Using Image Segmentation to Measure Horizontal Urban Spread.

Project: Using machine learning to analyze satellite data.

Johnny Lowis[a], Supervisor: Dr. Sha Luo[b]

[a] *Department of Physics, University of Canterbury,*
[b] *Department of Physics, University of Canterbury,*

Abstract

Segmentation and land cover classification has been implemented through the use of Convolutional Neural Networks(CNN's)[1][2]. Using a binary approach to identify buildings could be used to gauge the rate of urbanization, by applying the CNN model over a range of years of satellite data for a given area, to identify buildings. The chosen CNN was the UNET model [3], using the keras [4] python framework. Pre-processing, the act of preparing data to be interpretable by the CNN took the majority of the project run-time. Particularly, converting geospatial data from Land Information New Zealand to suitable formats and appropriately tiling large images. The CNN used was a basic implementation of the UNET structure. Computational power limited the training of the CNN on the entire processed dataset. As such, we analysed the UNET model trained on 30 epochs, with a batch size of 16 (Hence, 480 images or 21% of the prepared input data). In particular, the model was trained on the city of Rolleston, due to data being readily available. The output was not accurate enough to be used as a reliable tool for counting the numbers of buildings in an image, yielding a True Positive value of 15%. Changes to the model architecture to improve training efficiency, by reducing the time taken to train an epoch (in the analysed model, the time per epoch was approximately 20 minutes) as well as improve model accuracy through model design changes were discussed. The UNET implementation functioned as a proof of concept, demonstrating the validity of the proposed use as a tool for measuring urban expansion, given the appropriate development and computational resources.

1. Introduction

Urbanisation has a negative impact on local wildlife both globally and in New-Zealand [5]. Measuring urbanisation is either done statistically [6] by sampling a population of the larger city, or through laborious analysis through human inspection. The use of image segmentation in Convolutional Neural Networks (hereafter CNN's) allows the use of Machine Learning to identify features in data. In particular, the use of UNET [3] a semantic image segmentation model, allows us to automate the identification of buildings in both urban and rural environments.[7] This tool is of use to city-planners, governmental and healthcare agencies as city living introduces higher risk to infectious diseases, poverty and health hazards.[8][9] Land Information New-Zealand (hereafter LINZ) has produced an "NZ building outlines" data-set [10] which is a shape file (a geospatial data format containing polygons in some map projection), of the outlines of most buildings in New-Zealand, from the most recent satellite imagery taken by LINZ.

This data set was highly useful in constructing a training mask for the CNN, but it only provides outlines for the most recent satellite images, therefore not providing a measure of urbanisation for a given area. By training the CNN on the most recent satellite imagery and the "NZ building outlines" data-set, the trained CNN can be applied to historic satellite images and urban growth can be quantified. By having a method for obtaining a count of the total number of buildings in a satellite image, this method can, in theory, be applied anywhere in the world where satellite data of the appropriate resolution (0.03m) is available.

The use of this tool can highlight areas that current metrics of urban expansion (i.e. statistical analysis) have overlooked, by explicitly counting the number of buildings between certain time frames. So far, the tool has not yet been developed fully, due to time and computational limits, though further development in this area could lead to urban growth projections through regression analysis. This information would be highly useful for civil agencies to mitigate the hazards of under-prepared infrastructure for expanding urban areas.

2. Method

2.1. Overview and methodology

When implementing a machine learning (hereafter ML) approach to data processing, it is essential to note whether ML is appropriate for the desired outcome. Generally, there should be a lot of data available. For example, Lapeyre et al [11] trained UNET on a grid of over 33.6 million cells. Comparably, since the ML model trained on 512x512 pixel size, on 480 images, our implementation trained on 125,829,120 cells, or 125 million cells (from 512 times 512 times 480). Hence, the database required needed to be suitable for training the model. For our implementation, LINZ had high resolution data as well as good temporal coverage for the later stages of the model. In the later stages it was planned to use the model on the same area over different years, hence requiring satellite imagery over several years. Furthermore, there had to be some way to 'show' the model what parts of the images the user is interested in and by extension wants the model to identify. Thus, the LINZ NZ buildings database was good for this task as it contained the building outlines we wanted the model to identify. Given we had the data available and a personal computer powerful enough to produce a proof of concept implementation, the pre-requisites were fulfilled.

2.2. Data preparation

The UNET model requires, for each image in 512x512 pixel size (height x width), two inputs. First, the image itself, in our implementation, a .jpeg satellite image file with only RGB (Red Green Blue) colour bands, as well as a binary mask image. Here, 'mask' refers to an image, of the same dimensions as the satellite image, that has only two colours. In our use, it had white for buildings and black for background. This is used to 'show' the CNN what the user wants the model to recognise. This is known as 'training' in ML, where the CNN is taught to identify the masked shapes in the data, through comparing its results with a split of the data set. This dataset is split into two subsets: 'test' and 'train', where the test subset is randomly selected from the data, to verify the model's object detection. For the UNET model (hereafter the model), the architecture is shown in Figure 1[3], where UNET has the benefit of the input and output dimensions being the same.

Input data was obtained from LINZ, in the form of high resolution (0.03m) satellite imagery and their "NZ building outlines" data-set [10]. The latter was used to create a mask for the model. The pre-processing pipeline is shown

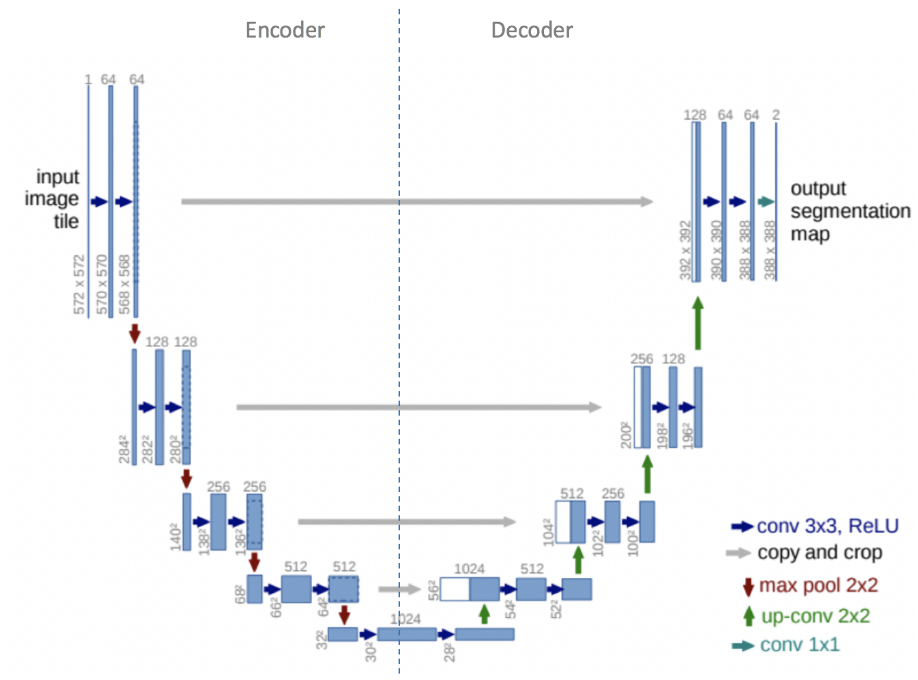


Figure 1: UNET architecture.[3]

in Figure 2. Following the stages shown at the top of the figure, in stage A, the two data-sets were overlaid in the LINZ database and cropped over the area of Rolleston. This ensured that the two data-sets would have the same dimensions. Here, the building outlines data-set was also processed to have the building polygons filled with white and the background as black. In stage B, the two data-sets were plotted as overlapping, in order to ensure they have the same projection, shown in the code appendix in the `mask_maker.ipynb` script. Here, the rasterio[12] and earthpy[13] python modules were used to ensure the data was being rendered in the correct projection. If these packages were not used, the data would be warped in the visualisation. Further on in stage C, the overlapping image was split into tiles, as per the input requirements of the UNET model stated previously. This stage was where overlapping the two data-sets in B proved to be essential, because if the locations of the buildings in the satellite image did not match up with the locations in the mask, the model would be trained incorrectly.

In stage D, the tiled satellite data had to be converted from geotiff (.tif) format to jpeg (.jpeg) format and the tiled building outlines files had to

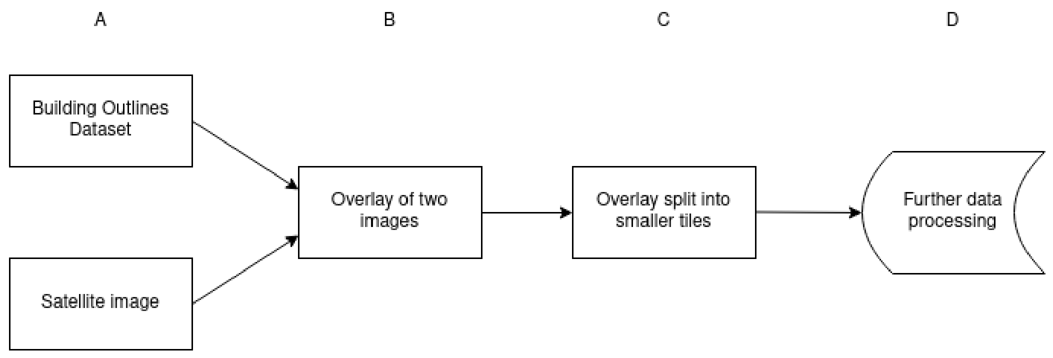


Figure 2: Pre-processing workflow

be converted from shapefiles (.shp) to jpeg (.jpeg) as well. This was done using the GDAL library [14] in the `file_conversion.ipynb` script. This processing pipeline then changed the data from an initial input shown in a screenshot from the LINZ database retrieval page in Figure 3, to the tiled .jpeg files in Figure 5.

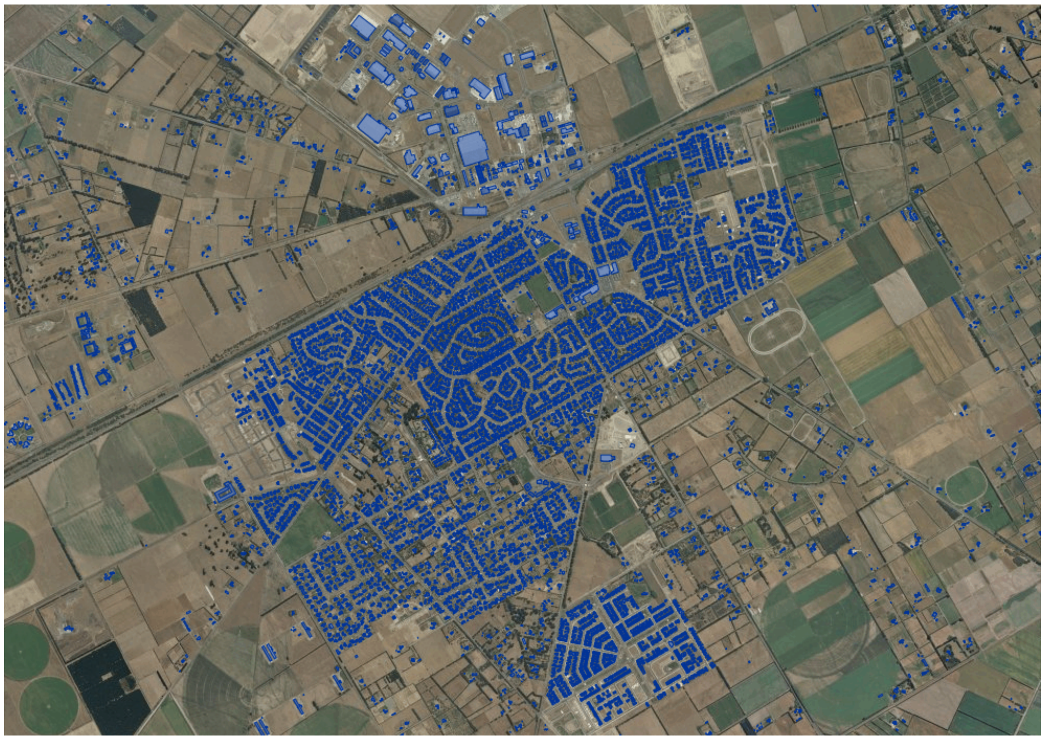
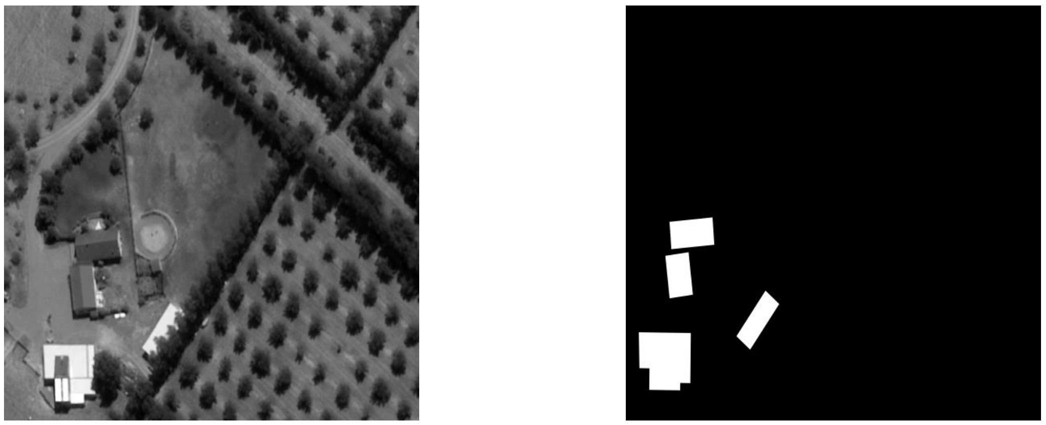


Figure 3: Satellite image of Rolleston overlapped with the building outlines data-set.



(a) Processed satellite image

(b) Processed mask

Figure 4: Processed tiles

2.3. Model implementation

Following the structure of the UNET model, many different frameworks for building a neural network architecture are available. For simplicity as well as rough optimisation of model training time by using batch normalisation, discussed as being more efficient for training CNN's [15], the keras library[4] was chosen. The final implementation is shown in the code appendix as `UNET.ipynb` where the original code skeleton was retrieved from a kaggle competition [16]. This was modified according to the project outline, particularly the file retrieval, data input pipeline and training parameters. Due to hardware limitations, the first version of the model training, could only train on 30 epochs, with a batch size of 16 (Hence, 480 images or 21% of the prepared input data). This meant that the model was not sufficiently trained for accurate outline matching, but was identifying buildings successfully. The use of the keras module allowed for easier implementations of the UNET structure through the use of the layers class. This allowed us to build the appropriate structure, shown in Figure 1, and choose parameters such as image size, number of convolutions, what convolving kernel to use, and what activation function to use. Furthermore, upon compiling the model, different optimizers could be chosen for gradient descent. For simplicity, in our implementation we used the Adam optimizer [17], which is natively supported by the Keras framework. Below is an example of the first downwards convolution, shown in the Encoder section of the UNET framework in Figure 1:

```
c1 = Conv2D(16, (3, 3), activation='elu',
            kernel_initializer='he_normal', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='elu',
            kernel_initializer='he_normal', padding='same') (c1)
```

Here 'c1' refers to the first convolution, where the Conv2D convolution was used. In the future development and improvement of the model, it is at this stage where changes to its architecture can be made, for example testing a different kernel or activation function.

3. Results and Discussion

3.1. Model training and output

The training of the model averaged around 20 minutes per epoch. Thus, for 30 epochs it took approximately 10 hours to train. This was using a batch size of 16 images. We then estimated the time needed to fully train the model:

We divide the total number of training images by the batch size, to get the number of epochs:
$2250/16 = 140.625$
Then we multiply the number of epochs by the training time of 20 minutes:
$140.625 * 20 = 2812.5$
And convert the result to hours:
$2812.5/60 = 46.875$

This was assuming that the training speed per epoch would not increase over the training duration as well as no temperature throttling, or hardware deceleration due to high temperatures, would occur. If there were alternative hardware available, the training time could be reduced and the output and changes to model architecture could be explored more fully. However, this not being the case, we in stead will examine the predictions of the briefly trained model and propose changes that would likely benefit future model training.

In Figure 5, we see an example of the predictions from the trained model. From left to right, column-wise, are the satellite image, the mask of the image, the raw prediction by the model and the prediction with a threshold applied. Here, 'threshold' means that only pixels above a certain probability, in this case where the pixel value was greater than or equal to 0.31 was used. This value was chosen by inspection. The probability referred to here, is the likelihood of a pixel being part of a building.

3.2. Analysing and interpreting output

Commonly, a confusion matrix is used to compare the results of a prediction with the ground truth for the application. This gives us a measure of the accuracy of a model's predictions. This was plotted using the scikit learn python module's implementation [18] in Figure 6.

Interpreting a confusion matrix is done by examining the axes labels.

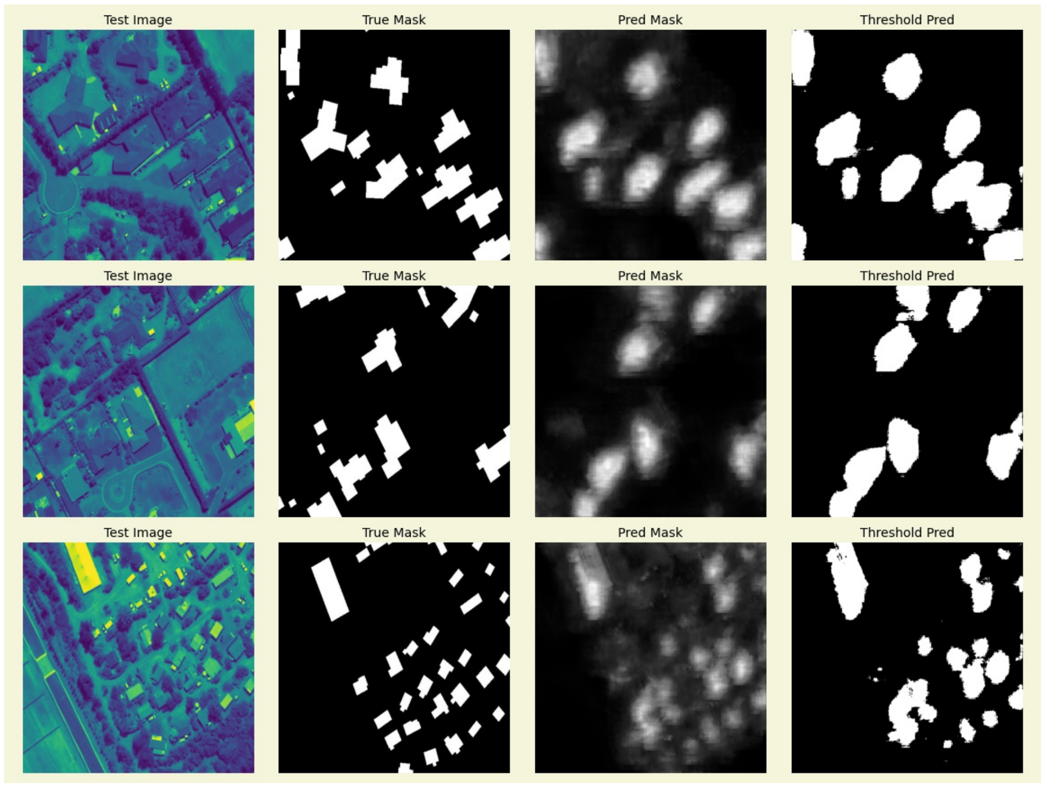


Figure 5: Urban building area predicted by model.

- The 1st row is ground truth that the pixel is not part of a building.
- The 2nd row is ground truth that the pixel is part of a building.
- The 1st column is the model's prediction that the pixel is not part of a building.
- The 2nd column is the model's prediction that the pixel is part of a building.

Here the confusion matrix has been normalised to a percentage as a decimal for each square. Given the above categories, we can identify the 'correct' predictions as True Negative and True Positive as follows:

- Coefficient (0, 0) is the True Negative count (TN).
- Coefficient (0, 1) is the False Positive count (FP).

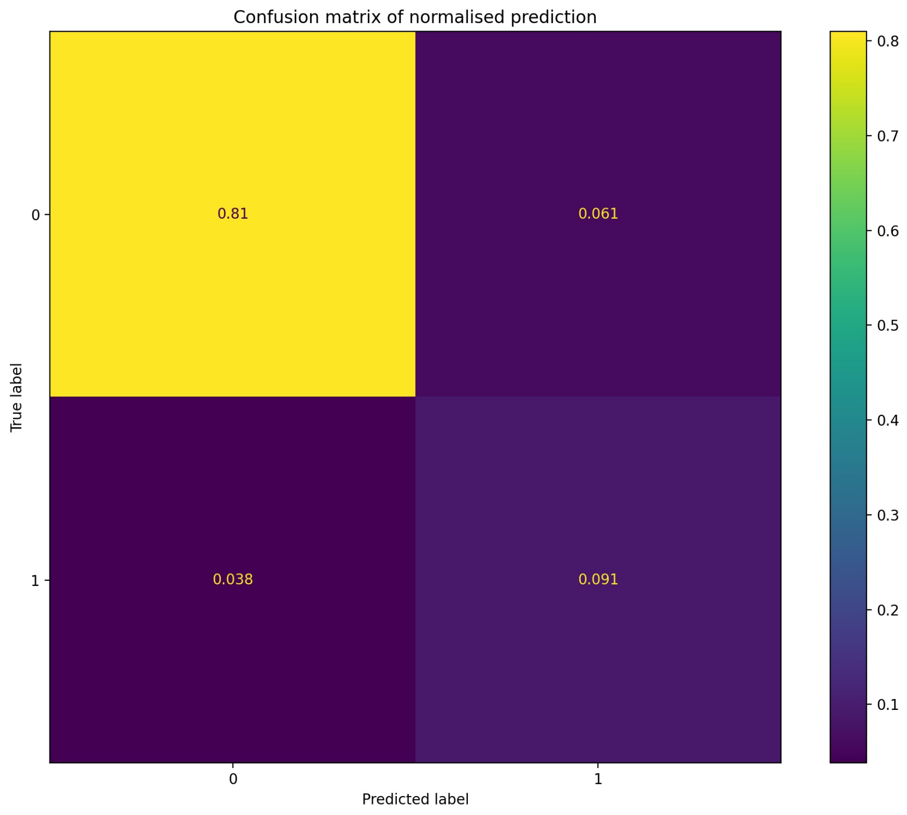


Figure 6: Confusion matrix of predicted vs. true values.

- Coefficient (1, 0) is the False Negative count (FN).
- Coefficient (1, 1) is the True Positive count (TP).

Another accuracy metric from the keras library, called binary accuracy was used to examine the model output, however this led to a poor visualisation and was not investigated in detail further. The sparse output is likely because only TP values were plotted. It is however attached in the appendix alongside some selected examples of rural and urban model output.

3.3. Proposed model improvements

As can be seen from the TN and TP cells in Figure 6, the model was correctly classifying a pixel as a building 70.54% of the time, and correctly classifying a pixel as not a building 81% of the time. The True Positive accuracy of 70.54% was calculated as follows:

```
0.038 + 0.091 = 0.129
0.091 ÷ 0.129 = 0.7054 = 70.54
```

Furthermore, the model had an overall accuracy of 90%, computed with:

```
Accuracy = (TP + TN) / (P + N) = 0.9010 = 90.1%
```

These are satisfying results considering the small amount of training.

The results could be improved by investigating why the model was performing poorer in urban areas, such as row 3 in Figure 5, where a high building density led to more miss-classification. This could be due to the small training size of 480 images, and since the batches were randomly retrieved from the data, it is possible that rural image tiles were sampled more often than urban tiles.

To analyse this, the overlaid image in section B of Figure 2, shown in Figure 3, could be tiled, and the tiles could be split into two sub-directories; one for urban tiles, and one for rural tiles, based off of how many buildings were in a tile, i.e. if more than 10 buildings were in a tile it would be classified as urban. However, this was not possible to implement, since this would need to be done by loading both data-sets into memory at once, which was much larger than the available memory of around 16GB of the personal computer used for this project. Figure 3 was captured via screenshot of the LINZ database, hence why this visualisation was possible.

In the "Pred Mask" column in Figure 5, the model is outputting its predictions in a smooth manner. By smooth it is meant that rather than a discrete prediction for a line of values, the model predicts a 'ramp-up' of higher probabilities as it tends to pixels that it is more certain are buildings. Whether this is due to a lack of training or due to the model's architecture is uncertain.

If it is due to lack of training, re-training the model on more epochs with smaller batch sizes should yield stronger predictions, as the model is more likely to have a greater distinction between what pixel is and is not a building. If it is due to the model's architecture, using a different convolving kernel, such as an atrous convolution, which is known as a 'sparse' kernel may lead to better predictions [7]. A further change could be to use an adapted model architecture focusing on feature extraction, similar to the model used by Stoian et. al [2].

4. Conclusion

The novel UNET architecture is shown to be capable of identifying buildings in high-resolution satellite imagery to a high degree of accuracy with little training. Data processing of readily available geospatial data-sets let users implement Machine Learning models capable of complex feature identification, useful as a metric for urban development.

Change detection in such datasets has already been proven and implemented [1]. In this more specific approach, focusing on building detection can be improved via proposed changes to the model's architecture and training to achieve results similar to existing studies. [1][2] Due to time and computational resource constraints, the development of the UNET model for this goal did not progress to a fully functional tool, however the further development and improvement of the current tool could prove to be useful for preventing the hazards of city dwelling, amplified by poor infrastructure due to rapid urbanization.[8][9] Specifically applied to New-Zealand, the tool could give a physical count of the number of buildings for each year and area it is applied to. This could then be used alongside current statistical analysis [19].

5. Appendix

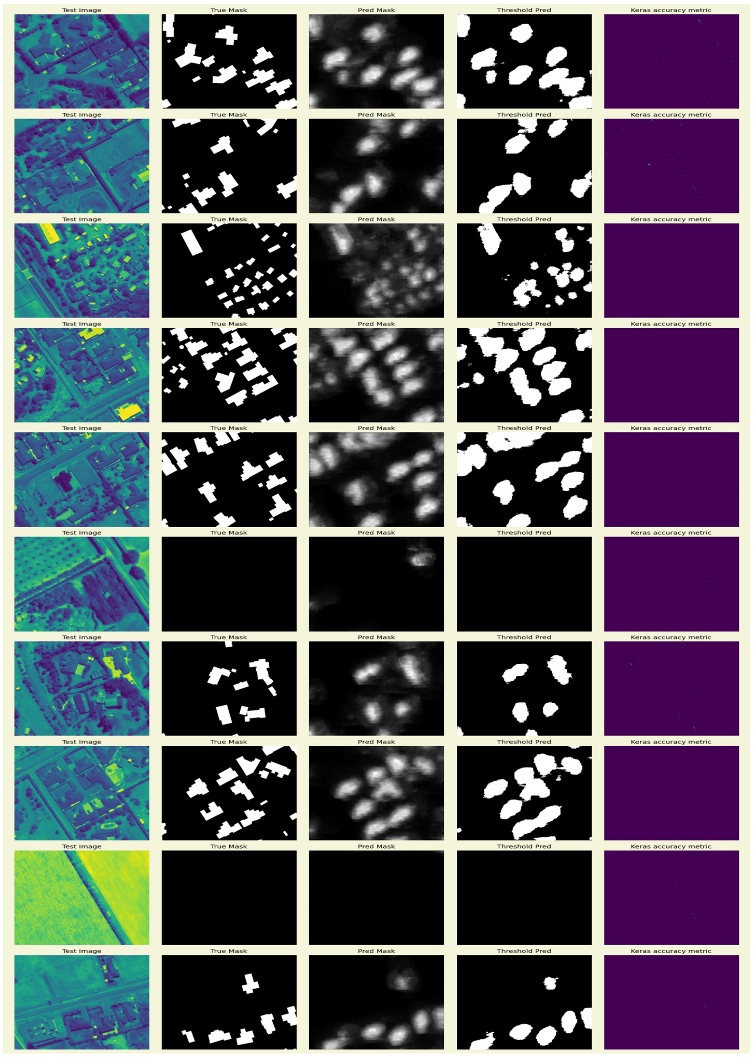


Figure 7: Further selected examples of model output.

References

[1] D. Peng, Y. Zhang, H. Guan, End-to-end change detection for high resolution satellite images using improved unet++, Remote Sensing 11 (11) (2019). doi:10.3390/rs11111382.
URL `https://www.mdpi.com/2072-4292/11/11/1382`

[2] A. Stoian, V. Poulain, J. Inglada, V. Poughon, D. Derksen, Land cover maps production with high resolution satellite image time series and convolutional neural networks: Adaptations and limits for operational systems, Remote Sensing 11 (17) (2019). doi:10.3390/rs11171986.
URL `https://www.mdpi.com/2072-4292/11/17/1986`

[3] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation (2015). arXiv:1505.04597.

[4] F. Chollet, et al., Keras, `https://keras.io` (2015).

[5] N. S. W. E. Boubée J., Jowett I., Fish passage at culverts – a review with possible solutions for new zealand indigenous species.

[6] J. P. Gibbs, Measures of Urbanization*, Social Forces 45 (2) (1966) 170–177. arXiv:https://academic.oup.com/sf/article-pdf/45/2/170/6506501/45-2-170.pdf, doi:10.2307/2574387.
URL `https://doi.org/10.2307/2574387`

[7] P. Zhang, Y. Ke, Z. Zhang, M. Wang, P. Li, S. Zhang, Urban land use and land cover classification using novel deep learning models based on high spatial resolution satellite imagery, Sensors 18 (11) (2018). doi:10.3390/s18113717.
URL `https://www.mdpi.com/1424-8220/18/11/3717`

[8] M. D. e. a. De Castro MC, Yamagata Y, Integrated Urban Malaria Control: A Case Study in Dar Es Salaam, Tanzania., American Journal of Tropical Medicine and Hygiene. 71 (2) (8 2004).
URL `https://www.ncbi.nlm.nih.gov/books/NBK3748/?report=classic`

[9] M. A. J., The urban environment and health in a world of increasing globalization: issues for developing countries., ABulletin of the World Health Organization. 78 (9) (2000) 1117–1126.
URL `https://www.ncbi.nlm.nih.gov/books/NBK3748/?report=classic`

[10] L. I. New-Zealand, Nz building outlines (2021).
URL `https://nz-buildings.readthedocs.io/en/latest/introduction.html`

[11] C. J. Lapeyre, A. Misdariis, N. Cazard, D. Veynante, T. Poinsot, Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates, Combustion and Flame 203 (2019) 255–264. doi:10.1016/j.combustflame.2019.02.019.
URL `http://dx.doi.org/10.1016/j.combustflame.2019.02.019`

[12] Mapbox (2021). [link].
URL `https://rasterio.readthedocs.io/en/latest/`

[13] E. Lab (2021). [link].
URL `https://earthpy.readthedocs.io/en/latest/`

[14] E. R. Frank Warmerdam (2021). [link].
URL `https://gdal.org/api/python.html`

[15] X.-Y. Zhou, G.-Z. Yang, Normalization in training u-net for 2d biomedical semantic segmentation, IEEE Robotics and Automation Letters PP (2019) 1–1. doi:10.1109/LRA.2019.2896518.

[16] D. S. Bowl (2018). [link].
URL `https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277`

[17] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). arXiv:1412.6980.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[19] stats.govt.nz, Subnational population estimates: At 30 june 2020.
URL `https://www.stats.govt.nz/information-releases/subnational-population-est`