Buildings Segmentation using the U-Net Convolutional Neural Network

Sibusiso Mgidi
School of Computer Science and Applied Mathematics
University of the Witwatersrand

Thabo Rachidi
School of Computer Science and Applied Mathematics
University of the Witwatersrand

***Abstract*—The ability to be able to perform automated interpretation of aerial images has been accelerated due to progression in deep learning. Several models have proved themselves useful and in this paper we attempt to reproduce and experiment with a U-Net convolutional neural network to perform semantic segmentation on aerial images taken of Boston.**

To perform the experiment we create two separate datasets where one has augmented data and the other does not. We also downscale the dimensions of the images and use these data sets to produce 4 models which we then evaluated. During training and validation we were able to discover the hyper-parameters for the best model. We then exported the best models and deployed them to a proof of concept website which allows a user to input an image and have the models perform semantic segmentation. In this work, we achieved IoU of 0.48 on a U-Net without data augmentation and 0.47 on a U-Net with data augmentation.

Although work still needs to be done to improve the performance of the models it is evident that deep learning has its applications for semantic segmentation.

***Index Terms*—semantic segmentation, convolutional artificial network, unet, deep learning, computer vision, over-fitting**

1. Introduction

Aerial images have been used since the First World War where images from airplanes were analysed for strategic planning. This birthed the discipline of aerial image interpretation by examining aerial images to identify objects and determine the properties of the identified objects. It is has now found applications in many areas such as urban planning, crop and forest management, disaster relief, and climate modeling [1].

Early applications relied on humans to examine the images by hand but this was expensive and time consuming. The recent high availability of high resolution images has also increased the need for better automated aerial image interpretation methods. Early methods of interpretation produced object detectors with high levels of accuracy, indicating that automated aerial interpretation systems are feasible.

Machine learning applications of aerial image interpretation have been widely experimented with. This is usually formulated as a pixel labelling task, for example given an aerial image the goal would be to produce a instance segmentation of the image such as the greenery, water, building. Thus in this paper we will explore semantic segmentation may be used as step in instance segmentation. Deep learning methods have been particularly well suited to this task and so the focus of the paper is on the implementation of a U-Net convolutional neural network.

The rest of the paper is organised as follows:

- Section 2 presents any related work done on semantic segmentation and the interpretation of aerial images
- Section 3 presents the strategy for formulating the implementation of the U-Net and the experimentation of training the model
- Section 4.1 presents a discussion of the results and evaluation metrics used to analyse the performance of the model
- Section 5 provides a summary of the paper and presents a discussion for future work on improving the methods for experimentation.

2. Background and Related Work

Semantic segmentation is a computer vision technique that matches pixels in an image to a class label. It is used in cases where a thorough understanding of images is needed such as diagnosing medical conditions, navigation for self-driving cars, photo and video editing and navigation for robots. Other commonly used computer vision techniques include, but are not limited to, image classification, object detection, semantic segmentation, instance segmentation, and so on.

To know how deep learning is used to tackle semantic segmentation we need to understand that it is not just an isolated field of machine learning but a step in a larger process of object detection and It helps with the progression from a normal image to detecting just not what an object is but how many objects there are in the image. The natural progression is image classification, object localisation, semantic segmentation and instance segmentation as mentioned above.

Early work of aerial image labelling focused on ad-hoc and knowledge-based approaches [1] but this research will focus on a machine learning-based approach, specifically a deep learning approach. Machine learning has led to more recent progression when interpreting aerial images and also

in general computer vision problems such as labelling in images using semantic segmentation.

The goal of this project is to segment building footprints from aerial images by predicting the segmentation masks on each image. This computer vision approach is referred to as semantic segmentation. Furthermore, to attain the above-mentioned goal we performed the following sub-tasks:

- Implement the U-Net convolutional neural network
- Use pre-trained weights from VGG16 as encoder
- Train the U-Net model on the augmented and non-augmented images
- Export the model for website deployment

In this project, we implemented the U-Net convolutional neural network using Python, TensorFlow and Keras. Furthermore, models trained with the original and augmented dataset are then deployed on a local server for ease of use. To deploy the model, we used Flask server with JavaScript and TensforFlow Serving server.

The remainder of the paper is organized as follows, Section 2 details the pipeline to which the model will be implemented, Results of the predictive models are discussed in Section 3, And finally, Section 4.1 concludes this paper with the summary of the findings.

3. Research Methodology

The research methodology presents section 3.1 where a brief overview of the research design where the aim of the research and how we will perform experimentation to fulfil this goal. Section 3.2 presents the source of the dataset and describes the features of the dataset. In section 3.3 a discussion of the pre-processing steps taken to prepare the dataset for learning are provided. Section 3.4 and 3.5 describe the method used for transfer learning and the architecture of the U-Net model used respectively. Section 3.6 discusses how we went about training the model and any challenges we faced while training it. The last section 3.7 presents a discussion of how the trained U-Net model is deployed locally using a Flask server.

3.1. Research design

The aim of this research was to improve on early methods of aerial image labelling by building a machine learning model that is able to perform semantic segmentation on aerial images of buildings. This is qualitative research and we needed to acquire sufficient data to train a deep learning model. The dataset needed to have aerial images of buildings that would have masks as the ground truth we would use in training the model.

3.2. Source Dataset

To conduct this study, we used the Massachusetts buildings dataset from Kaggle[1]. The dataset consists of 151 aerial pre-processed images and corresponding labels of the Boston area. The images cover an area of 2.25 square kilometres, so the entire dataset covers roughly 340 square kilometres. The images are of high quality with dimensions of 1500 x 1500 pixels. The data is already split into a training set of 137 images, a testing set of 10 images and a validation set of 4 images. Each image has a corresponding mask that is used as targets. These target maps were obtained by rasterising building footprints obtained from the OpenStreetMap project. The dataset covers urban and suburban areas with buildings of all sizes.

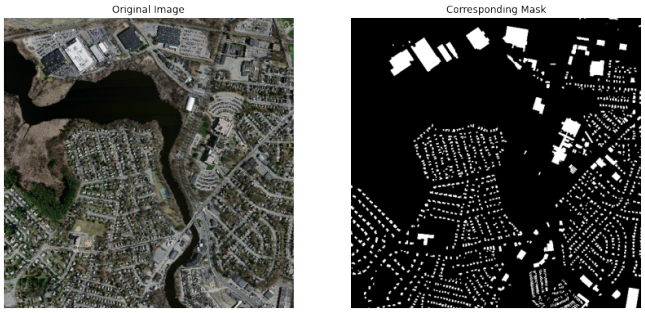


Figure 1: An example of the aerial image along with its corresponding binary label.

3.3. Data Preprocessing

To reduce computation time and memory usage we down-sampled the 1500x1500x3 pixel images together with the corresponding $1500\times1500\times1$ pixel masks to $512\times512\times3$ and $512\times512\times1$ pixels respectively. To exhaustively measure the models performance we further downsampled the images and masks to $256\times256\times3$ and $256\times256\times1$ respectively. The images and their corresponding masks pixels' where also normalised from the range $[0, 256]$ to $[0, 1]$.

3.3.1. Dataset split and Data Augmentation.

Deep learning models are known to be data hungry, therefore, given the size of our dataset we applied data augmentation as a pre-processing step when training the model. Data augmentation is a technique use mostly in machine learning and deep learning to discover knowledge scenarios from the dataset [2]. This technique is also used to avoid over-fittings as described in [3]. To help increasing the training data we performed data augmentation of the images by flipping the images and their corresponding masks horizontally and vertically. This increased the number of training images to 524 and validation images to 40 images.

We reduced the training set to 131 images and transferred them to the validation and test set. This work out to 131 training images, 10 validation and 10 test images and corresponding masks. This was done to consistently monitor the generalisation performance during training

1. https://www.kaggle.com/balraj98/massachusetts-buildings-dataset

3.4. Transfer Learning

Transfer learning is a way of getting knowledge that has been learned on different types of large dataset [4] such as Imagenet [5], The Microsoft Common Objects in Context (MS COCO) [6], MNIST [7], and so forth. The knowledge learned from these large datasets is then used migrated into deep learning models during trained to increase models performance. Transferring knowledge from one model to another is mostly used when the dataset is limited [8].

3.5. U-Net Architecture

The U-Net is a semantic segmentation deep learning model proposed by [9]. This architecture has a significant influence in the field of biomedical images. The U-Net model improves on the Fully Convolutional Network (FCN) [10], which learns to segment objects in an image from left to right side. It accepts any size image as input and generates a segmentation map using efficient inference and learning. The architecture has a U-shape that uses a encoder-decoder scheme. It consists of three sections a down-sampling, bottleneck and up-sampling section.

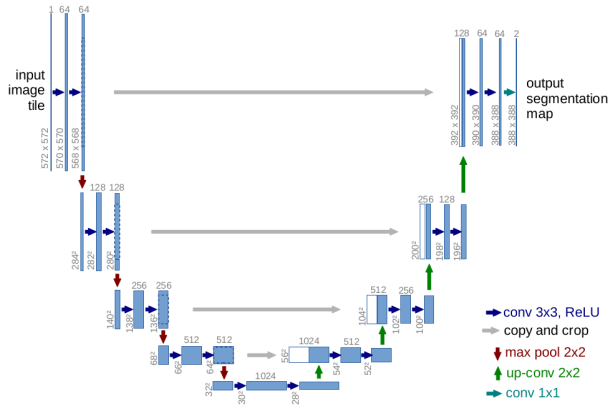


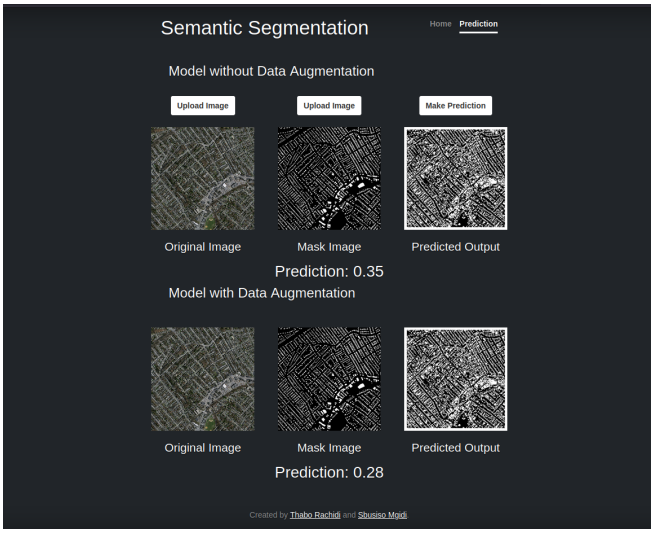
Figure 2: Illustration of U-Net architecture [9]

The encoder route consists of 3 x 3 unpadded convolutional layers with ReLU activation, followed by downsampling input pictures using max-pooling (2 x 2 kernel with 2 strides). In order to create an output activation map, the network performs max pooling on each channel of the input activation map independently. Using 2 x 2 transposed convolutions, the encoder layer's feature maps are enlarged in the decoder layer. The feature maps are then concatenated with decoder feature maps that have been upsampled to create a re-scaled high resolution segmentation map and a class for each pixel.

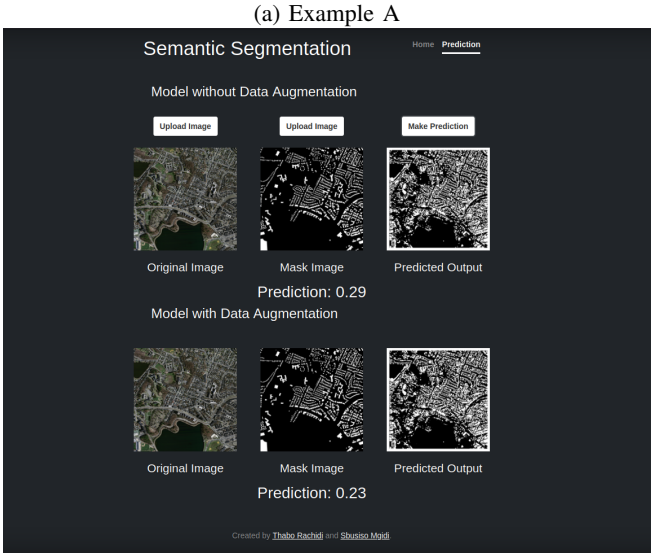
3.6. Model Deployment

To deploy the model we needed to abstract away the model details and just deploy an easy to use API endpoint. This would just be an URL endpoint which anyone can make a POST request and retrieve a JSON response of the prediction(s) made by the model.

As a proof of concept we created a TensorFlow Serving server to deploy the UNet model on our local machines. We fist exported our Keras model and to ensure that we have a model that TensorFlow Server could handle we exported it the SavedModel format. We could then start a TensorFlow Serving server on our local machines that can respond to POST requests. We then built a Flask server on top of the TensorFlow Serving server, this just makes things easier for image processing as we can use Python libraries with Flask such as NumPy, OpenCV and Keras.



(a) Example A



(b) Example B

Figure 3: Examples of the website developed with Flask

Ideally the TensorFlow Serving server and Flask server should be running on separate machines as high number of

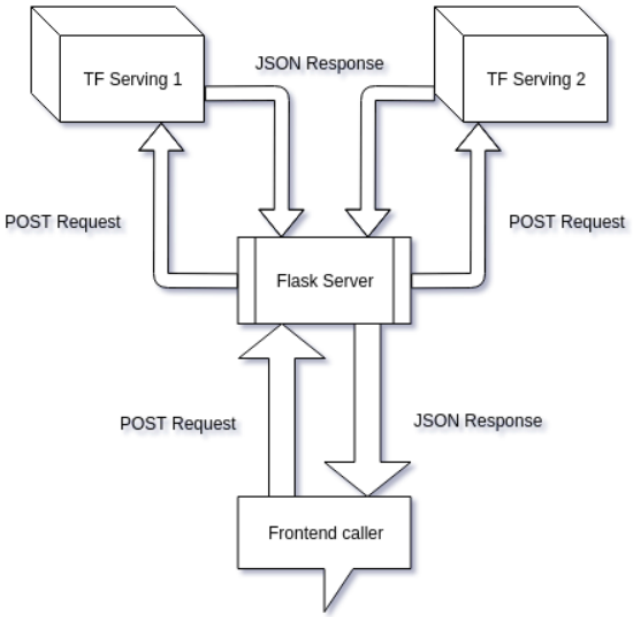


Figure 4: Deployment Architecture [11]

requests could slow down the Flask server because of the image processing being carried out. A single Flask server instance may also not be enough if there is a high number of requests. A queuing system would help alleviate some of the processing if we have multiple frontend callers. Since this is only of proof of concept the method described above is sufficient.

Figure 4 displays the Deployment Architecture we used as we have two models of the UNet to compare; one with augmented data and the other without augmented data.

3.7. Evaluation Metrics

To ensure that our model was performing correctly we need ways to monitor its prediction performance. The evaluation metrics we used to monitor the performance of the model were training and validation loss, accuracy and intersection over union.

Training and Validation loss.

To ensure that our model does not underfit or overfit on the training data we had to monitor the training and validation loss of the model. Underfitting occurs when the training loss is greater than the validation loss while overfitting occurs when the training loss is less than the validation loss this means the model is fitting to the training data and is not able to generalise. The aim is to have both losses equally low and to make sure they both converge over time.

Accuracy.

Accuracy is calculated as the number of true positive predictions and true negative predictions divided by the total number of data points we used for prediction. It gives us a basic indication of what fraction out of all predictions did we get right. Accuracy has its drawbacks because a model is trained using a biased dataset and evaluated using accuracy it can give a false sense of high accuracy.

Intersection over union.

Intersection over union is popular metric used in evaluation of image segmentation models. It is used to quantify the percent overlap between the mask images, which is the target, and the prediction output. It measures the number of common pixels the target and prediction masks divided by the total number of pixels present across both masks.

4. Results and Discussion

The results and discussion section presents Section 4.1 where evaluate results. Section 4.2 presents the results we achieved after training the model and adjusting hyper-parameters.

4.1. Experimental Results

In this work, to train the U-Net deep learning model we carried out this tasks on Google Colab Tesla T4 instance with 15 GB GPU of memory. Due to insufficient GPU memory on our local machines Google Colab was the best option to train and evaluate the model. Figure 5 indicates the best hyper-parameters for our model:

Batch size	6
Epochs	120
Early stoppers	30
Learning rate	0.001
Dropout rate	0.5
Optimizer	Adam
Loss Function	Binary Cross Entropy

Figure 5: Hyper-parameters for Best Model

For training we used pre-trained weights from VGG16. This was achieved by replacing the constracting path of the network with VGG16 architecture and then fine-tuned the network using the aerial training images and masks. This is also know as transfer learning as mentioned in 3.4. It took less than 3 hours to train the model. The model was trained with and without data augmentation to confirm models performance. Furthermore, we also downsampled the images and masks dimensions from 1500×1500, 512×512 and 256×256 to obtained increased speed. The findings after performing several experiments are summarized in Figure 6 below:

Models	Image Size	Average IoU	Validation Loss
U-Net without augmentation	256 x 256	0.33	0.192
	512 x 512	0.48	0.152
U-Net with augmentation	256 x 256	0.38	0.170
	512 x 512	0.47	0.159

Figure 6: Summary of best Models along with Image Dimensions

To achieve the results above, the intersection over union was calculated for each image and then averaged across 10 test images. it is evident that increasing the dimensions of the images has an impact on how the model performs.

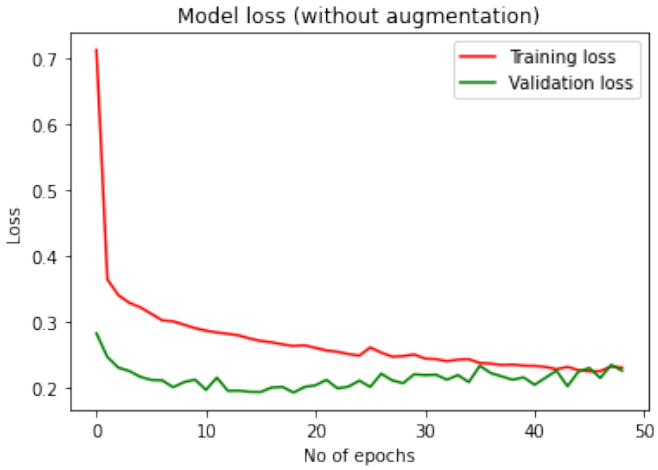


Figure 7: Training vs Validation Loss

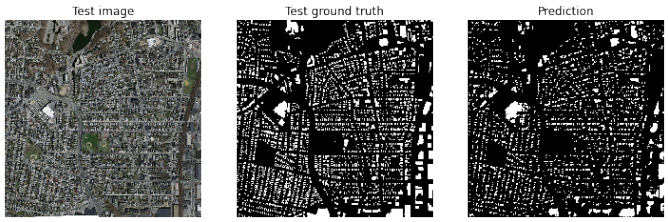


Figure 8: An example of test image, ground truth and predicted output of the U-Net without augmentation: IoU = 0.39

4.2. Discussion

The batch size of 6 was used when training the model over 120 epochs. We used a learning rate of 0.001 and a dropout rate of 0.5. Thea dam optimiser as the optimising algorithm. It is a extension of stochastic gradient descent and has seen a rise in popularity for deep learning applications in computer vision. It combines the advantages of two other extensions of stochastic gradient descent, specifically adaptive gradient algorithm and root mean square propagation. For the loss function we used a binary cross entropy.
Each experiment was run with Adam optimizer, and we discovered that a learning rate of 0.001 produces good results

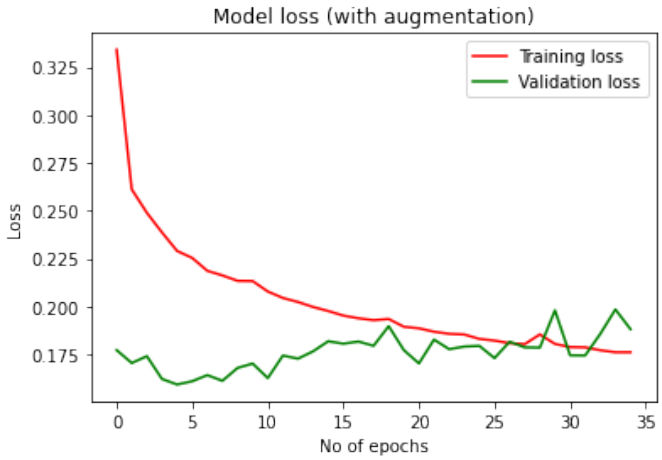


Figure 9: Training vs Validation Loss

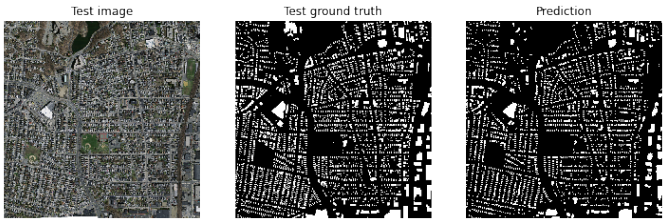


Figure 10: An example of test image, ground truth and predicted output of the U-Net with augmentation: IoU = 0.51

for the deep learning model. We also tried out the stochastic gradient descent optimizer(SGD), but it did not produce encouraging results, therefore we used the Adam optimizer for both U-Net with and without data augmentation.
It is evident from above Figures 8 and 10 that a model trained with data augmentation yields better results. Also Both models were trained with the over 120 epochs however, it worth noting that the model without data augmentation start over-fitting at epoch 27 therefore, using early stoppers proves to be efficient and saves computational resources as GPU in Google Colab it is only available for 12 hours. The

5. Conclusions and Future Work

The aim of the research was to improve on early methods of aerial image labelling by building a machine learning model that is able to perform semantic segmentation. The U-Net models produce where able to segment the images but did not have a high intersection over union score. The models that perform the best where the models that were only downsampled to $512\times512\times3$. The models that trained on the $256\times256\times3$ images performed poorly and this is due to the size of the pixels needed to do segmentation being greatly reduced.

To extend this research we could look at employing different architectures for comparison, such as the

Fully Convolutional Network (FCN), DeeplabV3, Residual Network(ResNet) and Semantic pixel-wise segmentation(SegNet) . We could also use data acquired from different areas so that the model is able to generalise. The biggest bottleneck experience during training was memory space available to use to train the models. Training on a machine with more GPU memory would allow us to train the model with images of higher dimensions.

References

[1] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[2] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[3] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8093–8104.

[4] W. Zhao, "Research on the deep learning of the small sample data based on transfer learning," in *AIP Conference Proceedings*, vol. 1864, no. 1. AIP Publishing LLC, 2017, p. 020018.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[7] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[8] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[11] H. Rawlani. Deploying keras models using tensorflow serving and flask. [Online]. Available: https://towardsdatascience.com/deploying-keras-models-using-tensorflow-serving-and-flask-508ba00f1037