Determining Landable Areas for Urban Air Vehicles via Image Segmentation

Andrew Denig
Stanford University
adenig3@stanford.edu

Seraj Desai
Stanford University
serajd@stanford.edu

Abstract

The task of image segmentation of satellite imagery to determine the landability of various environments was investigated through the study of two main network architectures: U-Net and SegNet. Due to the lack of readily available datasets for this purpose, the networks were suffering from overfitting and were initially unable to accurately determine the landability of some roads and green spaces. The unbalanced dataset (skewed toward more non-landable patches) further caused issues in this regard, but the authors chose to augment the initial dataset, increasing the number of training images threefold. The U-Net architecture had more favorable runtime characteristics compared to SegNet, and dropout rates greater than or equal to 0.5 seemed to yield very promising results, as long as the model was not trained for too many epochs because overtraining would lead to the network labeling all areas as non-landable.

1. Introduction

The growing Urban Air Mobility (UAM) movement seeks to integrate aerial transport into quotidian life--drone deliveries and on-demand air taxis represent a few key initiatives. Current day aviation operations rely on dedicated infrastructures (i.e. airports, heliports) for takeoff and landing. In contrast, UAM desires to integrate takeoff and landing infrastructures directly into existing buildings in urban environments; for example, prospective skyscraper rooftops would serve as "vertiports" that allow takeoff/landing. Little progress has been made on identifying possible candidates for the "vertiports". This project seeks to use deep learning as a tool for finding possible locations for such vertiports using satellite imagery. There has been previous work in the field of image segmentation of satellite imagery (e.g. mapping disaster risk from aerial imagery[1] and identifying rooftops for solar energy adaptation[2]). However, it seems like utilizing deep learning for the purposes of finding potential vertiport locations has not been done before.

This project focuses on the area of deep learning called computer vision in order to perform image segmentation on various satellite images. Image segmentation involves labeling each pixel in an image as a certain class (in this case either landable or not landable) such that the entire satellite image is partitioned into landable zones and non-landable zones. The first step in performing this work is acquiring a dataset of satellite images from which a training set, development set, and a test set can be drawn. After searching many online resources, including AWS public repository of datasets, a publicly available dataset that was collected by Lawrence Livermore National Laboratory[3] for the purpose of detecting and counting cars was found. This dataset contains more than 300,000 images of urban and rural environments where cars are likely to be found. This seems to be conducive for our purposes, as we are seeking to find locations in urban and suburban environments to build vertiports. These images are 192 pixels by 192 pixels, with white representing landable areas and black representing non-landable areas. This resolution is helpful for training purposes because it is fine enough that humans can look at the image and distinguish the important features while being coarse enough that training on a large number of images should not be too computationally expensive.

Labeling the large amount of data that may be required to train and test a well-functioning neural network is a problem, especially given the short timeframe of this project. A deal was arranged with the Director of Marketing at Labelbox in which we get free labeling services from their workforce for the duration of the project in return for providing a testimonial about our experience with the software after the conclusion of the project. As a result of establishing this relationship, our group has access to 5,252 labelled images from which to construct our training, validation, and test datasets. Even though 10,000 or more images would be ideal, our group should be able to focus on analyzing various neural network architectures and the effects of changing a number of hyperparameters.

2. Model Characteristics

There were two types of architectures explored in this project: standard U-Net and SegNet. Both of these architectures are designed for image segmentation tasks and share a common neural network structure, compared below in Fig. 1 and Fig. 2.

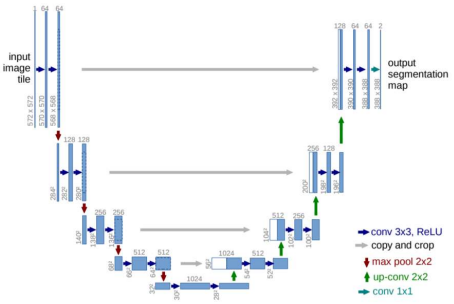


Figure 1: U-Net architecture[4].

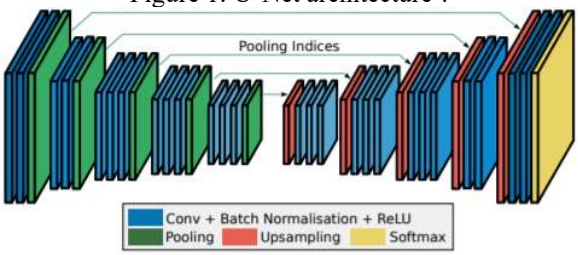


Figure 2: SegNet architecture[5].

These architectures are characterized by encoder and decoder networks. The encoder networks are characteristic of most conventional convolutional neural networks for classification problems in which the data is pooled such that the data size decreases in each layer, and number of filters becomes larger in each successive layer. However, once a bottleneck is reached, these architectures contain an analogous decoder network which upsamples the data using information from the encoder layers. This is where the two architectures differ. While a SegNet uses the max-pooling indices received from the corresponding encoder to perform non-linear upsampling of their input feature maps, the U-Net transfers the entire feature map to the corresponding decoders and concatenates them to upsampled decoder feature maps[5]. Transferring the entire feature map costs more memory than SegNet's implementation, but it gives the decoder network more information about the original image. The benefits of these networks is that they can typically achieve good performance with relatively few training examples compared to other computer vision tasks because the training data of the network is actually the number of patches present in the input labels rather than the input images as a whole. This allows each training image to essentially function as a collection of training patches from which the network can learn[4]. This justifies the use of a training set of roughly 4,500 images rather than training datasets on the order of tens or hundreds of thousands of images.

In the seminal paper introducing SegNets, the encoder network used a VGG16 architecture[5]. As a result, the structure of the encoder/decoder networks must be addressed in a little more detail. In our analysis, our group studied the speed and performance of using the VGG16 encoding architecture (SegNet) in comparison to a simpler U-Net architecture. Most notably, the standard U-Net network has fewer layers, fewer filters per convolution as well as upsampling in combination with same-padding convolution on the up-blocks rather than transposed convolutions. The simpler architecture had similar performance than the more complex VGG16 architecture, but it was able to train and fit models much quicker. To compare run-time performance, a "base" case was created for each architecture. The base case for both architectures had 16 filters in the first convolution, and doubled until reaching the bottleneck. Post-bottleneck, the number of filters halved until reaching the last sigmoid node. The following table outlines the run times for a single epoch on a personal computer for ~4,500 training images with a batch size of 16:

	Number Parameters	**Run Time/Epoch**	**10 Epoch Loss**
Base U-Net	1.9 million	100 seconds	0.395
Base SegNet	5.2 million	150 seconds	0.335

Table 1: U-Net vs. SegNet runtime.

The above table highlights that the more complex SegNet architecture takes longer but does not yield a significant improvement in the loss after 10 epochs. To reduce underfitting, it consequently appears more promising to investigate larger U-Net architectures rather than larger SegNet architectures. Further, because this network would ideally need to be quick enough to perform real time image segmentation using a drone with a camera to determine feasible landing sites, speed is critical to this network's success. As a result, the simpler architecture was adopted (and larger U-Net architectures were only adopted if underfitting was non-negligible).

3. Hyperparameter Studies

After settling on a network architecture, various hyperparameters, like dropout rate, learning rate decay, L_2 regularization and batch size, must be adjusted in hopes of improving network performance by reducing overfitting, which results in a high training set accuracy but a low validation set accuracy. Before the details of the hyperparameter study are discussed, it is important to describe the metrics our group is using to determine the quality of the network. There are two primary methods for computing validation performance for image segmentation: pixelwise-accuracy and IoU (Intersection over Union). Leading image segmentation models have an IoU around 0.7, but results of this quality should not be expected due to

the limited computational resources available for this project. However, the accuracy and IoU are severely limited by two key failure modes which are prevalent in preliminary models.

The two failure modes that were identified in our project were the neural network's behavior of marking roads as landable, despite the fact that the labeled training data is consistent with roads not being suitable landing zones, and the inability of our network to recognize rugged textures in green areas that delineate a nice, open field from a heavily forested area, both of which are shown by some examples in Fig. 3.

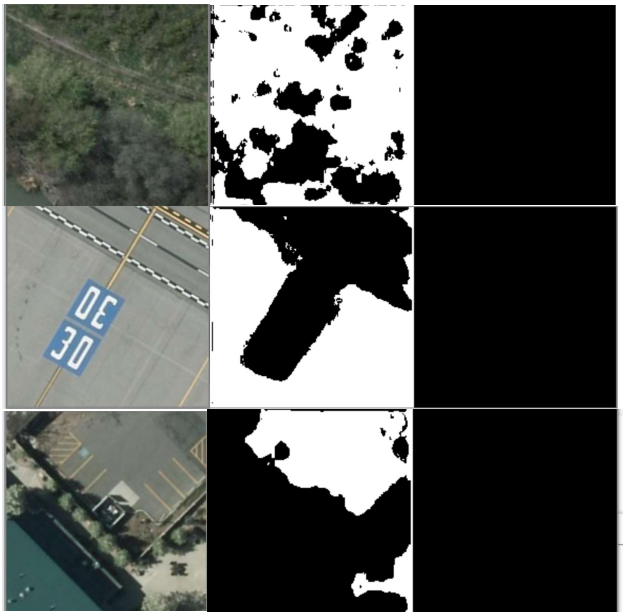


Figure 3: Examples of failure modes in preliminary models (true label: right, network prediction: center).

However, there are positives to take away from these failures, and there seems to be a solution to these issues. The ability of our neural network to identify roads accurately speaks to its ability to detect edges with high fidelity, indicating that the weights in the first few layers seem to be well-tuned for this purpose. Furthermore, our network is also able to recognize green spaces and separate them from other features. To fix these issues, the original training dataset is augmented by flipping the original images both left-right and up-down to create a dataset of 15,756 images in hopes that more training data of roads and green spaces will aid in the network's ability to correctly label them.

To methodically determine the optimal network hyperparameters for the simple U-Net architecture, a grid-search of hyperparameters was undertaken. Three separate parameters were varied: encoder dropout rate, decoder dropout rate, and number of filters. L_2 regularization was considered to reduce overfitting, but the effect of varying dropout rate was much more pronounced. Exploring these combinations allowed our group to better analyze overfitting and underfitting trends. The following table illustrates the tested hyperparameters.

Encoder Dropout Range	Decoder Dropout Range	Number Filters/Layer
0.0-0.7	0.0-0.7	[16, 32, 64, 128, 256] - [32, 64, 128. 256, 512]

Table 2: Sample hyperparameter grid search cases.

4. Results

The trends observed in the above hyperparameter search confirm expectation. Training accuracy decreases as dropout rate increases, but validation accuracy improves. This is suggestive of overtraining, even in the smaller U-Net architectures. Further interesting trends are observed when varying dropout rates between the encoder and decoder. Higher encoder dropout rates and slightly lower decoder dropout rates represents the most promising to combat overtraining. The following table illustrates a few examples for the smaller network:

Encoder Dropout	Decoder Dropout	Training Accuracy	Validation Accuracy
0.1	0.1	96%	86%
0.1	0.5	98%	88%
0.5	0.1	90%	87%

Table 3: Effects of dropout variation in encoder and decoder.

This may arise because the encoder does the bulk of "learning" by compressing and noting the salient features, whereas the decoder maps the learned features back to the appropriate areas based on the pooled indices. Therefore, low encoder dropout probabilities and high decoder dropout probabilities do not largely target the overfitting during "learning".

Overtraining can also pose serious problems. To coarsely investigate the impact of overtraining, two instances of the randomly selected architecture were trained for 70 and 150 epochs, respectively. Longer training significantly reduced network performance. While the validation loss does decrease slightly for more epochs because of the class-imbalance between non-landable and landable regions, lower validation loss does not necessarily correlate to a

better architecture. The following table illustrates the validation loss and provides a qualitative description:

Epochs	Validation Loss	Qualitative Description
70	-4500	Accurate contour detection
150	-4790	Mostly all-black

Table 4: Effects of overtraining.

Table 5 illustrates the parameters of the most promising model, and Table 6 describes some performance parameters of the final model.

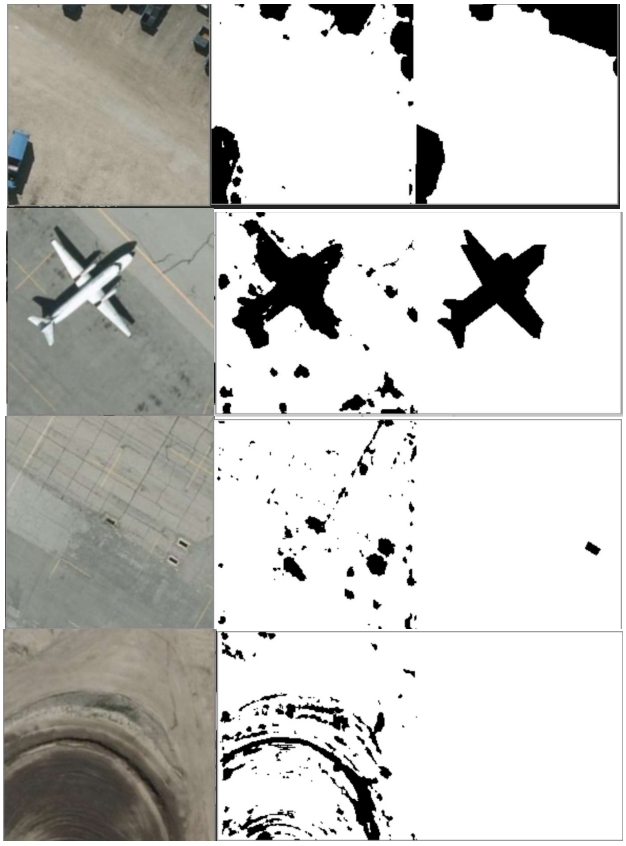
Type	Number Encoder Layers	Number Decoder Layers	Encoder Filters/Layer	Decoder Filters/Layer	Encoder Dropout Rate	Decoder Dropout Rate	Epochs
U-Net	4	4	16,32,64,128,256	256,128,64,32,16	0.7	0.5	70

Table 5: Selected model hyperparameters.

Training Accuracy	Validation Loss	Validation Accuracy	Qualitative Description
87%	-5004	82%	Follows contours well

Table 6: Performance characteristics of final model.

In addition, Fig. 4 shows 6 example images, labels, and network predictions made by the final model.



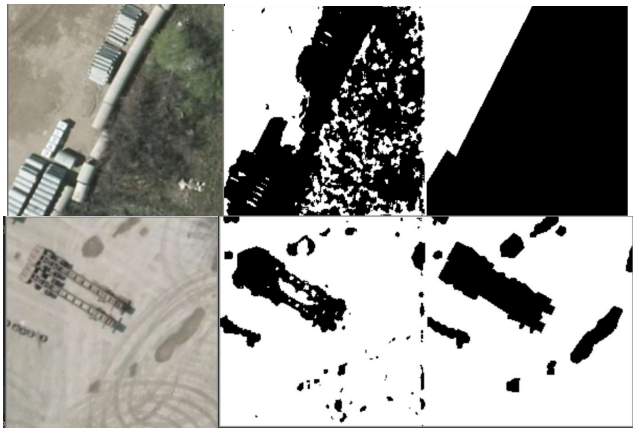


Figure 4: Six examples of predictions made by the tuned network.

There are several potential areas for improvement. Most notably, the largest obstacle stems from the lack of training data and labeling. 4,500 images (before image augmentation) is a fairly small training set. Furthermore, while dropout may have reduced overfitting, not all of the relevant features were learned even with larger architectures (more filters/layers)--this provides direct support for a larger training set. Another potential error stems from the class imbalance. Because of the relatively larger bias for non-landable regions in the training data, the network will often converge towards an "all not-landable" local minimum. A potential way to address this challenge, other than increased training data, could be to consider different loss functions, or to solely augment those images with significant landable zones. The dice loss function has been shown to handle class-imbalance challenges, but has also led to poor gradients that can slow down training. Consequently, batch normalization and potentially skip connection may be needed when implementing dice loss.

As mentioned previously, this network should be able to segment new images into landable and non-landable zones

quickly, as it will be deployed on the hardware of an urban air vehicle. This network's evaluation time per validation image on a machine with 16 GB of RAM and an NVIDIA GEFORCE GTX 1060 graphics card is 0.033 seconds, indicating that this simple U-Net architecture is fast enough to handle the demands it would experience being deployed on a real system.

5. Contribution

Andrew Denig was responsible for creating the software which managed the training, validation, and test data (automatically sorting it and performing image augmentation), and he created the main visualization tool for the project. In addition, he wrote the software which automated the generation of networks with various hyperparameters, and as a result, he was responsible for running and collecting data on most cases on AWS servers.

Seraj Desai was responsible for programming the neural network architectures, and he was the driving force behind the selection of U-Net over SegNet. He developed the software which is capable of running individual cases and showing their results, allowing the group to verify the performance of promising models.

6. References

1) "Open AI Caribbean Challenge: Mapping Disaster Risk from Aerial Imagery." *DrivenData*, MathWorks, 2020, www.drivendata.org/competitions/58/disaster-response-roof-type/page/143/.

2) Mitra, Rudradeb. "Using Image Segmentation to Identify Rooftops in Low-Resolution Satellite Images." *Medium*, Towards Data Science, 30 Jan. 2020, towardsdatascience.com/using-image-segmentation-to-identify-rooftops-in-low-resolution-satellite-images-c791975d91cc.

3) T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In ECCV, pages 785– 800, 2016.

4) O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, pages 234–241. Springer, 2015.

5) V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv:1511.00561v2 [cs.CV], 2015.