

https://dspace.org/

Self-supervised fully-convolutional neural networks for segmentation in the object-based image analysis workflow

Bruton, Joshua

2022
https://hdl.handle.net/10539/33879

Downloaded from a DSpace repository. https://dspace.org/

University of the Witwatersrand



WITS
UNIVERSITY

School of Computer Science and Applied Mathematics
Faculty of Science
Masters Research Dissertation

Self-supervised Fully-convolutional Neural Networks for Segmentation in the Object-based Image Analysis Workflow

Joshua Bruton

Supervisor: Dr. Hairong Wang

March 4, 2022

Declaration
University of the Witwatersrand, Johannesburg
School of Computer Science and Applied Mathematics
SENATE PLAGIARISM POLICY

I, Joshua Bruton, (Student number: 1407370) am a student registered for COMS8003A in the year 2021.

I hereby declare the following:

- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- I confirm that ALL the work submitted for assessment for the above course is my own unaided work except where I have explicitly indicated otherwise.
- I have followed the required conventions in referencing the thoughts and ideas of others.
- I understand that the University of the Witwatersrand may take disciplinary action against me if there is a belief that this in not my own unaided work or that I have failed to acknowledge the source of the ideas or words in my writing.

Signature: 

Signed on 4th day of March , 2022 in Johannesburg.

Abstract

The collection of object-based image analysis literature prescribes a particular workflow for achieving semantic segmentation in small satellite image data sets, relying on non-learned algorithmic segmentation techniques for object-proposal. This research proposes a novel, learned approach to object-proposal in object-based image analysis using fully convolutional neural networks trained with self-supervised learning and synthetic data. Fully-convolutional neural networks are a state-of-the-art approach to segmentation but have limited spatial reasoning and receptive fields, and they also require large amounts of semantically labelled data for end-to-end training. This research shows that using a fully convolutional neural network within object-based image analysis can effectively incorporate learning into the workflow and alleviate concerns regarding neural networks generalising from synthetic data.

Keywords - object-based image analysis, receptive fields, fully convolutional neural networks, semantic segmentation, self-supervised learning

Contents

Preamble i
Declaration . . . i
Abstract . . . ii

List of Figures vi

1 Introduction 1
1.1 Background . . . 1
1.2 Problem Statement . . . 2
1.3 Research Objectives . . . 2
1.4 Research Methodology . . . 3
1.5 Assumptions, Limitations, and Scope . . . 3
1.6 Summary . . . 4

2 Literature Review 5
2.1 Object Based Image Analysis . . . 5
2.1.1 Introduction . . . 5
2.1.2 Workflow . . . 6
2.1.2.1 Image Correction . . . 7
2.1.2.2 Object Proposal . . . 7
2.1.2.3 Feature Extraction . . . 11
2.1.2.4 Classification and Clustering . . . 14
2.1.3 Applications . . . 16
2.2 Neural Networks . . . 16
2.2.1 Convolutional Neural Networks . . . 16
2.2.1.1 Pooling . . . 19
2.2.1.2 Receptive Fields . . . 19
2.2.1.3 Dilated Convolution . . . 21
2.2.1.4 Skip Connections . . . 22
2.2.2 Fully Convolutional Neural Networks . . . 23
2.2.2.1 Introduction . . . 23
2.2.2.2 Transposed Convolution . . . 25
2.2.2.3 Prediction Refinement . . . 26
2.2.3 Loss Functions . . . 27
2.2.3.1 Binary Cross Entropy and Soft Dice Loss . . . 28
2.2.3.2 Kullback–Leibler Divergence and Donsker-Varadhan Loss . . . 28
2.2.3.3 Mutual Information Neural Estimation . . . 29

2.2.3.4 Cauchy-Schwarz Divergence . . . 30
2.2.3.5 Set Cross Entropy . . . 31
2.3 Learning Paradigms . . . 31
2.3.1 Supervised . . . 31
2.3.2 Unsupervised . . . 32
2.3.3 Self-supervised . . . 33
2.4 Summary . . . 33

3 Research Methodology **35**
3.1 Introduction . . . 35
3.2 Proposed Workflow . . . 35
3.3 Research Hypothesis . . . 36
3.4 Assumptions, Limitations, and Scope . . . 37
3.5 Optional Translational Equivariance . . . 38
3.6 Translated Skip Connections . . . 40
3.7 Permutation-invariant Loss Function . . . 42
3.7.1 Self-augmented Hungarian Cross-Entropy . . . 44
3.8 Datasets . . . 46
3.8.1 Real-world Data . . . 47
3.8.1.1 Inria Image Labelling Dataset . . . 47
3.8.1.2 Landcover.ai . . . 47
3.8.1.3 COVID-19 CT Scans Dataset . . . 48
3.8.1.4 PASCAL Visual Object Classes 2012 . . . 49
3.8.1.5 Describable Textures Dataset . . . 49
3.8.2 Synthetic Data . . . 50
3.8.2.1 CARLA Semantinc Segmentation for Self-driving Cars . . . 50
3.8.2.2 The Prague Texture Segmentation Datagenerator and Benchmark . . . 51
3.8.3 Synthetic Data Generation . . . 52
3.8.3.1 Ellipsoidal Dataset . . . 52
3.8.3.2 Simplex Silhouettes . . . 54
3.8.3.3 The Prime Texture Analysis Dataset . . . 55
3.9 Summary . . . 56

4 Results **57**
4.1 Introduction . . . 57
4.1.1 Comparison Architectures . . . 57
4.1.2 Experimental Setup . . . 58
4.2 Optional Translational Equivariance . . . 58
4.2.1 Image Size . . . 58
4.2.1.1 Experiments and Results . . . 58
4.2.1.2 Discussion . . . 60
4.2.2 Ablation Study . . . 60
4.2.3 Overfitting . . . 61
4.3 Translated Skip Connections . . . 62
4.3.1 Ablation Study . . . 62
4.3.2 Experiments and Results . . . 63
4.3.3 Discussion . . . 65

4.4 Permutation-invariant Loss Function . . . 66
4.5 Proposed Workflow . . . 67
4.5.1 Benchmark Performance . . . 67
4.5.2 Consistency . . . 69
4.5.3 Comparison to Traditional Algorithms . . . 69

5 Conclusion **74**
5.1 Summary . . . 74
5.2 Limitation Discussion . . . 76
5.3 Future Research . . . 77

Bibliography **78**

List of Figures

2.1 Increase in spatial resolution . . . 6
2.2 Flowchart depiction of the OBIA workflow . . . 6
2.3 Simple Linear Iterative Clustering Algorithm . . . 8
2.4 Example of SLIC Superpixel Segmentation . . . 9
2.5 Example of Watershed Segmentation . . . 10
2.6 Example of a grey-level co-occurrence matrix . . . 12
2.7 KMeans algorithm . . . 14
2.8 Example of image convolution . . . 17
2.9 An example of convolution with multiple channels . . . 18
2.10 Examples of pooling . . . 19
2.11 Example of Receptive Field . . . 20
2.12 Depiction of Dilated Convolution . . . 21
2.13 Illustration of a skip connection . . . 22
2.14 Architecture diagram of U-Net . . . 23
2.15 Architecture diagram of V-Net . . . 24
2.16 Convolution as matrix multiplication . . . 25
2.17 FCN Refinement Algorithm . . . 26
2.18 Example of FCN Refinement . . . 27

3.1 The proposed method . . . 35
3.2 OTE Rectangle Dataset Example . . . 38
3.3 OTE Meshgrid . . . 39
3.4 Top and Left Rectangle Dataset Example . . . 40
3.5 Translation Function . . . 41
3.6 Diagram of the proposed TSC architecture . . . 41
3.7 All Permutation Example Dataset . . . 42
3.8 FCN PDF Predictions trained with Donsker-Varadhan Loss . . . 43
3.9 Hungarian Algorithm Example . . . 45
3.10 Samples from IILD . . . 46
3.11 Samples from Landcover.ai . . . 47
3.12 Samples from the COVID-19 CT Scan Dataset . . . 48
3.13 Samples from PASCAL VOC2012 . . . 49
3.14 Textures drawn from the Describable Textures Dataset . . . 50
3.15 Examples from the CARLA Synthetic Dataset . . . 51
3.16 Samples from the Prague Texture Segmentation Benchmark . . . 52
3.17 Synthetic Ellipse Dataset . . . 53
3.18 Silhouettes Generated using ConvexHull and OpenSimplex . . . 53
3.19 Samples drawn from the Prime Texture Analysis Datagenerator . . . 54

4.1 OTE MIoU vs. Image Size 59
4.2 SU-Net's Predictions versus OTE 60
4.3 OTE causing overfitting 62
4.4 TSC Ablation Study 63
4.5 Permutation-invariant Loss Charts 67
4.6 TSC-Net Consistency 68
4.7 Heatmap of Spearman Correlation matrices for different object proposal methods 70
4.8 OBIA Segmentation Comparison 72

Chapter 1

Introduction

This research proposes four additions to object-based image analysis (OBIA) with the intention of incorporating learnability into the object proposal stage of the OBIA workflow. Including learnability in the object proposal step reduces the amount of human tuning required and makes the proposed objects more interprable. The modifications relate to the creation and training of a fully-convolutional neural network (FCN) capable of generalising from a self-supervised, synthetic texture analysis task to the task of object proposal. This chapter begins by providing an introduction to the research topic and intends to give an overview of the structure and purpose of this dissertation.

1.1 Background

Object-based image analysis has gained popularity over the last two decades as a sensible means to conduct semantic segmentation on satellite imagery with ever-increasing spatial resolution. The technique prescribes a particular workflow, henceforth referred to as the OBIA workflow, as a simple and adaptable guide for semantic segmentation in satellite images. The success of the workflow depends greatly on the algorithms used at each stage; one such stage involves the selection of a suitable segmentation algorithm. Segmentation algorithms have grown in complexity since watershed and region-merging and are now hybrids of many different approaches typically implemented in proprietary software.

More recently, fully-convolutional neural networks have come to the fore as a state-of-the-art approach to semantic segmentation in images. One challenge in using FCNs is the availability of large amounts of training data; this requirement often hampers the use of FCNs in certain domains. Solutions to this problem have been proposed; they include the creation of entirely synthetic datasets or pre-training on different datasets in similar domains. As FCNs continue to be researched it can be expected that they will become more generalisable and less data-starved. Another limitation of FCNs in this setting is their fully-supervised nature, but training on non-semantic data can be achieved with self-supervised relabelling schemes.

If these limitations can be overcome, the reliability and learnability of FCNs make them ideal candidates for induction into the paradigm of object-based image analysis. An FCN trained on synthetic data or with self-supervised learning may be able to generalise to the level of traditional segmentation algorithms. The accuracy an FCN would lose by not seeing as many training samples can be effectively accounted for by the remaining steps in the OBIA workflow, reducing pressure on the FCN. Opening this avenue of research would also reduce the field's current reliance on proprietary software for the implementation of segmentation algorithms.

1.2 Problem Statement

Contemporary OBIA approaches are characterised by fine-tuning and the implementation of complex hybrid segmentation methods; a wider variety of approaches to segmentation in OBIA could address this. In this dissertation, the approach used to solve this problem is outlined and details with which to validate the feasibility of such a methodology are provided. This research intends to construct a self-supervised FCN trained entirely on automatically annotated data that can effectively replace the algorithmic segmentation techniques used in the object proposal step of the OBIA workflow.

This research provides techniques that account for some of the limitations of FCNs performing segmentation in the OBIA workflow. The entire research project constitutes four independently justifiable contributions that synthesise into that single proposed method. The problems resolved by these contributions directly relate to those aforementioned limitations; the receptive fields of FCNs are expanded with Translated Skip Connections, their spatial reasoning is improved with Optional Translational Equivariance, learning from non-semantic data in an end-to-end fashion is allowed for by Hungarian Cross-Entropy, and a sufficiently complex texture analysis dataset, called the Prime Texture Analysis Dataset, is generated and released. This research hypothesised and demonstrated that the OBIA workflow would lessen the burden placed upon the segmentation provided by the FCN, thus reducing the negative impact of not having seen real-world training data and improving the generalisability of the model.

1.3 Research Objectives

The primary objective of this research is to modify the existing segmentation and object proposal step of the OBIA workflow, to make it more learnable than existing approaches while maintaining comparable performance. To this end, a more detailed enumeration of the steps conducted in the proposed framework is provided below and a deeper discussion is saved for chapter 3.

1. Begin with sample images
2. Use an FCN to segment the image, instead of algorithmic segmentation techniques
    - This FCN will be trained exclusively on an automatically annotated dataset for texture segmentation in a permutation-invariant fashion
    - The FCN's output can be tuned using FCN refinement and modifications to the training data
3. Construct meaningful feature vectors for the objects proposed by the FCN
4. Cluster the proposed objects in order to extract the semantic features of the image

The four alterations mentioned in section 1.2 all relate to the implementation and achievement of this modification to step 2, and thus implementing and verifying their functionality are crucial to this dissertation's objective: introducing learnability into step 2 of the OBIA workflow.

1.4 Research Methodology

To allow for the functioning and implementation of the stated research objective, four modifications were required and verified. The research dissertation is therefore framed around these four modifications and they are each independently verified before the overall workflow outlined in the previous section is considered in section 4.8. For clarity, the four modifications are introduced and verified in the order they were implemented, which is enumerated below.

1. Optional Translational Equivariance (OTE) in section 3.5, the ability for OTE to make an FCN architecture more spatially aware is tested in section 4.2.
2. Translated Skip Connections (TSC) are introduced in section 3.6 and experiments regarding how effectively they expand the receptive fields of FCNs are provided in section 4.3.
3. Hungarian Cross Entropy (HCE) is introduced in section 3.7 and empirically justified in section 4.4.
4. Finally, the process of creating the Prime Texture Analysis Dataset is described in section 3.8.3.3 and it is used to benchmark a number of architectures in section 4.5.1.

Given that these four additions are designed to be combined into the final workflow, an important decision of research design is that their performance together must also be verified. Therefore, this research makes use of ablation studies to confirm the inter-operability of the above approaches; these ablation studies will be found in their appropriate sections.

1.5 Assumptions, Limitations, and Scope

Given the breadth of the research topic, some assumptions and limitations should be introduced that provide for a more reasonable scope. An in-depth discussion of the research limitations and assumptions is given in section 3.4. Stated succinctly, the research limitations are: the benefit of OTE, TSC, and HCE to non-FCN architectures or to non-image segmentation domains is not considered, self-supervised texture segmentation may not generalise to object proposal in other domains or may require a larger texture dataset, and learnability will not be beneficial enough to justify the extra computational burden of generating a synthetic dataset in all domains.

Further, some assumptions are made prior to conducting experiments. These assumptions made conducting experiments more feasible and kept the scope manageable. In this regard, three assumptions were identified: it is assumed that the OBIA workflow is a sensible approach to image segmentation in low-resource domains and that FCNs are the most sensible starting point for introducing learnability into the OBIA paradigm. Further, it is assumed that increasing the size of the architectures considered in this work may reduce the benefit of dilated convolution and TSC due to the implicit expansion of the receptive field that larger architectures provide. A detailed discussion of each assumption and limitation is provided in section 3.4.

1.6 Summary

This dissertation is organised into five chapters. Following this introduction, a comprehensive literature review describing portions of the wider literature that are essential to a proper understanding of this research is provided. It begins with brief descriptions of the purpose and history of object-based image analysis in section 2.1. It then introduces convolutional and fully-convolutional neural networks in sections 2.2.1 and 2.2.2 and lastly provides a brief introduction to self-supervised learning and some other relevant learning paradigms.

All of the topics discussed as background material in the following chapter have a bearing on the validation of the research goals and hypothesis, which are provided and discussed in section 3.3. A detailed research methodology is also given in chapter 3, along with a description of the four contributions and the proposed framework. The descriptions provided relate to much of the literature introduced in the prior related work section.

Finally, in chapter 4, the results of several experiments and ablation studies are presented. These experiments aim to justify each of the four contributions independently and then benchmark the proposed framework against other options in the field. They also validate the research hypothesis, which states:

> An FCN trained using a synthetic texture analysis dataset can effectively replace traditional segmentation algorithms in the object proposal step of the object-based image analysis workflow with comparable or superior performance while maintaining a degree of generalisability

Lastly, some concluding remarks and a summary of the work are provided in chapter 5.

Chapter 2

Literature Review

2.1 Object Based Image Analysis

2.1.1 Introduction

Object-based image analysis (OBIA) is an umbrella term used to describe a collection of techniques for extracting semantically significant objects from remote sensing imagery [Veljanovski et al. 2011]. The term geo-spatial object-based image analysis (GEOBIA) can be used interchangeably to emphasise the close-ties that the field has to geographic information systems (GIS) [Blaschke 2010]. In this work, the term OBIA is used. OBIA's primary focus is to transcend the traditional pixel-level analysis of images that predominated in GIS in the early 2000s, and, as shown later, this focus has applications broader than GIS.

Over the first two decades of the 21st century, numerous high-resolution satellite images have been released; these include images from IKONOS, QuickBird, and the sentinel-1 and sentinel-2 satellites [Hossain and Chen 2019]. A marked increase in the number of high-resolution datasets captured using unmanned aerial vehicles (UAVs) has also occurred. This high-resolution imagery can have spatial resolutions as fine as 30cm. This is a remarkable improvement considering that popular remote sensors from the 20th century, like the AVIRIS sensor, have spatial resolutions in the range of 6m-30m [Green et al. 1993]. These high-resolution images are frequently being released in more convenient formats [Neumann et al. 2019], as in the Tensorflow datasets module [Google 2019], and more extensive collections, as with BigEarthNet [Sumbul et al. 2019]. These images are also openly available through Google Earth and other services.

The availability of this data presents an interesting and challenging opportunity for researchers. Objects that were once invisible to satellites are now available in relatively crisp view. As stated in Blaschke [2010], this means that geographic features that were once represented entirely in singular or small groups of pixels may now span across many pixels. Since each pixel now covers a far smaller portion of each geographic feature's area, the techniques must be adapted to incorporate the surrounding context of individual pixels. A visual depiction of this problem can be seen in figure 2.1. For this reason, the increase in the popularity of object-oriented approaches to satellite image analysis coincided with this rise in the availability of high-resolution images. Whereas singular pixels may once have classified into general classes based on their spectral and spatial similarity to other pixels, OBIA provides a way to group these pixels into larger objects and then classify them by incorporating spectral, spatial, and shape information [Hossain and Chen 2019].

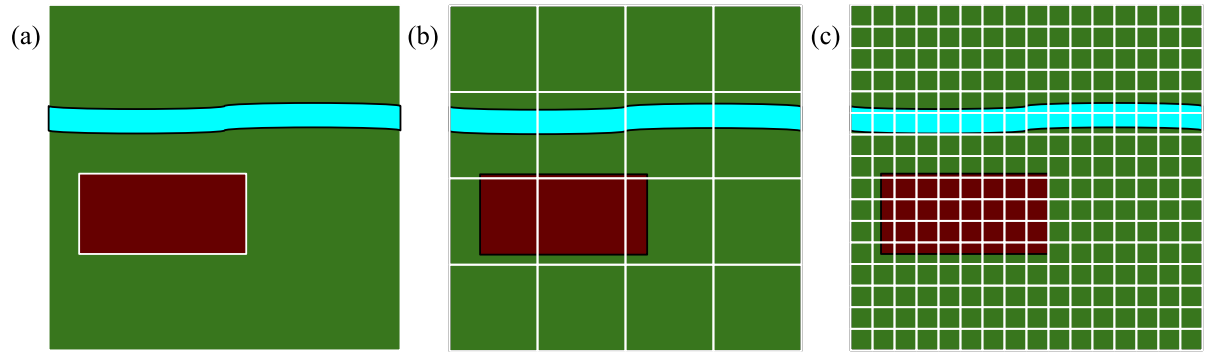


Figure 2.1: Increase in spatial resolution - (a) 5m resolution: pixel-level and sub-pixel-level analysis required. (b) 1.25m resolution: pixel-level analysis with spatial similarity measures necessary. (c) 30cm resolution: object-based and super-pixel level analysis beneficial. Graphic adapted from Blaschke [2010].

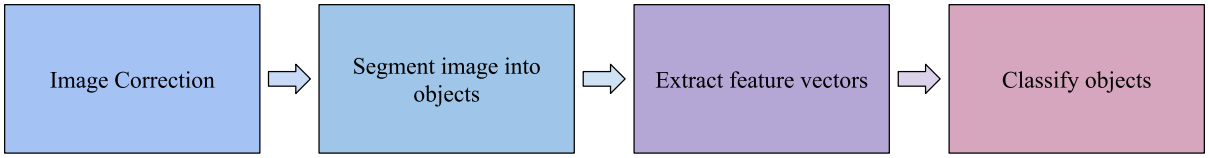


Figure 2.2: Flowchart depiction of the OBIA workflow

While the driving force behind the development of OBIA was the increase in the availability of high-resolution imagery, this framework has numerous other benefits. OBIA adapts very effectively to domains in which limited training data is available, frequently called low-resource domains [Platt and Rapoza 2008]. OBIA also allows for the easy incorporation of expert knowledge in the form of rule sets that apply to particular objects; this benefit relates to OBIA's low-resource adaptability [Veljanovski et al. 2011]. OBIA essentially eliminates the occurrence of salt and pepper noise in classification results, a prevalent and limiting problem in the pixel-level analysis of images [Blaschke 2010]. Lastly, OBIA allows for the selection of more salient feature vectors, which significantly improves the interpretability of this approach [Platt and Rapoza 2008; Veljanovski et al. 2011].

There exists a reasonably large body of work on the topic of object-based image analysis; however, researchers generally make alterations within what is called the OBIA workflow or framework [Blaschke 2010; Veljanovski et al. 2011; Hossain and Chen 2019]. In the next section, a detailed explanation of this workflow is provided.

2.1.2 Workflow

A large portion of the academic literature regarding OBIA prescribes a particular workflow, and this section is dedicated to expounding upon that workflow. The workflow can be adapted modestly to suit the application area. The variation shown here adequately summarises versions used in Luo et al. [2017], Liu et al. [2018b] and Gusella et al. [2005], amongst many others. A flowchart depiction of the OBIA workflow described in this section can be seen in figure 2.2. The workflow contains four steps: image correction, object proposal, feature extraction, and object classification; a subsection will be devoted to the discussion of each.

2.1.2.1 Image Correction

Image correction exemplifies the relationship between OBIA and GIS. This step is not necessarily required when applying an OBIA workflow in other domains. The term image correction is used in other domains to refer to the removal of noise from images or performing image inpainting [Aharon et al. 2006; Mao et al. 2016]. Typically, in OBIA, the term exhibits a different meaning; when capturing geographic scenes using UAVs or satellites, numerous faults appear in the image. The removal of these faults is what the term *image correction* refers to in this context; this process can also be termed orthorectification or geometric correction.

Some faults typically removed from remote sensing scenes include: tilt in the angle of the sensor during flight or orbit, a change in the angle of the sun causing shadows, and the effect that image perspective has on any terrain in the scene [Boccardo et al. 2004]. Once these problems are accounted for in the satellite image, it may be referred to as an orthophoto and suitable for various use-cases. Due to these corrections, the satellite image may be used as a normal planer map, and the true distance between geographic features in the scene can be approximated. This allows us to extract information about the area of objects from the image (if the spatial resolution of the orthophoto is known).

As will be discussed in section 3.8, this research does apply the OBIA workflow to domains outside of GIS; as such, this step also refers to any necessary preprocessing - such as noise removal - that applied to the data. This step is necessary for some domains because the presence of noise or faults in the image will substantially impact the accuracy of the next step - which is the primary focus of this work - object proposal and segmentation.

2.1.2.2 Object Proposal

This step involves grouping pixels in the image into distinct objects. The algorithms that perform these groupings are identified as *segmentation algorithms* and typically fall into one of four categories: pixel-based methods, edge-based methods, region-based methods, and hybrid methods [Hossain and Chen 2019]. Pixel-based methods compare the intensities and coordinates of all the pixels in the image to determine a class label. These methods are relatively naive and are now typically used as a starting point in more advanced hybrid methods. Standard algorithms used in pixel-based methods include naive Bayes, multi-layer perceptrons, and support vector machines; note that these algorithms can also be applied at the superpixel level [Khan et al. 2010]. Pixel-based methods do not have much bearing on the OBIA paradigm, nor do they feature in most contemporary approaches to image segmentation [Hossain and Chen 2019; Ronneberger et al. 2015]; for these reasons, they will not be discussed further here. To adequately evaluate the quality of the proposed method, a representative group of segmentation algorithms is required. In order to elucidate why this research makes use of those chosen, the remaining three categories of segmentation algorithms are itemised below with more detail and a thorough example of each.

1. Edge-based Methods
   As their name suggests, these methods segment objects based on edges that exist within the image. Edges are defined as an abrupt or substantial change in colour or intensity within some locus of pixels [Muthukrishnan and Radha 2011]. The task here differs from that of edge detection as described in

Prince [2012]; here, the aim is not to detect edges as points of interest but instead to detect edges as closed object boundaries. Nevertheless, many of the same techniques prevail.

Edge detection has been widely researched for a number of years, and contributions are made fairly regularly [Muthukrishnan and Radha 2011; Versaci and Morabito 2021]. Despite the abundance of research, two classical approaches - the Watershed algorithm from Vincent and Soille [1991] and the Canny edge detector from Canny [1986] - remain the most widely used gradient-based object segmentation techniques. The Canny edge detector is not designed for object segmentation directly; edge linking is required to convert open edges into closed boundaries, and connected components labelling is required to convert those solid edges into objects. As such, Watershed, an end-to-end gradient-based object segmentation approach that relies on simulated immersion, is considered. A deeper discussion of Watershed will be provided later in this section.

2. Region-based Methods
   Typically this refers to region merging techniques. First, a set of seed points is selected, either according to a grid or local extrema. Then, from these seed points, pixels are merged into regions provided that they satisfy a user-defined predicate [Ikonomatakis et al. 1997]. After this procedure is complete, the detected regions are merged into adjacent regions with similar properties. As a form of noise-removal, small regions may also be ignored or forced into larger regions. A variety of metrics can be defined for measuring the similarity of two regions or pixels, but a common approach is to apply a threshold to the difference of specific pixels, below which they are characterised as similar. The region-based segmentation technique considered in this dissertation is simple linear iterative clustering (SLIC), and a complete description of the algorithm will be provided shortly.

3. Hybrid Methods
   Hybrid methods fuse ideas from pixel-based, edge-based, and region-based methods. This synthesis is quite a difficult task, so they are highly complex methods to implement and require a large amount of parameter tuning. This complexity generally dissuades researchers from implementing them; most turn towards implementations already created in proprietary software solutions [Gusella et al. 2005; Liu et al. 2018b; Mboga et al. 2019]. Multiple reviews of the OBIA literature have established eCognition as the most popular choice in the field [Dronova 2015; Veljanovski et al. 2011; Blaschke 2010; Hossain and Chen 2019]. Some authors, including Blaschke [2010] and Veljanovski et al. [2011], have attributed a non-negligible portion of the growth in the popularity of OBIA research to eCognition.

Figure 2.3: Simple Linear Iterative Clustering Algorithm

Require: Number of clusters K, spatial priority m, number of repeats T, and original image I

BEGIN
Calculate S and assign grid of centroids
Find gradient image G and perturb centroids to local minima
for $t = 1$ to T:
    for each pixel p in I:
        find centroid in neighbourhood of p with minimum **D**, and assign p to that centroid
    reassign each centroid to the average pixel in its cluster
END

Simple Linear Iterative Clustering (SLIC), initially proposed by Achanta et al. [2010], is an algorithm that iteratively groups pixels into clusters, producing a segmentation map. These groups of coherent pixels are

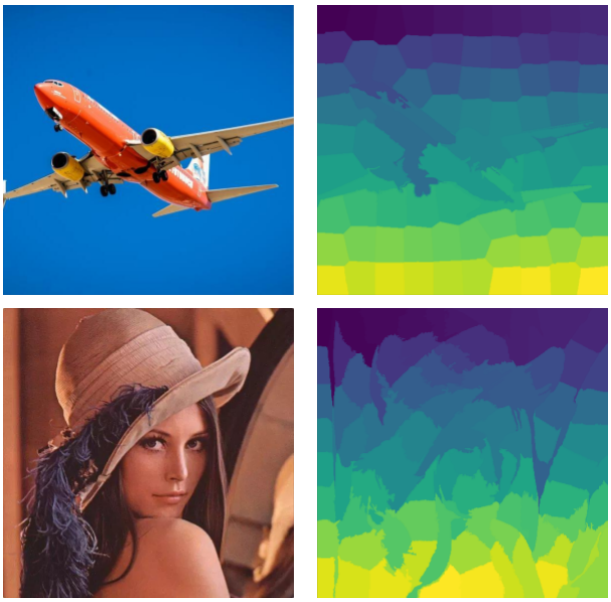


Figure 2.4: Figure contains the original images on the left and SLIC's proposed superpixels on the right

called "superpixels", and the algorithm constructs them in a way analogous to the popular clustering algorithm KMeans. Once obtained, these superpixels can be interpreted as proposed objects or regions for the purpose of OBIA.

The algorithm is initially provided with an image of size $N = H \times W$ and a certain number of superpixels to create, denoted K. It then calculates a scale parameter, $S = \sqrt{\frac{N}{K}}$, and uses it to create a grid of centroids each surrounded by a $2S \times 2S$ neighbourhood. The algorithm proceeds by computing the gradient image with the following equation:

$$G(x, y) = ||I(x + 1, y) - I(x - 1, y)||^2 + ||I(x, y + 1) - I(x, y - 1)||^2$$

for every pixel, (x, y), in the gradient image; where $I(x, y)$ is the grey-scale pixel intensity of pixel (x, y) in the original image. Once computed, every centroid is reassigned so that it is the minimum in its neighbourhood of the gradient image.

Finally, each centroid, $\mathrm{c} = (\mathrm{R_c}, \mathrm{G_c}, \mathrm{B_c}, \mathrm{x_c}, \mathrm{y_c})$, is compared to every pixel, $p = (\mathrm{R_p}, \mathrm{G_p}, \mathrm{B_p}, \mathrm{x_p}, \mathrm{y_p})$, in its $2S \times 2S$ neighbourhood based on a spectral-spatial euclidean distance, **D**, calculated as follows:

$$\begin{aligned} \mathrm{d_{RGB}(c, p)} &= \sqrt{(\mathrm{R_c} - \mathrm{R_p})^2 + (\mathrm{G_c} - \mathrm{G_p})^2 + (\mathrm{B_c} - \mathrm{B_p})^2} \\ \mathrm{d(c, p)} &= \sqrt{(\mathrm{x_c} - \mathrm{x_p})^2 + (\mathrm{y_c} - \mathrm{y_p})^2} \\ \mathbf{D}(\mathrm{c, p}) &= \mathrm{d_{RGB}(c, p)} + \frac{m}{S}\mathrm{d(c, p)} \end{aligned} \tag{2.1}$$

where m is a hyperparameter reflecting how heavily the spatial proximity is prioritised, typically set to $m = 10$ [Lowekamp et al. 2018], R, G, and B are the red, green, and blue intensities of the respective pixels, and x

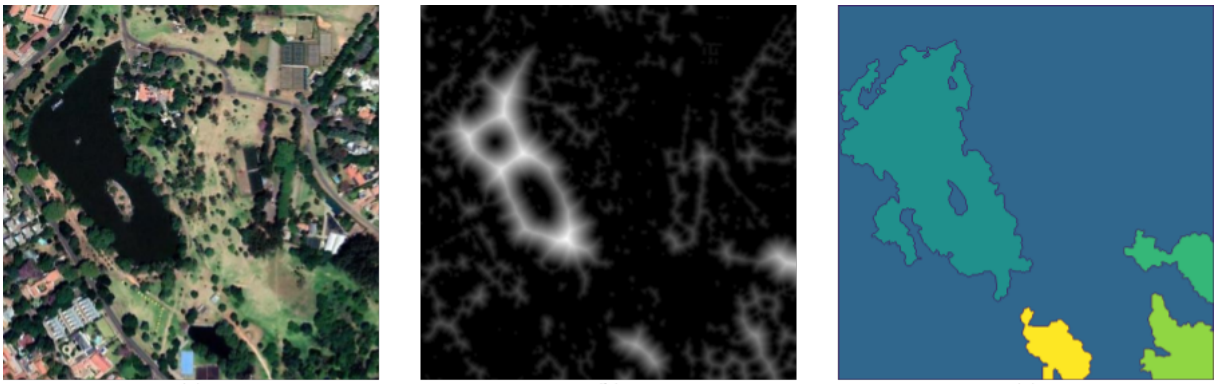


Figure 2.5: Depiction of (a) image of Zoo Lake, (b) distance transform to find basins, and (c) Watershed segmentation

and y are the coordinates of each pixel. Each pixel is assigned to the centroid with which it has the minimum distance and reassign the centroids to the average of each of the superpixels. This process is repeated a fixed number of times or until a state of stability - in which no further assignments are made - is reached.

A pseudo-code description of SLIC is provided in figure 2.3, along with an example of its application in figure 2.4. Despite the algorithm being more than a decade old, SLIC is still frequently applied to domains such as medical imagery [Mohamed et al. 2019], satellite imagery [Qin et al. 2014], and crop recognition [Sadeghi-Tehran et al. 2019]; and new improvements to the algorithm are regularly proposed [Wu et al. 2020].

Watershed [Vincent and Soille 1991] is a classical approach to image segmentation that effectively treats the image as a topographical map, in which the pixel intensities in the gradient image represent the height of the pixel. Watershed is widely used, particularly when efficient and consistent region proposal is required and training data is unavailable. Recently, Watershed was used as a part of an official initiative to segment cells for the detection of SARS-CoV-2, given that large datasets relating to the pandemic are not yet widely available [Pape et al. 2021]. The algorithm's popularity has resulted in a divergence of implementations, but the classical version will be described here. The image is first converted to grey-scale and binarised (thresholding), typically through the use of Otsu's thresholding technique [Otsu 1979]. Once this binarised image is found, a distance image is computed, in which each pixel in the binarised image is replaced with the euclidean distance between itself and the nearest non-zero pixel in the image. This step could also use a gradient vector image, as was done previously in the SLIC algorithm. Now Watershed requires seed points to be recommended, either by the user or according to a rule; one typical approach is to select a certain number of local minima, but seeding is an unsolved problem in image processing, and so, in more complex images, it is often done manually.

The gradient image is considered a topographical map, with each object ideally being a basin. Watershed then iteratively floods the image from the chosen seed points; ideally, each of these seeds occurs at a local minimum. From each seed point, this flooding continually envelops neighbouring pixels until the region comes into contact with another region. The pixel at which this contact occurs is then considered a "dam" or "watershed line", which defines the border between two objects. This flooding or immersion process is repeated until at least one region has visited every pixel in the image. A visualisation of Watershed being applied to an image, along with the distance transform found from the binarised version of the image, can be seen in figure 2.5.

As shown in figure 2.4, SLIC generates hundreds of superpixels which a classifier must filter down at a later stage. Contrary to this, Watershed proposes larger objects in its segmentation maps when it is functioning correctly and is not obstructed by noise. Given that Watershed naturally proposes these larger objects, the two algorithms contrast one another quite effectively and reasonably represent the array of traditional segmentation algorithms currently available.

Using SLIC, Watershed, or another segmentation technique, a collection of objects can be extracted that can then be classified with greater spatial consistency than if they were interpreted on the pixel-level. However, another divergence in terms is now reached; while the final purpose of OBIA is to extract semantically significant objects from an image, for example, buildings, roads, or rivers. The objects that are grouped into pixels into are not necessarily semantically significant [Blaschke 2010]. Two examples of this can be found in figure 2.4. The images on the left are provided for segmentation; each of the objects found in the images is shown in a different colour on the right. Notice that the images are needlessly split into multiple segments. This is not a negative result, nor is it unexpected; what the segmentation algorithm deems a sensible object may not correspond to any semantic interpretation. In OBIA, the term *semantic features* is usually used to describe the semantically significant parts of a scene; this leaves the term *objects* free to describe any grouping of pixels provided by the segmentation step.

This redefinition of terms also expresses the way in which segmentation algorithms in OBIA can be judged. In this context, object segmentation focuses on delineating solid objects that are different from one another. Less regard is given to precision and semantic significance than would otherwise be the case. This more forgiving context may allow us to exploit self-supervision in image segmentation. The segmentation step must simply identify any distinct objects with arbitrary labels, the final steps of the workflow will assign semantic meaning to the located objects.

2.1.2.3 Feature Extraction

After object proposal, but before classification, one has to convert each object into a *feature vector* appropriate for the classifier being used. This step is often considered implicit in the segmentation step discussed in the previous section, but it warrants further discussion. As could be seen in figures 2.4 and 2.5, segmentation algorithms typically locate objects of varying sizes and shapes. Generally, rather than using the full extent of each object, researchers use metrics that adequately encapsulate the objects; thus producing the fixed-size feature vectors most classifiers require [Platt and Rapoza 2008; Veljanovski et al. 2011]. For example, a feature vector comprising the mean, maximum, and minimum pixel intensities present in each object may be sensible for certain tasks. The metrics shown below - sourced from Platt and Rapoza [2008], Bhardwaj and Kumar [2019], and Krig [2014] - are fairly common in OBIA; here they are split into groups based on which aspect of the object they intend to summarise. In the following, O denotes a band of pixels in a particular object, n denotes the number of pixels in the band, and $o_i \in O$ is the pixel intensity of the ith pixel in O.

Colour - summarising the pixel intensities of each object:

6	3	3
6	1	1
3	5	5

1	0	0	0	0	1
0	0	0	0	0	0
0	0	1	0	0	1
0	0	0	0	0	0
0	0	1	0	1	0
0	0	0	0	0	0

Figure 2.6: Example of a grey-level co-occurrence matrix. Left - original image I. Right - GLCM computed from I using equation 2.6 with offset $(1, 0)$.

- Maximum pixel intensity:

$$o|o \geq o_i \quad \forall o_i \in O \tag{2.2}$$

- Minimum pixel intensity:

$$o|o \leq o_i \quad \forall o_i \in O \tag{2.3}$$

- Mean pixel intensity, $\bar{o}$:

$$\bar{o} = \frac{\sum o_i}{n} \tag{2.4}$$

- Variance in pixel intensity, σ_o^2:

$$\sigma_o^2 = \frac{\sum (o_i - \bar{o})^2}{n} \tag{2.5}$$

Texture - summarising gradients and relations between the pixel intensities of the object. One frequently used form of texture analysis is the grey-level co-occurrence matrix (GLCM); these matrices represent the frequency with which the intensities of pairs of pixels correspond with one another [Platt and Rapoza 2008; Gebejes and Huertas 2013]. These pairs are located according to a predefined spatial relationship, referred to as an offset. Given a $n \times m$ image patch $I(x, y)$ with p different intensities and an offset $(\delta x, \delta y)$, the $p \times p$ GLCM of I, $\mathrm{G}_{(\delta x, \delta y)}(i, j)$, may be represented mathematically as:

$$\mathrm{G}_{(\delta x, \delta y)}(i, j) = \sum_{x=1}^{n} \sum_{y=1}^{m} \begin{cases} 1 & \text{if } I(x, y) = i \text{ and } I(x + \delta x, y + \delta y) = j. \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

A visual representation of a GLCM can be seen in figure 2.6. An abundance of information can be extracted about an image's texture using GLCMs, some of which is presented below. In all of the following, (i, j) refers to the coordinates in the GLCM being considered:

- GLCM contrast - a low value suggests a texture with a smooth gradient and vice versa. Computed using:

$$\text{contrast}(G) = \sum_{(i,j)} |i - j|^2 G(i, j) \tag{2.7}$$

- GLCM energy - uniform textures will tend towards higher values, found with:

$$\text{energy}(G) = \sum_{(i,j)} G(i,j)^2 \tag{2.8}$$

- GLCM homogeneity - given by the equation:

$$\text{homogeneity}(G) = \sum_{(i,j)} = \frac{1}{1-(i-j)^2} G(i,j) \tag{2.9}$$

  Homogeneity reflects how repetitive the texture is, i.e. whether a pattern holds within the texture.

Shape - summarising the spatial relation of the pixels within the object and their relations to the pixels outside the object:

- Area - a count of the pixels within the object:

$$\text{Area}(O) = n = \sum_{o_i \in O} 1 = |O| \tag{2.10}$$

- Perimeter - a count of pixel edges on the border of the object, denoted:

$$\text{Perimeter}(O) \tag{2.11}$$

- Shape Index - attempts to encapsulate the ratio of perimeter to area in a manner reflective of the object's geometry. Using:

$$\text{SI}(O) = \frac{\text{Perimeter}(O)}{\text{Area}(O)} \tag{2.12}$$

  is not sufficient. Objects with larger perimeters relative to their area would have a greater shape index; this reflects their greater complexity and is desired. However, equation 2.12 does not effectively deal with objects of the same shape and differing sizes. Consider a 3×3 square, with area 9 and perimeter 12, and a 4×4 square, with area 16 and perimeter 16; note that corner pixels count twice in the perimeter because they have two exposed edges. Using equation 2.12, their respective shape indices are 1.33... and 1.0, despite having the same shape. Thus, consider:

$$\text{SI}(O) = \frac{\text{Perimeter}(O)}{4\sqrt{\text{Area}(O)}} \tag{2.13}$$

  from Platt and Rapoza [2008] and:

$$\text{SI}(O) = \frac{\text{Perimeter}(O)}{\sqrt{4\pi\text{Area}(O)}} \tag{2.14}$$

  from Bhardwaj and Kumar [2019]. The latter of which outputs 1.1283... for both squares and the former 1.0, reflecting that the squares are geometrically identical. In this research, this research refers to equation 2.14 due to the comprehensive review of the index conducted in Bhardwaj and Kumar [2019].

2.1.2.4 Classification and Clustering

Once objects have been segmented from the image and feature vectors have been designed, they may be input into a classifier to filter extraneous proposals. The quality of the segmentation provided will have a substantial effect on the quality of the final classification, and so this step cannot be tested in isolation. Until recent years, there was a predominance of classical classifiers, such as nearest neighbour and random forest, in this step of the OBIA workflow [Blaschke 2010; Veljanovski et al. 2011; Gusella et al. 2005; Xie et al. 2008; Platt and Rapoza 2008]. This predominance is likely due to the fact that these classifiers were present in eCognition, a proprietary software solution which was discussed in section 2.1.2.2.

As alternatives to these classical approaches, Liu et al. [2018b] tested patch-based deep convolutional neural networks, input with adaptive windows wrapping around segmented objects, and returning a one-hot encoding corresponding to a class label. Both Mboga et al. [2019] and Liu et al. [2018b] also implemented fully-convolutional networks in slightly different ways. Liu et al. [2018b] used a more standard approach in which proposed objects were wrapped with adaptive windows and input to an FCN, which updated the pixel labels. Mboga et al. [2019] performed both FCN image segmentation and a more standard OBIA segmentation, overlaid the two results and conserved only points of agreement; this corrected for the tendency of the FCNs to propose objects with smooth and curved boundaries since sharp decision boundaries are usually preferable.

The classification procedures discussed hitherto have required the selection and annotation of training sets. Unsupervised learning has occasionally been explored to circumvent the annotation process in OBIA. Lucieer and Lamarche [2011] used a more advanced variation of the kMeans algorithm, fuzzy c-means (FCM), in the classification step of OBIA. Given that the primary focus of this research is the segmentation step of the OBIA workflow, this research will also make use of clustering algorithms to filter proposed objects.

Figure 2.7: KMeans algorithm

Require: Number of classes K and set of inputs X
BEGIN
Randomly initialise centroids $c_1 ... c_k$ in C
loop until no change in C:
    for each x in X:
        find closest centroid c to x $\min_j(c_j, x)$
        assign x to c
    for each c in C:
        move c to mean of all X assigned to c in previous step
END

KMeans [MacQueen et al. 1967] is a popular approach to clustering that was alluded to earlier in section 2.1.2.2 during the discussion of the SLIC algorithm. Only a brief explanation is provided here, and pseudocode for the algorithm can also be seen in figure 2.7. A set of centroids are randomly initialised, and each pixel in the image is assigned to the centroids it is most similar to, as defined by a distance metric. The metric introduced in equation 2.1 would suffice. After all the assignments are complete, the centroids are updated to the centre (mean) of all the points in their respective clusters. As with SLIC, this process is repeated a fixed number of times or until there is minimal change in the position of each of the centroids. KMeans makes some limiting assumptions about the data [Aharon et al. 2006]. Most notably, the algorithm defines an ellipsoidal shape

around each centroid; if a sample falls within that ellipsoid, it is considered a member of the cluster that that centroid defines, and so kMeans is only capable of providing clusters that are convex in whatever feature space is being considered.

Spectral clustering (SC) [Ng et al. 2002] addresses this problem of convexity by clustering with respect to a mathematical graph. To achieve this a graph is first constructed from the data, the graph can be constructed using any technique but KNN is certainly the most common [Weiss 1999]. A graph containing N vertices, corresponding to N datum $\{S_i\}_{i=1}^{N}$, can be represented as an $N \times N$ affinity matrix. In this affinity matrix each entry (i, j) represents the weight of the edge between vertex i and vertex j, each of these weights is referred to as $W_{i,j}$. From an $N \times N$ affinity matrix N elements d_i can be obtained, each with the equation $d_i = \sum_{j=1}^{N} W_{i,j}$ for $i \in \{1, 2, ..., N\}$, these define the diagonal entries of the degree matrix. Finally, the $N \times N$ Laplacian matrix, $\mathbf{L}$, is constructed using the following rule from Ng et al. [2002]:

$$\mathbf{L}_{i,j} = \begin{cases} d_i & : i = j \\ -W_{i,j} & : \text{otherwise} \end{cases} \tag{2.15}$$

The goal in constructing the Laplacian matrix is to perform graph partitioning, the Laplacian matrix has useful properties to facilitate this. First the Laplacian matrix is decomposed into eigenvalues and eigenvectors, for an $N \times N$ Laplacian matrix N eigenvalues will be obtained, $\{\lambda_1, \lambda_2, ..., \lambda_N\}$, along with N corresponding eigenvectors, $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. The eigenvalues and eigenvectors are sorted in ascending order of eigenvalue, so $\lambda_1 < \lambda_2 < ... < \lambda_N$, this is called the "spectrum" of the Laplacian matrix [Weiss 1999].

Eigenvalues appearing earlier in the spectrum correspond to points of lower connectivity in the affinity graph. Each of those eigenvalues corresponds to an N-dimensional eigenvector, and each of the N entries in each eigenvector correspond to a datum in the original data. The data is clustered using these eigenvectors. Let $\mathbf{x}_{i,j}$ represent the j-th entry in the i-th eigenvector. Each of the datum S_i can be represented using a portion of each of the eigenvectors, denoted $(\mathbf{x}_{1,i}, \mathbf{x}_{2,i}, \mathbf{x}_{3,i}, ..., \mathbf{x}_{N,i})$. When deciding which of the eigenvectors to use in this representation, two factors must be considered. First, if the original affinity graph was fully-connected, then the first eigenvector (corresponding to the lowest eigenvalue) need not be considered as its components are constant. Second, the number of subsequent eigenvectors (not counting the first) should be one less than the number of desired clusters. Using this final representation of the data as input into the KMeans algorithm in figure 2.7 will typically provide a better clustering than it would if this step were skipped [Bruton and Wang 2021].

This workflow is not observed in all approaches to the segmentation of objects in images. Indeed, in certain scenarios, this approach to semantic segmentation is not needed. In high-resource domains, with a plethora of training data available, fully-learned approaches such as the convolutional autoencoders, which will be discussed in section 2.2, may be preferred. That notwithstanding, Platt and Rapoza [2008] aimed to explore the question of which methods used in the workflow contribute most to the increased accuracy observed in OBIA over pixel-based approaches. They came to the conclusion that:

> "we cannot attribute the advantage of the object-oriented paradigm to any one of these methods in isolation."

This idea will be expanded upon over the course of the dissertation - however, it is reasonable to suggest that treating the steps of noise-removal, object segmentation, feature selection, and object classification in isolation may improve the overall result of an approach to image segmentation. It is also likely that each of these steps is simpler to learn in isolation than jointly. Other benefits include applicability to low-resource domains, the ability to incorporate expert knowledge at various stages throughout the workflow, the compartmentalisation of the models and algorithms created for each step, and a reduction in the difficulty of interpreting the models used at each step.

The proposed research focuses on improving the second step of the OBIA workflow, which deals with segmentation, through the application of self-supervised fully convolutional neural networks. Instead of using the algorithmic segmentation techniques that characterise that step, neural networks could make for a learnable alternative. This discussion will continue in section 2.2 after a brief interlude discussing some of the problems to which OBIA has been successfully applied.

2.1.3 Applications

The frequent use of morphological operations, which require parameter-tuning, in the second step of OBIA excludes it from application to large datasets and reduces its generalisability. With that said, OBIA's performance in low-resource domains is essentially unmatched. The workflow is designed for application to entirely novel satellite images with concrete use-cases and removes reliance on cartographers to perform these tasks.

Xie et al. [2008] detected Australian Pine trees in newly captured satellite images of the South of Florida, USA and attained virtually perfect accuracy. Gusella et al. [2005] estimated the extent of damage to the city of Bam in south-east Iran from an earthquake that occurred in 2003. They detected buildings in a high-resolution satellite image captured by QuickBird before the earthquake, overlaid it with images captured shortly after the earthquake, and then located points of high change. Mboga et al. [2019] performed land-use land-cover classification for urban mapping in the Congo and attained 91% accuracy. The focus of OBIA is producing usable, real-world results as opposed to surpassing impressive benchmarks. Improvement of the techniques used within the OBIA workflow could have palpable effects downstream.

2.2 Neural Networks

2.2.1 Convolutional Neural Networks

Neural networks have formed an integral part of computer science research for a number of years now, and so a complete introduction to them will not be provided here. It suffices to say that a neural network is a system formed by layers of interconnected perceptrons, with each layer in the network learning more abstracted features than its predecessors [Nielsen 2015]. Numerous variations to this original structure have been researched over the past two decades, one of the most successful variants being the convolutional neural network (CNN).

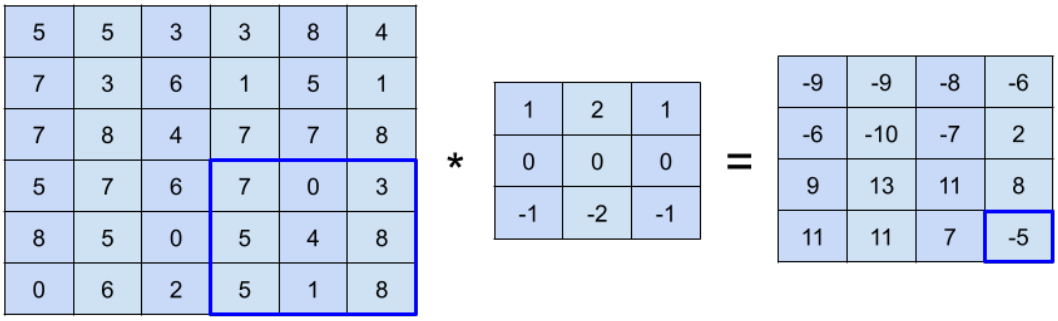


Figure 2.8: Example of image (left) convolution using a horizontal Sobel filter (center).

CNNs pass convolutional kernels over the image that compute a weighted sum for the neighbourhood around each pixel. Many of the features discussed later in section 2.1.2.3 can be described as filters and require painstaking refinement to create manually; CNNs provide a way to learn the most applicable filters directly from the data [Rawat and Wang 2017; Gao et al. 2010]. The standard mathematical definition of the term *convolution* applied at pixel coordinates (i, j) is:

$$C(i,j) = \sum_{x=-k}^{k} \sum_{y=-k}^{k} h(x,y) I(i+x, j+y) \tag{2.16}$$

where I is the original image, h is the $(2k+1) \times (2k+1)$ filter being applied, and $C(i,j)$ is the resultant matrix indexed at element (i,j). In this definition, the centre of the filter is at position (0, 0), and the top-left border pixel of the image is at position (-k, -k), to account for the down-sampling or padding of the image.

When working with CNNs, the matrix that results from a convolutional operation is often referred to as a feature map; this reflects the ability to interpret which features in the image are perceived by the applied filter. Zeiler and Fergus [2014] found that filters earlier in the network tend to learn low-order features, such as lines and edges, and filters deeper in CNNs tend to learn more abstract representations of objects and colour. In the wider mathematical field, the operation defined in equation 2.16 is often referred to as *cross-correlation* – in this context, the notions of convolution and cross-correlation are essentially equivalent; whether the filter is rotated $180°$ will not affect the CNN's learning of the weights.

An illustrative example of image convolution is provided in figure 2.8. As can be seen here, the process of image convolution computes the element-wise product of the filter with each locus of pixels being considered and stores the result in the feature map. The convolutional stride is also defined as the step size of the kernel as it convolves over the image; figure 2.8 shows convolution with a stride of 1 because the filter moves in steps of 1. If the filter instead skipped every second element, then the stride would be 2. This same kernel is passed over the entire image in a sliding-window fashion. Regardless of where the kernel is in the image spatially, the values in the feature map are the same given the same set of pixel intensities. This quality is referred to as *translational equivariance*.

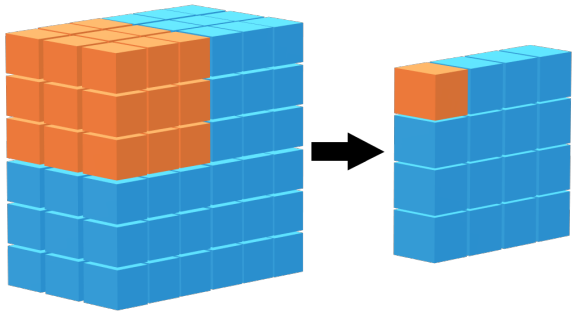


Figure 2.9: Example of 3D convolution using a 3×3 filter on a $6 \times 6 \times 3$ image

It is also important to discuss how CNNs apply 2D convolution to images with multiple channels; figure 2.9 is a graphical rendition of 2D image convolution across multiple channels. Researchers typically disregard the number of channels when describing filter size; for example, the filters shown in figures 2.8 and 2.9 may both be referred to as 3×3 filters. This liberty is taken because each filter only produces a singular feature map, regardless of its depth, and a layer deeper in the network will contain as many channels as the previous layer has filters, often more than 50. Thus, only considering the spatial dimensions of the filter allows us to refer to filters independently of the layer in which they are used.

The same filter is not applied to each channel; each channel receives its own set of weights. A 3×3 filter applied to an image with one channel will contain 9 weights and a $3 \times 3(\times 3)$ filter convolving over three channels will contain 27 weights. The weights of each of the filters form the set of trainable parameters in the CNN. As with standard neural networks, each parameter is updated towards a local minimum in proportion to its contribution to the overall loss of the network, using *backpropagation* and the *stochastic gradient descent* algorithm with update rule:

$$w_j \leftarrow w_j - \alpha(\nabla_w L(w).\hat{w}_j) \tag{2.17}$$

where w represents the jth filter in the network, α is the *learning rate*, and L is a loss function. The derivative of the loss function, $\nabla_w L(w).\hat{w}_j$, found using the chain rule, represents the direction and size of the ascent in the loss landscape. Moving in the opposite direction allows us to descend the loss landscape towards a local minimum. The loss function should reflect the model's competence in the given task and is an essential consideration. Before proposing and justifying the augmented loss function designed in chapter 3, a brief description of the other loss functions considered in section 2.2.3 is provided here.

CNNs began to gain mainstream popularity when Krizhevsky et al. [2012] used AlexNet, a CNN much larger than had been created before, to claim victory on the ImageNet benchmark [Deng et al. 2009]. Since then, researchers have made many improvements, such that modern CNNs outperform humans on numerous image classification tasks and frequently achieve near-perfect accuracy. CNNs have also been altered to perform not just image classification but also image segmentation. Fully Convolutional Neural Networks are one variation of CNNs that excel at image segmentation, and they will be introduced in section 2.2.2. First, some common techniques used within CNNs and FCNs need to be introduced since the understanding of these techniques had significant bearing on this research.

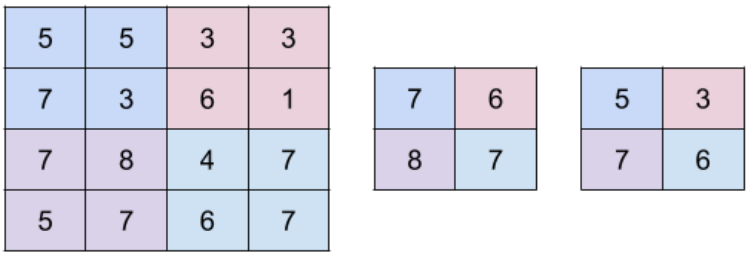


Figure 2.10: Examples of pooling - original image (left), max-pooling (centre), and average-pooling (right).

2.2.1.1 Pooling

As can be seen in figures 2.8 and 2.9, convolution is typically considered a down-sampling operation. By ignoring border pixels that the kernel cannot be centred on, it reduces the spatial dimensions of the image. This mild down-sampling may be avoided by padding the image such that the kernel can centre on the border pixels; however, the information lost in this spatial reduction is typically not significant. Down-sampling is often considered beneficial because it reduces the memory requirements of the network and reduces its propensity to overfit. When applying a $k \times k$ kernel to an $m \times m$ image, without padding and with a stride S, the size of the output layer can be found using:

$$\lfloor \frac{(m-k)}{S} + 1 \rfloor \tag{2.18}$$

With a low stride, the reduction in size is negligible, particularly in high-resolution images. Researchers have thus developed strategies to increase the speed at which this down-sampling occurs; one such strategy is called *pooling*. The aim of pooling is to reduce the spatial complexity of the image while minimising information loss [Nagi et al. 2011]. Different approaches to pooling exist, but the intention is always to encapsulate the information of a locus of pixels into a single pixel; figure 2.10 shows a graphical illustration of two types of 2×2 pooling. As their names suggest, max-pooling (centre) preserves the maximum intensity, and average-pooling (right) computes the average intensity for each locus of pixels, respectively. However, average-pooling is less computationally efficient due to the division operation, and in-practice provides negligible information gain; most researchers default to implementing max-pooling or min-pooling depending on the use-case [Nagi et al. 2011; Ronneberger et al. 2015].

Since pooling is performed according to a fixed rule instead of being learnable, it is immune to overfitting. It also does not add learnable weights to the network, nor is it a computational burden. However, this lack of learnability is less adaptable and expressive, as will be shown in section 4.3. The down-sampling pooling provides is widely used in the context of fully convolutional neural networks, and it is returned to in section 2.2.2, but it is not without its costs and limitations.

2.2.1.2 Receptive Fields

The "receptive field" of a pixel or unit in a layer of the network is defined as the region of that layer's input that non-negligibly impacts the unit in question. This amounts to the region of the feature map covered by the

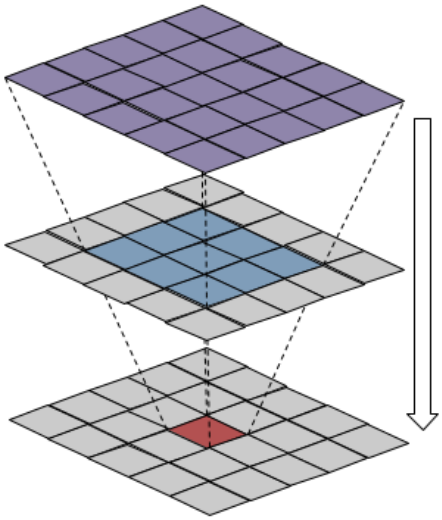


Figure 2.11: This diagram depicts the effective field of a network using 3×3 convolution in purple and the receptive field in blue for the pixel in red

kernel that provided that unit's value. Propagating this concept back through an entire network, one can find the region of the original input to the network that affected the value of the unit under consideration. That region of the original input is called the effective receptive field (ERF), or just the "effective field" [Luo et al. 2016]. Both of these notions are important for a proper conceptualisation of this work, and so an illustration is provided in figure 2.11.

The ERF of the network often determines which tasks the network can undertake. For example, tasks that are more reliant on the shape of objects than their texture, or tasks that require the detection of very spatially dispersed objects, may be impossible without effective fields large enough to include the relevant features of the input space. There are several ways to increase the ERF of a network:

1. increasing the depth of the network,
2. resizing the input image,
3. using larger convolutional kernels,
4. using pooling,
5. strided convolution or strided pooling,
6. and dilated convolution

Options 1-4 are all fairly common techniques for neural network architectures to apply, so limitations tied to the effective field of the network are often dealt with implicitly and inefficiently by making the network deeper or larger than would otherwise be necessary for the task at hand. Dilated convolution is the only approach designed specifically to increase the receptive field while minimising the number of extra parameters added to the network, although one major disadvantage is its slower computational speed. The advantages and drawbacks of each approach will be discussed below, and an alternative is introduced in section 3.6.

2.2.1.3 Dilated Convolution

Increasing the depth of the network, resizing the input image, using larger convolutional kernels, and using pooling were all mentioned in section 2.2.1.2 as methods that increase the receptive fields of convolutional neural networks. All of these approaches have drawbacks; increasing the depth of neural networks often leads to vanishing gradients and over-fitting, besides a general increase in computational burden. Pooling and downsizing the input image can dramatically decrease the dimensionality of the input data and can result in information loss. They also manipulate the data in a fixed way as opposed to learnable techniques, which are more adaptive. While using larger convolutional kernels does increase the receptive field, it has been shown that larger convolutional kernels are disproportionately computationally expensive [Szegedy et al. 2016]. Architectures with more layers and smaller kernel sizes are more efficient despite having the same number of parameters, which explains the tendency of contemporary CNN architectures towards smaller kernel sizes. Strided convolution provides a learnable way to increase the ERF of an FCN; however, it typically replaces strided pooling, which means an increase in the number of parameters and a greater propensity to overfit. Dilated convolution is the most popular way to increase the receptive field of neural networks as it does not affect the input data or the number of parameters in the network.

Dilated convolution [Holschneider et al. 1990] provides a way to maintain the kernel size while increasing the receptive field of convolutional kernels in a memory-efficient way. The *dilation rate* is also defined as the horizontal, vertical, or diagonal distance from each weight to its nearest neighbours in the filter. Traditional convolution, shown on the left in figure 2.12, is said to have a dilation rate of 1. Examples of dilated convolution with dilation rates of 2 and 3 are provided in sub-figures (b) and (c) in figure 2.12 as well. As shown by Yu and Koltun [2015] and Oord et al. [2016], dilated convolution out-performs traditional convolution in a variety of cases.

While dilated convolution does not require an increase in the number of parameters used in each kernel and so has similar memory requirements to traditional convolution, it does create gaps in the kernel - this can be seen in figure 2.12. These gaps mean that the optimisations generally applied to matrix multiplication are not effective when dilated convolution is being used; because of this, dilated convolution often takes up to three times longer than traditional convolution despite not using more memory. This drawback is reflected upon further - and contrasted to other approaches - in section 4.3.3.

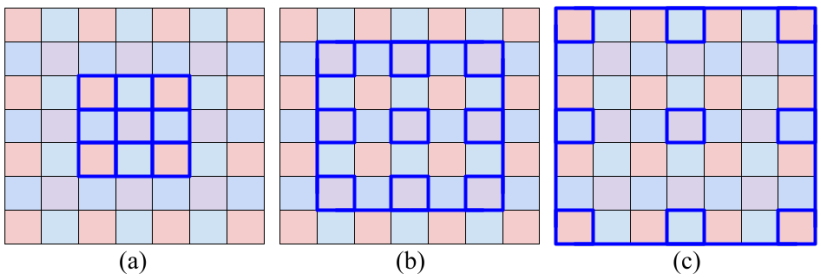


Figure 2.12: From left to right, 3×3 convolutional kernel with (a) dilation rate of 1, (b) dilation rate of 2, and (c) dilation rate of 3

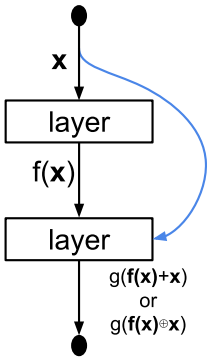


Figure 2.13: Illustration of a skip connection

2.2.1.4 Skip Connections

Skip connections [He et al. 2016] (residual blocks) are one innovation that contributed to the rise in popularity of deep CNNs. They serve to counteract the problem of deeper networks increasing training error through continually vanishing gradients. By connecting layers in the network directly to the inputs of layers that occurred earlier in the network, the tendency for weights and biases to degrade the inputs of deeper layers is curtailed. As weights deeper in the network get smaller, these residual blocks may easily tend towards the identity function instead of tending to an output of zero, particularly when the ReLU activation function is used. Thus, skip connections allow us to add layers to CNNs almost indiscriminately; if the layers are of no benefit, then the network's output will not be altered.

An illustration of a skip connection is shown in figure 2.13. Two different implementations are depicted in the illustration, additive and concatenative. The first, $g(\mathbf{f(x)} + \mathbf{x})$, is refered to as an "additive skip connection". This is simply the element-wise summation of the output of one layer with the input of an earlier layer in the network. This is how skip connections were originally implemented in ResNet by He et al. [2016]. Despite some information loss, this approach still effectively addresses gradient degradation and does not require any extra channels to be added to the filter, and therefore no extra parameters.

More recent implementations of skip connections often make use of the second form, $g(\mathbf{f(x)} \oplus \mathbf{x})$, where "$\oplus$" refers to the concatenate operation. In this research, the second form is referred to as "concatenative skip connections". Instead of element-wise summation, concatenative skip connections concatenate the output of one layer in the network with the input of an earlier layer along their channels (depth), typically doubling the number of channels in the output. As mentioned in section 2.2.1, 2D convolution applies a different filter to every channel of the input feature map, and so this doubling in the number of channels also requires a doubling in the number of parameters in that layer. While this additional memory requirement is costly, concatenative skip connections can lead to better performance, given that there is less information loss and more expressivity. Additionally, because the features are entirely reused, concatenative skip connections often reduce gradient decay even more effectively than additive skip connections. Concatenative skip connections have been used in InceptionNet [Szegedy et al. 2015], DenseNet [Huang et al. 2017], and in some implementations of V-Net [Milletari et al. 2016].

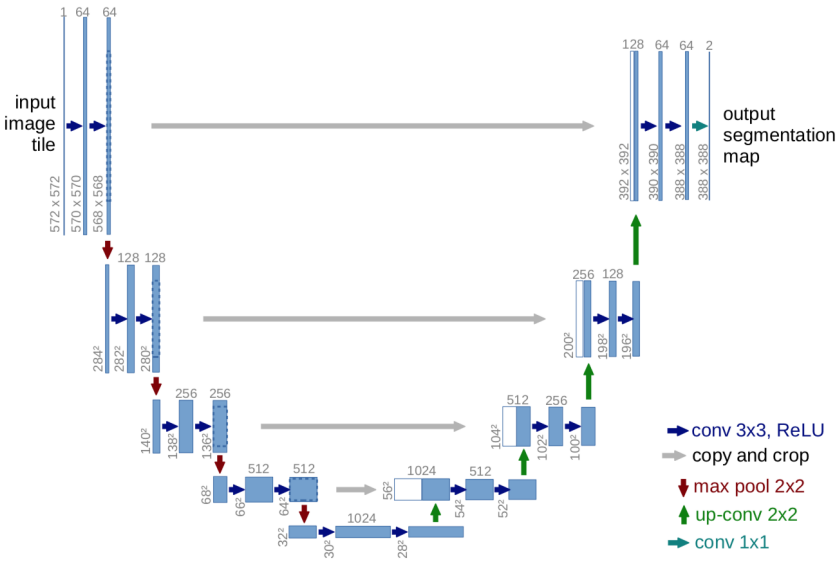


Figure 2.14: Architecture diagram of U-Net – from Ronneberger et al. [2015]

Which form of skip connection an architecture makes use of is typically just an implementation detail dependent on whether researchers require a simpler, less computationally expensive method or a marginally more expensive approach that offers slight performance gains. In this work, both forms of skip connections are used, and in section 3.6 a module for fully convolutional neural networks is proposed in which the choice of skip connection is a more fundamental consideration.

2.2.2 Fully Convolutional Neural Networks

2.2.2.1 Introduction

Fully convolutional neural networks (FCNs) are CNNs comprised exclusively of convolutional layers. Typically FCNs are described in two phases, down-sampling and up-sampling. The first phase decreases the spatial complexity of the input but typically increases the number of feature maps (channels). The second phase then up-samples the input to its original spatial dimension, often using transposed convolution. This process is analogous to autoencoders – that is, neural networks with equally sized input and output layers; and, indeed, convolutional autoencoders could be considered a subset of FCNs. However, an RGB image may be input to an FCN and require that it produce a segmentation mask, thus leaving input and output layers with drastically different sizes. This segmentation mask will have the same spatial resolution as the original image, but each pixel intensity will be replaced with a vector or one-hot encoding. The vector will have the same length as the number of possible labels the network can assign to the pixel, with each value being a probability and the entire vector summing to one. The index of the largest probability is the label value predicted by the network.

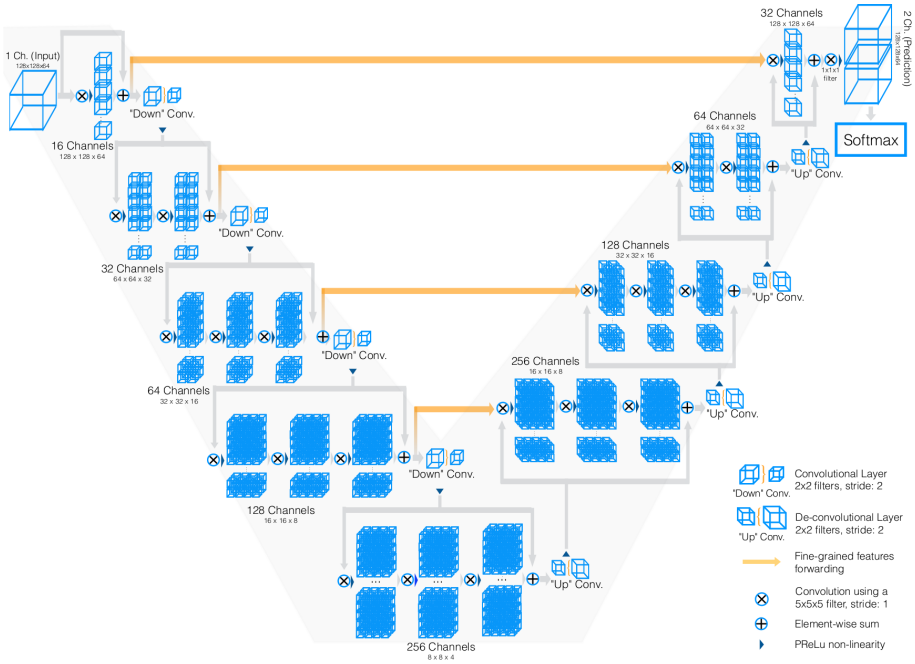


Figure 2.15: Architecture diagram of V-Net – from Milletari et al. [2016]

Ronneberger et al. [2015] devised an FCN architecture for medical image segmentation that has since gained in popularity and been applied to other machine learning problems. They dubbed this network U-Net due to the distinctive shape of the architecture diagram, shown in figure 2.14. U-Net spans 27 layers and, depending on the number of filters chosen for each layer, upwards of 10 million parameters; U-Net's layers are grouped in trios with two 3×3 convolutions followed by a single 2×2 operation. The latter of which handles the down and up-sampling; the 2×2 operation alternates from strided max-pooling, introduced in section 2.2.1.1, to up-convolution (fractionally strided convolution) in the down and up-sampling phases, respectively. U-Net also contains four skip connections between the decoder and encoder layers, allowing the network to retain some of the information it would otherwise lose in the down-sampling phase. As alluded to earlier, that down-sampling phase uses 2×2 strided max-pooling, which halves the spatial size of the input at each layer. This strided max-pooling is not the only way down-sampling could be performed; indeed, U-Net's design has inspired the development of similar, more learnable architectures.

V-Net is a U-Net inspired architecture that makes some significant alterations to the original design that allow it to be applied to multi-spectral imagery. Figure 2.15 contains V-Net's architecture diagram for reference, but the change this research wishes to highlight is the use of strided convolution, instead of strided max-pooling, for downsampling. This research uses versions of both U-Net and V-Net as benchmarks for approaches to increasing the receptive fields of neural networks in section 4.3. Additionally, their juxtaposition allows us to draw interesting conclusions regarding the bias-variance trade-off.

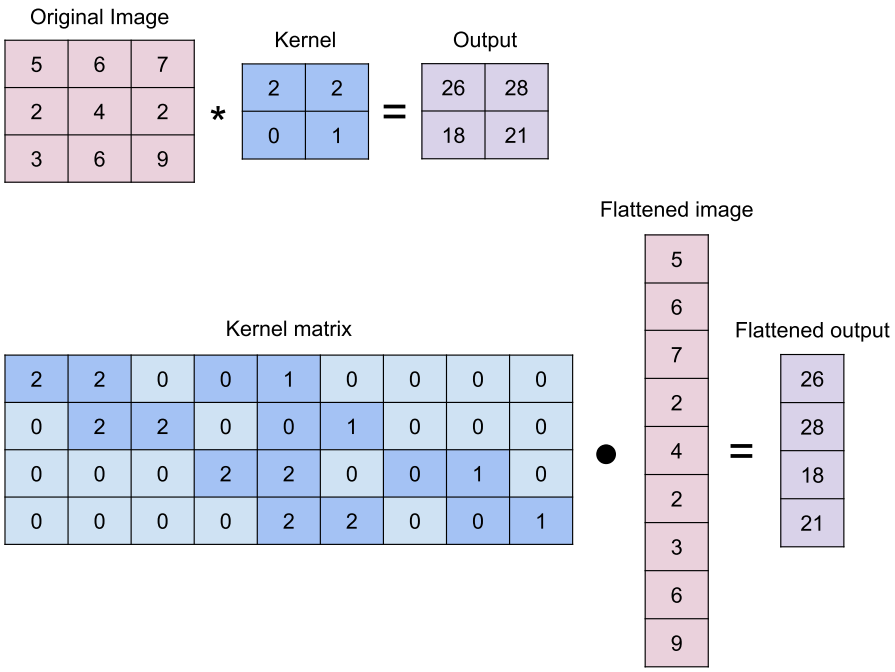


Figure 2.16: Convolution as matrix multiplication – Standard 2×2 convolution (top), same convolutional operation rephrased as matrix multiplication (bottom).

2.2.2.2 Transposed Convolution

The down-sampling that occurs in the encoder portion of FCNs naturally leads to questions about the reversibility of the operation (up-sampling). This learned up-sampling may also be used to enhance low-resolution images. It is worth noting that the goal is not the perfect reversal of convolution or pooling, but rather the ability to *learn* a layer that may increase the spatial resolution of its input [Dumoulin and Visin 2016].

The solution simply requires us to rephrase the process of convolution as one of matrix multiplication using convolutional matrices. A visual example can be seen in figure 2.16. The kernels used in a standard convolutional operation can be converted into kernel matrices. Right multiplying said matrices by a flattened version of the input image yields a flattened feature map. After reshaping, this feature map is identical to the one obtained by performing the convolution operation. Given a kernel matrix K and a flattened image I, the equation is as follows:

$$K \cdot I = F \tag{2.19}$$

where F is a flattened feature map. Now one can left-multiply both sides of equation 2.19 by the transpose of the kernel matrix:

$$K^{\mathrm{T}} \cdot K \cdot I = K^{\mathrm{T}} \cdot F \tag{2.20}$$

In equation 2.20, $K^{\mathrm{T}} \cdot K$ is invertible and as such the equation can be reduced to convolution with a different kernel matrix, K_2, so:

$$K_2 \cdot F = I \tag{2.21}$$

Due to the use of the transpose operation, this is often termed *transposed convolution* but the terms up-convolution and fractionally strided convolution also refer to this procedure [Long et al. 2015; Dumoulin and Visin 2016].

2.2.2.3 Prediction Refinement

Especially on small datasets or anomalous samples in a dataset, an FCN may produce a segmentation map with a certain amount of salt-and-pepper noise or just very small objects [Andrearczyk and Whelan 2017; Mboga et al. 2019]. These artefacts massively reduce the number of conclusions that can be drawn from a segmentation map and should ideally be eliminated. Some algorithmic approaches, like dilation or median filtering [Prince 2012], can perform quite effectively, especially in binary images. However, in a multi-class setting, they are unlikely to reassign a small object or singular pixel to the correct class.

Andrearczyk and Whelan [2017] devised a form of segmentation map refinement that can remove these artefacts in the map and that takes advantage of the predictions made by an FCN. As mentioned in section 2.2.2.1, FCNs produce a vector of probabilities for each pixel, in which each value of the vector represents the confidence that the pixel is a part of the class corresponding to the value's index in the vector. Particularly in small objects and salt-and-pepper noise, the FCN is likely to be somewhat split between at least two classes in its prediction vector - that is, the probability values for those two or more classes will be close together in the prediction vector.

Figure 2.17: FCN Refinement Algorithm

```
Require: Number of patches to target N, number of classes C, FCN prediction vectors for each pixel
BEGIN
SET N̂ to the number of patches in the image
SET previous to current image
loop until N̂ <=N:
    for each class C:
        assign all pixels in small patches to next best prediction for that pixel
        if current!=previous:
            find new N̂ and its corresponding small patches
            break back to while loop
END
```

The algorithm is fairly intuitive; one begins by selecting the N largest objects in the image by their pixel area. Andrearczyk and Whelan [2017] set N to the number of classes the network can predict, which is a sensible choice for the Prague Texture Analysis dataset they are designing for, which is described in section 3.8.2.2. Selecting a higher or lower number could be sensible for alternative datasets in which the number of objects in each image is less consistent, so this is a parameter tuning decision. The remaining objects in the image are considered isolated regions. The pixels in these isolated regions are iteratively replaced with the class with the next highest probability value. A pseudocode implementation of the algorithm can be seen in figure 2.17. Despite its simplicity, this mode of refinement produces good results in some cases; some examples from this research's re-implementation of it are shown in figure 2.18.

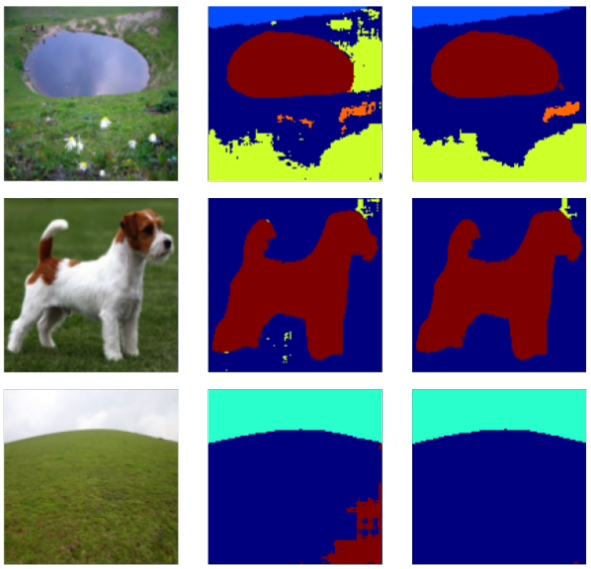


Figure 2.18: FCN Refinement from Andrearczyk and Whelan [2017] – (a) original inputs, (b) FCN Predictions, (c) Refined Predictions

2.2.3 Loss Functions

As mentioned in the discussion of equation 2.17, the choice of loss function is integral to neural network training. Particularly in this research since a loss function that is capable of learning without consistent semantic categories is required; that is, in a *label-agnostic* or *permutation-invariant* fashion. Furthermore, the loss function should not be influenced by the assignment of a particular human category to a specific numeric value. For example, if in a particular ground truth image "cat" is 1, and "dog" is 0, the network selecting the opposite labelling scheme in its prediction should not influence the loss function as long as the network's choice of numeric assignment is still consistent. There exists a myriad of loss functions in the literature, and this research directly considered seven of them before resorting to the proposed permutation-invariant design. Here, a background and description of each of the loss functions considered is provided, but a deeper analysis into their performance - and into why they were insufficient - is saved for section 3.7. These descriptions are provided to emphasise that the choice to design an alternative loss function was not made lightly.

For the multi-class problem of image segmentation, a common approach is to have each pixel in the prediction of the network represented as a vector of softmax probabilities, which is called $\mathbf{X}$ here, and each pixel in the ground truth represented as a one-hot encoding, which is called $\mathbf{Y}$. This notation is the default throughout this section.

2.2.3.1 Binary Cross Entropy and Soft Dice Loss

Cross-Entropy (CE) and Soft Dice Loss [Crum et al. 2006] are the most common loss functions for training FCNs in the literature [Sudre et al. 2017; Jadon 2020]. Many researchers even use them in conjunction, hence the inclusion of both under this heading. Given the variables described in the introductory section, the cross-entropy loss of each pixel prediction with the ground truth is given by:

$$\mathrm{CE}(\mathbf{X}, \mathbf{Y}) = -\sum_{C} y_i \log\left(x_i\right) \tag{2.22}$$

Once calculated, it is reduced to a scalar by aggregating all of the pixel losses. CE falls in the range $[0, \infty)$, with zero being the output when the prediction corresponds perfectly with the ground truth, and so the intention is to minimise CE. Framing the problem of image segmentation as getting each pixel-level label's correct one-hot encoding allows us to consider each pixel as a collection of binary classifications; this justifies the use of *binary* cross-entropy (BCE) even in multi-class image segmentation.

Soft Dice Loss (DL) [Sokolova et al. 2006; Zijdenbos et al. 1994] was developed as a more lenient, differentiable, and generalised alternative to the popular Mean Intersection-Over-Union (MIoU) metric [Rezatofighi et al. 2019]. MIoU is the most widely used metric in image segmentation, it is calculated as follows:

$$\mathrm{IoU} = \frac{\text{ground truth} \cap \text{prediction}}{\text{ground truth} \cup \text{prediction}} \tag{2.23}$$

and so measures the extent to which the predicted segmentation mask and ground truth overlap, as opposed to a naive count of corresponding pixels. DL is described by the following equation:

$$\mathrm{DL}(\mathbf{X}, \mathbf{Y}) = \frac{2\sum x_i y_i}{\sum x_i + y_i} \tag{2.24}$$

Do note that, in practice, $1 - \mathrm{DL}$ is generally used so that 0 denotes perfect performance and the function can be minimised during training. Due to its relation to F1-score, DL is also more suited to optimisation than CE, since DL trades-off precision and recall [Sokolova et al. 2006; Jadon 2020]. Although CE and DL are often used independently, it is common to average them together, yielding $\frac{\mathrm{CE}+\mathrm{DL}}{2}$. This combination often produces faster and more stable convergence than either loss function can achieve alone [Jadon 2020]. Both DL and CE are fully semantic loss functions; they require the place and value of the predictions and ground truth to correspond entirely for perfect performance. This research requires other loss functions to be considered because loss functions that are permutation-invariant or label-agnostic are required to learn from automatically annotated data.

2.2.3.2 Kullback–Leibler Divergence and Donsker-Varadhan Loss

Kullback-Leibler Divergence (KLD) [Kullback and Leibler 1951] is closely related to cross-entropy as a lower-bound representation of conditional entropy that is less computationally expensive to calculate [Liu 2020].

KLD is generally saved for when CE is being applied in a continuous setting and, like CE, is not technically a metric because it is not symmetric [Kullback and Leibler 1951; Liu 2020]. As mentioned earlier, the multi-class image segmentation problem is typically represented as multiple, pixel-level, one-hot classification problems which allows the use of BCE to calculate the loss. Since KLD is specifically designed for application to continuous probability distributions (when working in a discrete case, CE is used), this research attempts another approach in which the network outputs a single layer that is interpreted as a probability heat map. The equation for KLD is:

$$\mathrm{KL}(\mathbf{Q}||\mathbf{P}) = -\int_{-\infty}^{\infty} q(x) \log\left(\frac{q(x)}{p(x)}\right) \mathrm{d}x \tag{2.25}$$

Which, in the binary (discrete) case, is equivalent to equation 2.22; since the fraction in the log term falls away to avoid division by zero and is replaced with just the denominator, which in the binary case is not probabilistically different. Here, $\mathbf{P}$ and $\mathbf{Q}$ represent the two-dimensional max of each one-hot encoding in $\mathbf{X}$ and the argmax of each one-hot encoding in $\mathbf{Y}$, respectively. When structured in this way, the network is trained to replicate a probability distribution with its output layer instead of predicting label values. That probability map can then be thresholded to acquire the desired segmentation map. This is not an approach encountered elsewhere in the literature; as stated above, the standard technique is to train with CE. As will be shown in section 3.7, this can produce a decent outcome in some use cases, but it is not ideal. KLD is known for highly variable, unbounded outputs, hence the numerous alternative representations that have been created for it [Liu 2020; Belghazi et al. 2018].

To address KLDs exploding values, this work also considered Donsker-Varadhan Loss (DVL) [Donsker and Varadhan 1975], which is provably a lower-bound alternative representation of KLD. DVL improved convergence and prevented the exploding values but was still insufficient, as will be returned to later. Nonetheless, this alternative to BCE is more closely suited to the research requirements. It does not prioritise the precise location of the label values as much as BCE since it also compares their distribution in the heat map.

2.2.3.3 Mutual Information Neural Estimation

Mutual Information Neural Estimation (MINE) is a loss function proposed by Belghazi et al. [2018]. It is based on a metric known as mutual information (MI) score and intends to measure the dependence between two random variables; or, more relevantly – the similarity between two segmentations or clusterings of the same data, it does this in a label-agnostic fashion, any permutation of label assignments is considered equivalent under MI score. Suppose $\mathbf{X}$ and $\mathbf{Y}$ are two proposed segmentations of the same image, with o objects in each segmentation and n pixels in the image. Here $|\mathbf{X}_i|$ and $|\mathbf{Y}_i|$ are used to denote the number of pixels in object i in segmentations $\mathbf{X}$ and $\mathbf{Y}$, respectively. Then the probability that a pixel drawn at random from a segmentation belongs to the ith object can be calculated using $\mathbf{P}(i) = \frac{|\mathbf{X}_i|}{n}$ and $\mathbf{P}'(i) = \frac{|\mathbf{Y}_i|}{n}$. One can also compute the probability that a random pixel is in object i in the prediction and object j in the target by evaluating $\mathbf{P}(i,j) = \frac{|\mathbf{X}_i \cap \mathbf{Y}_j|}{n}$. Using the above computations, the mutual information of $\mathbf{X}$ and $\mathbf{Y}$ can be found using:

$$\mathrm{MI}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{o}\sum_{j=1}^{o} \mathbf{P}(i,j) \log \frac{\mathbf{P}(i,j)}{\mathbf{P}(i)\mathbf{P}'(j)} \tag{2.26}$$

Note that equation 2.26 is a discrete version of KLD from equation 2.25 [Moore 2001]. However, MI cannot be used as a loss function for neural networks directly because it is non-differentiable and inefficient to calculate over each object. To circumvent this, some alternative representations of MI that are more efficient to calculate have been devised [Liu 2020], and some have been used as regularisers [Peng et al. 2020]. These approximations usually work quite effectively in domains with lower dimensionality but are not as appropriate for images or complex multi-class clustering problems. MINE loss is an improved approximation designed for those more complex tasks. MINE aims to train a neural network with parameters θ such that:

$$|\text{MI}_\theta(\mathbf{X}, \mathbf{Y}) - \text{MI}(\mathbf{X}, \mathbf{Y})| < \epsilon \tag{2.27}$$

where ϵ is an arbitrarily small value. MINE loss in equation 2.27 is approximating MI from equation 2.26, and thus it is not sensitive to any permutations of class labels. Thus, one can refer to MINE loss as *label-agnostic*. It measures the relative size and placement of all the clusters without considering the numeric class labels themselves. Consider the case of an object labelled 2 in the target mask and 3 in the prediction mask; typical loss functions and metrics would deem this solution incorrect because they are tailored towards label values having semantic meaning. MINE loss could consider this labelling correct, provided that the alternative predicted labelling scheme still has overlapping classes of the same size. A greater $\text{MI}(\mathbf{X}, \mathbf{Y})$ suggests greater dependence between $\mathbf{X}$ and $\mathbf{Y}$ and so neural networks should aim to maximise MINE loss. The results produced by each of these loss functions will be discussed in section 3.7, but the limiting factor of MINE is that it is a learned approximation of the loss function actually required by this research - that learning can be impeded if the underlying distribution is changing drastically.

2.2.3.4 Cauchy-Schwarz Divergence

Cauchy-Schwarz Divergence (CSD) [Hoang et al. 2015] measures the distance between two probability density functions (PDFs). It is frequently used in signal processing, regression, and information theory; however, since it is defined for PDFs and not random variables, its application to classification, clustering, and segmentation problems is less frequent. The equation for CSD when comparing two PDFs, $\mathbf{P}$ and $\mathbf{Q}$, is:

$$\text{CS}(\mathbf{P}, \mathbf{Q}) = -\log \frac{\int \mathbf{P}(x)\mathbf{Q}(x)\,dx}{\sqrt{\int \mathbf{P}^2(x)\,dx \int \mathbf{Q}^2(x)\,dx}} \tag{2.28}$$

In the general multi-class segmentation problem described earlier in this section, one-hot encoded random variables $\mathbf{X}$ and $\mathbf{Y}$ are considered. To apply CSD to these variables, they must first be converted to PDFs using Gaussian Kernel Density Estimation. Another consideration is that CSD has the range $[0, \infty)$, with 0 being the output if the PDFs match exactly. The potential for infinite values cannot be easily accounted for if maintaining differentiability is desired, but values high enough to cause infinite value errors are quite unlikely in practice.

Although it resembles MINE loss from the previous section, CSD is not label-agnostic. CSD relies on a comparison of the general number (density) of each label value in the prediction vector than it does on their placement. This should produce similar results given a large enough image but offers fewer theoretical guarantees. This distinction is quite subtle so consider the following example in which $\mathbf{Y}$ is the ground truth and $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ are predictions:

$$\mathbf{Y} = [0, 1, 0] \qquad \mathbf{X}_1 = [0, 1, 0] \qquad \mathbf{X}_2 = [1, 0, 1] \qquad \mathbf{X}_3 = [1, 1, 0]$$

$\mathbf{X}_1$ is the correct prediction, and all the loss functions will consider it such. The label-agnostic loss function required would also consider $\mathbf{X}_2$ correct since it is just a permutation of the same labelling scheme. Both MI and CSD would consider $\mathbf{X}_2$ correct. However, CSD also considers $\mathbf{X}_3$ correct because the density of the relative label values in the prediction and ground truth is the same. MI does not consider $\mathbf{X}_3$ correct because it also requires the permutation of label values to be positionally consistent.

2.2.3.5 Set Cross Entropy

Similarly to CSD and MINE Loss, Set Cross-Entropy (SCE) [Asai 2018] produces results in line with what is required from a label-agnostic loss function. It is designed for a different application - creating permutation-invariant encoder layers. Instead of comparing the network output and ground truth directly, as is usual, SCE compares them as sets. The equivalence of sets is defined differently, and so this produces a permutation-invariant likelihood. The actual computation of this is taxing, and so the author proves that the Hausdorff distance, with CE as the distance metric, will find the upper bound of the actual permutation-invariant CE of two probability distributions. They prove it for the binary case and negative-log likelihood but state that it should hold more generally.

This loss function, while close to what this research requires, is still an approximation of the actually maximally-overlapping permutation-invariant loss. The applicability of the loss functions introduced in this section will be discussed more deeply in section 3.7, where an alternative loss function is designed to solve the research problem proposed in this dissertation.

2.3 Learning Paradigms

2.3.1 Supervised

Supervised learning constitutes the most widely-explored paradigm in machine learning. Supervised learning considers a set of tuples containing inputs, $\mathbf{X}$, and labels, $\mathbf{Y}$, and attempt to learn a function f that maps from $\mathbf{X}$ to $\mathbf{Y}$:

$$f(x) \rightarrow y \in \mathbf{Y}, \forall x \in \mathbf{X}$$

Supervised learning frequently plays a role in the final step of the OBIA workflow, that of classifying the objects. Most implementations of the OBIA framework include a manually performed selection stage after segmentation occurs in which the user selects example objects representative of certain classes and selects salient features to distinguish them from other objects. These example objects are used in a nearest neighbour approach to classify the other objects found in the image. This is the most representative example of OBIA, as discussed in Blaschke [2010], Veljanovski et al. [2011], Platt and Rapoza [2008], and Xie et al. [2008].

Liu et al. [2018b] also explored other supervised approaches within the classification step of OBIA. Namely, deep convolutional neural networks (DCNNs) and FCNs. As stated in Hossain and Chen [2019], the OBIA research community eschews the use of convolutional neural networks due to their data-intensive nature. For most forms of semantic segmentation tasks, a plethora of labelled data is not a luxury researchers are afforded. However, recent developments in neural networks have made allowances for data scarcity, and to test this Liu et al. [2018b] tested their DCNN and FCN on different amounts of labelled data. They found that, even in low-resource settings, the FCN could outperform the traditional classification techniques of random forest and support vector machines. The next paradigm discussed here aims to circumvent labelled data entirely. It is also more relevant to the second step of the OBIA workflow - object proposal and segmentation - which constitutes the primary focus of this research.

2.3.2 Unsupervised

Unsupervised learning is considered the antithesis of supervised learning. Given a set $\mathbf{X}$, but no labels, the goal is to learn a function f that maps samples from $\mathbf{X}$ to a set of classes, $\mathbf{C}$. The intention is that the classes in $\mathbf{C}$ maximise either inter-class separability or intra-class compactness; this distinction refers to the focus of the metric being used to rate the classes. The primary advantage of unsupervised learning is the ability to bypass the time-consuming labelling and annotation process, which is particularly helpful in pixel-level segmentation tasks.

Unsupervised learning has not garnered much attention in the classification step of the OBIA workflow. This is because OBIA typically prioritises the accuracy and real-world usability of methods over the speed at which they may be applied. OBIA case studies typically involve large amounts of expert-tuning and data labelling, as with Gusella et al. [2005] and Liu et al. [2018b]; the researchers themselves often take on the difficulty of data annotation. Noting this, Lucieer and Lamarche [2011] performed wetland mapping in New Zealand using a variant of the k-means algorithm in the final step of the OBIA workflow.

However, as can be seen in section 2.1.2.2, the algorithmic segmentation techniques applied in the segmentation step of the OBIA workflow are generally unsupervised. This is possible because when segmenting the image into proposed objects first - and only then choosing objects that most closely correspond to the geographic features being detected - the requirements placed upon the segmentation step are reduced. Any inaccuracies produced by the segmentation algorithm can be merged or accounted for in the clustering and classification step.

Unsupervised approaches also often require alternative accuracy measures since there is unlikely to be overlap in the arbitrary numerical value assigned to each predicted cluster and those chosen for the ground truth. The two most commonly applied unsupervised evaluation metrics are certainly Adjusted Mutual Information score (AMI) [Vinh et al. 2009] and Adjusted Rand Index (ARI) [Hubert and Arabie 1985]. Romano et al. [2016] states that when considering predictions for datasets with larger or more balanced clusters, ARI should be prioritised.

2.3.3 Self-supervised

Self-supervised learning bears a technical resemblance to supervised learning in that function approximation is still performed in the same way; however, it is a subset of unsupervised learning [Zhai et al. 2019]. These paradigms differ in terms of how the data is created. In a supervised setting, the training set of $\mathbf{X}, \mathbf{Y}$ pairs is annotated or labelled by experts. This is a highly time-consuming process, particularly within the context of image segmentation.

In self-supervised learning, the aim is to bypass the process of data annotation. This is typically done by augmenting existing datasets with new labels that can be assigned automatically according to transformations of that data, but, occasionally, the loss functions of a model may also be altered [Doersch and Zisserman 2017; Zhai et al. 2019; Nousi and Tefas 2018]. These transformations must affect the data in predictable ways. The goal is that by learning from the augmented datasets (pre-text tasks), the network will learn knowledge that could be transferred to a downstream task.

The term pre-text task refers to a dataset whose labels were automatically generated through transformations or augmentations as opposed to being annotated by a human [Zhai et al. 2019]. These augmentations can occur before training begins, during training before the data is passed to the model, or while the loss is being calculated. One of the pressing limitations in machine learning is that the models are *data-starved*, pre-text tasks attempt to address that limitation. The models should be able to learn some generalisable knowledge from the pre-text tasks that can be applied to real-world problems.

2.4 Summary

This chapter intended to describe the background of this work sufficiently enough to understand the proposed aims. With this context established, this research now explores whether self-supervised texture segmentation with FCNs provides for significant enough generalisation for the downstream task of object proposal. Here, object proposal refers to the segmentation step of the OBIA workflow discussed in section 2.1.2.2, not semantic segmentation. This will require the generation of an automatically labelled pre-text texture segmentation task in section 3.8.3. Since neural networks were designed in the supervised learning paradigm, either the task itself or the learning process must be modified.

Andrearczyk and Whelan [2017] modified the task by performing a "pre-segmentation" step that made the automatically labelled data correspond to semantic classes. This requires the grouping of multiple textures into general categories such as "bark", "bubbles", or "checkerboard"; so that the network can still learn to assign fixed labels to certain textures. This severely limits the number of textures that can be used when training the network and requires a pre-segmentation step; this research aims to avoid both of these limitations and so must resort to modifying the learning process itself.

This encouraged the consideration and discussion of numerous loss functions in section 2.2.3. The aim here was to locate a loss function in the literature that allows for predictions that contain permutations of the ground truth labelling scheme. Section 2.2.3 also defines the terms "permutation-invariant" and "label-agnostic",

which describe such a loss function. That investigation will proceed in section 3.7, where an alternative self-augmented loss function is proposed.

The alteration made to the loss function also necessitates two other adjustments to the model: one to the FCN's architecture in section 3.6 and another to its feature space in section 3.5. An understanding of those additions would be aided by the descriptions of FCNs, dilated convolution, pooling, and skip connections that were provided in section 2.2. More specifically, for a neural network to adjust to a permutation-invariant loss function, it requires a much larger receptive field than would otherwise be necessary. Additionally, in a permutation-invariant case, the network cannot derive a label value from the object itself - but must instead consider the surrounding context and what labels will be assigned elsewhere - and thus requires much greater spatial awareness. This necessitated that the FCNs forego translational equivariance as defined in section 2.2. The data augmentation that removes translational equivariance and the self-augmented loss function alluded to above firmly embed the work in the self-supervised learning paradigm as defined in section 2.3.3.

By embedding a self-supervised FCN into the OBIA workflow, many of the problems that would typically arise when performing self-supervised image segmentation can be circumvented, namely the lack of specificity and semantic information provided by synthetic data. Such an approach could reduce OBIA's reliance on proprietary software while improving on the performance of the algorithmic segmentation techniques that predominate in the field. This approach will be described in more detail in chapter 3 and results are provided in section 4.5.

Chapter 3

Research Methodology

3.1 Introduction

In this chapter, the proposed method that was alluded to in chapter 2 is expanded; and a more detailed description of it can be found in the next section. Then the hypothesis that must be validated to suggest that the proposed pipeline was successful will be expressed. Then each of the four modifications that were required to ensure the eventual success of the proposed pipeline will be described; they are: Optional Translational Equivariance in section 3.5, Translated Skip Connections from section 3.6, Hungarian Self-augmented Cross-Entropy in section 3.7 and, finally, the Prime Texture Analysis Dataset (PTAD), the introduction of which is saved for section 3.8.3.

3.2 Proposed Workflow

This research modifies the existing OBIA workflow outlined in section 2.1.2, particularly in step 2, the object proposal or image segmentation step. To this end, a more detailed enumeration of the steps conducted in the proposed framework can be seen below and an illustration is shown in figure 3.1.

1. Begin with sample images
2. Use an FCN to segment the image, instead of algorithmic segmentation techniques

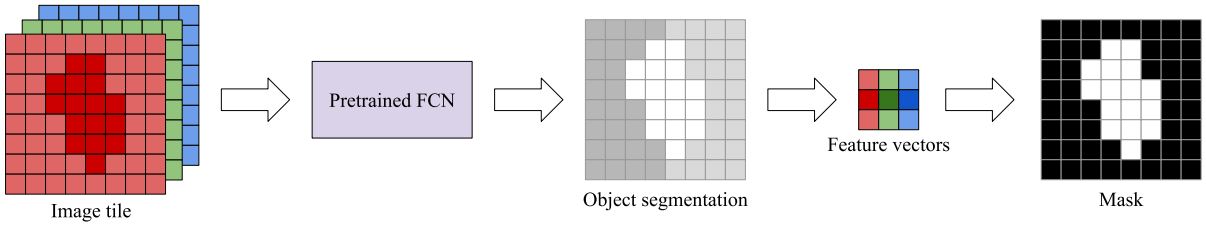


Figure 3.1: An illustrated example of the proposed workflow

- This FCN will be trained exclusively on an automatically annotated dataset for texture segmentation in a permutation-invariant fashion
- The FCN's output can be tuned using FCN refinement and modifications to the training data

3. Construct meaningful feature vectors for the objects proposed by the FCN
4. Cluster the proposed objects in order to extract the semantic features of the image

The four aforementioned alterations all centre around the implementation and achievement of step 2, and those will be the primary focus of this chapter. First, a research hypothesis that can accurately judge the success of the workflow proposed above is proposed.

3.3 Research Hypothesis

The validation of the following hypothesis constitutes the primary aim of the proposed research:

> An FCN trained using a synthetic texture analysis dataset can effectively replace traditional segmentation algorithms in the object proposal step of the object-based image analysis workflow with comparable or superior performance while maintaining a degree of generalisability

While seemingly quite specific, the above hypothesis requires many modifications to conventional FCN architectures and image segmentation datasets to be validated. More specifically:

- As introduced in section 2.2.1, FCNs are typically translationally equivariant, and so they cannot determine where the kernel is spatially in the image. The implicit effect of this is that the FCN cannot assign different labels based on the amount of the image it has already seen, which makes clustering substantially more difficult. This problem is addressed in section 3.5.
- In certain tasks, the surrounding area of a pixel or object contributes non-negligibly to the label that needs to be assigned to an object; this is particularly true in clustering or region proposal. This research found that the standard approaches to expanding the receptive fields of FCNs mentioned in section 2.2.1.2 were insufficient, and so a more specific alternative is proposed in section 3.6.
- Learning from the automatically annotated synthetic dataset created here requires a permutation-invariant loss function. In section 3.7 this loss function is introduced, and the section describes why those mentioned in section 2.2.3 were not applicable.
- The algorithmic segmentation techniques that are compared to the proposed framework do not require training data, let alone human-annotated training data; this justifies the focus on synthetic pre-text datasets. After surveying some existing texture analysis and image segmentation datasets in section 3.8.2, this research resorts to generating a more complex and related pre-text task in section 3.8.3.

Besides validating each of these modifications to FCNs independently, this hypothesis also required several experiments comparing the algorithmic segmentation techniques from section 2.1.2.2 to the proposed approach. Those experimental results establish the effectiveness of the proposed workflow, but they are saved for the next chapter.

3.4 Assumptions, Limitations, and Scope

This section is devoted to expounding upon the assumptions and limitations that frame the scope of this research. Considering the research as a whole, three primary limitations were identified and they can be stated as follows:

- While OTE, TSC, and HCE could hypothetically be expanded to different forms of neural network architectures, this research only consider FCNs being applied to normalised image segmentation. The performance of these approaches may differ substantially when applied in other contexts.
- While self-supervised texture segmentation can generalise to object proposal within the context of images that contain human-interprable textures, this approach may not function in a more general case. A larger texture dataset or another approach to generating synthetic images may be required in different circumstances.
- In section 4.8 the benefits that learnability can provide in the object proposal case considered in this research are shown. An important consideration is that this learnability requires the overhead of creating the pre-text synthetic dataset and the training on it. The benefit of learnability may not be sufficient justification for these two extra processes. Creating synthetic datasets is challenging, time-consuming, and a computational burden that in some cases will outweigh the benefit of learnability and reduced tuning time. In section 3.8.3.3 the dataset proposed in this research and the code used to generate it is provided and there are many alternative synthetic datasets in the literature, so the burden of creating a dataset will not be a factor in every case. With that said, training the required architecture on the chosen synthetic dataset cannot be avoided and so this trade-off between learnability and training time must be considered when choosing learnable approaches over their algorithmic alternatives.

Further, some assumptions must be made made prior to even conducting experiments, given the potential number of methods expressed in contemporary research. These assumptions made conducting experiments more feasible and prevented the scope from expanding too greatly for the time afforded to this research. The three major scope-limiting assumptions made by this work are:

- It is assumed based on an analysis of prior literature in chapter 2 that OBIA is a sensible approach to image segmentation in low-resource domains, this research does not compare OBIA to other approaches.
- Further, based on the same literature review, it is assumed that FCNs are the most reasonable starting point for introducing learnability into the OBIA workflow. This assumption is justified by FCNs already being considered in the wider-literature and by them being more adaptable to data scarcity than some other approaches discussed in chapter 2.

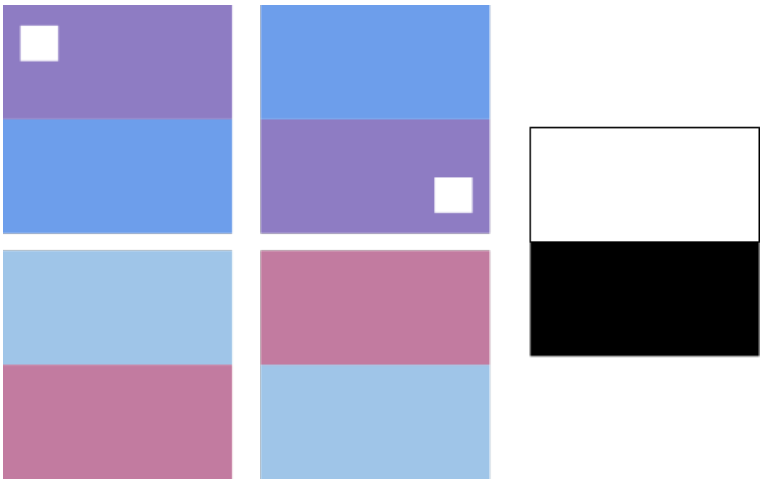


Figure 3.2: Diagram depicts four samples from the top rectangle only example dataset on the left, and the ground truth for every sample on the right; the white squares represent the network's ERF

- Finally, experimentally, it is assumed that increasing the size of the architectures considered in this work may reduce the benefit of dilated convolution and TSC due to the implicit expansion of the receptive fields achieved in larger networks. However, it would be empirically challenging to determine whether a performance improvement results from an expansion of the receptive field or the presence of extra parameters, and therefore all the architectures considered in this research contain a similar number of parameters. When working with substantially larger networks, the benefits of using TSC and OTE need to be further investigated.

With the research scope now outlined more concretely, this chapter will proceed by introducing the four major additions to the literature made by this research. The focus of these introductions to the proposed methods is on how they were implemented and the problem they address, experimental validation will occur in chapter 4.

3.5 Optional Translational Equivariance

Here a technique that allows FCNs to forego translational equivariance is proposed. Although this technique was developed independently, a similar idea was explored by Liu et al. [2018a]; they describe it as a convolutional layer while this research considers it a pre-processing step. This divergence in purpose is justifiable since they explore solving datasets that require more than just the forgoing of the translational equivariance demonstrated by CNNs. Translational equivariance is typically considered advantageous and was described in section 2.2.1. In image classification tasks, the location of an object in an image seldom has a significant impact on its class label. Similarly, when performing semantic segmentation with FCNs, the object's position often is not relevant given that the FCN's primary endeavour is to determine whether the pixels contained in a particular kernel are comparable to the texture and shape of a relevant class. The benefit of translational equivariance is no longer as obvious when considering segmenting or clustering textural information in a label-agnostic fashion.



(a) (b) (c)

Figure 3.3: Visualisation of the meshgrid produced by OTE

To demonstrate this more intuitively, some samples from a simple, synthetic dataset of rectangles are shown in figure 3.2. Over the course of this chapter, this dataset will be expanded two further times until it effectively represents a simple clustering problem; this should explain the purpose of each of the intended contributions in this dissertation. Two different colour rectangles are generated and stacked vertically, each of them spanning half the image. The FCN is always trained to label the top rectangle 1 and the bottom rectangle 0. Although simple, this task demonstrates a deficiency in contemporary FCNs when considering the problem of label-agnostic clustering; given an FCN of depth 3 with 3×3 kernels and a 128×128 image, this task is impossible for the FCN to solve optimally. The figure contains two small white squares representing the ERF of a network, both covering the same colour. Due to translational equivariance, the network can never determine whether the colour it is observing is at the top or bottom of the image. Those two kernels will produce an identical response in the feature vector of the network. Continually increasing the depth of the network in proportion to the size of the image is a suitable - albeit inefficient and non-scalable - approach to solving this problem. In that way, one may increase the ERF of the network enough for it to detect the line bisecting the image. Optional Translational Equivariance (OTE) is a simpler and more efficient solution that concatenates the (x, y) position of each pixel to its feature vector.

OTE allows the network to decide how much to weigh the position of an object in its classification. Rather than making translational equivariance inevitable, this modification makes it optional. However, this alone is not sufficient, larger images have larger coordinate values for some pixels, and so each coordinate must also be normalised between 0 and 1 by dividing by the image height, H, and width, W. The transformation for each pixel's feature vector is:

$$(R, G, B) \rightarrow (R, G, B, \frac{x}{W}, \frac{y}{H})$$

yielding the five input channels of the network. The two extra channels add location information in a textural representation. A visualisation of the information contained in these two extra channels is shown in figure 3.3.

Once implemented, an FCN of depth 3 with 3×3 kernels is capable of solving the task shown in figure 3.2 optimally for any image size; thus making the FCN more comparable to the algorithmic segmentation techniques currently used. OTE consistently improves performance since the network can ignore the x, y coordinates if they are irrelevant. To emphasise this, a brief ablation study is performed in section 4.2.2. One disadvantage of OTE is that the increased number of features requires a corresponding increase in the number of parameters when using an FCN. Adding to this, when working with datasets in which the position is more explanatory in the training set than in certain testing samples, OTE can result in overfitting. This limitation is reflected on in section 4.2.3.

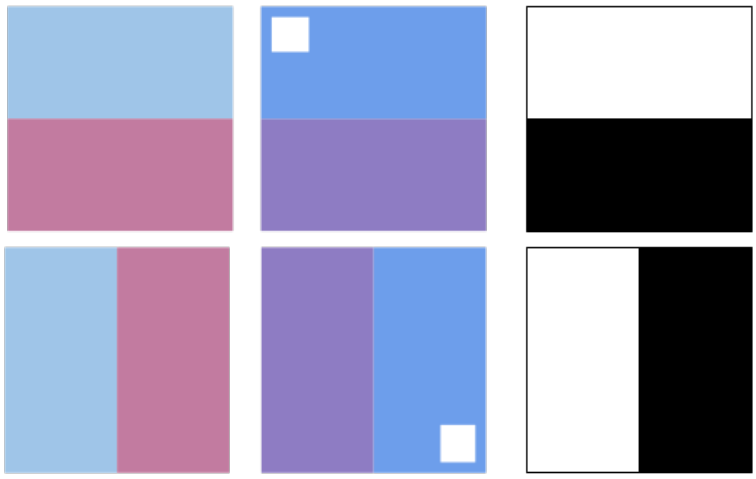


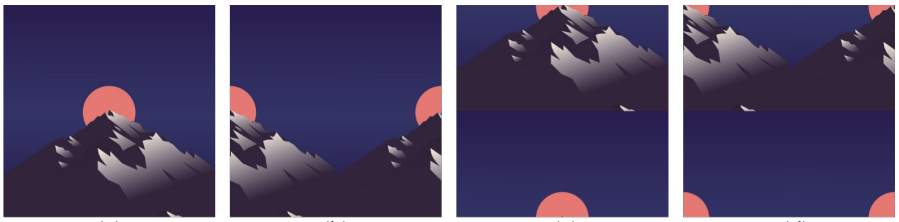
Figure 3.4: Diagram depicts the ERF of the network as white squares, two vertical and two horizontal samples on the left, and the two reference images for the dataset on the right

3.6 Translated Skip Connections

To demonstrate the second challenge posed to us, the simple task introduced in the previous section is expanded upon. As shown in figure 3.4, it now includes both horizontal and vertical rectangles and the network is instructed to always label either the left or top of the image depending on whether the bisection is horizontal or vertical. Due to OTE, the network can now differentiate top and left pixels from the bottom and right pixels in the image, but it still cannot solve this task optimally for all image sizes. When the network's ERF is sufficiently small compared to the image size, the receptive field of some kernels cannot determine whether the line bisecting the image is horizontal or vertical (because it cannot detect the line at all) and thus cannot determine whether to label the top or left rectangle. Once again, two illustrative kernels are shown in figure 3.4 as white squares. Notice that whether the network uses the colour, shape, or texture of an object to determine its label, it will always fail at this simple task if the ERFs of some kernels are not sufficiently large to detect the bisecting line.

In order to resolve this, one must either increase the depth of the network proportionally to the image size, add a fully connected layer somewhere in the network (reducing the ability of the network to generalise between image sizes), or - preferably and more efficiently - increase the ERF of the network. Quite a detailed discussion of receptive fields is provided in section 2.2.1.2, and the primary method of increasing the receptive field size of FCNs - dilated convolution - is also introduced in section 2.2.1.3. In those sections, some of the limitations of traditional techniques for increasing the ERFs of FCNs are also examined, but the primary among them is that the expansion provided is insufficient for the task at hand. Therefore, this research proposes an alternative approach called the Translated Skip Connection (TSC), which is a novel extension of the standard skip connection discussed in section 2.2.1.4, that increases the receptive field much more significantly than dilated convolution and the other techniques mentioned in section 2.2.1.2.

Before discussing TSCs more specifically, the translation function $\mathbf{T} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ must also be defined. It shifts an input matrix in a given direction by a given factor. For example, given a feature map $\mathbf{X}$, $\mathbf{T}_{\uparrow}(\mathbf{X}, \frac{1}{4})$



(a) (b) (c) (d)

Figure 3.5: Examples of translation function, left to right: (a) $\mathbf{X}$, (b) $\mathbf{T}_{\leftarrow}(\mathbf{X}, \frac{1}{2})$, (c) $\mathbf{T}_{\uparrow}(\mathbf{X}, \frac{1}{2})$, and (d) $\mathbf{T}_{\nwarrow}(\mathbf{X}, \frac{1}{2})$

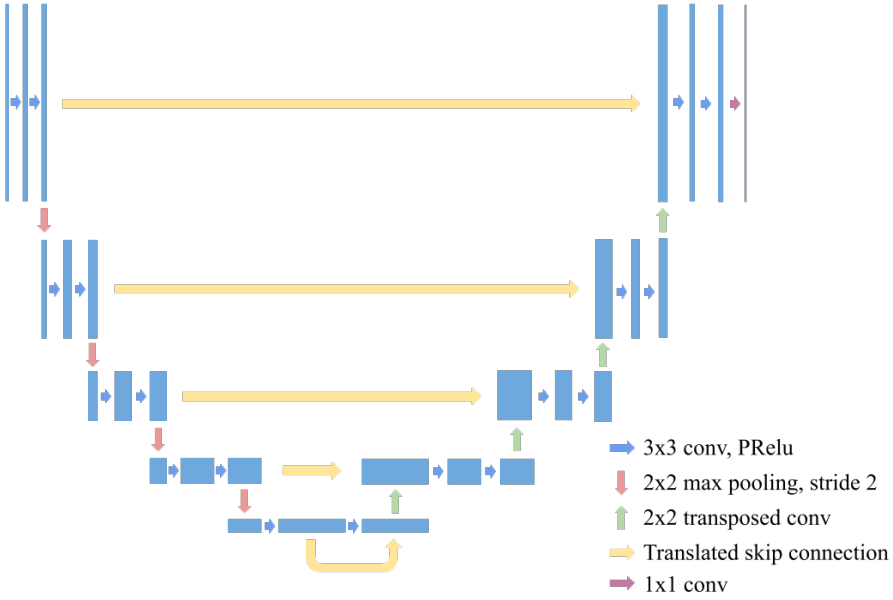


Figure 3.6: Diagram of the proposed TSC architecture

would translate the feature map upwards one fourth the height of the feature map, with the top of the feature map reentering at the bottom. Figure 3.5 contains more examples of the translation function being applied to an image. A variable, l, is also defined, that represents the layer in the decoder that the skip connection is skipping to, and the hyperparameter $\mathbf{D}$ that defines the depth of the network. Although a TSC is a single neural network module, it is intended to be embedded in a wider architecture with multiple TSCs, hence the need for these variables.

The TSC architecture used in these experiments is depicted in figure 3.6. The architecture combines ideas from two earlier networks, V-Net and U-Net, which are depicted in section 2.2.2.1; for brevity, this dissertation will refer to this architecture as TSC-Net. The network includes five skips between the encoder and decoder portions, shown in yellow. Each yellow skip includes a single additive skip connection and three translated concatenative skip connections that translate the encoder output left, up, and diagonally (left and up), respectively. The translation factor in each TSC is $\frac{l}{\mathbf{D}+1}$, where l is the index of the layer in the network, from 1 to 5 starting at the bottleneck, and $\mathbf{D}$ is the network's depth, 5. Given the notation defined above, each of the five skip connections in the network can be described with the following equation, where $\mathbf{X}_s$ is the skipped

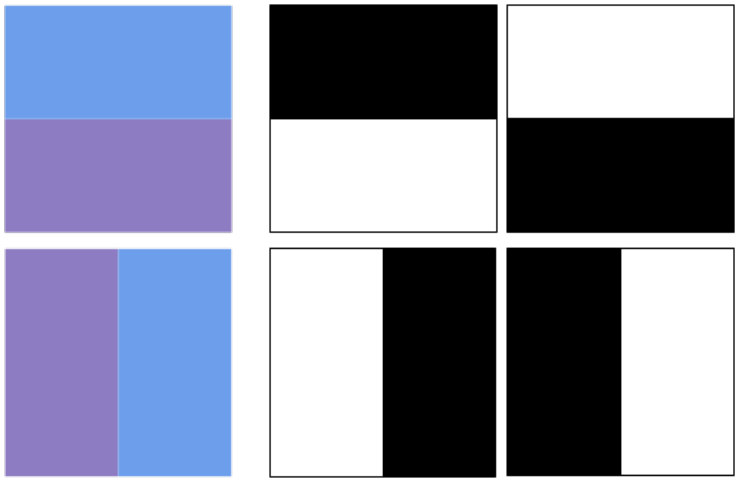


Figure 3.7: Diagram depicts one vertical and one horizontal sample on the left, and the four reference images for the dataset on the right

input to the layer and $\mathbf{f}(\mathbf{X})$ is the output of the previous layer:

$$(\mathbf{f}(\mathbf{X}) + \mathbf{X}_s) \oplus \mathbf{T}_{\leftarrow}(\mathbf{X}_s, \frac{l}{6}) \oplus \mathbf{T}_{\uparrow}(\mathbf{X}_s, \frac{l}{6}) \oplus \mathbf{T}_{\nwarrow}(\mathbf{X}_s, \frac{l}{6}) \tag{3.1}$$

This equation comprises a single TSC, so each TSC is made of a single additive skip connection, $\mathbf{f}(\mathbf{X}) + \mathbf{X}_s$, and three concatenative skip connections, denoted $\oplus$.

As mentioned in section 2.2.1.4, the choice between additive and concatenative skip connections is typically entirely arbitrary. However, the decision to construct a single TSC with one untranslated additive skip connection and three translated concatenative skip connections is justifiable.

The untranslated additive skip connection maintains the value of the untranslated skipped input and, relative to a concatenative skip connection, reduces the number of parameters required by each TSC; this essentially fulfils the purpose of a traditional skip connection. Additionally, if three additive skip connections had been used instead of the concatenative ones, each kernel would have a uniform weight for the skipped input and the three translated inputs. With concatenative skip connections, each translated input has its own weights and so can be appropriately down-weighted when greater spatial specificity is required.

3.7 Permutation-invariant Loss Function

After resolving the limitations regarding the ERF of FCNs above, the complete clustering problem this research aims to solve can now be considered. The example dataset is extended for the last time and the result is shown in figure 3.7. Every sample in the dataset now has two reference images, one of which is selected by the dataloader at random. The reference images now represent the information that needs to be transfer ed - namely that there

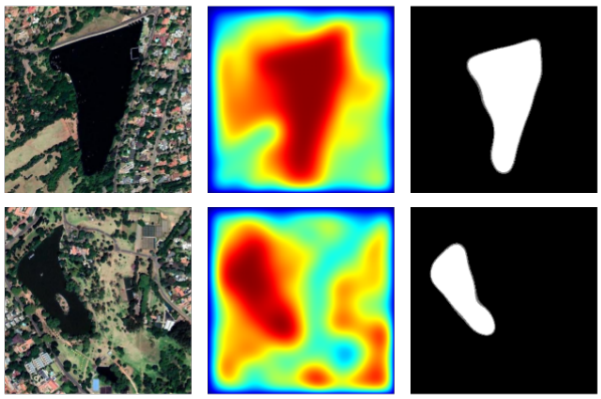


Figure 3.8: Depictions of (a) original images of Emmarentia and Zoo Lake, (b) probability distributions produced by FCN, and (c) the thresholded distributions

are two colours in the image split horizontally or vertically - as opposed to just a positional labelling of "top" or "left". This problem is meaningfully different from those before; instead of the network learning in a fixed fashion that left and top are 1, and bottom and right are 0; it must now simply learn that they are distinct notions and choose a labelling scheme that adequately represents that understanding.

One might assume that the task has become substantially easier since the dataset now contains more correct answers but the same complexity. However, the task now also contains contradictory labelling from the perspective of a semantic loss function. For example, if the network trains on a portion of the dataset and learns to label the left of the image 1, that will have to be unlearnt when it encounters reference images that, given the same input, require the right to be labelled 1. Over the course of an entire epoch, the network will make no meaningful progress towards solving the problem. In order to solve this task, the loss function used during training requires the property of permutation-invariance or label-agnosticism; the loss function must be indifferent about which label is assigned to top, left, bottom, or right; as long as the information that there are two distinct regions in each image is transferred correctly. An example of this form of permutation invariance is shown in section 2.2.3.4. This section is devoted to the development of said loss function, but first, the reasons why the other loss functions from the literature, introduced in section 2.2.3, were not applicable for this use-case must be explored.

Binary Cross-Entropy and Soft Dice Loss are fully semantic loss functions, as was already mentioned in section 2.2.3.1. BCE is still incorporated into the augmented loss function proposed later in this section; however, BCE alone is not suitable for this task without significant modification.

Kullback–Leibler Divergence and Donsker-Varadhan Loss can account for the similarity of two probability distributions. The segmentation masks considered here can be represented as probability distributions through Kernel Density Estimation (KDE). As was alluded to in section 2.2.3.2, KLD, when used directly in place of BCE for image segmentation, often encounters exploding values. DVL, as a lower-bound alternative representation of KLD, provides greater stability and smaller values. Using KDE and DVL, the network is trained to produce probability distributions instead of one-hot encoded predictions, which can then be thresholded into segmentation masks. Two examples of this approach are shown in figure 3.8, these were obtained by training U-Net on the synthetic dataset described later in section 3.8.3.1. While intriguing and somewhat effective, the

distributions produced are typically centred around specific parts of the image, and it is quite challenging to decide upon an adequate threshold to apply to each of the distributions the network produces. Ultimately, the lack of both theoretical soundness and interpretability drove us towards other approaches.

Mutual Information Neural Estimation Loss is a learnt loss function trained specifically to approximate mutual information calculations for particular distributions. MINE functions are effective for training Generative Adversarial Networks to generate images from particular classes because, in the adversarial setting, it can learn the underlying distribution of the class while the generator and discriminator train [Belghazi et al. 2018]. This research also attempted the simultaneous training of an FCN and MINE on a single set of training images, but FCNs are not as robust to this inconsistent feedback, so the results were not promising. Additionally, for image segmentation, the distribution being estimated is quite complex and changes with every image, so MINE cannot learn the underlying distribution any more effectively than the FCN itself.

Cauchy-Schwarz Divergence is an approximation of the permutation-invariant loss function required by this research. As can be seen in section 2.2.3.4, CSD compares the density of each label value but does not consider their index in the vector. This should produce reasonable results given a large enough image, but more theoretical guarantees are preferred. Additionally, CSD cannot be directly applied to the softmax probabilities produced by the network, further complicating the training process.

Set Cross-Entropy was designed by Asai [2018] to allow for the unsupervised training of the encoder portion of autoencoder networks. This approximation is more efficient and quite sensible for their use case. However, as was the case with both MINE and CSD, training the entire network with a loss function that potentially provides some inexact responses would result in less stability than desired.

3.7.1 Self-augmented Hungarian Cross-Entropy

After considering the loss functions discussed above, a self-augmented loss function was designed, with $O(n^3)$ complexity, that is guaranteed to find the loss value for the network with a maximally-overlapping permutation of the ground truth. The label values in the ground truth - in this case, the correct segmentation map - are permuted to overlap with the network's prediction maximally, and then the loss is calculated. This approach is also fully-differentiable and operates with the softmax probabilities provided by the network. It also works with any loss function, so DL, CE, and the combination of the two can all be used.

Self-augmented loss functions have the following structure [Aljalbout et al. 2018]:

$$\mathbf{L}(\mathbf{P}, \mathbf{T}) = f(\mathbf{P}, \pi(\mathbf{T})) \tag{3.2}$$

where f represents any loss function, $\mathbf{P}$ represents the prediction, and $\mathbf{T}$ represents the ground truth. π is a permutation function that permutes the ground truth in some meaningful way. In this case, π permutes $\mathbf{T}$ to maximally-overlap with the prediction. The initial, naive approach to solve this was to test every permutation of $\mathbf{T}$ to find the one that maximises the accuracy between $\mathbf{P}$ and $\mathbf{T}$. This naive implementation has complexity $\Omega(n!)$ and is unusable in most cases. The problem can be reframed as a linear sum assignment to improve on this.

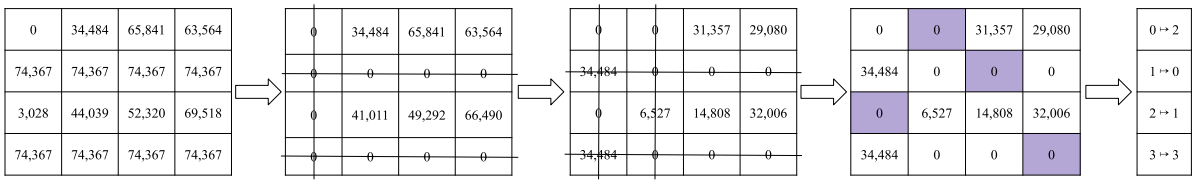


Figure 3.9: Example of matrix being processed with the Hungarian algorithm. In the original matrix row 0 becomes row 2, row 1 becomes row 0, row 2 becomes row 1, and row 3 stays row 3. That reassignment will minimise the trace of the matrix (the diagonal sum).

First the contingency matrix between the original prediction $\mathbf{P}$ and ground truth $\mathbf{T}$ is acquired. In this contingency matrix, each row represents a label in $\mathbf{T}$, and each column a label in $\mathbf{P}$. Thus, if the reassignment of columns that maximises the diagonal of the contingency matrix is found, that corresponds to the maximally overlapping permutation of $\mathbf{T}$. The optimal algorithm to find the diagonal with the *minimum* sum is known as the Hungarian algorithm, developed by Kuhn [1955], and it can be described in 6 steps:

1. Iterate through each row, subtract the minimum of each row from every element in that row; producing at least one zero in each row
2. Iterate through each column, subtract the minimum of each column from every element in that column; producing at least one zero in each column
3. Cover all the zeros with the minimum number of horizontal or vertical lines. If the number of lines equals the number of rows, skip to step 6; otherwise, continue to step 4.
4. Find the minimum value of the elements not covered with lines in step 3 and subtract it from every uncovered element
5. Add that minimum to every element that is at the intersection of two lines and then return to step 3
6. Find an assignment that selects only one zero from each row and column, that is the minimal assignment

To solve this in the maximal case, one simply alters the matrix element-wise in the following way:

$$\mathbf{T}' = |\mathbf{T} - \max(\mathbf{T})|$$

and then applies the Hungarian algorithm to $\mathbf{T}'$. The solution will maximise the diagonal sum of the unaltered matrix, $\mathbf{T}$, which corresponds to the maximally overlapping permutation, π. This improves the runtime of the algorithm from $O(n!)$ to $O(n^3)$, while maintaining the important theoretical guarantee that:

$$\text{Accuracy}(\mathbf{P}, \mathbf{T}) <= \text{Accuracy}(\mathbf{P}, \pi(\mathbf{T}))$$

An example of the Hungarian algorithm being applied to a matrix is provided in figure 3.9. This algorithm is applied before calculating the loss at every training step, so some further considerations arise. Firstly, since neural network training is batched, it is possible to perform the label reassignment once for every batch or once for every sample in the batch. Secondly, one may either calculate the loss from the argmax label values used during reassignment or the original softmax probabilities. In both instances, this research opts for the latter.

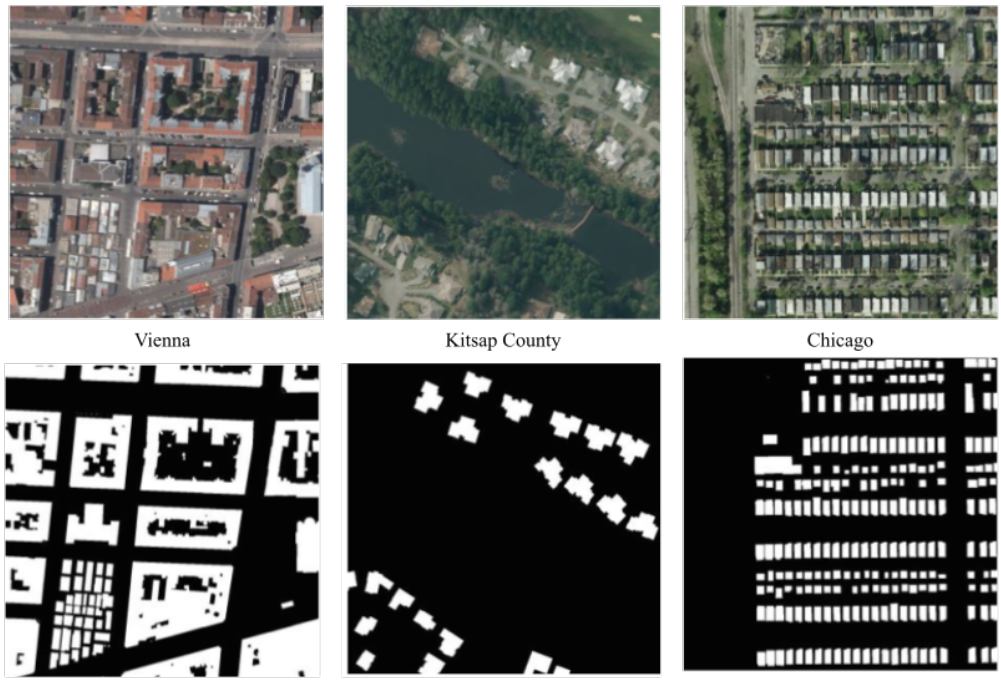


Figure 3.10: Samples from the Inria Image Labelling Dataset; top row: the ground truth tiles; bottom row: their respective reference images

This augmentation based on the Hungarian algorithm is used in conjunction with binary cross-entropy from equation 2.22, and so it is referred to as Hungarian Cross-Entropy (HCE). Besides not requiring us to perform a pre-segmentation step [Andrearczyk and Whelan 2017] or forcing us to ignore annotations entirely [Kiechle et al. 2018], this loss function and its guarantees allow us to take some liberties that were not afforded to prior researchers when constructing synthetic texture segmentation datasets.

3.8 Datasets

The methods proposed in this chapter are validated across six real-world and synthetic datasets. This section begins by briefly introducing the pre-existing datasets considered in this work, including one which is antecedent to the synthetic dataset proposed later: the Prague Texture Segmentation Dataset [Haindl and Mikes 2008]. This section then describes how the proposed synthetic data generators and texture segmentation pre-text tasks were created.



Figure 3.11: Samples from Landcover.ai; top row: original images; bottom row: their respected reference images

3.8.1 Real-world Data

3.8.1.1 Inria Image Labelling Dataset

The Inria Image Labelling Dataset[1] (IILD) is a collection of satellite images of suburban and urban scenes from around the world released by Maggiori et al. [2017]. The dataset comprises images from 9 different cities, including San Francisco, Chicago, and Vienna, and ten flights. The images span a total area of 810km^2 with a pixel resolution of 30cm and are accompanied by reference images corresponding to a pixel-level labelling of buildings. A small sample of the dataset can be seen in figure 3.10.

As can be seen in the sample, IILD contains a massive amount of variance. The cities have different architectural styles and different urban densities. Capturing the images over multiple flights also subtly affects the data through slight changes in lighting and altitude. Maggiori et al. [2017] split the data into training and testing sets in an interesting way; the training set will not contain images from the same cities as the testing set. Thus, a model trained on IILD is required to generalise information learnt about buildings to cities it has never seen before.

3.8.1.2 Landcover.ai

Landcover.ai is another publicly available RGB-only satellite image dataset released in 2020 by Boguszewski et al. [2020]. Landcover.ai was captured over Poland and Central Europe; it contains 41 orthophoto tiles covering a landmass of approximately 216.27 km^2. The pixels in each orthophoto were manually categorised as containing background, building, woodland, and water. The orthophotos are then split into $40,000$ yet smaller 256×256 tiles to expand the amount of training data; three of those tiles are depicted in figure 3.11.

[1] https://project.inria.fr/aerialimagelabeling/

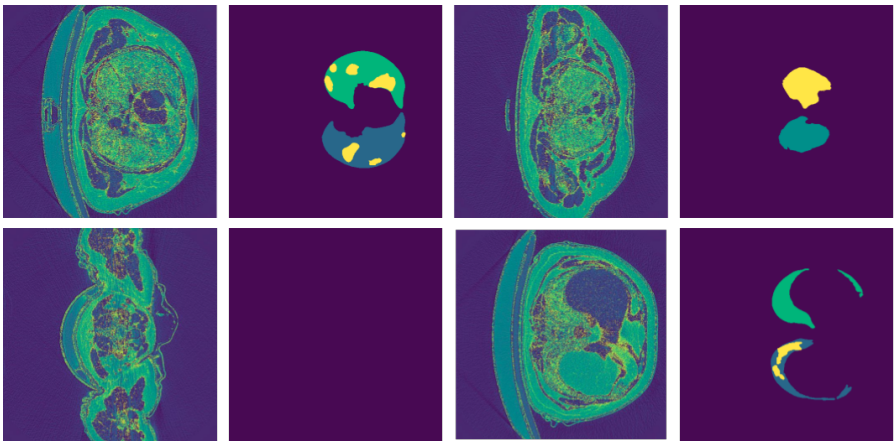


Figure 3.12: Samples from the COVID-19 CT Scans Dataset, four channels extracted from MRI scans are depicted with their corresponding reference images

Compared to IILD, Landcover.ai has greater class-complexity and class-imbalance, with only 4% of pixels being classified as buildings, and therefore poses a different challenge. Since satellite images have quite spatially dispersed objects, larger receptive fields will benefit models applied here. Unlike IILD, Landcover.ai's validation and test sets are more representative of the training data, and this is also coupled with the immense size of the dataset; over-fitting is not a substantial concern.

3.8.1.3 COVID-19 CT Scans Dataset

The COVID-19 CT scans dataset [Jun et al. 2020; Glick 2020] is a medical imaging dataset that comprises CT scans taken from 20 different patients who were diagnosed by medical professionals with COVID-19. The pixels are grouped into four categories: background, left lung, right lung, and infection. Each CT scan represents a three-dimensional object containing a varying number of channels, which are split into $2,000$ different tiles. This transformation from three to two dimensions is standard practice when working with medical images and converts the problem into a conventional image processing one [Milletari et al. 2016].

This dataset is substantially smaller than both IILD and Landcover.ai and contains some class-imbalance, so it is used to evaluate how effectively a model adjusts to limited data availability. This dataset also effectively tests the receptive field of a given architecture because a small receptive field would be unable to discern whether it is detecting the left or right lung. Some representative samples that showcase both the class imbalance and the left and right lung detection are provided in figure 3.12. There is great variety in both the body types of the patients and their orientation - which affects the relative location and shape of the lungs in each channel - so generalisation is essential to avoid a model over-fitting to scans from particular patients.

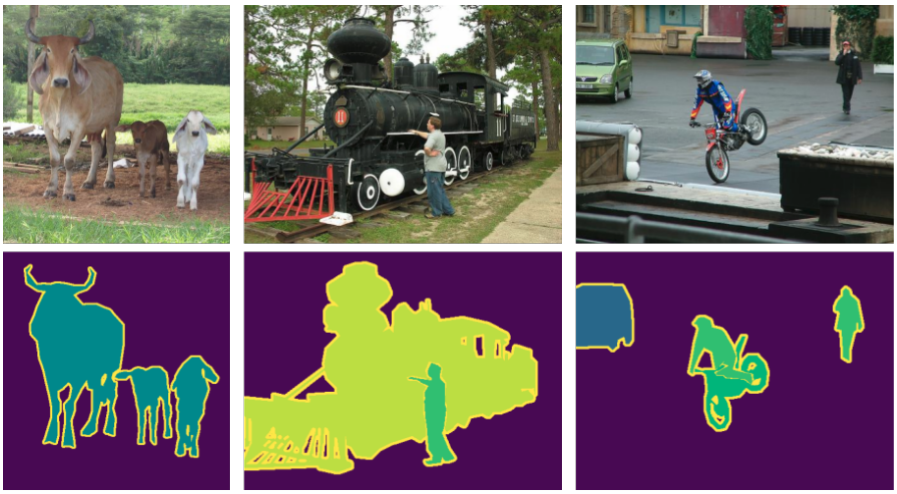


Figure 3.13: Samples from PASCAL VOC2012; top row: original images; bottom row: their respective ground truths

3.8.1.4 PASCAL Visual Object Classes 2012

PASCAL Visual Object Classes 2012 (VOC2012) [Everingham et al. 2010] has 22 classes, including aeroplanes, dogs, cats, and people; two of the classes are reserved for border pixels and an "other" category. VOC2012 contains complex, real-world images and has a substantially smaller training set than the other datasets introduced in this section. With approximately 1450 images, overfitting and not adjusting to the label complexity are common problems that networks must overcome to perform effectively on VOC2012. Samples from VOC2012 are provided in figure 3.13.

This dataset is too complex to solve optimally with the limited amount of training data available, and so it is generally used in conjunction with larger synthetic datasets or in a weakly-supervised setting. Additionally, it is common for architectures much larger than the ones considered here to be applied to this dataset; since this research is performing a comparative analysis, state-of-the-art performance on VOC2012 is not intended. However, as shown later, even the models applied here can learn a more general, textural approach to the semantic segmentation required and attain suitable results.

3.8.1.5 Describable Textures Dataset

The Describable Textures Dataset [Cimpoi et al. 2014] is the largest collection of grouped textures found in the literature. It contains 5,652 textures spanning 57 semantic categories. Some sample textures are provided in figure 3.14. The categories for these textures include bubble, braided, webbed, bumpy, and blotched, among others. As can be seen in the figure, many of the textures contain gaps or multiple colours; models applied to this dataset have to adjust to that noise. The recognition of these textures alone could likely provide for fairly rich image analysis.



Figure 3.14: Textures drawn from the Describable Textures Dataset

As will be discussed later in this chapter, in order to create a synthetic dataset sufficiently complicated to encapsulate the problem of texture segmentation, the Describable Textures Dataset can be merged with a Simplex data generator; this process is described in section 3.8.3.3. If a standard FCN and loss function were used, the network trained to segment these images would require a 57 channel output layer to select a category for each texture. The application of HCE from section 3.7 can surmount this limitation.

3.8.2 Synthetic Data

Instead of creating pre-text tasks by transforming existing data or using real-world data, it is possible to construct entirely synthetic datasets. This is not common practice in image segmentation due to the complexity of the task and the difficulty of generalisation, but some recent examples do exist. For example, Ros et al. [2016] has created a widely used synthetic dataset for the semantic segmentation of urban scenes. More recently, and more closely aligned to this research, is the creation of procedurally generated remote sensing urban scenes for the vision of autonomous vehicles by Dosovitskiy et al. [2017]. Both works use game engines to generate realistic-looking datasets that adequately summarise real-world tasks. Alternatively, one can create quite realistic imagery using pseudo-random noise and texture generators; this approach is favoured because it allows one to circumvent the creation of entire game environments.

3.8.2.1 CARLA Semantinc Segmentation for Self-driving Cars

The Semantic Segmentation for Self Driving Cars dataset [Manickavelu 2018] was generated using the aforementioned Carla self-driving car simulator [Dosovitskiy et al. 2017] and has been publicly available on Kaggle since 2018. This dataset contains $5,000$ synthetic images captured in Carla. The pixels in each image are segmented into 13 different label categories, including pedestrian, vegetation, road, and car. Some examples from this dataset are provided in figure 3.15.



Figure 3.15: Examples from the CARLA Synthetic Dataset; top row: synthetic images; bottom row: respective synthetic reference images

Similarly to the PASCAL VOC2012 Dataset, the number of samples is small relative to the label complexity. Furthermore, given that the images are synthetic, the textures are relatively simple and have limited blending compared to real-world images. This means that the receptive field of an architecture does not have as significant an effect as it does in real-world data since the model can singularly focus on the texture of each object. This will be expanded upon empirically in the next chapter.

3.8.2.2 The Prague Texture Segmentation Datagenerator and Benchmark

The Prague Texture Segmentation Datagenerator was created to dynamically generate a synthetic Dataset that allows neural networks to be trained to perform general texture analysis. The Prague dataset - samples of which can be seen in figure 3.16 - is frequently used to benchmark FCN-based texture analysers. The generator also provides options that may rotate, scale, blur, and illuminate the data variably, making the segmentation task more complex and realistic.

The *large* version of the dataset contains 80 images with 97 textures spanning nine textural types, including bark, stone, and glass, taken from the creator's own collection of textures and the DynTex Database [Péteri et al. 2010]. Splitting the textures into these more general types allows the dataset to be used to train standard semantic FCNs. This works since the models simply need to classify the type of texture into a general semantic category for which it has multiple examples and so only requires nine labels. However, this does mean that the models trained on this dataset are not traditional texture analysers since they are limited to the texture types already contained in the dataset.

The creation of synthetic data is a significant challenge. Despite the variety of textures, noise generators, and blurring algorithms used to create the Prague Dataset, one can see that it is still fairly spatially simplistic compared to real-world imagery. The models trained on this dataset are typically only applied to solving this dataset, as can be seen with Andrearczyk and Whelan [2017] and Huang et al. [2019], and it is seen as a synthetic benchmark for texture analysing architectures and algorithms. This dataset would have to be expanded



Figure 3.16: Samples from the Prague Texture Segmentation Benchmark; top row: synthetic mosaics of textures; bottom row: respective ground truth labellings of textures

on to make these models more generalisable but, without alternative loss functions, limitations regarding the grouping of textures into categories have to be kept in mind. This can be transcended by applying HCE from section 3.7, and so another synthetic dataset, inspired by Prague, is proposed in section 3.8.3.3; it has many more textural categories and greater spatial complexity.

3.8.3 Synthetic Data Generation

After considering the existing synthetic datasets listed above, this research also resorts to constructing alternative synthetic datasets. Numerous iterations were required to arrive at a sufficiently complex dataset worthy of release. Some of the prior iterations were used for illustrative experiments in this manuscript, and others were used directly in the final release. The final version is presented here in section 3.8.3.3 along with two relevant prior implementations.

3.8.3.1 Ellipsoidal Dataset

Synthetic data generation is a complex task; thus, some assurances were required before committing fully to this course. To that end, the first simple synthetic dataset that represents a microcosm of the broader problem this research intends to solve was created. A pre-text task that would allow an FCN architecture to transfer to the task of lake segmentation; that is, learning to cluster one texture instead of many. To create this task, a random number of ellipses with random diameters are generated and then scattered across a canvas. A simple water texture is scaled, rotated, and cropped before being overlaid on each ellipse. The background of the image is also replaced with a texture representing either a desert, urban, or forested area drawn from a collection of

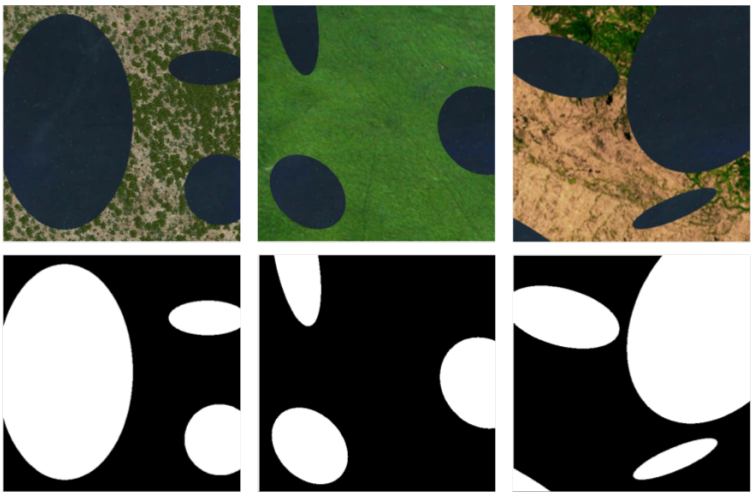


Figure 3.17: Ellipsoidal synthetic dataset for lake and water segmentation, three synthetic images are depicted in the top row, with their respective ground truths in the bottom row

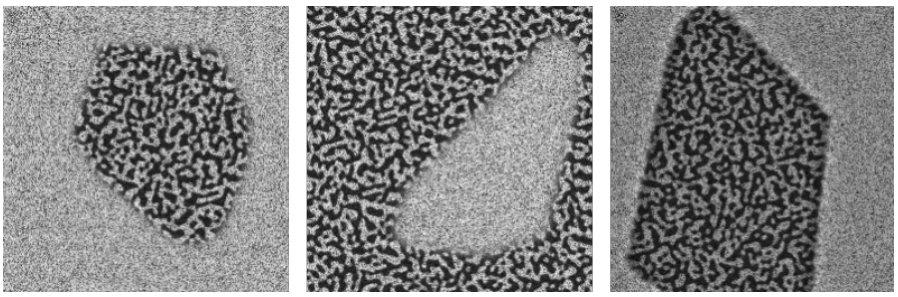


Figure 3.18: Silhouettes Generated using ConvexHull and OpenSimplex

scenes captured manually on Google Earth [Google 2021]. Finally, a degree of salt-and-pepper noise is added to each generated image to improve realism. While generating the canvas in this way, a binary canvas must also be maintained; this binary canvas contains ones wherever an ellipse is present on the main canvas and zeroes elsewhere. An example of some samples from this synthetic dataset can be seen in figure 3.17. This dataset is quite simplistic, although advantaged by the variety of textures; it is improved substantially in the two subsequent sections, but this served as a suitable proof of concept.

$10,000$ input/target pairs were generated in this way and split into $6,000$ training and $2,000 \times 2$ validation and testing samples. Then popular FCN architectures, including U-Net discussed in section 2.2.2, were trained on this synthetic dataset with different loss functions. One of the results obtained from this training is shown in figure 3.8 for the test of the DVL loss function. Of course, this dataset is fully-semantic and thus inadequate for the final aims of this research, but the results were promising enough to justify further exploration.

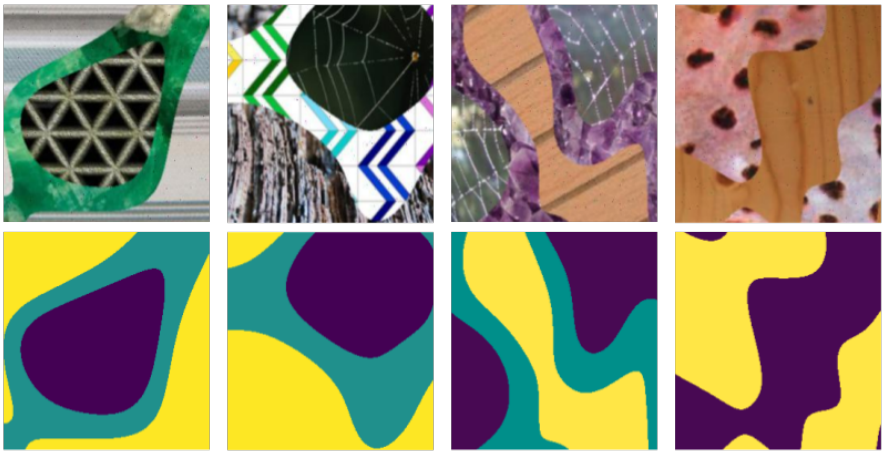


Figure 3.19: Samples drawn from the Prime Texture Analysis Datagenerator, synthetically generated mosaics of textures in the top row and their reference images in the bottom row

3.8.3.2 Simplex Silhouettes

To increase the variety of shapes present in the synthetic images beyond basic geometries such as ellipses and to avoid the manual collation of relevant textures, a modular silhouette generator with Simplex noise generation [Kurt Spencer et al. 2019] was implemented. The simplex generator will provide a level of blending and consistency to the textures present in the generated images, and the silhouette generator provides complex shapes. The two pseudo-random generators are merged into a composite algorithm to create a more intricate outcome. The results of this texture generator are shown in figure 3.18. The steps undertaken to generate this dataset are as follows:

1. Generate a random number of (x, y) points on a canvas
2. Perform the ConvexHull algorithm [Barber et al. 1996] around the points to find a blob
3. Apply Simplex to generate a texture inside the blob, permute the frequency and scale of the distribution, and then apply Simplex again outside the blob

The result represents a random silhouette that could be described as a basic background-foreground segmentation task. It can generate numerous examples in parallel and provide a decent amount of variety. The rudimentary textural complexity is a limitation, but this could be accounted for by overlaying textures from an extensive texture database. Indeed, by combining this algorithm with the textural overlay from the previous section the final proposed texture analysis dataset was created.

3.8.3.3 The Prime Texture Analysis Dataset

The Prime Texture Analysis Dataset (PTAD) is the final synthetic dataset created for this research. It is inspired by the Prague Texture Segmentation Dataset from section 3.8.2.2 and can be accessed online. The generation process is an amalgamation of ideas drawn from Prague, the Ellipsoidal Dataset, and the Simplex Silhouettes.

OpenSimplex is used to generate a canvas with between 2 and k silhouettes, where k specifies the maximum number of classes present in the dataset. Since these silhouettes are drawn from simplex distributions with different frequencies, they can easily be thresholded into clusters with highly intricate but consistent shapes. After this is done, like the Prague and Ellipse datasets, a randomly cropped, scaled, and rotated texture is embedded into each generated cluster.

In the Ellipse dataset, the number of classes was limited by the semantic nature of the problem; the generator had to consistently label the water textures in each ellipse as 1s and the backgrounds - which could not contain water textures - as 0s. This required a time-consuming texture grouping stage akin to manual labelling. Similarly, Prague is limited by the grouping of textures into broad semantic categories. This limits both the number of textures - because every new texture has to be manually classified - and the number of categories - because FCNs become more challenging to train the more classes they are required to learn. However, due to the development of HCE in section 3.7, both of these limitations can be bypassed since the labelling scheme is arbitrary.

The permutation-invariant loss function allows for a training regime in which the network only requires a one-hot encoding equal to the maximum number of classes in each image - that is, the aforementioned parameter k. The network then learns a label-agnostic clustering rather than essentially performing a classification of each texture as with Andrearczyk and Whelan [2017]'s pre-segmentation; this allows us to increase the number of textures present in PTAD dramatically. PTAD was generated using both the textures acquired for the Ellipse Dataset and the Textures from DTD in section 3.8.1.5; this results in a database of over $5,600$ textures, which allows for much greater generalisability than Prague's 97 textures. Since models trained with HCE do not require semantic categories, it is also simple to expand the database further.

Both the parallelisable Python data generator[2] and the version of the dataset[3] used to acquire the results presented in section 4.8 are available online. This version of the dataset contains $80,000$ training, $20,000$ validation, and $5,000$ test images and took approximately 4 hours across 16 cores (64 CPU hours) to generate. Due to the large number of textures and variety of shapes, even this amount of data only encapsulates a small portion of the overall complexity. This allows for the training of more robust and generalisable texture analysis models. A subset drawn from this dataset can be seen in figure 3.19.

[2] https://github.com/JoshuaDBruton/WTSD
[3] https://webwtad.firebaseapp.com/

3.9 Summary

This chapter introduced and described the research methodology and aims. Additionally, several datasets used in experiments in the following chapter were introduced. More specifically, first, an outline of the proposed workflow was provided in section 3.2, that of a self-supervised FCN embedded in the OBIA workflow. Following that, the related research hypothesis was described. As anticipated, some limitations regarding FCNs had to be overcome before the proposed workflow could succeed. The proposed solutions comprise three of the four contributions: OTE (section 3.5), TSC (section 3.6), and HCE (section 3.7). The chapter concluded with a description of PTAD, a synthetic texture segmentation dataset.

Over the course of chapters 2 and 3, sufficient rationale for why and a description of how the proposed solutions were implemented has been provided. The next chapter will establish empirically whether each of the proposed solutions works individually, and then proceed to validate whether the proposed workflow functions satisfactorily as a whole. To that end, ten quantitative and three qualitative experiments are conducted in the next chapter. These experiments span multiple datasets and aim to alleviate doubts regarding the functioning of the proposed workflow and the techniques that allow for it.

Chapter 4

Results

4.1 Introduction

In this chapter, experiments that evaluate the proposed methods and pipeline introduced in the previous chapter are carried out. The experiments are grouped by the method they relate to and in the order they were introduced. Proceeding in this way emphasises that each of the proposed methods is also applicable beyond the proposed workflow. For comparison, the six datasets presented in section 3.8 are used in conjunction with seven different FCN architectures. Before proceeding with the experiments, those architectures and the general experimental setup are introduced.

4.1.1 Comparison Architectures

The FCN architectures considered here are designed to meet the requirements of each of the experiments conducted in this chapter. The architectures are designed with a focus on the size of their effective receptive fields (ERFs) and the mode by which they increase it. More specifically, the architectures represent the primary approaches to increasing the receptive fields of FCNs presented in section 2.2.1.2.

The baseline model considered is the popular U-Net architecture [Ronneberger et al. 2015], it is shown in figure 2.14. U-Net uses 2×2 strided max-pooling to scale down the input image and thus increase the effective field; this is the only unlearned approach to increasing the effective field included in these results. The U-Net architecture is extended twice, using dilated convolution with dilation rates of two and three, since dilation rates higher than three generally result in poor performance. Dilation expands the receptive field beyond that of the original U-Net in a learned fashion. This this architecture is also shrunk while maintaining the same input and output size but reducing the number of hidden layers by two so that the network has a depth of three. This U-Net derivative will be referred to as Small U-Net, or SU-Net.

The fifth architecture examined here is a scaled-down version of V-Net from Milletari et al. [2016], referred to as B-Net, a more contemporary version of U-Net. The original architecture can be seen in figure 2.15, although here 3×3 and not 5×5 convolution is used. The primary differences are replacing U-Net's max-pooling with strided convolution and U-Net's ReLU with PReLU. The 2×2 strided convolution with stride two in B-Net increases the receptive field in a similar way to U-Net's pooling; only the convolutional approach is also learnable. Due to the introduction of this strided convolution, even B-Net, the scaled-down version

of V-Net, still has slightly more parameters than U-Net, approximately 10 million and 8 million parameters, respectively. These extra parameters often result in higher performance but can also lead to overfitting.

The six architectures discussed above are compared to the TSC architecture, TSC-Net, depicted in figure 3.6. TSC-Net would also have more parameters than U-Net and B-Net, because of the two extra concatenative skip connections in each decoder layer. To more fairly assess the performance of the architectures and to assess if it is the placement of these parameters and not their quantity that improves the performance of the architecture, the number of filters in each layer of TSC-Net are occasionally reduced so that it totals approximately 9.6 million parameters, fewer than B-Net. This shrinking is performed whenever the network is being compared to B-Net and U-Net, namely in sections 4.2.2 and 4.3. A version of TSC-Net with just three layers is also applied in section 4.3, following the same standard as SU-Net, this shrunked architecture is referred to as Small TSC-Net (STSC-Net).

4.1.2 Experimental Setup

The specifics of the design of each experiment will be discussed later, but here the general experimental methodology is described. The networks use the same optimiser during their training, AdaDelta [Zeiler 2012]. This optimiser allows us to forego the selection of a learning rate, automating some hyperparameter selection while maintaining a fair standard. The batch size for essentially every architecture on every dataset was 4. The exceptions are the two particularly large datasets; Landcover.ai from section 3.8.1.2 and IILD from section 3.8.1.1, on which a batch size of 8 is used to speed up training.

All the implementations are GPU compatible and used Pytorch with Pytorch Lightning. The implementations are available on GitHub under a GNU General Public License[1]. All the experiments were run on the Wits MS Cluster.

4.2 Optional Translational Equivariance

4.2.1 Image Size

4.2.1.1 Experiments and Results

First one must consider whether OTE achieves its primary purpose of allowing FCNs to detect spatial information texturally, thus removing the limitations of translational equivariance. Further, it must detect this textural information generalisably. The same FCN architecture are trained with and without OTE on the same dataset to test this. Ideally, the network should learn to extract the relevant information provided by OTE,

[1] https://github.com/JoshuaDBruton/TSC

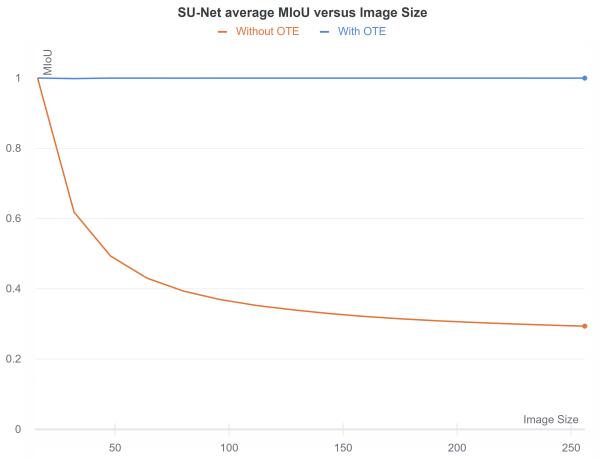


Figure 4.1: Chart showing how OTE affects SU-Net's test MIoU as image size increases

and the information it learns should apply for any image size since the pixel coordinates provided by OTE are normalised.

The network considered is SU-Net, the downsized version of U-Net. It has three layers in its encoder portion instead of U-Net's four and uses 3×3 convolution. At the bottleneck, the ERF of SU-Net is approximately 7×7; however, the border pixels of each kernel contain information beyond the kernel, so, in the decoder layer, the network can extract some extra information. The dataset shown in figure 3.2 is used, in which the network is simply trying to label the top half of the image one and the bottom half zero. Training starts on a 16×16 image, for which the task is solvable without OTE. Next, the same test is run on a similar dataset with different colours but the images are continually scaled up from 16×16 to 256×256 to observe how OTE affects the performance of the models.

The results of this experiment are shown in figure 4.1. The Y-axis represents the MIoU achieved by the model on test data, with 1 representing perfect overlap with the ground truth. The X-axis is the size of the input image, with 16×16 being the smallest, increasing in steps of 16 to 256×256. The blue line is SU-Net with OTE, and the orange line is SU-Net without OTE. Both models start with perfect accuracy on the 16×16 images on which they were trained, but the model without OTE quickly trails off as the image size increases.

A more qualitative visualisation is also provided in figure 4.2. This figure shows the artefacts - or lack thereof - that each model starts producing as the image size increases. Here, the input image is shown on the left and five predictions provided by the OTE and without-OTE models for each image size are visible on the right. These results will be discussed further in the next section, but the without-OTE model's performance decreases as it resorts to guessing more regarding each kernels location in the image.

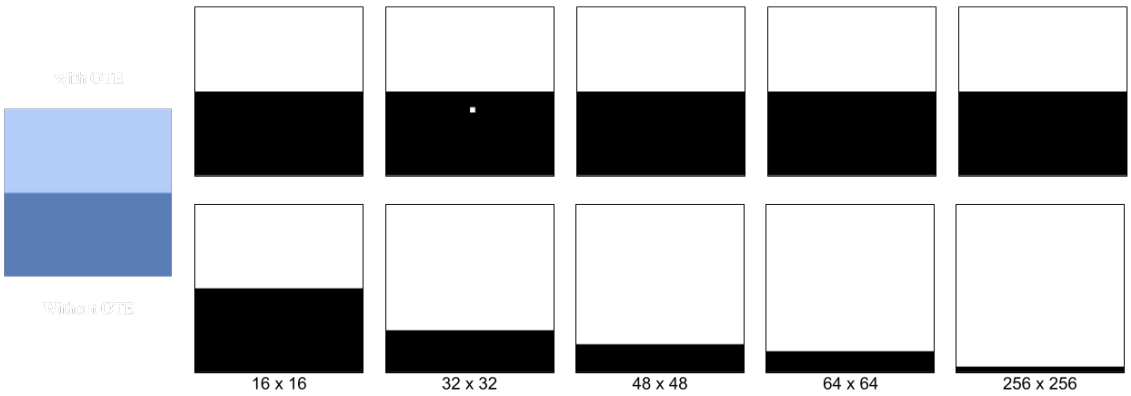


Figure 4.2: Diagram showing SU-Net's predictions with and without OTE for the input on the left at different scales

4.2.1.2 Discussion

These results show that OTE makes some spatial reasoning independent of the receptive field size of the network. The implementation from section 3.5 also functions for any image size due to the normalisation of the x and y coordinates. Since this task depends on the ERF of each network, and the network architecture is designed specifically with respect to the image scaling, these results only illustrate OTE's benefit. A more complex and real-world ablation study is provided in the next section.

More interestingly, as shown in figure 4.1, SU-Net with OTE does extract the correct information during training and so can generalise. Both architectures are only trained on 16×16 images, for which the task can be solved using either spatial or colour information. Once the image scale increases, the task can only be solved spatially. It can be seen that the network with OTE learns to use the more consistent and meaningful spatial features provided by OTE.

Another result of interest is shown in figure 4.2. The performance of SU-Net without OTE trails off geometrically with its ERF size - or, rather, the proportion of the image that its ERF covers. As the image scales up, SU-Net without OTE must resort to guessing more pixels, resulting in more and more of the image being classified as "top". SU-Net without OTE is unable to learn a sense of scale from the 16×16 training data and thus cannot scale up to larger images; it is ostensibly possible for it to solve the problem perfectly up to the 64×64 case, but it does not.

4.2.2 Ablation Study

Now experiments considering a real-world scenario similar to the example dataset used in the previous section's experiments are conducted. More specifically, the five main comparison architectures presented in section 4.1.1 - U-Net, Dilated U-Net with dilation rates two and three, B-Net, and TSC-Net - are applied to the COVID-19 Infection dataset from section 3.8.1.3. The COVID-19 dataset's similar quality stems from the classification of lungs in the CT-Scans as either left or right lungs, an explicitly spatial distinction.

Table 4.1: Brief Ablation Study, network performance with and without OTE on the COVID-19 CT Scans dataset

Mean Intersection Over Union					
	U-Net	**Dilation 2**	**Dilation 3**	**B-Net**	**TSC-Net**
With OTE	0.862	0.871	0.862	0.897	**0.903**
Without OTE	0.854	0.863	0.862	0.895	**0.903**

Each of the five networks are trained five times for one hundred epochs on the COVID-19 dataset and the average MIoU achieved in each run is reported, with and without OTE. This resulted in fifty training runs and constitutes a small ablation study for OTE. There was no deviation in any of the runs to the fourth decimal place, so these results can be attained consistently. The results are presented in table 4.1.

As can be seen in the table, OTE consistently improves or at least does not negatively affect the performance of all of the models. This result is not surprising since the information provided by OTE for this dataset is at worst useless and so can easily be ignored by the network, which would result in equivalent performance as when the model does not utilise OTE.

It is worth noting that OTE does not have as significant an impact in this real-world case as in the example dataset. In this experiment, the images are 128×128, and the models are all of depth five with relatively large ERFs compared to SU-Net, and thus they can potentially account for spatial information implicitly without OTE. Interestingly, in descending order, the models on which OTE has the most significant effect are U-Net, U-Net with dilation rate 2, B-Net, U-Net with dilation rate 3, and TSC-Net. The model's with the largest ERFs benefit the least from OTE, which is expected under this interpretation. Model's with ERFs large enough to easily detect both lungs do not require the spatial information provided by OTE and so ignore it.

4.2.3 Overfitting

Given that the translational equivariance of FCNs is assumed in most of the literature, some datasets rely on it. This is particularly true for synthetic datasets, in which generating spatial and textural complexity comparable to real-world tasks is exceptionally challenging. Note, for example, the Prague dataset in section 3.8.2.2; the spatial information contained in every image is essentially the same. When using OTE, the network learns to rely on this unrealistic spatial simplicity, which results in serious over-fitting.

The Prague dataset contains just 80 images and is quite simplistic compared to the actual task of texture segmentation. When training on this data with OTE, the FCN can exploit this spatial simplicity to achieve near-perfect performance on every image in the Prague dataset while learning effectively nothing regarding actual texture analysis. Two examples of this are provided in figure 4.3. As can be seen there, the FCN overfits to the simple images on the left; the effect of that over-fitting is quite noticeable when applied to the real-world images on the right.

As mentioned before, this limitation motivated the creation of datasets that more sufficiently correlate with the downstream tasks to which the models are applied. This over-fitting is also not a problem when training on real-

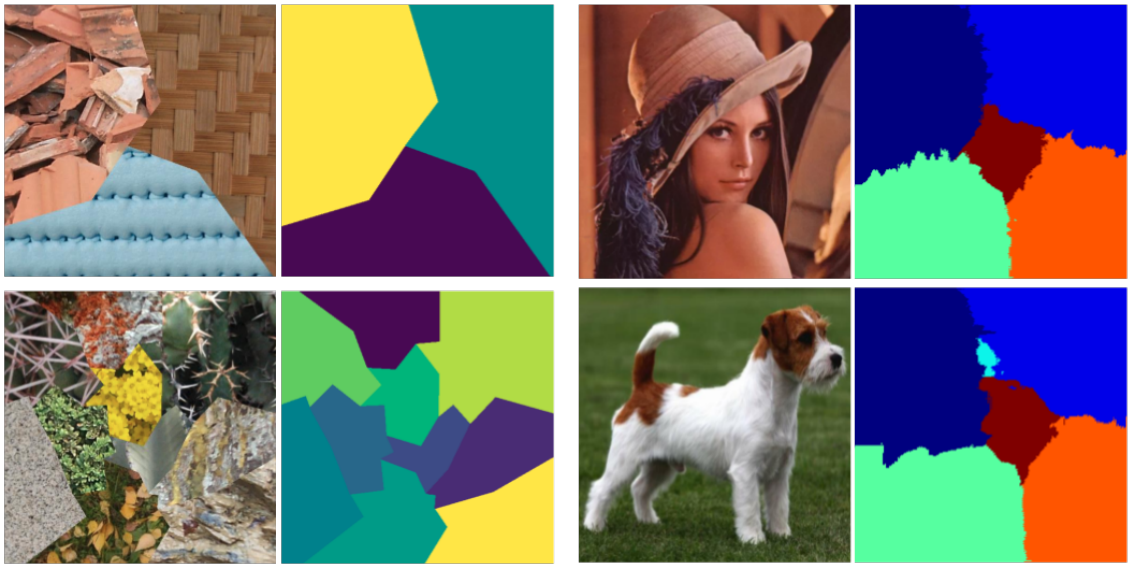


Figure 4.3: Diagram depicts two samples from Prague on the left, and two predictions made by the network on real-world images on the right

world data, in which the spatial information present in the images is of actual relevance to the task. From this point on, all of the results provided in this chapter utilise OTE. This being the case, the over-fitting discussed here is accounted for by the increased complexity of the data used. All training regimes were conducted both with and without OTE, and, as anticipated, OTE consistently improves or has no effect on the model's performance when the data is sufficiently complex.

4.3 Translated Skip Connections

4.3.1 Ablation Study

Much like the last section, it must first be established whether TSC satisfies its intended purpose, that is, geometrically expanding the receptive fields of FCNs. These experiments must also test whether a network, after having its receptive field expanded by TSC, still benefits from OTE. The first experiment is conducted on the simple synthetic dataset shown in figure 3.4, in which images contain two colours and are bisected either horizontally or vertically. This dramatically increases the dataset's complexity, in that now it is not enough for the model to be able to detect whether the kernel is on the top or bottom of the image, or even just the left and right; the model must also ascertain whether the line bisecting the image is horizontal or vertical. This task is impossible for many architectures depending on their depth and the image's resolution. Therefore, these experiments ensure that all the architectures are the same depth and contain approximately the same number of parameters; the reason this allowance is made is because larger models could perform better for reasons not related to an expansion in their receptive fields. This is reflected on more specifically in section 3.4. This section proceeds by showing that TSC with OTE makes this task solvable for a network of any depth on any image size.

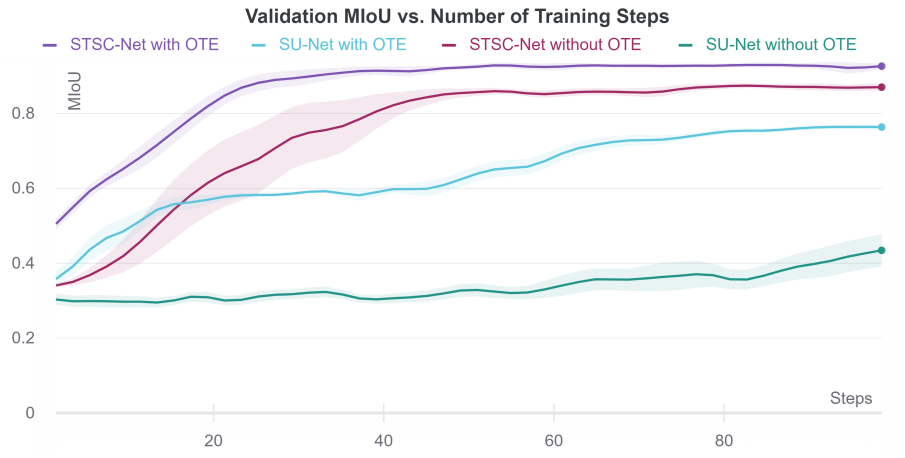


Figure 4.4: Chart depicts the validation MIoU of four different models while training on the top and left rectangle dataset, each line is averaged over five training regimes, with the standard error depicted in a lighter shade

This makes use of two different architectures introduced in section 4.1.1, SU-Net and STSC-Net, the three depth versions of U-Net and TSC-Net, respectively. These architectures are trained with and without OTE on the top and left rectangle dataset for 50 epochs each, averaging the results for each network over five training runs. All the samples in the dataset have the size 128×128. The results of this experiment are shown in figure 4.4. As can be seen there, every addition of OTE and TSC results in a great performance improvement compared to the baseline model; but, more interesting conclusions can also be drawn from this result.

Firstly, SU-Net without OTE's performance barely improves throughout training, and - contrary to the norm - the stability of the model decreases the more it trains, signifying the impossibility of it solving this task. The rest of the architectures all show convergence; however, the least stable architecture is STSC-Net without OTE. It can be argued that this is because the information provided by TSC is quite complex for the model to interpret without OTE. After all, it does not know where in the image the kernels in the TSC portion of its feature map are coming from. Two other results attest to this argument: when adding OTE to STSC-Net, the stability of convergence notably improves, and, at the beginning of training, SU-Net with OTE performs better than STSC-Net without OTE until STSC-Net learns to make use of the complex information it has access to.

Therefore, the primary conclusion that can be drawn from the results shown here and in the previous section is not just that TSC and OTE are independently useful; it can also be concluded that they are complimentary. A network with TSC benefits from OTE in that it can easily interpret where each of the translated kernels is in the feature map. Additionally, a model with OTE still benefits from TSC due to the massive increase in its receptive field.

4.3.2 Experiments and Results

This chapter now proceeds by applying TSC to more realistic datasets to validate the hypothesis that architectures will benefit from the expanded receptive fields. The TSC results are split into three tables, table 4.1 from the OTE ablation study, and tables 4.2 and 4.3. They contain results spanning five different FCN architectures

Table 4.2: Comparison of 5 different architectures on 3 different datasets

Mean Intersection Over Union[a]					
Dataset	**U-Net**	**Dilation 2**	**Dilation 3**	**B-Net**	**TSC-Net**
IILD	0.72	0.74	0.75	0.79	**0.81**
Landcover.ai	0.45	0.49	0.50	0.50	**0.53**
Carla	0.73	0.71	0.70	**0.76**	**0.76**

[a]Ranges from 0 to 1, closer to 1 is better

and five contemporary image segmentation datasets, all of the datasets were outlined in section 3.8, they are: the COVID-19 CT Scans dataset (3.8.1.3), IILD (3.8.1.1), Landcover.ai (3.8.1.2), Carla (3.8.2.1), and PASCAL VOC2012 (3.8.1.4).

The five architectures made use of were described in section 4.1.1; they are: U-Net, U-Net with dilation rates of two and three, B-Net, and the 9.6 million parameter version of TSC-Net. This experiment aims to evaluate how effectively TSCs expand the receptive fields of FCN architectures; thus, the four other architectures were chosen to represent alternative approaches to doing so, more specifically: strided pooling for U-Net, dilated convolution of varying sizes, and B-Net with strided convolution. These alternatives were described in section 2.2.1.2. Every dataset is resized to have the same image size, and all the networks have a depth of five, so those factors do not impact the results. Since the purpose of these experiments is comparison, these results do not consider combinations of different approaches, although such an architecture is possible and could result in even greater performance.

In table 4.1, results for the COVID-19 CT Scans dataset are presented. As has already been mentioned, all of the remaining results also use OTE. All architectures achieve their greatest MIoU on the COVID-19 CT Scans dataset, which is expected given that U-Net was initially designed for application to medical imaging and the other architectures are extensions of U-Net. B-Net's strided convolution provides learnable down-sampling and thus greater expressiveness, and in this domain did not cause over-fitting because the training and testing sets are fairly consistent. Despite not having strided convolution, TSC-Net's translated skip connections allow for a similar increase in expressiveness and provide a larger ERF, so TSC-Net slightly outperforms B-Net. Medical imaging is fairly spatially simplistic; lungs in a CT scan are typically in the same location, so expansions in the receptive field should not provide much benefit. This spatial simplicity would be undermined by the fact that this dataset requires labelling the left and right lungs separately; however, the fact that all the models make use of OTE makes it clear whether the left or right of the image is being observed.

Now proceeding to table 4.2, the results for IILD and Landcover.ai, two satellite imaging datasets with similar characteristics are presented first; unlike the COVID-19 dataset, satellite image segmentation certainly requires large receptive fields as objects typically span wide areas and their class is often inferred from the surrounding region. IILD and Landcover.ai are also large datasets, so over-fitting is not a concern. This explains why larger, more learnable, receptive fields consistently improve performance on both datasets. Landcover.ai has a greater label-complexity, hence the lower MIoUs achieved by all architectures on that dataset; however, the relative performance of the architectures was markedly similar in both domains.

Table 4.3: Label level performance of U-Net, dilated convolution, B-Net, and TSC-Net on VOC2012

	Mean Intersection Over Union[a]								
Architecture	**Person**	**Cat**	**Dog**	**Bus**	**Monitor**	**Boat**	**Plant**	**Bicycle**	**Overall Average**
U-Net	**0.59**	0.50	0.48	0.45	0.45	0.45	0.46	0.47	0.48
Dilation 2	0.58	**0.60**	0.41	0.38	0.50	0.49	0.46	0.47	0.49
Dilation 3	0.58	0.53	0.44	**0.57**	**0.50**	**0.50**	**0.49**	0.47	**0.50**
B-Net	0.47	0.40	0.40	0.43	0.44	0.46	0.46	0.49	0.44
TSC-Net	0.52	0.52	**0.48**	0.55	0.48	0.49	0.47	**0.49**	**0.50**

[a]Averaged over 5 100 epoch training regimes

The last result presented in Table 4.2 is for the CARLA segmentation for self-driving cars dataset. It is a synthetic dataset, and so the textures and the gradients between them are simplistic compared to real-world images; this necessarily means that higher receptive fields will confer less benefit. Thus, the expressivity of each architecture will have more impact on their performance. B-Net's learnable down-scaling and TSC-Net's learnable skip connections are the sources of their extra expressivity and better performance.

Finally, results for the PASCAL VOC2012 dataset are shown in Table 4.3; VOC2012 is the most complex domain considered in these experiments, it contains 22 labels, so random performance would result in an MIoU below 0.05. VOC2012 is also a small dataset, with approximately 1,500 training images, adding to the difficulty. Typically, researchers would apply deeper architectures to solve this domain, but these models are sufficient since the primary concern here is their relative performance. For the first time, B-Net achieved the worst performance of all the architectures; this is likely due to overfitting and will be discussed more in subsection 4.3.3. Only two of the architectures achieved the acceptable margin of 0.5 defined for MIoU [Song et al. 2021], the dilated model with dilation rate 3, and TSC-Net. Unsurprisingly, these are the architectures with the largest, learnable receptive fields; these receptive fields are an essential quality on this dataset due to the large number of correlated classes. For example, horse and grass or desk and monitor would frequently appear in single images; being able to perceive both objects - an ability determined by the size of the network's ERF - will improve performance in those cases. Both of these architectures also use strided pooling when down-sampling; this counteracts the overfitting that befell B-Net's strided convolution, which has more learnable parameters.

4.3.3 Discussion

The five models considered vary in their performance across all five of the substantially different domains experimented on. However, despite this variation, TSC-Net's performance always matches or surpasses the next best architecture. This result is justified since the advantages of the other approaches are all implicitly encapsulated in the TSC-Net architecture. These results validate the primary claim made in this section: that TSC is an independently justifiable neural network module. With that said, TSC-Net was designed for application within a wider workflow, which will be discussed in section 4.5. Before that, these results should be inspected and discussed more precisely.

Strided max-pooling provides an architecture with greater resistance towards overfitting, the trade-off being that it has no learnable parameters and so is less expressive. Contrary to this, strided convolution, as used in B-Net, provides learnable parameters and therefore greater expressivity; however, the architecture then has a greater propensity to overfit and, due to the larger number of parameters, greater memory requirements. TSC-Net can still use strided max-pooling to offset overfitting while not losing the expressivity gained by the learnable parameters in strided convolution.

The receptive fields of strided max-pooling and strided convolution are limited by the size of their filters; this limitation is overcome by dilated convolution, which can increase the receptive field without increasing the number of parameters in the filter. However, as a consequence of the gaps in each filter, dilated convolution can reduce performance on datasets that depend more on textural analysis than spatial analysis; for example, the CARLA segmentation for self-driving cars result presented in Table 4.2. A secondary consequence of dilated convolution is that the traditional optimisations of matrix multiplication cannot function with the gaps created in the convolutional filter; this greatly increases the computational time of architectures that use dilated convolution despite not increasing memory requirements. TSC-Net avoids both of those disadvantages by using traditional convolution while still increasing the receptive field of the architecture by an even greater amount than dilated convolution. This is confirmed empirically by showing that even in domains that dilated convolution works effectively on, such as VOC2012 in Table 4.3 and IILD in Table 4.2, TSC-Net still confers either equal or greater benefit, with much lower computational time.

A limitation of TSCs is that they depend entirely on the encoder/decoder structure and can only function in autoencoders and FCNs, whereas the other approaches discussed here can extend to traditional CNNs and - in the case of skip connections - even traditional neural networks. This greater specificity towards FCNs is likely why the proposed module outperforms other similar approaches in this context.

4.4 Permutation-invariant Loss Function

In order to evaluate a learned alternative to algorithmic image segmentation with FCNs, a dataset more complex than those available in the literature was required. A hurdle in producing this dataset is that the network has to learn from it; ideally, it should be an automatically generated dataset with minimal human intervention since the segmentation techniques this research desires to replace require no such training. To achieve this form of training, this research resorts to a permutation-invariant loss function since the approaches proposed in Andrearczyk and Whelan [2017] and Huang et al. [2019], among others, require the human intervention in pre-segmentation that this work wishes to avoid. The development of the loss function is discussed in section 3.7, this section shows the improved time complexity of HCE over a more naive implementation and a small performance gain it provides outside of the context of the proposed workflow.

Figure 4.5 contains two charts; chart (a) shows a time comparison of the proposed approaches. The algorithmic complexities of HCE, the Naive approach to HCE, and BCE are $O(n^3)$, $O(n!)$, and $O(n^2)$, respectively, in the label space. Those are worst-case complexities, and so the actual time performance achieved by each algorithm as the number of classes in the prediction and ground truth increases is shown here; the Naive approach becomes impractical at 7 labels, while HCE functions adequately even on the dataset with the most labels considered in these experiments, VOC2012 with 22 classes.

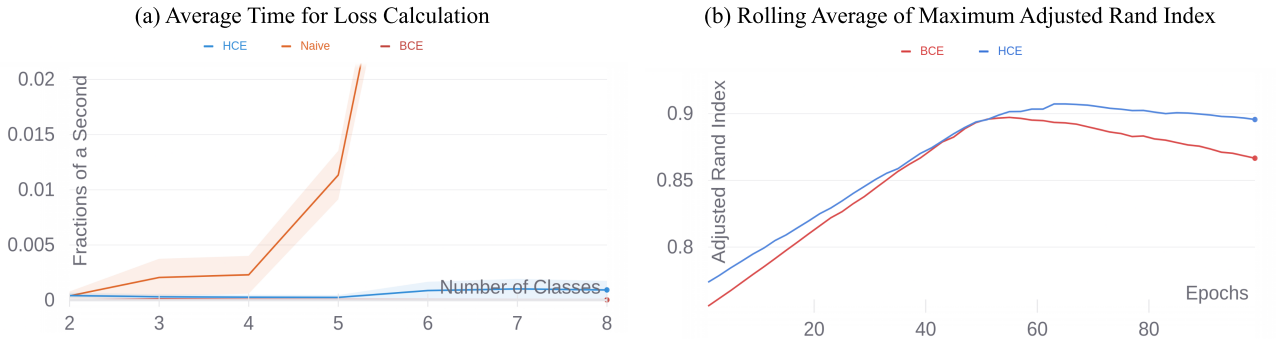


Figure 4.5: Chart (a) shows running time of HCE, BCE, and the naive permutation-invariant loss function. Chart (b) depicts the rolling average of maximums of the adjusted rand index achieved by 2 groups of 5 models trained with BCE and HCE, respectively, on the COVID-19 datset.

Table 4.4: Performance on Prime Texture Analysis Test set of different architectures

Architecture	Adjusted Rand Index	Adjusted Mutual Information	Hungarian Cross-Entropy
U-Net	0.42	0.46	0.98
Dilation 2	0.41	0.42	0.93
Dilation 3	0.60	0.58	0.69
B-Net	0.85	0.84	0.23
TSC-Net	**0.91**	**0.89**	**0.15**

Secondarily, TSC-Net is also trained on the semantic COVID-19 dataset from section 3.8.1.3 with and without HCE. The rationale behind this decision is that, even when applied to a fully semantic dataset, permutation-invariant networks are usually initialised closer to their best performance and thus converge faster and more effectively. Each line on the right of figure 4.5 shows the average of the maximum adjusted rand index - a popular performance metric in unsupervised learning discussed in section 2.3.2 - that TSC-Net achieved with and without HCE. The permutation-invariant models start higher on average and reach a higher peak by the end of training; this is achieved by simply altering the loss function. It could be theorised that this would be the case because it is fundamentally arbitrary whether the background or foreground is consistently labelled 0, and the other 1, or vice versa, as long as the same information is transferred. A permutation-invariant loss function is training exclusively for this information transfer, as opposed to a semantic loss function which is training to perform that information transfer while also obeying the arbitrary labelling scheme.

4.5 Proposed Workflow

4.5.1 Benchmark Performance

As was explained in section 3.8.3.3, this research has released a synthetic dataset called PTAD with which the neural networks used in the remainder of this section are trained. PTAD is split into training, validation,

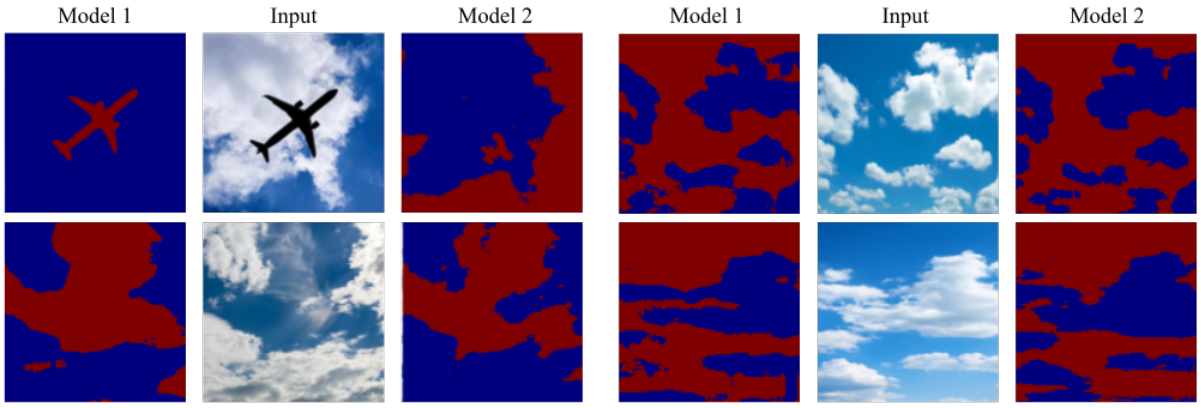


Figure 4.6: Diagram showing object proposals made by TSC-Net trained with different ERFs, model 1 has a smaller ERF than model 2 but both produce sensible object proposals

and test sets to create a benchmark more spatially complex than the Prague texture segmentation benchmark and data generator; the sets are of size 80000, 20000, and 5000, respectively. PTAD's creation was necessary to prevent the critical addition of OTE from causing over-fitting on an overly simplistic dataset. Each of the comparison architectures and TSC-Net are trained on the 80000 training images for 25 epochs, keeping the version of the model that minimises the validation loss, and finally its performance is evaluated on the test set.

Given that the entire pipeline is being used here, HCE must be incorporated so that the model does not produce semantic labels but instead chooses its labelling scheme automatically. Therefore, to properly assess the models, the unsupervised metrics Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI) - both of which should be maximised - must be used. The HCE Loss itself is also reported, which should be minimised. Both AMI and ARI are reported because they are subtly different in what they measure, as was described in section 2.3.2. In this case, ARI should be prioritised [Romano et al. 2016], but both are presented to justify these conclusions further.

The results of this experiment are shown in Table 4.4. TSC-Net outperforms all of the other models, this time by an even larger margin than was witnessed in section 4.3. The use of HCE necessitates a larger receptive field for the network to be able to perceive which label values have already been used in the image; if the network cannot perceive other parts of the image, it might assign the same label to different objects in the image since it cannot derive the correct label from the object itself. Additionally, the ARI reported for the Dilation 3 model, B-Net, and TSC-Net, are greater than the AMI, whereas, for the two models with the smallest, least expressive receptive fields, the reverse is true. Given that ARI is the preferred metric for datasets like PTAD, these results suggest that the former three architectures are better suited to the task. Finally, the results in Table 4.4 also strengthen the previous claims regarding HCE from section 4.4. Note that a decrease in the HCE Loss achieved by the model corresponds quite closely to an increase in the AMI and ARI, which shows that HCE can adequately train a network to satisfy the unsupervised learning metrics agreed upon in the literature.

4.5.2 Consistency

The results also have to demonstrate that the proposed approach has some consistency to sufficiently compete with algorithmic segmentation techniques. If the quality of a model's performance is highly dependent on initialisation or some mild parameter tuning, then researchers are understandably less likely to take on the task of creating and training them. While the consistency of the proposed architecture has already been implicitly demonstrated by the fact that it has trained with three different depths, multiple different image sizes, and on six different datasets throughout this chapter, here those results are augmented with some visualisations.

Two different versions of TSC-Net are trained on the PTAD dataset from section 3.8.3.3, model 1 and model 2, on two different image sizes, 256×256 and 64×64, respectively. This means that the proportion of the inputs covered by the ERFs of the models are substantially different. The objects proposed by these models are shown in figure 4.6, however they are refined down to two labels in every case. Considering that both models were entirely trained on synthetic data and have substantially different ERFs, the predictions they make are remarkably consistent. In these examples, the models chose the same labelling scheme; however, that is arbitrary. The primary distinctions between the predictions made by each model are interpretable; since model 1 has the smaller ERF, the objects it proposes are less spatially dispersed and, especially in the top left prediction, smaller. This shows that the architecture's design can impact the output of the network. If the downstream object proposal task being designed towards contains smaller objects, an architecture with a smaller ERF should be given preference. However, importantly, the predictions made by both models are still reasonable. With this being shown, these experiments may finally compare the proposed approach to using traditional segmentation algorithms in the OBIA workflow.

4.5.3 Comparison to Traditional Algorithms

Finally, this chapter arrives at a comparison of the proposed workflow to two popular alternatives used in proprietary OBIA software, Watershed and SLIC; both were described in detail in section 2.1.2.2. These approaches will be compared to the end-to-end workflow described in section 3.2, which is the focus of this dissertation. This comparison is fair because - much like traditional segmentation algorithms - the proposed approach requires no human labelling, it can generalise to different image resolutions, and it produces consistent outputs. This also justifies the choice to use self-supervised learning and synthetic data instead of pre-training on real-world segmentation datasets, which would not generalise as effectively to datasets outside the chosen domain and require human annotation.

The first step in this workflow is training an FCN on a texture analysis dataset that provides generalisability sufficient for the downstream task of object proposal; the dataset made use of is PTAD from section 3.8.3.3. Generalisability will be illustrated later by testing on vastly different images, including people, animals, clouds, lakes, and rivers. As shown by the experiments in section 4.5.1, an FCN capable of learning from the PTAD dataset was not present in the wider literature; for this purpose, the full 12 million parameter version of TSC-Net with OTE is applied, as well as using HCE, the proposed permutation-invariant loss function. The network is trained on PTAD with a batch size of 8 for 100 epochs.

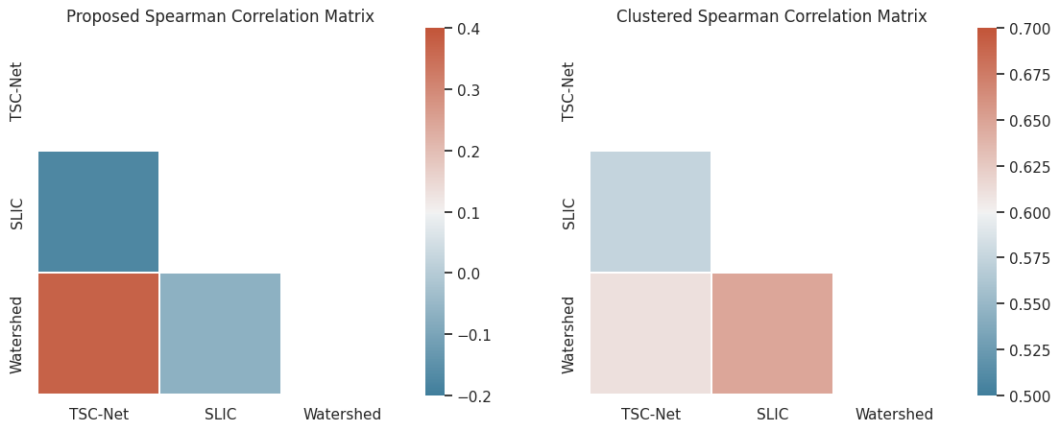


Figure 4.7: Left: Heatmap of Spearman correlation coefficients of each method's proposed objects. Right: Heatmap of Spearman correlation coefficients of the final classification of each method's proposed objects.

The resultant network is then used for object proposal; the clustering algorithm that is used to extract the desired objects from those proposed is spectral clustering from section 2.1.2.4. For all three approaches, Watershed, SLIC, and TSC-Net, a combination of GLCM homogeneity, GLCM energy, GLCM contrast, mean pixel intensity, and shape index is used in the feature vectors of each object; detailed descriptions of every one of these features is provided in section 2.1.2.3. Given the general application and structure of OBIA, which requires human oversight and tuning, a dataset of 51 images is considered - this sample is large enough to validate the hypothesis to a statistically significant degree but small enough so that Watershed and SLIC can be tuned to produce representative outputs.

The entire workflow is for each image, using all three object proposal methods and manually tuning Watershed and SLIC to produce competitive results. To validate the first part of the research hypothesis - that the proposed workflow produces results comparable to those of traditional segmentation techniques - a statistical correlation test is performed using the Spearman correlation coefficient, which is widely used when comparing discrete random variables; the results of this correlation test are shown in figure 4.7. Spearman Coefficients fall on the range [-1.0, 1.0], where -1.0 and 1.0 denote perfect negative and positive correlations, respectively. Given two proposed labelling schemes produced by different functions, a negative correlation suggests that the two functions consistently choose different label values for the same input and vice-versa for a positive correlation.

These two correlation matrices correspond to 6 null hypotheses regarding the correlation of the 3 methods, 3 for the objects proposed by the methods and 3 for the final clustered results. In all but two cases, the P-value suggests a statistically significant correlation between the methods with $p < 0.01$, so the null hypotheses can be rejected. The two accepted null hypotheses state that there is no statistically significant correlation between the objects proposed by SLIC and those proposed by Watershed and TSC-Net. This is anticipated because SLIC proposes substantially more objects than Watershed and TSC-Net, as will be shown later in this chapter.

The objects proposed by TSC-Net result in higher correlation coefficients with SLIC and Watershed than the two traditional segmentation algorithms have with each other. This relationship is flipped when considering the final OBIA clusters, with Watershed and SLIC being most highly correlated with coefficient 0.648, followed closely by TSC-Net and Watershed with coefficient 0.611, and lastly, TSC-Net and SLIC with coefficient 0.575. There is a 95% certainty that the true correlation coefficient of the Watershed and SLIC pair falls on the

interval $[0.4534, 0.7836]$, suggesting substantial overlap with the 95% confidence interval $[0.4033, 0.7588]$ for the correlation coefficient of the TSC-Net and Watershed pair.

TSC-Net is correlated with Watershed and SLIC with $P <= 0.00001$, and the result is statistically significant at $p < 0.01$. This shows that the methods perform comparably, thus validating the first part of the research hypothesis. The second part of the research hypothesis considers whether the proposed method performs better than traditional methods in some cases. Due to the nature of the OBIA workflow and the human-tuning required to produce the final classification results, as well as the subjectivity of what constitutes superior performance, this experiment proceeds qualitatively. Samples of the objects proposed by the 3 methods and the final predictions of the OBIA workflow are depicted in figure 4.8.

Watershed is most closely correlated with the object proposal TSC-Net produces and so it will be considered first; both Watershed and TSC-Net propose objects larger than their alternative in SLIC. Watershed proposes sensible objects; however, it requires more substantial tuning to get the desired results. The location of the initial seed points and whether thresholding is performed in grey-scale or colour has a substantial effect on the output. This extra tuning is likely required due to the larger objects Watershed recommends, making more assumptions about the type of objects being searched for. The benefit of recommending larger objects is that the segmentation algorithm performs more of the work that the clustering algorithm would otherwise be left with, making the classification step more straightforward and accurate. This is contrary to SLIC, which recommends many smaller objects compared to Watershed and TSC-Net.

SLIC is much less sensitive to tuning and correspondingly much less tunable than Watershed. Given that its proposed objects are relatively small and texturally consistent, it is unlikely to make unreasonable proposals. One downside is that the actual object map produced by SLIC does not contain much meaning at all; it simply corresponds to a textural superpixel grouping. As a result, a more significant burden is placed upon spectral clustering in the final step; thus, results equivalent to those obtained by Watershed and TSC-Net require a greater focus on tuning the final clustering algorithm.

Finally, consider TSC-Net; given that the model is pre-trained on a synthetic dataset, tuning is limited to modifying the predictions made by the FCN using the refinement step described in section 2.2.2.3. However, because the model is trained on textures recognisable to humans, tuning is not as necessary as in the algorithmic case; this is a notable advantage of using a learnable approach. For example, consider image (e); the proposed method recognises fur as a distinct textural category regardless of its colour, which corresponds more closely to human interpretation than both SLIC's and Watershed's solutions. Another consequence of this learnability is that the model proposes objects recognisable to humans, much like Watershed does when tuned correctly, which makes the clustering step that follows much simpler than in SLIC's case. However, a disadvantage of using FCNs in this way was already noticed by Mboga et al. [2019] - FCNs tend to propose smooth object boundaries. While Mboga et al. [2019] noticed this in the fully supervised case, there was no reason or expectation for it not to transfer to the self-supervised case.

The results shown in this section demonstrate that using FCNs within the proposed workflow produces outcomes comparable or superior to algorithmic segmentation techniques. These results also show that the segmentation maps produced are generalisable enough to apply in various settings. The predictions made by networks trained on synthetic data are generally unusable, but, as hypothesised, embedding a modified version of a traditional FCN architecture in the OBIA workflow can account for many limitations. Within the OBIA

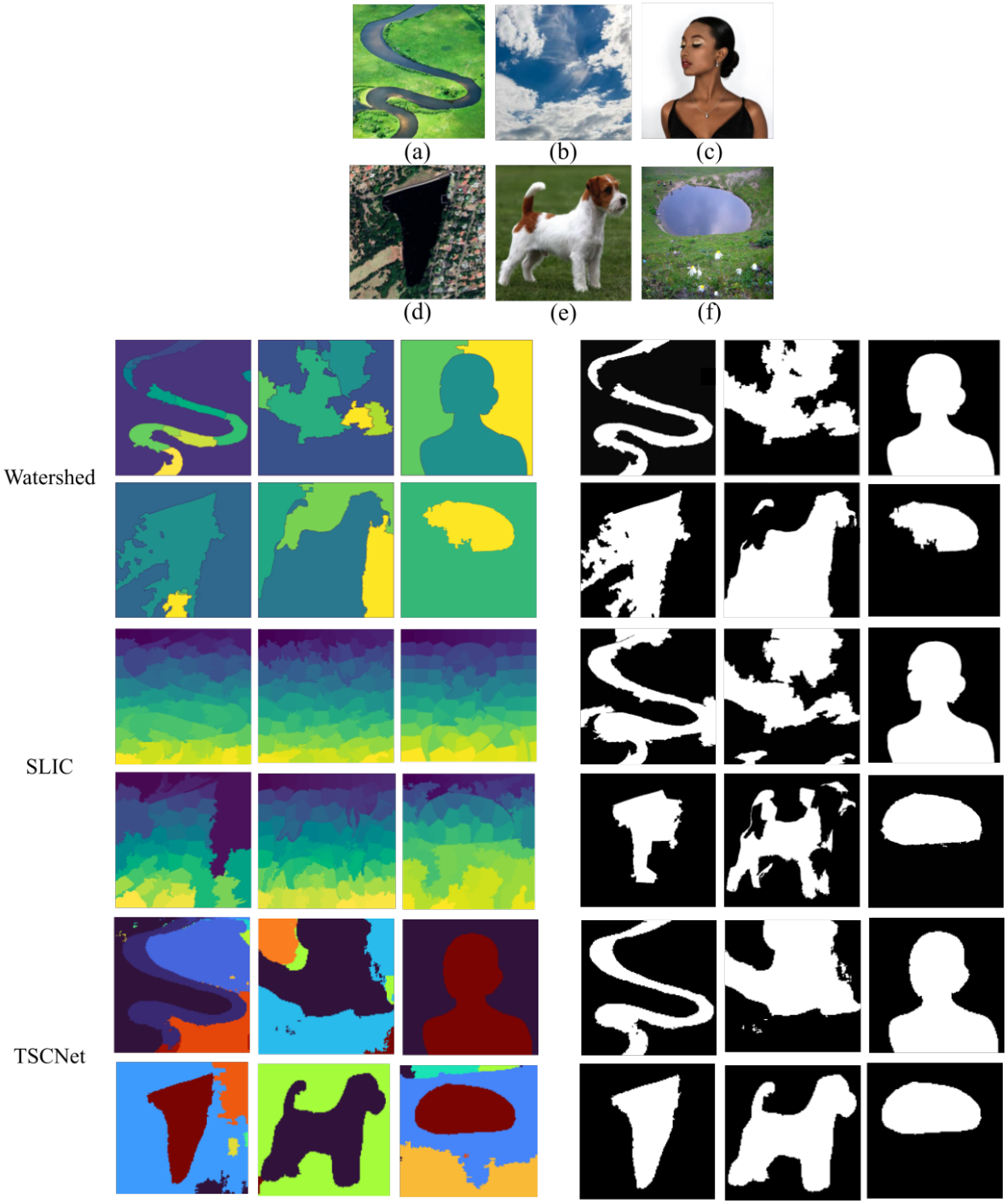


Figure 4.8: Comparison of Watershed, SLIC, and TSC-Net in the object proposal and classification steps of the OBIA workflow.

workflow, an FCN trained completely with self-supervised learning on a fully-synthetic dataset can produce results equivalent or superior to those attained by algorithmic segmentation techniques with suitable generalisability and no human labelling.

Chapter 5

Conclusion

5.1 Summary

This work introduces a modified fully convolutional neural network into the object proposal step of the object-based image analysis workflow. The justification for making this modification to the OBIA workflow is three-fold:

- Introducing learnability into the object proposal step of OBIA can improve performance in some cases
- Training a single generalisable and reusable FCN can reduce the amount of tuning required in the object proposal step when compared to traditional segmentation algorithms
- A larger number of approaches to object proposal could reduce the field's reliance on proprietary software for the implementation of complex, hybrid segmentation algorithms

In order to succeed in this research aim, the FCN had to satisfy certain constraints since it was to replace the algorithmic segmentation techniques that currently predominate in OBIA. Algorithmic segmentation techniques generally function effectively for any image size, and so the FCN should be able to generalise to multiple image sizes from a single training regime. Algorithmic segmentation techniques do not require the manual annotation of training data, adding a further constraint that the FCN should learn from a synthetic or automatically annotated dataset. Lastly, algorithmic segmentation techniques are applicable in multiple domains and thus do not associate specific labels with semantic categories; the FCN also had to be capable of that label agnosticism. Given these constraints, this research proposed two significant modifications to FCN architectures.

The first of these alterations is optional translational equivariance (OTE), introduced in section 3.5. Given that the FCN would now label things based not on their correspondence to a specific category but instead according to their relation to other objects in the input image, greater spatial awareness is required. Additionally, without explicitly considering spatial information, adapting to different image resolutions is a substantial challenge. This necessitates that the FCN forego translational equivariance and instead explicitly consider whether the coordinate of each pixel is relevant to the label that should be assigned. OTE achieves this by representing the spatial information of the image texturally and concatenating it onto each pixel's feature vector - thus making translational equivariance optional instead of inevitable. In section 4.2 it is found that OTE can make the FCN adaptive to different image resolutions and improve performance on specific tasks. Further, since the FCN can ignore the additional features if they are not significant for the given task, OTE never worsens performance.

Finally, in section 4.3, it is shown that OTE can overfit to spatial information in simplistic synthetic training data; however, that limitation does not transfer to real-world images in which the spatial information contained in the training and test datasets is more correlated.

Following OTE, the translated skip connection (TSC) neural network module is introduced in section 3.6. It aims to expand the receptive field of neural networks significantly enough to solve the problem of object proposal, in which an object in one part of an image can often affect the correct label assignment of the kernel in a different part of the image. This work illustrated that notion with a simple example dataset, in which an FCN had to identify whether a line bisecting an image was horizontal or vertical. Without resolutions small enough or networks large enough, this task is impossible without substantial expansions to the network's ERF. TSCs expand the network's ERF by continually translating skipped inputs by different factors as they propagate up the decoder portion of the network. Several experiments relating to TSCs are carried out in section 4.3, beginning with an ablation study with both OTE and TSC. Interestingly, that ablation study not only shows that networks with TSC and OTE outperform networks with neither component but also that the two proposed methods are complimentary. The translated information provided by TSC is difficult for a network to interpret without OTE. In section 4.3.3 the results of an experiment comparing a network containing both OTE and TSC - dubbed TSC-Net - to four other neural network architectures across five distinct semantic segmentation datasets are discussed. TSC-Net matches or outperforms the next best architecture on all five datasets.

With an FCN architecture capable of performing the task of object proposal, a loss function that can assess and train it in a permutation-invariant fashion so as not to require human annotation of objects in the synthetic dataset was still required. Mutual Information score - a permutation-invariant clustering metric - would have been ideal for this task. However, it is not computationally tractable to calculate for every batch in an FCN's training epoch, nor is it differentiable for use with stochastic gradient descent. Seven different loss functions from the literature were evaluated, all of which are described in section 2.2.3, before resorting to designing a self-augmented loss function called Hungarian cross-entropy in section 3.7. The loss function augments the data with a permutation function to find the labelling scheme that maximally overlaps with the network's own before computing the loss, allowing the network to select any labelling scheme as long as it corresponds spatially and in density with that of the ground truth. In section 4.4, HCE is compared to CE on the COVID-19 CT Scans dataset. Although HCE is less computationally efficient, permutation-invariant networks are, predictably, initialised more closely to the optimal solution and, more surprisingly, reach a better solution by the end of training.

TSC-Net was then trained with HCE on a synthetic dataset called the Prime Texture Analysis Dataset (PTAD), a dataset inspired by the Prague Texture Segmentation Dataset, with over 5600 distinct textures spanning 57 semantic categories. This dataset was specifically designed to encapsulate the problem of texture analysis while obeying the constraints outlined earlier; a detailed description of the generation process can be found in section 3.8.3.3, and the dataset has since been publicly released[1]. The quantity of textures present in the data is made possible by HCE, since the network only requires one-hot encoding vectors equal in length to the maximum number of objects in each image, as opposed to requiring support for 5600 textures or 57 semantic categories. The architecture could now be embedded into the OBIA workflow and evaluated against the following research hypothesis, which was established in section 3.3:

[1] https://webwtad.firebaseapp.com/

> An FCN trained using a synthetic texture analysis dataset can effectively replace traditional segmentation algorithms in the object proposal step of the object-based image analysis workflow with comparable or superior performance while maintaining a degree of generalisability

The proposed workflow comprises a neural network architecture called TSC-Net - incorporating OTE, TSC, and HCE - trained on the PTAD dataset. The architecture, once trained, is used in the object proposal step of the OBIA workflow before performing spectral clustering. TSC-Net is first benchmarked against four other networks on the PTAD dataset in section 4.5.1, in which it outperforms all of them. Additionally, the results in section 4.5.2 demonstrate that TSC-Net's training is robust to changes in scale and generalisable, validating the portion of the research hypothesis dedicated to generalisability. The remainder of the research hypothesis outlines two distinct requirements that are addressed in section 4.5.3. First, that the proposed workflow performs comparably to its alternatives, and second, that it can outperform them. Concerning the former requirement, a statistical p-test is performed to verify that the outputs of an OBIA workflow using TSC-Net in the object proposal step are statistically correlated with workflows using SLIC and Watershed and that the correlation persists through both the object proposal and clustering stages. Finally, a qualitative assessment of the proposed workflow that compares it to those two alternatives is performed. This assessment demonstrated some of the advantages of using a learned object proposal technique, primarily, greater accordance with human-interpretable textures. It is hoped that the TSC-Net architecture and the contributions that allowed for its creation are simply a starting point for further research and that the greater complexity found in the PTAD dataset could serve as a more realistic and challenging texture analysis benchmark.

5.2 Limitation Discussion

Although the research hypothesis was empirically validated and thus it can be concluded that incorporating learnability into the paradigm of OBIA can be beneficial and at the very least will perform comparably to existing approaches, it is necessary to restate the limitations of applying this research in other domains. Firstly, this research makes no attempt to validate the performance of TSC, OTE, or HCE outside the context of FCNs. Reasonable assumptions can be made about how these techniques may generalise to a different context, but these considerations are left for future work.

Further, this work does not exhaustively explore or quantify what properties allow for a object proposal task to be generalised to from a self-supervised texture analysis pre-text task. It is not trivial to ensure that the domain is accurately represented by the pre-text task. However, an adequate pre-text task is essential for the successful application of this approach. Indeed, in some cases it may be impossible to create a synthetic dataset capable of generalising to the downstream task or expansions to the texture database may be required.

Finally, while the benefits of learnability were established in this work, they may not be clear in all cases. Some domains, such as crop detection, will require less tuning than those that were considered in this research, and in those cases algorithmic segmentation techniques may be preferable. Alternatively, certain domains contain more uni-modal textures, such as the detection of specific tree species Xie et al. [2008], and so having a broader, learned understanding of how humans characterise texture may not improve performance as much in those cases. Given these limitations, the following future work based on this research may be carried out.

5.3 Future Research

A plethora of future work could still be conducted. For example, using more advanced pyramid architectures could allow for a model that proposed objects at different scales, allowing for researchers to more easily select for their preference. Alternatively, customising the training regime to explicitly target objects of specific sizes through deep learning clustering loss functions or regularisers could also provide greater tunability. Lastly, evaluating FCN-hybrid models in the object proposal step of the OBIA workflow, akin to those used by Mboga et al. [2019] in the clustering step, would be of interest.

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2010). Slic superpixels. Technical report.

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322.

Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., and Cremers, D. (2018). Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*.

Andrearczyk, V. and Whelan, P. F. (2017). Texture segmentation with fully convolutional networks. *arXiv preprint arXiv:1703.05230*.

Asai, M. (2018). Set cross entropy: Likelihood-based permutation invariant loss function for probability distributions. *arXiv preprint arXiv:1812.01217*.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. (2018). MINE: Mutual Information Neural Estimation. *arXiv preprint arXiv:1801.04062*.

Bhardwaj, G. and Kumar, A. (2019). The comparison of shape indices and perimeter interface of selected protected areas especially with reference to sariska tiger reserve, india. *Global Ecology and Conservation*, 17:e00504.

Blaschke, T. (2010). Object based image analysis for remote sensing. *ISPRS journal of photogrammetry and remote sensing*, 65(1):2–16.

Boccardo, P., Borgogno Mondino, E., Giulio Tonolo, F., and LINGUA, A. (2004). Orthorectification of high resolution satellite images.

Boguszewski, A., Batorski, D., Ziemba-Jankowska, N., Zambrzycka, A., and Dziedzic, T. (2020). Landcover. ai: Dataset for automatic mapping of buildings, woodlands and water from aerial imagery. *arXiv preprint arXiv:2005.02264*.

Bruton, J. and Wang, H. (2021). Dictionary learning for clustering on hyperspectral images. *Signal, Image and Video Processing*, 15(2):255–261.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , and Vedaldi, A. (2014). Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Crum, W. R., Camara, O., and Hill, D. L. (2006). Generalized overlap measures for evaluation and validation in medical image analysis. *IEEE transactions on medical imaging*, 25(11):1451–1461.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Doersch, C. and Zisserman, A. (2017). Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060.

Donsker, M. and Varadhan, S. (1975). Large deviations for markov processes and the asymptotic evaluation of certain markov process expectations for large times. In *Probabilistic Methods in Differential Equations*, pages 82–88. Springer.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

Dronova, I. (2015). Object-based image analysis in wetland research: A review. *Remote Sensing*, 7(5):6380–6413.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.

Gao, W., Zhang, X., Yang, L., and Liu, H. (2010). An improved sobel edge detection. In *2010 3rd international conference on computer science and information technology*, volume 5, pages 67–71. IEEE.

Gebejes, A. and Huertas, R. (2013). Texture characterization based on grey-level co-occurrence matrix. *databases*, 9:10.

Glick, Y. (2020). COVID-19 pneumonia. https://radiopaedia.org/playlists/25887.

Google (2019). TensorFlow Datasets, a collection of ready-to-use datasets. https://www.tensorflow.org/datasets.

Google (2021). Google earth professional. https://www.google.com/earth/versions/.

Green, R. O., Chrien, T. G., Nielson, P., Sarture, C. M., Eng, B. T., Chovit, C. J., Murray, A. T., Eastwood, M. L., and Novack, H. I. (1993). Airborne visible/infrared imaging spectrometer (aviris): recent improvements to the sensor and data facility. In *Defense, Security, and Sensing*.

Gusella, L., Adams, B. J., Bitelli, G., Huyck, C. K., and Mognol, A. (2005). Object-oriented image understanding and post-earthquake damage assessment for the 2003 bam, iran, earthquake. *Earthquake Spectra*, 21(S1):225–238.

Haindl, M. and Mikes, S. (2008). Texture segmentation benchmark. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hoang, H. G., Vo, B.-N., Vo, B.-T., and Mahler, R. (2015). The cauchy–schwarz divergence for poisson point processes. *IEEE Transactions on Information Theory*, 61(8):4475–4485.

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer.

Hossain, M. D. and Chen, D. (2019). Segmentation for object-based image analysis (obia): A review of algorithms and challenges from remote sensing perspective. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:115–134.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, Y., Zhou, F., and Gilles, J. (2019). Empirical curvelet based fully convolutional network for supervised texture image segmentation. *Neurocomputing*, 349:31–43.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

Ikonomatakis, N., Plataniotis, K., Zervakis, M., and Venetsanopoulos, A. (1997). Region growing and region merging image segmentation. In *Proceedings of 13th International Conference on Digital Signal Processing*, volume 1, pages 299–302. IEEE.

Jadon, S. (2020). A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7. IEEE.

Jun, M., Cheng, G., Yixin, W., Xingle, A., Jiantao, G., Ziqi, Y., Minqing, Z., Xin, L., Xueyuan, D., Shucheng, C., Hao, W., Sen, M., Xiaoyu, Y., Ziwei, N., Chen, L., Lu, T., Yuntao, Z., Qiongjie, Z., Guoqiang, D., and Jian, H. (2020). COVID-19 CT Lung and Infection Segmentation Dataset. https://www.zenodo.org/record/3757476.

Khan, R., Hanbury, A., and Stoettinger, J. (2010). Skin detection: A random forest approach. In *2010 IEEE International Conference on Image Processing*, pages 4613–4616. IEEE.

Kiechle, M., Storath, M., Weinmann, A., and Kleinsteuber, M. (2018). Model-based learning of local image features for unsupervised texture segmentation. *IEEE Transactions on Image Processing*, 27(4):1994–2007.

Krig, S. (2014). *Global and Regional Features*, pages 85–129. Apress, Berkeley, CA.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kurt Spencer, K., Svensson, A., and Raccuglia, O. (2019). Opensimplex. https://github.com/lmas/opensimplex.

Liu, R., Lehman, J., Molino, P., Such, F. P., Frank, E., Sergeev, A., and Yosinski, J. (2018a). An intriguing failing of convolutional neural networks and the coordconv solution. *arXiv preprint arXiv:1807.03247*.

Liu, S. (2020). Lower bounds for mutual information in representation learning. https://modelingsimulation.github.io/TeachingWriting2020/Resources/reports/ShengLiu.pdf.

Liu, T., Abd-Elrahman, A., Morton, J., and Wilhelm, V. L. (2018b). Comparing fully convolutional networks, random forest, support vector machine, and patch-based deep convolutional neural networks for object-based wetland mapping using images from small unmanned aircraft system. *GIScience & remote sensing*, 55(2):243–264.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Lowekamp, B. C., Chen, D. T., Yaniv, Z., and Yoo, T. S. (2018). Scalable simple linear iterative clustering (sslic) using a generic and parallel approach. *arXiv preprint arXiv:1806.08741*.

Lucieer, V. and Lamarche, G. (2011). Unsupervised fuzzy classification and object-based image analysis of multibeam data to map deep water substrates, cook strait, new zealand. *Continental Shelf Research*, 31(11):1236 – 1247.

Luo, K., Tao, F., and Moiwo, J. P. (2017). When to detect changes in object-based image analysis: Before or after classification?

Luo, W., Li, Y., Urtasun, R., and Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4905–4913.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2017). Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3226–3229. IEEE.

Manickavelu, K. (2018). Semantic segmentation for self driving cars. https://www.kaggle.com/kumaresanmanickavelu/lyft-udacity-challenge.

Mao, X.-J., Shen, C., and Yang, Y.-B. (2016). Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv preprint arXiv:1606.08921*.

Mboga, N., Georganos, S., Grippa, T., Lennert, M., Vanhuysse, S., and Wolff, E. (2019). Fully convolutional networks and geographic object-based image analysis for the classification of vhr imagery. *Remote Sensing*, 11(5):597.

Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571.

Mohamed, N. A., Zulkifley, M. A., Zaki, W. M. D. W., and Hussain, A. (2019). An automated glaucoma screening system using cup-to-disc ratio via simple linear iterative clustering superpixel approach. *Biomedical Signal Processing and Control*, 53:101454.

Moore, A. (2001). Information gain. https://netman.aiops.org/~peidan/ANM2020/2.MachineLearningBasics/ReadingList/infogain.pdf.

Muthukrishnan, R. and Radha, M. (2011). Edge detection techniques for image segmentation. *International Journal of Computer Science & Information Technology*, 3(6):259–267.

Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., and Gambardella, L. M. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347. IEEE.

Neumann, M., Pinto, A. S., Zhai, X., and Houlsby, N. (2019). In-domain representation learning for remote sensing. *arXiv preprint arXiv:1911.06721*.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, USA:.

Nousi, P. and Tefas, A. (2018). Self-supervised autoencoders for clustering and classification. *Evolving Systems*, pages 1–14.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

Pape, C., Remme, R., Wolny, A., Olberg, S., Wolf, S., Cerrone, L., Cortese, M., Klaus, S., Lucic, B., Ullrich, S., et al. (2021). Microscopy-based assay for semi-quantitative detection of sars-cov-2 specific antibodies in human sera: A semi-quantitative, high throughput, microscopy-based assay expands existing approaches to measure sars-cov-2 specific antibody levels in human sera. *BioEssays*, 43(3):2000257.

Peng, J., Pedersoli, M., and Desrosiers, C. (2020). Mutual information deep regularization for semi-supervised segmentation. In *Medical Imaging with Deep Learning*, pages 601–613. PMLR.

Péteri, R., Fazekas, S., and Huiskes, M. J. (2010). Dyntex: A comprehensive database of dynamic textures. 31(12):1627–1632.

Platt, R. V. and Rapoza, L. (2008). An evaluation of an object-oriented paradigm for land use/land cover classification. *The Professional Geographer*, 60(1):87–100.

Prince, S. (2012). *Computer Vision: Models Learning and Inference*. Cambridge University Press.

Qin, F., Guo, J., and Lang, F. (2014). Superpixel segmentation for polarimetric sar imagery using local iterative clustering. *IEEE Geoscience and Remote Sensing Letters*, 12(1):13–17.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666.

Romano, S., Vinh, N. X., Bailey, J., and Verspoor, K. (2016). Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17(1):4635–4666.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243.

Sadeghi-Tehran, P., Virlet, N., Ampe, E. M., Reyns, P., and Hawkesford, M. J. (2019). Deepcount: in-field automatic quantification of wheat spikes using simple linear iterative clustering and deep convolutional neural networks. *Frontiers in plant science*, 10:1176.

Sokolova, M., Japkowicz, N., and Szpakowicz, S. (2006). Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer.

Song, J., Ahn, W., Park, S., and Lim, M. (2021). Failure detection for semantic segmentation on road scenes using deep learning. *Applied Sciences*, 11(4):1870.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., and Cardoso, M. J. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer.

Sumbul, G., Charfuelan, M., Demir, B., and Markl, V. (2019). Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5901–5904. IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Veljanovski, T., Kanjir, U., and Ostir, K. (2011). Object-based image analysis of remote sensing data. *Geodetski vestnik*, 55(4):678–688.

Versaci, M. and Morabito, F. C. (2021). Image edge detection: A new approach based on fuzzy entropy and fuzzy divergence. *International Journal of Fuzzy Systems*, pages 1–19.

Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):583–598.

Vinh, N. X., Epps, J., and Bailey, J. (2009). Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1073–1080, New York, NY, USA. Association for Computing Machinery.

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 975–982. IEEE.

Wu, C., Zheng, J., Feng, Z., Zhang, H., Zhang, L., Cao, J., and Yan, H. (2020). Fuzzy slic: Fuzzy simple linear iterative clustering. *IEEE Transactions on Circuits and Systems for Video Technology*.

Xie, Z., Roberts, C., and Johnson, B. (2008). Object-based target search using remotely sensed data: A case study in detecting invasive exotic australian pine in south florida. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(6):647–660.

Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485.

Zijdenbos, A., Dawant, B., Margolin, R., and Palmer, A. C. (1994). Morphometric analysis of white matter lesions in mr images: method and validation. *IEEE transactions on medical imaging*, 13 4:716–24.