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

CHAN, Yan-Chak Christopher

Under the supervision of

Prof. Dr. Hannes Taubenöck [1] [2]

Matthias Weigand [1]

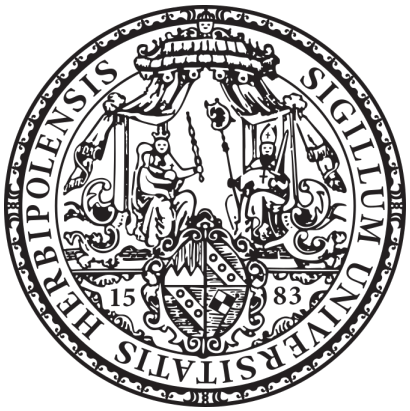
Emran Alchikh Alnajar [3]

Master thesis submitted for the degree of

Master of Science (MSc.)

in

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



Faculty of Philosophy

Julius-Maximilians-Universität Würzburg

[1]Department of Geo-Risks and Civil Security, German Remote Sensing Data Center (DFD), German Aerospace Center (DLR)

[2]Department of Remote Sensing, Institute of Geography and Geology, Julius-Maximilians-Universität Würzburg

[3]Humanitarian OpenStreetMap Team

Abstract

Refugee camps and informal settlements provide accommodation to some of the most vulnerable populations, with many of them located in Sub-Saharan Africa. Many of these settlements lack up-to-date geoinformation that we take for granted in developed world. Having up-to-date maps on their dimension, spatial layout is important. They are essential tools for assisting administration tasks such as crisis intervention, infrastructure development, and population estimates which encourage economic productivity. In the OpenStreetMap ecosystem, there is a disparity between built-up being digitised in the developed and the developing areas. This data inequality are results of multiple reasons ranging from a lack of commercial interest to knowledge gaps in data contributors and such disparity can be reduced with the help of assisted mapping technology. Very High Resolution remote sensing imagery and Machine Learning based methods can exploit the textural, spectral, and morphological characteristics and are commonly used to extract information from these complex environments. In particular recent advances in Deep Learning based Computer Vision have achieved significant results. This study is connected to a larger initiative to open-source the AI assisted mapping platform in the current Humanitarian OpenStreetMap Team's ecosystem, to investigate the capabilities of applying Deep Learning for building footprint delineation in refugee camps based on open-data Unmaned Aerial Vehicle (UAV) imagery from partner organisation OpeAerialMap.

The objective of this study is to test the U-Net and several variations of the architectures' performance for building footprint segmentation, The performance of the different Deep Learning models on datasets of various complexity were collected. A comparison of the models' responses using class-based accuracy assessments metrics allows detail evaluation into how the different architectures and experiment setup respond to data quality.

Given the computation and resources constraint of this project, the result suggests that increase in architectural depths corresponds with increase in precision. Models that were initialised on pre-trained weights from ImageNet could reduce recall. Lastly, to our surprise, the transferability of a competition winning network trained on similar resolution but on formal building performs worse than many models trained from scratch.

This study showcased the ability to use Deep Learning semantic segmentation to perform building footprint delineation in complex humanitarian applications. Having increased access to open-data Very High Resolution UAV imagery from the OpenAerialMap initiative is an advantage to building AI-assisted humanitarian mapping. The study demonstrated a careful and rigourous approach to model evaluation. Yet, the variation of the study results not only emphasised the complexity of Deep Learning based methods, but also indicate the direction for further investigation that would be justifiable when further resources becomes available.

0.1 Forewords and Acknowledgements

Service to the less-advantaged! This is the value taught by my parents since I was small, and have since became the main motivator for the completion of this Masterarbeit. I have been extremely fortunate to have been able to carry out such an important work in partnership with the Humanitarian OpenStreetMap Team and the German Aerospace Center (DLR). This project came about as a hope to do something scientifically rigour, yet practically impactful. Through providing a technical, careful study, I, the supervisors, and partners hope that this work can provide direction and minimise future mishaps of the open-sourced AI-assited mapping solution. Working on this project have been a journey of surprises, long nights, new friendships, and adventures, of which I gained bountiful from. Ultimately culminating into the work being accepted by the Proceedings of the Academic Track at State of the Map 2022 (*Chan et al., 2022*). All of this could not have been possible without the generous facilitation from Emran Alchikh Alnajar and the passionate yet pedantic Matthias Weigand who maintained high standards and scientific rigour while motivating me to focus on the light at the end of the tunnel. I would also like to thank my past classmates, collagues, and lecturers, who tolerated and kept patience with me at times of desparation and frustration. Lastly, I must extend my appreciation to my parents Lucia Leung and Savio Chan. Without their support, I would have never had the resources and opportunities to get to where I am now and complete this work.

0.2 Declaration of Independent Work

I confirm that all this work is my own, and that I have

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)

Contents

List of Figures

List of Tables

0.3 Abbreviations

- SDG: Sustainable Development Goals
- HOT: Humanitarian OpenStreetMap Team
- CV: Computer Vision
- CNN: Convolutional Neural Network
- SGD: Stochastic Gradient Descent
- OCC: Open-Cities-AI-Challenge
- OA: Overall Accuracy
- IoU: Intersection-over-Union

1 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 and 2050, most of which will be in Asia and Africa (UN DESA., 2018). When population growth outpace development, informal settlemnt 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. In particular, refugee camps and their development are often dependent on international aid and humanitarian efforts, resuting in unique urban morhopology where parts of older refugee camps resembles the chaotic characteristics of informal settlement while newer extension received careful planning.

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, only around 1.4 million out of 26.4 million refugees have access to third country solution

between 2016 and 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 become "Secondary Cities" and therefore suffer 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. Reolcation 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 soverign-state. Recent studies have suggested that local integration often have a net positive economic impact on the surrounding region (Alix-Garcia et al., 2018, Rummery A., 2019, IFC., 2018).

Since the 2000, the United Nations have codified a set of global development goals which the member states commited to, development projects are therefore encouraged to align their goals to achieve such global development goals. 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 as a successor to the 2015 Millenium Development Goals (herein MDG) (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, appropiriate new technology and financial*

services, including microfinance

- *Target 1.b: Create sound policy frameworks at the national, regional 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-humanitian context. Although we have seen an overall net increase in the amount of buildings mapped on the OpenStreetMap (herein OSM) 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*).

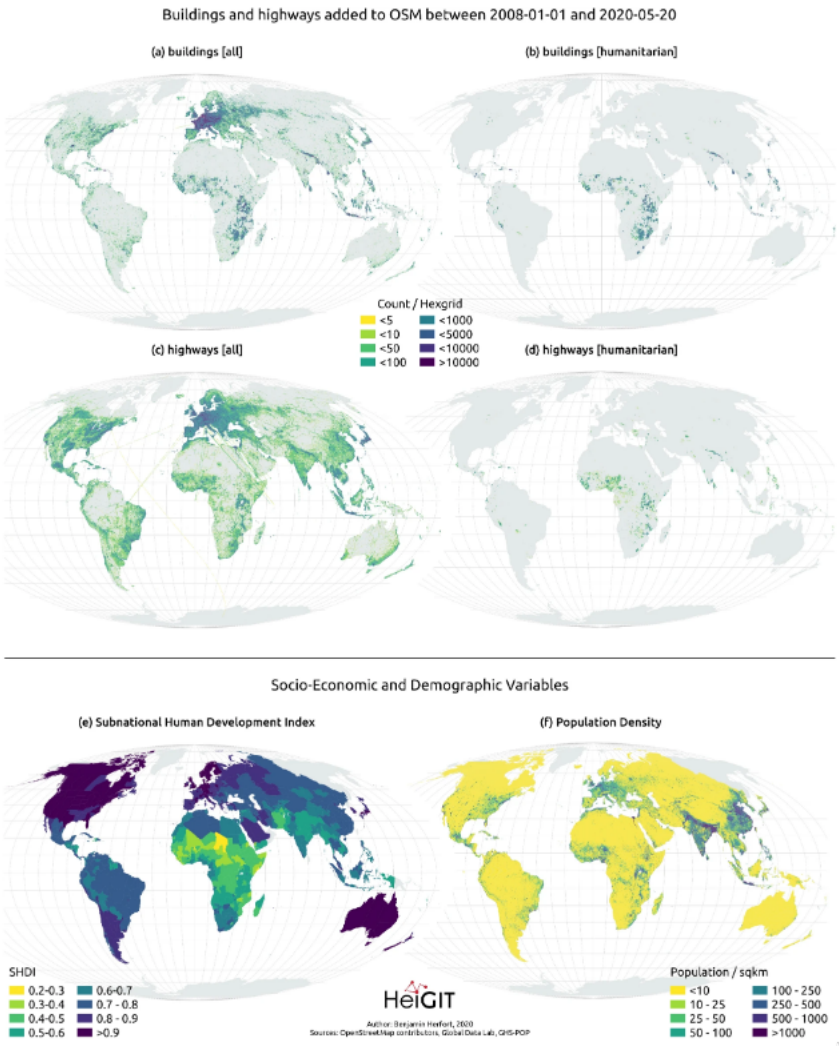


Figure 1.1: Data contribution of buildings and highway in all and humanitarian settings within OSM (Herfort et al., 2021)

Part of the reason for this could be traced to both the knowledge of volunteer contributors and commercial interest, for detailed discussion, see Anderson et al., 2019, Veselovsky et al., 2021 and Yang et al., 2016. As the topic and data provider of this project, the Humanitarian 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). HOT would like to make use of the recent advancement of Deep Learning in the geospatial field (e.g. Herfort et al., 2019, Kuffer et al., 2016, Wurm et al., 2021, Quinn et al., 2018) to develop an open-sourced, open-data AI-assisted mapping solution to reduce the geospatial data inequality in the OSM ecosystem. The result of this work will form part of the pilot study to trial the use of Unmaned Aerial Vehicle (herein UAV) imagery and existing labelled data in the segmentation of two major refugee camps of East Africa.

2 Literature Review

2.1 Remote Sensing of Informal Settlements

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 enviornments 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 opposed to the more common spectral based index approach used in ecological or enviornmental 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). The use of remote sensing derived data for socio-economic proxies have been able to compensate for traditional sources such as temporally infrequent census (e.g. Watmough et al., 2012, Watmough et al., 2015, Watmough et al., 2019) Census especially conducted in developing areas also falls short in capturing socio-spatial patterns, potentially over-

looking others socio-economic determinants such as access to amenities and infrastrcture, adequate pro-poor policy development hinges on the availablity 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 socio-economic context and urban morphology, ssensing 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 applications of deep learning in remote sensing have therefore been in the urban domain (Ma et al., 2019).

2.2 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.

2.2.1 Computer Vision and a brief review of Convolutional Neural Networks

Computer Vision is a practice of extracting information from digitised imagery, first applied in robotics, it has become an interdisciplinary field of study utilised by medicine, biologist, and remote sensing scientists alike (Rosenfeld A., 1988, Szeliski R., 2010).

The field of CV traditionally used mathematical operations that are applied on the imagery represented as multi-dimensional arrays. These include point-and-local operations, statistical computation, geometric operations, transformations, and extraction of geometric entities, for detail discussion see Rosenfeld A. (1988). The following decades saw CV operations developed into more specialised applications. For example, edge detectors (e.g. Sobel, Prewitt, Marr-Hildreth) or Grey-Level Co-occurence Matrix kernel (e.g. Haralich Texture) (e.g. Kuffer et al., 2014, Kuffer et al., 2016, Wurm et al., 2017) (Pal & Pal, 1993, Blaschke T., 2010, Blaschke et al., 2014). CV based segmentation experienced an akin to a Kuhnian paradigm shift (Kuhn T., 1962) when AlexNet, a Convolutional Neural Network trained on a GPU (Krizhevsky et al., 2012) won the preeminent large-scale ImageNet CV challenge (Deng et al., 2009), it reinvigorated the use of multi-layered neural network in CV tasks (LeCun et al., 2015, Bengio et al., 2017). 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 applicated 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.

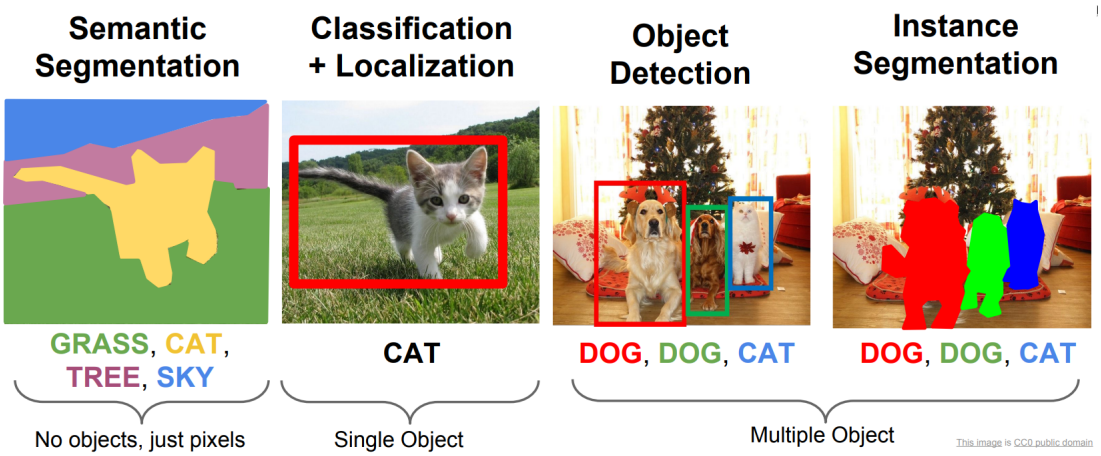


Figure 2.1: The four main types of Computer Vision tasks (Stanford University, 2022)

The purpose of semantic segmentation is to assign a named (semantic) classification to each and every single pixel of the input image (Rosenfeld A., 1988, Szeliski R., 2010). 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 classication 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.

2.2.2 Deep Learning and Convolutional Neural Networks

The closest resemblance to modern methods first appeared around the 1960s, with the first CNNs appearing around 1980s. However, DL based methodologies went in and out of popularity driven partly by lack of computaional resources and partly the non-linearity of the field's progression (i.e. the parts did not come together at the right

time.). Causing the "AI winter" between 1970s to 1990s (Schmidhuber J., 2014). Prior to the resurgence of popularity in DL, the set of methodology now associated with neural networks 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 (*see equation 2.1*). 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:

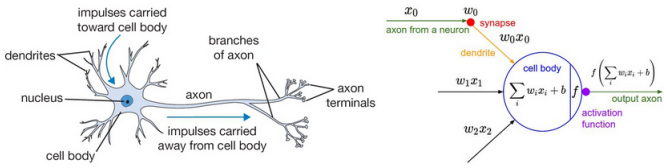


Figure 2.2: Schematic analogy diagram between a biological neuron and an artificial perceptron (Fumo D., 2017).

$$f(\sum_{i...n} w_i x_i + b) \tag{2.1}$$

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

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.

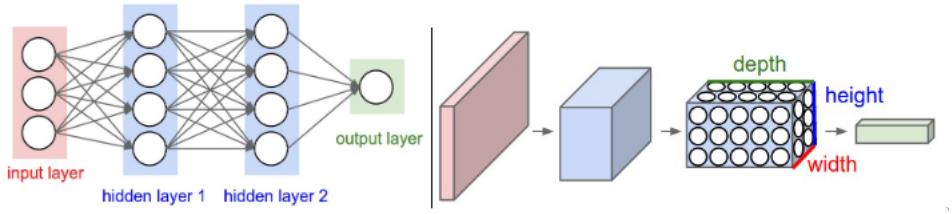


Figure 2.3: Schematic diagram of a CNN (Stanford University, 2022).

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 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 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: 2.2.2*.

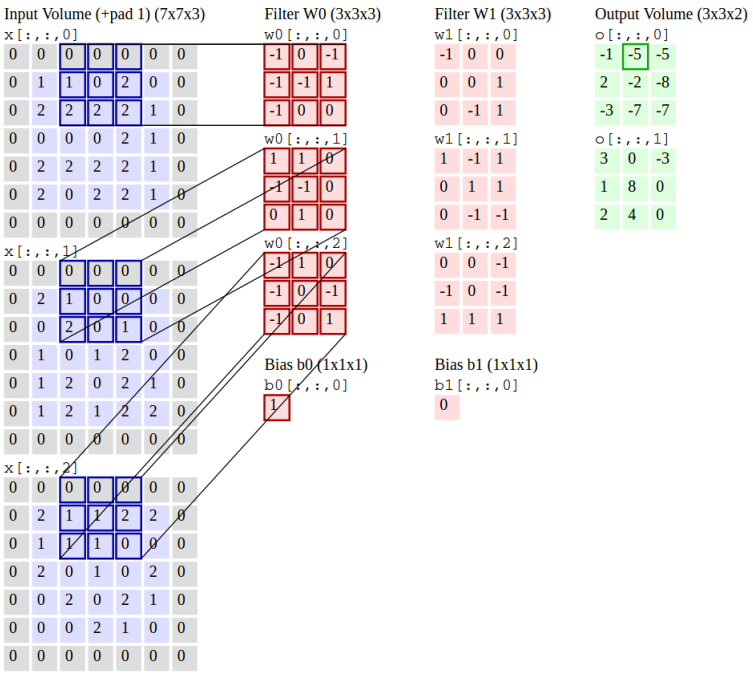


Figure 2.4: 3 x 3 Convolution (Stanford University, 2022).

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

- Where:
  - N = Batch Size
  - C = Channels
  - k = Kernel Size

(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 downsampling 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*).

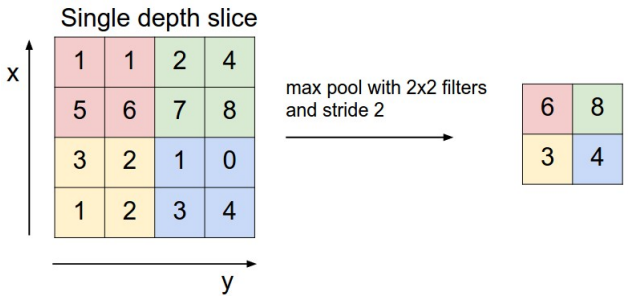


Figure 2.5: Max pooling (Stanford University, 2022).

Optimiser and the Binary Cross Entropy Loss function

With each batch of data (aka. a complete pass) is ingested through the neural network, the ouput is then compared against the validation result for error calculation. The summed average of loss defines the cost function landscape from which the error value is calculated against. From which the score is penalised when prediction is incorrect and rewaedd if otherwise. The backpropagation finds the sensitivity of the cost function to each weights and biases through the chain rule. This enables 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: 2.2.2*.

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

- Where:
  - N = Batch Size

- $l(x, y)$ = loss(Probability, Binary Classification)
- l_n = 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 experiement 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 become the standard pick of optimiser since.

Backpropagation and the chain rule

In order for a neural network to improve, the weights and biases are changed accordingly to minimise 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.

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

- 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)

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

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

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. Althought 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 3.5.

3 Data and Methodologies

3.1 Study Areas of Interest

3.1.1 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 Somali 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 3.1*). 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 improverished Turkana county significantly. However, challenges remain in integration into the wider Kenyan economy.

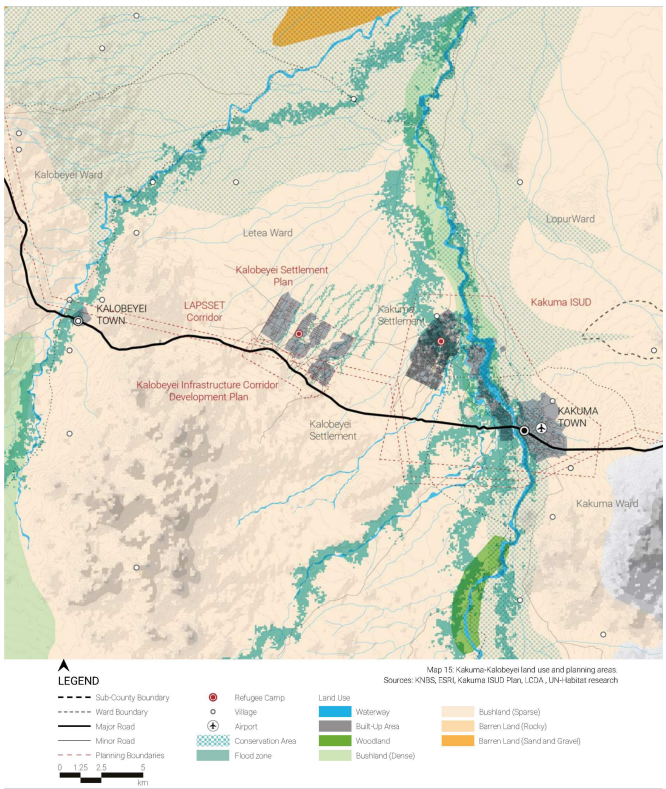


Figure 3.1: The Kakuma-Kalobeyei land use and planning areas (UN-HABITAT, 2018)

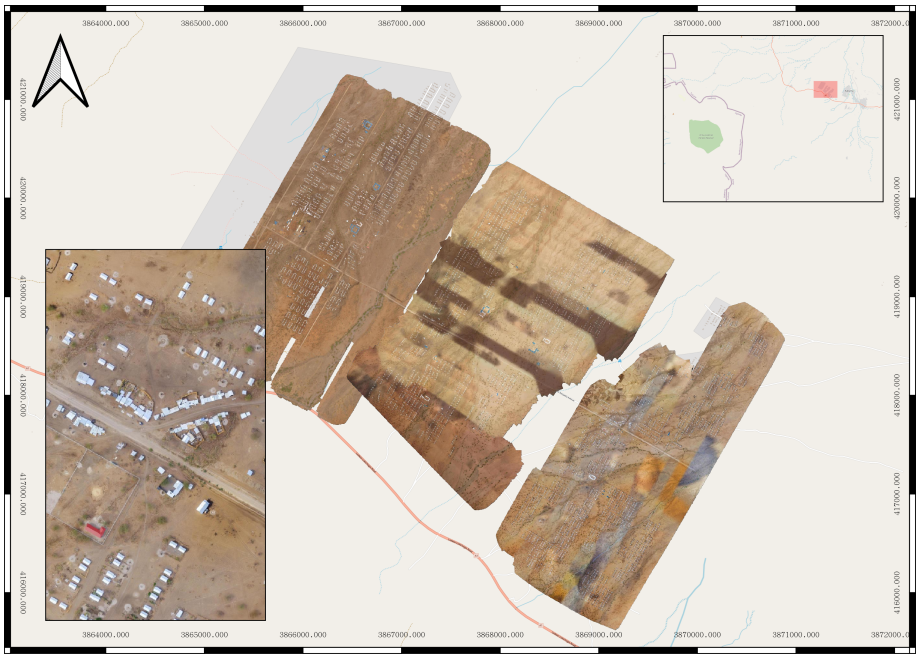


Figure 3.2: RGB UAV imagery of the Kalobeyei settlements in rural Turkana from OpenAerialMap

3.2 Dzaleka, Dowa, Malawi

Originally an infamous prison camp under the Banda's Malawi Congree 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, Kavalo 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 inconstruction as of March 2021 (Gross G., 2021 & UNHCR, 2021) (*see figure 3.3*). For the rest of this report, reference to the datasets of Kalobeyei, Dzaleka, and Dzaleka North will be denoted as KBY, DZK, and DZKN respectively.

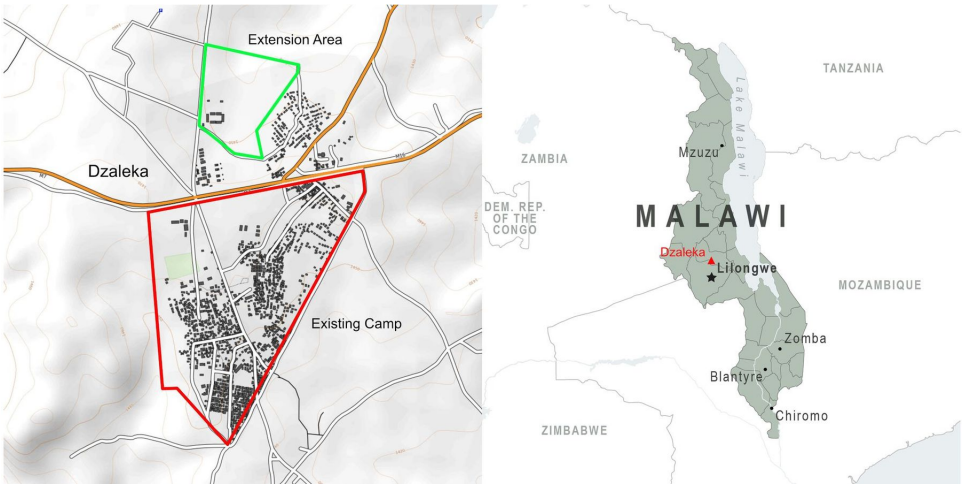


Figure 3.3: The main Dzaleka Refugee Camp and the Katubza extension plan (Dzaleka North) designed by Urban Design Advisor to the UNHCR Werner Schnellenberg (Gross G., 2021).

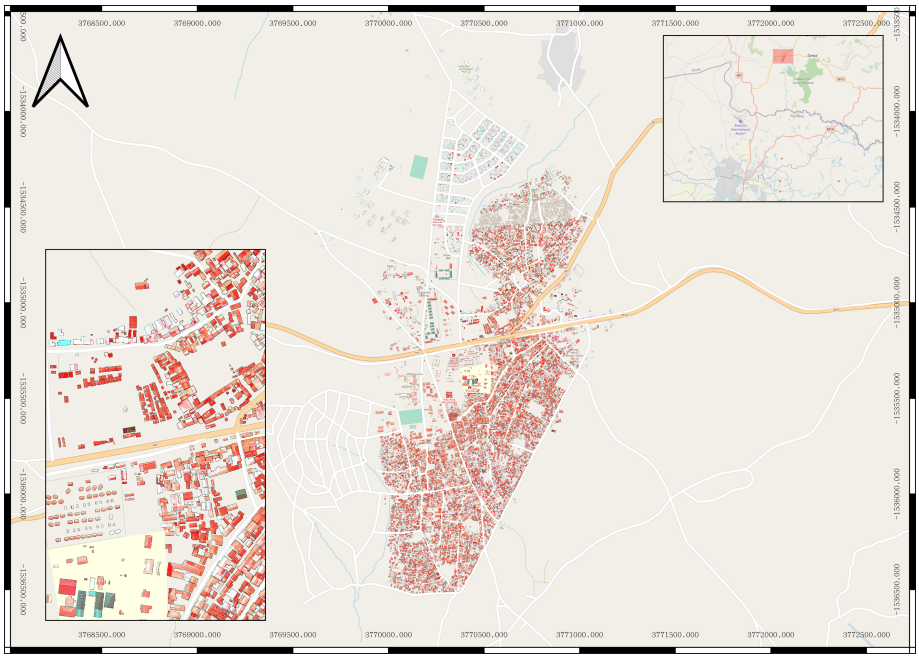


Figure 3.4: Digitised rooftop of the Dzaleka and Dzaleka North camps by HOT volunteers

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).

3.3 Data

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 begun collecting labels specifically digitised on the UAV imagery of Dzaleka and Dzaleka North camps *see figure 3.4*. 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 3.5* may have been missed.

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

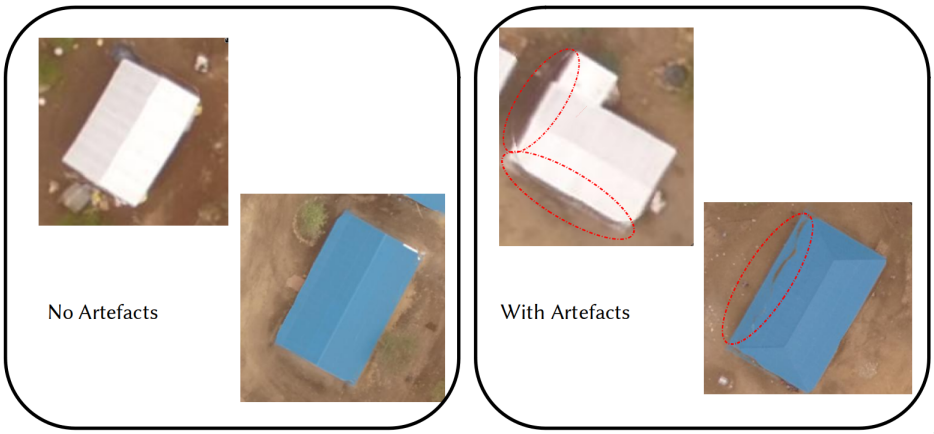


Figure 3.5: Motion artefacts unique to UAV imagery

3.3.1 Raster pre-processing

The UAV imagery from OpenAerialMap 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, as normalised images enable easier training and prevents weight explosion (Harrison K., 2020). 2-step normalisation were performed on each band for each UAV imagery to ensure the distribution of value is preserved (Gonzalez & Woods., 2002).

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*).

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

- Where:
  - p_z = z-score normalised pixel value
  - p = Original pixel value
  - μ = Mean value of pixel
  - σ = Standard Deviation of pixel

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

- Where:
  - p_{8_bit} = Pixel output normalised between 0 and 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 set were split at a ratio of 60, 30, and 10 % respectively. With augemntation applied, this increased the available data to Train n = 18242, Validation n = 3909, and Test n = 435. (*see figure 3.7 & see table* **??**)

3.3.2 Data Augmentation

Data augmentation one of the most crucial step in training a robust neural-network and reduce generalisation error (Bengio et al., 2017, Stevens et al., 2020). 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 throught 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 3.6*) (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...

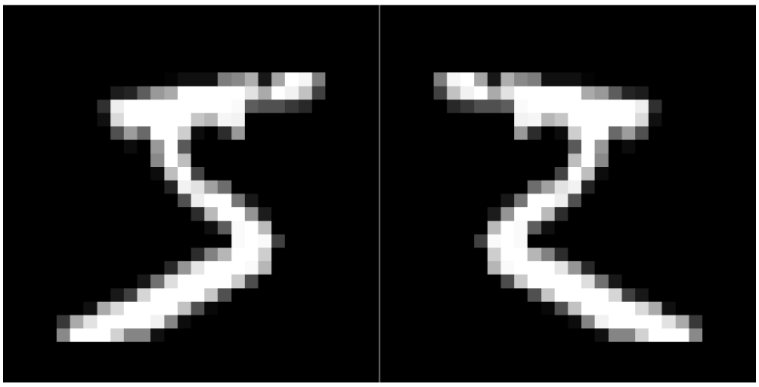


Figure 3.6: Perhaps geometric augmentation of horizontal flipping shall not be applied on the MNIST number of 5

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

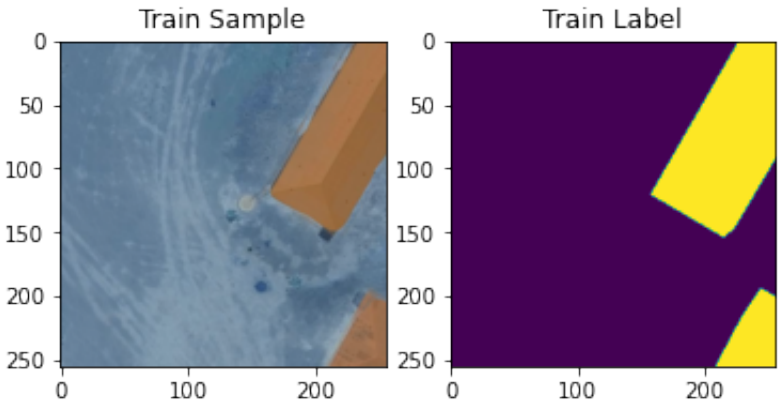


Figure 3.7: An example of Inverse RGB augmentation applied to the Train dataset.

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

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

Pre-trained weights and transfer-learning

Pre-trained networks are neural networks that were already trained and optimised for a particular dataset. Therefore, the models' adjustable parameters have already learnt the basic features such as edges and shades etc. Training built upon a pre-trained network therefore could potentially reduce variation in the training results (Bengio et al., 2017), this process is known as transfer-learning. By comparing architectures initialised randomly and on pre-trained weights, this study investigates the affect of weight-initialised, transfer-training of different datasets.

Fortunately for recent DL pracitioner, many of the classical and sometimes novel architectures have already been trained on the large-scale CV datasets (e.g. ImageNet, CIFAR) and have been made available to the community. This has made such models which often require large computational resources available to resources constrained

projects.

3.4 Research Questions and experiment design

To provide HOT with objective, clear results that could provide evidence-based direction for future research, clear guiding research questions and careful experiment desgins were required. In order to train a model which performs well on UAV imagery, the motion artefact was a signficant feature for the models 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). Numerous studies have showcased 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), therefore one would expect the transfer training of CNN pre-trained on any dataset would provide it with an advantage. Thus, it is important that this study also test the effects of the CNNs response when initialised with weights from ImageNet and the OCC building segmentation model.

1. RQ1: Do state-of-the-art models allow for accurate detection of buildings from UAV data in refugee camps?

2. RQ2: 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 environment such as heterogeneous urban morphologies, roofing materials, and UAV drone artefacts affect result?

3. RQ3: How do existing models pre-trained on classical CV datasets' and/or building datasets' response when applied to the setting of refugee camps?

The selection of 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 3.8* 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 3.5.1*).

Figure 3.8: Collections of diverse and heterogeneous rooftops from the Kalobeyei, Dzaleka, and Dzaleka North datasets.

3.5 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.

3.5.1 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 3.9*) 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 with concatenated skipped-connections 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).

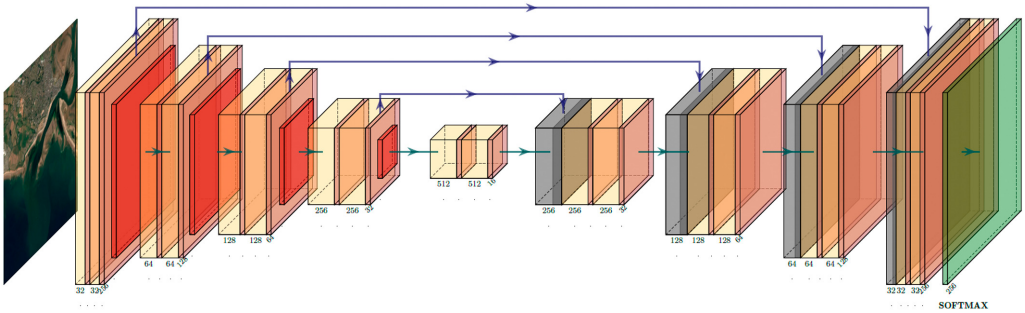


Figure 3.9: The Encoder-Decoder U-Net architecture (Ronneberger et al., 2015, Seale et al., 2022)

Changing the encoder architecture and the EfficientNet family

The ability to switch out the encoder structure allows the DL practioner 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 experiments in this study were carried out using the high-level PyTorch API Segmentation-Model-PyTorch created by

Yakubovyskiyl P. (2021). Who was also the winner of the OCC challenge for UAV building segmentation. This study will compare and contrast the unchanged 4-layer and 5-layer U-Nets architectures with the U-Nets with changed encoders. The changed encoders were based on the EfficientNet family. There are three reasons for this decision. Firstly, at one of the last stage of the OCC compeition winning network, the EfficeintNet 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 represntation of generalised state-of-the-art architectures that have been tested and performed well in classical CV datasets (*see figure 3.10*) (Tan & Le, 2020). In essence, these are sets of experiments that mix and match old and new architectural design.

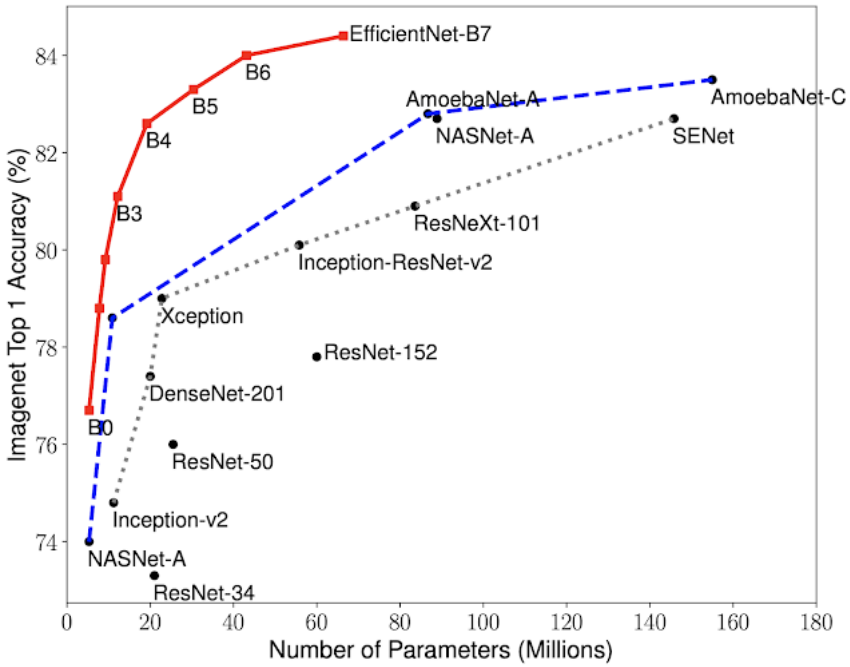


Figure 3.10: EfficientNet family Top 1% Accuracy Assessment in ImageNet (Tan & Le, 2020).

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	Initalised weights	Trainable parameters	Batch-size (8 GB 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

3.6 Hyperparameters and baseline model perforamce

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 3.6*). One of the most difficult processes in DL is finding the correct hyperparameters value that cause the model to neither overfit nor underfit the dataset. The strategies and options are often overwhelming, therefore this study does not concern itself with changing or tuning the hyperparameters for all the models but rather withholding them from changes so that a comprehensive controlled experiment can be performed. This allows for the **baseline** performance of each architecture setup (*see table 3.5.1*) to be identified. This will provide a clear picture of eachs'

feasibility, uncover challenges and potentials, and insight into where 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 *EfficientNet B1 U-Net OCC* with a 5-layer U-Net decoder
Optimiser	Adam *see appendix 6.1*	Adaptive Momentum Estimator developed by Kingma & Ba. (2017)
Loss function	Binary Cross Entropy	... *see equation 2.2.2*
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 experiment in this study.

3.7 Accuracy Assessment

Detail and scrutable accuracy assessments are fundamental towards any ML 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 metric, this will provide a much more granular nature of accuracy assessment in the findings of section **??**. In general, accuracy assessment in remote sensing can be divided into 2 categories: 1. Positional Accuracy & 2. Thematic Accuracy. Of which, Thematic Accuracy deals with the labels or attributes accuracy (Congalton & Green, 2019 & Bolstad P., 2019) which will be the focus of assessment. Here the study differentiate between 2 groups of class-based accuracy assessments. With binary classification metrics being more granular, they focus on assessing relevant or irrelevant classifications. Meanwhile, the statistical metrics are more triturated but generalised, they are often a statistical combination of the binary classification metrics.

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 3.11*)

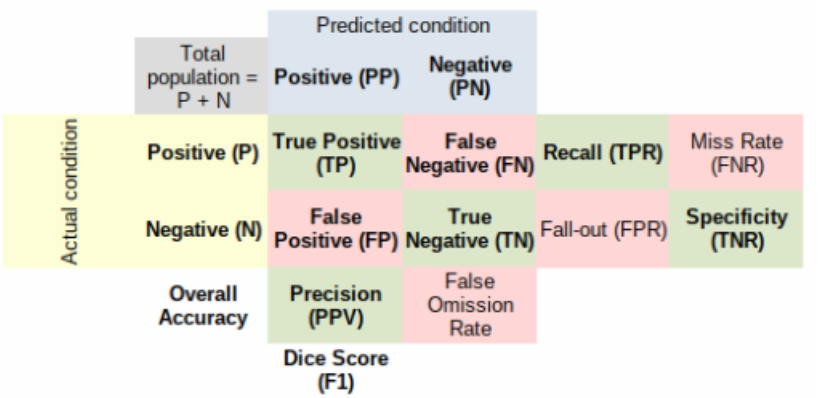


Figure 3.11: The Confusion Matrix

3.7.1 Binary classification metrics

The building blocks of any binary classification and statistical analysis metrics described in the following section are based upon the counting of segmented pixels. The segmentation output will cover the whole imagery with each pixel assigned to be *True Positive*, *False Positive*, *True Negative*, or *False Negative*. A schematic theoretical example is described below:

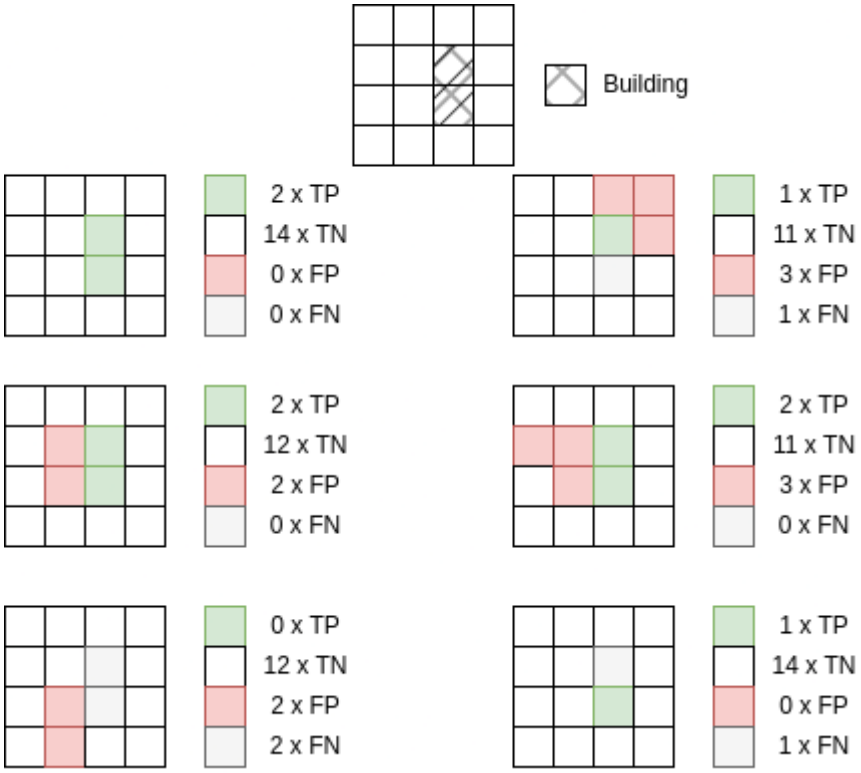


Figure 3.12: Examples of theoretical binary building classification.

Precision and **Recall**, aka. Positivie-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:

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

Meanwhile, **Recall** measures the correctly predicted Positive class $True\ Positive$ against both the correct and incorrect predicton 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:

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

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:

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

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 seg-

mentation classifier, emphasise will be placed on maximising **Precision** and **Recall**.

3.7.2 Statistical analysis metrics

The following are statistical accuracy assessment metrics, where they often encompass the binary classification accuracy assessment metrics of the above section, Thus, although the statistical analysis metrics provide a more generalised overview, they often omit detail responses only available with simpler metrics. 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.

$$Overall\ Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.6}$$

The **Overall Accuracy** (herein OA) gives an easy to implement, general but an aggregated answer of how well classification is doing which omit the details. The metrics suffers when inbalance count of multiple-classes are involved.

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

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

$$IoU = \frac{A \cap B}{A \cup B} = \frac{True\ Positive}{True\ Positive + False\ Positive + False\ Negative} \tag{3.8}$$

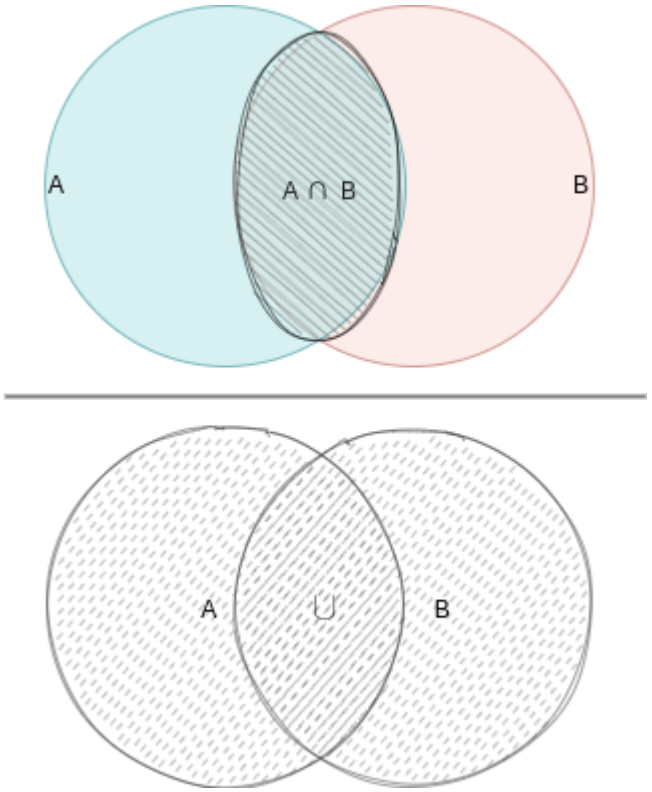


Figure 3.13: Schematic diagram of Intersection-over-Union

One of the most commonly used metrics as an assessment in Deep Learning Computer Vision competiton. The Intersection-over-Union (IoU) aka. Jaccard Index is a geometric based accuraccy assessment. The metrics calculate geometrically the area in common between the prediced and actual labels, quantifying the similarities between the two sets. It is mathematically very similar and positively correlates with the **Dice Score** but places emphasis to the false classification. The IoU is a metrics easy to conceptualise and compare against other ML results due to its established prevalence, a good metrics for comparison against the OCC competition results.

3.7.3 Project workflow

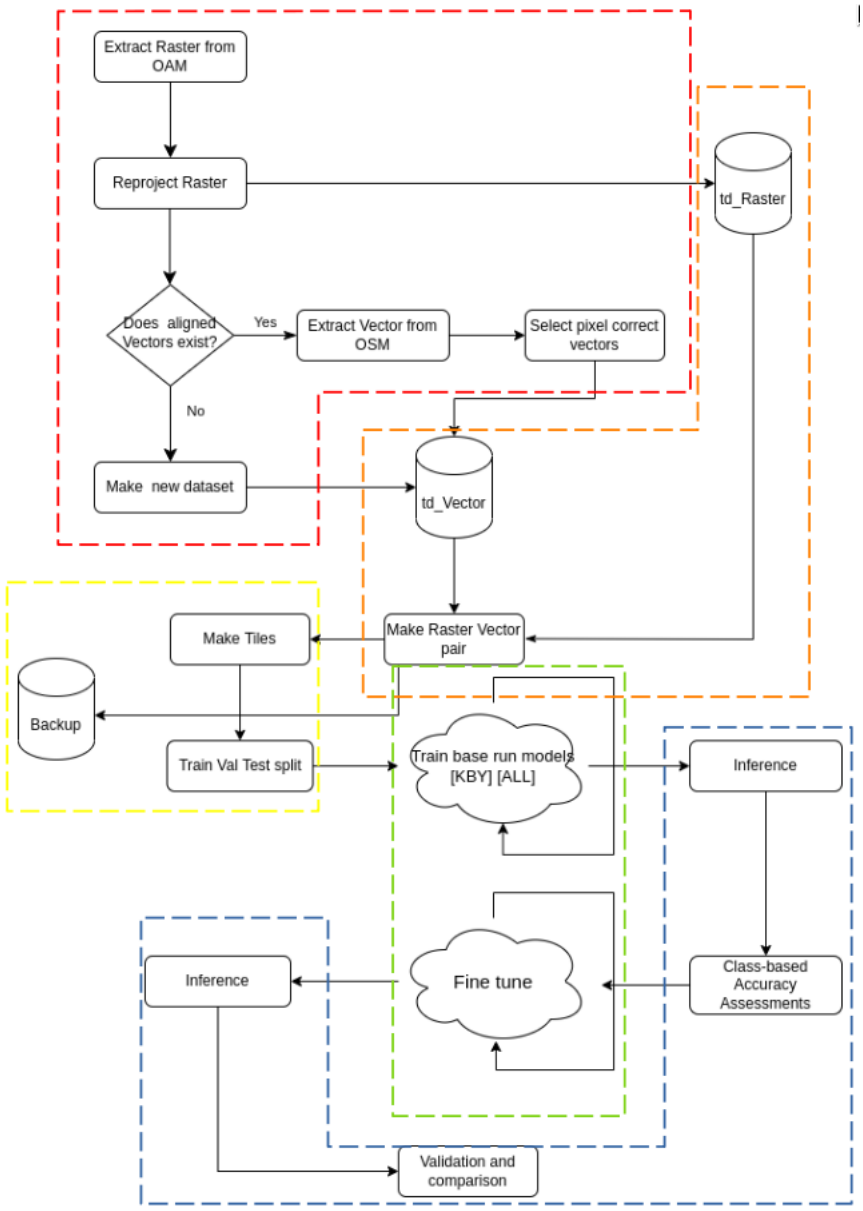


Figure 3.14: Project workflow

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 3.14 & 3.15*).

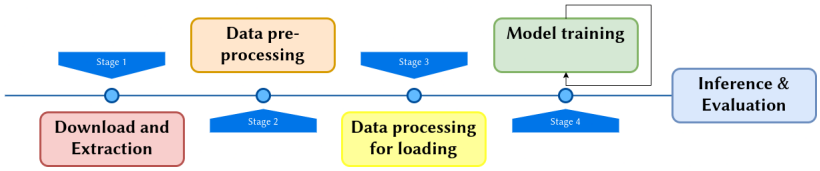


Figure 3.15: Simplified 5 steps project workflow with reference to 3.14

4 Findings and Discussion

4.1 Findings

A total of 5 different architectures with either no initalised weights or initalised weights pre-trained from ImageNet or OCC building segmentation models have been trained. For each experiment setup, there were 2 sets of dataset input (KBY and KBY + DZK + DZKN), for details (*see table 3.5.1*). Producing 16 sets of trained CNN and associated class-based accuracy assessments (*see figure 4.2*), for the experimental results where the plot and the following analysis are derived from, (*see Appendix 6.0.3*). On the whole, there were both expected and unexpected results. A reduction in every metric were observed in every single experiment setup when the more complex Dzaleka camps datasets were introduced (*see table 4.4*). 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 transfered-untrained *EfficientNet B1 U-Net OCC* model. In contrast, the performance in differences is a lot more visible with the introduction of the Dzaleka and Dzaleka North datasets.

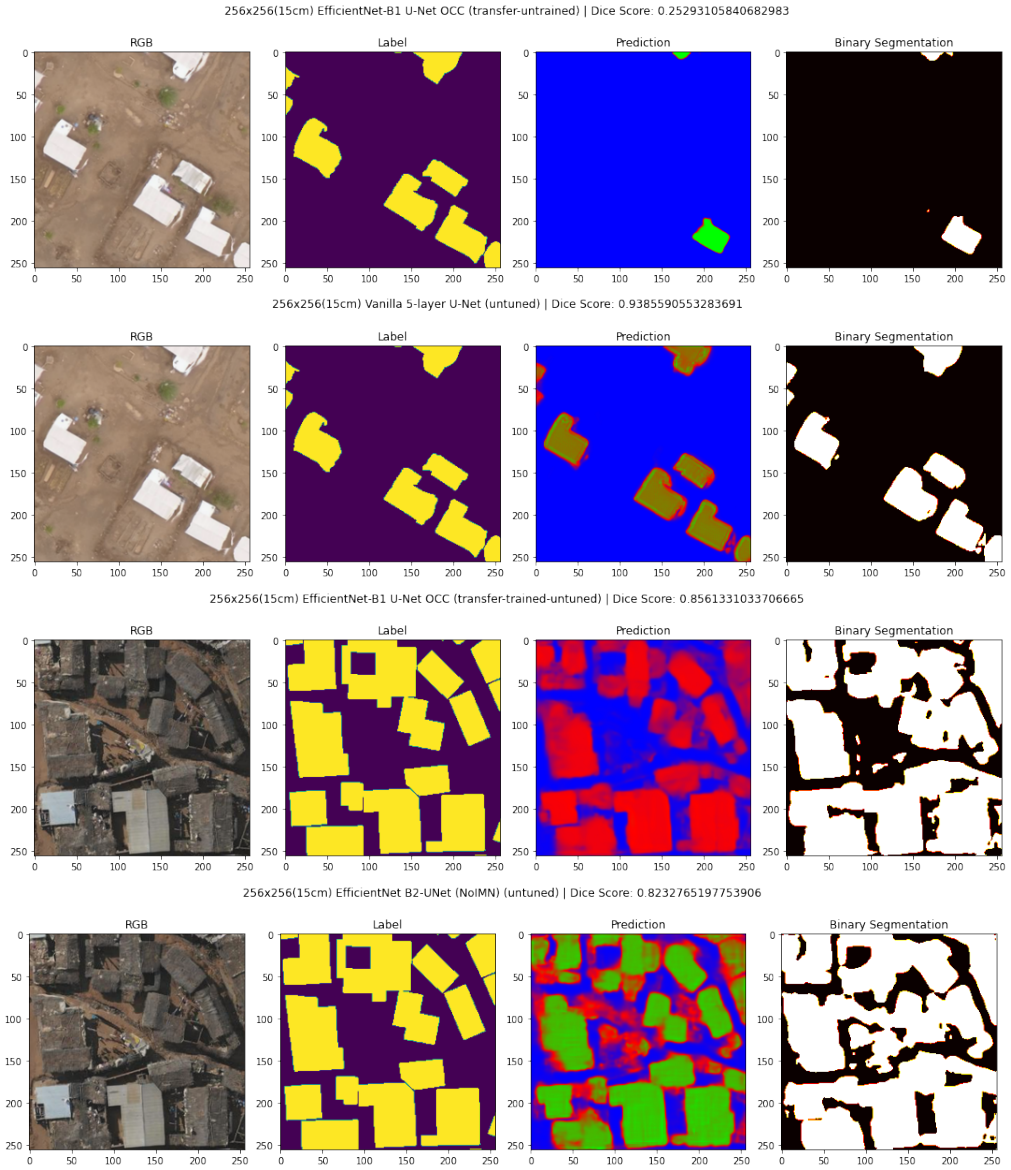


Figure 4.1: Sample of binary segmentation output of various combinations of tested architecture and experiment setup.

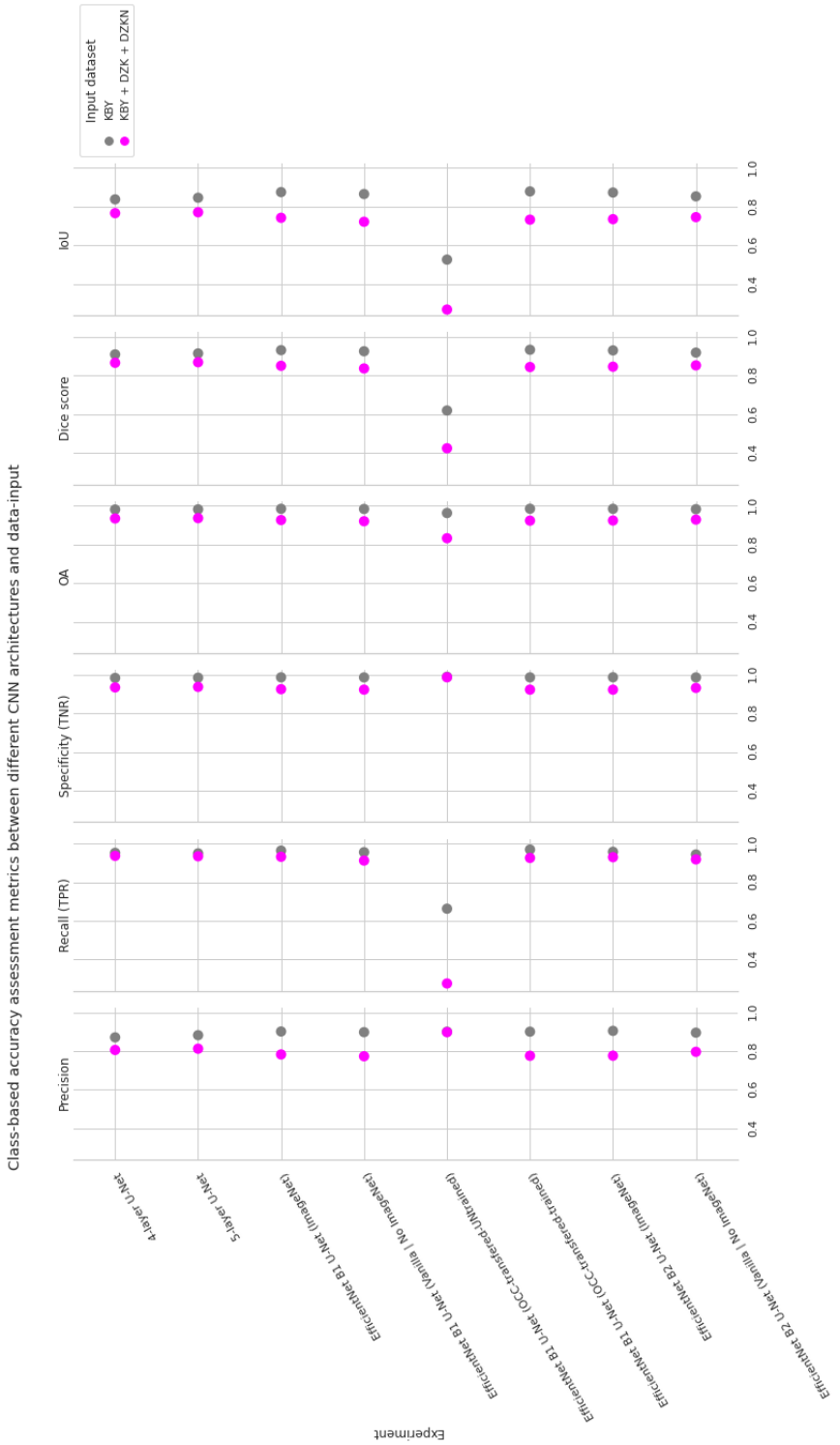


Figure 4.2: Class-based Accuracy Assesment metrics for respective CNN architectures and experiment input dataset.

When comparing the results trained on the Kalobeyei dataset (KBY), the differences between the unchanged architectures (*4-layers & 5-layers U-Net*) and the architectures with novel EfficientNet encoders (*EfficientNet B1 & B2*), the novel encoders achieve better performance on the metrics of *Precision*, *Dice Score*, and *IoU*. With the exception to *EfficientNet B1 U-Net* initialised on the OCC weights, the *Efficient B1* encoder performs better in these metrics than the *EfficientNet B2* encoder regardless of initialised weights. Meanwhile, there were negligible differences between the other metrics. This suggests that for *RQ1 & 2(a)*, with the introduction of the Dzaleka and Dzaleka North (DZK + DZKN) datasets, all EfficientNet encoders suffer from larger performance loss across all accuracy assessment metrics when compared against the unchanged U-Nets. A conclusion for *RQ1* could be drawn that the novel U-Net architectures with encoders of EfficientNet of B1 and B2 could perform better, but only with accuratly labelled data and homogeneous roofs and urban morphology. While the unchanged U-Nets could be more robust when dealing with more complicated, older refugee camps. Further details will be addressed in the discussion sections of 4.3 & 4.4.

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 4.5*), with the exception to *EfficientNet B2 U-Net* where a decrease in *Precision* but an increase in *Recall* was observed when compared to the network with non-initalised weights. This suggests that the increase in depth does not necessarily correspond 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 a consistent albeit slight improvements in *Recall* but not *Precision*. This contrast the changes in Depth-wise changes discussed in section 4.3, where increase in depth shows improvement mostly in the *Precision* but not *Recall see table 4.3*.

4.2 Discission

4.3 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.

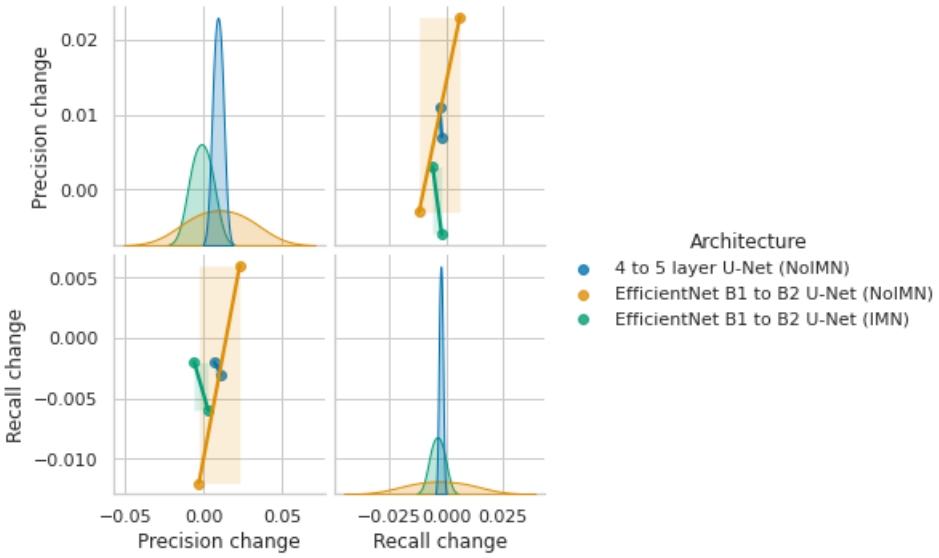


Figure 4.3: Regression plot for *Precision* and *Recall* change in relation to architectural depth-wise change.

There is a common misconception in the DL realm that deeper networks would always perform better. The comparison in *table 4.3* 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 datset input with initalised 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 differ drastically with increase of dataset, increase in batch size, and when experimenting with much deeper architectures which was unfortunately not available to this study due to computation constraints.

4.4 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 U-Net	None	-0.124	-0.044
EfficientNet B1 U-Net	ImageNet	-0.119	-0.032
EfficientNet B1 U-Net (OCC)	OCC	**-0.002**	-0.387
EfficientNet B1 U-Net (OCC)	OCC transfer-trained	-0.125	-0.043
EfficientNet B2 U-Net	None	-0.099	-0.026
EfficientNet B2 U-Net	ImageNet	-0.128	-0.028

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

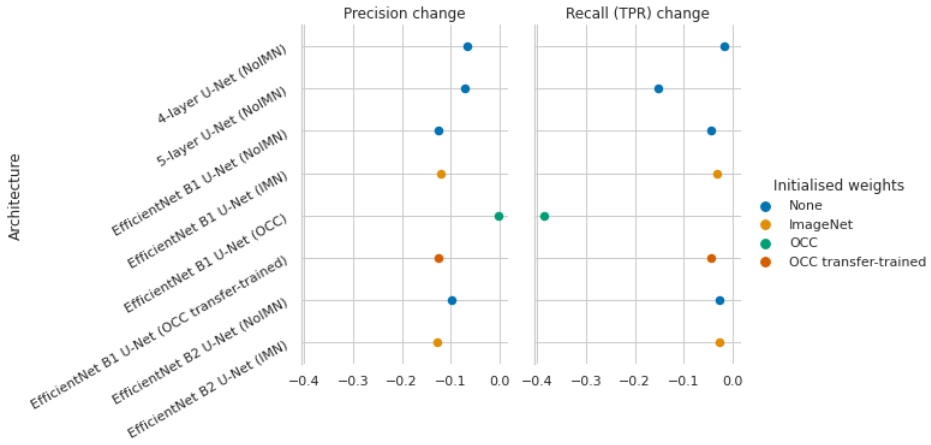


Figure 4.4: Detailed strip plot for *Precision* and *Recall* change in relation to dataset input change.

With the Kalobeyei dataset as constant, the introduction of the Dzaleka and the Dzaleka North datasets have resulted in the reduction in *Precision* and *Recall* for all architectures with or without initalised weights. Table 4.4 might suggest that the least reduction in *Precision* came from the *EfficientNet B1 U-Net* initalised on OCC building segmentation weights. However, figures 4.1 and 4.2 informs that with such poor *Dice Score* and *IoU*, it reflects that the OCC competition winning network is being either very confident at 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 result between the *Precision* and *Recall* which shows the metrics are least affected by the introduction of the Dzaleka camps datasets. However, it does not seem to be the case that deeper or shallower version of the architectures cause more or less reduction, therefore it is difficult to draw any conclusion regarding *RQ2(a)*.

4.5 Initialised weight Precision and Recall change

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

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

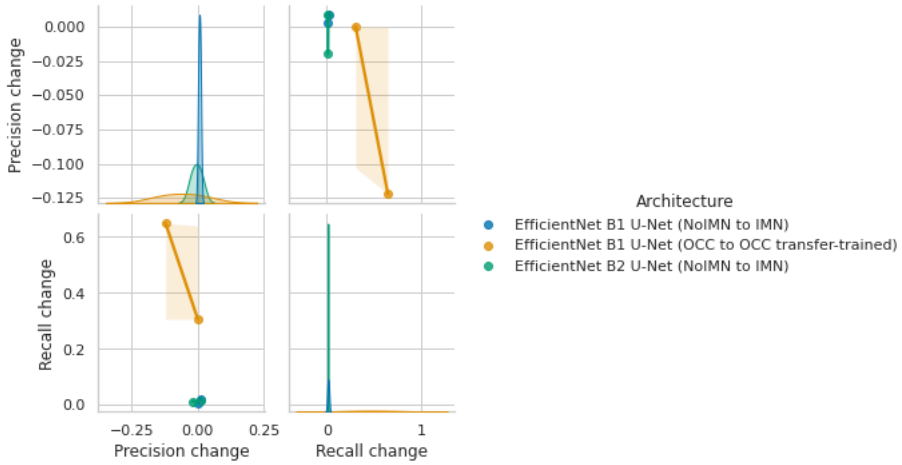


Figure 4.5: Regression plot for *Precision* and *Recall* change in relation to architectures' initalised weight change.

In section 3.4, we hypothesised that CNNs trained on pre-initalised 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 4.5 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 sigificant 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* initalised from ImageNet weights saw both improvement in the metrics but not the *EfficientNet B2 U-Net*. The result is inconclusive therefore to address the *RQ3*

Few assumptions could be drawn from the above results. Firstly, depth increase in architecture tend to favour improvement in *Precision* meaning that deeper networks would reduce the classification of $False\ Positive$. Meanwhile, ImageNet initalised weights in most setting tend to favour imrpovements in *Recall* which indicates a reduction in $False\ Negative$. 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 worse, but no significant conclusion can be drawn whether the introduction of deeper version of the architecture or initalised weights in tandem with particular dataset introduced would reduce the rate of *Precision* or *Recall* decrease. Finally, a point of contention remains in the human error in labelling. Particularly for the Dzaleka dataset, there are many instances that could cause a contribution to a $False\ Positive$ in the prediction output, which in reality were $True\ Positive$ *see figure 4.6*. These ambiguitities often arises at the courtyard of a particular building or when multiple buildings are interconnected, which is especially rampant in the complex Dzaleka camp. This might suggest that networks trained on all datasets (KBY + DZK + DZKN) might be achieving higher *Precision* than actually displayed.

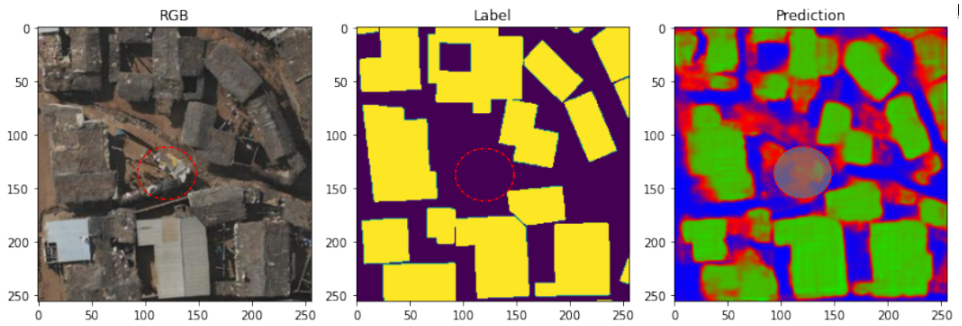


Figure 4.6: Ambiguity which arise from labelling could cause a *True Positive* prediction to be classified as *False Positive*.

5 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 rigourous 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 and/or pre-trained in large-scale CV dataset. 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 were a lot more complicated, where the models 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 ambiguitiy 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 metric does not necessarily guarantee easy transfer and good performance in new environments. This study hinted that the OCC model may have become

too customised for its original dataset of mainly formal urban structures. Secondly, it is worthwhile to carefully and critically evaluate the segmentation output by examining more granular binary classification 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 the complex Dzaleka camps datasets were introduced remained inconclusive. Thus, it is important for future studies to address this. However, the results indicate 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 investigations 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-constrainted extension of this study. Additionally, baseline experiments with 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 diagnostic understanding of how HOT could begin their open-sourced AI-assisted mapping initiative. The study defined a rigourous experiment and assessment methods of which are 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. Future development should focus on expanding this study in a pedantic and controlled way, where hyperparameters and additional input datasets are studied in a structured fashion. This will allow for a data based approach to justify resources for scaling experiemnts and future AI based products.

6 Appendix

6.0.1 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)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

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

6.0.2 EfficientNet

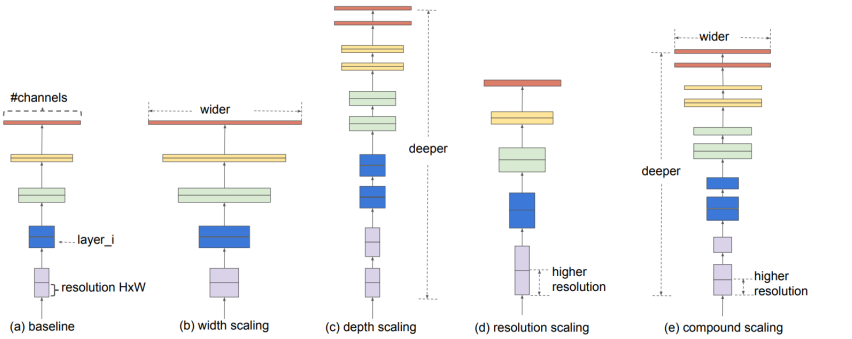


Figure 6.2: Compound scaling of the EfficientNet (Tan & Le, 2020)

6.0.3 Mean class-based accuracy assessments per experiment

Experiment	Input_dataset	mean_Precision	mean_Recall(TPR)	mean_Specificity(TNR)	mean_OA	mean_Dice	mean_IoU
4 layer U-Net	KBY	0.876088648455542	0.957042285911804	0.988149726987857	0.985645630780388	0.914777954002117	0.842940813242474
5 layer U-Net	KBY	0.887337540331463	0.954440097397393	0.989391029602743	0.986577538882985	0.919666425598122	0.85128005589134
EB1-U-Net (IMN)	KBY	0.906423641002363	0.968574427190988	0.99124608943309	0.989421059103573	0.936468974309045	0.880528119713893
EB1-U-Net (NoIMN)	KBY	0.902977163157557	0.960412447576424	0.990965811038674	0.988506317138672	0.930809641073578	0.870574293251397
EB1-U-Net (OCC-transfer-Untrained)	KBY	0.905312122866095	0.668110158849183	0.993882439015045	0.967658323400161	0.624472876164818	0.531484127044678
EB1-U-Net (OCC-transfer-trained)	KBY	0.905570133665066	0.974019928276047	0.991108207593434	0.989732630112592	0.938548648986219	0.88421259070406
EB2-U-Net (IMN)	KBY	0.909726348854154	0.962328896000268	0.991639967821063	0.989280476289637	0.935288587622051	0.878443282140799
EB2-U-Net (NoIMN)	KBY	0.900404947208695	0.948630819556973	0.990813896976398	0.987418230842141	0.923888978630002	0.85854429541461
4 layer U-Net	ALL	0.810815285020404	0.94124034523009	0.939060222754339	0.939533768577137	0.871173318487984	0.771751175584442
5 layer U-Net	ALL	0.817440288456213	0.939154520890219	0.941800109733497	0.941225459657866	0.874080622778446	0.776326121089972
EB1-U-Net (IMN)	ALL	0.787711495941837	0.936535774568188	0.929964235340173	0.931391643655711	0.855700929449568	0.747794830452809
EB1-U-Net (NoIMN)	ALL	0.778490971025587	0.916876282646316	0.927608962297886	0.925277709960938	0.842035737143943	0.727169019074135
EB1-U-Net (OCC-transfer-Untrained)	ALL	0.902935449121644	0.281553987580657	0.991601497838938	0.837371615705819	0.429256806098516	0.273282614093211
EB1-U-Net (OCC-transfer-trained)	ALL	0.780991001404883	0.930697856490893	0.927579611007119	0.928256927139458	0.84929763676765	0.738068908090146
EB2-U-Net (IMN)	ALL	0.781346604686634	0.93447336649723	0.927436932366202	0.928965320806394	0.8510771576798	0.740760933920581
EB2-U-Net (NoIMN)	ALL	0.801352691957954	0.922925754601378	0.93651633966192	0.933564319829831	0.857853369799912	0.751088649317845