# DATS 6203 Individual Report

## Image Segmentation and Feature Detection with U-Net

### Introduction

Image classification, segmentation, and feature detection are extremely important machine learning applications used in many fields, including medical imaging,[[1]](#footnote-1) land use,[[2]](#footnote-2) and general object detection.[[3]](#footnote-3) Developed in 2015, the U-Net is a fully convolutional network (FCN) that performs image segmentation very well.[[4]](#footnote-4) As a fully convolutional network, the U-Net is able to segment images on a limited set of annotated data and retain that information as it relates to the original image.

In 2016, the United Kingdom’s Defence Science and Technology Laboratory (DSTL) created a Kaggle competition challenging participants to classify features in satellite images.[[5]](#footnote-5) The dataset contained labeling images with up to ten different features:

1. Buildings - large building, residential, non-residential, fuel storage facility, fortified building
2. Misc. Manmade structures
3. Road
4. Track - poor/dirt/cart track, footpath/trail
5. Trees - woodland, hedgerows, groups of trees, standalone trees
6. Crops - contour ploughing/cropland, grain (wheat) crops, row (potatoes, turnips) crops
7. Waterway
8. Standing water
9. Vehicle Large - large vehicle (e.g. lorry, truck, bus), logistics vehicle
10. Vehicle Small - small vehicle (car, van), motorbike

While the competition is not new, it does contain data that offers an opportunity to classify specific features in the images. The high number of different features also allows us to build models that can distinguish between similar features.

In this report, we will present results from training a U-Net on this dataset with the goal of distinguishing between five sets of features through binary classification:

1. Buildings and Misc. Manmade Structures
2. Roads and Tracks
3. Trees and Crops
4. Waterways and Standing Water
5. Large and Small Vehicles

For this project Xinyu and I split up different parts of the project. I primarily focused on loading the data, training the models for trees and crops, waterways and standing water, and large and small vehicles, and working on the report and presentation. Xinyu took the lead on building the training algorithm and visualizing the models/results. I originally set out to create the full pipeline for this project in Pytorch as well, but due to time constraints spent my time tuning and training the models that Xinyu designed in Keras.

### Description of Individual Work and Results

As mentioned earlier, I originally intended on developing a U-Net in Pytorch, but the Keras implementation worked well and didn’t run as slowly as some of my original test code which was not efficient (the full script is included in my code folder and requires a lot of GPU to run before it runs out of memory).

Example of U-Net in Pytorch:

        self.dconv\_down0 = double\_conv(3, 32)

        self.dconv\_down1 = double\_conv(32, 64)

        self.dconv\_down2 = double\_conv(64, 128)

        self.dconv\_down3 = double\_conv(128, 256)

        self.dconv\_down4 = double\_conv(256, 512)

        self.maxpool = nn.MaxPool2d(2)

        self.upsample = nn.Upsample(scale\_factor=2, mode='bilinear', align\_corners=True)

        self.dconv\_up3 = double\_conv(256 + 512, 256)

        self.dconv\_up2 = double\_conv(128 + 256, 128)

        self.dconv\_up1 = double\_conv(128 + 64, 64)

        self.dconv\_up0 = double\_conv(64 + 32, 32)

        self.conv\_last = nn.Conv2d(32, n\_class, 1)

To work on training, I modified the original code that Xinyu wrote for distinguishing buildings and structures so it would work for the three sets of features I worked on. Since the dataset was standardized during the loading process, I did not need to write too much code for this.

Next, I ran the models and modified the hyperparameters to increase performance, which resulted in the following:

|  |  |  |
| --- | --- | --- |
| **Type of Feature Detection** | **Training Jaccard Index** | **Test Jaccard Index** |
| Trees and Crops | 0.55 | 0.48 |
| Large and Small Vehicles | 0.26 | 0.11 |
| Standing and Moving Water | 0.21 | 0.04 |

*Table 1 – Summary of each U-Net’s performance for distinguishing between two similar features.*

#### Trees and Crops

The model performed reasonably well for distinguishing between trees and crops from the dataset. While training, the number of epochs were kept low to minimize overfitting

A screenshot of a cell phone

Description automatically generated

Train\_Jaccard

Train\_BCE

Val\_Jaccard

Val\_BCE

*Figure 1 – Graph of model loss at each epoch for training and validation data*

A close up of a map

Description automatically generated

Increasing BCE Indicated Overfitting

*Figure 2 – Example of overfitting indicated by increasing BCE with each epoch*

#### Large and Small Vehicles

The model did not perform very well for distinguishing between large and small vehicles from the dataset. As with other training, epochs were kept low and a larger number of augmented samples were introduced to improve performance

A close up of a map

Description automatically generated

Val\_BCE

Val\_Jaccard

Train\_Jaccard

Train\_BCE

*Figure 3 – Graph of model loss at each epoch for training and validation data*

#### Standing and Moving Water

As with large and small vehicles, the model did not perform well for distinguishing between standing and moving water from the dataset. Once again, epochs were kept low and a larger number of augmented samples were introduced to improve performance

A picture containing screenshot

Description automatically generated

Val\_Jaccard

Train\_BCE

Train\_Jaccard

Val\_BCE

*Figure 4 – Graph of model loss at each epoch for training and validation data*

Finally, I spent a good chunk of time drafting the final report and presentation for this project.

### Summary and Conclusions

This project was very exciting to work on because of the amount that we were able to learn from it. The DSTL dataset posed some unique challenges, but a multitude of existing resources and a lot of iteration helped us to achieve our goal of building U-Nets that can distinguish between similar features in an image.

There were two significant limitations while working on this project: time and the data itself.

The most impactful limitation to this project was a lack of time. Because our goal was to distinguish between two similar features, we had to build many models. The large size of the data meant it took a very long time to run these models and perform our analysis, even with the help of GPU.

A surprising challenge was building models from the training data. Each picture was massive, but there were not that many unique images. This increased the likelihood of overfitting and reduced performance because subsamples of our training data were augmented and duplicated multiple times. Some features were also remarkably similar and hard to distinguish. As noted in the discussion of results, features like vehicles and water were very hard to distinguish and additional outside information or data could have helped with this problem.

Even with these challenges, the project was a very good opportunity to explore a new type of neural network. The U-Net architecture lent itself well to new techniques such as batching the images from our dataset or implementing new types of augmentation. This project could also be a good steppingstone for performing further research on this dataset. For example, it would be interesting to build a conglomerated model could learn even more from other objects nearby. A vehicle might increase the likelihood of a road being classified and vice-versa.

This project also showcased the power of testing out neural network architectures. While our scope was limited to a Keras implementation of a U-Net its not hard to image what sort of tasks could be accomplished by branching out into a Pytorch implementation of a ResNet. Finally, the model building and training processes used in this project are not limited to this dataset. All the techniques and tricks are new skills we can use in future data science projects.

### Additional References

Additional references that were helpful for this project and not footnoted earlier are included below.

#### Data Loading and Preprocessing

Using the Kaggle API to download data: <https://gist.github.com/jayspeidell/d10b84b8d3da52df723beacc5b15cb27>

Loading the large files and mitigating errors: <http://stackoverflow.com/questions/15063936/csv-error-field-larger-than-field-limit-131072>

Help with fixing errors in .h5 files: <https://github.com/h5py/h5py/issues/441>

Process masking with polygons and cv2: <http://docs.opencv.org/3.1.0/d9/d8b/tutorial_py_contours_hierarchy.html>

#### Model Building/Training

Great end-to-end example with metrics: <https://www.kaggle.com/drn01z3/end-to-end-baseline-with-u-net-keras>

Additional example of U-Net: <https://www.kaggle.com/ceperaang/lb-0-42-ultimate-full-solution-run-on-your-hw>

### Percentage of Code Found or Copied

~1,155/1,235 total lines or 93.5% were mostly copied with no significant modifications (variable names, file locations, etc…)

1. Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597. [↑](#footnote-ref-1)
2. Ma, Lei et. al (2017). “A review of supervised object-based land-cover image classification”. *ISPRS Journal of Photogrammetry and Remote Sensing*. <https://doi.org/10.1016/j.isprsjprs.2017.06.001> [↑](#footnote-ref-2)
3. Dhillon, Anamika; Verma, Gyanendra K. (2019). “Convolutional neural network: a review of models, methodologies and applications to object detection”. *Progress in Artificial Intelligence*. <https://doi.org/10.1007/s13748-019-00203-0> [↑](#footnote-ref-3)
4. Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). [↑](#footnote-ref-4)
5. https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection [↑](#footnote-ref-5)