

**Investigating the capability of UAV imagery for AI-assisted
mapping of Refugee Camps in East Africa**

CHAN, Yan-Chak Christopher

Supervised by:

Prof. Dr. Hannes Taubenöck ¹

Matthias Weigand ²

Emran Alchikh Alnajar ³

Masterarbeit submitted for the degree of
Master der Naturwissenschaften (MSc.)

in

Applied Earth Observation and Geoanalysis of the Living Environment
(EAGLE)



Philosophische Fakultät (Historische, Philologische, Kultur- und
Geographische Wissenschaften)
Julius-Maximilians-Universität Würzburg

Forewords and Acknowledgements

Declaration of Independent Work

Figure list

Abbreviations

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur dignissim, quam maximus posuere cursus, magna justo rutrum erat, at mattis magna magna nec risus. Duis lacus lectus, condimentum a viverra eu, fermentum molestie lorem. Sed maximus, enim eu scelerisque dictum, sem erat mollis massa, in dictum ante libero a tortor. Cras nulla nisi, sollicitudin ac suscipit cursus, maximus non dui. Sed venenatis ligula id efficitur imperdiet. Vivamus ut magna eleifend, rutrum ante facilisis, pulvinar turpis. Maecenas at interdum lorem. Duis vel varius ligula. Sed magna erat, egestas vitae varius id, cursus vitae neque. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Phasellus interdum lectus mi, a dapibus lorem tristique a. Phasellus molestie vestibulum metus a fringilla. Pellentesque at rhoncus nulla. Praesent posuere turpis nec leo fringilla egestas.

Pellentesque auctor vel dolor eu viverra. Ut faucibus nunc orci, eu aliquam justo hendrerit vel. Proin auctor sed nisl non posuere. Vivamus orci orci, commodo eget semper nec, tempus at arcu. Nam eget leo cursus velit aliquam varius. Curabitur nisi dui, rutrum vitae elit a, mollis volutpat mauris. Suspendisse potenti. Nam convallis magna iaculis posuere aliquam. Quisque tristique rutrum placerat. Quisque ultricies molestie lacinia. Maecenas at nisi in neque dictum consequat.

Contents

0.1	Forewords and Acknowledgements	ii
0.2	Declaration of Independent Work	iii
0.3	Figure list	iv
0.4	Abbreviations	v
1	Introduction	3
1.1	Study Area of Interest	7
1.1.1	Kalobeyei, Kakuma, Turkana, Kenya	7
1.1.2	Dzaleka, Dowa, Malawi	11
2	Literature Review	14
2.1	Remote Sensing of Informal Settlements	14
2.2	Deep Learning in Urban Remote Sensing	15
2.2.1	Computer Vision and a brief review of Convolutional Neural Networks	16
2.2.2	UAV-based informal settlement segmentation	18
2.2.3	Fundamentals of Deep Learning and Convolutional Neural Networks	18
3	Data and Methodologies	25
3.1	Data	25
3.1.1	Raster pre-processing	25
3.1.2	Data Augmentation	28

3.2	Research Questions and experiment design	30
3.3	Architecture and hyperparameter selection	32
3.3.1	The U-Net and U-Net variants	32
3.4	Hyperparameters and baseline model performance	35
3.5	Accuracy Assessment	38
3.5.1	Lower-order metrics	39
3.5.2	Higher-order metrics	40
3.5.3	Project workflow	42
4	Findings and Discussion	44
4.1	Depth-wise Precision and Recall change	48
4.2	Dataset-wise Precision and Recall change	49
4.3	Initialised weight Precision and Recall change	51
5	Conclusion	54
6	Bibliography	56
6.1	Appendix	57

Introduction

The world's population is more urbanised than ever before. As of 2018, approximately 4 billion (55%) (UN DESA., 2018, Taubenöck et al., 2009) reside in urban areas, of which 60% reside in slums often located at the fringes of the city (Venables A., 2018). Urbanisation growth is expected to increase by 2.5 billions between 2018 to 2050, most of which will be in Asia and Africa (UN DESA., 2018). When population growth outpace development, informal settlement become the supplier of significant housing stocks. These informal settlements are dynamic and represent a good reflection of cultural practices, access to resources, financial limitations and other socio-economic conditions. This means the informal settlement differs significantly between urban and rural settlements from roof covers, densities, and are subjected to different levels of access to resources and the types of resources.

Refugee camps are often the common or only way for displaced people to receive shelters and assistance. They are often setup in place of proximity to displaced population, whether that be from natural disasters, human caused disasters, or other reasons. Throughout history, refugee sites have provided haven to the world's most vulnerable population (UN, 2018, Turner S., 2016, UNHCR, 2021). However as of 2020, out of the 26.4 million refugees, only around 1.4 million have access to third country solution between 2016 to 2021 (UNHCR, 2021). Additionally, although officially defined as temporary settlement, many refugee camps have had longer than expected life

cycle, some of them have even became "Secondary Cities" and therefore suffers similar problems of poor governance and rapid urbanisation which consequentially makes them unattractive as investment (Cities Alliance & AfDB., 2022). For the many refugee camps and informal settlements that have lasted well beyond their expected temporary role, there are generally 3 ways of solving the issue: 1. Voluntary repatriation, 2. Relocation to third country, 3. Local integration as outlined by the Global Compact on Refugees (UN, 2018), although actual implementations are often subjected to the wills of the host sovereign-state. Recent studies have suggested that local integration often have a net positive economical impact on the surrounding region (Alix-Garcia et al., 2018, Rummery A., 2019, IFC., 2018).

The United Nations Department of Economic and Social Affairs have published a set of 17 Sustainable Development Goals (herein SDG) to be achieved by 2030 (UN, 2015). Special attention are drawn to Goals 1 and 10 that are particularly relevant to this study.

- *Goal 1: End poverty in all its forms everywhere*
 - *Target 1.1: By 2030, eradicate extreme poverty for all people everywhere, currently measured as people living on less than \$1.25 a day*
 - *Target 1.4: By 2030, ensure that all men and women, in particular the poor and the vulnerable, have equal rights to economic resources, as well as access to basic services, ownership and control over land and other forms of property, inheritance, natural resources, appropriate new technology and financial services, including microfinance*
 - *Target 1.b: Create sound policy frameworks at the national, re-*

gional and international levels, based on pro-poor and gender-sensitive development strategies, to support accelerated investment in poverty eradication actions

- *Goal 10: Reduce inequality within and among countries*
 - *Target 10.1: By 2030, empower and promote the social, economic and political inclusion of all, irrespective of age, sex, disability, race, ethnicity, origin, religion or economic or other status*
 - *Target 10.7: Facilitate orderly, safe, regular and responsible migration and mobility of people, including through the implementation of planned and well-managed migration policies*

Having up-to-date map is therefore paramount for short and long term humanitarian projects, from the delivery of essential medicine, spatial and policy planning, to population estimation, quality and timely maps are essential to improve future decision making in both humanitarian and non-humanitarian context. Although we have seen an overall net increase in the amount of buildings mapped on the OpenStreetMap platform, contribution is still skewed towards developed cities and countries. Patterns of episodic contribution maybe observed post disasters, but the contributed data inequality is not easily reconciled (Herfort et al., 2021) (*see figure 1.1*).

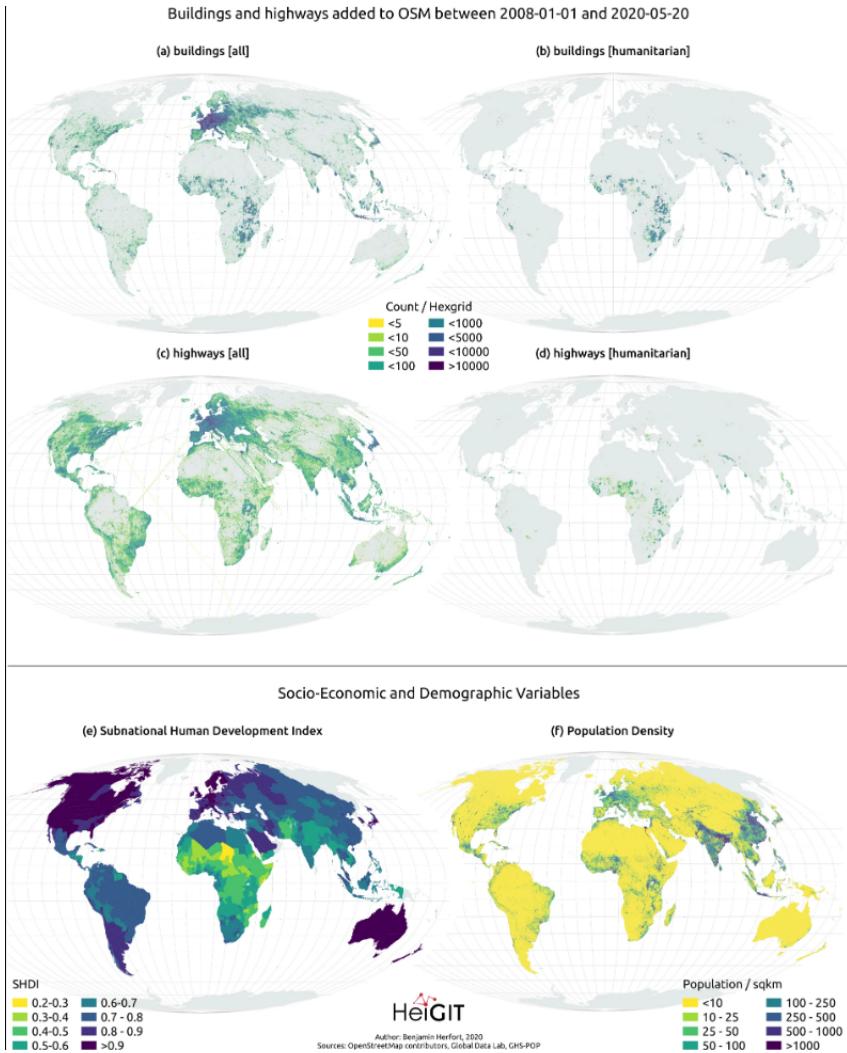


Figure 1.1: Data contribution of buildings and highway in all and humanitarian settings within OSM (Herfort et al., 2021)

Part of the reason for this could be traced to both the knowledge of volunteer contributors and commercial interest, but further discussion will be beyond the scope of this study (Anderson et al., 2019, Veselovsky et al., 2021, Yang et al., 2016). Henceforth, the overarching theme of this study is how can technology help reduce the geospatial data inequality in informal settlements. As the topic and data provider of this project, the Humanitar-

ian OpenStreetMap Team (herein HOT) have been at the forefront of using open and crowd sourced mapping data to support humanitarian causes from shorter term disaster response to longer epidemiology and microfinance campaigns (HOT, 2021). and with the advancement of Deep Learning in the last decades, AI-assisted mapping have became a major topic for innovation (e.g. Herfort et al., 2019, Kuffer et al., 2016, Wurm et al., 2021, Quinn et al., 2018).

Study Area of Interest

Kalobeyei, Kakuma, Turkana, Kenya

The Kakuma camp was first established in 1992, located in the rural North-West county of Turkana, Kenya. The camp was initially established to provide accomdation to the refugees fleeing the Second Sudanese Civil War as a temporary solution. However, as the conflict dragged out and followed by subsequent conflicts in the nearby region, the Kakuma camp have therefore been running for the past 30 years. As of 2020, Kakuma is home to 157,718 refugees with increasing number coming from the more recent Somalian and Ethiopian-Eritrean conflict (IFC., 2018, UN-HABITAT, 2021).

The Kakuma refugee camp have fluctuated in population as a response to demand, however, a dramatic increase in population between 2013 and 2014 has culminated into the development of Kakuma 4 Camp and the Kalobeyei Settlement and the Kalobeyei Integrated Socio-Economic Development Plan (KISEDP). The settlements benefited from a much better spatial planning in order to facilitate inclusive socio-economic development (UN-HABITAT, 2021, UNHCR & DANIDA, 2019) (*see figure 1.2*). Both the Kakuma and Kalobeyei refugee camps have local integration as the targeted solution (UN-

HABITAT, 2021, UNHCR & DANIDA, 2019). A comprehensive study of the formal and informal economy of Kakuma refugee camp conducted by the International Financial Corporation (IFC, 2018) suggests that that market catering for the refugees and surrounding towns is estimated at KES 1.7 billion (USD \$16.4 million). The economical vibrancy of local integration have improved the economy of impoverished Turkana county significantly. However, challenges remain in integration into the wider Kenyan economy.

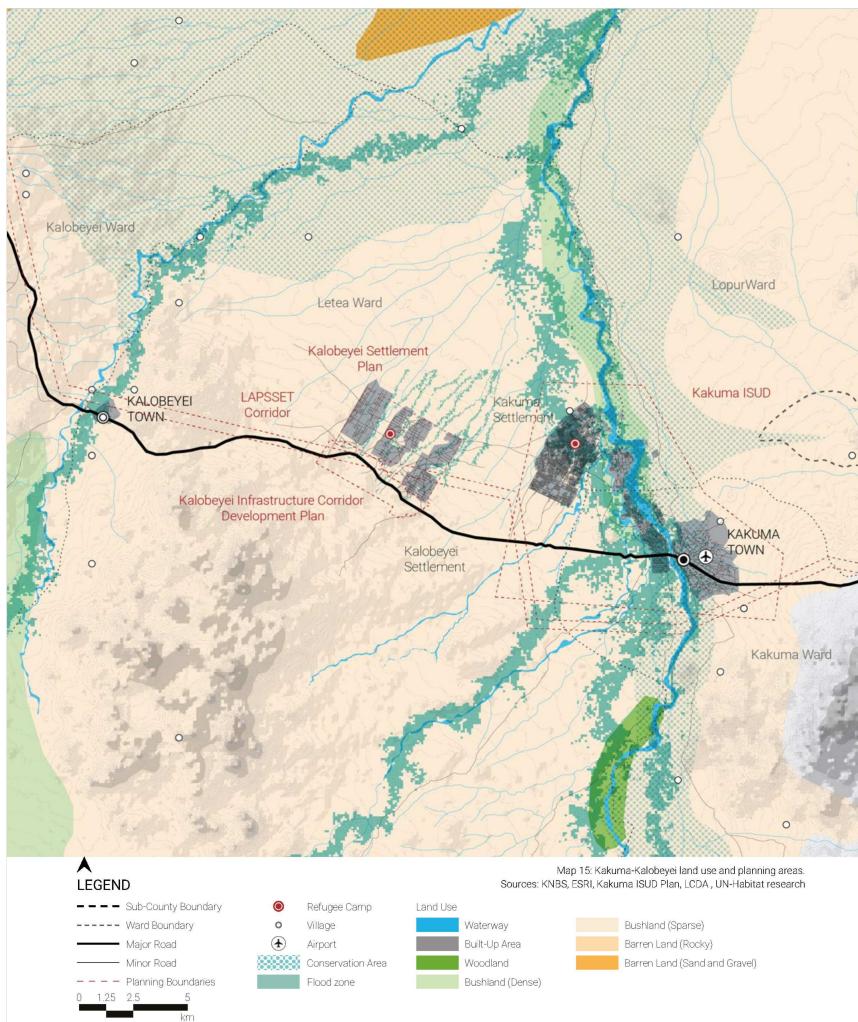


Figure 1.2: The Kakuma-Kalobeyei land use and planning areas (UN-HABITAT, 2018)

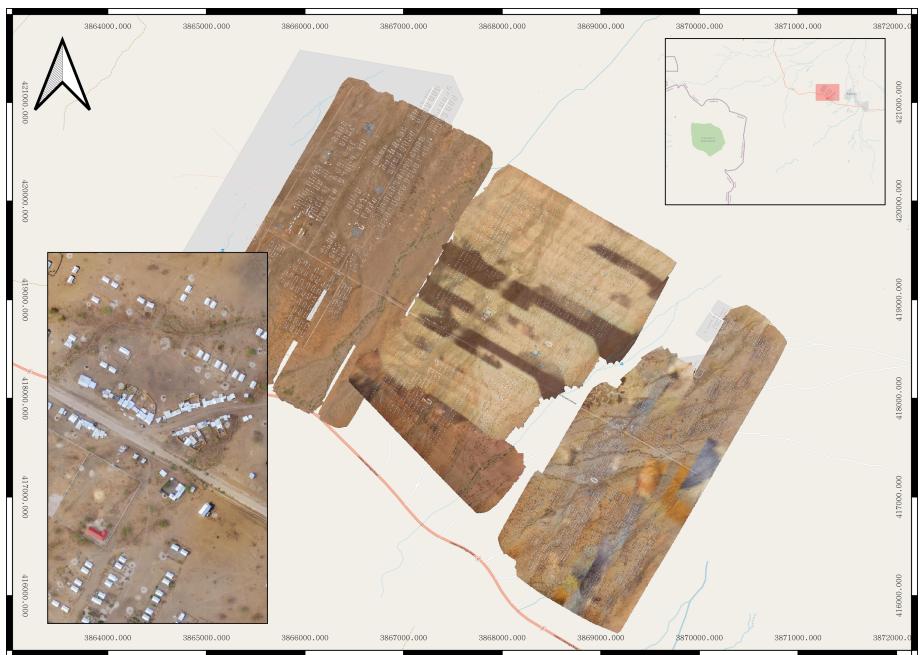


Figure 1.3: RGB UAV imagery of the Kalobeyei settlements in rural Turkana from OpenAerialMap

Dzaleka, Dowa, Malawi

Originally an infamous prison camp under the Banda's Malawi Congress Party regime, the area was converted to become the Dzaleka Refugee Camp in 1994. Unlike the Kakuma and Kalobeyei camps, the Dzaleka Refugee Camp is located in the heart of Malawi, 45 km away from the capital Lilongwe, where it is home to around 52,000 refugees and receive on average 300 new residents every month. Most coming from the Great Lakes area, in particular, the Democratic Republic of Congo and Burundi. However, resurgence of past conflicts between the Republic of Congo and D.R. Congo have caused an increased of influx in recent years (UNHCR, 2014, Kavaloski E., 2016). Much of the infrastructure in the Dzaleka camp remain rudimentary at best, and very little resources and statistics were available via the UNHCR and UNDP portals. The Northern extension to the Dzaleka main camp is known as the Katubza extension (*referred to as Dzaleka North by the rest of this study*), it is a well-planned plot of land consisting of 423 shelter shelters and were still in construction as of March 2021 (Gross G., 2021 & UNHCR, 2021) (*see figure 1.4*).

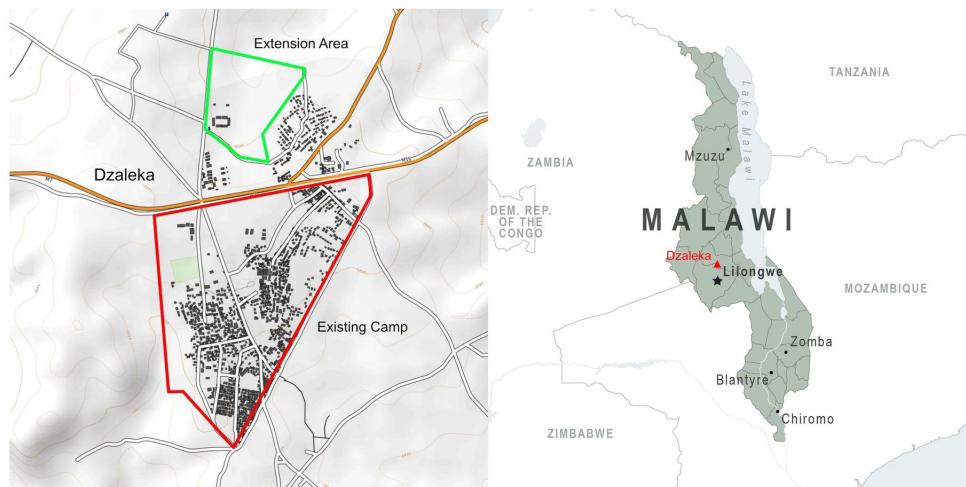


Figure 1.4: The main Dzaleka Refugee Camp and the Katubza extension plan (Dzaleka North) designed by Urban Design Advisor to the UNHCR Werner Schnellenberg (Gerhard G., 2021).

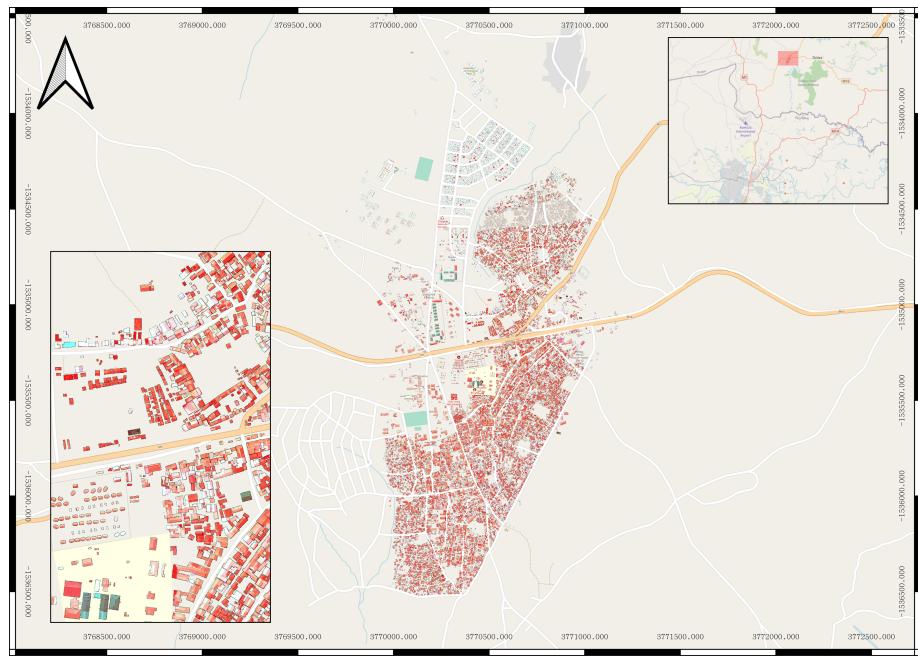


Figure 1.5: Digitised rooftop of the Dzaleka and Dzaleka North camps by HOT volunteers

Although hosting of refugee camps are often seen as a burden on the surface level, in reality, many refugees are often more educated than the local population, which brings with them entrepreneurial ability and provide the local-area with extra labour force (Alix-Garcia et al., 2018). With constant stakeholder pressure for relocation and closure, showcasing of the refugee camps local economic impact and the potential of, aiding the formation of pro-poor policy development(Cities Alliance, 2022).

Literature Review

Remote Sensing of Informal Settlements

Remote sensing of the urban environments have always been the less orthodox topic in remote sensing research, but simultaneously, it is the topic that essentially funded many long-standing satellite programs due to its proximity to reconnaissance and surveillance applications (e.g. the Landsat and SPOT programs). The complexity of human built environments often consist of using very different materials in conjunction to each other in a dense environment. From power-lines, factories, car-park, to leisure-parks, imaging of urban environments therefore requires imagery high in both spatial and temporal resolution. Urban remote sensing calls for techniques that extracts geometry, textural, and other physical features as oppose to the more common spectral based index approach used in ecological or environmental remote sensing (Jensen J., 2007, NRC., 1998). Higher granularity urban remote sensing have until very recently remained in the monopoly of defense and reconnaissance services.

Informal settlement and slums mapping of developing countries require very high resolution (VHR) images which was unavailable until the turn of the century. The relatively new technology thus only began to gain traction within the last 2 decades. Particularly with the increase in the availability of civilian commercial VHR satellites. Increase in computational power

had enabled novel techniques such as multi-layer machine learning, textural analysis, and novel geostatistical methods to emerge (Kuffer et al., 2016). Due to frequent repeat coverage of satellite's orbit, they can be used to fill in between periodic census that are costly and time consuming to conduct. Census especially conducted in developing areas also does not do a good job in capturing larger scale units and spatial patterns, potentially overlooking others socio-economic determinants such as access to amenities and infrastructure, adequate pro-poor policy development hinges on the availability of up-to-date and good quality analysis (Kuffer et al., 2016, Sliuzas et al., 2017). Remote sensing of settlements largely falls under 2 categories, rural or urban. Due to the different make-up of socioeconomic context and urban morphology, sensing of rural and urban settlements require different parameters. Additionally, there's no "one size fit all" way to generalise informal and formal settlement across the world, as physical geography, topography, cultural, and available resources often dictate the distribution, development, and settlement clusters pattern. These unique requirements have thus made deep learning techniques particularly useful, and many application of deep learning in remote sensing have therefore been in the urban domain (Ma et al., 2019).

Deep Learning in Urban Remote Sensing

The following section will be divided into 3 parts. The first part reviews the concept of Computer Vision (herein CV) as a study subject, previous common practices, and the evolution towards data-driven Deep Learning. The second part will focus on domain specific review of recent AI-based segmentation practices on building segmentation and particularly the recent practices in informal settlement segmentation. Lastly, the third part will explain the mechanism of the Convolutional Neural Network, the class of neural networks commonly used for CV tasks.

Computer Vision and a brief review of Convolutional Neural Networks

Prior to the paradigm shift towards data-driven and ML based segmentation techniques, image segmentation were performed manually with the aid of a few algorithms (Pal & Pal, 1993). The image segmentation tasks often starts with acquiring less-noisy imagery or data, this is followed by applying multitudes of CV based edge detector (e.g. Sobel, Prewitt, Marr-Hildreth) or Grey-Level Co-occurrence Matrix kernel (e.g. Haralich Texture) (e.g. Kuffer et al., 2014, Kuffer et al., 2016, Wurm et al., 2017). This can be considered to be the pre-processing steps necessary to extract information to select the parameters for the segmentation algorithms. (Pal & Pal, 1993, Blaschke T., 2010, Blaschke et al., 2014). The field of CV based segmentation experienced an akin to a Kuhnian paradigm shift (Kuhn T., 1962) when Krizhevsky et al. (2012) AlexNet won the ImageNet challenge, it reinvigorated the use of multi-layered neural network in CV tasks (LeCun et al., 2015). The timing of the paradigm shift coincided with the increase in computation power provided by Graphical Processing Unit (GPU) have enabled CNNs to be successfully applied across domains ranging from biomedical imaging to remote sensing (Ma et al., 2018, Zhu et al., 2017, Zhang et al., 2016, Wurm et al., 2019).

In the field of CV, there is generally 4 types of application: 1. Semantic segmentation, 2. Classification and localisation, 3. Object detection, and 4. Instance segmentation (*see figure 2.1*) (Stevens et al., 2020). This study will conduct semantic segmentation of binary classification between built-up and no built-up.

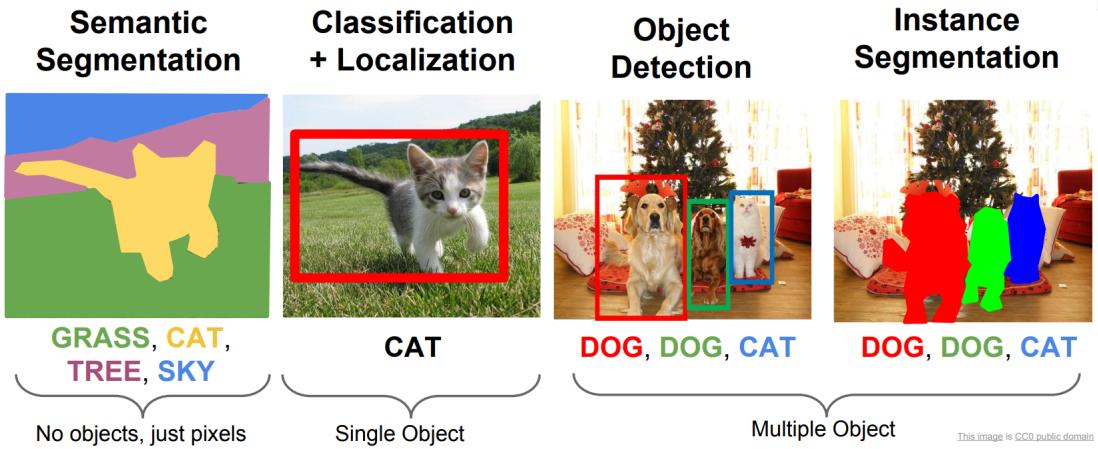


Figure 2.1: The four main types of Computer Vision tasks (Stanford University, 2022)

The purpose of semantic segmentation is to assign a named (semantic) classification to each and every single pixel of the input image. This is commonly applied in remote sensing of Land Use Land Cover classification where every single pixel will be assigned and Land Cover or Land Use type. Another application is binary segmentation where the model will only be trained to assign a named classification to a particular clusters of associated pixels. This is more common in single class segmentation. The difference between mere classification and segmentation is that the semantic segmentation output a mask over the pixel will be created, where each pixel within the mask belongs to the same semantic task; meanwhile, classification only gives a confidence of semantic of the whole scene without assigning the classification to each pixel.

UAV-based informal settlement segmentation

The advent of orthorectified photos from Surface-from-Motion have drastically democratised the collection of VHR imagery. Not only are UAV images economical to procure, the spatial resolution for informal settlement application could only be rivaled by perhaps the best reconnaissance or commercial satellite which is either difficult or very expensive to obtain. Therefore the availability of open-data platform such as the OpenAerialMap could play a key role in enabling the AI-assisted mapping.

Fundamentals of Deep Learning and Convolutional Neural Networks

Prior to the resurgence of popularity in DL, the set of methodology associated was known as a multi-layered perceptron. Initially inspired by a mathematical analogy to codify the function of a single neuron by the seminal Psychological review paper published by Rosenblatt F. (1958). Like a human neuron, The properties of a perceptron on the most fundamental level takes an information/numerical input, stores and apply transformation, and create an output. Through stacking of basic perceptrons, a multi-layered perceptrons structure can be created. In order for such structure to be computationally useful, it must satisfy the following criteria:

1. Collections of connected perceptrons are capable of plasticity (i.e. changing values) through training.
2. Perceptrons will form dominant pathways that "fire" (activate) together.
3. Through training, perceptrons will learn to apply positive or negative reinforcement to facilitate minimising error (e.g. assigning and

changing "weights").

The particular group of such perceptron structures used in CVs are known as Convolutional Neural Network (herein CNN). The most basics of CNN consist of 3 parts: 1. A hidden layer, 2. Multiple hidden convolution and pool layers, 3. An output layer which provides with the segmentation result and the associated confidence level (*see figure: 2.2 2.3*). Therefore, a Deep-Learning Neural Network system is string together by a series of inter-connected layer of which its parameters are adjustable to adapt to the data provided. Through careful iterative training and adjustment, the system can generalise well not only to the training and validation data, but to future datasets as well.

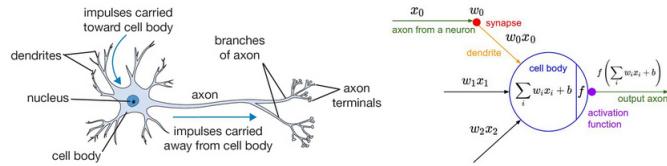


Figure 2.2: Schematic analogy diagram between a biological neuron and an artificial perceptron (Fumo D., 2017).

$$f\left(\sum_{i \dots n} w_i x_i + b\right) \quad (2.1)$$

- Where:

- f = Activation function
- $\sum_{(i \dots n)}$ = Summation of i to nth dimension
- $w_i x_i$ = Weights multiplied by original input variable (x)
- b = bias

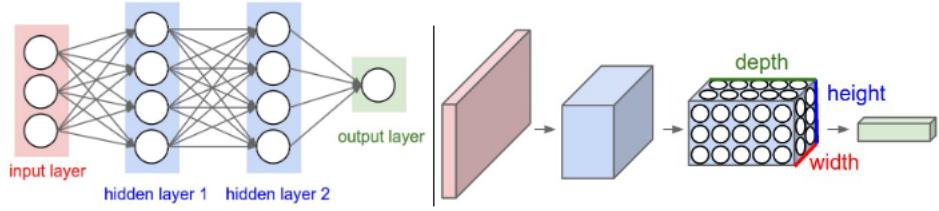


Figure 2.3: Schematic diagram of a CNN (Stanford University, 2022).

Convolution and Pooling

The hidden layers of the CNN is where the network performs representation learning, where with each layer in depth learns more abstract features of the input training image. While not often the case, it is conventional practice to interleave the convolutional and the pooling layers (Stevens et al., 2020). The convolutional layer employs a sliding kernel which applies the weighted filter as displayed in 2.1.

The convolutional layer essentially treats every image pixel as vector in a 3-Dimensional layout with input of ($Batch\ Size, Channel_{in}, Height, Width$), the convolutional kernel slides and apply the weighting and bias terms to extract deeper features (see figure: 2.4) (Stevens et al., 2020), thus, learning more deeper features which creates the output of ($BatchSize, Channel_{out}, Height, Width$). The full transformation per convolutional layer transform equation: 2.1 of each pixel into equation: 2.2.3.

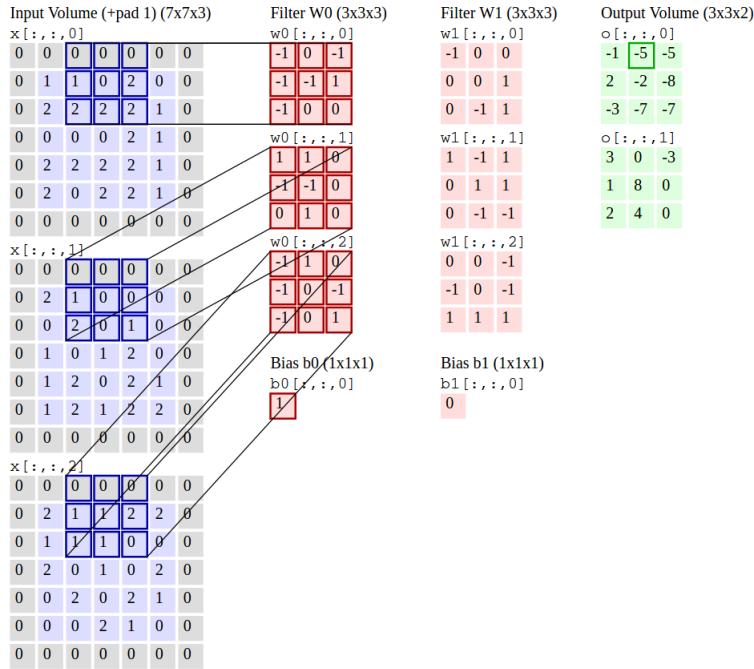


Figure 2.4: 3 x 3 Convolution (Stanford University, 2022).

$$out(N_i, C_{outj}) = bias(C_{outj}) + \sum_{k=0}^{C_{in}-1} weight(C_{outj}, k) * input(N_i, k)$$

- Where:

- N = Batch Size
- C = Channels
- k = Kernel Size
- *Further details can be found in [PyTorch documentation for nn.Conv2d](#)

(2.2)

The pooling layer reduces the dimension by downsampling by applying conventionally, a smaller kernel which extract the desirable value when applied. Maximum Pooling, which only return the largest value in the down-sampling kernel is common and is used for the network architecture of this experiment (Stevens et al., 2020). The pooling layer scale down the image while retaining the most crucial information (*see figure: 2.5*).

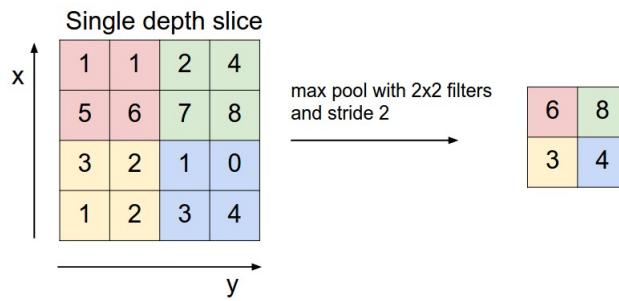


Figure 2.5: Max pooling (Stanford University, 2022).

Optimiser and the Binary Cross Entropy Loss function

With each complete pass through the neural network, the ouput is then compared against the validation result for error calculation. The summed average of loss thus defines the cost function landscape from which the error value is calculated against, penalising when prediction is incorrect and rewarding if otherwise. This enables backpropagation and the adjustments of the weights and biases in the proceeding pass. Due to the binary segmentation task for this study, the Binary Cross Entropy loss function was used to measure error *equation: 2.2.3*.

$$l(x, y) = L = \{l_1 \dots l_N\}^T, l_n = -w_n[y_n * \log x_n + (1 - y_n) * \log(1 - x_n)]$$

- Where:
 - N = Batch Size

- $l(x, y) = \text{loss}(\text{Probability}, \text{Binary Classification})$
 - $l_n = \text{loss at sample } n$
 - *Further details can be found in [PyTorch documentation for nn.BCELoss](#)
- (2.3)

The optimiser controls the gradient descent, it is the hyperparameter which controls the size of step being taken down the negative gradient of $-\nabla C$ in the cost function landscape. A suitable optimiser can prevent gradient descent to be trapped at a local minimum through iterations. The optimiser of choice for the experiment is the Adam optimiser, The Adam (Adaptive Momentum Estimator) developed by Kingma & Ba (2017) extends the Stochastic Gradient Descent (herein SGD) by introducing concepts of momentum and second moments of gradient (*see appendix 6.1*). Adam maintains a separate learning rate for each parameter and have became the standard pick of optimiser since.

Backpropagation and the chain rule

In order for a neural network to improve, the weights and bias are changed accordingly to minimisation of the cost. This is computed as a step-wise function as a negative vector against the cost landscape. For which each parameter of the weights and biases of each neuron within the neural network is defined as a chained function against the cost in *equation 2.4*. In other words, what is the derivate of the cost function with respect to the chain of weights and biases derivatives.

$$\frac{\delta C}{\delta P^i} = \frac{\delta(w^L x^{i-1} + b^i)}{\delta P^i} \frac{\sigma[\delta(w^i x^{i-1} + b^i)]}{\delta(w^i x^{i-1} + b^i)} \frac{\delta C}{\sigma[\delta(w^i x^{i-1} + b^i)]} \quad (2.4)$$

- Where:

- δC = Derivative of Cost Function
- δP^i = Derivative of a Parameter, which could be the w^i weight or b^i bias or for activation function $\delta(w^i x^{i-1} + b^i)$ at layer i
- x = Input Variable
- σ = Activation Function (e.g. ReLU, Sigmoid)

Which when *equation 2.4* is summed over all parameters of layer i becomes *equation 2.5*

$$\nabla C = \frac{\delta C}{\delta P^i} = \sum_{n_i=1}^{n_i-1} \frac{\delta(w^L x^{i-1} + b^i)}{\delta P^i} \frac{\sigma[\delta(w^i x^{i-1} + b^i)]}{\delta(w^i x^{i-1} + b^i)} \frac{\delta C}{\sigma[\delta(w^i x^{i-1} + b^i)]} \quad (2.5)$$

Thus, taking the negative gradient $-\nabla C$ will provide the gradient descent step hopefully towards the global minimum.

Essentially, using CNN and DL methods to perform semantic segmentation through repeated iterations of the above processes have proven to generalise well to complex dataset. Although the aim of this study is not to produce a deployable model necessarily, the study will lay the foundational groundwork for a data-drive evaluation of different CNNs elaborated in section [3.3](#).

Data and Methodologies

Data

Raster pre-processing

All imagery were first downloaded from OpenAerialMap and resampled to 15 cm resolution using cubic-spline interpolation, subsequently reprojected to EPSG:3857. Normalisation of raster data per colour band (RGB) to adequately re-scale the raster value to be converted to the PNG file format. This is perhaps one of the most important pre-processing step. A 2-step normalisation were performed on each band for each UAV imagery.

First, the z-score normalisation noramlises the images according to the retrieved standard deviation (*see equation 3.1*). This scales the every pixel to the global statistics for each colour band, keeping proportional ratio while reducing the effects of outlier. The z-score normalised result is then linearly scaled to range of 0 to 255 to be converted to 8-bit .png type file (*see equation 3.2*).

$$p_z = \frac{(p - \mu)}{\sigma} \quad (3.1)$$

- Where:
 - p_z = z-score normalised pixel value

- p = Original pixel value
- μ = Mean value of global pixel
- σ = Standard Deviation of global pixel

$$p_{8bit} = \frac{[p_z - \min(p_z)] * 255}{[\max(p_z) - \min(p_z)]} \quad (3.2)$$

- Where:

- p_{8bit} = Pixel output normalised between 0 to 255
- p_z = z-score normalised pixel value from 3.1

After per-band normalisation, the imagery bands were stacked with the associated labels. In order to increase the data quantity, $\frac{2}{3}$ overlapping steps cropping was performed. This resulted in the image label pair count of Train n = 2606, Validation, n = 1303, and Test n = 435 respectively where each sets were split at a ratio of 60, 30, and 10 % respectively. With augmentation applied, this increased the available data to Train n = 18242, Validation n = 3909, and Test n = 435. (*see figure. 3.3 & see table ??*)

Vector pre-processing

Semantic segmentation tasks require very high quality and quantity of data input in order to successfully perform. The reference dataset must therefore be highly accurate, otherwise this could cause the model to misclassify. There were two significant issues, firstly, due to the centermeter level resolution of the UAV raster data, the abundance of building polygons from OpenStreetMap and [Google Open Buildings V1 Polygon](#) did not spatially align well even after reprojection. Secondly, there were a temporal mismatch between such vector labels and the UAV in collection, causing some labels

without building and buildings without labels.

Fortunately, prior to this study, the HOT team and volunteers have began collecting labels specifically digitised on the UAV imagery of Dzaleka and Dzaleka North camps *see figure 1.5*. Although spatially and temporally aligned, these vector labels still did not have a DL CV tasks in mind, hence, labelling around edges of buildings and UAV motion artefacts (Smith et al., 2016, Caravick et al., 2016) *see figure 3.1* may have been missed.

With the Kalobeyei camp unlabelled, this study created albeit in less quantity, a carefully digitised, pixel-aligned dataset which is suitable for DL tasks. The combination of datasets provided gave this study an unique opportunity to investigate how the different datasets could influence segmentation results.

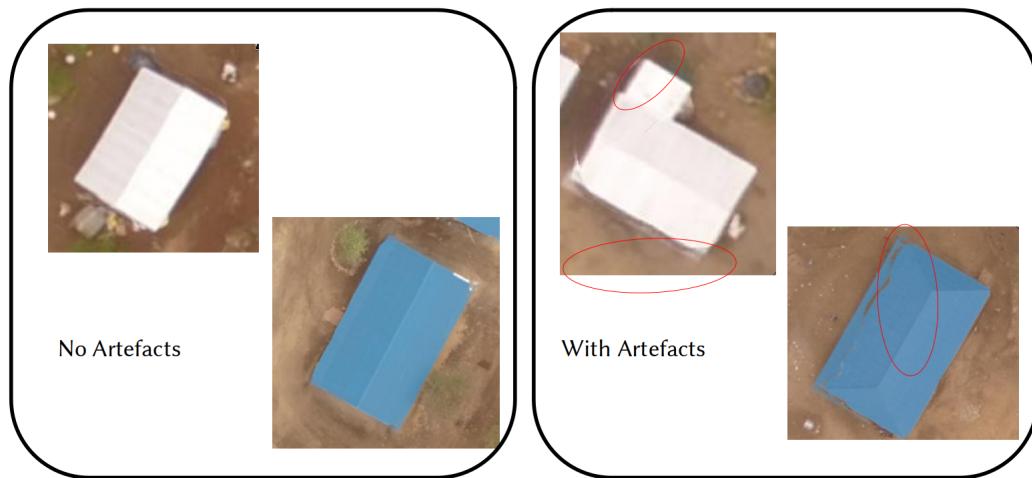


Figure 3.1: Motion artefacts unique to UAV imagery

Data Augmentation

Data augmentation is perhaps one of the most crucial task in training a robust neural-network. It is an economical way of increasing generalisability without increasing model complexity, data augmentation achieve this through, firstly increasing the quantity of training and validation data, secondly encompassing a greater range of textural, geometrical, and colour variability through the creation of augmented pseudo-data (Shorten & Khoshgoftaar, 2019; Kinsley & Kukiela, 2020; Howard & Gugger, 2020; Zoph et al., 2019).

Data augmentation can generally be split into 3 categories: 1. Geometric/Affine distortion, 2. Colour distortion, and 3. Noise distortion. The application of which types of distortion to the *Train* and *Validation* dataset is highly dependent on the context of the semantic task. Therefore, care must be taken as to not introduce mislabelling (*see figure 3.2*) (Ng A., 2018).

Augmentation categories:

- Geometric/Affine distortion
 - e.g. Fliping, Stretching, Rotation...
- Colour distortion
 - e.g. Colour Inversion, Solarise Colour, Greyscale...
- Noise distortion
 - e.g. Blurring, Contrasting, Salt & Pepper...

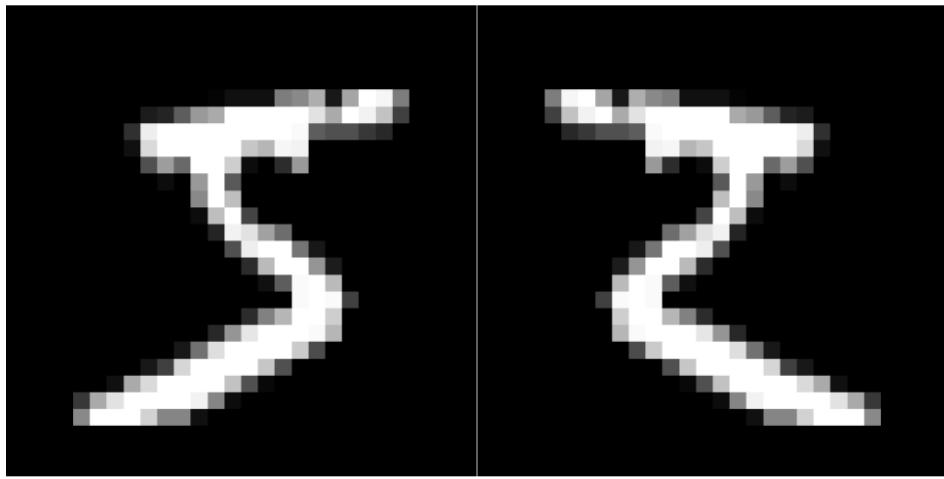


Figure 3.2: Perhaps geometric augmentation of horizontal flipping shall not be applied on the MNIST number of 5

Thus, the following augmentation were applied to the Train, Validation, and Test datasets respectively:

- Train - Inverse RGB, Horizontal Flip, Vertical Flip, Gaussian Blur, Contrast Increase, Solarise Colour
- Validation - Horizontal Flip, Vertical Flip
- Test - None

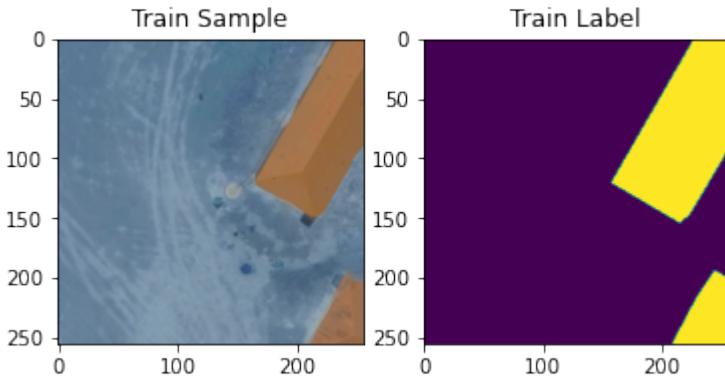


Figure 3.3: An example of Inverse RGB augmentation applied to the Train dataset.

Dataset input with augmentation				
Datasets		Train	Validation	Test
KBY		5719	1224	272
KBV + DZK +		18242	3909	435
DZKN				

Table 3.1: Resulted image and label pair for each dataset input configuration

Research Questions and experiment design

In order to train a model which performs well on UAV imagery, the motion artefact will be a significant feature for the model to learn. The combination of data availability have allowed a unique set of research questions concerning the input data quality and experiment setup to surface.

Additionally, using pre-trained weights as a strategy is well-documented in literature to improve performance across many domains (Stevens et al., 2020 , Howard & Gugger, 2020), with numerous studies showcasing the success of cross-domain transfer training from classical CV datasets to remote sensing tasks (e.g. Audebert et al., 2017, Marmanis et al., 2016), One would expect the transfer training of CNN pre-trained on any dataset would

provide it with an advantage. Therefore, it is important that this study also test the effects of the CNNs response to the initialised weights from ImageNet and the OCC building segmentation model.

1. RQ1: What is the optimal mixture of accurate and less-accurate labels and how does that affect the segmentation output result?

(a) How does the introduction of complex roofing materials affect result?

2. RQ2: Which is the best lightweight model given the limited data and computational resources for binary semantic segmentation?

(a) How does transfer training from various initialised weights affect results at different setup?

To test out U-Net and a few variations of the U-Net performance, The models will initially be trained on the pixel-perfect and less complex Kalobeyei dataset, this will be then be followed by introducing the Dzaleka datasets of higher complexity. *Figure 3.4* show a snapshot of the diverse rooftops to be segmented in the available datasets. A comparison of performance between the U-Net variations (Ronneberger et al., 2015) and the [Open-Cities-AI-Challenge \(herein OCC\) winning model](#) is conducted (*see table 3.3.1*).



Figure 3.4: Collections of diverse and heterogeneous rooftops from the Kalobeyei, Dzaleka, and Dzaleka North datasets.

Architecture and hyperparameter selection

Model architecture and their associated hyperparameters selection is highly dependent on the computational resources and the task at hand (Ng A., 2018, Howard & Gugger, 2020). As this study aims to output a pixel-based binary segmentation which delineates building and non-building, and given the computational resources constraint, model selection were based on tried and tested architectures with relatively low number of training parameters.

The U-Net and U-Net variants

The U-Net architecture was first developed by Ronneberger et al. (2015) for the task of cell segmentation in biomedical electronmicroscope images. The architecture feature a symmetrical Encoder-Decoder structure (*see figure 3.5*) and as with many other CNN, the architecture have transferred successfully well into the remote sensing domains (Höser & Künzer, 2020, Höser et al., 2020, Xu et al., 2019). This symmetrical encoder-decoder type

architecture is able to extract deeper features in the encoder layers, then recover and interpolate spatial features in the connected unsampling decoder layers (Wurm et al., 2019).

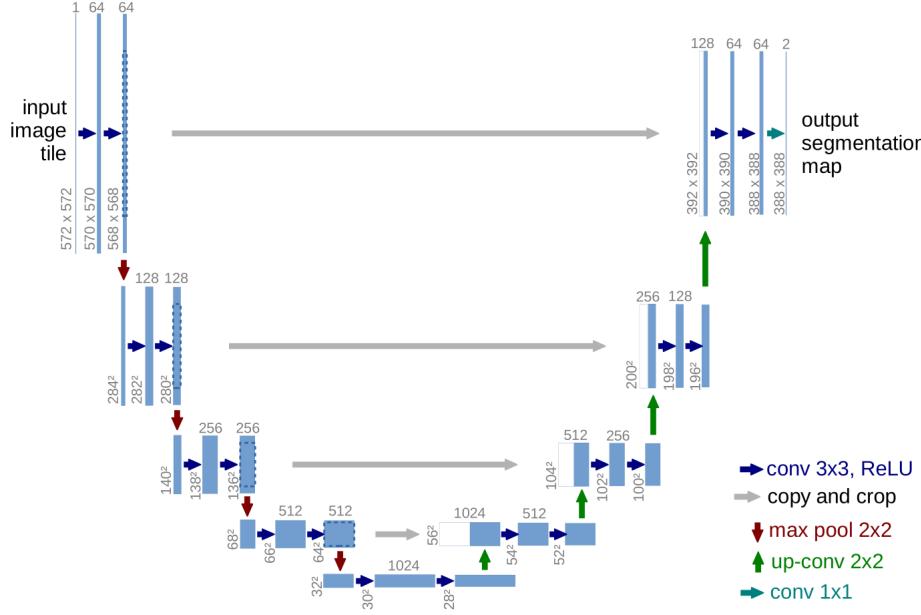


Figure 3.5: The Encoder-Decoder U-Net architecture (Ronneberger et al., 2015)

Changing the encoder architecture and the EfficientNet family

The ability to switch out the encoder structure allows the DL practitioner to experiment with more up-to-date architectures without changing the output shape. This drastically increases the combination of experiments that allow testing the best combinations of encoder-decoder structure suitable for the dataset. All of the experiments in this study were carried out using the high-level PyTorch API [Segmentation-Model-PyTorch](#) created by Yakubovyskiyl P. (2021). Whom was also the winner of the OCC challenge for UAV building segmentation which this study will transfer-train, and compare against.

In this study, the experiments with changed encoder were based on the EfficientNet family. There are three reasons for this decision. Firstly, at one of the last stage of the OCC competition winning network, the EfficientNet B1 was used as an encoder. Secondly, the EfficientNet family are a set of network architectures that are structured and easy to scale up when computational resources become available. Thirdly, they are perhaps the best representation of generalised state-of-the-art architectures that have been tested and performed phenomenally well in classical CV datasets (*see figure 3.6*) (Tan & Le, 2020). In essence, these are sets of experiments that mix and match old and new architectural design.

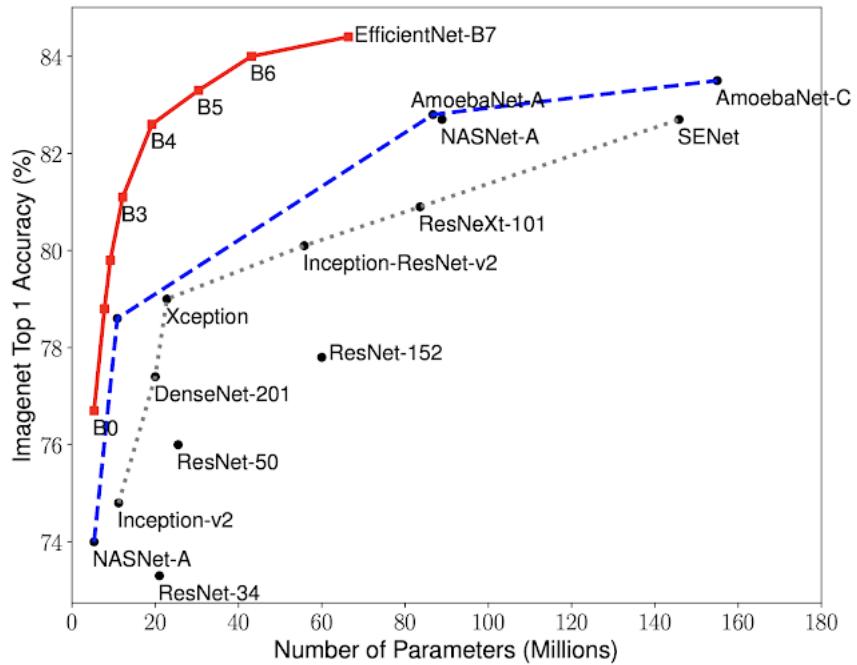


Figure 3.6: EfficientNet family Top 1% Accuracy Assessment in ImageNet (Tan & Le, 2020).

The EfficientNet family uses compound scaling which increases the height,

width, and depth. The baseline architecture was generated using AutoML Neural Architecture Search (Elsken et al., 2019) which optimised for computational efficiency and accuracy (*see appendix 6.2*).

Therefore, the unweighted Four-layer U-Net, Five-layer U-Net, and the OCC winner weighted EfficientNet B1 U-Net are the key architectures the rest will compare against.

Trained Architecture Specification Table				
Encoder	Decoder	Initialised weights	Trainable parameters	Batch-size (8GB GeForce GTX 1070Ti)
4-layer U-Net Encoder	4-layer U-Net Decoder	None	776,3041	32
5-layer U-Net Encoder	5-layer U-Net Decoder	None	3110,0513	32
EfficientNet-B1	4-layer U-Net Decoder	None	700,5041	32
EfficientNet-B1	4-layer U-Net Decoder	ImageNet	700,5041	32
EfficientNet-B1	5-layer U-Net Decoder	OCC	875,7105	16
EfficientNet-B2	4-layer U-Net Decoder	None	821,1283	32
EfficientNet-B2	4-layer U-Net Decoder	ImageNet	821,1283	32

Table 3.2: The U-Nets and the variations thereof selected for this study

Hyperparameters and baseline model performance

The hyperparameters of a neural network are changable parameters which control the training process (Bengio et al., 2017, Stevens et al., 2020, Howard & Gugger, 2020). They include the batch size, optimiser, learning rate, weight decay, loss function, and learning rate scheduler etc. (*see table 3.4*). One of the most difficult process in DL is finding the correct hyperparam-

eters value that cause the model to neither overfit or underfit the dataset. The strategies and options are often overwhelming, therefore this study does not concern itself with tuning the hyperparameters for all of the models but rather withholding them from changes so that a comprehensive **baseline** performance could be identified. This will provide a clear picture of the feasibility, uncover challenges and potentials. The result will provide insight on how to take this project further, so that further resources could be justified to scale future experiments.

Unchanged hyperparameters for the study experiments		
Hyperparameter	Value to be held constant	Description
Batch size	32 (16)	Amount of image and label pair shown to the CNN per iteration until the dataset is exhausted, standardised to batch size of 32 other than for the architecture of <i>EfficientNet B1 U-Net OCC</i> with a 5-layer U-Net decoder
Optimiser	Adam <i>see appendix 6.1</i>	Adaptive Momentum Estimator developed by Kingma & Ba. (2017)
Loss function	Binary Cross Entropy	... <i>see equation 2.2.9</i>
Learning rate	1e-3	Size of step to be taken down the negative gradient of the cost function landscape
Weight decay	1e-5	aka. L2 regularisation, is the sum of all weight squared added to the Loss function. This limits the weights from growing too much, making the loss function easier to fit
Training epochs	500	Number of complete cycles through either the training or validation dataset at designated batch size
Validation rate	10 epochs	Validation data are loaded every 10 epochs to monitor performance
Learning rate scheduler	Reduce Learning Rate on Plateau [factor (0.1), minimum (1e-8), epoch (20)]	A learning rate reduction when learning stagnates and stop improving for 20 epochs

Table 3.3: The hyperparameters and respective values to be held constant for every experiments in this study.

Accuracy Assessment

Detail and scrutable accuracy assessments are fundamental towards any classification based analysis. This section will introduce and break down the various lower order and higher order class-based (thematic) accuracy assessment. By explaining the characteristics of each metrics, this will provide a much more granular nature of accuracy assessment in the findings of section 4. In general, accuracy assessment in remote sensing can be divided into 2 categories: 1. Positional Accuracy & 2. Thematic Accuracy. Of which, Positional Accuracy deals with the accuracy of the location while Thematic Accuracy deals with the labels or attributes accuracy (Congalton & Green, 2019 & Bolstad P., 2019). With lower order metrics being more granular while higher order metrics more triturated but generalised.

The metrics described in this section form part of the larger family of accuracy assessment metrics that can be constructed from the confusion matrix (*see figure 3.7*)

		True condition		Prevalence $= \frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Total population	Condition positive	Condition negative			
Predicted condition	Predicted condition positive	True positive	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR-}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
	False negative rate (FNR), Miss rate = $\frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) $= \frac{\text{FNR}}{\text{TNR}}$		

Figure 3.7: The Confusion Matrix

Lower-order metrics

Precision, Recall, and Specificity

Precision and **Recall**, aka. Positive-Predictive-Value and Sensitivity/True-Positive-Rate respectively. The two metrics are often used together, another common denomination especially in remote sensing literature are User's Accuracy and Producer's Accuracy (Congalton & Green, 2019 & Wegmann et al., 2016). To avoid further confusion in nomenclature, **Precision** and **Recall** will be used from hereon.

Precision is the measure of correctly predicted Positive class *True Positive* against all positive prediction assigned to that class *True Positive+False Positive* i.e. Given the predicted results, of those that are predicted as positive, what proportion were True. It can be expressed mathematically as:

$$Precision = \frac{True\ Positive}{(True\ Positive + False\ Positive)} \quad (3.3)$$

Meanwhile, **Recall** measures the correctly predicted Positive class *True Positive* against both the correct and incorrect prediction on the Positive reference class *True Positive + False Negative* i.e. Given the predicted results, of those that are referenced as positive, what proportion of those were True. It can be expressed mathematically as:

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)} \quad (3.4)$$

Specificity, aka. True-Negative-Rate measures correctly predicted Negative class *True Negative* against the correct and incorrect prediction on the Negative reference class *False Positive + True Negative* i.e. Given the predicted results, of those that are referenced as negative, what proportion of those were True. It can be expressed mathematically as:

$$Specificity = \frac{True\ Negative}{(False\ Positive + True\ Negative)} \quad (3.5)$$

Therefore, higher **Recall** suggests the model is better at identifying positives and vice-versa higher **Specificity** suggests the model is better at identifying negatives. Since this is an exercise that aim to maximise the positive prediction as a binary building segmentation classifier, emphasise will be placed on maximising **Precision** and **Recall**.

Higher-order metrics

The following are higher-order accuracy assessment metrics, where they often encompass the lower-order accuracy assessment metrics mentioned in the above section. Thus, although higher-order metrics provide a more generalised assessments, they often suffer from granularity. Hence such metrics are often employed as a means to evaluate and rank DL based CV challenges and competitions (e.g. Kaggle Challenges), but they are less so effective in the detailed assessment of segmentation results.

Overall Accuracy

$$Overall\ Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad (3.6)$$

The Overall Accuracy (herein OA) gives an easy to implement, general but incomplete answer of how well classification is doing. The metrics suffers when imbalance count of multiple-classes are involved.

Dice Score

$$Dice\ Score = 2 * \frac{Precision - Recall}{Precision + Recall} \quad (3.7)$$

The Dice Score aka. the F1 score calculates the harmonic mean of **Precision** and **Recall**, with contribution for both to be of balanced weightings, the Dice Score could be skewed by classification results with higher performance in either Precision or Recall. Additionally, the metrics does not take *True Negative* values into account if such statistics might be of interest.

Intersection-over-Union

$$IoU = \frac{A \cap B}{A \cup B} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{False Negative}} \quad (3.8)$$

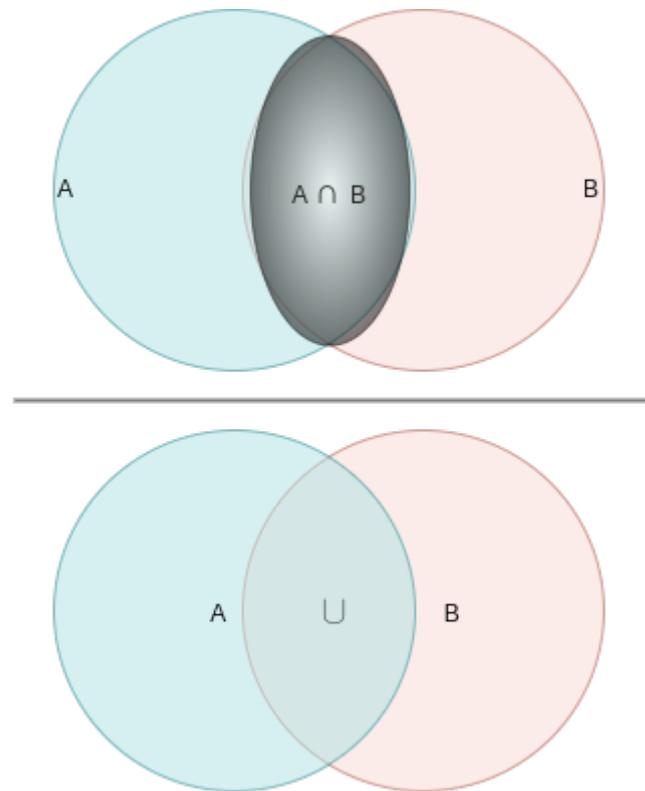


Figure 3.8: Schematic diagram of Intersection-over-Union

One of the most commonly used metrics as an assessment in Deep Learning Computer Vision competition. The Intersection-over-Union (IoU) aka.

Jaccard Index is a geometric based accuracy assessment

Project workflow

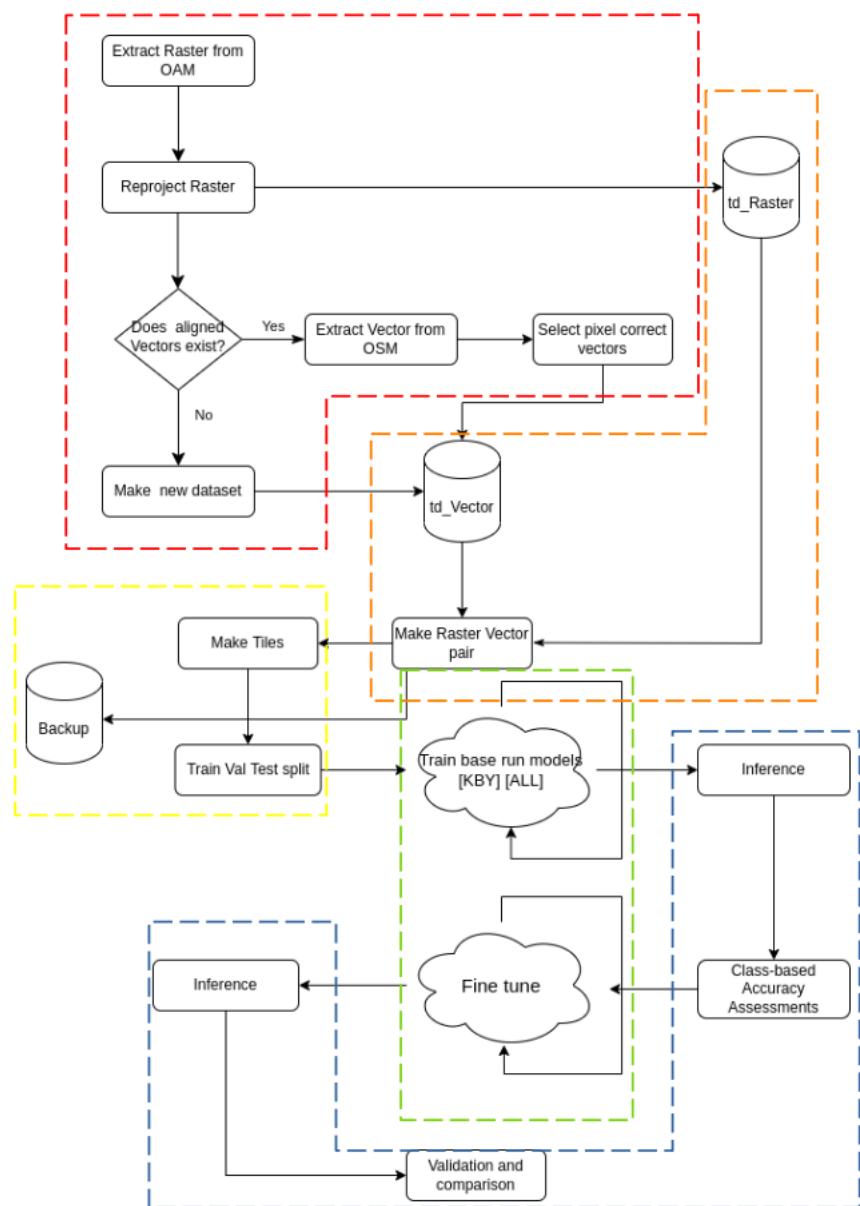


Figure 3.9: Project workflow

The workflow for this study consist of 5 main stages: 1. Download and Extraction, 2. Data pre-processing, 3. Data processing for loading, 4. Iterative Model training, 5. Inference and Evaluation (*see figures 3.9 & 3.10*).

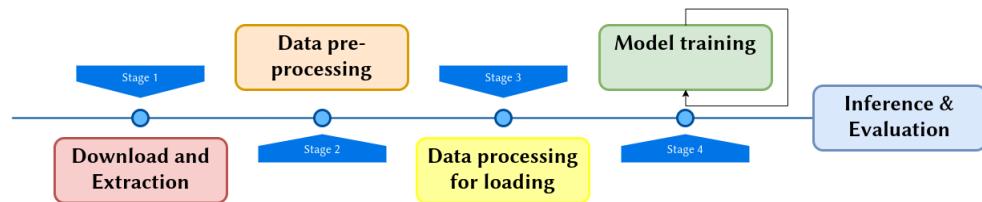


Figure 3.10: Simplified 5 steps project workflow with reference to 3.9

Findings and Discussion

The study have trained and tested on 5 different architectures with either no initialised weights or initialised weights pre-trained from ImageNet or OCC building segmentation models. For each experiment setup, there were 2 sets of dataset input (KBY and KBY + DZK + DZKN), for details (*see table 3.3.1*). Producing 16 sets of trained CNN and associated class-based accuracy assessments (*see figure 4.2*). On the whole, there were both expected and unexpected results. A reduction in every metrics were observed in every single experiment setup when the more complex Dzaleka camps datasets were introduced (*see table 4.2*). This was expected as it is more difficult to train the CNNs on the highly heterogeneous rooftops with similar texture to the surrounding environments. The *Precision* and *Recall* metrics did not vary too much between the architectures when trained only on the Kalobeyei dataset, with the exception to the transferred-untrained *EfficientNet B1 U-Net OCC* model. In contrast, the performance in differences is a lot more visible with the introduction of Dzaleka and Dzaleka North datasets.

An interesting observation is that with the *EfficientNet B1 U-Net* and the symmetrical *4-layer & 5-layer U-Net*, both *Precision* and *Recall* increased when initialised on pre-trained ImageNet weights (*see table 4.3*), with the exception to *EfficientNet B2 U-Net* where it a decrease in *Precision* but a increase in *Recall* when compared to the network with non-initialised weights. This suggests that the increase in depth does not necessarily cor-

respond to either an increase or decrease in performance in any particular direction. For the non-weight initialised *4-layer to 5-layer U-Net*, there is an consistent albeit slight improvement in *Recall* but not *Precision*. This contrast the changes in Depth-wise changes, where increase in depth shows improvement mostly in the *Precision* but not *Recall* see table [4.1](#).

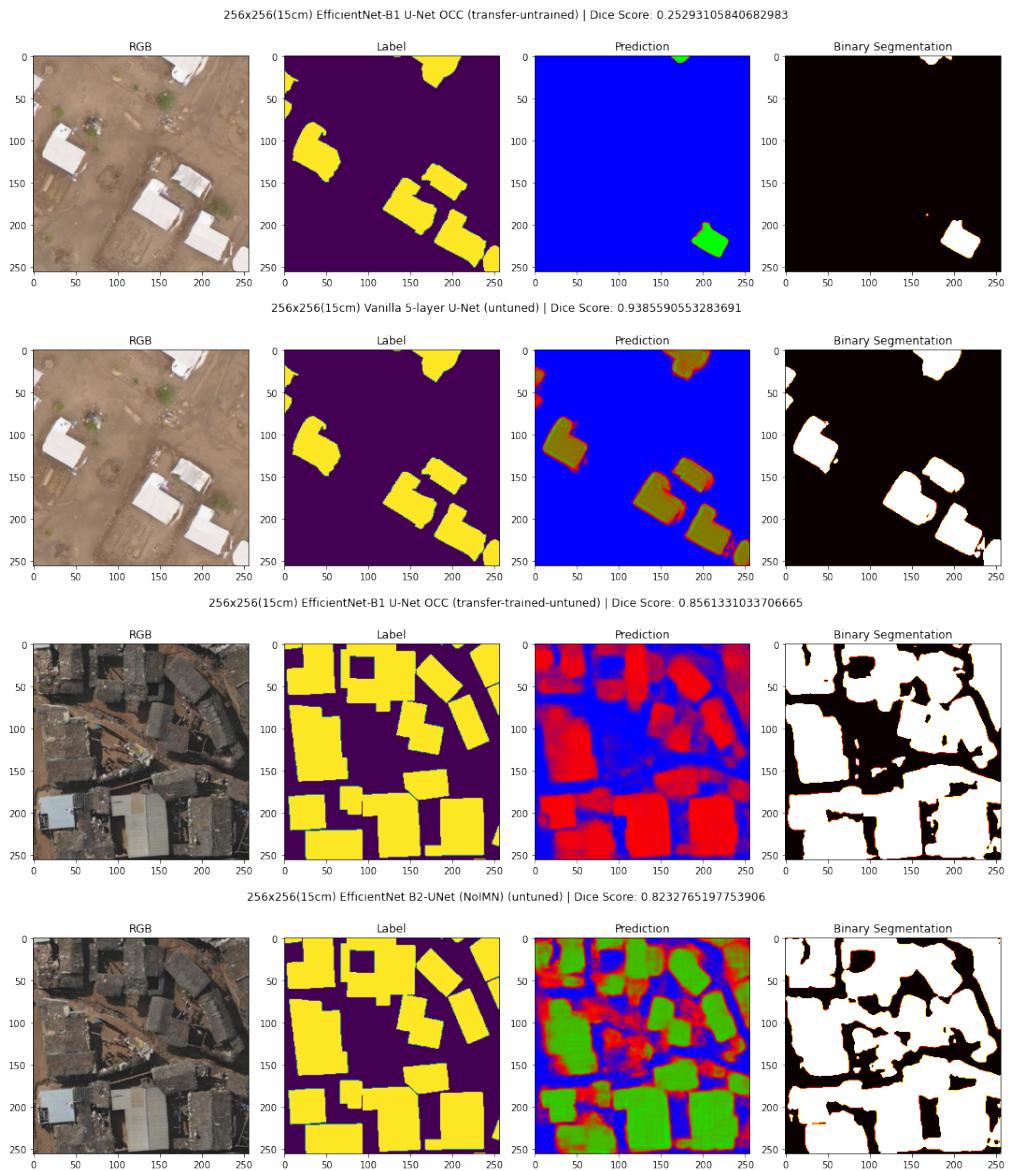


Figure 4.1: Sample of binary segmentation output of various combinations of tested architecture and experiment setup.

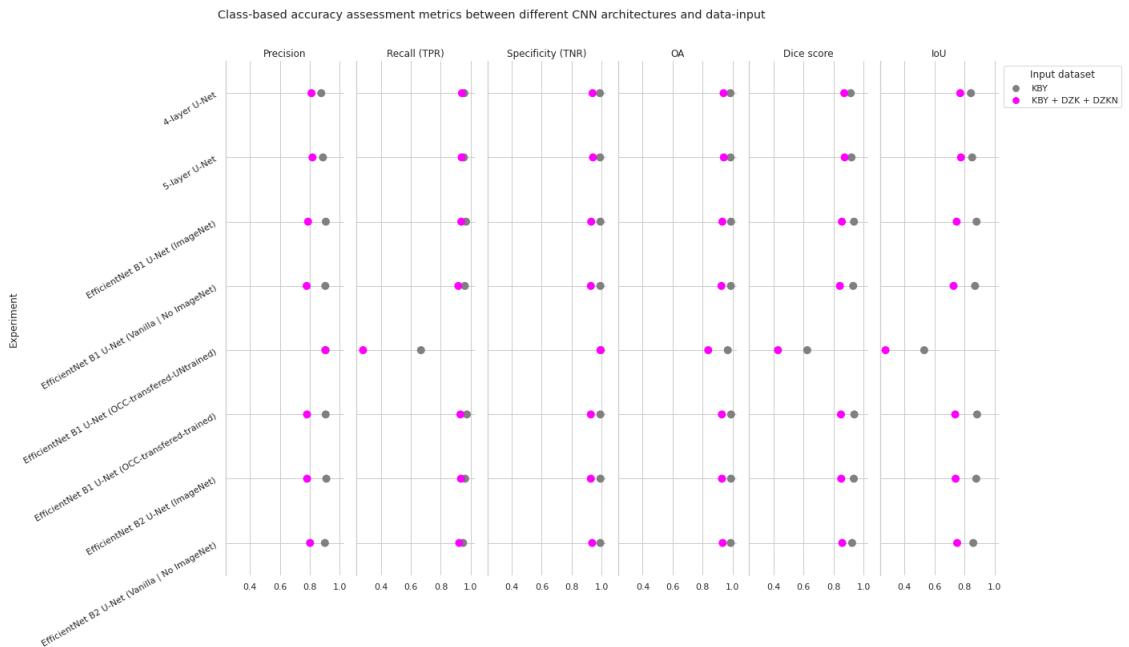


Figure 4.2: Class-based Accuracy Assesment metrics for respective CNN architectures and experiment input dataset.

The following sections compare the mean changes of *Precision* and *Recall* in the test dataset at paired CNNs' depth-wise, dataset-wise, and weight-wise change.

Depth-wise Precision and Recall change

Depth-wise Precision and Recall change				
Architecture	Initialised weights	Input dataset	Precision change	Recall change
4 to 5 layer U-Net	None	KBY	+0.011	-0.003
4 to 5 layer U-Net	None	KBY + DZK + DZKN	+0.007	-0.002
EfficientNet B1 to B2 U-Net	None	KBY	-0.003	-0.012
EfficientNet B1 to B2 U-Net	ImageNet	KBY	+0.003	-0.006
EfficientNet B1 to B2 U-Net	None	KBY + DZK + DZKN	+0.023	+0.006
EfficientNet B1 to B2 U-Net	ImageNet	KBY + DZK + DZKN	-0.006	-0.002

Table 4.1: Changes with architectures that had a depth-wise increased for each setup..

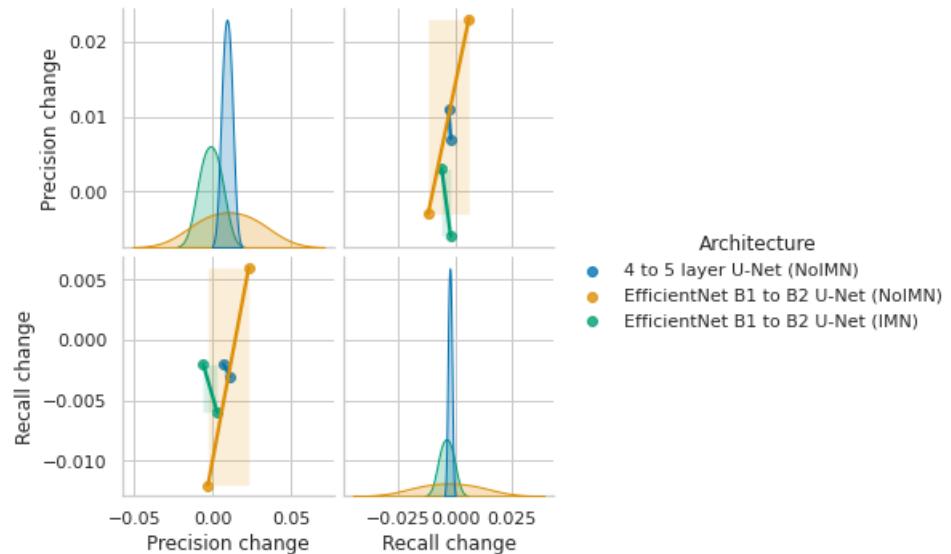


Figure 4.3: Regression plot for *Precision* and *Recall* change in relation to architectural depth-wise change.

There is a common misconception in the DL realm that deeper networks would always perform better. The comparison in *table 4.1* in a limited scope tries to address this question, the result suggests that improvements in both *Precision* and *Recall* only happened in non-weight initialised depth-increase from *EfficientNet B1* to *B2 U-Net* trained on all datasets, achieving both the highest rate of increase in both metrics. In comparison to the same architecture change and dataset input with initialised weights from ImageNet, both metrics experienced a decrease. Meanwhile in other experiment setup, no significant trends can be drawn and thus the assumption does not hold, although results might drastically differ with increase of dataset, increase in batch size, and when experimenting with much deeper architectures not available to this study due to computation constraint.

Dataset-wise Precision and Recall change

Dataset-wise (KBY to KBY + DZK + DZKN) Precision and Recall change			
Architecture	Initialised weights	Precision change	Recall change
4-layer U-Net	None	-0.065	-0.016
5 layer U-Net	None	-0.07	-0.0153
EfficientNet B1	None	-0.124	-0.044
U-Net			
EfficientNet B1	ImageNet	-0.119	-0.032
U-Net			
EfficientNet B1	OCC	-0.002	-0.387
U-Net (OCC)			
EfficientNet B1	OCC transfer-trained	-0.125	-0.043
U-Net (OCC)			
EfficientNet B2	None	-0.099	-0.026
U-Net			
EfficientNet B2	ImageNet	-0.128	-0.028
U-Net			

Table 4.2: Changes when the Dzaleka and Dzaleka North datasets were introduced to each setup.

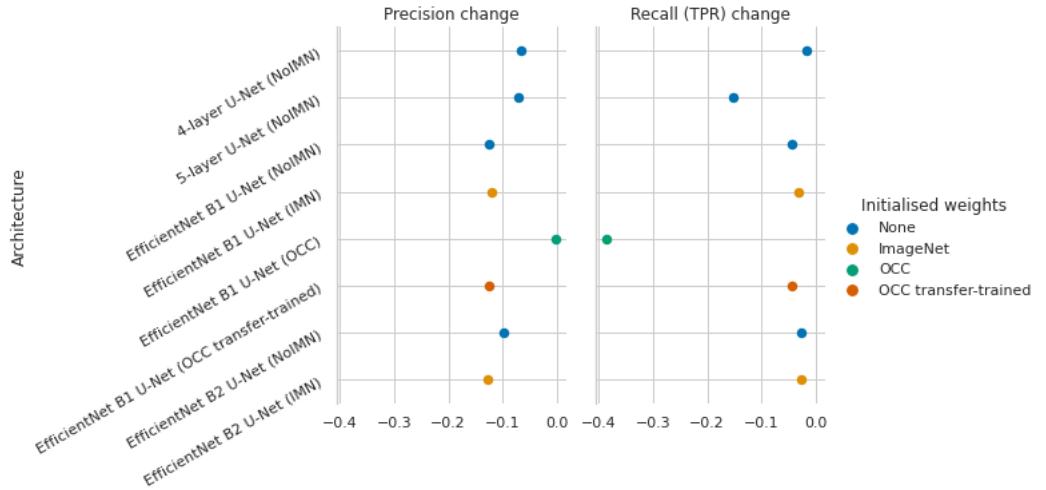


Figure 4.4: Detailed strip plot for *Precision* and *Recall* change in relation to dataset input change.

With the Kalobeyei dataset as constant, the introduction of the Dzaleka and the Dzaleka North dataset have resulted in the reduction in *Precision* and *Recall* for all architectures with any or no initialised weights. Table 4.2 might suggest that the least reduction in *Precision* came from the *EfficientNet B1 U-Net* initialised on OCC building segmentation weights. However figures 4.1 and 4.2 informs that with such poor Dice and IoU score, it reflects that the OCC competition winning network being either very confident at the the segmentation or completing missing the other buildings. Thus the *Precision* diverge from the *Recall* results and the statistics (*see figure 4.4*) suggests prediction results with high *False Negative* and therefore does not reflect overall performance. The *5-layer U-Net* had a much more corresponding between *Precision* and *Recall* which shows the metrics are least affected by the introduction of the Dzaleka camps dataset. However, it does not seem to be the case that deeper or shallower version of the architectures more or less reduction, therefore it is difficult to draw any conclusion here.

Initialised weight Precision and Recall change

Pre-initialised weights Precision and Recall change				
Architecture	Weights changed	Dataset input	Precision change	Recall change
EfficientNet B1	None to ImageNet	KBY	+0.003	+0.008
U-Net				
EfficientNet B1	None to ImageNet	KBY + DZK + DZKN	+0.009	+0.0197
U-Net				
EfficientNet B1	OCC to OCC transfer-trained	KBY	0	+0.306
U-Net (5-layer)				
EfficientNet B1	OCC to OCC transfer-trained	KBY + DZK + DZKN	-0.122	+0.649
U-Net (5-layer)				
EfficientNet B2	None to ImageNet	KBY	+0.009	+0.014
U-Net				
EfficientNet B2	None to ImageNet	KBY + DZK + DZKN	-0.02	+0.012
U-Net				

Table 4.3: Initialised weight change in available CNNs and their effects on the metrics

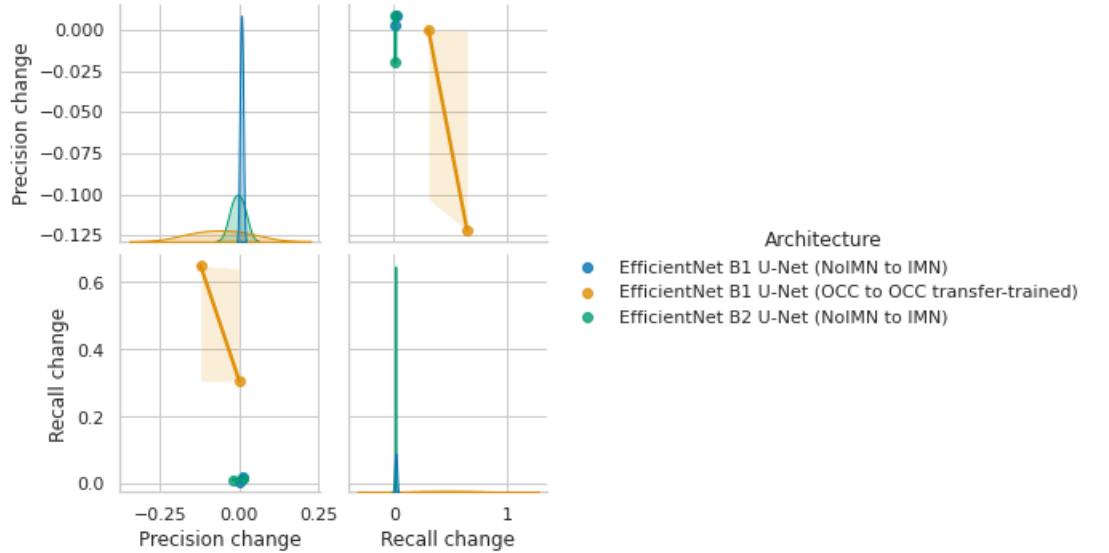


Figure 4.5: Regression plot for *Precision* and *Recall* change in relation to architectures' initialised weight change.

As mentioned previously in section 3.2 that CNNs trained on pre-initialised weights might have significant advantages in performance. Therefore, one might expect that although the OCC model suffers from low *Recall* and high *False Negativity* in it's segmentation output, perhaps further training on the weighted network would result in drastic improvement. Table 4.3 indicates that this was indeed the result, with *EfficientNet B1 U-Net OCC to OCC transfer-trained* achieving the highest *Recall* change, it however also caused the highest decrease in *Precision*, this trend is less significant in the version only trained in the Kalobeyei dataset, but the result reflects the assumption.. This suggest that transfer training from the OCC model compensated for the *False Negative* issue, the improvement in *True Positive* segmentation is not as significant. Meanwhile, the *EfficientNet B1 U-Net* initialised from ImageNet weights saw both improvement in the metrics but not the *EfficientNet B2 U-Net*.

Few assumptions could be drawn from the above results. Firstly, depth increase in architecture tend to favour improvement in *Precision* meaning that deeper networks redyce the classification of *False Positive*, meanwhile, ImageNet initalised weights in most setting tend to favour imrpovement in *Recall* which reduce *False Negative*. These are useful generalisation for future experiment and especially for when tuning any architectures. In general, introducion of the complex Dzaleka camps datasets cause all architectures to perform worst, but no significant conclusion can be drawn whether the introduction of deeper version of the architecture or initalised weights in tandem with dataset introduction would reduce the rate of *Precision* or *Recall* decrease.

Conclusion

The beginning of a DL project is a momentous task. Errors from the data pre-processing to architecture selection could be costly in both time and resources, especially in the setting of humanitarian NGOs (Private Communication, 2022). It is important for a pilot project to therefore uncover the possibilities and challenges with well-designed, small-scale yet rigorous experiments. This study presented a series of experiments which tested variations of the U-Nets on the shallow-end of the spectrum, it also explored the possibility of transfer-training from a previous competition winning neural network and how they compared against networks trained from scratch. Initially, there was an assumption that transfer-training from pre-trained networks and deeper architectures would perform better. However, results shows that the overall picture is a lot more complicated, where they may experience drastic improvement in some metrics, but not others. Furthermore, focusing on the *True Positive* centric class-based accuracy assessment metrics have shown that even performance in *Precision & Recall* does not necessarily corroborate with each other and that there's nuance relationship between their improvement, architectural depths, and pre-trained weights.

Albeit many ambiguities and questions remain, there are several key takeaways that have surfaced as a result. First, transferring a competition winning network which scored very well in the IoU metrics does not necessarily guarantee easy transfer and good performance in new environments. This

study hinted that the OCC model may have became too customised for its original dataset. Secondly, it is important to carefully and critically evaluate the segmentation output by examining more granular low-level metrics. Thirdly, rudimentary conclusion can be drawn from the depth-wise and weight-wise changes. Unfortunately, the results for how depth and weight change influence performance when complex Dzaleka camps datasets were introduced remained inconclusive. Thus, it is important that future studies could address this. However, It is speculated that this might scale differently and at a more predictable way when higher batch-size, much deeper network, and when significantly more data can be provided.

To narrow down the choices, further investigation will be needed. Scaling from the experimental framework defined for this study could be a good strategy for succession. This will allow for a well-constrained extension of this study. Additionally, baseline experiment at different hyperparameters, where the parameters of isolation are the initialised weights and architectures would be very interesting. Nevertheless, scaling will only be possible with significantly more computational and data resources.

In conclusion, this study provided a pilot baseline understanding of how HOT could begin their open-sourced AI-assisted mapping initiative. The study defined a rigorous experiment and assessment methods which is scalable. With increasing data availability from the OpenAerialMap initiative, this study hope to have made a foundational contribution to a beginning of a much larger study. Perhaps a prequel to which will culminate into an open-sourced AI-assisted mapping tool that could help reduce the geospatial data-inequality in the HOT and OpenStreetMap ecosystem.

Bibliography

Appendix

Adam optimiser

Algorithm 1: Adam, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t .

Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0 : Initial parameter vector
 $m_0 \leftarrow 0$ (Initialize 1st moment vector)
 $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
 $t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
 $t \leftarrow t + 1$
 $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
 $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
 $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
 $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
 $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
 $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 6.1: The algorithm of Adam (Kingma & Ba., 2017).

EfficientNet

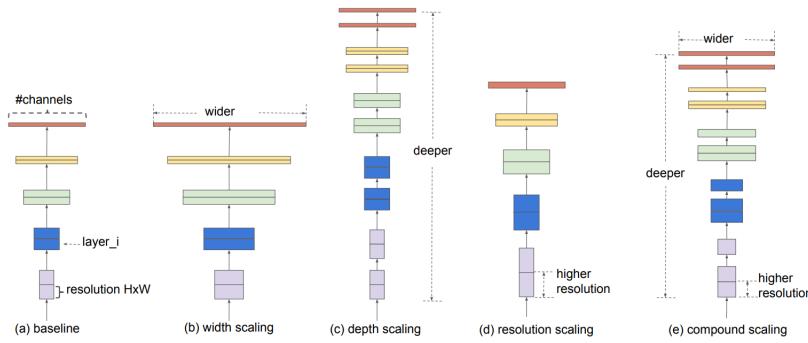


Figure 6.2: Compound scaling of the EfficientNet (Tan & Le, 2020)