts of data quickly, making it more efficient than manual image analysis.
- *Handling large datasets and high-resolution images*

To answer this problem, we seek to answer the following questions: How can we develop the most relevant and accurate deep learning model? Under which parameters and conditions?

Potential applications of aerial post-flood maps are:

- *Disaster response*: they can be used to quickly assess the extent of flooding and damage, which can aid in the response efforts by emergency services.
- *Flood risk management*: These maps can be used to identify areas that are at high risk of flooding, and thus help in the development of flood risk management plans.
- *Insurance claims*: they can be used to support insurance claims by providing detailed information.
- *Urban planning*: They can be used to identify areas that are susceptible to flooding, which can be taken into account in urban planning decisions.
- *Restoration and rebuilding*: they can be used to identify areas that need to be restored and rebuilt following a flood event.
- *Drainage and infrastructure design*: authorities can use the maps to design better drainage and infrastructure systems to reduce the risk of flooding in the future.

1 PROBLEM DEFINITION

The project we are working on involves performing semantic image segmentation in order to categorize aerial images. The challenge we are facing is how to find, develop and adjust the best deep learning model in order to have the highest accuracy. In this specific challenge, we wanted to maximize the accuracy of our model and minimize the loss function.

The proposed algorithm aims to accurately identify and segment different land cover classes, such as water, buildings, roads, and vegetation, in post-flood aerial imagery.

2 METHODOLOGY AND TECHNIQUES USED

The dataset includes hundreds of images of residential areas in Houston captured by drones in 2017. For each image, a detailed segmentation mask has been created for specific classes, and the masks have been converted into vector form for use in modeling,

visualization, and analysis. To develop an accurate dense segmentation model, we followed the following steps.

1. Data collection

High resolution image data of Houston, Texas, USA were provided via GeoEngine. The dataset consists of 299 Tag Image File Format (TIFF) that have 299 matching masks (PNG format) (the training dataset) and 75 images without their matching mask (the validation dataset). Hence, the goal of this project is to determine the 75 matching masks - the final score will be determined by this validation dataset.

2. Data exploration

We did this step in order to get familiar with the data. For example, we looked at the repartition of the 27 labels to see which ones are the most used.

3. Data preprocessing

We separated the data between a train folder and a validation folder. The validation folder contains the images that do not have a matching mask. We resized the pictures to reach square images of 256x256 pixels.

4. Data augmentation and transformation

We choose to do a data augmentation in order to artificially expand the size of the training dataset by creating modified versions of images in the dataset. We randomly picked 45 images to implement this data transformation. We applied horizontal and vertical flips as well as a rotation (180°) (15 images per transformation) to these original images. Our goal was to expose the model to different variations of the same image, making it more robust to variations in the images it will encounter during testing. Thus we can improve the performance of a model by reducing overfitting and increasing the generalization of the model.

5. Deep learning architecture development

The development of deep learning architectures was the main task, the most time consuming but the accuracy of our model depends on the implementation and parameters of the chosen model(s). We tried several models in order to select the one that best fits our dataset and provides the highest accuracy. Different models may have different strengths and weaknesses, and the best model for a task will depend on the specific characteristics of the data.
By comparing the performance of different models on a validation dataset, we can determine which model generalizes best.

5.1. UNet

Our first try was a UNet model. We choose UNet because it is a popular architecture for image segmentation tasks thanks to its ability to preserve both spatial and semantic information during the encoding and decoding process.
A UNet is a type of convolutional neural network. The architecture of a UNet consists of a contracting path (the encoder) followed by an expansive path (the decoder). It uses a U-shaped architecture - hence the name.

The contracting path is made up of several layers of convolutional and max pooling operations that reduce the spatial dimensions of the input image, while increasing the number of feature maps. The encoder follows the following formula: two 3x3 convolutions, a convolution layer is followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation. The decoder is made up of several layers of convolutional and up-sampling operations that increase the spatial dimensions of the feature maps, while reducing the number of feature maps.

To function optimally on semantic segmentation problems, UNet utilizes two types of information: high level spatial and contextual features in the expansive path filters, and detailed fine-grained structural information in the contraction path.

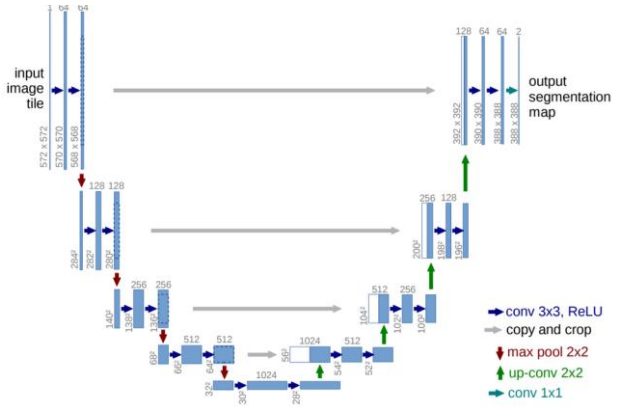

*Figure 1. U-net architecture (example with 32*32 pixels which is the lowest resolution). Each blue box represents a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows represent the different operations.*
[Ronneberger et al., 2015]

As an optimizer parameter, we used Adam - Adaptive Moment Estimation. We choose this specific optimizer because it has interesting properties in our case of image segmentation:

- *Adaptive learning rate*: it adapts the learning rate on a per-parameter basis, which can help the model converge faster and have better performance. This is important in image segmentation tasks, the data is highly variable and the model may need to adjust its learning rate.
- *Robustness to noise and outliers*: it uses the moving averages of the gradients to estimate the mean and variance, this helps to reduce the impact of noise and outliers in the gradients. This can help the model to converge more quickly and achieve better performance.
- *Computationally efficient*: it does not require the calculation of the inverse of the Hessian matrix which makes it faster to compute.

- *Regularization*: it includes a weight decay parameter - can be used to apply L2 regularization to the parameters. It prevents overfitting.
- *Bias correction term*: this term is useful to improve the accuracy of the moving averages at the beginning of the training when the moving averages have not yet had enough time to converge.

We used 18 epochs and a learning rate of 0.001. The optimum number of epochs and learning rate was determined after a few trials. Having a higher number of epochs may lead to overfitting -the model should not "learn too much" about the training dataset in order to generalize well on the testing and validation dataset.
We combined the UNet implementation with a loss function in order to measure the gap between the predicted label and the expected label. UNet uses a loss function for each pixel of the image.
We choose to use the cross-entropy loss function which is commonly used for multi-class classification tasks. A soft-max function is first applied to the output for every pixel.

$$p(s)_i = \frac{e^{s_i}}{\sum_j^C \quad e^{s_j}}$$

Formula 1. Soft-max formula with i the ie pixel, C the number of classes (here 27) and s the score that results of the neural network for each pixel.

Then, the cross-entropy loss function is applied to the resulting predicted probability.

$$CE = -\sum_i^C \quad t_i log(p(s)_i)$$

Formula 2. Cross-entropy loss formula with i the ie pixel, C the number of classes (here 27) and p(s) the output of the soft-max function (from Formula1)

Furthermore, we used a dice score to evaluate our model - this metric will be explained in the evaluation section.

5.2. Combining UNet with ResNet encoder

ResNet is a Convolutional Neural Network architecture, it consists of a series of residual networks. The architecture uses 'skip connection', a method to bypass some layers allowing the gradients to flow more easily through the network. It will thus skip layers that diminish the performance. This CNN lowers the error rate because the error is caused by a vanishing or exploding gradient.
We use the encoder part of the ResNet as the backbone network for the UNet. We used the output from the ResNet as the input for the decoder of the UNet. This will provide the UNet with the improved feature representation capabilities of the ResNet while still allowing it to perform pixel-wise prediction tasks.

We used 20 epochs and a learning rate of 0.0001. The optimum number of epochs and learning rate was determined by experimenting. The optimizer we applied is Adam (cf. section 5.1). The loss function is the cross-entropy loss function (cf. section 5.1).

5.3. Combining PSPNet with ResNet encoder

PSPNet (Pyramid Scene Parsing Network) is a deep neural network architecture for semantic image segmentation. It is designed to segment an image into regions corresponding to different objects or parts of objects. PSPNet is based on a pyramid pooling module which is able to capture contextual information at different scales, and a residual network architecture which allows for efficient training and inference.

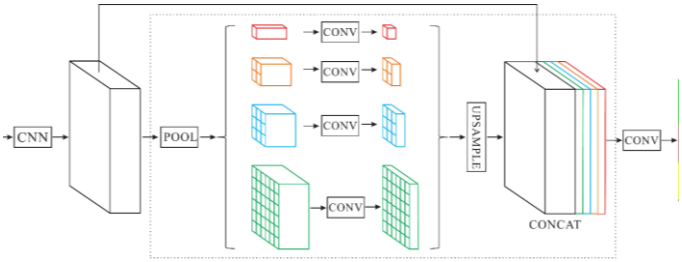

Figure 2. PSPNet architecture. The image input is on the left, the output is on the right.
[Zhao et al., 2017]

We use the encoder part of the ResNet as the backbone network for the PSPNet. This way the ResNet's encoder can extract feature maps from the input image. Those features are then passed through the pyramid pooling module of the PSPNet. The resulting output is combined with the feature maps from the ResNet encoder, which are then passed to the decoder part of the PSPNet to perform the semantic segmentation.
The ResNet architecture is well suited for the encoder part of the PSPNet model because it is a very deep and powerful architecture that can extract rich and high-level features from the input image. The residual connections in ResNet also help to mitigate the vanishing gradient problem, allowing the network to learn deeper and more complex features.
We used 25 epochs and a learning rate of 0. 001.The optimizer we applied is Adam (cf. section 5.1.). The loss function is the cross-entropy loss function (cf. section 5.1.)

6. Model training

Finally, we train our model on the training dataset. We created the training and validation dataset with the function train_test_split from the Scikit-learn library. The test dataset represents 10% of the whole dataset - this number was arbitrarily chosen.

3 EVALUATION

In order to evaluate our models, we used different metrics. We took into consideration:

- Accuracies on both the training dataset and validation dataset. To prevent overfitting, we had to make sure that the gap between the accuracies on both dataset was not too big as it shows that the model cannot generalize well.
- Cross-entropy losses of the training dataset and validation dataset: we are looking to minimize the loss. To better visualize the performance with respect to the number of epochs, we used graphs.
- Dice score (F1 score). It is used to evaluate the similarity between two sets of data.

1. Accuracies

Model accuracy is defined as the number of classifications correctly predicted divided by the total number of predictions.

Table 1. Accuracies obtained from the models

Model	**UNet**	**UNet x ResNet**	**PSPNet x ResNet**
Accuracy (train)	32.5%	57.3%	76.9%
Accuracy (validation)	24.9%	58.1%	73.4%
Accuracy (testing) → accuracy that was calculated on granular.io			75.47%

The model combining PSPNet with the ResNet encoder has the highest accuracy.

In the first UNet model, we did not apply any data transformation and data augmentation (which we did afterwards, see section 2.3.). Furthermore, the split share between the training dataset and the validation was 30% (not 10% as for the rest of the project). The accuracy on the train set was 12% on the training set and 9% on the validation dataset. Thus, we can acknowledge that data transformation and data augmentation was useful in our case and can influence the performance of the model.

2. Cross-entropy losses

As explained in section 2.5.1., we used the cross-entropy loss function. The three models gave us the following results:

Table 2. Losses obtained from the models

Model	**UNet**	**UNet x ResNet**	**PSPNet x ResNet**
Loss (train)	1.12	1.21	0.65
Loss (validation)	1.01	0.95	0.78

As for the accuracies, the PSPNet x ResNet model has the lowest cross-entropy loss function among the models.

The results have been plotted in order to see the evolution as the number of epochs increases.

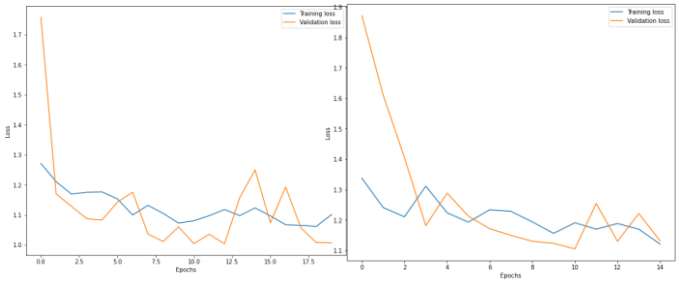

Figure 3 and 4. Loss of the UNet model (left) and of the 'beta' UNet model (right) with respect to the number of epochs

We plotted the loss function of the 'beta' model (cf. section 3.1.). We can see that the loss was higher.

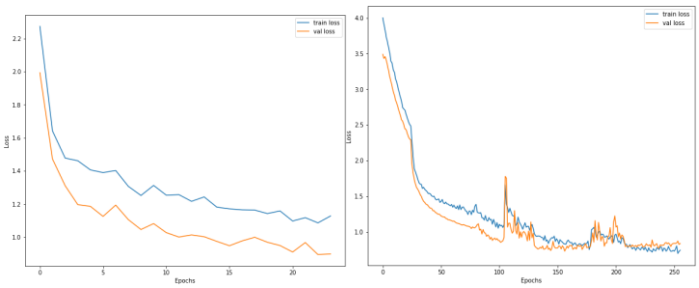

Figure 5 and 6. Loss of the UNet x ResNet (left) and of the PSPNet x ResNet (right) with respect to the number of epochs.

We notice that the losses decrease with the number of epochs.

3. Dice score

The dice coefficient/score is a metric used to evaluate the similarity between two sets of data. The score ranges between 0 and 1, with 1 indicating that the two sets are identical and 0 when there is no similarity. Here, we are comparing the images and their masks - hence, we are trying to achieve a score close to 1 as it will show that the image and the matching mask are similar. It is commonly used in the field of computer vision, natural language processing, and image segmentation.

The dice score is calculated as below:

$$dice\ score = \frac{2 * |X \cap \quad Y|}{|X| + |Y|}$$

Formula 3. Formula of the dice score with |X| and |Y|

Table 3. Dice score obtained from the models

Model	**UNet**	**UNet x ResNet**	**PSPNet x ResNet**
Score (train)	0.18	0.20	0.29
Score (test)	0.13	0.27	0.36

Once again, the evaluation metric demonstrates the model that combines PSPNet with ResNet encoders has the better dice score.

4 DISCUSSION

The results obtained from the PSPNet model and ResNet encoder are relevant and allow us to use the model developed for future applications.

We could have improved the model by adjusting better the hyper parameters - increasing the number of epochs for example. Some related works recommend around 100 epochs for an image classification problem. However, our main constraint was the computational power available - training on 20 epochs was the maximum we could achieve without running out of RAM or GPU.

5 CONCLUSION

In this project, we manage to find an accurate model that fits our dataset and is able to predict labels on others images.

Our aim was to find the best model for aerial image semantic segmentation using several deep learning models. We evaluated several models, including UNet PSPNet and ResNet, and found that the PSPNet model combined with ResNet encoder performed the best in terms of overall accuracy, loss function and dice score. Additionally, we also explored the effect of not doing data augmentation and transformation, it decreases the model's overall performance. These findings demonstrate the potential of deep learning for semantic segmentation of aerial images and can be used to improve the accuracy and efficiency of mapping and can be expanded to other real-life applications. Future work could focus on incorporating additional data sources, in order to increase the number of classes/labels, to further improve the performance of the model.

REFERENCES

[1] Ronneberger Olaf, Fischer Philipp Fischer and Brox Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation (2015). arXiv:1505.04597v1

[2] Jadon Shruti. A survey of loss functions for semantic segmentation. (2020). arXiv:2006.14822v4

[3] Wang Wei. Using PSPNet and UNet to analyze the internal parameter relationship and visualization of the convolutional neural network (2020). arXiv:2008.03411v1

[4] Bertels Jeroen. Optimizing the Dice Score and Jaccard Index for Medical Image Segmentation: Theory & Practice (2019). arXiv:1911.01685v1

[5] Wang Wei. Using UNet and PSPNet to explore the reusability principle of CNN parameters (2020). arXiv:2008.03414v1

[6] La Wei-Sheng et al. Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks (2018). arXiv:1710.01992v3

[7] He Kaiming et al. Deep Residual Learning for Image Recognition (2015). arXiv:1512.03385v1

[8] Kingma Diederik, Ba Jimmy Lei Ba. Adam: a method for stochastic optimization (2017). arXiv:1412.6980v9

[9] Zhao Hengshuang et al. Pyramid Scene Parsing Network (2017). arXiv:1612.01105v2

[10] Bangar Siddhesh. Resnet Architecture Explained. July 5 2022. Medium. https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d

[11] Pavel Iakubovskii, Segmentation Models Pytorch (2019). GitHub. http://github.com/qubvel/segmentation_models.pytorch

[12] Koppert-Anisimova Inara. Cross-Entropy Loss in ML. Jan 4 2021. Medium. http://medium.com/unpackai/cross-entropy-loss -in-ml-d9f22fc11fe0

[12] K Bharath. U-Net Architecture For Image Segmentation (2021). https://blog.paperspace.com/unet-architecture-image-segmentation/

[13] Lamba Harshall. Understanding Semantic Segmentation with UNET. Feb 17 2019 Towards Data Science. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[14] Davies Andrew Joseph. Semantic Segmentation of Aerial Imagery Using U-Net in Python. March 31 2022. Medium. https://towardsdatascience.com/semantic-segmentation-of-aerial-imagery-using-u-net-in-python-552705238514