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

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 expect 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, slums became 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 of 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 refuggee camps and informal settlements that have lasted well beyond their expected temporary role, there are in 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 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 set out a set of 17 Sustainable Development Goals (herein SDG) to be achieved as of 2030 (UN, 2015). Special attention are drawn to Goals 1 and 10 that are particularly relevant, of which this study hope to contribute to.

- *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, 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*

Thus, the overarching theme of this thesis is about reducing the geospatial data inequality of informal settlements (Herfort et al., 2021), thereby to improve future decision making in both humanitarian and non-humanitian context. As the topic and data provider of this project, the Humanitarian OpenStreetMap (herein HOTOSM) 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 (HOTOSM, 2021). Having up-to-date map is therefore paramount for short and long term humanitarian projects, and with the advent of Deep Learning in the last decade, 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 North-West of Kenya in Turkana County. 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 became drag 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 Kalobeyei Integrated Settlement established in 2015 benefited from a much better spatial planning in order to facilitate inclusive socio-economic development (UN-HABITAT, 2021, UNHCR & DANIDA, 2019) (*see figure 1.1*).

The Kakuma refugee camp have fluctuated in population as a response to demand, however, a dramatic increase in population in 2013 to 2014 has culminated into the development of Kakuma 4 Camp and the Kalobeyei Settlement and the Kalobeyei Integrated Socio-Economic Development Plan (KISEDP) which is the local integration strategies. 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 improve significantly the improverished Turkana county. However, challenges remain in integration into the wider Kenyan economy.

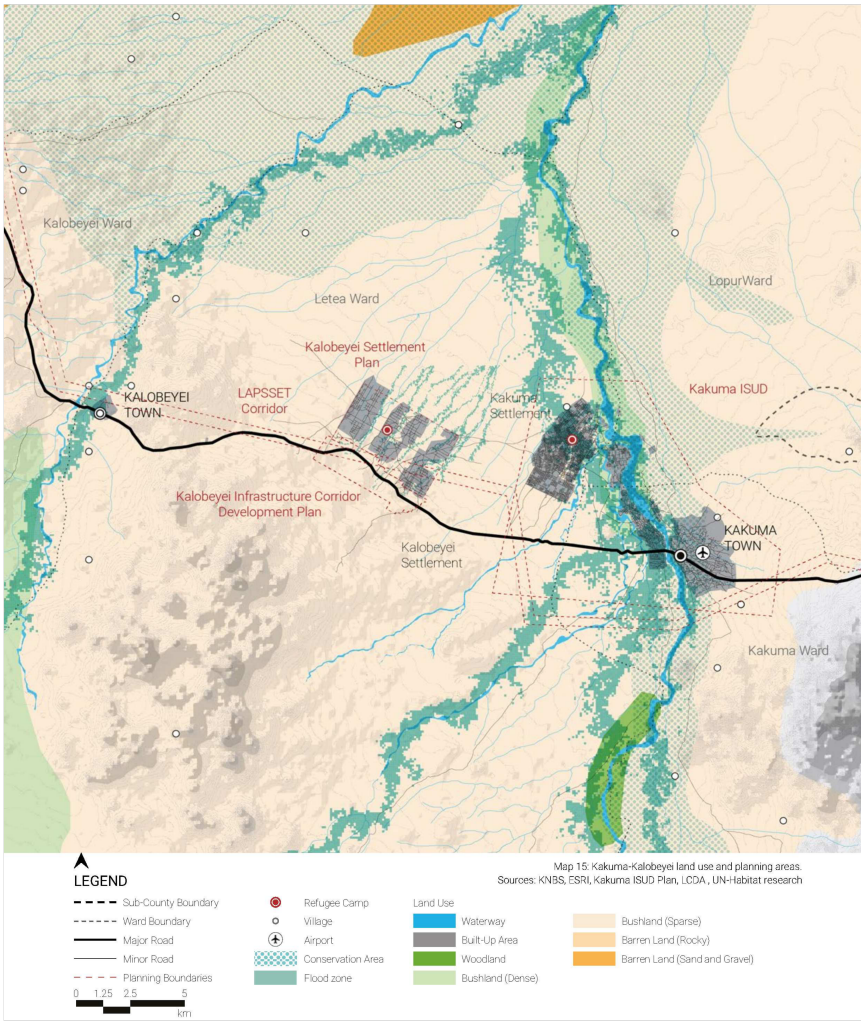

Figure 1.1: The Kakuma-Kalobeyei land use and planning areas (UN-HABITAT, 2018)

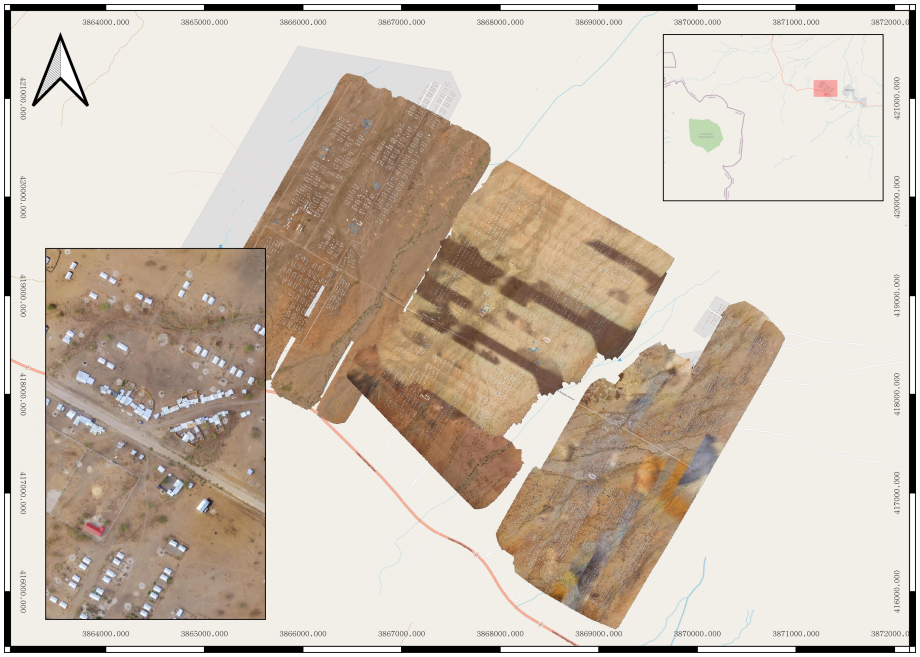

Figure 1.2: RGB drone imagery of the Kalobeyei settlements from OpenAerialMap

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 Kakuman and Kalobeyei camp, 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 infrastructures remain in 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 thesis*), 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 1.3*).

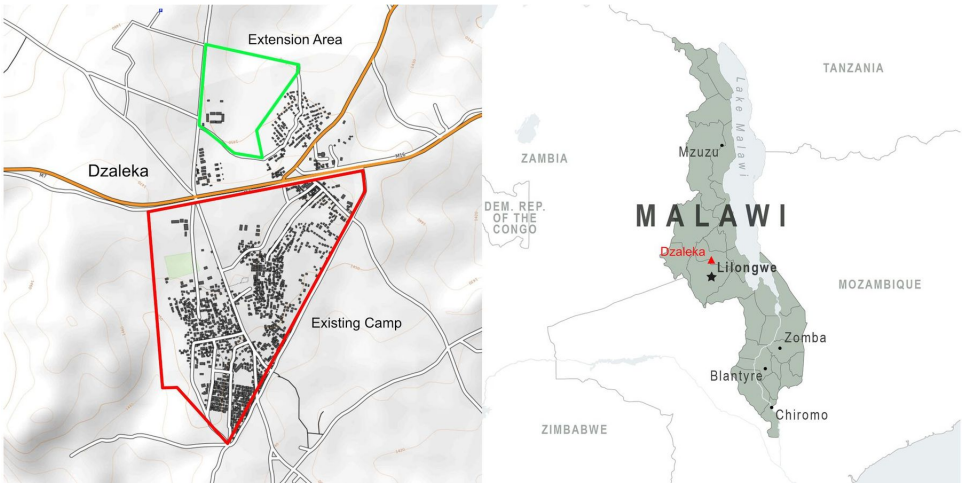

Figure 1.3: The main Dzaleka Refugee Camp and the Katubza extension plan (Dzaleka North) designed by Urban Design Advisor to the UNHCR Werner Schnellenberg (Gerhard G., 2021).

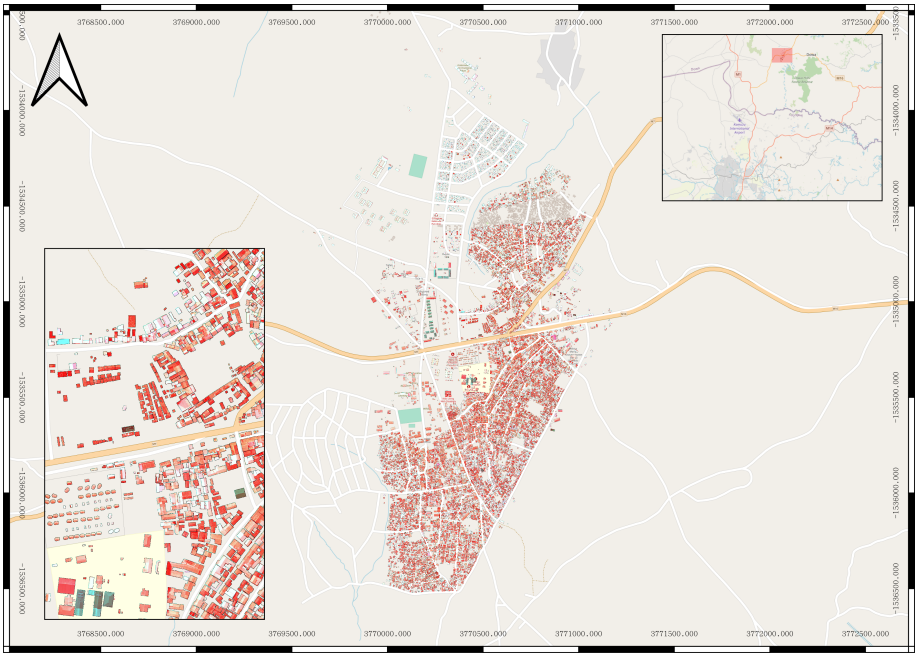

Figure 1.4: Digitised rooftop of the Dzaleka and Dzaleka North camps by HOT volunteers

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 potentialof stimulating previously improverished area could make the case for their remain (Cities Alliance, 2022).

Literature Review

Remote Sensing of Informal Settlements

Remote sensing of the urban environments have always been the unothordox in remote sensing, but simultaneously, it is the topic that essentially funded many long-standing satellite programs due to its proximity to reconissance and surveliiance 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 enviornments therefore requires data high in both spatial and temporal resolution and often employs techniques that extracts geometry, textural, and other physical features as oppose 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 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 also does not do a good job in capturing larger scale units and spatial patterns, potentially overlooking others socioeconomic determinants such as cropping cycles and infrastructure access etc.Availability of VHR sensors and publications on slums and remote sensing (Kuffer et al., 2016). 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 rural and urban settlements require different parameters. Additionally, there's no "one size fit all" way to generalise informaland 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 useuful, 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 as a study subject, previous common practice, 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 tasksj

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-occurence 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 computer-vision 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 Computer Vision 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 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 Computer Vision, 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 class between built-up and no built-up.

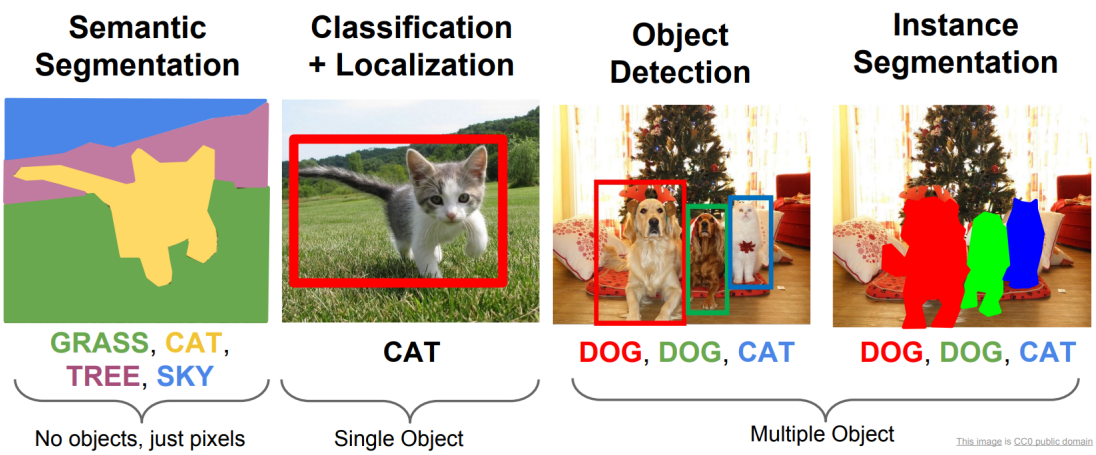

Figure 2.1: The four main types of Computer Vision tasks (Stanford University, 2022)

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 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.

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 extremely economical to procure, the spatial resolution for informal settlement application could only be rivaled by perhaps the best reconassance or commercial satellite which is either difficult or very expensive to obtain. ,

...

The issue with any Deep Learning Project is the high amount of data required (Tan et al., 2018,)

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 codified the function of a single neuron. The seminal Psychological review paper published by Rosenblatt F, (1958). Like a human neuron, The properties of a perceptron ont he most fundamental level takes an information/numerical input, stores and apply transformation, and create an output. Through stacking of basic perceptrons, a multilayered 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 the 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 an series of inter-connected layer of which its parameters are adjustable to adapt to the data provided to it. 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.

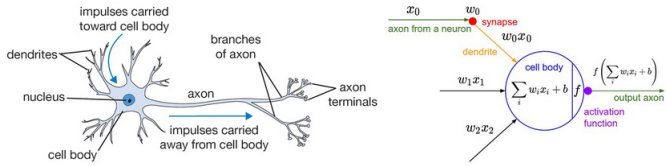

Figure 2.2: Schematic analogy diagram between a biological neuron and an artificial perceptron (Fumo D., 2017).

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

- 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

(2.1)

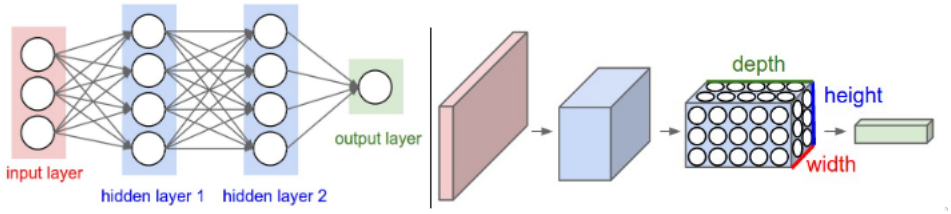

Figure 2.3: Schematic diagrom of a CNN (Stanford University, 2022).

Convolution and Pooling

The hidden layers of the CNN is where the network performs representation learning, where with each depth layer learns more abstract features of the inputted 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 2.2.3

The convolutional layer is essentially treats every image pixel as vector in a 3-Dimensional layout with input of $(BatchSize, Channel_{in}, Height, Width)$, 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: 2.2.3* of each pixel into *equation: 2.2.3*:

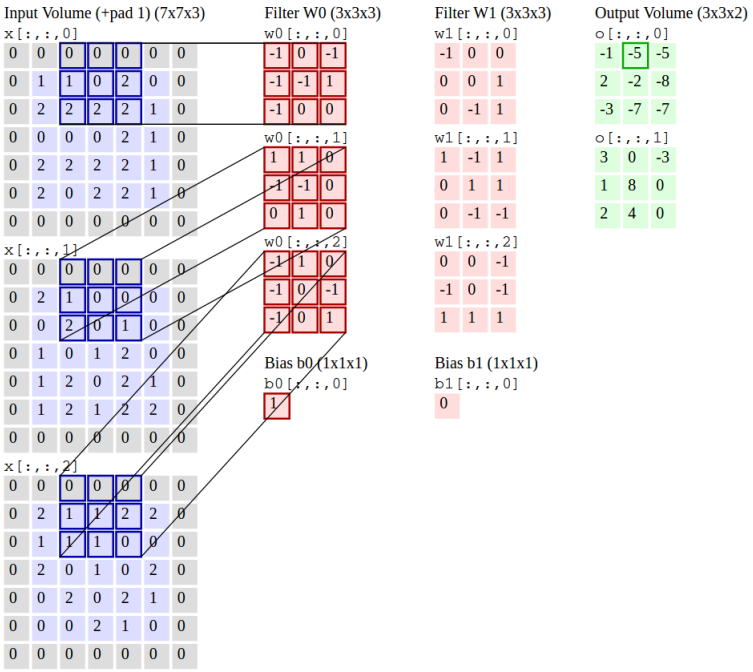

Figure 2.4: 3 x 3 Convolution (Stanford University, 2022).

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

- 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 retains 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 should scale down the image while retaining the most crucial information (*see figure: 2.5*).

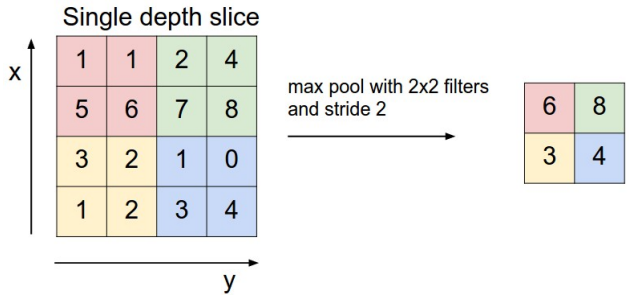

Figure 2.5: Max pooling (Stanford University, 2022).

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*.

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

- 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 prevents 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 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. 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: 2.2.3*. In other words, what is the derivate of the cost function with respect to the chain of weights and biases derivatives.

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

- 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)

(2.4)

Which when *equation:* *2.2.3* is summed over all parameters of layer i becomes *2.5*

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

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

Essentially, using CNN and DL methods to perform semantic segmentation throguh 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 3.3.

Data and Methodologies

Data

Raster pre-processing

All imagery were cubic-spline resampled to 15 cm resolution and subsequently reprojected to EPSG:3857. Normalisation of per-band raster data is perhaps one of the most important pre-processing step. 2-step normalisation were performed on each band for each drone images.

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 file type (*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 global pixel

- σ = Standard Deviation of global 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 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 augemntation applied, this increased the available data to Train n = 18242, Validation n = 3909, and Test n = 435. (*see figure. 3.2*)

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 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 4. 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 3.1*) (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...

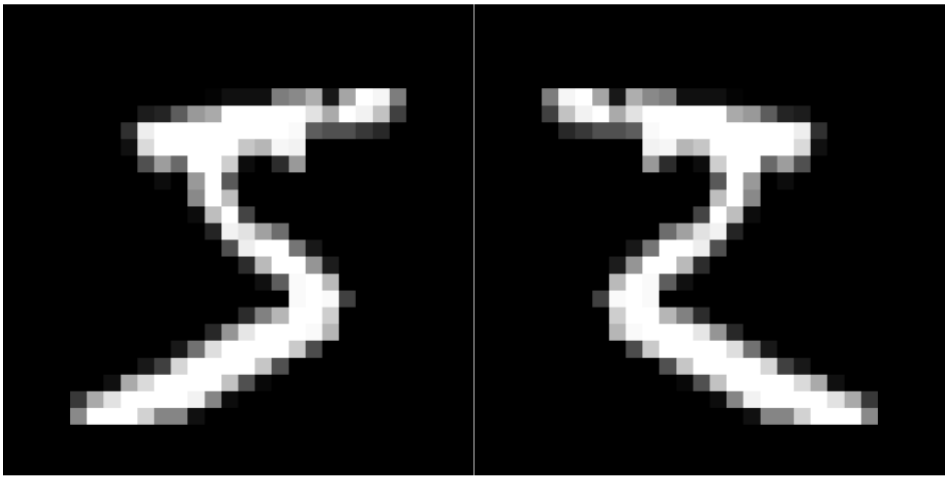

Figure 3.1: Perhaps geometric augmentation of horizontal flipping shall not be applied on the MNIST number of 5

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

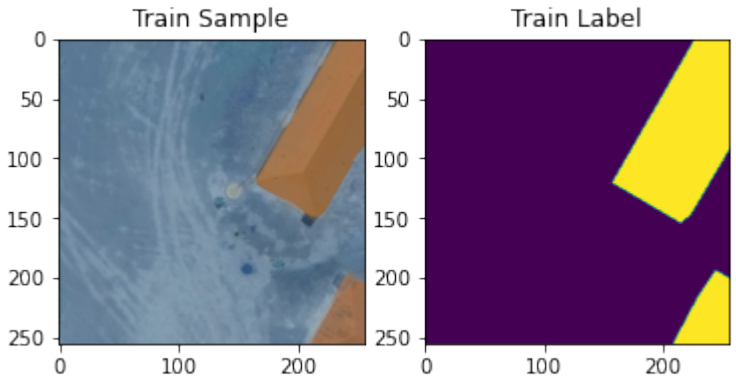

Figure 3.2: An example of Inverse RGB augmentation applied to the Train dataset.

Research Questions and experiment design

In order to train a model which performs well on drone imagery, the motion artefact will be a signficant feature for the model to learn. The combination of data availability have allow 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 sucess of cross-domain transfer training from classical CV datasets to remote sensing tasks (e.g. Audebert et al., 2017, Marmanis et al., 2015), One would expect the transfer training of CNN pretrained 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 initalised weights affect result?

Therefore, to test out U-Net and a few variation of the U-Net performance, an additional set of label data was created in order to supplement the imperfection in the labelled data of the Dzaleka camps. Initially, the models will 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 3.3* 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 (OCC) winning model is conducted.

Figure 3.3: 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, tried and tested architectures with relatively low number of training parameters is ideal.

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 fig-*

ure 3.4) 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 and recover and interpolate spatial features in the connected unsampling decoder layers (Wurm et al., 2019).

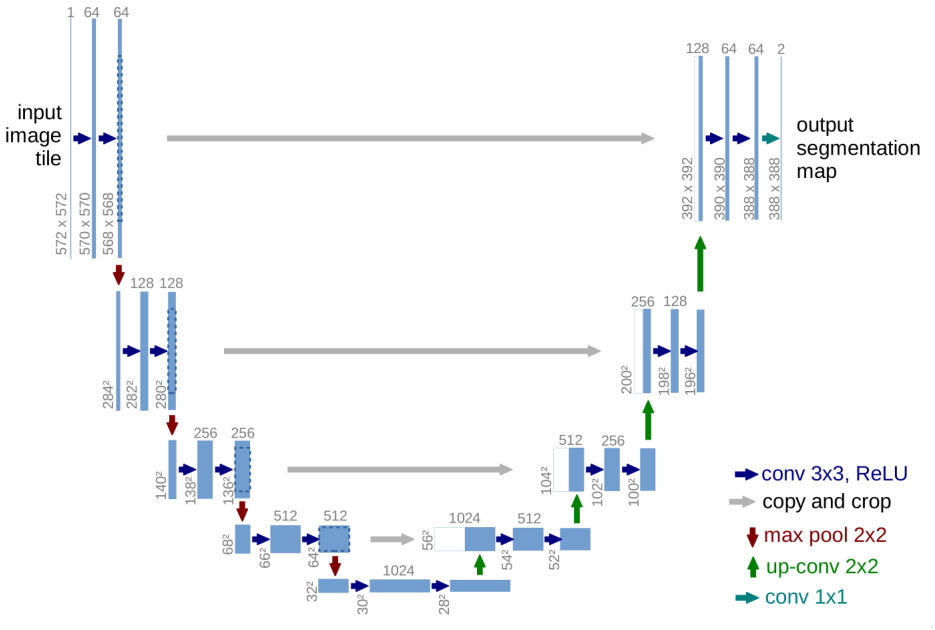

Figure 3.4: The Encoder-Decoder U-Net architecture (Ronneberger et al., 2015)

Changing the encoder architecture and the EfficientNet family

The ability to switch out the encoder structure allows the DL 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 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 drone building segmentation which this study will transfer, and compare against.

In this study, the experiments with changed encoder will be from the EfficientNet family. There are three reasons for this decision. Firstly, at one of the last stage of the OCC compeition winning network, 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 phenomenally well in classical CV datasets (*see figure 3.5*) (Tan & Le, 2020). In essence, these are sets of experiments that mix and match old and new architectural design.

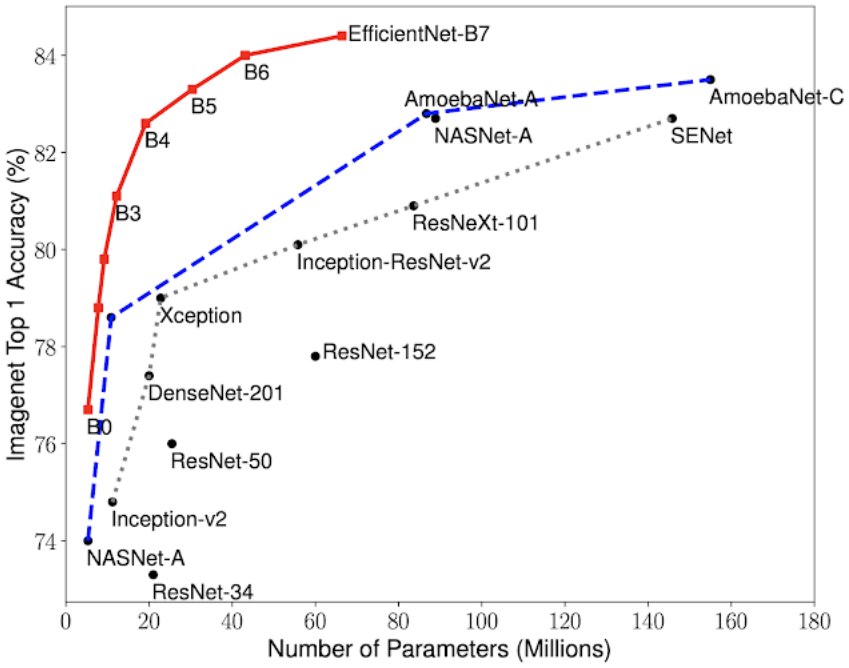

Figure 3.5: EfficientNet family Top 1% Accuracy Assessment in ImageNet (Tan & Le, 2020).

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.

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 (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

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 3.4*). One of the most difficult process in DL is finding the correct hyperparameters value that cause the model to neither overfit or underfit the dataset. Thus, the strategies and options are often overwhelming, therefore this study does not concern itself with tuning the hyperparameters to all of the models but rather withholding them from changing so that a comprehensive **baseline** performance could be identified. This will provide a clear picture of the feasibility and 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 *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 2.2.3*
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 3.1: 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, 2019). The rest of this section will consider the lower order and higher order accuracy metrics, 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 3.6*)

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

Figure 3.6: The Confusion Matrix

Precision, Recall, Sensitivity, and Specificity

Precision, Recall, and Specificity

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

Overall Accuracy, Dice Score, and Intersection-over-Union

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.

Overall Accuracy

$$Overall\ Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \tag{3.6}$$

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

Dice Score

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

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

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

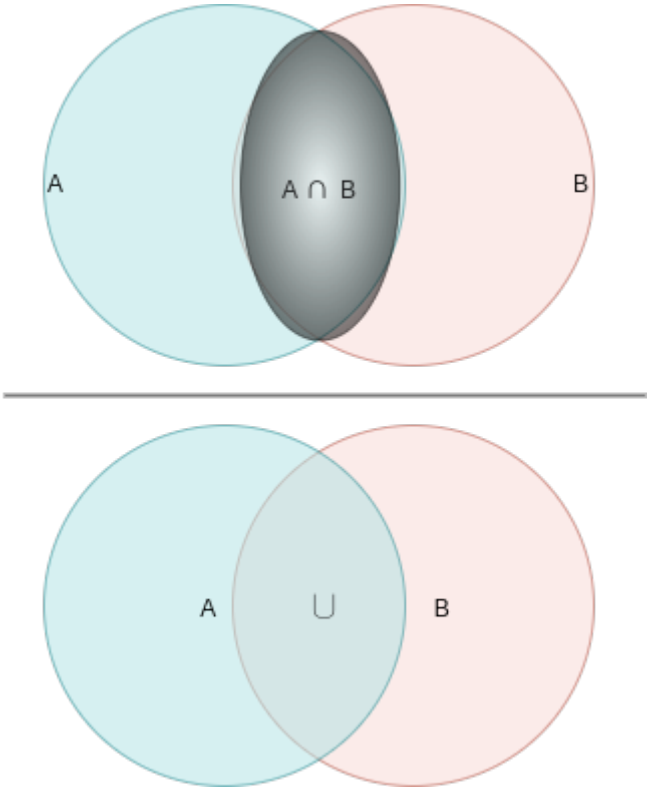

Figure 3.7: Schematic diagram of Intersection-over-Union

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

Experimentation setup

Each network architecture and their associated weights will be trained on 2 data setup. The first setup consist of only the Kalobeyei, Kakuma camp where the labels include drone motion artefacts and rooftops are relatively homogeneous. The second setup consist of data from the Kalobeyei camp and also the rest of Dzaleka camps. The second setup will introduce imperfection in labelling and complex hetereogeneous rooftops and morphologies. The two data setup will allow comparison between the models response of each class-based accuracy assessment metrics. To assess the performance of the architectures and experiment setup, a combination of the validation loss, class-based accuracy assessments, and empirical output were considered.

Project workflow

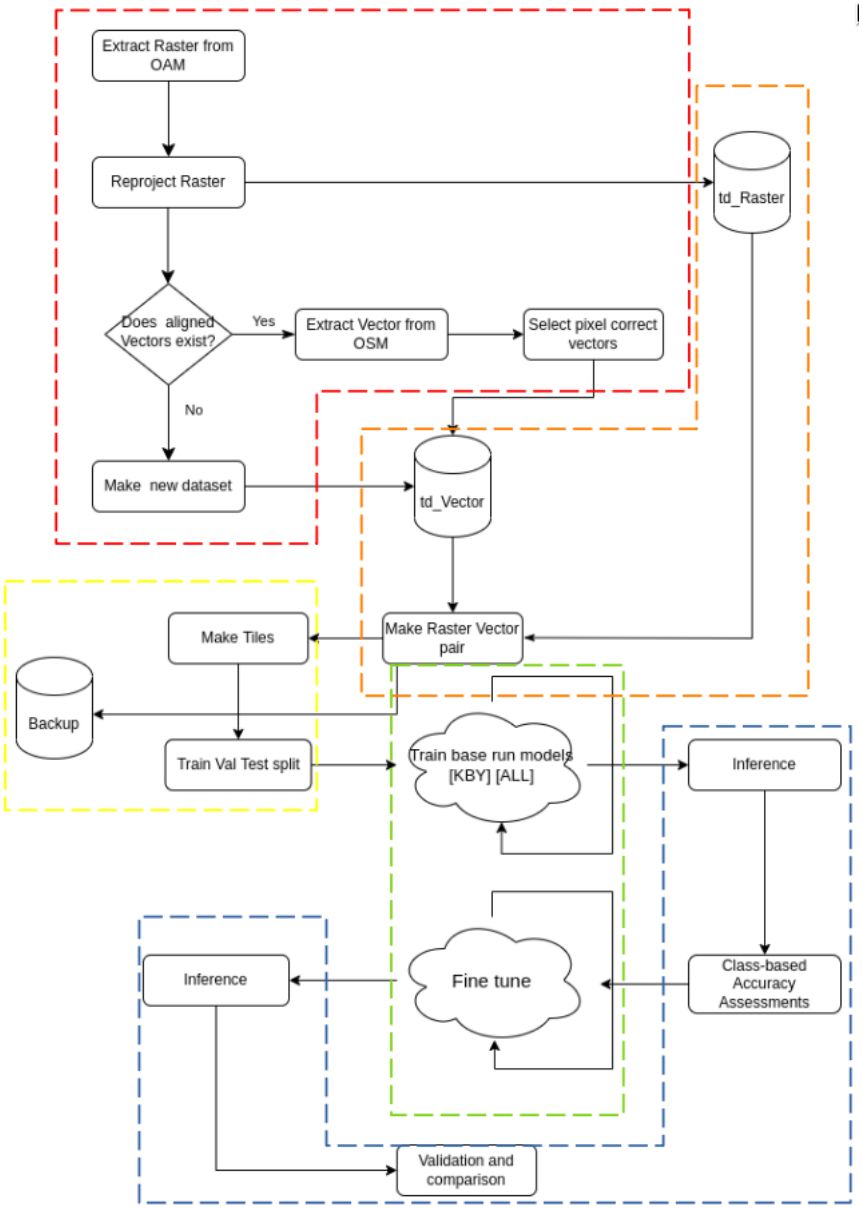

Figure 3.8: Project workflow

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 3.8 & 3.9*).

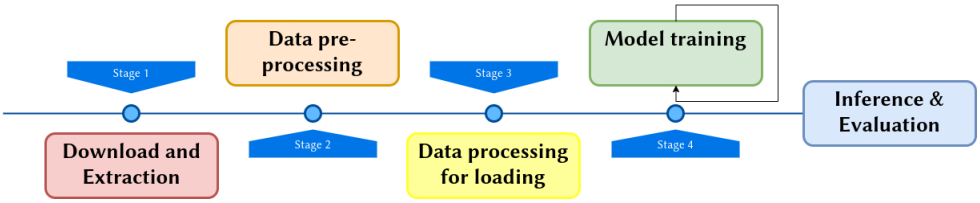

Figure 3.9: Simplified project workflow in reference to 3.8

Findings and Discussion

This study have tested on 5 different architectures with either no initalised weights or initalised weights from ImageNet or OCC building segmentation models. For each experimental 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 camp 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 transfered-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 point 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 pretrained ImageNet weights (*see table 4.3*). However, this was not the case for the *EfficientNet B2 U-Net* where if the network was initalised on ImageNet pretrained weights, saw a decrease in *Precision* but a increase in *Recall* 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 an consistent but slight increase in *Precision* but not *Recall*. This mirrors the changes in *EfficientNet B1 to B2 U-Net*. Which might suggests that complications are originating from the initalised weights of ImageNet (*see figure 4.2*).

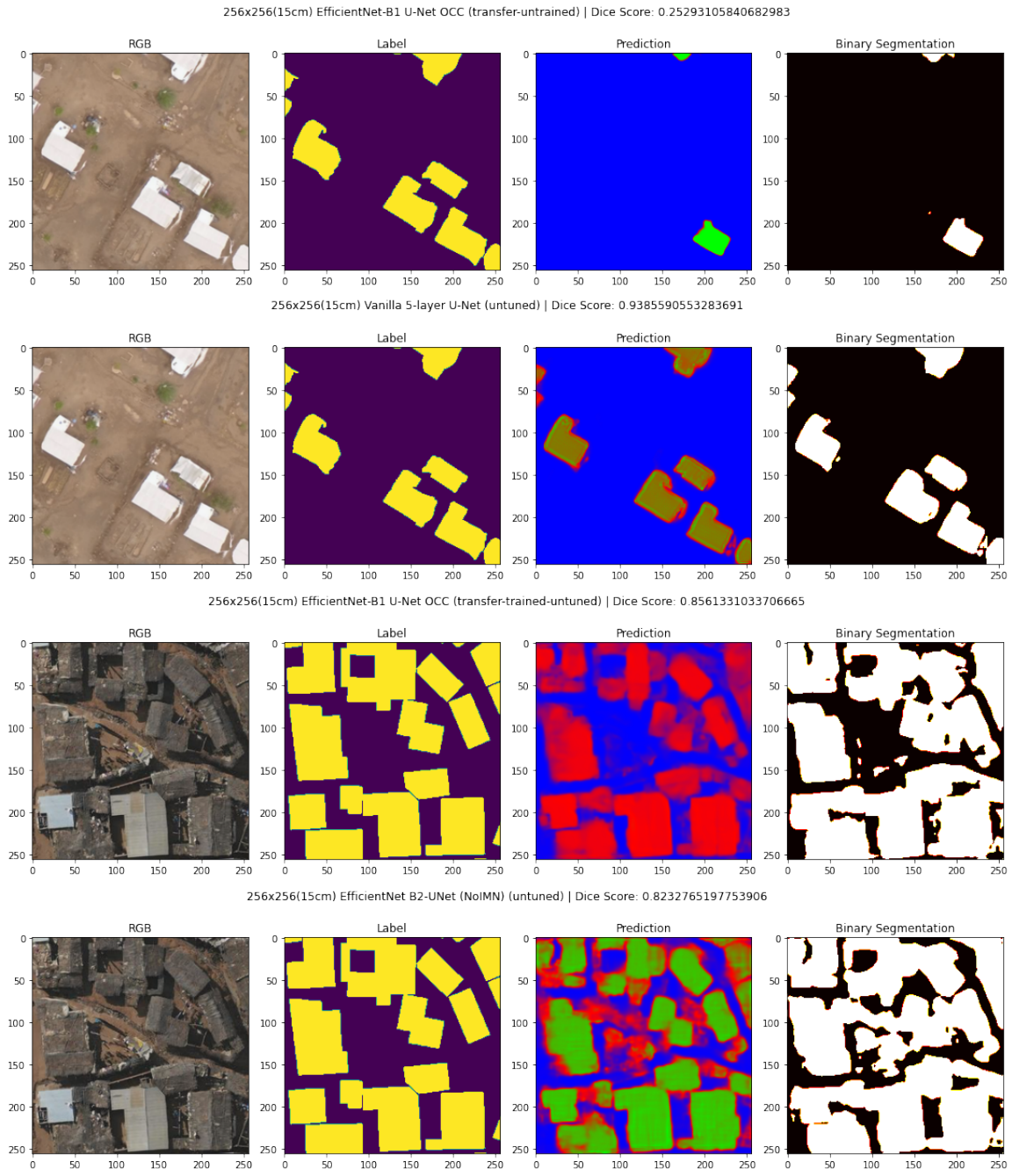

Figure 4.1: Sample of binary segmentation output of various combinations of tested architecture and experiment setup.

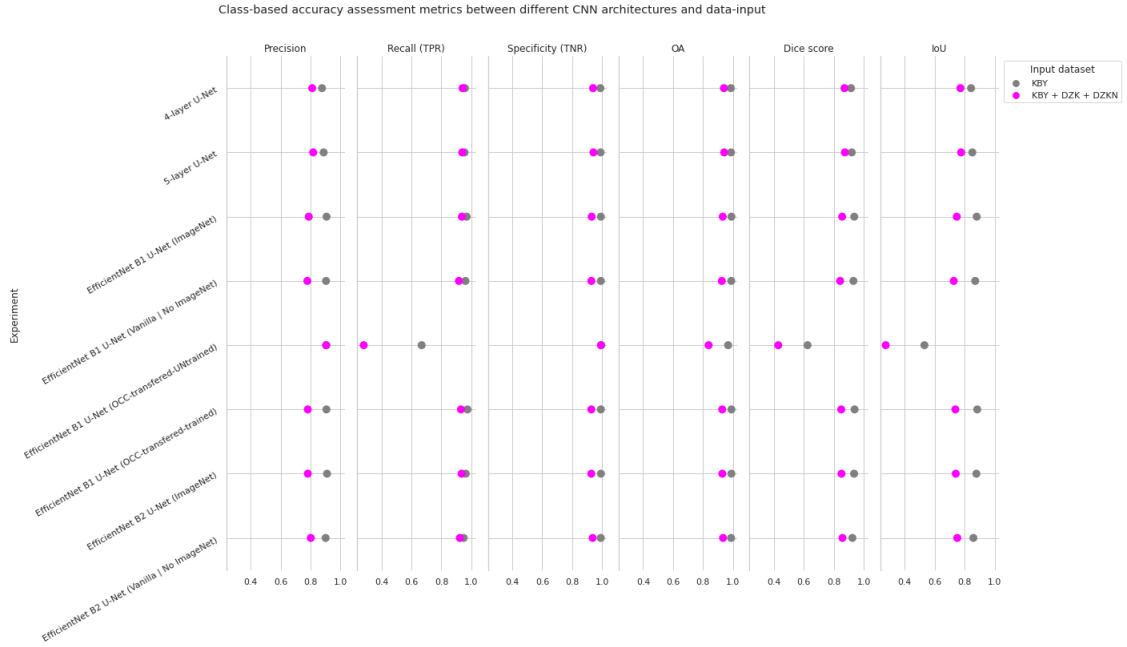

Figure 4.2: Class-based Accuracy Assesment metrics for respective CNN architectures and experiment input dataset.

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..

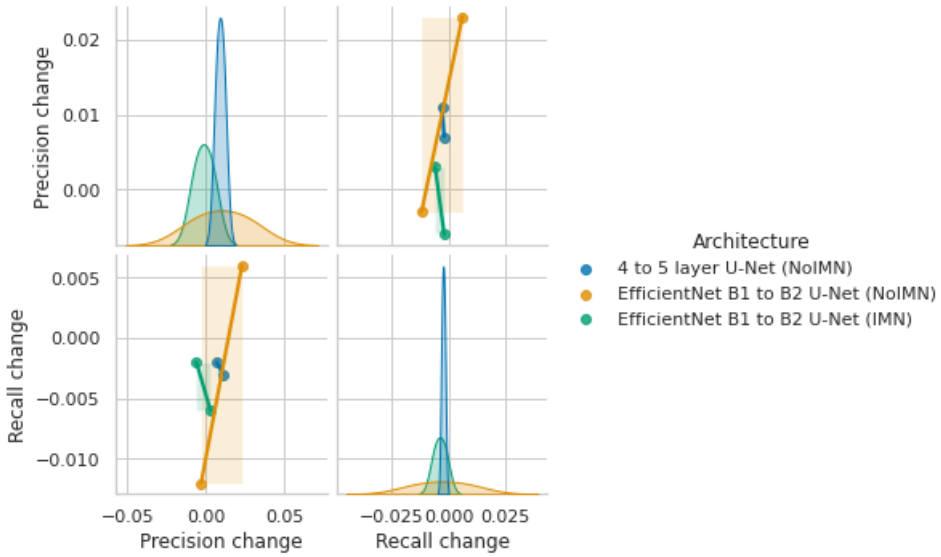

Figure 4.3: Regression plot for *Precision* and *Recall* change in relation to architectural depth-wise change.

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 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 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	-0.153
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.

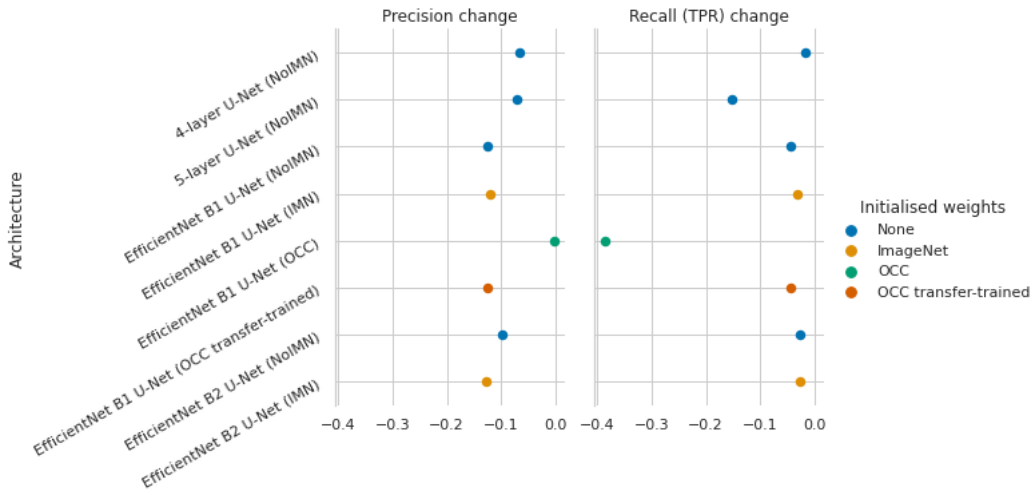

Figure 4.4: Detailed strip plot for *Precision* and *Recall* change in relation to dataset input change.

With the Kalobeyei dataset as constant, the introduction of the Dzaleka and the Dzaleka North dataset have resulted in the reduction in *Precision* and *Recall* for all architectures with any or no initalised weights. Table 4.2 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 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 runner-up *4-layer U-Net* had a much more corresponding *Precision* and *Recall*.

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

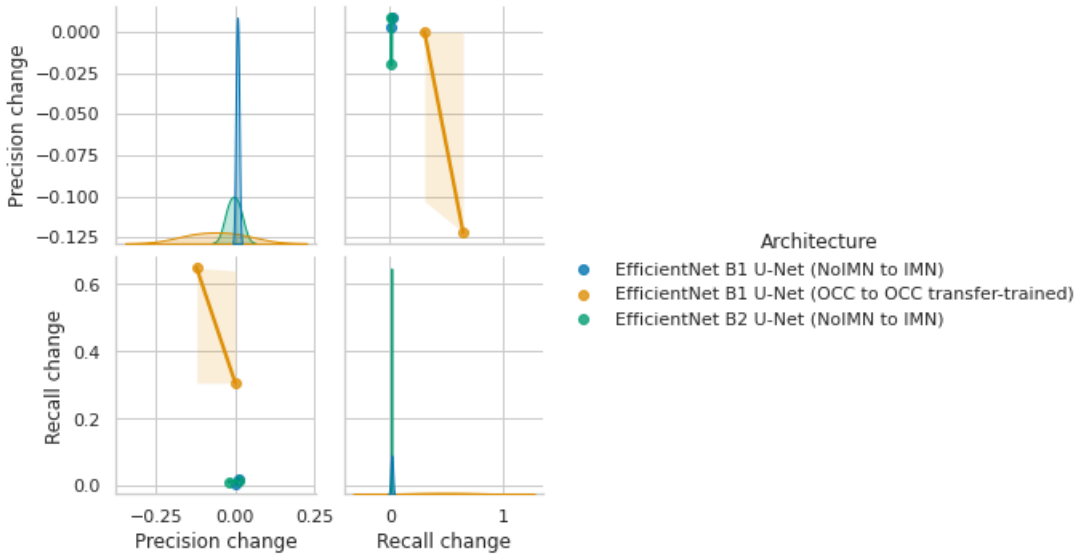

Figure 4.5: Regression plot for *Precision* and *Recall* change in relation to architectures' initalised weight change.

As mentioned perviously in section 3.2 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 4.3 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 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*. It is therefore difficult to draw any conclusion here.

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 1^{st} moment vector)
$v_0 \leftarrow 0$ (Initialize 2^{nd} 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 4.6: The algorithm of Adam (Kingma & Ba., 2017).

Conclusion

The beginning of a DL initiative 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 discover the possibilities and challenges with small-scale yet rigourous evaluations. This study presented a series of experiments

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)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (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).