Semantic Segmentation of residential properties and assets after Hurricane Harvey

Jyotishka Das

CentraleSupélec & ESSEC Business School, Paris

1 Introduction

Hurricane Harvey was a devastating Category 4 hurricane that made landfall on Texas and Louisiana in August 2017, causing catastrophic flooding and more than 100 deaths. It caused 125 billion dollars in damage, according to the National Hurricane Center. That's more than any other natural disaster in U.S. history except Hurricane Katrina.

We tried to develop an accurate dense segmentation model for aerial post-flood aerial imagery [1, 5]. The goal was to identify property attributes in the Houston area following Hurricane Harvey in an effort to autonomously generate a post-flood map. High-resolution image data of Houston, Texas, USA were provided to participants via Geo-Engine. The aim was to leverage the provided drone image dataset in order to train segmentation models to better understand ground conditions after Hurricanes and 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.

The code of this project is hosted in GitHub[1].

2 Preliminaries, Tools and Techniques

Semantic segmentation is a challenging research topic, and multiple solutions have emerged over the years to tackle this problem. Among the possibilities, we can cite SegNet [2], U-Net [6], PSPNet [8] and DeepLabv3+ [4].

U-Net [6] is a neural network architecture for semantic segmentation. The main idea behind the architecture is to find a reduced features' representation of an image, the encoder part, and later expand the representation, the decoder part, resulting in a semantic mask of the parts of the image. The encoder part of the architecture follows a similar part of many convolutional networks, while the decoder consists of upsampling and convolution layers. The entire feature map is passed from the encoder to the decoder part with the same dimension on the U-net architecture.

Differently from the previously developed fully convolutional networks, PSP-Net, [8], takes into account a global context, which improves its accuracy. The

[1] https://github.com/dasjyotishka/Semantic-Segmentation-of-residential-properties-and-assets-after-Hurricane-Harvey

PSPNet encoder is similar to the other networks, but the two last layers are replaced with dilated convolutional layers [7] which helps the encoder to capture more information. Following the encoder, there is a pyramid pooling module that helps learn global information from the image. This module pools from the encoder on different sizes, passing through a convolution and pooling layer, then each size is concatenated after being upsampled to the same size. Finally, the decoder generates the resulting semantic mask.

DeepLabV3 [3] is a semantic segmentation architecture designed by a Google Research group. The architecture employs both a pyramid pooling module and an encoder-decoder architecture, but it improves previous works by adding to the architecture atrous separable convolutions. DeepLabV3 outperformed PSPNet in the 2016 ILSVRC Scene Parsing Challenge [14]. DeepLabV3+ [4] improves on DeepLabv3 with a new decoder module that can better predict objects boundaries. It has several improvements over DeepLabv3, such as adding a simple yet effective decoder module to achieve an encoder-decoder structure. The encoder module processes multiscale contextual information by applying dilated convolution at multiple scales, while the decoder module refines the segmentation results along object boundaries.

Deeplabv3+ encoding structure is composed of ResNet-101 network and void space pyramid module, in which ResNet-101 is mainly divided into 5 parts. The first part is a convolutional layer of 7×7×64, and the other 4 parts contain 3, 4, 23, and 3 blocks respectively. As shown in the Fig.1., each Block is composed of three convolutional layers plus shortcut connections. The void space pyramid pooling module is connected at the end of the ResNet-101 network to obtain the semantic information of multi-scale images. The decoding structure performs 4-fold bilinear up-sampling on the feature map output by the encoder network, and then merges it with the low-level feature information at the channel level to restore the spatial information lost during the downsampling process.

We have used a DeepLabv3+ architecture to train our semantic segmentation model. It has been quite popular among segmenting images in urban settings and have been used in several competitions in the past.

3 Experimental Setup

3.1 Dataset preparation

The dataset used for this project is the Hurricane Harvey dataset, which contains 374 drone images of residential areas in Houston, each accompanied by a corresponding classification mask that can be used during the evaluation process to visualize how the model performs. An example of an image and its segmentation mask is shown in Fig.2.

Additionally, another dataset is accessible, called Hurricane Harvey Synthetic, which is a synthetic dataset containing 2093 images and 156,933 annotations that large models can be trained on, in order to improve the performance of the model.

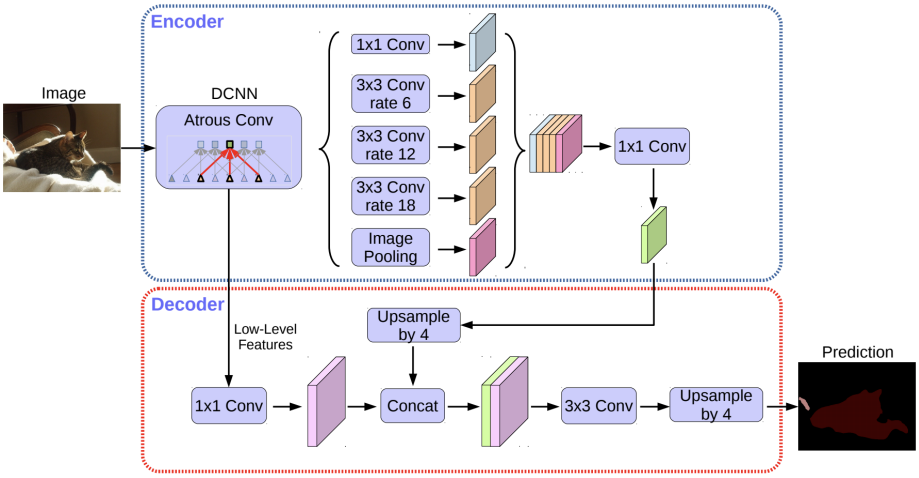

Fig. 1. Architecture of DeepLabv3+ model

The objective is to well identify the attributes of properties in Houston using the drone imagery from the Hurricane Harvey dataset. The dataset includes 26 different potential property attributes:

1. Property Roof
2. Secondary Structure
3. Swimming Pool
4. Vehicle
5. Grass
6. Trees / Shrubs
7. Solar Panels
8. Chimney
9. Street Light
10. Window
11. Satellite Antenna
12. Garbage Bins
13. Trampoline
14. Road / Highway
15. Under Construction / In Progress Status
16. Power Lines / Cables
17. Bridge
18. Water Tank / Oil Tank
19. Parking Area - Commercial
20. Sports Complex / Arena
21. Industrial Site
22. Dense Vegetation / Forest
23. Water Body

24. Flooded
25. Boat
26. Parking Area

Fig. 2. Original image and classification mask

3.2 Preprocessing

The training images were downloaded from the competition website are stored in a folder called train images, and the corresponding masks were stored in a folder

called train masks. Two blank folders were also created in the same directory called test images. Now, for every item in the train images, we checked whether its corresponding mask existed in the train mask. If it did not exist, we moved the image from the train-images folder to the test-images folder.

To reduce the possibility of overfitting, we splitted the dataset into two parts, the training images, and the test images. the train-image folder contained 299 images and the test-image folder contained 75 images. For training the model, we further splitted the training dataset into train and validation dataset. Hence, we extracted 15% of the images (45 random images) and their corresponding masks from the training folders to the validation folders. Now, at this stage, the training images and masks folders would contain 254 images for both, and the validation images and masks folders would contain 45 images. We created blank folders where the tensors would be stored and stored the corresponding tensors in those folders. We iterated through all masks and images in the training set, resized them, transformed them into tensors, and stored them in the new folders.

For the next step, we loaded the image and the mask tensors, then converted the grayscale mask tensors from the dimension [1, H, W] to [H, W] because the loss function accepts images in the form [B, H, W], where B is the batch-size, H is the height and W is the width of the images. After that, an ad-hoc transformation of the images was performed to improve the segmentation accuracy. If the images were too bright or too dark, then we re-scaled their brightnesses.

4 Methodology

4.1 Model selection

Before building the model, two different models were considered for our object classification task : U-Net and DeepLabv3+. However, these two models are suited to different tasks. DeepLabv3+ is a general-purpose model that uses multiscale feature pyramid networks (FPNs) to improve the resolution of the segmentation map, and it also uses an encoder-decoder architecture, with advanced feature extraction in the encoder and fine-grained feature refinement.

Compares to U-Net, one of the key advantages of DeepLabv3+ is that it can effectively handle objects at different scales, which is important for image classification tasks where objects in an image can be relatively small or large. U-Net, on the other hand, is an encoder-decoder architecture, and it is specifically designed for medical image segmentation tasks, where images are of high resolution and the target objects are small. U-Net uses skip connections between the encoder and decoder in order to keep most of the high-resolution information in the images. For such reason, U-Net is relatively slower than DeepLabv3+, which is more efficient in large datasets.

Finally, I chose to use **DeepLabv3+** over U-Net because a large number of high-resolution images are available in our synthetic dataset and can be used for pre-training our model for specific tasks such as object detection in our case.

4.2 Network Training

The DeepLabv3+ model with a ResNet-101 encoder was selected. It was pre-trained on the Imagenet images for obtaining a better accuracy. The model was simulated in a Jupyter notebook file hosted in Google Colab using K80 GPU and 12 GB of RAM. The prediction model took almost 45 minutes to be built. The model was trained using many alterable hyper-parameters. A list of all the trainable hyper-parameters which produced the highest performance metrics, are summarised below:

- Epochs: 50
- Learning Rate: 0.0001
- Optimizer: Adam
- Batch Size: 10

The same training configuration was used for all networks tested. The loss function we chose was categorical cross-entropy, the optimizer we chose was the standard configuration of the Adam optimizer.

The reason why I chose Loss function Cross-entropy were:

1. It is easy to compute: Cross-entropy is a computationally efficient loss function that can be easily calculated using simple operations such as taking the negative logarithm.
2. It can handle imbalanced dataset: Cross-entropy loss is less sensitive to imbalanced datasets, and it can produce more accurate results, as it tends to focus more on the misclassified examples.

The reason why we chose Adam optimizer were:

1. Adaptive learning rate: Adam optimizer can adjust the learning rate for each parameter individually, which allows the model to converge faster with better stability.
2. Faster convergence: Adam optimizer is known to converge faster than other optimization algorithms such as stochastic gradient descent (SGD). Due to the fact that it uses a combination of gradient information and historical gradient information to update the parameters.
3. Low memory requirements: Adam optimizer requires low memory storage, which makes it suitable for use with large datasets and models.

5 Evaluation

5.1 Loss Function

Fig.3 illustrates the loss function while training the model. It shows that the loss function is decreasing along with the number of epochs, which indicates that our model's performance is improving when passing through more epochs. This means that the model is learning from the training data and is becoming better at making predictions.

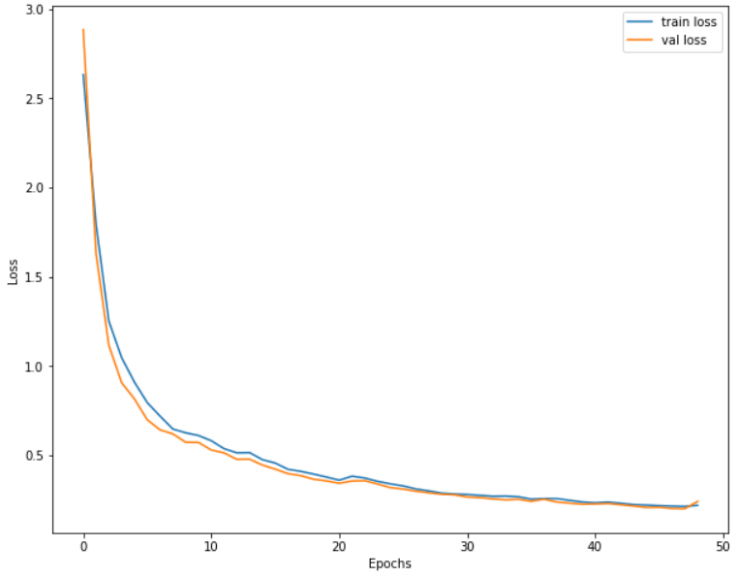

Fig. 3. Evolution of loss function along with epochs

Overfitting occurs when a model becomes too complex and starts to memorize the training data, which leads to a poor generalization of new data. The decreasing pattern of our loss function indicates that the model is not overfitting till 50 epochs, as it is able to generalize well to the training data.

Another fact we can conclude from Fig.3 is that the learning rate we chose is appropriate in a way that our model is able to make progress in each iteration. The optimal point is at the 46th epoch where it gives the highest validation accuracy as well as the lowest loss. Hence, we loaded the model at epoch 46 for which we wanted to generate prediction masks and then generated the masks. We generated the prediction masks and resize the mask sizes to that of their corresponding image sizes efore submission.

5.2 Accuracy and Dice-Score

The Dice score coefficient was used to calculate the accuracy of our segmentation. The spatial overlap between two segmentations, the original mask A and the predicted mask B target regions is measured through the Dice Coefficient, and is calculated as:

$$\text{Dice Score (A,B)} = 2 \times \frac{(A \cap B)}{(A \cup B)}$$

Our model achieved a high dice-score of 92.9% at the optimal epoch of 42 when validated against the validation dataset, and a dice-score of 72.6% on final submission to the challenge.

5.3 Predicted Mask

The next step we did after getting a decent dice score for the optimal epoch from our model was to check the difference between the original image, the original mask, and the predicted mask obtained from the model, as shown in Fig.4.

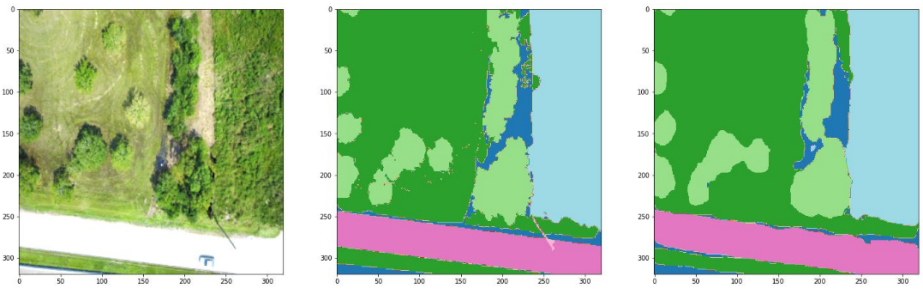

Fig. 4. Original image, Original mask and Predicted mask

Comparing the predicted mask (far right) with the original mask (in the middle), it was found that our model was relatively accurate in predicting roads (pink), country lanes (light green), areas covered with vegetation (dark green), and isolated objects(blue). However, on some smaller objects, our model is less accurate than we expected.

5.4 F1 Score

Another metric we used to measure model performance is the F1 score, which takes both precision and recall into account. The F1 score is the harmonic mean of precision and recall given by the formula as follows:

$$F1 \text{ Score } = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

- Precision is the number of true positive predictions divided by the number of true positive and false positive predictions.

- Recall is the number of true positive predictions divided by the number of true positive and false negative predictions.

The model had a high F1 score of around 0.6 when tested against the validation dataset.

6 Conclusions

A DeepLabv3+ based segmentation model with a ResNet-101 encoder and pre-trained on Imagenet images was utilised to solve a challenge involving utilisation of deep learning to perform semantic segmentation on residential properties and assets after Hurricane Harvey. It provided a improved dice-score of 76.25% over an U-Net model on test dataset on final submission.

References

1. Hurricane harvey segmentation challenge (2022), https://engine.granular.ai/organizations/centralesupelec-deep-learning/tasks/6373eb4b7930b27221315989/overview
2. Badrinarayanan, V., Kendall, A., SegNet, R.C.: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 5 (2015)
3. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
4. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
5. Goswami, S., Verma, S., Gupta, K., Gupta, S.: FloodNet-to-FloodGAN : Generating Flood Scenes in Aerial Images (2022)
6. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
7. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision 115(3), 211–252 (2015)
8. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)