# River Ice Segmentation under a Limited Compute and Annotation Budget

by

Daniel Sola

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2022

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

River ice segmentation, used to differentiate ice and water, can give valuable information regarding ice cover and ice distribution. These are important factors when evaluating flooding risks caused by ice jams that may harm local ecosystems and infrastructure. Furthermore, discriminating specifically between anchor ice and frazil ice is important in understanding sediment transport and release events that can affect geomorphology and cause landslide risks. Modern deep learning techniques have proved to deliver promising segmentation results; however, they can require hours of expensive manual image labelling, can show poor generalization ability, and can be inefficient when hardware and computing power are limited. As river ice images are often collected in remote locations by unmanned aerial vehicles with limited computation power, we explore the performance-latency trade-offs for river ice segmentation. We propose a novel convolution block inspired by both depthwise separable convolutions and local binary convolutions giving additional efficiency, parameter savings, and generalization ability to river ice segmentation networks. Our novel convolution block is used in a shallow architecture that has 99.9% fewer trainable parameters, 99% fewer multiply-add operations, and 69.8% less memory usage than a UNet, while achieving virtually the same segmentation performance. We find that this network trains fast and is able to achieve high segmentation performance early in training due to an emphasis on both pixel intensity and texture. When compared to very efficient segmentation networks such as LR-ASPP with a MobileNetV3 backbone, we achieve good performance (mIoU of 64) 91% faster during training on a CPU and and an overall mIoU that is 7.7% higher. We also find that our novel convolution block is able to generalize better to new domains such as snowy environments or datasets with varying illumination. Diving deeper into river ice segmentation with resource constraints, we take on a separate task of training a segmentation model when labelling time is limited. As the ice type, environment, and image quality can vary drastically between rivers of interest, training new segmentation models for new environments can be infeasible due to the laborious task of pixel-wise annotation. We explore a point labelling method leveraging object proposals and a post processing technique that delivers a 14.6% increase in mIoU as compared to a fully supervised UNet with the same labelling budget. Our point labelling method also achieves a mIoU that is only 6.3% lower than a fully supervised model with a annotation budget that is 23× larger.

**Acknowledgements**

First and foremost, I would like to thank my supervisor, Professor Andrea Scott, for her guidance, patience and encouragement throughout my masters. I would also like to thank my thesis readers Professor Linlin Xu and Professor Zhao Pan for reviewing this thesis and providing valuable feedback.

I would like to extend my appreciation to the members of the VIP lab, especially my close lab mates Anmol Nagi, Peter Lee, and Keerthijan Radhakrishnan, for their advice, instruction, and engaging chats during every stage of my masters. A special thanks to my roommates and fellow masters students Aidan Keaveny, Andrew Downie, and Matthew Cann for many discussions both technical and entertaining during the COVID-19 pandemic.

I would like to thank Abhineet Singh for giving public access to the Alberta River Ice Segmentation Dataset that was used repeatedly in this thesis. Finally, I would like to acknowledge the financial support from the University of Waterloo and the Natural Sciences and Engineering Research Council of Canada.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

**ANN** Artificial Neural Network 8

**CNN** Convolutional Neural Network 8–12, 20–22

**CPU** Central Processing Unit 2, 43, 44, 46, 60, 79

**CRF** Conditional Random Field 19, 26, 68, 71–73

**DSC** Depthwise Separable Convolution 13–16, 23, 40, 41, 43, 49, 51, 52, 54

**E-Net** Embedding Network 30, 31, 67

**EM** Expectation-Maximization 24, 26

**FCN** Fully Convolutional Network 20, 25, 27, 30, 67

**GPU** Graphics Processing Unit 1, 2, 5, 44, 46, 60, 79

**L-Net** Localization Network 30, 31, 67

**LBC** Local Binary Convolution 15–17, 40, 41, 43, 49, 51, 52

**LBP** Local Binary Pattern 15, 16, 49

**LR-ASPP** Lite Reduced Atrous Spatial Pyramid Pooling 23, 43, 49, 57, 58, 60, 97

**ReLU** Rectified Linear Unit 10, 16

**SVM** Support Vector Machine 18, 19, 21, 22

**UAV** Unmanned Aerial Vehicle 1, 2, 5, 22, 35, 44, 78, 79

**WISE** Weakly-supervised Instance SEgmentation 29, 30, 65–67

# Chapter 1

# Introduction

River ice in the northern hemisphere can have a lasting impact on both the local ecosystems and communities in the area. In terms of the ecosystem, river ice events can cause flooding, adversely impact fish habitats and spawning grounds, and release toxins by disrupting sediments [33]. In terms of direct human impact, remote northern communities often rely on river ice bridges as a primary means of transportation. Additionally, river ice events can impact water supply management, hydroelectric power generation, and cause damage to infrastructure [33, 6, 20]. Anchor ice, a class of river ice that is very sediment rich, is a key factor in sediment transport and release events [47]. Anchor ice therefore affects erosion and the overall geomorphology of the fluvial system [75]. Due to climate change, the frequency and intensity of river ice events are ever evolving [94]. This adds another layer of demand for efficient and accurate monitoring systems that are easy to implement in the field.

River ice imagery from a drone or Unmanned Aerial Vehicle (UAV) can give an observer a detailed understanding of the state and distribution of river ice at a given moment in time [47, 4, 87, 104, 105]. The segmentation of river ice imagery into one or more classes can be an automated means for a computer to process a large number of images and produce valuable information regarding ice cover density, ice cover distribution, drift ice speed, ice class distribution, and more [104]. Deep learning methods have proven to perform extremely well for segmenting general image datasets [14] as well as river ice specific datasets [87, 105]. There are two major drawbacks however to segmentation using deep learning.

The first drawback to deep learning for segmentation is that as deep learning models perform better by getting deeper and more complex, they also become inefficient and require specialized hardware such as a Graphics Processing Unit (GPU) [37]. When it

pertains to river ice, due to the limited hardware aboard many UAVs used to capture river ice imagery, deep learning algorithms are difficult to implement in real time [100]. If real time inference is not the goal of an operator, working offline on a local machine can also present problems if there is not a GPU present. As a result, an efficient network architecture for river ice segmentation can be valuable to operators in remote regions with limited computation power.

The second drawback to deep learning for segmentation is that it is very expensive to collect training data. For a segmentation task, it is required that an annotator assign a class label to each pixel for all the training images [5]. For a published river ice dataset, the Alberta River Ice Segmentation Dataset [86], we estimate that the pixel-wise annotation of the 50 labelled images in the dataset took a single human over 26 hours to annotate (Section 6.2.1). In order for it to be practical to train deep learning models on varying river ice environments in different locations, the labelling obstacle associated with pixel-wise annotations must be overcome.

## 1.1 Contributions

In this thesis we aim to address both the computational efficiency and annotation limitations of applying deep learning to river ice segmentation. The two primary contributions of this thesis are summarized as follows:

1. In order to move toward real-time training and inference of river ice imagery in the field, we explore the trade off between performance and latency in various segmentation models on the Alberta River Ice Segmentation Dataset. One of such models includes a novel DSC LBC block that can replace convolution operations in common neural network architectures. We show that DSC LBC blocks used with a shallow UNet style architecture improves efficiency as well as a network's ability to generalize to various environmental changes including illumination variation and snow noise. We show that when compared to models such as UNet, our architecture achieves very similar performance with significant improvements in efficiency. When compared to state-of-the-art networks optimized for efficiency on mobile devices, our network shows a significant improvement in performance as well as a slight improvement in training and inference efficiency on a GPU. On a Central Processing Unit (CPU), our network proves to be more efficient during training, however during inference the mobile optimized networks are slightly more efficient.

2. As it can be extremely time intensive to collect ground truth pixel-wise masks for datasets like the Alberta River Ice Segmentation Dataset, we experiment with weakly

supervised methods where ground truth labels are much easier to collect. We leverage the shape of the ice pans in the Alberta River Ice Segmentation Dataset and show that with only point labels at each ice pan and some custom post processing, we can approach fully supervised performance metrics with only a small fraction of the labelling effort. We also show that fully supervised methods under the same time constraints required to collect point labels greatly under-perform our weakly supervised method.

## 1.2 Outline

The remainder of thesis is organized as follows:

- Chapter 2 provides background relating to river ice, convolutional neural networks, efficient convolutions, the task of image segmentation, and weakly supervised segmentation methods.
- Chapter 3 gives details regarding where the data was collected and how it was arranged for the experiments in this thesis.
- Chapter 4 gives an explanation of the various metrics used to evaluate the various methods in this thesis.
- Chapter 5 details the methodology, results, and conclusions of a novel DSC LBC convolution block developed to build efficient networks for river ice segmentation.
- Chapter 6 details the methodology, results, and conclusions of a weakly supervised segmentation workflow developed to segment river ice using only point labels.

# Chapter 2

# Background

## 2.1 River Ice

In a general sense, an understanding of river ice concentration can be important for a variety of goals. The temporal and spatial ice distributions can help to validate river process and ice formation models [87]. This understanding can also be extended to water supply management, hydroelectric power generation, and predicting ice jam risks [33, 27, 6]. Ice jam risks are comparable to dam break events, causing a sudden release of water previously confined by the river ice [40]. Ice jam events on the Athabasca River have resulted in multiple floods causing millions of dollars of damages to the Town of Fort McMurray [20], one of such events resulted in waves measured at over 4 m in height [40]. Analyzing river ice at a closer scale, various types of river ice exist, each having unique characteristics and distinct effects on their environment. Discriminating between two common classes of ice, surface forming frazil ice and sediment rich anchor ice, can be important in understanding sediment transport and sediment release events [47].

### 2.1.1 Frazil and Anchor Ice

When a river is exposed to sustained periods of freezing air temperatures, ice cover can begin to form on the surface of the river. A drone image of river ice taken on the Peace River can be seen in Figure 2.1. The first ice type of interest seen in Figure 2.1, frazil ice, forms as ice crystals that flocculate together, float to the surface forming surface slush, and and consolidate to form frazil ice pans. In the late stages of freeze up, frazil ice pans can freeze together to form frazil ice rafts. Frazil ice pans often appear circular and can have

upturned edges caused by collisions with other ice pans. The second ice type of interest seen in Figure 2.1, anchor ice, forms on river beds, resulting in a high sediment content within the ice. Through thermal or mechanical means, the anchor ice can be released from the river bed and float to the surface [93, 50]. On the surface, anchor ice usually appears darker than frazil ice due to the high sediment content, and may have fewer upturned edges due to less time on the surface and different structural properties. When anchor ice release events occur, frazil ice generation may have already stopped and can result in a much higher concentration of anchor ice than frazil ice [47].

It was observed on the Peace River in Alberta Canada that anchor ice can transport significant quantities of sand, gravel, and cobble sized particles as seen in Figure 2.2 [48]. This is an important factor in calculating an annual sediment budget which is necessary to understand the amounts of erosion, accretion, and geomorphology in a fluvial system [75].

### 2.1.2 River Ice Data Collection

One means of collecting imagery of river ice is through the use of an UAV [47, 4, 87, 104, 105]. Due to the limited hardware aboard many UAVs, data intensive processes such as deep learning algorithms, which require large compute and memory resources, are difficult to implement in real time. When operating on UAV data in real time, the information collected on the UAV can be sent to a remote server; however, a wireless link with high bandwidth, minimal latency and an ultra-reliable connection is then necessary [96], making this difficult in remote regions. Alternatively, data can be processed on the UAV itself, although depending on the model of the UAV this would involve a low/middle grade GPU [100] or no GPU at all. This computation obstacle is also present in other domains such as forest fire detection [45], crop phenology [99], and sea ice segmentation [61] where computation limitations hinder operational sea ice prediction [98].

### 2.1.3 Available Data

The Alberta River Ice Segmentation Dataset [86] is a publicly available dataset containing labelled and unlabelled images of river ice taken from a UAV on the Peace River and North Saskatchewan River in Alberta, Canada. These rivers were selected by the creators of the dataset as their large nature leads to high winter discharge and images that can better facilitate the estimation of sediment transport [49]. An example of one of the unlabelled images in the Alberta River Ice Segmentation Dataset can be seen in Figure 2.1. The labelled images in the dataset have accompanying labels that indicate the class of each

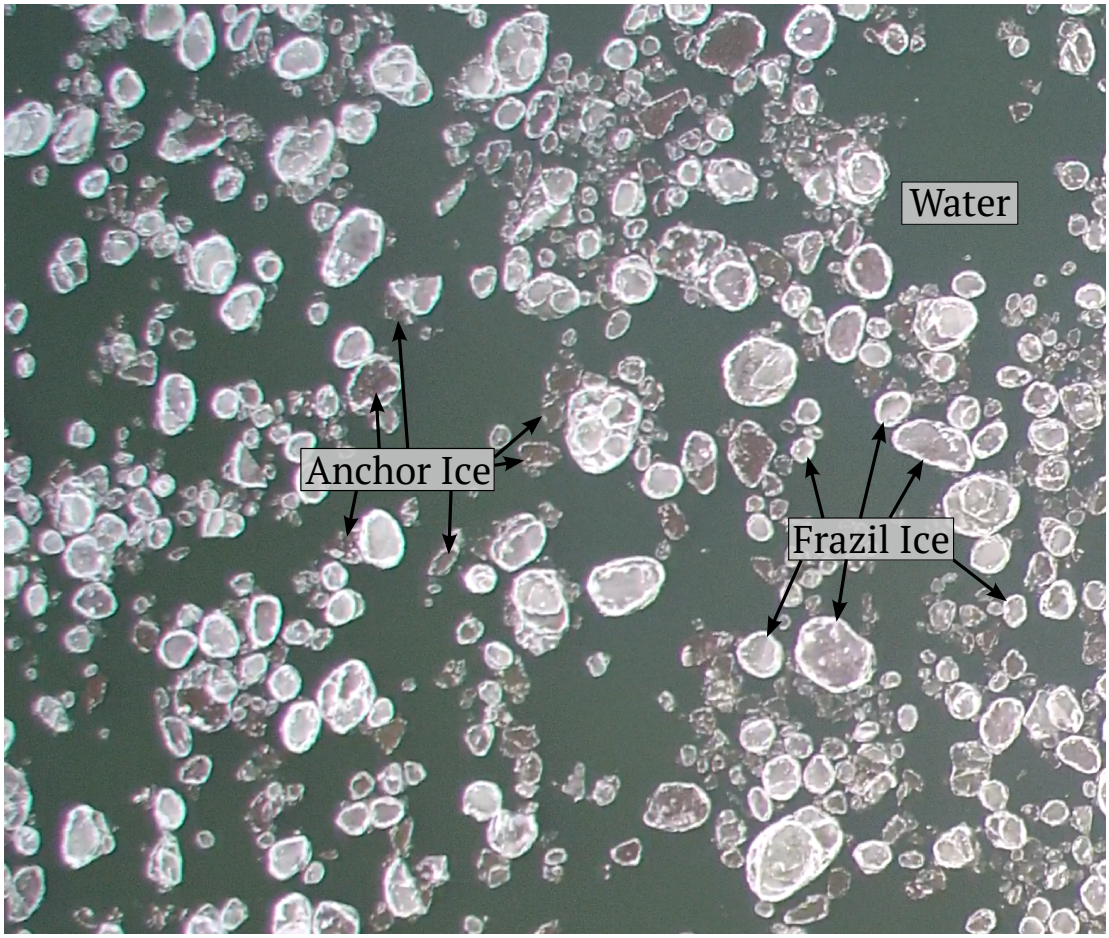



Figure 2.1: UAV image of river ice from the Alberta River Ice Segmentation Dataset [86]. Brighter frazil ice and darker sediment rich anchor ice are identified with arrows. Upturned edges appear as bright outlines to ice pans and are thin or non-existent on anchor ice pans.

Figure 2.2: Image taken on the Peace River of an anchor ice pan rafting large cobbles and fine sediment. This photograph was taken on Dec. 19, 2014 by Kalke *et al.* [48]. Proof of permission for this figure is shown in Appendix 7.2.

pixel. Pixel classes are either water, frazil ice, or anchor ice. The pixel-wise nature of the labels lend themselves well to the task of estimating river ice concentration as well as discriminating between frazil and anchor ice for an improved understanding of sediment transport.

Various environmental factors affect the quality of the images in the dataset. The labelled images were captured during clear conditions while it was not snowing; however, the dataset provides unlabelled videos captured while it was snowing, showing a realistic scenario where natural noise could contaminate the data. Within both the labelled and unlabelled images, the illumination is variable, likely affected by the time of day and the amount of sunlight at the time of acquisition.

## 2.2 Convolutional Neural Networks

An Artificial Neural Network (ANN) is a weighted directed graph where each node is some non-linear function of a weighted sum of the inputs. The architecture of an ANN is inspired by the network of biological neurons, axons, and synapses in the human brain [43]. An ANN learns by adjusting the weights at each node in an effort to minimize the observed error according to a cost function that compares the network predictions to the known ground truth. The process of adjusting the weights is known as backpropagation, which calculates the gradient of the cost function with respect to the current weights using the chain rule and adjusts the weights according to the calculated gradients and a learning rate [29]. ANNs and their more advanced variants have produced state of the art results in various fields including natural language processing [101], computer vision [97], and robotic control [89].

A Convolutional Neural Network (CNN) is a type of ANN that makes use of of convolutional layers. Convolutional layers only consider data points within a given receptive field, often a square window of an image or feature map. This is in contrast to early ANNs, which used all data points from the previous layer in calculating the next layer. CNNs are often well suited for data with a grid-like topology; including images as they are a 2D grid of pixels [29]. In the field of computer vision, CNNs have achieved state of the art results in various tasks including image classification [73], image segmentation [26], object detection [106].

### 2.2.1 Convolutional Layer

Convolution, denoted by an asterisk, is an operation on two functions, $f$ and $g$,

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau, \tag{2.1}$$

where $t$ is a real valued argument and $\tau$ is a dummy variable of integration [9]. In the discrete case this can be represented as follows,

$$(f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t-\tau). \tag{2.2}$$

In the case of an image, $f$ can be considered the image, and $g$ can be considered a 2D kernel, typically orders of magnitude smaller than the image. The convolution operation can therefore be represented as,

$$(f * g)(i,j) = \sum_{m}\sum_{n} f(m,n)g(i-n, j-n), \tag{2.3}$$

where $i$ and $j$ are pixel coordinates. A similar function known as cross-correlation is often used for machine learning applications as the cumulative property of convolution, or the "flipping" of the kernel, does not affect implementation results in a CNN [29]. As a result, machine learning libraries require slightly less code to implement cross-correlation and therefore opt to build CNNs with cross-correlation as opposed to convolution. Cross-correlation can be expressed as follows,

$$(g * f)(i,j) = \sum_{m}\sum_{n} f(i+m, j+n)g(m,n). \tag{2.4}$$

In a convolutional layer, the kernel contains the learnable weights that are tuned during backpropagation. The sparse interactions of the small kernel as well as parameter sharing in convolutional layers results in the ability to build deeper networks as there are lower memory requirements as compared to a fully connected network.

In practice, various hyperparameters exist around a convolution including the kernel size, stride, padding and dilation [67]. Kernel size is simply the 2D dimensions of the kernel that is convolved with an image or feature map. Stride determines the distance, in pixels, that the kernel shifts from one convolution operation to another as it moves across the image or feature map. If stride is one, then the kernel passes over the image or feature map one pixel at a time. If the stride is increased, the output feature map will often have a smaller size, though this is also determined by padding. Padding is an addition of pixels,

often of value zero, to the edges of an image or feature map prior to a convolution operation occurring. Padding can preserve the spatial dimensions of feature maps after convolution. For example, a convolution with an input of size $h \times w$, a kernel size of $3 \times 3$, and a stride of one will output a feature map of size $h - 2 \times w - 2$ since the $3 \times 3$ kernel never has its centre at the edge pixels of the input. However, if in the same example padding is applied in the form of a a single-pixel border to the input, the output feature map would be of size $h \times w$ since the centre of the $3 \times 3$ kernel was able to lie at edge of the original input. Finally, dilation allows for control over the receptive field of a convolution and is described more in Section 2.4.2.

### 2.2.2 Activation Function

Inspired by the binary impulses carried by the axons of a biological neuron, an activation function decides how information is transferred from one node to the next. Activation functions are non-linear functions that allow CNNs perform tasks with a degree of non-linearity that would otherwise not be captured by the linear nature of the convolution function. An activation function $\phi$ can be applied element-wise to the output of a convolution operation. A very effective and therefore ubiquitous activation function is the Rectified Linear Unit (ReLU) [62, 44] described by

$$\phi(z) = max(0, z), \tag{2.5}$$

where $z$ is an output element of a convolution operation. Another well known activation is the sigmoid function [30]

$$\phi(z) = \frac{1}{1 + e^{-z}}. \tag{2.6}$$

An advantage of ReLU over the sigmoid function is related to the gradient at high values of $z$. The gradient of ReLU is constant when $z > 0$ (Figure 2.3 (a)) while the gradient of the sigmoid function is very small at high values of $z$ (Figure 2.3 (b)). This can result in faster learning for networks with the ReLU activation function [29]. Additionally, as ReLU assigns a value of zero to all inputs $z < 0$, sparsity is introduced into the network, which can reduce the computational resources and data necessary to train a network [83].

A common activation function in the last layer of a CNN is the softmax function

$$\phi(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{2.7}$$

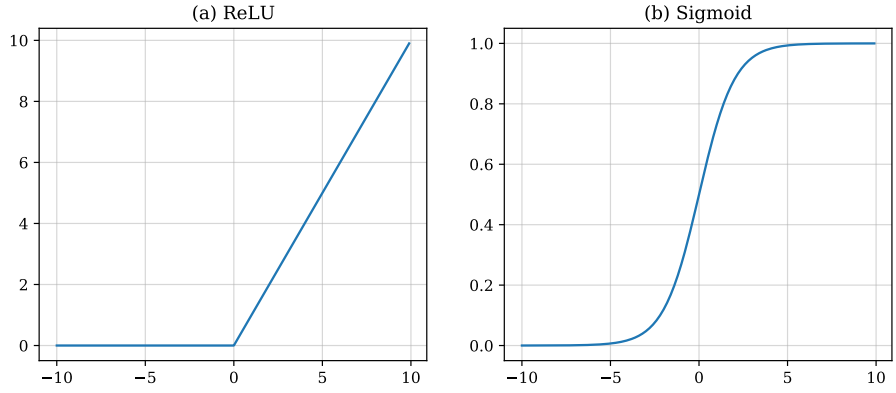



Figure 2.3: ReLU activation function shown in (a) and sigmoid activation shown in (b). When the input is large (e.g. 10), the the ReLU function can be seen to have a constant gradient, while the sigmoid function can be seen to be very flat as it has a very small gradient.

where $\vec{z}$ in an input vector, $K$ is the number of classes in a classification task as well as the length of $\vec{z}$, and $i$ is an index in $\vec{z}$. Equation 2.7 can be applied to each index in $\vec{z}$ in order to complete the softmax calculation on $\vec{z}$. The softmax function is useful at the end of a network as it creates an empirical probability distribution over the predicted output classes [29].

### 2.2.3 Down-sampling and Up-sampling

Many CNN architectures use an encoder architecture that acts to extract features from an image [54, 76, 32]. As an image travels through subsequent levels of an encoder, there is often an increase in the number of channels facilitated by the convolution operations, and often a reduction in spatial resolution facilitated by pooling operations. Pooling is applied to a feature map, which is the output of a kernel convolved with an image and passed through an activation. The process of pooling replaces a localized section of the feature map with a summary statistic such as the mean or maximum value [29]. In a well-designed encoder, the sizes of feature map tensors are decreased with subsequent convolution and pooling events, resulting in less memory requirements and an improved statistical and computational efficiency within the deeper layers of the encoder. Max pooling is a popular version of pooling in which a $k \times k$ window is simply replaced by the maximum value in

that window [107].

Depending on the task and architecture of the CNN, the encoded features may be resized to the original shape of the image. In these scenarios, non-learnable operations such as bilinear and bicubic interpolation can be used [71], or leanable operations such as transpose convolutions can be used where, similar to a convolution, a learnable kernel is employed [22].

### 2.2.4 Batch Normalization

When an input tensor is normalized and passed to a network, the distribution of the tensor changes slightly from layer to layer as the convolutions and activation functions are applied and can cause the tensor to no longer be normalized. This effect is known as covariate shift and can intensify as a tensor travels deeper and deeper in a network. Batch normalization aims to make training faster and more stable by mitigating this covariate shift, particularly in deep networks with many layers [41]. As the name implies, the batch normalization between layers occurs over a batch, or a subset of all the images used during an optimization step.

Let $B$ be a batch of size $m$ images, where $x_{1...m}$ are the images in $B$. The mean and variance of the batch are respectively calculated as follows,

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{2.8}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2. \tag{2.9}$$

For a layer with $d$-dimensional input $x = (x^{(1)}, ..., x^{(d)})$, each dimension, or channel, is normalized by,

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)2} + \epsilon}}, \tag{2.10}$$

where $\epsilon$ is arbitrarily small and added for numerical stability. To ensure the batch normalization transformation inserted in the network can represent the identity transform, a pair of learnable parameters $\gamma^{(k)}$ and $\beta^{(k)}$ are used to scale and shift the normalized value respectively [41],

$$y^k = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}. \tag{2.11}$$

Other benefits to batch normalization were discovered by the authors including regularization effects improving generalization ability as well as the mitigation of vanishing or exploding gradients with an increased learning rate. This vanishing or exploding gradient phenomena in deep networks occurs when the chain rule leads to small gradients decreasing exponentially or large gradients increasing exponentially [35]. The exact cause of these additional benefits are debated in the literature as some believe covariate shift is not reduced and rather the objective function is smoothed considerably by batch normalization [80].

## 2.3 Efficient Convolutions

### 2.3.1 Depthwise Separable Convolutions

A Depthwise Separable Convolution (DSC) is a form of a factorized convolution that results in significant increases in efficiency while minimizing performance losses. As a result, they are key components of modern efficient networks where they replace standard convolution operations [37, 17, 103, 79, 36]. While a standard convolution operation filters and combines inputs in one step, DSCs split standard convolution operations into two parts, a depthwise convolution for filtering and a pointwise convolution for combining across depth.

During a standard convolution operation, an input layer can take an input feature map $\mathbf{F}$ of size $D_F \times D_F \times M$ and outputs an output feature map $\mathbf{G}$ of size $D_F \times D_F \times N$, where $D_F$ are the height and width of a square feature map, and $M$ and $N$ are the number of input channels and output channels in the convolution. This convolution has a kernel $\mathbf{K}$ of size $D_K \times D_K \times M \times N$ where $D_K$ is height and width of a square kernel. The output feature map $\mathbf{G}$ can therefore be calculated as follows,

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \cdot \mathbf{F}_{k+i-1,l+j-1,m}, \tag{2.12}$$

where $i, j$ are kernel indices and $k, l$ are feature map indices. Equation 2.12 has an associated computational cost of

$$Cost_{standard} = D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F [37]. \tag{2.13}$$

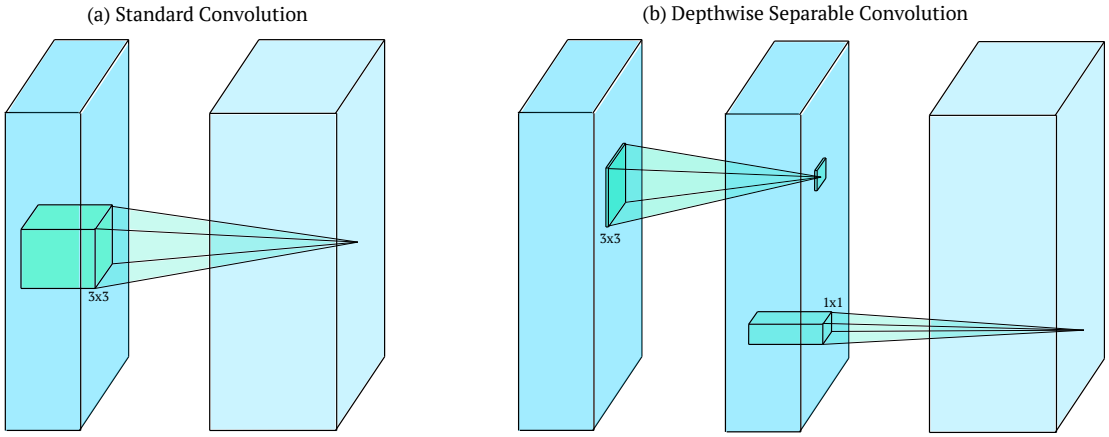



Figure 2.4: Visualization of a standard convolution in (a) where filtering and combining happen in one step. Visualization of a DSC in (b) showing the depthwise filtering step followed by the pointwise combining step. Figure took inspiration from Howard *et al.* [37].

As mentioned, DSCs factorize a standard convolution into two steps, a depthwise convolution and a pointwise convolution. The depthwise convolution occurs first and is similar to a standard convolution; however, it only applies a single $D_K \times D_K$ filter per channel of the input image or feature map. This is followed by the pointwise convolution, which is a 1×1 convolution across depth, allowing for a change in the number of channels of the output feature map if desired.

As the depthwise step only has one filter per image channel, the depthwise kernel $\hat{\mathbf{K}}$ is only of size $D_K \times D_K \times M$, where the $m_{th}$ filter in $\hat{\mathbf{K}}$ is applied to the $m_{th}$ channel in $\mathbf{F}$ to produce the $m_{th}$ channel of the depthwise outputted feature map $\hat{\mathbf{G}}$. This can be written as,

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \cdot \mathbf{F}_{k+i-1,l+j-1,m}, \tag{2.14}$$

with a computational cost of

$$Cost_{depthwise} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F[37]. \tag{2.15}$$

The pointwise step in a DSC is a 1×1 convolution over the channels of $\hat{\mathbf{G}}$ inorder to create new features. This step has a computational cost of

$$Cost_{pointwise} = M \cdot N \cdot D_F \cdot D_F, \tag{2.16}$$

giving a DSC operation a computational cost of

$$Cost_{dsc} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F, \tag{2.17}$$

which is the sum of both the depthwise and pointwise computational cost.

Comparing Equations 2.13 and 2.17, DSCs affect the computational cost of a standard convolution by a factor of

$$CostFactor = \frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}. \tag{2.18}$$

This factor can be approximated to $\frac{1}{D_K^2}$ when $N \gg D_K$. Therefore, when using a standard kernel size of $3 \times 3$, computational cost is reduced by 8 to 9 times by using DSCs. The trade-off in performance was found to be minimal in a study where a network was built with DSCs had an ImageNet accuracy of 70.6% and 569M Mult-Adds, while an identical network with standard convolutions had an ImageNet accuracy of 71.7% and 4866M Mult-Adds [37].

### 2.3.2 Local Binary Convolutions

A Local Binary Convolution (LBC) is another alternative to a standard convolution that significantly reduces the number of trainable parameters [46]. LBCs are inspired by the Local Binary Pattern (LBP); a texture pattern descriptor used to characterize local texture patterns in an image based on a neighbourhood of pixels [63]. Based on contrasting pixels in a neighbourhood, a string of bits is calculated and converted to a base 2 decimal number that is used as the feature to the central pixel. A LBP for a $3 \times 3$ window can be calculated as follows,

$$LBP(x_c, y_c) = \sum_{n=0}^{N-1} g(i_n - i_c)2^n, \tag{2.19}$$

$$g(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}, \tag{2.20}$$

where $i_c$ is the intensity of the centre pixel at location $x_c, y_c$, and $i_n$ is the the intensity of the $n^{th}$ neighbouring pixel out of $N$ total neighbours[1]. LBPs are illumination invariant as they focus on contrasting pixel intensities, which describe texture. As a result, they have become popular image descriptor for facial recognition tasks among others [2].

Similar to DSCs, LBCs also operate in two parts; first a spatial convolution, which we will refer to as $s_1$, and second a $1 \times 1$ channel-wise convolution which we will refer to as $s_2$. The first stage, $s_1$, can be looked at as equivalent to a standard convolution, with a few modifications.

- Similar to a standard convolution, the kernel weights of $s_1$, $\mathbf{K}_{s_1}$, are of size $D_K \times D_K \times M \times N$[2] where $D_K$ is height and width of a square kernel, and $M$ and $N$ are the number of input channels and output channels in the convolution. However, $s_1$ differs from a standard convolution in that $\mathbf{K}_{s_1}$ are randomly initialized such that $\mathbf{K}_{s_1} \in \{-1, 0, 1\}$. More specifically, to determine the arrangement of -1, 0, and 1 in $\mathbf{K}_{s_1}$, a sparsity proportion, $sp$, is set which determines the amount of non-zero values in $\mathbf{K}_{s_1}$. This involves setting $(100 - sp)\%$ of $\mathbf{K}_{s_1}$ to zero according to a random uniform distribution[3]. Then, the remaining non-zero values are randomly assigned to $-1$ or 1 with equal probability using a Bernoulli distribution.
- Second, the kernel weights $\mathbf{K}_{s_1}$ are set to be non-trainable and therefore cannot be updated during training.

Next, the output of $s_1$ is then passed through an activation function (sigmoid or ReLU), and used as an input to the second stage, $s_2$. Similar to the pointwise convolution in a DSC, $s_2$ consists of a $1 \times 1$ convolution that acts only on the channels of the feature maps and contains the only trainable parameters in LBCs.

If an input feature map has $M$ channels, an output feature map has $N$ channels, and a kernel size of $D_K$ is used, a standard convolution operation has the following number of trainable parameters,

[1]For example, a $3 \times 3$ window would have eight neighbours to the centre pixel. The top, top right, right, bottom right, bottom, bottom left, left, and top left pixels pixels.

[2]Recall that this is contrast to the depthwise stage of a DSC where weights are of size $D_K \times D_K \times M \times 1$, or a single unique $D_K \times D_K$ filter is applied to each channel of the input. The first stage in a LBC applies multiple filters to each channel of the input similar to a standard convolution.

[3]Note that we follow the same convention as the LBCNN authors where the sparsity percentage refers to the percentage of non-zero elements. For example, a sparsity value of 100% corresponds to a dense weight tensor with no zeros [46].

$$TrainParams_{standard} = M \cdot N \cdot D_K \cdot D_K. \tag{2.21}$$

Comparatively, a LBC operation only has

$$TrainParams_{lbc} = M \cdot N \tag{2.22}$$

trainable parameters since $D_K = 1$ in the $1 \times 1$ convolution[4].

Combining Equations 2.21 and 2.22, LBCs affect the trainable parameters by a factor of

$$TrainParamsFactor = \frac{M \cdot N}{M \cdot N \cdot D_K \cdot D_K} = \frac{1}{D_K^2}. \tag{2.23}$$

Therefore, given two equivalent networks architectures, one with standard convolutions and the other with LBCs, LBCs save at least $9\times$, $25\times$, or $49\times$ the number of trainable parameters if kernel sized of $3 \times 3$, $5 \times 5$, or $7 \times 7$ were used respectively [46].

## 2.4 Segmentation

Image segmentation is the process of splitting an image into multiple subgroups, where the pixels within a subgroup often have similarities such as belonging to the same type of object or scene. Image segmentation can be accomplished through classical image processing, machine learning, and deep learning algorithms. The task of image segmentation can be grouped into two categories, semantic segmentation and instance segmentation. Semantic segmentation is simply concerned with the class label of each pixel in the image, while instance segmentation is concerned with both the class label of each pixel, as well as the distinct instances of a given class [51]. For example, consider a binary classification problem with images of multiple river ice pans in open water where it is desired that ice class be separated from the water class. A semantic segmentation algorithm would aim to categorize pixels in the water as the water class, and the pixels in the multiple different ice pans the same; simply as the ice class. An instance segmentation algorithm will similarly aim to categorize the pixels in water as the water class, however it differs from the semantic segmentation algorithm when it comes to categorizing the ice. An instance segmentation algorithm would aim to categorize the pixels within one ice pan the same, but would

[4]Note that this calculation assumes that $s_1$ outputs the same number of channels its the input.

categorize pixels of another ice pan as distinct from the other ice pans since there are multiple instances of the ice pans. With respect to river ice, instance segmentation would be valuable if one wished to also know the number of ice pans. This can help estimate floe size distribution, a component in estimating ice jam severity [11].

### 2.4.1 Classical Methods

#### Thresholding

Prior to the widespread adoption of deep learning for image segmentation tasks, many classical image processing and machine learning methods were used for this purpose. Thresholding, arguably the most straight forward method, separates objects simply based on their grayscale or colour pixel values. Depending on the number of classes of interest, one or more thresholds are selected. Then, pixels are assigned to a given class depending on if their value is greater or less than a given threshold. If the image histograms, or pixel value distributions, vary significantly between images or images change over time, this kind of thresholding cannot generalize well [77]. As a result, various adaptive global [64] and local [3] thresholding techniques have been developed that can adapt to changing histograms.

#### Clustering

Clustering is an unsupervised method often used in image segmentation in which pixels are grouped with other similar pixels. K-means is a popular clustering algorithm that aims to minimize the sum of squared distances between all points and the cluster centres [92]. First, $K$ initial cluster centres are chosen. Second, all samples are assigned to a cluster based on their euclidean distance to the cluster centres. Third, the means of each cluster is calculated and set to be the new cluster centre. Step two and three are repeated until the clusters do not change or a maximum number of iterations is met. Although this algorithm is simple, various work has been done to address some shortfalls including the dependence on good initial cluster centres and the chosen value of $K$ [21, 74].

#### Support Vector Machines

A Support Vector Machine (SVM) is a supervised learning algorithm than can be used for classification. A SVM discriminates between classes by finding one or more hyperplanes that have the maximum margin between the classes [18] This margin is the distance between the hyperplane that splits classes, and the nearest data point. An advantage SVMs have

over traditional linear separators is that they can project the training data into a higher dimensional space, using a kernel function, in an effort to to find a separating hyperplane. If a non-linear kernel is used, these higher dimensional hyperplanes are non-linear in the original space, expanding the prediction capabilities of SVMs [78]. When using an SVM for image segmentation, hand crafted features such as summary statistics can be used as training data. One aid in hand crafting features can be the use of superpixels, which are small pixels grouped together based on a meaningful similarity measure. One such algorithm for creating superpixels is the Simple Linear Iterative Clustering (SLIC) algorithm known for it's boundary adherence, memory usage, and computation speed [1]. Superpixels can help for crafting features as redundancy is captured within a single superpixel, reducing computational complexity as millions of pixels can be replaced by thousands of pixels [47].

### Conditional Random Fields

The Conditional Random Field (CRF) has been broadly used in semantic segmentation tasks to combine the class scores of segmentation models with the low-level information captured by the local interactions of pixels and edges [13]. A common implementation of a CRF is for the post processing of segmentation results to refine the edges of objects. The fully connected pairwise CRF is commonly employed for post processing due to its efficiency, ability to capture fine edge details, and maintain long range connections [52]. The fully connected pairwise CRF connects all pairs of individual pixels in an image through pairwise potentials defined by a linear combination of Gaussian kernels. The following equation defines a Gaussian kernel,

$$k(\mathbf{f}_i, \mathbf{f}_j) = w^{(1)} \underbrace{\exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right)}_{\text{appearance kernel}} + w^{(2)} \underbrace{\exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right)}_{\text{smoothness kernel}}, \tag{2.24}$$

where the vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ are feature vectors for pixels $i$ and $j$ in an arbitrary feature space, $w^{(1)}$ and $w^{(2)}$ are weights, $I_i$ and $I_j$ are colour vectors, $p_i$ and $p_j$ are positions, and $\theta_\alpha$, $\theta_\beta$, and $\theta_\gamma$ are parameters that control degrees of nearness and similarity. The *appearance kernel* leverages the idea that nearby pixels with a similar color are likely to be in the same object, while the *smoothness kernel* targets small isolated regions that have significant contrast to their surroundings [52]. The authors of the fully connected pairwise CRF found that searching for parameters $w^{(1)}$, $\theta_\alpha$, and $\theta_\beta$ to be most effective, while $w^{(2)}$ and $\theta_\gamma$ have little impact on the results and can be set to one in most cases. For more information regarding CRFs, refer to the paper by Krähenbühl *et al.* [52].

### 2.4.2 CNN Architectures

Since the introduction of AlexNet [54] for image classification, CNNs have been ubiquitous in the computer vision field and have indeed been employed for the task of image segmentation.

#### Fully Convolutional Networks

A major step forward in image segmentation using CNNs was the introduction of Fully Convolutional Network (FCN) [60]. The original FCNs leveraged the ability of CNNs to learn hierarchical features, and replaced fully connected layers with convolutional layers [26]. These networks therefore output a spatial map, rather than a classification score, which is up-sampled to a dense per-pixel label. The introduction of FCNs represented a significant milestone as it allowed for a CNN to be trained end-to-end for segmentation and allowed for arbitrary sized inputs as there are no constraining fully connected layers [26].

#### UNet

The authors of UNet built upon the FCN for the task of medical image segmentation [76]. UNet uses an encoder-decoder structure in which the encoder creates low-resolution feature maps, while the decoder converts the low resolution feature maps into a dense pixel-wise predictions. The idea behind UNet is that the decoder uses the feature maps calculated by the encoder through the use of skip connections, which concatenate the feature maps from the encoder with the output of the decoder at the equivalent level. These skip connections preserve the structural integrity of the image and reduce distortion. UNet also uses a large number of feature channels in the decoder, allowing for contextual information to propagate to higher resolution layers and resulting in a symmetric, u-shaped architecture [76].

#### DeepLab

The DeepLab series of CNNs also built upon the work on the FCN by enlisting dilated convolutions and atrous spatial pyramid pooling (ASPP) for the task of segmentation [13, 14, 15, 16]. A dilated convolution, also know as an atrous or à trous convolution, is a way of expanding the receptive field of a convolution. A dilated convolution has a dilation rate $r$ that defines the interval for which an input is sampled [14]. In a standard

convolution using a $3 \times 3$ kernel, $r = 1$ since all points in the kernel are directly adjacent to one another. However, if a dilation rate of $r = 2$ is used with a $3 \times 3$ kernel, the kernel would essentially cover a $5 \times 5$ area with zeros between each kernel value both vertically and horizontally. The dilation does not increase the computational complexity as there are still only $3 \times 3 = 9$ points being considered; however, the dilation allows for the convolution to have a larger spatial extent. The dilated convolutions also allow for deep networks to find rich features without continuous down-sampling reducing the resolution of the feature maps [15]. Atrous Spatial Pyramid Pooling (ASPP) combines the idea of dilated convolutions with the established spatial pyramid pooling (SPP) [14]. SPP was introduced for classification based CNNs in which, prior to the fully connected layers at the end of the network, multiple pooling layers occur in parallel and are concatenated before they are given to the fully connected layer [31]. Rather than having multiple pooling layers in parallel, ASPP used multiple dilated convolutions in parallel, each with a different dilation rate. This is done to help account for the segmentation of objects of varying scale [15].

### 2.4.3 Segmentation for River Ice

#### Classical Approaches

Classical image processing, often referring to methods that do not involve machine learning or deep learning, has been employed in previous studies involving the segmentation of river ice. Texture segmentation of infrared river ice images was performed using a Gabor filter and K-means clustering technique [8]. A method for automatically estimating surface ice concentration in the Lower Nelson River was developed using a suite of image processing techniques including median filtering, Canny edge detection, Hough transforms, and thresholding [3]. River ice concentration estimates and the delineation of frazil ice from anchor ice in the Peace River and North Saskatchewan River was attempted using a SVM trained using superpixel features [47]. Similarly, superpixels and Iterative Edge Refinement were used to train an SVM to segment image scenes of the Dauphin River into six different classes [4].

#### Deep Learning Approaches

As mentioned in Section 2.4.2, CNNs have been ubiquitous in the computer vision field and have indeed been employed for the task of river ice segmentation. Various CNNs were used to segment frazil ice, anchor ice, and water in the North Saskatchewan River

with results that improved upon an SVM [87]. In the Yellow River region, ICENET was used to segment river ice images taken by a UAV by fusing positional and channel-wise attentive features within a CNN [104]. Following this work, ICENETv2 was introduced which segments shore ice, drift ice, and water in the Yellow River region by modifying ICENET to more effectively fuse multi-scale features [105]. These deep learning based approaches show promising results with respect to performance; however, none of them address efficiency trade-offs that exist in the various architectures tested.

### 2.4.4 Previous Work on Alberta River Ice Segmentation Dataset

As mentioned in Section 2.1.3, the Alberta River Ice Segmentation Dataset contains UAV images of frazil and anchor ice from the Peace River and North Saskatchewan River in Alberta, Canada [86]. The first work that attempted to discriminate frazil and anchor ice used superpixel features with an SVM [47]. The authors first smoothed the images with a Gaussian filter to aid the subsequent creation of 1000 superpixels per image using MATLABs built in functionality. Next, they converted the images from RGB channels to HSV channels. To train the SVM, the authors calculated 33 summary statistics of the pixels contained in a superpixel, as well as 48 statistical features of pixels in a 100×100 pixel grid around the centre of the super pixel; for a total of 81 features. Specifically, the 33 superpixel features consisted of the mean, standard deviation, max, min, median, root-mean-square, skewness, kurtosis, variance, and normalized x, y spatial values, all for three HSV channels. The 48 features for the 100×100 window were created by calculating the max, min, mean, and standard deviation for the four 50×50 quadrants of the 100×100 window for each of the HSV channels. The authors found that the SVM struggled in discriminating anchor and frazil ice compared to other experiments where a SVM with the same features was used to simple discriminate ice from water [47].

The second body of work to discriminate frazil and anchor ice in the Peace River and North Saskatchewan River aimed to improve upon the previously mentioned SVM by testing a suite of deep learning semantic segmentation models [87]. The authors tested four deep learning architectures against the SVM; DeepLabV3+, UNet, SegNet, and a DenseNet architecture modified for segmentation. The authors found that all four models improved upon the results of the SVM when classifying water and anchor ice; however, when classifying frazil ice, SegNet and DenseNet under-performed relative to the SVM. UNet and DeepLab, however, outperformed relative to the SVM for all classes and for all metrics tested (pixel accuracy, mean pixel accuracy, mean intersection-over-union, and frequency weighted intersection-over-union). Between UNet and DeepLab, DeepLab quantitatively outperformed UNet on an external test set of 18 labelled images. However, when evaluating

the four models on various unlabelled images, the authors found that DenseNet performed the best visually, doing especially well when frazil and anchor ice existed on the same ice pan. They also found that DeepLab performed the worst on the unlabelled images, often misclassifying regions along the edges of image patches. As a result the authors concluded that UNet was the ideal model of the four, performing well both quantitatively and qualitatively [87].

### 2.4.5 Efficient Networks and Segmentation

Since the introduction of DSCs for efficient networks, various architectures have been developed, including the suite of MobileNets (V1, V2, V3) [37, 79, 36], MnasNet [90], and others. The latest of the aforementioned networks, MobileNetV3, borrows various aspects from other networks as well as introduces advancements of its own. MobileNetV3 uses DSCs from MobileNetV1 [37], as well as a linear bottleneck and inverted residual structure from MobileNetV2 [79] and lightweight attention modules based on squeeze and excitation from MnasNet [90], both of which reduce the number of operations and memory requirements. Novel contributions of MobileNetV3 include a hardware-aware network architecture search that optimizes the number of filters in a layer, the h-swish nonlinearity which replaces ReLU; improving accuracy while reducing computational cost on embedded devices, and the rearrangement of computationally-expensive layers at the beginning and end of the network [36].

For the task of semantic segmentation, MobileNetV3 can be used as a backbone feature extractor in a segmentation network such as DeepLabV3 [15]. The authors of MobileNetV2 designed a reduced Atrous Spatial Pyramid Pooling module (ASPP) [14] that was found to outperform the standard DeepLabV3. MobileNetV3 designed a Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP) which made further improvements by deploying global-average pooling similar to the Squeeze-and-Excitation module [39].

Other methods exist for improving network efficiency such as quantization [42, 53], pruning [25], and knowledge distillation [34]; however, the scope of this thesis is focused on the network architecture. As the aforementioned techniques can be applied to any architecture, we choose to focus on getting the best performance-latency trade-off out of the architecture alone, allowing for these techniques to be brought in later on if so desired.

## 2.5 Weakly Supervised Segmentation

Although fully supervised methods have achieved state-of-the-art performance for segmentation tasks, the acquisition of pixel-level labels is time consuming, laborious, and expensive [102]. For example, the labellers of the MS COCO dataset took on average 10.1 minutes to label each image by pixel-level instances, while it only took them 4.1 seconds to label each image by category [12, 58]. Weakly supervised approaches offer a solution to expensive label acquisition by using less spatially-informative annotations that require much less time to collect. Various labelling techniques include bounding boxes, scribbles, points, and image-level labels.

### 2.5.1 Labelling Techniques

#### Image-Level Labelling

The cheapest of these labels to collect are image-level labels (Figure 2.6 (a)) which simply specify the presence or absence of a semantic class in a given image [65]. Though these labels require minimum effort to collect, they are the most challenging to use due to the lack of positional information [12]. For the task of semantic segmentation with only image-level labels, an online Expectation-Maximization (EM) method was developed [65], which outperformed previous attempts to use image-level labels. This method trains a DeepLab architecture where a pixel-level pseudo-ground truth is inferred from embeddings from an intermediate layer.

#### Point Labelling

A labelling technique that requires slightly more effort than image-level labeling, though is still very cheap to collect, is the use of point labels (Figure 2.6 (b)). A point label is simply an $x$-$y$ location in an image indicating the approximate centre of an object as well as the class it belongs to. One study using point labelling created a loss function which has a image-level term that encourages some predicted pixels to be of the same class as the image-level label, as well as a point-level term that acts as the cross entropy for only the labelled points [5]. This study also incorporates an objectness prior which helps with the spatial extent of the objects. This prior gives a probability that each pixel belongs to *any* class, rather than the background, and is similar to object proposals described in Section 2.5.2. Results of this study can be seen in Figure 2.5. Another study combined a point-supervised counting network [55] with object proposals for the task instance segmentation;

achieving state-of-the-art results [56]. This study is described in more detail in Section 2.5.3.

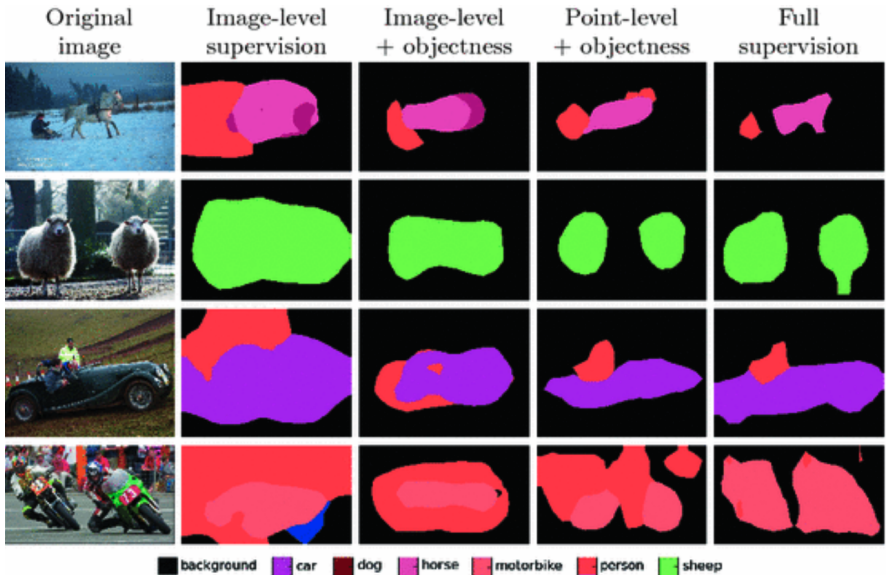



Figure 2.5: Examples of results using point labels for image segmentation from Bearman *et al.* [5]. Various results are shown, including results using image level supervision, image level supervision and objecness priors, point level supervision and objectness priors, and the results using a fully supervised network (from left to right). Permission to use this figure was granted and can be referenced in Appendix 7.4.

### Scribble Labelling

Scribble labels (Figure 2.6 (c)) can offer a larger spatial context than point labels as they consist of lines that span the objects in an image. One study used a graphical model to propagate information from the scribbles to the unmarked pixels using spatial constraints, appearance, and semantic context [57]. This resulted in a pseudo-ground truth that was used to train a FCN.

#### Bounding Box Labelling

Bounding boxes are rectangles that fully surround objects of various classes (Figure 2.6 (d)). Although bounding boxes still contain the background class, they offer a more descriptive label than image-level or point level annotations. Various methods have been created for segmentation with bounding box labels, from simply using the bounding box as pseudo-ground truth, using a CRF to shape the bounding box around the object borders for a more detailed pseudo-ground truth, or using reinforced embedding layers with an EM procedure similar to that described for image-level labels [65]. Another study used the overlap between bounding boxes and object proposals (See Section 2.5.2) in their loss function to guide training [19].

### 2.5.2 Object Proposals

The goal of an object proposal algorithm is to find regions in an image that are likely to contain objects [69]. These algorithms are class agnostic and can be employed for object detection tasks [28] and weakly supervised tasks [19, 56], among others. An ideal object proposal algorithm, has high recall, where this is achieved with the fewest number of proposals as possible which match the objects as accurately as possible [69]. A high recall score maximizes the number of true positives and minimizes the number of false negatives, meaning we hope to have all objects covered by proposals, even if it might mean sacrificing false positives.

#### DeepMask

DeepMask is a popular object proposal network which, given an image, aims to output a collection of object proposals, each with an associated score that indicates the likelihood of that proposal indeed being an entire object [69]. The DeepMask architecture consists of a backbone feature extractor such as VGG [84], which then splits into two branches; one for segmentation and one for scoring. The segmentation branch is responsible for outlining the location of a object, while the scoring branch is responsible for deciding if the input image patch indeed contains an object and if it is roughly centered in the patch. Branching off from the backbone feature extractor, the segmentation branch contains a $1 \times 1$ convolution to reduce the channel dimension, followed by a classification for each pixel in the output mask. The scoring branch consists of $2 \times 2$ max-pooling to reduce the spatial dimensions, followed by two fully connected layers, eventually outputting a single *objectness* score [69].

DeepMask employs joint learning in the sense that the loss function is a sum of two binary logistic regression losses, one for predicted object segmentation masks and one for the *objectness* score. During training, a training triple is provided that contains an input image patch, a binary ground truth segmentation mask, a ground truth indicator variable that indicates if there is indeed an centred object in the patch. Backpropagation is alternated between the scoring branch and the segmentation branch, only backpropagating over the segmentation branch if the ground truth indicator variable for that patch indicates that there is indeed an image in the patch. During training, an equal number of training patches contain an object as those that do not contain an object [69]. During inference, the trained DeepMask model is densely applied at multiple locations and scales to encourage at least one patch to fully contain each individual object in the image.

At the time of its publishing, DeepMask achieved state-of-the-art results. Specifically, DeepMask outperformed another algorithm known as Selective Search [95], a proposal algorithm employed by one of the weakly supervised bounding box algorithms mentioned in Section 2.5.1 [19].

### 2.5.3 Point Labelling for Instance Segmentation

#### Counting Network - LC-FCN

A counting network is a neural network that counts the instances of objects of various classes and can be used in many applications including surveillance, traffic monitoring, ecological surveys, and cell counting [55]. Although object detection networks are theoretically able to count objects by simply counting how many objects are detected by the network, they are also given the difficult task of determining the location, shape, and size of the objects, and are therefore not optimized for counting. Counting networks simplify the task by localizing object instances without a focus on the exact spatial extent of the object. One study proposed a loss function that uses only point-level supervision, and encourages the model to output approximate instance regions in which each instance region overlaps only a single object instance [55]. This method uses a FCN as the network architecture, and a novel localization-based counting loss, and is therefore referred to as LC-FCN [55].

For this method, any FCN can be used. Similar to a segmentation network, LC-FCN outputs a prediction mask with the same spatial dimensions as the input image. Since LC-FCN does not concern itself with the spatial extent of the object, once LC-FCN is trained, it outputs a *blob*-like mask for each object. Unlike a traditional segmentation network, a successful blob segmentation output of LC-FCN does not need to be a detailed outline of the object, but rather there should ideally be a single blob for every object. This means

that one blob should not overlap multiple objects, and there should be no blobs that do not overlap an object. The loss function developed by the authors of LC-FCN can be defined as follows,

$$\mathcal{L}(S,T) = \underbrace{\mathcal{L}_I(S,T)}_{\text{Image-Level Loss}} + \underbrace{\mathcal{L}_P(S,T)}_{\text{Point-Level Loss}} + \underbrace{\mathcal{L}_S(S,T)}_{\text{Split-Level Loss}} + \underbrace{\mathcal{L}_F(S,T)}_{\text{False Positive Loss}}, \tag{2.25}$$

where $T$ is the point annotation ground truth, and $S$ is the softmax output of the network [55]. The loss function is comprised of four separate terms, an image-level loss, a point-level loss, a split-level loss, and a false positive loss; each designed for a specific purpose.

The image-level loss, $\mathcal{L}_I$, encourages at least one pixel in the prediction to be from each class present in the image as defined by the ground truth point labels. It also discourages any of the pixels in the prediction to be from the classes not present in the ground truth point labels. The image-level loss can be defined as follows,

$$\mathcal{L}_I(S,T) = -\frac{1}{|C_e|}\sum_{c\in C_e}\log(S_{t_c c}) - \frac{1}{|C_{\neg e}|}\sum_{c\in C_{\neg e}}\log(1-S_{t_c c}), \tag{2.26}$$

where $C_e$ is the set of classes present in the image as defined by the point labels, $C_{\neg e}$ is the set of classes not present in the image, and the vertical bars $||$ indicate the cardinality or size of the set. $S_{ic}$ is the probability that pixel $i$ in the image $\mathcal{I}$ belongs to class $c$, while $t_c$ is equal to $\text{argmax}_{i\in\mathcal{I}} S_{ic}$. Finally, the variable $S_{t_c c}$ is the probability of the pixel with the highest probability of being class $c$.

The point-level loss, $\mathcal{L}_P$, encourages the model to correctly label the pixels that are used as ground truth point labels. The point-level loss ignores all non-annotated pixels and can be defined as follows,

$$\mathcal{L}_P(S,T) = -\sum_{i\in\mathcal{I}_s}\log(S_{iT_i}), \tag{2.27}$$

where $\mathcal{I}_s$ is the set of supervised pixels, $T_i$ are the ground truth labels of $\mathcal{I}_s$, and $S_{iT_i}$ is the probability that pixel $i$ belongs to the ground truth label.

The split-level loss, $\mathcal{L}_S$, discourages the model from predicting blobs that have more than one point-annotations within. In such a scenario, LC-FCN would count multiple objects as only one, resulting in an inaccurate estimate of the number of objects in the

image. The split-level loss therefore enforces that if a blob has $n$ point annotations within, the blob must be split into $n$ blobs, as so new blob corresponds to a single unique object. LC-FCN uses a watershed segmentation algorithm [7] within $\mathcal{L}_S$ to split objects where ground truth point labels are used as seeds. If we let $T_b$ be the set of pixels that define the boundaries of the watershed segmentation, as illustrated by the yellow lines in Figure 2.7, the split-level loss can be defined as follows,

$$\mathcal{L}_S(S, T) = -\sum_{i \in T_b} \alpha_i \log(S_{i0}), \tag{2.28}$$

where $S_{i0}$ is the probability that the boundary pixel $i$ belongs to the background class, and $\alpha_i$ is the number of point annotation encapsulated by the blob that also encapsulates pixel $i$. This $\alpha_i$ term therefore promotes the model to split blobs that have the most point annotations within. Overall, the split-level loss encourages the model to learn boundaries between objects and therefore output a single blob for each object instance.

The final term in Equation 2.25 is the false positive loss, $\mathcal{L}_F$. This term discourages the model from predicting blobs where no object, and therefore point annotation, exists. The false positive loss can be defined as follows,

$$\mathcal{L}_F(S, T) = -\sum_{i \in B_{fp}} \log(S_{i0}), \tag{2.29}$$

where $B_{fp}$ are the set of pixels that make up blobs for which no ground truth point exists within, and $S_{i0}$ is again the probability that pixel $i$ belongs to the background class.

Figure 2.8 shows how each loss term, $\mathcal{L}_I$, $\mathcal{L}_P$, $\mathcal{L}_S$, and $\mathcal{L}_F$, affects the predictions of LC-FCN. We can see in Figure 2.8 (b), that without $\mathcal{L}_S$ and $\mathcal{L}_F$, the network predicts usually a single mask that covers all ground truth points, satisfying $\mathcal{L}_I + \mathcal{L}_P$. The addition of $\mathcal{L}_S$ splits this single mask into many blobs, where some overlap objects, and some do not, resulting if false positives (Figure 2.8 (c)). Finally, $\mathcal{L}_F$ reduces the number of these false positives, resulting in blobs with only overlap a single object (Figure 2.8 (d)). When published, LC-FCN outperformed all state-of-the-art counting models on a variety of data sets [55].

### Combining Points with Proposals

Some of the authors of LC-FCN extended their work to weakly supervised instance segmentation using point labelling to create a network called Weakly-supervised Instance

SEgmentation (WISE) [56]. As described in Section 2.4, instance segmentation is concerned with the class label of each pixel, as well as the distinct instances of a given class. The WISE network has three main components; a Localization Network (L-Net), an object proposal network, and an Embedding Network (E-Net).

The L-Net is a renaming of the LC-FCN described in Section 2.5.3. This means that given training images of multiple objects and ground truth point labels at each object, L-Net will learn to predict a point at each object, and an associated class with each predicted point. In order to turn these point predictions into an instance segmentation prediction, WISE leverages object proposals and selects an object proposal using the E-Net.

As described in Section 2.5.2, object proposal networks are pre-trained networks designed to output one or more class agnostic object regions for all objects in an image. The WISE network elected to use the SharpMask object proposal network that builds upon DeepMask using an encoder-decoder architecture for more refined object proposal edges [70].

Various methods to turn the point predictions of the L-Net into instance segmentation predictions were tested by the authors of WISE. Recalling that LC-FCN, and therefore L-Net, use the blob predictions from an FCN to estimate the point predictions for objects, the authors of WISE simply used the blob outputs as the instance segmentation predictions. LC-FCN was not designed for the blob outputs to be detailed representations of objects, but rather to simply correspond to the general location of a single object. As a result, it is expected and was shown that the blobs make poor instance segmentation predictions. Referencing Section 2.5.2, object proposal networks have *objectness* scores associated with each proposal indicating the likelihood that the proposal aligns entirely with an object. The authors of WISE tested out a method of selecting the object proposal with the highest *objectness* score which aligns with a point prediction from L-Net. By choosing a single proposal or each point prediction, the authors built an instance segmentation prediction. Finally, rather than simply selecting the proposal with the highest *objectness* score, the authors built E-Net in an effort to more intelligently select an object proposal that aligns with a point prediction.

As mentioned above, the E-Net serves as a means of selecting the best object proposal that aligns with a point prediction from the L-Net. E-Net is a FCN that outputs an embedding vector for each pixel in the input image. The goal of E-Net is for the output embedding vectors to be similar if they belong to the same object instance. The loss function of E-Net, $\mathcal{L}_E$, uses a similarity function, $S(i, j)$, over all pixel embedding pairs where the similarity tends 1 if the embedding pairs are similar, and tends to 0 if they are farther in the embedding space. $\mathcal{L}_E$ and $S(i, j)$ are defined as follows,

$$\mathcal{L}_E = -\sum_{(i,j)\in P} \left[\mathbb{1}_{\{y_i=y_j\}} \log(S(i,j)) + \mathbb{1}_{\{y_i\neq y_j\}} \log(1-S(i,j))\right], \tag{2.30}$$

$$S(i,j) = \exp\left(-\frac{\|E_i - E_j\|_2^2}{2d}\right), \tag{2.31}$$

where $i$ and $j$ are two pixels where $i \neq j$, $E_i$ and $E_j$ are the embedding vectors for pixels $i$ and $j$ respectively, and $d$ is the size of the embedding vectors.

Since there is no object ground truth, but rather only point labels, a random pseudo mask is selected from the object proposals aligning with the one of the points in order to get an estimate of if two embeddings are from the same object allowing for the use of Equation 2.30. Once E-Net is trained, given an input image, the output embeddings of E-Net are used to create a pseudo masks where pixel embeddings are grouped based on similarity (Equation 2.31) to the embeddings of the points predicted by L-Net. Finally, the intersect over union score (see Chapter 4) between these pseudo masks and the set object proposals are calculated. The object proposal with the highest intersect over union score is selected as a mask for the object point of interest and is used in the final instance segmentation prediction.

(a) Imgae-Level Labelling

(b) Point Labelling


(c) Scribble Labelling

(d) Bounding Box Labelling

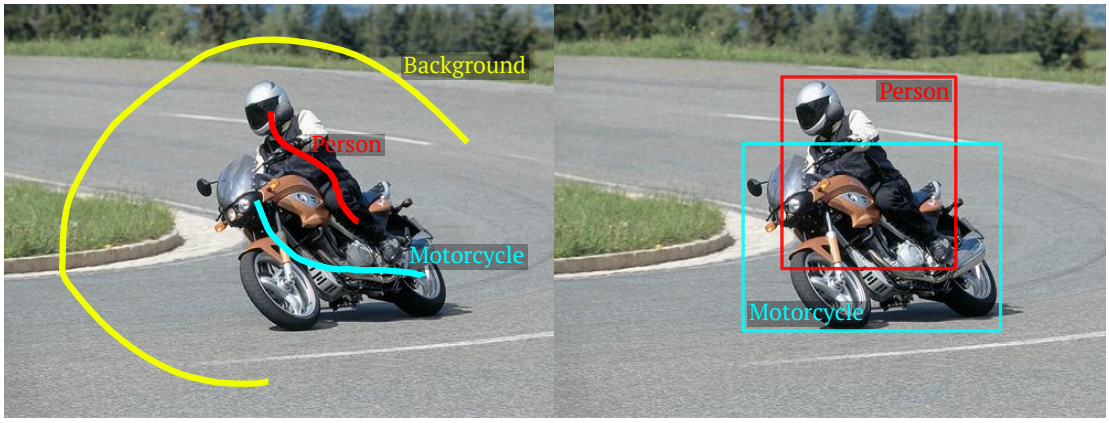



Figure 2.6: Examples of image-level labelling (a), point labelling (b), scribble labelling (c), and bounding box labelling (d) for an example image from the PASCAL VOC 2012 Dataset [23].

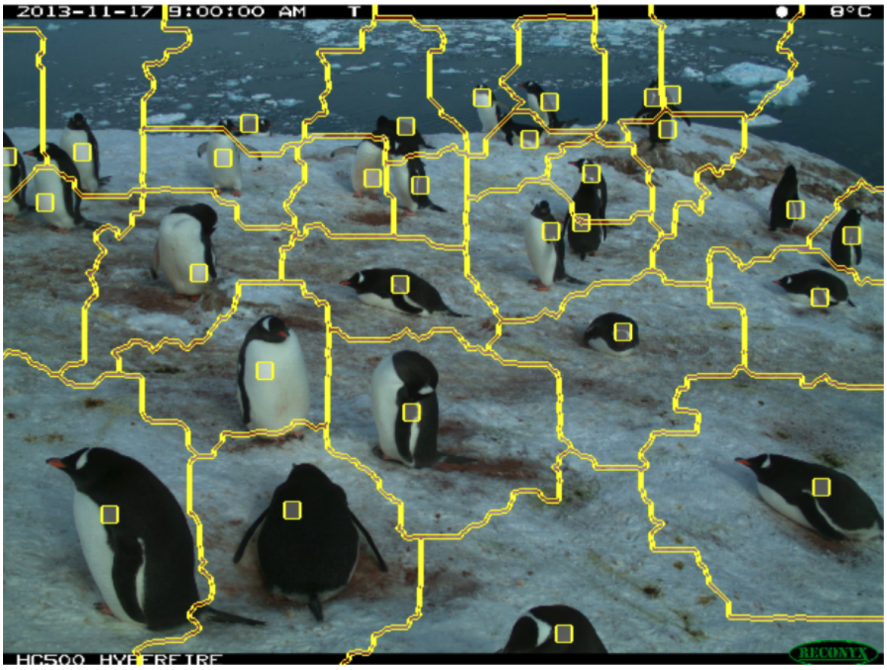



Figure 2.7: Figure showing an example of watershed segmentation from Laradji *et al.* [55] with proof of permission shown in Appendix 7.3. Small yellow square points are ground truth points, while the jagged lines between the points are the watershed segmentation. Pixels intersecting with the watershed segmentation are used in the split-level loss outlined in 2.28.

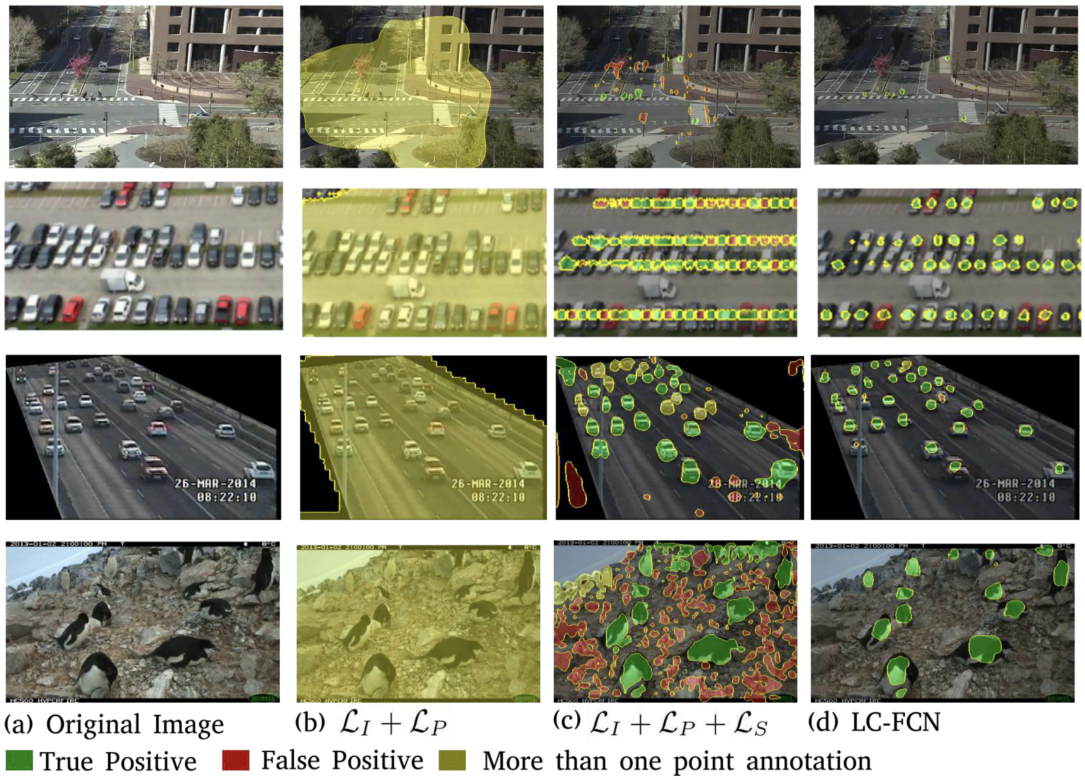



Figure 2.8: Figure showing qualitative LC-FCN results from Laradji *et al.* [55] with proof of permission shown in Appendix 7.3. Test images are shown in column (a), predictions using the sum of Equations 2.26 and 2.27 as a loss function are shown in column (b), predictions using the sum of Equations 2.26, 2.27, and 2.28 as a loss function are shown in column (c), and predictions using the final loss function defined in Equation 2.25 are shown in column (d). Green regions indicate blobs that are true positives, red regions indicate blobs that are false positives, and yellow regions indicate blobs that contain more than one object instance as defined by the ground truth point labels.

# Chapter 3

# Data

## 3.1 Study Area

The Alberta River Ice Segmentation Dataset contains a variety of labelled and unlabelled RGB images collected from the Peace River and North Saskatchewan River in Alberta, Canada, using a Blade Chroma UAV [86]. The Peace River has a rich history of ice jams and flooding [59], and while ice jams and flooding are less common in the North Saskatchewan River, other ice related challenges exist including water intakes blocked by frazil ice and turbidity during breakup giving issue to water treatment plant operators [38]. Images collected on the Peace River were collected at the Dunvegan Bridge boat launch and Shaftesbury Ferry crossing (Figure 3.1 (a)) on January 21-23, 2016 and January 14-15, 2017. Images collected on the North Saskatchewan River were collected at the Genesse boat launch (Figure 3.1 (b)) on December 1 and 3, 2016 [47].

## 3.2 Data Split

The Alberta River Ice Segmentation Dataset contains 50 $1280 \times 1080$ labelled RGB images, in which each labelled segmentation mask details the locations of frazil ice pans, anchor ice pans, and open water. For our experiments, the 50 images were randomly split such that 30 images are used for training, 10 images are used for validation during training, and 10 images are used for testing. Proportions of each pixel class in the entire labelled dataset, and each subset can be seen in Table 3.1. The images in each split were down-sampled by a factor of 3.125 using local averaging, and cropped to a size of $320 \times 320$

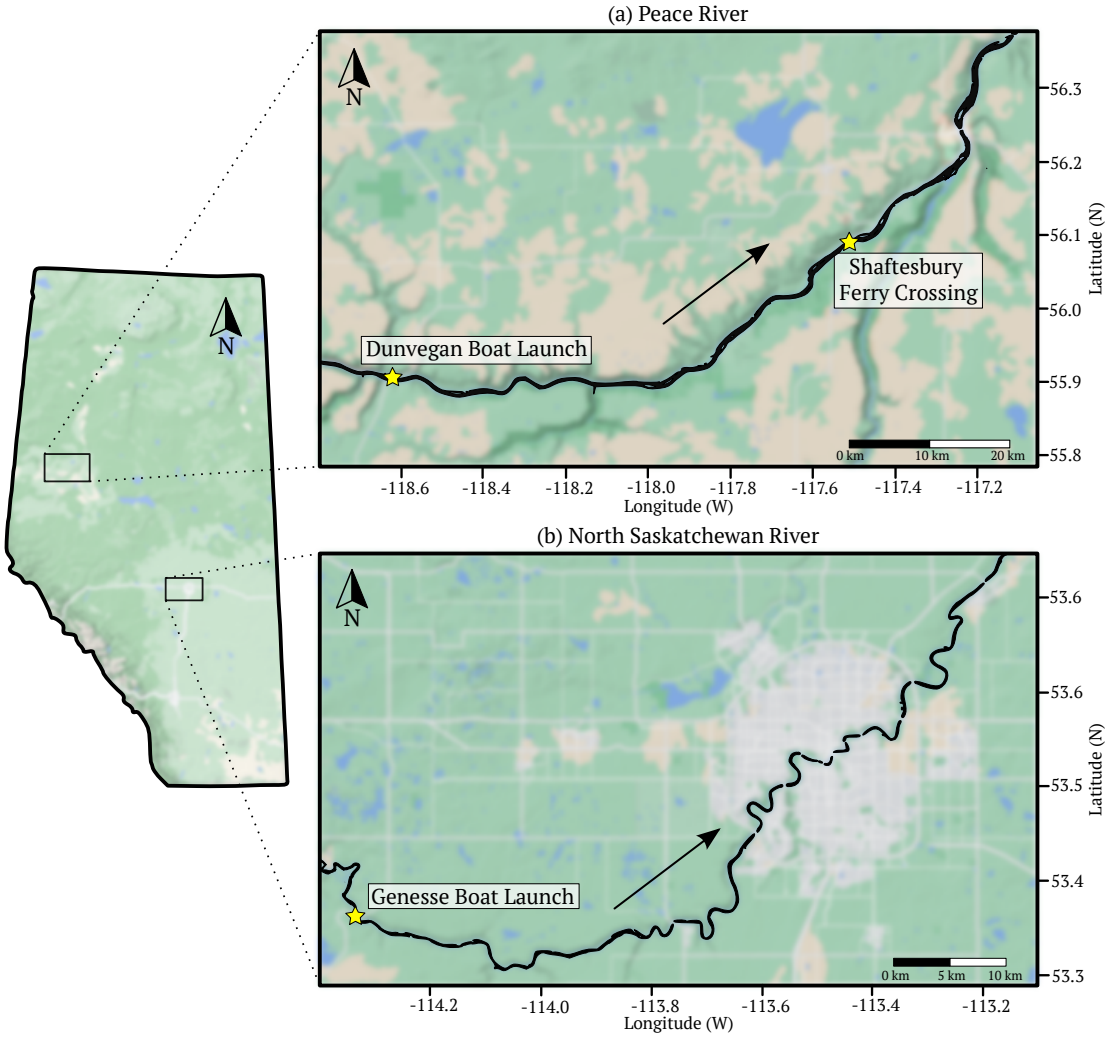



Figure 3.1: Peace River (a) and North Saskatchewan River (b) study sites. UAV images were taken at the Dunvegan Boat Launch and Shaftesbury Ferry Crossing on the Peace River and at the Genesee Boat Launch on the North Saskatchewan River, all shown with gold stars. Arrows indicate the direction of flow.

for more efficient usage on hardware with limited memory. Note that a previous study on the Alberta River Ice Segmentation Dataset chose to split the data into only a training set and a test set consisting of 32 and 18 images respectively [87]. We chose to split the data differently as we require a validation set for some of our experiments. Specifically we utilize a validation set for early stopping and an analysis of overfitting. Nonetheless, for consistency and comparability, we also conduct some experiments using the same 32 image training set and 18 image test set as used in previous work.

Table 3.1: Percentage of water, anchor ice, and frazil ice in the entire Alberta River Ice Segmentation Dataset, as well as the training set, validation set, and test set. Proportions were calculated according to the ground truth pixel labels.

| **Data Split** | **Water** | **Anchor Ice** | **Frazil Ice** |
|---|---|---|---|
| All Data | 61.3% | 16.5% | 22.2% |
| Training Set | 62.7% | 16.8% | 20.5% |
| Validation Set | 58.2% | 14.3% | 27.5% |
| Testing Set | 60.5% | 17.1% | 22.4% |

# Chapter 4

# Metrics

Pixel Accuracy ($pA$), Mean Pixel Accuracy ($mPA$), Mean Intersect over Union ($mIoU$), and Frequency Weighted Intersect over Union ($fwIoU$) are the common metrics used to evaluate the performance of image segmentation models [60].

- Pixel Accuracy is the ratio of all correctly classified pixels to the total number of pixels,

$$pA = \frac{\sum_{j=1}^{k} p_{jj}}{\sum_j t_j}, \tag{4.1}$$

  where $k$ is the number of classes, $p_{jj}$ is the total number of pixels both classified and labelled as class $j$, and $t_j$ is the total number of pixels labelled as class $j$.

- Mean Pixel Accuracy is the ratio of correctly classified pixels to total labelled pixels per class, averaged over the total number of classes. This metric weights the accuracy of each class equally, avoiding one class dominating the overall score if, for example, the classes are imbalanced. For $k$ classes, Mean Pixel Accuracy can be calculated by

$$mPA = \frac{1}{k} \sum_{j=1}^{k} \frac{p_{jj}}{t_j}, \tag{4.2}$$

  where $p_{jj}$ is the total number of pixels both classified and labelled as class $j$, and $t_j$ is the total number of pixels labelled as class $j$.

- Mean Intersect over Union is the ratio of the intersection of the predicted segmentation with the ground truth to the union of the predicted segmentation with the ground truth.

$$mIoU = \frac{1}{k}\sum_{j=1}^{k}\frac{p_{jj}}{p_{ij}+p_{ji}+p_{jj}}, \quad i \neq j \tag{4.3}$$

  where $k$ is the number of classes, $p_{jj}$ is the total number of pixels both classified and labelled as class $j$, $p_{ij}$ is the number of pixels labelled as class $i$ but classified as class $j$, and $p_{ji}$ is the total number of pixels labelled as class $j$ but classified as class $i$.

- Frequency Weighted IoU ($fwIoU$) is a variation of $mIoU$ which aims to account for class imbalances. This is done by weighting the $IoU$ value of a specific class inversely proportional to the frequency of that class' occurance in the dataset. Frequency Weighted IoU can be calculated as follows,

$$fwIoU = \left(\sum_{j=1}^{k} t_j\right)^{-1} \sum_{j=1}^{k}\frac{t_j p_{jj}}{t_j + \sum_{i=1}^{k} p_{ij} - p_{jj}}, \quad i \neq j \tag{4.4}$$

  and similar to variable definitions of $mIoU$, $k$ is the number of classes, $p_{jj}$ is the total number of pixels both classified and labelled as class $j$, $p_{ij}$ is the number of pixels labelled as class $i$ but classified as class $j$, and $t_j$ is the total number of pixels labelled as class $j$.

All of $pA$, $mPA$, $mIoU$, and $fwIoU$ can be expressed as a percentage from 0 to 100, where higher numbers indicate better performance. Note that $mPA$ and $mIoU$ are primarily used to evaluate results in this thesis. We use $pA$ and $fwIoU$ to make direct comparisons to previous work that also used these metrics [87].

# Chapter 5

# Shallow DSC-LBC Network

## 5.1 Methodology

### 5.1.1 Architecture

We propose a novel convolutional DSC LBC block that melds DSCs and LBCs in order to reduce the number of operations and trainable parameters. This operation is illustrated in Figure 5.1 and operates similar to a DSC in that there is first a depthwise convolution, followed by a pointwise convolution. Similar to a DSC, the depthwise convolution stage of a DSC LBC block has $c_{in}$ filters of size $k \times k \times 1$, where a single distinct filter is applied to each input channel. The DSC LBC block, however, differs from a DSC in that the depthwise convolution stage uses non-trainable kernels with values initialized from the set $\{-1, 0, 1\}$, similar to a LBC. This adds sparsity and reduces the total trainable parameters of the DSC LBC block, in contrast to a DSC which uses trainable kernels in the depthwise stage. The initialization of the depthwise kernels in a DSC LBC block occurs in the same manner as described in Section 2.3.2. When initializing the non-trainable kernels, we use a sparsity of 80%[1] as this level of sparsity achieved the best performance in previous experiments [46]. Finally, the only trainable parameters in a DSC LBC block exist in the $1 \times 1$ pointwise convolution. As a result of this DSC LBC block architecture, we achieve the same computational cost as a DSC of $h \cdot w \cdot c_{in}(k^2 + c_{out})$, while using the same number of trainable parameters as a LBC, specifically $c_{in} \cdot c_{out}$.

[1]Note that we follow the same convention as the LBCNN authors where the sparsity percentage refers to the percentage of non-zero elements *i.e.*, sparsity=100% corresponds to a dense weight tensor [46].

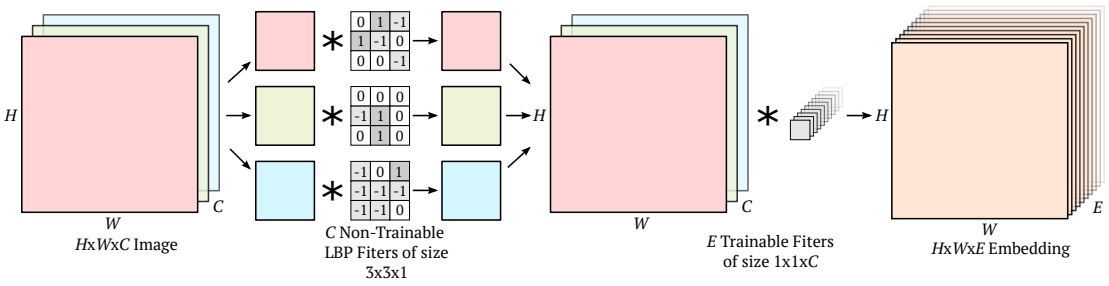



Figure 5.1: DSC LBC convolution block used to replace a traditional convolution block. A given convolution operation can be considered to have $X$ filters of size $K$x$K$x$Y$ where $X$ is the number of output channels, $Y$ is the number of input channels divided by the number of convolution groups, and $K$ is the width and the height the kernel. A DSC LBC convolution replaces the trainable depthwise filters of a depthwise separable convolution [17] with sparse, non-trainable filters inspired by local binary patters [46]. Note that there is only one distinct non-trainable binary filter convolved with each input channel in the first stage of the DSC LBC convolution block.

A UNet was chosen as a baseline model due to its simplicity in implementation, widespread use, intuitive interpretation, and success in both performance and generalization ability in previous work [87]. Figure 5.2 (a) shows the full architecture of a standard UNet as it was originally proposed [76]. For the sake of comparison, various versions of the UNet architecture were created, each with a different convolution operation. Here, the term UNet simply refers to a standard UNet with standard convolution operations, while a DSC UNet replaces standard convolutions with DSCs, and a LBC UNet replaces standard convolutions with LBCs. Finally we compare these UNet variations to a DSC LBC UNet which replaces standard convolution operations in a UNet with our DSC LBC convolution block. All UNet variations use the same kernel size of $3 \times 3$.

Previous research has suggested that shallower networks perform better than deeper networks when the dataset of interest is small [81, 66]. As the Alberta River Ice Segmentation Dataset is considerably small with only 50 images, we experiment with the size of the UNet with regards to the number of down-sampling and up-sampling layers. The aforementioned UNet variations, which we will now refer to as Full UNets, have four down-sampling and up-sampling layers. We have created similar UNets with only two down-sampling and up-sampling layers and will refer to these as Small UNets (Figure 5.2 (b)). The Small UNets also differ from the Full UNets in the number of embedding dimensions, or channels used in the intermediate layers. The first two layers of the Full UNets contain 64 and 128 embedding dimension respectively, while the first two layers of the Small UNets contain

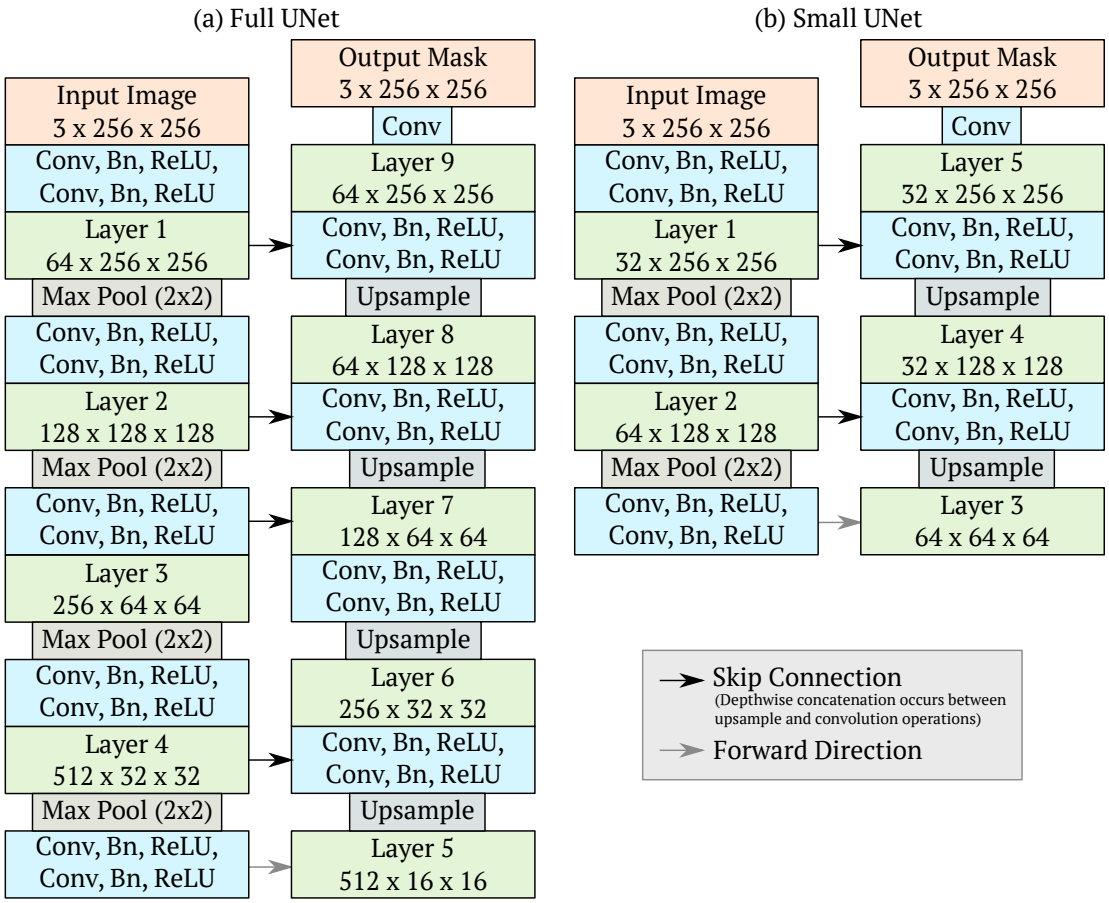



Figure 5.2: Original UNet architecture shown in (a) [76], and smaller UNet architecture shown in (b) with only two down-sampling stages. If the Conv operation in the blue boxes is a standard convolution, the networks are referred to as Full and Small UNets. If the Conv operation is a DSC operation, the networks are referred to as Full and Small DSC UNets. If the Conv operation is a LBC operation, the networks are referred to as Full and Small LBC UNets. If the Conv operation is a DSC LBC operation (Figure 5.1), the networks are referred to as Full and Small DSC LBC UNets.

32 and 64 dimensions respectively. This further reduces the memory requirements of these small networks with the goal to improve training and inference efficiency. Finalizing our terminology, the Small UNet uses standard convolutions, while the Small DSC UNet, Small LBC UNet, and Small DSC LBC UNet use DSCs, LBCs, and DSC LBC blocks respectively.

Finally, as we are exploring the performance-latency trade-off for river ice segmentation, we compare the UNet approaches to state of the art MobileNets which are optimized for CPU usage on mobile devises [36]. These networks are built to be fast and efficient, while minimizing any drop in performance. We elect to use use both DeepLabV3 and and the more modern LR-ASPP models with a MobileNetV3 backbone. Both these networks have a performance-latency trade-off associated with an output stride; the ratio of the input image size to the output feature map size [79]. A default output stride of 16 was used for DeepLabV3, while a combination of an output stride 8 and 16 was used for low and high level features in the LR-ASPP. These two networks will be referred to as MobileNetV3 (DeepLabV3) and MobileNetV3 (LR-ASPP) respectively.

### 5.1.2 Experiments

#### Training Procedure

We run a series of experiments to compare the suite of Full UNets, Small UNets, and MobileNets. The PyTorch library [68] was used to train all models until a stopping criteria had been satisfied. Training was stopped after 30 consecutive training epochs occurred where the validation loss had not reached a new low. After this stopping criteria is triggered, we roll back to the 30th last training epoch where validation loss was at its lowest. This was done in order to mitigate any overfitting that may have occurred during the 30 epochs where validation loss did not decrease. Thirty epochs was chosen for the stopping criteria as there was significant variation in the validation loss during training due to a small validation set of only 10 images. A batch size of one was used for all experiments in order to maintain consistency and satisfy hardware limitation for the more memory intensive models. A learning rate of 1e-4 was used along with a standard cross entropy loss function,

$$\mathcal{L}_{CE} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i), \tag{5.1}$$

where $y_i$ and $\hat{y}_i$ are the ground-truth and network output for the $i^{th}$ class of $n$ total classes.

We use an RMSprop optimizer with a weight decay of 1e-8 and a momentum of 0.9 [91]. Training was conducted on both a GPU and CPU, specifically a Nvidia GeForce GTX 1060 and AMD Ryzen 5 2600 Six-Core Processor respectively. After every three training iterations, a validation step is performed in where the loss function and other evaluation metrics are evaluated for all 10 validation images. This procedure allows for further insight regarding over-fitting and how fast the loss function of validation predictions decreases during training.

### Synthetic Data for Generalization Experiments

After the suite of UNets and MobileNets have been compared on the Alberta River Ice Segmentation Dataset, we explore the ability of these networks to generalize by adding synthetic snow to a varying number of images in the training and test sets. We evaluate the results of these tests with the non-contaminated ground truth to get quantitative results to supplement the qualitative results mentioned in previous work [87]. The synthetic snow was designed by observing UAV footage obtained during a real snowfall [86] (Figure 5.3 (a)), and was added using using the OpenCV library [10] (Figure 5.3 (c)). The snow observed in the UAV footage has either a point-like or streak-like appearance as seen in Figure 5.3 (a). As a result, a line in OpenCV was used to simulate snow where the length and the width of the lines were sampled from a gamma distribution. If the length and width of the line are similar, the snow appears point-like, while if the length is larger than the width, the snow appears streak-like (Figure 5.3 (d) & (e)). A gamma distribution was chosen since the majority of snow flakes are far from the camera and therefore appear smaller, while there are fewer snowflakes near the camera that appear larger. The distribution for the length of the line was sampled from a gamma distribution with a slightly higher mean and standard deviation than the distribution used to sample the width of the line; resulting in some streak-like snow. The slant and the color of the line was chosen from a uniform distribution, where the red, green, and blue colour channels were sampled in the range of 190 to 210 to give the snowflakes a varying greyish white colour.

Another environmental factor that can affect the generalization ability of a network is the illumination [24]. If images are collected early or late in the day they can appear dark as compared to if they are collected during the day. We test the ability of the networks to generalize to illumination by synthetically darkening and lightening the images, and training on one group and testing on the other. Specifically, keeping the same training, validation, and testing splits, we train on dark images and test on light images, as well as train on light images and test on dark images. We choose to characterize the illumination of an image by the mean grey-scale pixel intensity of only the water in an image. By

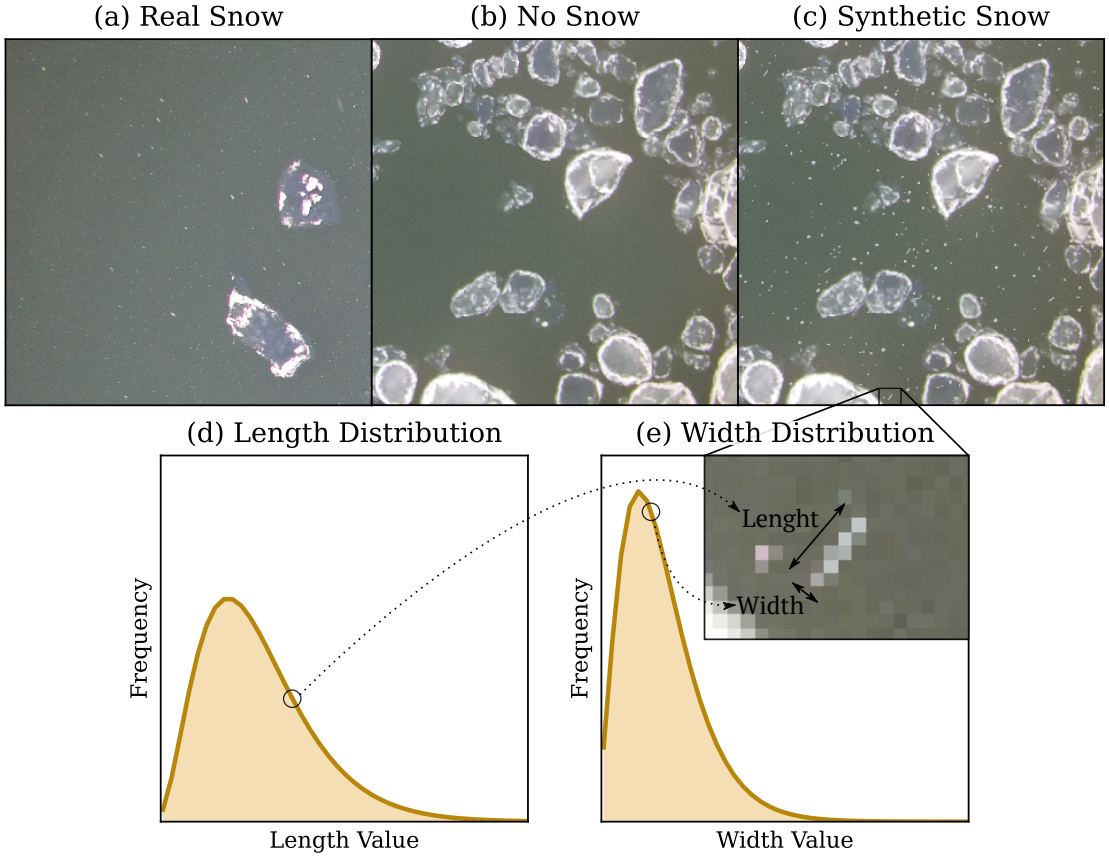



Figure 5.3: Examples from the Alberta River Ice Segmentation Dataset [86] taken while it is snowing (a), while it is not snowing (b), and with synthetic snow added to an image which originally had no snow (c). An example of a synthetic streak-like snow flake is shown with its length and width sampled from gamma distributions for length (d) and width (e). The distribution for length (d) can be seen to have a larger mean and standard deviation than the distribution for width, forcing some streak-like snow as shown in the zoomed in square. Note that snow can be seen better when zoomed in.

browsing the dark and light images in the Alberta River Ice Segmentation Dataset, we find that the darkest images in the dataset have a mean water grey-scale value of around 75, and the lightest images have a mean water grey-scale value of around 120. We then make duplicates of the Alberta River Ice Segmentation Dataset, forcing all images to have a mean water grey-scale value of 75 in one duplicate of the dataset, and forcing all images to have a mean water grey-scale value of 120 in another duplicate of the dataset. This involves either synthetically darkening or lightening the image depending on if was originally darker or lighter than the target value of 75 or 120. Examples of original images from the dataset can be seen in Figure 5.4 (a) and (b), while a synthetically lightened image can be seen in Figure 5.4 (c), and a synthetically darkened image can be seen in Figure 5.4 (d). The OpenCV library [10] was used to change the illumination through an inverse gamma correction defined as follows,

$$p_{out} = 255 \left( \frac{p_{in}}{255} \right)^{\frac{1}{\gamma}}, \tag{5.2}$$

where $p_{in}$ is an input pixel intensity, $p_{out}$ is the output pixel intensity, and $\gamma$ is a scaling factor for which a higher value results in a lighter image, a lower value results in a darker image, and a value of one results in no change.

### Model Timing

Finally, we attempt to get a better understanding of how model factors such as trainable parameters, multiply-add operations, and memory usage in a model affect the latency during training and inference. We first measure training time over three training epochs for all models on a GPU and CPU to see how fast the various architectures train. Then, the best models from this analysis are compared with respect to the time required to reach a specific metric threshold. This comparison will give a more practical idea of which model is most appropriate to achieve good results when training in a resource and time constrained setting. Then, we finish by measuring the inference time over 10 images for all models on a GPU and a CPU to see how fast river ice images can be evaluation in practice once a model in trained.

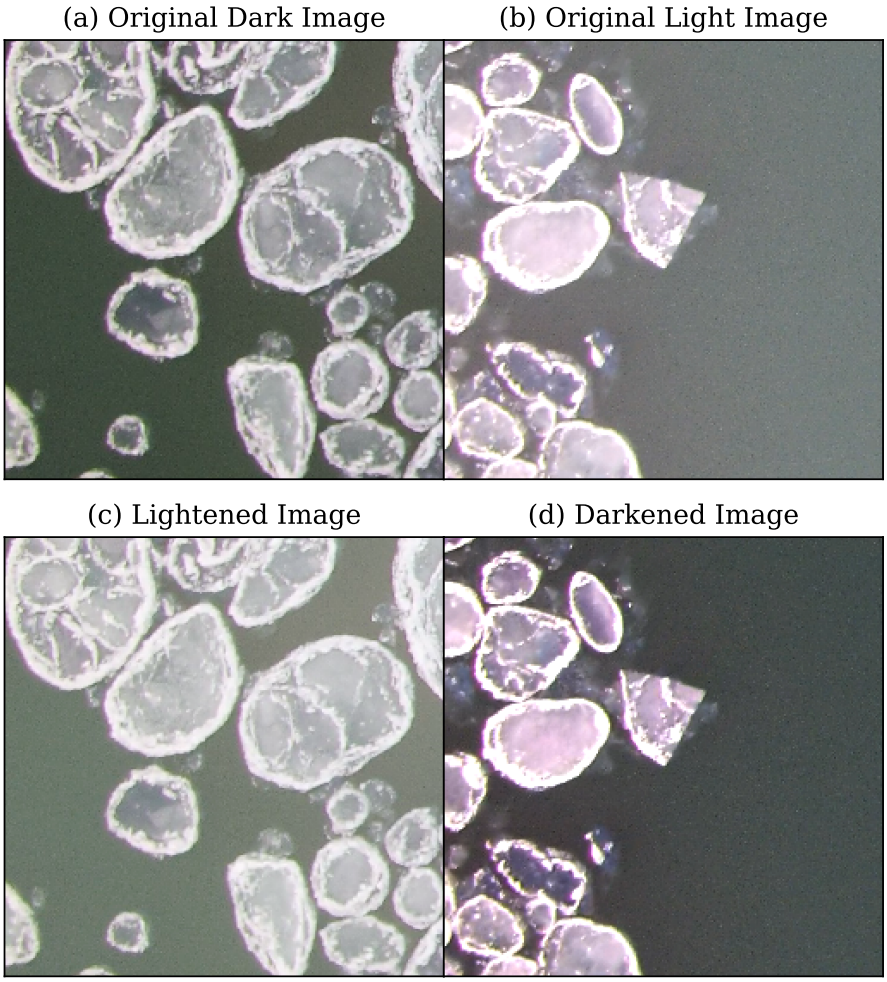



Figure 5.4: Examples of images from the Alberta River Ice Segmentation Dataset [86] shown in (a) and (b) along with examples of synthetically lightened (c) and synthetically darkened (d) images. Images (a), (b), (c), and (d) have mean water grey-scale pixel intensities of 78.2, 124.5, 121.1, and 73.1 respectively.

## 5.2 Results

### 5.2.1 Model Comparison

Table 5.1 shows a comparison of the various models tested with respect to the evaluation metrics on the test set, the number of total and trainable parameters in the model, the number of multiply-add operations in a forward pass of the model, and the memory usage of the model. The metrics were calculated by training three instances of each model, saving the respective model weights at the stopping epoch determined during training (Section 5.1.2), evaluating the models on the test set, and averaging the three results for each model. Memory usage and Mult-Add operations were calculated according to one $320 \times 320 \times 3$ image passed through the model. For a more direct comparison to previous results of UNet on the Alberta River Ice Segmentation Dataset, see Appendix A. Due to our experiment structure we elect to use a validation and test set, rather than a single test set, as well as more general metric reporting.

Table 5.1: Metrics, number of parameters, number of multiply-add operations and memory usage in the tested networks. Metrics were calculated by training three instances of each model, evaluating them on an external test set using weights saved at the stopping criteria, and averaging the three results. Memory and Mult-Adds were calculated using an input size of $320 \times 320 \times 3$. Numbers in bold represent the highest metric or lowest number of parameters/operations/memory of the models tested. Small DSC LBC UNet is in bold as it shows a healthy trade-off between high performance metrics and low parameters/operations/memory.

| **Model** | **mIoU** | **mPA** | **Total Params** | **Trainable Params** | **Mult-Adds** | **Memory** |
|---|---|---|---|---|---|---|
| Full UNet | 69.7 | 82.7 | 17,267,523 | 17,267,523 | 62,764 M | 1542 MB |
| Full DSC UNet | 69.7 | 82.9 | 1,983,713 | 1,983,713 | 7,717 M | 2242 MB |
| Full LBC UNet | 67.7 | 82.2 | 16,498,842 | 794,697 | 71,048 M | 1164 MB |
| Full DSC LBC UNet | 69.1 | 82.7 | 821,220 | 794,697 | 3,402 M | 1104 MB |
| Small UNet | **71.2** | **85.0** | 260,451 | 260,451 | 8,237 M | 622 MB |
| Small DSC UNet | 70.4 | 84.3 | 35,777 | 35,777 | 1,123 M | 936 MB |
| Small LBC UNet | 69.7 | 83.8 | 253,050 | 13,353 | 9,479 M | 468 MB |
| **Small DSC LBC UNet** | 70.1 | 83.5 | **16,260** | **13,353** | **637 M** | 466 MB |
| MobileNetV3 (DeepLabV3) | 63.7 | 78.5 | 11,020,851 | 11,020,851 | 3,898 M | 589 MB |
| MobileNetV3 (LR-ASPP) | 65.1 | 80.0 | 3,218,478 | 3,218,478 | 826 M | **307 MB** |

### Evaluation Metric Results

When comparing the models within Table 5.1, the first observation that can be made with respect to evaluation metrics is that the Small UNet variants outperform their Full UNet equivalents. For example, Small UNet outperforms Full UNet, Small DSC UNet outperforms Full DSC UNet, and so on. This gives credibility to the notion that shallower networks outperform deeper networks on smaller datasets.

The Small UNet achieved the highest $mIoU$ and $mPA$ results, while the Small DSC LBC UNet used the lowest number of trainable parameters and Mult-Add operations. The Small DSC LBC UNet also had the second smallest memory footprint, second only to MobileNetV3 (LR-ASPP). When comparing the Small DSC LBC UNet to the Full UNet, the Small DSC LBC UNet has a $mIoU$ that is virtually the same, only 0.6% higher, but with 99.9% less trainable parameters, 99.0% less Mult-Add operations, and 69.8% less memory usage. When comparing the Small DSC LBC UNet to the best performing Small UNet, the Small DSC LBC UNet has a $mIoU$ that is 1.5% lower; however, is still has 94.9% less trainable parameters, 92.3% less Mult-Add operations, and 25.1% less memory usage. Finally, when comparing the Small DSC LBC UNet to MobileNetV3 (LR-ASPP), although the Small DSC LBC UNet uses 51.8% more memory, it uses 23% less Mult-Adds, has 99.6% less trainable parameters, and has a $mIoU$ that is 7.7% higher.

### Visual Results

Figure 5.5 shows model predictions on three test images using model weights saved at the stopping epoch determined during training. When looking at the predicted segmentation outputs, an obvious observation is that the MobileNets have a blobby nature to their segmentation predictions where edges do not have fine detail and smaller ice pans are unnoticed or combined into larger predictions. This nature of MobileNets is acknowledged by the authors and is attributed a larger output stride which allows for parameter saving but reduces the resolution of the predicted masks [79].

Another observation can be made regarding the LBC portion of the networks which can be seen most clearly in the Small LBC UNet of Figure 5.5 and even more so in the Small LBC UNet of Figure 5.7. The LBC component of the networks results in a noisy pixelated appearance of the predicted masks, especially around the boarders of ice pans. The pixilated artifacts are still present, but to lesser extent, in the Small DSC LBC UNet and appear to be reduced with more training. These artifacts are likely due to the sparse binary nature of the LBP filters that propagate through the $1 \times 1$ convolutions early in training. As training progresses, the $1 \times 1$ convolutions are able to account for the sparsity

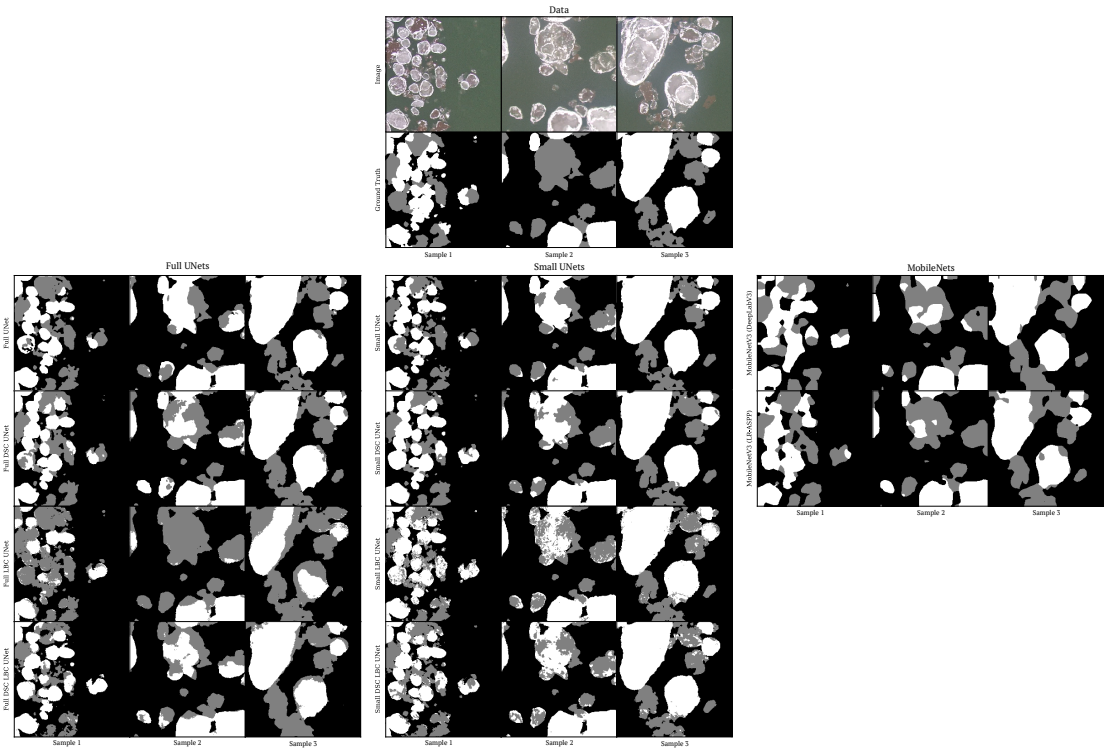



Figure 5.5: Segmentation results on three images from the test set. Model weights for the various models were chosen based on a stopping criteria during training (see Section 5.1.2). The original image and ground-truth are shown at the top, the Full UNet variations are shown on the left, the Small UNet variations are shown in the centre, and the two MobileNet variations are shown on the right. Water, anchor ice, and frazil ice are coloured in black, gray, and white respectively. Note that details in the images and segmentation predictions can be best seen when zoomed in.

and reduce the pixelated appearance of the predictions. Additionally, this artifact is not as obvious in the Full LBC UNet likely due to the deeper architecture, allowing for more superposition and smoothing of the sparse kernels, limiting the ability for the sparsity to propagate to the predictions early in training.

Finally a comparison can be made between the predictions of the Full and Small UNet variations. The Small UNets appear to contain more noise within individual ice pans, while the Full UNets have little variation in class prediction within an individual pan. This can be attributed to the increased down-sampling that occurs within the Full UNets. As down-sampling continues, noise within feature maps get smoothed out and the network learns to value the up-sampled feature maps where the noise is no longer present. Although the smooth predictions appear more realistic, the actual class prediction of the smooth masks are not always correct, which is reflected in the $mIoU$ and $mPA$ scores.

### 5.2.2 Training Curves

#### Learning Efficiency

Analyzing the $mIoU$ and cross entropy loss of the validation set during training can give insight into the speed of training on an epoch-per-epoch basis, as well as any overfitting that may be occurring. Figure 5.6 shows the $mIoU$ and cross entropy loss of the validation set during 80 epochs of training. It can be seen in Figures 5.6 (a) and (b), that certain models require fewer training iterations in order to reach a given $mIoU$ value. In particular, the Small DSC LBC UNet requires the lowest number of training iterations to reach high scores, with the Full DSC LBC UNet, Small DSC UNet, and Small LBC UNet also reaching higher scores considerably sooner than the other models. When comparing these results to Table 5.1, models with a low number of trainable parameters seem to be the ones that require few iterations to reach high scores. Even still, the combination of DSCs and LBCs, regardless of if the network is small or large, appears to result in high performance early in training.

Figure 5.7 shows the predictions of the Small UNets on one image of the test set after one epoch of training. The predictions made by the Small DSC LBC UNet are visually much more consistent with the ground truth than the other models at this early stage of training, showing consistency with the training curves of Figure 5.6. Taking a closer look at Figure 5.7, there is a key difference between how the Small UNet, Small DSC UNet, and Small LBC UNet learn early in training. The Small UNet and Small DSC UNet are similar in that they appear to focus on pixel intensity in the early stages of training. The upturned bright edges of the anchor ice are often mistaken for the light frazil ice, and

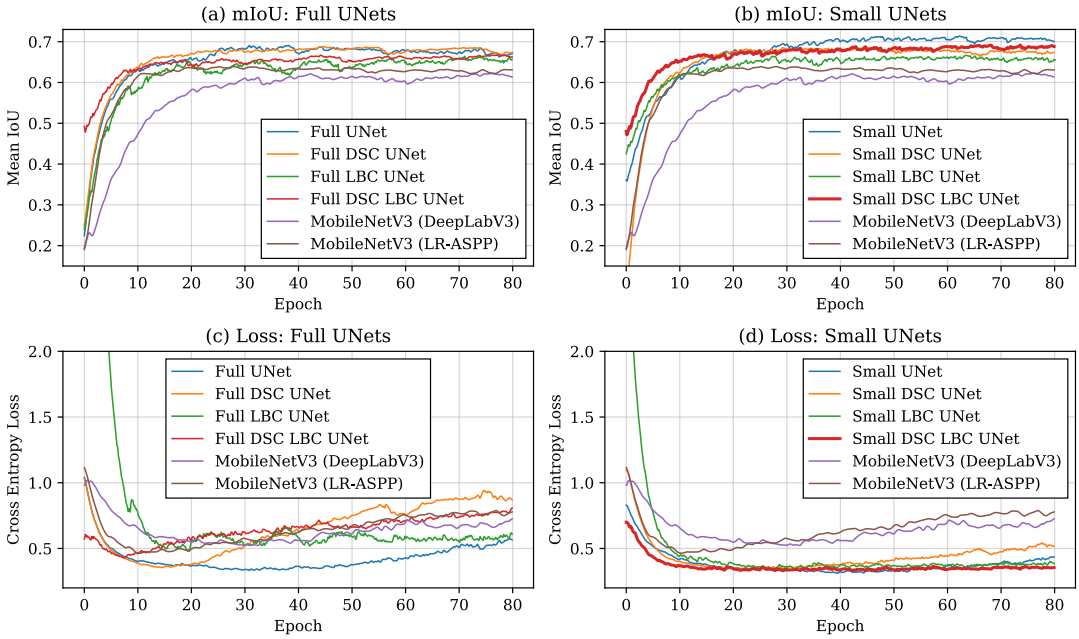



Figure 5.6: Mean IoU of the validation set after 80 training epochs is shown for the Full UNets (a) and Small UNets (b). Cross entropy loss for the validation set is also shown over 80 epochs for the Full UNets (c) and Small UNets (d). Note that the MobileNets are shown in all chats for the sake of comparison. The data in all of the charts has an exponential moving average applied with a smoothing factor or 0.95 to filter noise caused by a small validation set.

the dark interior of the anchor ice is often mistaken for water. On the other hand, the Small LBC UNet tends to focus more on texture in the early stages of training. The rough upturned edges of either ice class are categorized to be anchor ice, while the smooth interiors of the ice pans of either class are categorized as frazil ice. This behaviour of the Small LBC UNet to focus on texture is understandable as the local binary patterns in the kernels are designed to be texture descriptors. The contrasting behaviour of the Small DSC and LBC UNets likely aids in the performance of the Small DSC LBC UNet early in training as it does not appear to make either of the early-training mistakes of the DSC or LBC networks previously mentioned.

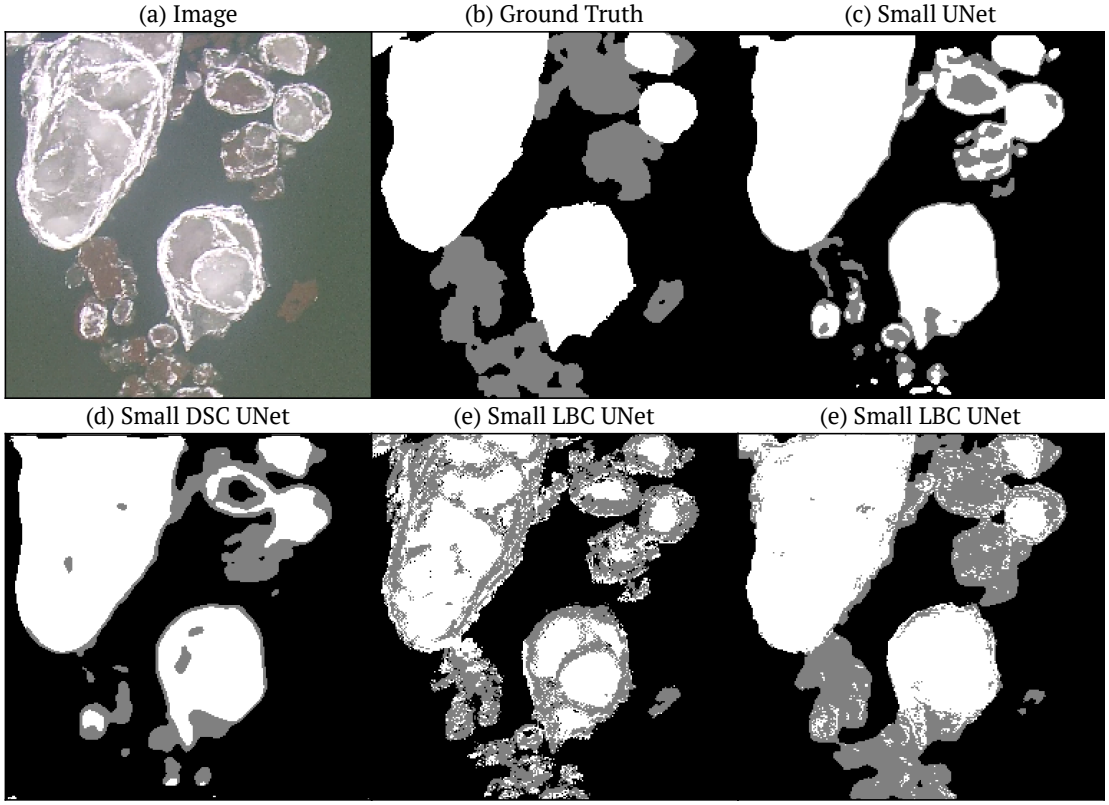



Figure 5.7: Comparison of predictions after one epoch of training on the suite of Small UNets. The mean IoU for the Small UNet, Small DSC UNet, Small LBC UNet and Small DSC LBC UNet are 65.9, 67.2, 56.7, and 78.7 respecivley. The mean pixel accuracy for the Small UNet, Small DSC UNet, Small LBC UNet and Small DSC LBC UNet are 75.0, 76.0, 69.2, and 88.8 respectively. Note that detail can be more easily observed when zoomed in.

### Overfitting Observation

It is also important to mention the behaviour of the cross entropy loss of the validation set during training. It can be seen in Figure 5.6 (c) that validation loss of the Full UNets and MobileNets eventually increases later in training; a sign of overfitting. When contrasting Figures 5.6 (a) and (c), we see that although cross entropy loss increases significantly, the $mIoU$ values decrease only very slightly if at all. This discrepancy is attributed to the discrepancy between the softmax output that is used to calculate cross entropy and the argmax output that is used to calculate $mIoU$ and $mPA$. As training proceeds, results are worsening from the perspective of the loss function due to less confident pixel predictions expressed by lowering softmax values. Due to the logarithmic nature of the cross entropy function, changes in smaller softmax values have a larger effect than changes in larger softmax values. However, once an argmax is applied to the softmax, as long as the lowered softmax values are still larger than those of the other classes, the argmax will still have the same result as when the softmax values were more confident. This leads to similar segmentation predictions and no significant decrease in $mIoU$ and $mPA$ scores.

### Overfitting Explanation

An example of this phenomenon can be seen in Figure 5.8 in which a Full DSC UNet makes segmentation predictions on an example image at an early and late stage of training. Recall from Figures 5.6 (a) and (c) that the Full DSC UNet has a significant increase in its cross entropy loss value later in training, without significant losses in $mIoU$. Figure 5.8 shows that for a Full DSC UNet trained for only 30 epochs, a $mIoU$ and cross entropy loss of 0.5 and 1.39 are achieved respectively, while when the network is trained for 70 epochs, a $mIoU$ and cross entropy loss of 0.5 and 2.32 are achieved respectively. This example therefore reflects the trends observed in Figure 5.6 for a Full DSC UNet on the entire dataset. If we examine the computation of the cross entropy loss for only frazil ice, we first arrive at the softmax predictions at 30 and 70 epochs in Figures 5.8 (d) and (j) respectively. In calculating the cross entropy loss for only a specific class (Equation 5.1), pixel predictions are only considered in locations where the ground truth is indeed the class of interest. If we observe the frazil softmax predictions in locations where frazil is in the ground truth, we arrive at Figures 5.8 (e) and (k) for the models trained for 30 and 70 epochs respectively. Creating a histogram of pixel values in Figures 5.8 (e) and (k), we arrive at Figures 5.8 (f) and (l), which show the number of pixels for each value in the range $[0, 1]$. We see in Figure 5.8 (f) that although the majority of pixels are near a value of one or zero, there are a significant number of pixels in the range $(0.1, 0.9)$ during this early stage of training. Later in training, Figure 5.8 (l) shows that a lot of these pixel values in the range $(0.1, 0.9)$

have changed to be much closer to zero or one. The cross entropy calculation takes the negative logarithm of the softmax pixel intensities, meaning that small values have a much larger effect than larger ones. As a result, the pixel values in Figure 5.8 (f) in the range $(0.1, 0.5)$ will have less of an affect on the final loss value than those pixels in the range $[0, 0.1]$. In contrast, the pixel values in Figure 5.8 (f) in the range $(0.5, 0.9)$ will have a similarly negligible affect on the loss value as the pixels in the range $[0.9, 1]$. Since the $IoU$ is calculated by a argmax function, if all the pixels in the range $(0.1, 0.5)$ theoretically moved to the range $[0, 0.1]$ with more training, and all the pixels in the range $(0.5, 0.9)$ moved to the range $[0.9, 1]$ with more training; the resulting $IoU$ values would be the same. Observing Figures 5.8 (c) and (i) we see that, although the $mIoU$ values are the same, the segmentation predictions are not exactly the same, showing that pixels in the ranges $(0.1, 0.5)$ and $(0.5, 0.9)$ did not strictly move to $[0, 0.1]$ and $[0.9, 1]$ as described in the theoretical explanation above.

Although this disconnect between $mIoU$ and cross entropy loss has not caused any negative effects over 80 epochs, over longer training we can expect the errors in the softmax output to leak into the argmax and therefore the visual quality of the predicted masks. Looking at Figure 5.6 (d), we see that the Small UNets experience this overfitting effect to a much smaller scale. This effect is almost completely non-existent in the Small DSC LBC UNet, allowing for more training to be done before early stopping is triggered. These results are consistent with beliefs that highly complex networks are prone to overfitting when limited data is available [88].

### 5.2.3 Generalization Ability

#### Snow Generalization

As this overfitting effect does not appear to visually degrade results over the measured epochs, the practical ability for these networks to generalize is still unclear. To get more clarity on this topic, and meaningfully test generalization ability as it relates to river ice, we can reference Table 5.2 which shows the results of the various networks trained on data with no snow, but evaluated on data with synthetic snow. We first notice that the general trend of Full UNets outperforming MobileNets, and Small UNets outperforming Full UNets to be consistent with experiments on non-snowy data (Table 5.1). The results of Table 5.2, however, differ from Table 5.1 as the best performing Small network is now the Small DSC LBC UNet and the best performing Full network is the Full DSC LBC UNet, the two of which perform virtually the same. In perhaps in a more realistic scenario, some training and testing images may have snow while others do not. Figure 5.9 shows a variety

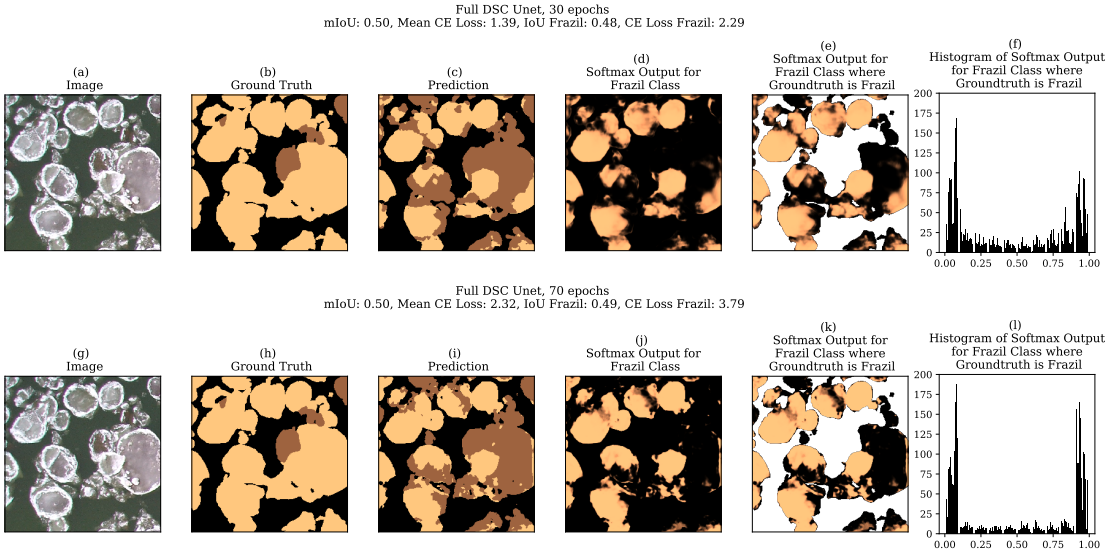



Figure 5.8: Comparison of predictions, metrics, and loss values of a Full DSC UNet on frazil ice at different stages of training. Sub-figures (a)-(f) correspond to results after 30 epochs of training while sub-figures (g)-(l) correspond to results after only 70 epochs of training. The $mIoU$ value at both stages are the same and the $IoU$ value for frazil ice at both training stages is virtually the same. In contrast, the mean cross entropy loss and frazil cross entropy loss are significantly higher at 70 epoch stage than the 30 epoch stage. Sub-figures (a) and (g) are an example image, (b) and (h) are the corresponding ground truth, (c) and (i) are the predictions at the 30 and 70 epoch stage respectively, (d) and (j) are the softmax output of each network for frazil ice, (e) and (k) are the are the softmax of each network for frazil ice only at locations of the image where frazil ice is in the ground truth, and finally (f) and (l) are the histogram of pixel values from (e) and (k) respectively. The pixel values of (e) are used in a cross entropy function result in a value of 2.29; the frazil portion of the mean cross entropy of 1.39 for the prediction (c). The pixel values of (k) used in a cross entropy function result in a value of 3.79; the frazil portion of the mean cross entropy of 2.32 for the prediction (i).

of scenarios where different amounts of snow are present in the training and testing sets and the associated $mIoU$ for three models of interest; the Full UNet, Small DSC LBC UNet, and MobileNetV3 (LR-ASPP). It can be seen that the results of MobileNetV3 (LR-ASPP) under-perform the Full UNet and are highly correlated. The Small DSC LBC UNet performs relatively similar to the Full UNet when the test set has a snow image proportion of $\frac{1}{3}$ or $\frac{2}{3}$; however, the Small DSC LBC UNet outperforms the Full UNet significantly in edge cases. These edge cases include high snow during training and no snow during testing, high snow during training and half snow during testing, no snow during training and high snow during testing, and high snow during both training and testing. These results suggest that the DSC LBC convolution mechanism improves a network's ability to generalize to and from snowy environments; an important feature for a network used in regions where snow is common.

Table 5.2: mIoU and mPA scores on the test set corrupted with snow noise. Models were trained on the training set without snow noise. Numbers in bold highlight the best scores while the Small DSC LBC UNet is in bold as it is the model of interest.

| **Model** | **mIoU** | **mPA** |
|---|---|---|
| Full UNet | 58.1 | 71.0 |
| Full DSC UNet | 57.6 | 72.0 |
| Full LBC UNet | 56.5 | 70.5 |
| **Full DSC LBC UNet** | **62.9** | **76.8** |
| Small UNet | 58.9 | 71.0 |
| Small DSC UNet | 59.1 | 71.1 |
| Small LBC UNet | 57.0 | 71.7 |
| **Small DSC LBC UNet** | 62.4 | 76.1 |
| MobileNetV3 (DeepLabV3) | 49.7 | 62.3 |
| MobileNetV3 (LR-ASPP) | 53.0 | 62.8 |

### Illumination Generalization

The other characteristic for which generalization ability was tested was the illumination of the image. Figure 5.10 shows a bar chart comparing the $mIoU$ of different train-test splits with varying illumination. Specifically, models trained and tested on the natural illumination variation of Alberta River Ice Segmentation Dataset can be compared to models trained on a darkened dataset and tested on a lightened dataset, as well as models trained on a lightened dataset and tested on a darkened dataset. In Figure 5.10 we see

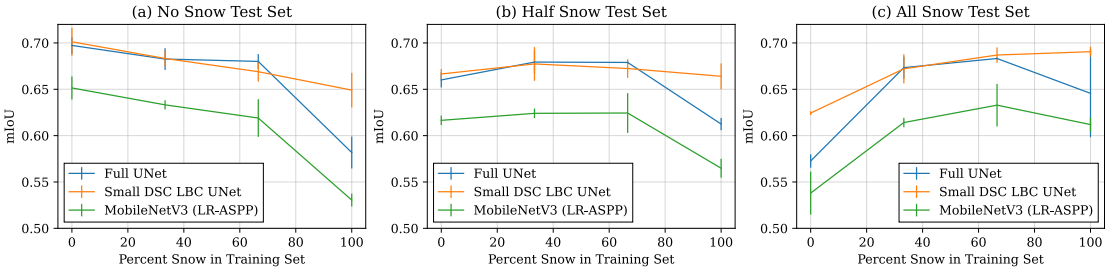



Figure 5.9: mIoU results for a Full UNet, Small DSC LBC UNet, and MobileNetV3 (LR-ASPP) for varying numbers of images with snow in the training and test sets. Training sets had either 0/30, 10/30, 20/30, or 30/30 of the images containing snow, while the test set had either 0/10 (a), 5/10 (b), or 10/10 (c) of the images containing snow. Three instances of each model were trained under each scenario and their mean value was used with the standard deviation shown by the error bar.

that among all three scenarios, the general trends of Small UNets outperforming Full UNets still persists. In a general sense we can also say that models trained on darker data do not generalize as well as models trained on lighter data as see by comparing the brown and tan bars in Figure 5.10 respectively. However, it is worth noting that models such as the Small UNet, Small DSC LBC UNet, and MobileNetV3 (LR-ASPP) do not exhibit this same trend and actually show that models trained on dark data generalize slightly better than those trained on light data. For these same three models we also observe that the $mIoU$ values are quite similar under all three scenarios for a given model. This is in contrast to the other models which show greater variability in $mIoU$ depending on if the training or testing data is light, dark, or mixed (original dataset). As a result we can say that the Small UNet, Small DSC LBC UNet, and MobileNetV3 (LR-ASPP) are more reliable in environments with varying illumination.

### 5.2.4 Performance-Latency Trade-Off

A significant factor affecting the latency of a model is the number of multiply-add operations and memory usage of the model. As seen in Table 5.1, a trade-off is made between performance, the number of Mult-Add operations and memory usage. The Small UNet has high performance, a high number of Mult-Adds and relatively high memory usage, while MobileNetV3 (LR-ASPP) has slightly lower metric performance but significantly less Mult-Adds and memory requirements. This introduces a performance-latency trade-off where a model can train in much less time but sacrifices some performance. The Small DSC LBC

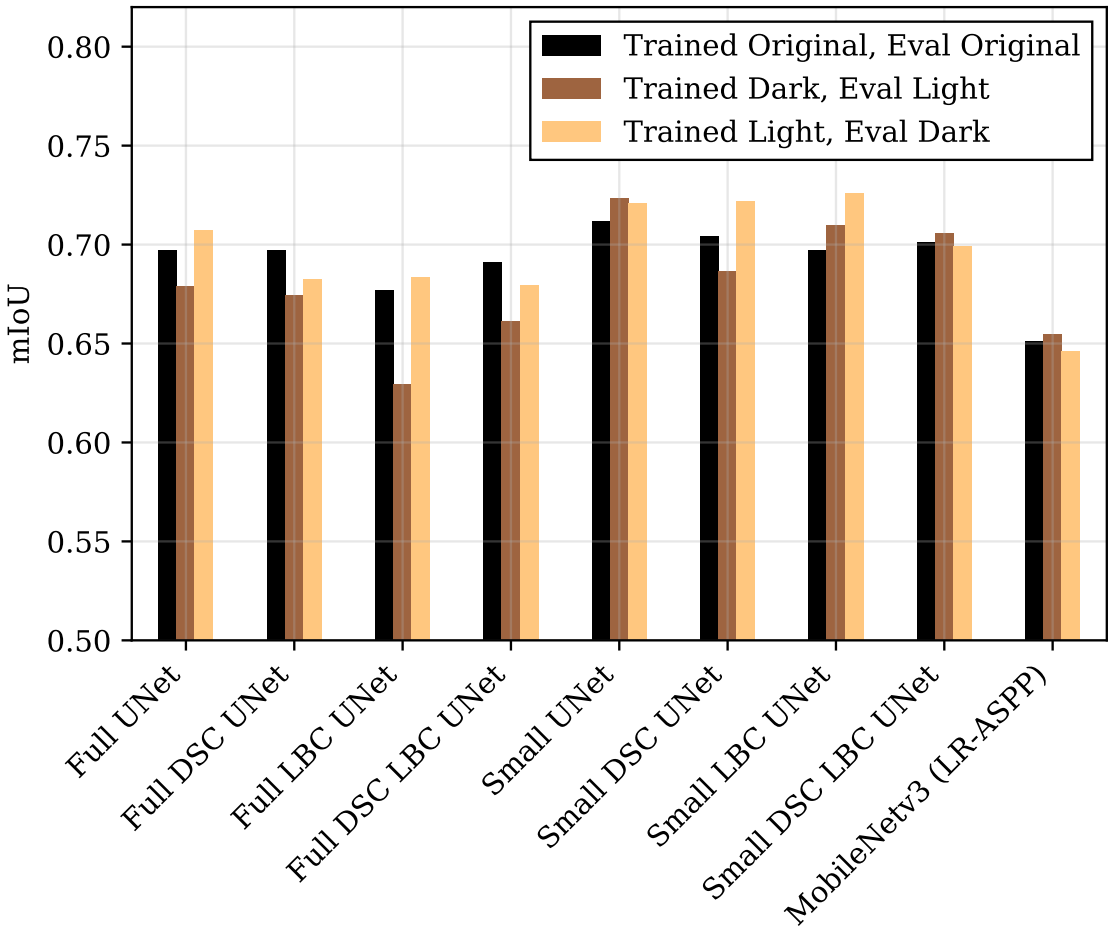



Figure 5.10: Bar chart comparing $mIoU$ values achieved by different models trained and tested on data sets with varying illumination levels. The black bars represent $mIoU$ values calculated from models trained and tested on images with natural illumination as captured in the Alberta River Ice Segmentation Dataset. The brown bars represent $mIoU$ values calculated from models trained on images synthetically darkened to a mean water grey-scale pixel intensity of 75, and tested on images synthetically lightened to a mean water grey-scale pixel intensity of 120. The tan bars represent $mIoU$ values calculated from models trained on images synthetically lightened to a mean water grey-scale pixel intensity of 120, and tested on images synthetically darkened to a mean water grey-scale pixel intensity of 75.

UNet can be considered a good compromise between the performance of the Small UNet and the resource requirements of MobileNetV3 (LR-ASPP).

Table 5.3 compares both inference and training time for the various models on a GPU and CPU. Inference time was measured by segmenting 10 $320 \times 320 \times 3$ images, while training time was measured over three epochs where 30 $320 \times 320 \times 3$ images were used per epoch. An initial observation reveals that the Small DSC LBC UNet had the fastest training and inference on the GPU, while MobileNetV3 (LR-ASPP) had the fastest training and inference on the CPU. The low number of trainable parameters of the Small DSC LBC UNet may have attributed to its success on the GPU, while the CPU specific modifications made to MobileNetV3 (LR-ASPP) can be attributed to its success during CPU training and inference. These CPU-centric modifications, including a hardware-aware network architecture search which exists for MobileNetV3s but not the UNets [36]. We can convert the results of Table 5.3 to a frames-per-second (fps) measurement to understand how fast inference can happen on real-time video. As the inference times were calculated for 10 $320 \times 320 \times 3$ images, we can say that the Small DSC LBC UNet has a fps of 22.5 and 6.5 on a Nvidia GeForce GTX 1060 GPU and AMD Ryzen 5 2600 Six-Core CPU respectively. On the same hardware and for the same image size, MobileNetV3 (LR-ASPP) has a fps of 21.2 and 12.3.

Although it takes MobileNetV3 (LR-ASPP) less time than the Small DSC LBC UNet to complete a training epoch on a CPU, we saw from Figure 5.6 (b) that the Small DSC LBC UNet requires very few training iterations to achieve good performance. We can therefore frame the idea of run-time slightly differently to make a more fair and practical comparison. Rather than looking at the time to complete $n$ number of training epochs, we can look at the time required to achieve some threshold of performance. Recall from Figure 5.6 that the training curves are smoothed using an exponential moving average with a smoothing factor of 0.95. When looking at the time to reach a $mIoU$ threshold, we also look at a smooth curve as to not stop at an unsustainably high $mIoU$ value that is primarily caused by noise.

Table 5.4 shows the time for the Small DSC LBC UNet and MobileNetV3 (LR-ASPP) to reach various $mIoU$ thresholds. We see that the Small DSC LBC UNet outperforms MobileNetV3 (LR-ASPP) for all $mIoU$ thresholds. Notably, the largest discrepancy between runtimes occurs at a low ($\leq$48%) $mIoU$ and a high ($\geq$64%) $mIoU$, showing that Small DSC LBC UNet is most effective when aiming for either very quick results or very high performing results. If we consider a $mIoU$ of 56% to be satisfactory and a $mIoU$ of 64% to be good, then the Small DSC LBC UNet achieves satisfactory results 36% faster than MobileNetV3 (LR-ASPP), and achieves good results 91% faster than MobileNetV3 (LR-ASPP) when trained on a CPU.

Table 5.3: Training and inference runtime on a Nvidia GeForce GTX 1060 GPU and AMD Ryzen 5 2600 Six-Core CPU. Training time was measured over three epochs, with 30 $320 \times 320 \times 3$ images per epoch. Inference time was measured for 10 $320 \times 320 \times 3$ images. Bold numbers indicate the fastest runtime, while the Small DSC LBC UNet is in bold as it is the model of interest.

| **Model** | **GPU Train (s)** | **GPU Inference ($\times 10^{-1}$s)** | **CPU Train (s)** | **CPU Inference (s)** |
|---|---|---|---|---|
| Full UNet | 8.50 | 4.61 | 125.37 | 7.74 |
| Full DSC UNet | 8.78 | 4.59 | 65.96 | 4.34 |
| Full LBC UNet | 8.06 | 4.58 | 97.84 | 8.47 |
| Full DSC LBC UNet | 5.60 | 4.55 | 40.41 | 3.34 |
| Small UNet | 4.78 | 4.57 | 28.08 | 2.07 |
| Small DSC UNet | 5.00 | 4.56 | 24.26 | 1.78 |
| Small LBC UNet | 4.65 | 4.53 | 22.86 | 2.13 |
| **Small DSC LBC UNet** | **4.60** | **4.45** | 15.58 | 1.53 |
| MobileNetV3 (DeepLabV3) | 6.78 | 4.79 | 27.51 | 1.25 |
| MobileNetV3 (LR-ASPP) | 5.79 | 4.72 | **14.25** | **0.81** |

## 5.3 Summary

In this chapter we balance the trade-off between performance and latency for the task of river ice segmentation on the Peace River and North Saskatchewan River. We introduce a new convolutional block, the DSC LBC block, and use it to construct a shallow UNet style architecture. The DSC LBC block combines the benefits of depthwise separable convolutions and local binary convolutions to minimize both the number of operations and the number of trainable parameters in a convolution. We find that the DSC LBC convolution adds efficiency and improves a network's ability to generalize to other domains such as a snowy environment or areas with illumination variation. Our novel architecture has performance on par with UNet, but with 99.9% less trainable parameters, 99% less multiply-add operations, and 69.8% less memory usage, resulting in significantly faster training and inference. When compared to state-of-the-art efficient networks such as LR-ASPP MobleNetV3, our architecture not only achieves a $mIoU$ value that is 7.7% higher over extended training, it can achieve a satisfactory results ($mIoU$ of 56) 36% faster than LR-ASPP MobleNetV3, and a good result ($mIoU$ of 64) 91% faster than LR-ASPP MobleNetV3 on a CPU, and even more impressive results on a GPU. This success attributed to the shallow nature of

Table 5.4: Time in seconds for Small DSC LBC UNet and MobineNetV3 (LR-ASPP) to reach various mIoU thresholds on an AMD Ryzen 5 2600 Six-Core CPU.

| **mIoU Threshold** | **Small DSC LBC UNet Time** | **MobileNetV3 (LR-ASPP) Time** |
|---|---|---|
| 48 | **2.16s** | 18.05s |
| 52 | **15.57s** | 24.70s |
| 56 | **20.24s** | 31.83s |
| 60 | **27.51s** | 43.70s |
| 64 | **43.08s** | 481.17s |

the architecture, the lightweight nature of DSC LBC convolutions, and the ability for the DSC LBC convolution to focus on both texture and pixel intensity, allowing for better performance early in the training process. These results give promise to real time river ice segmentation in remote regions where limited hardware and computation power is available and winter weather conditions can affect image quality.

# Chapter 6

# Weakly Supervised River Ice Segmentation

## 6.1 Methodology

### 6.1.1 Point Labeling

The Alberta River Ice Segmentation Dataset contains image labels in the form of pixel-wise masks [86]. When experimenting with weakly supervised segmentation, a labelling method can be selected that favours the shape of the ice pans. Figure 2.5 shows that point level supervision, used with an objectness prior, often results in segmentation predictions that appear rounded with smooth edges. For objects that are indeed round such as the sheep in Figure 2.5, the difference between the *Point-level + objectness* prediction and the *Full supervision* prediction is quite minor compared to the other objects that have more complex shapes. As the ice pans in the Alberta River Ice Segmentation Dataset are often quite rounded, point labels can be efficient to collect while potentially offering better segmentation performance than can be expected when point labelling is used with other common object based datasets.

In order to re-label the Alberta River Ice Segmentation Dataset with point labels, the software package CVAT can be used [82]. In practice, annotating images with point labels simply involves clicking on the centres of objects to add a point, and selecting the class associated with the point depending on the object class. When determining where points should be placed on the ice pans, some liberty was taken when determining if two or more adjacent ice pans had been frozen together resulting in a single point annotation, or if

they were separate resulting in multiple point annotations. This is somewhat important since if LC-FCN commonly predicts multiple points for a single ice pan, there will be more confusion down the line for the algorithm that combines point predictions with object proposals to create the final segmentation. A general guideline was established that states that if a single upturned, or bright, edge encapsulates one or more ice pans, all the pans within the upturned edge can be considered a single ice pan. This means that multiple ice pans that have frozen together and share a common upturned edge are considered one ice pan and therefore receive one point label as seen in the red dotted oval of Figure 6.1. In terms of determining the class of the point labels, in order to maintain consistency with the Alberta River Ice Segmentation Dataset, no liberty was taken in determining the class of the labels. Whichever class that the point label overlaps with in the pixel-wise ground truth is the class of the point label, as exemplified by the green dotted oval of Figure 6.1.

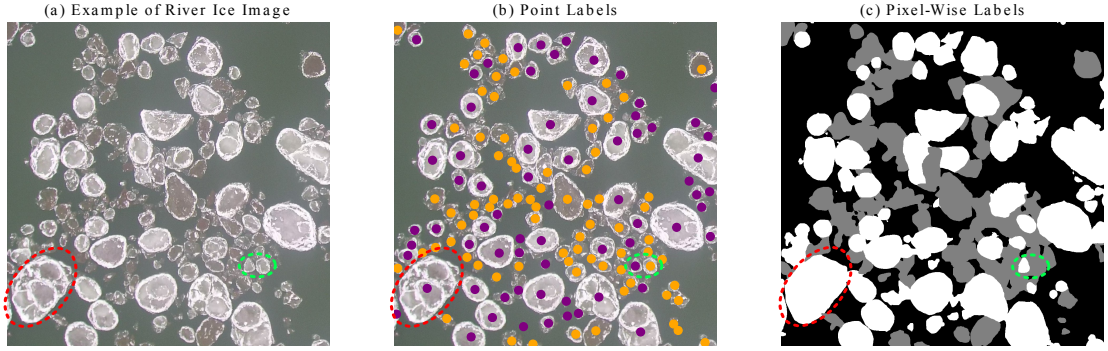



Figure 6.1: Example of how point labels and pixel-wise masks appear for a given river ice image. In (b), purple points represent frazil ice, while orange points represent anchor ice. In (c), frazil ice is in white, while anchor ice is in gray. Note that when multiple frazil pans are frozen together, this is considered a single point as seen by the red dotted oval. Also note that the classes of the point labels are determined by the pixel-wise ground truth as seen in the ambiguous case denoted by the green dotted oval.

To be able to compare the results of a fully supervised model and a weakly supervised model under labelling time constraints, the labelling time to collect both the fully and weakly supervised labels must be measured. Measuring the time to collect the point labels is straight forward since a timer can simply be run during the collection of these labels. Measuring the time to collect the fully supervised pixel-wise masks is less straightforward since they were already collected by the creators of the Alberta River Ice Segmentation Dataset and are very time consuming to re-collect. As a compromise, we propose to select five diversified images from the Alberta River Ice Segmentation Dataset as seen in Figure

6.2, and re-label them while recording the time. The average of these five times can be considered the time to label a single image and can be used to create an estimated labelling time for the entire dataset.

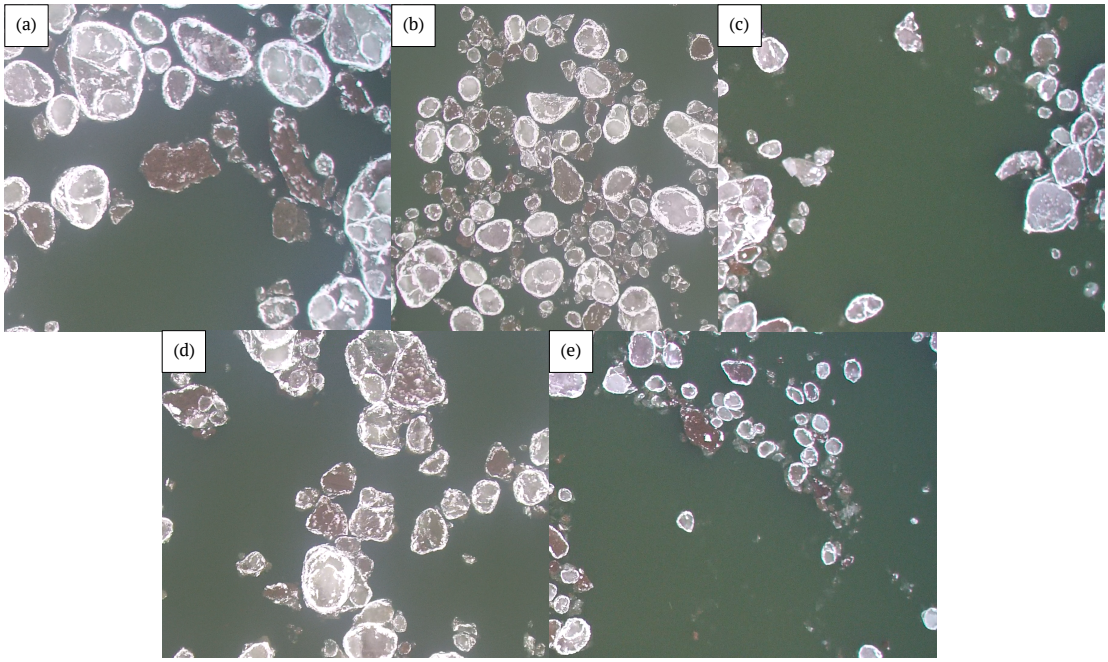



Figure 6.2: Five images selected for timed fully supervised pixel-wise labelling. These five images are considered diverse in the scope of the Alberta River Ice Segmentation Dataset as they vary in scale in terms of proximity to the ice pans. The images also vary in the proportion of water and ice that cover the image.

### 6.1.2 Combining Points and Proposals

In order to predict a segmentation mask from point labels, we will take inspiration from the WISE network, which achieved state of the art results for instance segmentation using point labels. Following the methodology of the WISE network, the point predictions from LC-FCN must be combined with the object proposals from an algorithm such as DeepMask in order to arrive at a final segmentation mask [56]. As DeepMask requires full pixel-wise labels for training and we are limiting ourselves to point labels for the Alberta River Ice Segmentation Dataset, we choose to use a pre-trained DeepMask weights trained on the

MS COCO dataset [58, 72]. Recall that as DeepMask is class agnostic, it does not need to be trained on the dataset of interest in order to be implemented on that dataset [69]. LC-FCN on the other hand was trained on the point labels of Alberta River Ice Segmentation Dataset. The same 30-10-10 train-validation-test split as well as the same stopping criteria as Section 5.1.2 was employed to train the LC-FCN. Again similar to previous experiments, both DeepMask and LC-FCN were implemented in PyTorch [68] and used with a Nvidia GeForce GTX 1060 GPU.

The authors of WISE experimented with various methods of combining points and proposals as we will as well. The following three methods can be used for combining point predictions and object proposal predictions.

#### LC-FCN + DeepMask: Max Proposal Score

As mentioned in Section 2.5.2, the DeepMask object proposal algorithm outputs a confidence score associated with each object proposal. One of the simplest methods for choosing which proposal to use can simply be choosing the object proposal with the highest score that aligns with a predicted point. This can be considered a greedy algorithm. A possible drawback of this method is that DeepMask may have a high scoring proposal that overlaps multiple ice pans, reducing the detail in the final segmentation. To partly combat this, we choose to experiment with a single constraint limiting the size of the object proposal being used. We decide that a safe threshold requires the number of pixels in any proposal to be no greater than $\frac{1}{8}$th of the total number of pixels in the image. A visual inspection of the Alberta River Ice Segmentation Dataset shows that there are no ice pans larger than this threshold and therefore any DeepMask predictions larger than this can be considered inaccurate.

#### LC-FCN + DeepMask: Min Points in Proposal

As seen in some example images of the Alberta River Ice Segmentation Dataset, there can be multiple anchor and frazil ice pans within a small area. To further discourage the greedy algorithm from selecting a single proposal that overlaps other ice pans of the opposite class, we experiment with optimizing for the number of predicted points within a proposal. Specifically, when trying to match a point prediction from LC-FCN for a given class with a proposal, we choose the overlapping DeepMask proposal with the highest score that also overlaps with the least number of point predictions for the opposite class. For example, for a frazil point prediction we let the tuple $((i), (ii))$ represent $(i)$ the proposal score for an overlapping DeepMask proposal and $(ii)$ the number of anchor point predictions also

overlapping with the proposal. For the proposal tuples $(0.99, 4), (0.96, 2), (0.86, 1), (0.81, 1)$, this method would select the third proposal as it has the highest score of 86% while having only one anchor point prediction within; the lowest possible of all the proposals. Although the first proposal has a high score of 99%, it overlaps four anchor ice point predictions. If this proposal is selected, as it would be by the greedy algorithm, these four potential anchor ice pans would be miss-classified, resulting in a poor final segmentation.

#### Embedding Network

As described in Section 2.5.3, the authors of WISE developed E-Net to intelligently select which object proposal to use for a give point prediction of LC-FCN[1]. E-Net is trained using a similarity loss that aims to score pixels from the same object instance with a high similarly. This is intuitive if object instances are relatively distinct; however, as the various instances of frazil pans appear very similar, and the various instances of anchor pans appear very similar, this strategy may struggle to discriminate between instances. The training process for E-Net is also quite complex, involving randomly selecting proposals and pixel pairs, and evaluating E-Net output against the proposals. Due to this complexity and the fact that ice pan instances of a given class a quite similar, we choose to experiment with using the two alternatives described above as an alternative to E-Net.

### 6.1.3 Post Processing

#### LC-FCN + DeepMask: Blob Post Processing

A simple post processing technique can be employed when DeepMask fails to generate a proposal at a location where LC-FCN predicted a point. If we make the assumption that the LC-FCN point prediction is accurate, then by adding some sort of object proposal at the location of the point prediction, we can hopefully improve the final segmentation mask. Recall from Section 2.5.3 that LC-FCN uses *blob* predictions from an FCN to generate its point predictions. Although these *blobs* are not expected to accurately align with the object boundary, they can offer some more spatial information than a simple point, and therefore could be added as a proposal at locations where DeepMask did not find an object.

[1]Recall that LC-FCN is referred to as L-Net in the WISE paper.

#### LC-FCN + DeepMask: CRF Post Processing

A noted area of struggle for DeepMask lies in its ability to identify precise object boundaries [69]. As mentioned in Section 2.4.1, fully connected pairwise CRFs can refine coarse segmentation outputs. In order to improve the boundaries of the segmentation constructed from DeepMask proposals, we experiment with a CRF to refine edges after LC-FCN blobs are added. When applying the CRF, the $w^{(1)}$, $\theta_\alpha$, and $\theta_\beta$ arguments from Equation 2.24 are adjusted manually to better refine the segmentation boundaries while attempting to keep them smooth and the interiors of the proposals continuous.

## 6.2 Results

### 6.2.1 Labelling Time

As mentioned in Section 6.1.1, two labelling methods were timed; the point labelling of the entire Alberta River Ice Segmentation Dataset, and the full pixel-wise labelling of five example images listed (a)-(e) in Figure 6.2. To fully label the Alberta River Ice Segmentation Dataset with point annotations it look 65 minutes and 50 seconds, or 3,950 seconds. To fully label the same data with full pixel-wise annotations it was estimated to take 95,350 seconds, or 26 hours, 29 minutes, and 10 seconds, based on the calculations of Table 6.1.

As seen in Table 6.1, it takes on average 1,907 seconds to provide full pixel-wise annotations to an image from the Alberta River Ice Segmentation Dataset. This means, in the time one could fully annotate the dataset with point labels (3,950 seconds), one could only provide approximately two full pixel-wise annotations. As a result, when comparing results in Section 6.2.4 with a *limited annotation budget*, the fully supervised results will be generated using only two training images. To mitigate the bias of this extremely small training set, 15 different pairs of images will be used to train a given model, and their scores on the test set will be averaged to arrive at a final metric.

### 6.2.2 LC-FCN Results

LC-FCN was originally proposed as a counting network to find the total number of objects in an image, and not a precursor network for weakly supervised segmentation. Due to the nature of our weakly supervised segmentation task, it is difficult to quantitatively assess the results. Recall that LC-FCN returns blobs and points. The points are determined by

Table 6.1: Full pixel-wise labelling times in seconds for the five images in Figure 6.2 labelled (a)-(e). *Single Image Mean* is the mean labelling time for images (a)-(e), while *Total Dataset Estimate* is the *Single Image Mean* multiplied by 50 as there are 50 images in the Alberta River Ice Segmentation Dataset.

| **Image** | **Time (seconds)** |
|---|---|
| (a) | 1,801s |
| (b) | 2,602s |
| (c) | 1,434s |
| (d) | 1,921s |
| (e) | 1,777s |
| Single Image Mean | 1,907s |
| Total Dataset Estimate (50 imgs) | 95,350s |

the location of highest probability on each blob, as the blobs are outputs of a softmax pixel-wise map. We desire the point output in order to later combine with the object proposals of DeepMask. However, we do not necessarily care if the output points align exactly with the ground-truth points since a point can be shifted slightly and still align with the same object proposal. As a result we will analyze the results of LC-FCN qualitatively.

Example results of LC-FCN on a test set image can be seen in Figure 6.3. Though there is a lot happening in Figure 6.3, LC-FCN appears to have learnt the difference between frazil and anchor ice somewhat. For example, the cluster of anchor ice pans in the bottom right corner of Figure 6.3 (a) elicit a strong response from the anchor ice blobs in Figure 6.3 (b) and a weak or non-existent response from the frazil blobs in Figure 6.3 (d). In general there appears to be a recall rate that is quite high for both classes, meaning that if there is indeed a ground truth point on an ice pan for a given class, it is very likely that there is a blob prediction on that ice pan for that class.

A downside of this high recall is that there are also many false positives. This means that for an ice pan of a given class, LC-FCN may predict blobs of the opposite class on that ice pan. This is exemplified by the ice pans and predictions in the the pink dashed oval of Figure 6.3. In Figures 6.3 (a) and (c) we can see that there are two anchor ice pans in the top-right part of the pink oval, and one frazil ice pan in the bottom-left part of the pink oval. In Figure 6.3 (b) LC-FCN predicts all the ice pans in the pink oval to be anchor ice, and in Figure 6.3 (d) LC-FCN predicts all the ice pans in the pink oval to be frazil ice. With a closer look, however, it appears that the frazil blob in Figure 6.3 (d) on the

pan that is indeed frazil ice is larger than the anchor blob in Figure 6.3 (b) on the same pan. Similarly, the the anchor blob in Figure 6.3 (b) on the pans that are indeed anchor ice is larger than the frazil blob in Figure 6.3 (d) on the same pans. This hints that for ice pans with point predictions from both classes, the size of the blob for each class can be an indicator of the true class. Looking at other examples of contradicting predictions in Figure 6.3 shows a similar pattern.

Finally, there are predictions that LC-FCN makes that are incorrect according to the ground truth. For example, the red dashed oval in Figure 6.3 shows a cluster of ice pans that are labelled as frazil ice by the ground truth, yet LC-FCN predicts as anchor ice. This example is a curious case as it is arguable that these ice pans may indeed be anchor ice based on their dark colour as well as size. Observing the other anchor ice pans in Figure 6.3, the frazil pans in the red oval appear much more like anchor ice than frazil ice. Recall from Section 6.1.1 that the classes of the point labels were chosen strictly by observing the original pixel-wise labels of the Alberta River Ice Segmentation Dataset which are subject to errors. Although it is up for debate, it is possible that LC-FCN is still making the *correct* choice in this situation based on the true classes of the ice in these images and not the ground-truth labels.

Though LC-FCN is not perfect and displays some visual errors relating to high recall and false positives, based on the nature of the false positives and the size of the blobs, it is evident that LC-FCN can offer valuable information when selecting object proposals from DeepMask.

### 6.2.3 DeepMask Results

An example of DeepMask object proposals can be seen in Figure 6.4. The top 10 proposals with the highest score from DeepMask are shown in Figure 6.4 (b). An observation from Figure 6.4 (b) is that there is a large proposal that covers nearly the entire image (yellow dashed outline). This proposal clearly does not correspond to an object in (a) and can therefore be considered a miss-prediction. Recall from Section 6.1.2 that proposals will only be included in a final segmentation prediction if the number of pixels in a proposal to be no greater than $\frac{1}{8}$th of the total number of pixels in the image. This would result in the miss-prediction not being included in a final segmentation.

It is also worth noting that multiple proposals are generated for the same object or set of objects. For example, one proposal may encompass two ice pans, yet there could be two more proposals that cover each of the two ice pan individually (green dashed oval). Finally, we can observe that DeepMask occasionally fails to generate proposals for some

of the ice pans (red dashed circle). This can cause problems when there is no proposals to draw from for a given ice pan that may have a point prediction.

### 6.2.4 Segmentation Results

Table 6.2 shows a comparison of the $mIoU$ and $mPA$ for various models with a limited training budget. The limited budget constrains a training set to two images with full pixel-wise ground truth masks, or all the training images with point labels. All of the listed *LC-FCN + DeepMask* models use the processing techniques of the previously listed model. For example, the model with *Blob Post Processing* also uses the *Min Points in Proposal* method for selecting which DeepMask proposal to use (as described in Section 6.1.2). Similarly, the *CRF Post Processing* also uses the blobs from the results of *Blob Post Processing*. We see that in general, each additional step in refining the *LC-FCN + DeepMask* results ends up improving the $mIoU$ and $mPA$. As compared to the UNet with a limited training budget, the *LC-FCN + DeepMask: CRF Post Processing* model improves $mIoU$ and $mPA$ by 14.6% and 7.9% respectively. This is a large increase in $mIoU$ which suggests that the predictions of *LC-FCN + DeepMask: CRF Post Processing* have a much better appearance that is more spatially accurate. As compared to a UNet with a full labelling budget, the *LC-FCN + DeepMask: CRF Post Processing* model results in a decrease of $mIoU$ and $mPA$ of only 6.3% and 7.1% respectively. Considering that the UNet with a full budget requires approximately 23× more time to label, these results are quite encouraging.

An observation regarding the anchor ice class accuracy is that there is actually a decrease in accuracy from 65.1 for *Blob Post Processing* to 63.1 for *CRF Post Processing*. Interestingly, the $IoU$ for the anchor ice class actually increases between the two models. This can likely be explained by observing the prediction results in Figure 6.5.

Figure 6.5 shows results of a UNet with a full labelling budget, a UNet with a limited labelling budget, and various versions of the *LC-FCN + DeepMask* models. The first observation that can be made is that the UNet with a limited labelling budget in Figure 6.5 (d) struggles heavily with the anchor ice class. While the frazil ice pans are segmented relatively well, the network predicts anchor ice to be surrounding many of the frazil pans where there is no anchor ice in reality. Furthermore, in locations where there is indeed anchor ice, the network is unable to locate the edges of the pans and the anchor predictions often overflow into the water class. This observation is also reflected in the pixel accuracy and $IoU$ for the anchor class in Table 6.2 for the *UNet Limited Budget* model. The anchor class scores are very low, while the frazil class scores are not nearly as low.

Next, observations in Figure 6.5 can be made with regards to each additional processing step of the *LC-FCN + DeepMask* results. First, we can observe changes when the method of selecting proposals is changed from simply selecting the proposal with the highest DeepMask score, to selecting the proposal with the highest DeepMask score that also contains the least LC-FCN point predictions of the opposite class. This can be exemplified by the green dotted oval in Figures 6.5 (e) and (f). In (e), there is a single DeepMask proposal that covers both the anchor and the frazil pans within the green oval. This is due to the fact that this proposal has a very high DeepMask score as shown by the green oval in Figure 6.4 (b). As a result, the *Max Proposal Score* algorithm uses this single proposal to classify both pans as frazil ice. In (f) however, since the LC-FCN predicts anchor points on the anchor pan and frazil points on the frazil pan, the *Min Points Proposal* method realizes that the single frazil proposal used in (e) would overlap one or more anchor points and therefore elects to use two smaller proposals with slightly lower DeepMask scores that contain fewer LC-FCN points of the opposite class.

Continuing to look at Figure 6.5 (f), there are ice pans that do not have an associated prediction such as the frazil ice pan in the blue dotted oval. The reason there is no prediction at this location is because DeepMask failed to generate a proposal for that ice pan. The *Blob Post Processing* method in (g) improves slightly upon (f) by using the blob from LC-FCN as a proposal. As seen in the blue dotten oval in (g), the blob does not border the ice pan perfectly, however is provides more spatial context than what was present in (f).

Finally, comparing the results of the CRF between Figures 6.5 (g) and (h) we can see why the $IoU$ increased but the pixel accuracy decreased as previously mentioned. Looking at the pink dotted circle, we see an example of the benefits of the CRF where a false anchor ice pan prediction over water is eliminated. Visually we can see that this is an improvement which is reflected by the improved $IoU$. However, as the pixel accuracy cannot take false positives into account, and there are still some anchor pans with false negatives resulting in an under-prediction in the number of anchor ice pixels, the removed anchor ice lowers the pixel accuracy since the total pixels labelled anchor is decreased even lower. This decrease in pixel accuracy therefore does not indicate a shortcoming of the model but rather a shortcoming of the metric as we can see that the CRF visually improves the outcome of the final segmentation prediction.

## 6.3 Summary

In this chapter we explore the task of segmentation on the Alberta River Ice Segmentation Dataset using only point labels. We find that annotating the dataset with point labels is 95.9% faster than annotating with pixel-wise masks. Practically, this means that in the time it takes to annotate the entire Alberta River Ice Segmentation Dataset with point labels, one would only be able to fully annotate approximately two images with pixel-wise masks. In order to arrive at a segmentation mask using only point labels, we combine the results of a counting network that uses point labelling called LC-FCN, and a class agnostic object proposal network called DeepMask. Both models individually show satisfactory qualitative results. The LC-FCN for the most part is able to discriminate between frazil and anchor ice; however, it has high recall and can often predict a single ice pan to be from both classes. DeepMask is able to predict an object mask for nearly all ice pans in the image, though the edges are sometimes not sharp and occasionally one object proposal encapsulates multiple ice pans of different classes. We experiment with different methods of combining the point predictions from LC-FCN and DeepMask. As DeepMask outputs a score indicating the likelihood of a proposal being an object, we first try selecting the proposal with the highest score for each point prediction to build our segmentation mask. We find that minimizing the number of point predictions of the opposite class within a proposal is a better criteria for selecting a proposal for a given point prediction as this avoids larger proposals encapsulating multiple ice pans of different classes. We also find that two post processing techniques improve results. First, we include *blobs*, or intermediate layer results, from LC-FCN as an object proposal when DeepMask does not generate a proposal at a location of a point prediction. Second, we use a CRF to refine the edges of the object proposals. In the end, as compared to a UNet with the same labelling budget, we achieve a $mIoU$ score that is 14.6% higher. Compared to a UNet with a full labelling budget where labels require approximately 23× more time to collect, our method achieves a $mIoU$ that is only 6.3% lower. This is encouraging as it shows that acceptable results can be achieved when very limited time is available to label river ice data.

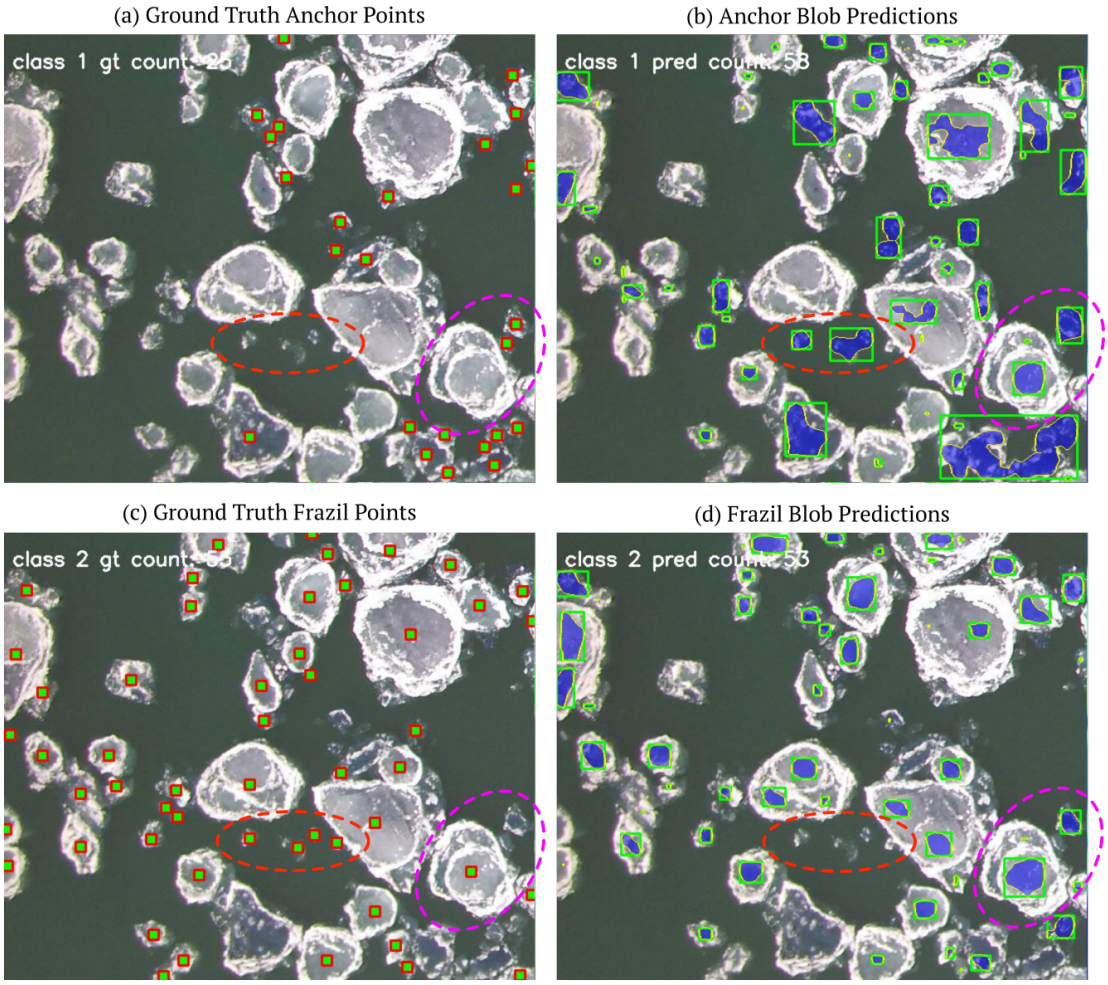



Figure 6.3: LC-FCN results on a test image. Sub-figures (a) and (c) show the ground truth point labels for the anchor and frazil classes respectively. Sub-figures (b) and (d) show the blob predictions of LC-FCN where each blob results in a point prediction. The point prediction for each blob is located at the point of maximum probability in the FCN softmax output. The ice pans in the red dashed oval show a disagreement between the ground truth and LC-FCN. The ice pans in the pink dashed line are predicted to be both frazil and anchor by LC-FCN.

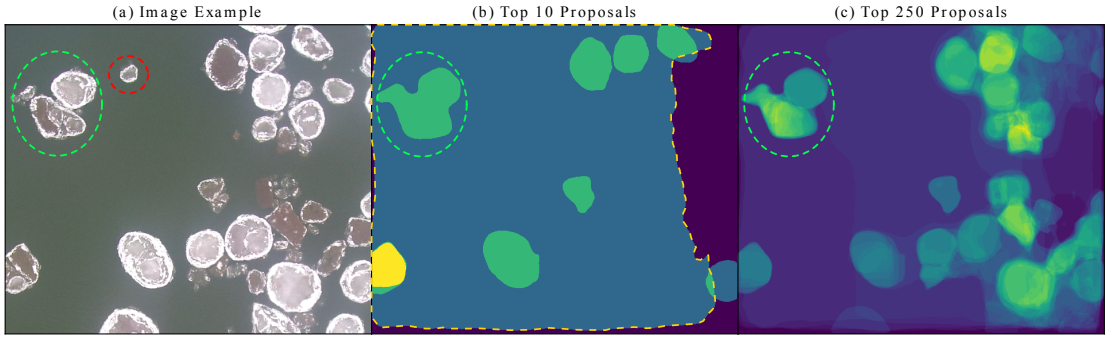



Figure 6.4: Example of DeepMask object proposals on an image from the test set (a). (b) Shows the 10 proposals with the highest DeepMask score, while (c) shows the 250 proposals with the highest DeepMask score. Note that certain regions in (b) and (c) are brighter as a result of overlapping proposals. The red dashed circle shows an ice pan that was not recognized by DeepMask. The green dashed oval shows ice pans that have multiple proposal predictions. Yellow outline shows a miss-predicted proposal.

Table 6.2: Mean IoU and Mean Pixel Accuracy for various models tested on the Alberta River Ice Segmentation Dataset. The scores for *UNet Limited Budget* were calculated by training 15 models, each with a unique two-image training set, and averaging their 15 scores. Recall from Section 6.2.1 that in the time it takes to annotate two images with full pixel-wise masks, one can annotate the entire dataset using point labels. All *LC-FCN + DeepMask* models were trained with point labels only, while the *UNet Full Budget* was trained with the all the pixel-wise ground truth masks of the Alberta River Ice Segmentation Dataset. All metric scores were calculated using the entire 10 image test set with original pixel-wise ground truth masks. Numbers in bold show the highest scores of all models trained with a limited budget. Note that these scores can be directly compared to the results of Table 5.1 in Chapter 5 due to the use of the same training and testing sets.

| **Method** | **mIoU**<br>Mean<br>(Water/Anchor/Frazil) | **mPA**<br>Mean<br>(Water/Anchor/Frazil) |
|---|---|---|
| UNet Limited Budget | 57.0<br>(82.1/36.9/52.0) | 71.2<br>(90.6/54.9/68.1) |
| LC-FCN + DeepMask:<br>Max Proposal Score | 59.8<br>(82.6/43.4/53.6) | 73.4<br>(90.1/64.5/65.1) |
| LC-FCN + DeepMask:<br>Min Points in Proposal | 61.9<br>(83.7/45.4/56.7) | 74.8<br>(92.1/63.1/69.3) |
| LC-FCN + DeepMask:<br>Blob Post Processing | 62.8<br>(84.1/46.8/57.0) | 75.7<br>(92.1/**65.1**/69.8) |
| LC-FCN + DeepMask:<br>CRF Post Processing | **65.3**<br>(**87.3**/**48.2**/**60.5**) | **76.8**<br>(**95.4**/63.1/**71.9**) |
| UNet Full Budget | 69.7<br>(92.3/53.2/63.6) | 82.7<br>(98.7/71.1/78.3) |

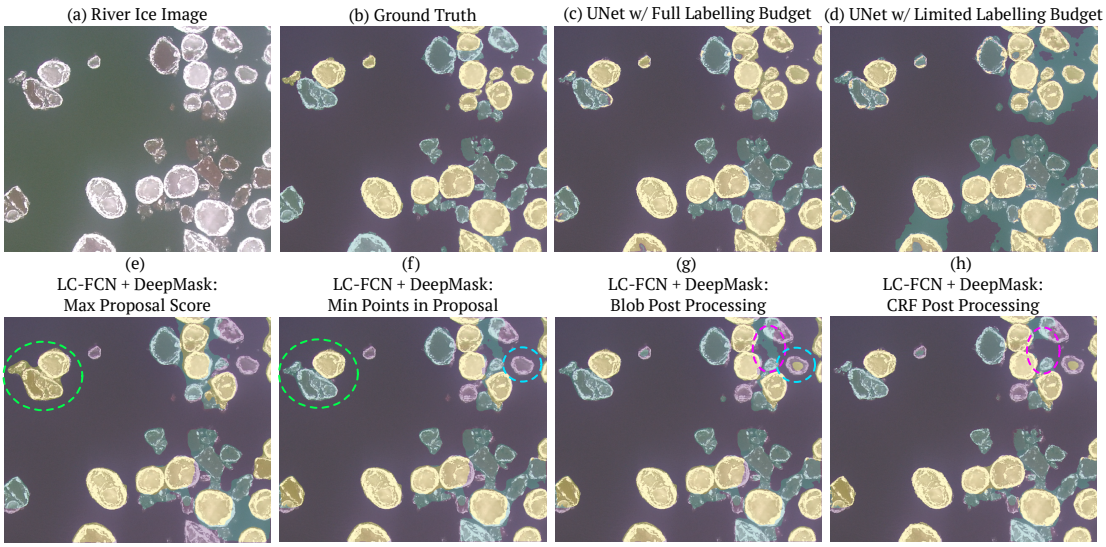



Figure 6.5: Final segmentation predictions of various models for an example image from the test set. The segmentation predictions overlay the original image to provide more detail regarding edge accuracy. Yellow predictions represent frazil ice, while green predictions represent anchor ice. The green dotted oval in (e) and (f) show a benefit of the *Min Points in Proposal* method. The blue dotted oval in (f) and (g) show a location that did not have a DeepMask proposal, however the blob from LC-FCN acted as a proposal. The pink dotted oval in (g) and (h) shows a benefit of the CRF in eliminating an incorrect prediction of anchor ice.

# Chapter 7

# Conclusions

In summary, this thesis aims to independently address two pain points of deep learning as it applies to the segmentation of river ice imagery. The first pain point is with regards to training and inference efficiency of deep learning models and their ability to be used with limited hardware. Typically as a deep learning model is optimized for efficiency, performance decreases. This is particularly relevant to river ice segmentation as data can often be collected in remote locations with limited computing power and on UAVs where operators wish to make accurate predictions in real time. To address this performance-latency trade off, a convolution method is developed that makes use of depthwise separable convolutions and local binary convolutions. This convolution method can replace standard convolutions in common network architectures to improve efficiency and generalization ability while maintaining high performance. When this convolution method is used in a shallow UNet-style architecture, performance is higher than a UNet with standard convolutions, while training and inference are similar if not more efficient than state-of-the art efficient networks such as MobileNetV3. It is also shown that the novel convolution method is robust to snow artifacts in images as well as illumination variation, both of which are common in river ice datasets. The second pain point addressed by this thesis is the expensive annotation budgets associated with training segmentation models. As the task of segmentation involves assigning a label to each pixel in an image, an annotator must therefore laboriously label each pixel in a set of training images. Leveraging the simple round shape of river ice pans and the rounded nature of class agnostic object proposals, a point labelling method is developed that uses object proposals and a tailored post processing technique. It is found that this method greatly outperforms fully supervised methods with the same labelling budget and only slightly under-performs fully supervised methods with with a labelling budget an order of magnitude larger. The results for both the novel convolution

method and the point labelling method leave questions regarding implementation in the field and generalization ability, both of which are addressed in Section 7.1 outlining future work.

## 7.1 Future Work

Both studies in this thesis leave questions regarding the generalization ability of these methods. With regards to the shallow network using the novel efficient convolutions, it was shown that this method works well on the Alberta River Ice Segmentation Dataset in varying environmental conditions, however it is unknown how this method will perform on other datasets both river ice an otherwise. Considering the large size and object variation in many machine learning benchmark datasets, intuition suggests that such a small model would not be as successful, though this remains to be tested. For larger, more diverse, datasets, it would be worth experimenting with the novel convolution method in larger network architectures, both for segmentation as well as other tasks such as image classification and object detection. In terms of generalization ability and the point labelling method, it remains to be seen how well this method performs when ice cover is more uniform and not separated into distinct ice pans. As areas with more uniform ice cover may not appear distinct objects to the object proposal network, the object proposal network may struggle to find ice class boundaries. Additionally, in areas with uniform ice cover, the *Blob Post Processing* described in Section 6.1.3 would likely be irrelevant as blobs from LC-FCN are small and discrete similar to ice pans and unlike uniform ice cover.

Another area worth exploring relates to the hardware used to test the efficient convolution method. As explained in Chapter 5, the shallow network was tested on both a GPU and CPU as CPUs are more common on simple work stations or hardware on-board UAVs. However, to get a more practical idea of how this architecture would perform for real-time training and inference on a UAV, it would be valuable to acquire either a UAV, or the computing hardware on-board a UAV and recreate the tests done in Section 5.1. Of course, different UAVs use different computing hardware, therefore it would also be valuable to compare performance on multiple popular UAV models.

# Letters of Copyright Permission

## 7.2 Figure 2.2 - Close Up Image of Anchor Ice

Figure 2.2 was taken from Kalke *et al.* [48] which was published in the 18th Workshop on the Hydraulics of Ice Covered Rivers as part of the Committee on River Ice Processes and the Environment (CRIPE) and the Canadian Geophysical Union - Hydrology Section (CGU-HS).

Figure 7.1 is email correspondence between Daniel Sola - the author of this thesis, April James - the Hydrology Section Secretary at the CGU, and Shawn Clark - the Chair of CRIPE.

## 7.3 Figure 2.8 and 2.7 - LC-FCN Methods and Results

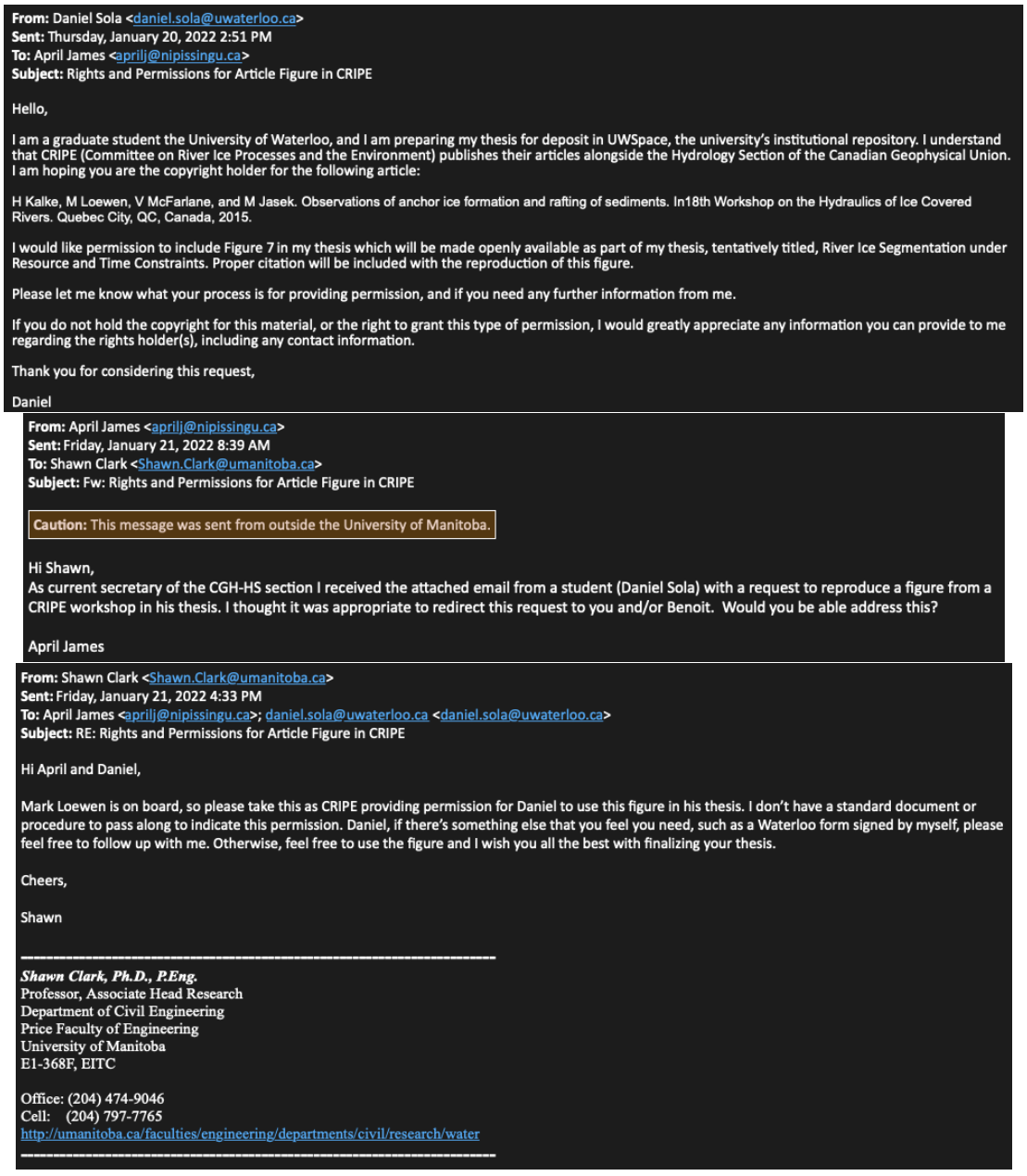
Figures 2.8 and 2.7 were taken from Laradji *et al.* [55] which was published in the Proceedings of the European Conference on Computer Vision (ECCV), part of Springer Nature.

Figure 7.2, reading left to right and up to down, is a the letter of permission to use Figures 2.8 and 2.7 from Springer Nature.

## 7.4 Figure 2.5 - Point Labelling Results

Figure 2.5 was taken from Bearman *et al.* [5] which was published in the Proceedings of the European Conference on Computer Vision (ECCV), part of Springer Nature.

Figure 7.3, reading left to right and up to down, is a the letter of permission to use Figure 2.5 from Springer Nature.



**From:** Daniel Sola <daniel.sola@uwaterloo.ca>
**Sent:** Thursday, January 20, 2022 2:51 PM
**To:** April James <aprilj@nipissingu.ca>
**Subject:** Rights and Permissions for Article Figure in CRIPE

Hello,

I am a graduate student the University of Waterloo, and I am preparing my thesis for deposit in UWSpace, the university's institutional repository. I understand that CRIPE (Committee on River Ice Processes and the Environment) publishes their articles alongside the Hydrology Section of the Canadian Geophysical Union. I am hoping you are the copyright holder for the following article:

H Kalke, M Loewen, V McFarlane, and M Jasek. Observations of anchor ice formation and rafting of sediments. In18th Workshop on the Hydraulics of Ice Covered Rivers. Quebec City, QC, Canada, 2015.

I would like permission to include Figure 7 in my thesis which will be made openly available as part of my thesis, tentatively titled, River Ice Segmentation under Resource and Time Constraints. Proper citation will be included with the reproduction of this figure.

Please let me know what your process is for providing permission, and if you need any further information from me.

If you do not hold the copyright for this material, or the right to grant this type of permission, I would greatly appreciate any information you can provide to me regarding the rights holder(s), including any contact information.

Thank you for considering this request,

Daniel

**From:** April James <aprilj@nipissingu.ca>
**Sent:** Friday, January 21, 2022 8:39 AM
**To:** Shawn Clark <Shawn.Clark@umanitoba.ca>
**Subject:** Fw: Rights and Permissions for Article Figure in CRIPE

**Caution:** This message was sent from outside the University of Manitoba.

Hi Shawn,
As current secretary of the CGH-HS section I received the attached email from a student (Daniel Sola) with a request to reproduce a figure from a CRIPE workshop in his thesis. I thought it was appropriate to redirect this request to you and/or Benoit. Would you be able address this?

April James

**From:** Shawn Clark <Shawn.Clark@umanitoba.ca>
**Sent:** Friday, January 21, 2022 4:33 PM
**To:** April James <aprilj@nipissingu.ca>; daniel.sola@uwaterloo.ca <daniel.sola@uwaterloo.ca>
**Subject:** RE: Rights and Permissions for Article Figure in CRIPE

Hi April and Daniel,

Mark Loewen is on board, so please take this as CRIPE providing permission for Daniel to use this figure in his thesis. I don't have a standard document or procedure to pass along to indicate this permission. Daniel, if there's something else that you feel you need, such as a Waterloo form signed by myself, please feel free to follow up with me. Otherwise, feel free to use the figure and I wish you all the best with finalizing your thesis.

Cheers,

Shawn

---

***Shawn Clark, Ph.D., P.Eng.***
Professor, Associate Head Research
Department of Civil Engineering
Price Faculty of Engineering
University of Manitoba
E1-368F, EITC

Office: (204) 474-9046
Cell: (204) 797-7765
http://umanitoba.ca/faculties/engineering/departments/civil/research/water

---

Figure 7.1: Email from Daniel Sola to April James asking for permission to use Figure 2.2; a figure from Kalke et al. [48]. April James forwards the request to Shawn Clark who grants permission.

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Jan 24, 2022

This Agreement between University of Waterloo -- Daniel Sola ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

| | |
|---|---|
| License Number | 5233200340465 |
| License date | Jan 20, 2022 |
| Licensed Content Publisher | Springer Nature |
| Licensed Content Publication | Springer eBook |
| Licensed Content Title | Where Are the Blobs: Counting by Localization with Point Supervision |
| Licensed Content Author | Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro et al |
| Licensed Content Date | Jan 1, 2018 |
| Type of Use | Thesis/Dissertation |
| Requestor type | academic/university or research institute |
| Format | electronic |

| | |
|---|---|
| Portion | figures/tables/illustrations |
| Number of figures/tables/illustrations | 2 |
| Will you be translating? | no |
| Circulation/distribution | 1 - 29 |
| Author of this Springer Nature content | no |
| Title | Thesis: River Ice Segmentation under Resource and Time Constraints |
| Institution name | University of Waterloo |
| Expected presentation date | Apr 2022 |
| Order reference number | 12345 |
| Portions | I would like to reuse a portion of Figure 2 (specifically the watershed splits) and the entirety of Figure 5. |
| Requestor Location | University of Waterloo<br>200 University Ave W, Waterloo, ON<br><br>Waterloo, AB N2L 3G1<br>Canada<br>Attn: University of Waterloo |
| Total | 0.00 CAD |

Terms and Conditions

**Springer Nature Customer Service Centre GmbH**
**Terms and Conditions**

This agreement sets out the terms and conditions of the licence (the **Licence**) between you and **Springer Nature Customer Service Centre GmbH** (the **Licensor**). By clicking 'accept' and completing the transaction for the material (**Licensed Material**), you also confirm your acceptance of these terms and conditions.

**1. Grant of License**

**1. 1.** The Licensor grants you a personal, non-exclusive, non-transferable, world-wide licence to reproduce the Licensed Material for the purpose specified in your order only. Licences are granted for the specific use requested in the order and for no other use, subject to the conditions below.

**1. 2.** The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of another entity (as credited in the published version).

**1. 3.** If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

**2. Scope of Licence**

**2. 1.** You may only use the Licensed Content in the manner and to the extent permitted by these Ts&Cs and any applicable laws.

**2. 2.** A separate licence may be required for any additional use of the Licensed Material, e.g. where a licence has been purchased for print only use, separate permission must be obtained for electronic re-use. Similarly, a licence is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence. Any content owned by third parties are expressly excluded from the licence.

**2. 3.** Similarly, rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to Journalpermissions@springernature.com/bookpermissions@springernature.com for these rights.

**2. 4.** Where permission has been granted **free of charge** for material in print, permission may also be granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

**2. 5.** An alternative scope of licence may apply to signatories of the STM Permissions Guidelines, as amended from time to time.

**3. Duration of Licence**

**3. 1.** A licence for is valid from the date of purchase ('Licence Date') at the end of the relevant period in the below table:

| Scope of Licence | Duration of Licence |
|---|---|
| Post on a website | 12 months |
| Presentations | 12 months |
| Books and journals | Lifetime of the edition in the language purchased |

**4. Acknowledgement**

**4. 1.** The Licensor's permission must be acknowledged next to the Licenced Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

**5. Restrictions on use**

**5. 1.** Use of the Licensed Material may be permitted for incidental promotional use and minor editing privileges e.g. minor adaptations of single figures, changes of format, colour and/or style where the adaptation is credited as set out in Appendix 1 below. Any other changes including but not limited to, cropping, adapting, omitting material that affect the meaning, intention or moral rights of the author are strictly prohibited.

**5. 2.** You must not use any Licensed Material as part of any design or trademark.

**5. 3.** Licensed Material may be used in Open Access Publications (OAP) before publication by Springer Nature, but any Licensed Material must be removed from OAP sites prior to final publication.

**6. Ownership of Rights**

**6. 1.** Licensed Material remains the property of either Licensor or the relevant third party and any rights not explicitly granted herein are expressly reserved.

**7. Warranty**

IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND
WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

**8. Limitations**

**8. 1.** ***BOOKS ONLY***Where **'reuse in a dissertation/thesis'** has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/).

**8. 2.** For content reuse requests that qualify for permission under the STM Permissions Guidelines, which may be updated from time to time, the STM Permissions Guidelines supersede the terms and conditions contained in this licence.

**9. Termination and Cancellation**

**9. 1.** Licences will expire after the period shown in Clause 3 (above).

**9. 2.** Licensee reserves the right to terminate the Licence in the event that payment is not received in full or if there has been a breach of this agreement by you.

**Appendix 1 — Acknowledgements:**

**For Journal Content:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication papers:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)

**For Adaptations/Translations:**
Adapted/Translated by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

**Note: For any republication from the British Journal of Cancer, the following credit line style applies:**

Reprinted/adapted/translated by permission from [**the Licensor**]: on behalf of Cancer Research UK: : [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication** papers:
Reprinted by permission from The [**the Licensor**]: on behalf of Cancer Research UK: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM])

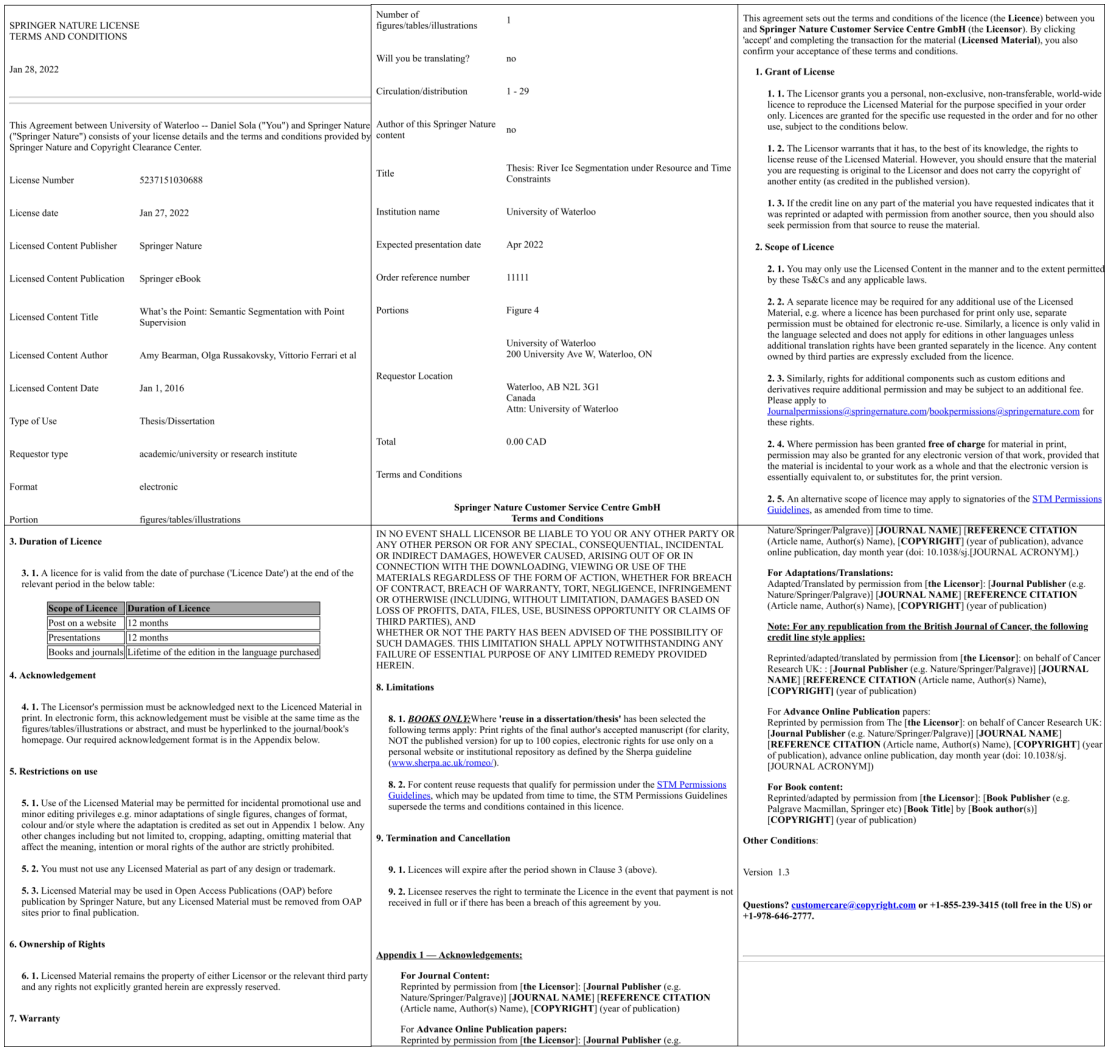
**For Book content:**
Reprinted/adapted by permission from [**the Licensor**]: [**Book Publisher** (e.g. Palgrave Macmillan, Springer etc) [**Book Title**] by [**Book author**(s)] [**COPYRIGHT**] (year of publication)

**Other Conditions**:

Version 1.3

**Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.**

Figure 7.2: Springer Nature License for Figures 2.8 and 2.7. Zoom in to see details.



SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Jan 28, 2022

This Agreement between University of Waterloo -- Daniel Sola ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

| | |
|---|---|
| License Number | 5237151030688 |
| License date | Jan 27, 2022 |
| Licensed Content Publisher | Springer Nature |
| Licensed Content Publication | Springer eBook |
| Licensed Content Title | What's the Point: Semantic Segmentation with Point Supervision |
| Licensed Content Author | Amy Bearman, Olga Russakovsky, Vittorio Ferrari et al |
| Licensed Content Date | Jan 1, 2016 |
| Type of Use | Thesis/Dissertation |
| Requestor type | academic/university or research institute |
| Format | electronic |
| Portion | figures/tables/illustrations |
| Number of figures/tables/illustrations | 1 |
| Will you be translating? | no |
| Circulation/distribution | 1 - 29 |
| Author of this Springer Nature content | no |
| Title | Thesis: River Ice Segmentation under Resource and Time Constraints |
| Institution name | University of Waterloo |
| Expected presentation date | Apr 2022 |
| Order reference number | 11111 |
| Portions | Figure 4 |
| Requestor Location | University of Waterloo<br>200 University Ave W, Waterloo, ON |
| | Waterloo, AB N2L 3G1<br>Canada<br>Attn: University of Waterloo |
| Total | 0.00 CAD |
| Terms and Conditions | |

This agreement sets out the terms and conditions of the licence (the **Licence**) between you and **Springer Nature Customer Service Centre GmbH** (the **Licensor**). By clicking 'accept' and completing the transaction for the material (**Licensed Material**), you also confirm your acceptance of these terms and conditions.

**1. Grant of License**

**1. 1.** The Licensor grants you a personal, non-exclusive, non-transferable, world-wide licence to reproduce the Licensed Material for the purpose specified in your order only. Licences are granted for the specific use requested in the order and for no other use, subject to the conditions below.

**1. 2.** The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of another entity (as credited in the published version).

**1. 3.** If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

**2. Scope of Licence**

**2. 1.** You may only use the Licensed Content in the manner and to the extent permitted by these Ts&Cs and any applicable laws.

**2. 2.** A separate licence may be required for any additional use of the Licensed Material, e.g. where a licence has been purchased for print only use, separate permission must be obtained for electronic re-use. Similarly, a licence is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence. Any content owned by third parties are expressly excluded from the licence.

**2. 3.** Similarly, rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to Journalpermissions@springernature.com/bookpermissions@springernature.com for these rights.

**2. 4.** Where permission has been granted **free of charge** for material in print, permission may also be granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

**2. 5.** An alternative scope of licence may apply to signatories of the STM Permissions Guidelines, as amended from time to time.

**Springer Nature Customer Service Centre GmbH**
**Terms and Conditions**

**3. Duration of Licence**

**3. 1.** A licence for is valid from the date of purchase ('Licence Date') at the end of the relevant period in the below table:

| Scope of Licence | Duration of Licence |
|---|---|
| Post on a website | 12 months |
| Presentations | 12 months |
| Books and journals | Lifetime of the edition in the language purchased |

**4. Acknowledgement**

**4. 1.** The Licensor's permission must be acknowledged next to the Licenced Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

**5. Restrictions on use**

**5. 1.** Use of the Licensed Material may be permitted for incidental promotional use and minor editing privileges e.g. minor adaptations of single figures, changes of format, colour and/or style where the adaptation is credited as set out in Appendix 1 below. Any other changes including but not limited to, cropping, adapting, omitting material that affect the meaning, intention or moral rights of the author are strictly prohibited.

**5. 2.** You must not use any Licensed Material as part of any design or trademark.

**5. 3.** Licensed Material may be used in Open Access Publications (OAP) before publication by Springer Nature, but any Licensed Material must be removed from OAP sites prior to final publication.

**6. Ownership of Rights**

**6. 1.** Licensed Material remains the property of either Licensor or the relevant third party and any rights not explicitly granted herein are expressly reserved.

**7. Warranty**

IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND
WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

**8. Limitations**

**8. 1.** ***BOOKS ONLY:*** Where **'reuse in a dissertation/thesis'** has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/).

**8. 2.** For content reuse requests that qualify for permission under the STM Permissions Guidelines, which may be updated from time to time, the STM Permissions Guidelines supersede the terms and conditions contained in this licence.

**9. Termination and Cancellation**

**9. 1.** Licences will expire after the period shown in Clause 3 (above).

**9. 2.** Licensee reserves the right to terminate the Licence in the event that payment is not received in full or if there has been a breach of this agreement by you.

**Appendix 1 — Acknowledgements:**

**For Journal Content:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication papers:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)

**For Adaptations/Translations:**
Adapted/Translated by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

**Note: For any republication from the British Journal of Cancer, the following credit line style applies:**

Reprinted/adapted/translated by permission from [**the Licensor**]: on behalf of Cancer Research UK: : [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication** papers:
Reprinted by permission from The [**the Licensor**]: on behalf of Cancer Research UK: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM])

**For Book content:**
Reprinted/adapted by permission from [**the Licensor**]: [**Book Publisher** (e.g. Palgrave Macmillan, Springer etc) [**Book Title**] by [**Book author**(s)] [**COPYRIGHT**] (year of publication)

**Other Conditions**:

Version 1.3

**Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.**

Figure 7.3: Springer Nature License for Figure 2.5. Zoom in to see details.

# References


[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[3] S Ansari, CD Rennie, O Seidou, J Malenchak, and SG Zare. Automated monitoring of river ice processes using shore-based imagery. *Cold Regions Science and Technology*, 142:1–16, 2017.

[4] Saber Ansari, Colin D Rennie, Shawn P Clark, and Ousmane Seidou. Application of a fast superpixel segmentation algorithm in river ice classification. In *Proceedings of the 20th Workshop on the Hydraulics of Ice Covered Rivers*, 2019.

[5] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[6] Spyros Beltaos. Progress in the study and management of river ice jams. *Cold regions science and technology*, 51(1):2–19, 2008.

[7] Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. In *Mathematical morphology in image processing*, pages 433–481. CRC Press, 2018.

[8] PT Bharathi and P Subashini. Texture based color segmentation for infrared river ice images using k-means clustering. In *2013 International Conference on Signal Processing, Image Processing & Pattern Recognition*, pages 298–302. IEEE, 2013.

[9] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[10] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] Darryl J Calkins. Physical measurements of river ice jams. *Water Resources Research*, 14(4):693–695, 1978.

[12] Lyndon Chan, Mahdi S Hosseini, and Konstantinos N Plataniotis. A comprehensive analysis of weakly-supervised semantic segmentation in different image domains. *International Journal of Computer Vision*, 129(2):361–384, 2021.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[15] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[17] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[19] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1635–1643, 2015.

[20] Apurba Das, Prabin Rokaya, and Karl-Erich Lindenschmidt. Assessing the impacts of climate change on ice jams along the athabasca river at fort mcmurray, alberta, canada. In *19th CRIPE workshop on the Hydraulics of Ice Covered Rivers, Whitehorse, Yukon, Canada*, 2017.

[21] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.

[22] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[24] Graham D Finlayson. Colour and illumination in computer vision. *Interface focus*, 8(4):20180008, 2018.

[25] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[26] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[27] Solomon Gebre, Knut Alfredsen, Leif Lia, Morten Stickler, and Einar Tesaker. Review of ice effects on hydropower systems. *Journal of Cold Regions Engineering*, 27(4):196–222, 2013.

[28] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[30] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Faye Hicks. An overview of river ice problems: Cripe07 guest editorial, 2009.

[34] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[35] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[36] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[37] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[38] Rhodri Howley. A study of complex river ice processes in an urban reach of the north saskatchewan river. 2021.

[39] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[40] T Kowalczyk Hutchison and FE Hicks. Observations of ice jam release waves on the athabasca river near fort mcmurray, alberta. *Canadian Journal of Civil Engineering*, 34(4):473–484, 2007.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[42] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[43] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[44] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[45] Zhentian Jiao, Youmin Zhang, Jing Xin, Lingxia Mu, Yingmin Yi, Han Liu, and Ding Liu. A deep learning based forest fire detection approach using uav and yolov3. In *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*, pages 1–5. IEEE, 2019.

[46] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 19–28, 2017.

[47] H Kalke and M Loewen. Support vector machine learning applied to digital images of river ice conditions. *Cold Regions Science and Technology*, 155:225–236, 2018.

[48] H Kalke, M Loewen, V McFarlane, and M Jasek. Observations of anchor ice formation and rafting of sediments. In *18th Workshop on the Hydraulics of Ice Covered Rivers. Quebec City, QC, Canada*, 2015.

[49] H Kalke, V McFarlane, C Schneck, and M Loewen. The transport of sediments by released anchor ice. *Cold Regions Science and Technology*, 143:70–80, 2017.

[50] David J Kerr, Hung Tao Shen, and Steven F Daly. Evolution and hydraulic resistance of anchor ice on gravel bed. *Cold Regions Science and Technology*, 35(2):101–114, 2002.

[51] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 876–885, 2017.

[52] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24:109–117, 2011.

[53] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[55] Issam H Laradji, Negar Rostamzadeh, Pedro O Pinheiro, David Vazquez, and Mark Schmidt. Where are the blobs: Counting by localization with point supervision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 547–562, 2018.

[56] Issam H Laradji, Negar Rostamzadeh, Pedro O Pinheiro, David Vázquez, and Mark Schmidt. Instance segmentation with point supervision. *arXiv preprint arXiv:1906.06392*, 2019.

[57] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3159–3167, 2016.

[58] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[59] Karl-Erich Lindenschmidt, Apurba Das, Prabin Rokaya, Kwok Chun, and Thuan Chu. Ice jam flood hazard assessment and mapping of the peace river at the town of peace river. In *18th Workshop on the Hydraulics of Ice Covered Rivers*, pages 18–21, 2015.

[60] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[61] Anmol Sharan Nagi, Devinder Kumar, Daniel Sola, and K Andrea Scott. Ruf: Effective sea ice floe segmentation using end-to-end res-unet-crf with dual loss. *Remote Sensing*, 13(13):2460, 2021.

[62] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[63] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

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

[65] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

[66] Kitsuchart Pasupa and Wisuwat Sunhem. A comparison between shallow and deep architecture classifiers on small dataset. In *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–6. IEEE, 2016.

[67] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[69] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. *arXiv preprint arXiv:1506.06204*, 2015.

[70] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer, 2016.

[71] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes in c, 1988.

[72] Wang Qiang. deepmask-pytorch. https://github.com/foolwood/deepmask_pytorch, 2018.

[73] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[74] Siddheswar Ray and Rose H Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, pages 137–143. Citeseer, 1999.

[75] Leslie M Reid and Thomas Dunne. Sediment budgets as an organizing framework in fluvial geomorphology. *Tools in fluvial geomorphology*, pages 357–380, 2016.

[76] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[77] John C Russ. *The image processing handbook*. CRC press, 2006.

[78] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[79] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[80] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. How does batch normalization help optimization? In *Proceedings of the 32nd international conference on neural information processing systems*, pages 2488–2498, 2018.

[81] Alexander Schindler, Thomas Lidy, and Andreas Rauber. Comparing shallow versus deep neural network architectures for automatic music genre classification. In *FMT*, pages 17–21, 2016.

[82] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. opencv/cvat: v1.1.0, August 2020.

[83] Vatsal Sharan, Kai Sheng Tai, Peter Bailis, and Gregory Valiant. There and back again: A general approach to learning sparse models. *arXiv preprint arXiv:1706.08146*, 2017.

[84] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[85] Abhineet Singh. river_ice_segmentation. *GitHub repository*, 2021.

[86] Abhineet Singh, Hayden Kalke, Mark Loewen, and Nilanjan Ray. Alberta river ice segmentation dataset, 2019.

[87] Abhineet Singh, Hayden Kalke, Mark Loewen, and Nilanjan Ray. River ice segmentation with deep learning. *IEEE Transactions on Geoscience and Remote Sensing*, 58(11):7570–7579, 2020.

[88] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[89] Lei Tai, Jingwei Zhang, Ming Liu, Joschka Boedecker, and Wolfram Burgard. A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. *arXiv preprint arXiv:1612.07139*, 2016.

[90] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[91] T Tieleman and G Hinton. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *Technical Report*, 2017.

[92] Julius T Tou and Rafael C Gonzalez. Pattern recognition principles. 1974.

[93] Gee Tsang. Frazil and anchor ice: a monograph: Nrc subcommittee on hydraulics of ice covered rivers. *Ottawa, Ontario, Canada*, page 90, 1982.

[94] Benoit Turcotte. Flooding processes and recent trends in ice-induced high-water levels along rivers of northwestern canada. In *21st Workshop on the Hydraulics of Ice, Saskatoon, SK, CGU HS Committee on River Ice Processes and the Environment*, 2021.

[95] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[96] Kristof Van Beeck, Tinne Tuytelaars, Davide Scarramuza, and Toon Goedemé. Real-time embedded computer vision on uavs. In *European Conference on Computer Vision*, pages 3–10. Springer, 2018.

[97] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[98] Penelope Mae Wagner, Nick Hughes, Pascale Bourbonnais, Julienne Stroeve, Lasse Rabenstein, Uma Bhatt, Joe Little, Helen Wiggins, and Andrew Fleming. Sea-ice information and forecast needs for industry maritime stakeholders. *Polar Geography*, 43(2-3):160–187, 2020.

[99] Qi Yang, Liangsheng Shi, Jingye Han, Jin Yu, and Kai Huang. A near real-time deep learning approach for detecting rice phenology based on uav images. *Agricultural and Forest Meteorology*, 287:107938, 2020.

[100] Xin Yang, Jingyu Chen, Yuanjie Dang, Hongcheng Luo, Yuesheng Tang, Chunyuan Liao, Peng Chen, and Kwang-Ting Cheng. Fast depth prediction and obstacle avoidance on a monocular drone using probabilistic convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[101] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[102] Man Zhang, Yong Zhou, Jiaqi Zhao, Yiyun Man, Bing Liu, and Rui Yao. A survey of semi-and weakly supervised semantic segmentation of images. *Artificial Intelligence Review*, 53(6):4259–4288, 2020.

[103] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[104] Xiuwei Zhang, Jiaojiao Jin, Zeze Lan, Chunjiang Li, Minhao Fan, Yafei Wang, Xin Yu, and Yanning Zhang. Icenet: A semantic segmentation deep network for river ice by fusing positional and channel-wise attentive features. *Remote Sensing*, 12(2):221, 2020.

[105] Xiuwei Zhang, Yang Zhou, Jiaojiao Jin, Yafei Wang, Minhao Fan, Ning Wang, and Yanning Zhang. Icenetv2: A fine-grained river ice semantic segmentation network based on uav images. *Remote Sensing*, 13(4):633, 2021.

[106] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[107] Yi-Tong Zhou and Rama Chellappa. Computation of optical flow using a neural network. In *ICNN*, pages 71–78, 1988.

# APPENDICES

# Appendix A

# Small DSC LBC UNet - Direct Comparison To Previous Work

We compare the results of our Small DSC LBC UNet directly to previous work [87] on the Alberta River Ice Segmentation Dataset. We use the same metrics used in previous work and use the same implementation as what was shown in the publicly available code [85]. These metrics are pixel accuracy, $pA$, mean pixel accuracy $mPA$, mean intersect over union, $mIoU$, and frequency weighted intersect over union, $fwIoU$. $pA$, $mPA$, $mIoU$, and $fwIoU$ are calculated using Equations 4.1, 4.2, 4.3, and 4.4 respectively.

Table A.1 shows the results of our networks as well the results a previous study, using the same training and testing sets that used that previous study [87]. The training set contains 32 images while the test set contains 18 images. The authors of previous experiments elected to refer to $mPA$ as recall and $mIoU$ as precision as accuracy measures only the rate of true-positives while intersect over union also accounts for false-positives. They also note that $pA$ is a frequency weighted measure of recall and $fwIoU$ is a frequency weighted measure of precision. Finally, the authors record these metrics on a class specific bases [87]. As a result, the formatting of Table A.1 is different from the other tables in this manuscript as we follow the same reporting style as previous work[1].

First we can compare the scores for our Full UNet with the scores reported for the UNet used in previous studies; referred to now on as Previous UNet [87]. This is important as we require confidence that our Full UNet, and other networks tested, can be meaningfully

[1]Note that in Table A.1, *Ice+Water Recall* is equivalent to what we refer to as $mPA$ in Table 5.1, while *Ice+Water Precision* is equivalent to what we refer to as $mIoU$ in Table 5.1. The reported metrics in both tables are of course different due to different training and tests sets in both experiments.

compared to previous work during our experiments. We notice that although the scores for *Frazil Ice Recall*, *Frazil Ice Precision*, and *Ice + Water Precision* are slightly higher for the Previous UNet [87] as compared to our Full UNet, the scores for *Anchor Ice Recall*, *Ice + Water Recall*, *Ice + Water Recall (fw)*, and *Ice + Water Precision (fw)* are slightly higher for our Full UNet. The proximity of these scores and the lack of a clear winner indicates that these models likely perform very similar. One outlier however is with the scores for *Anchor Ice Recall* where our Full UNet has a noticeably higher score than the Previous UNet. As mentioned, *Anchor Ice Recall* is the pixel accuracy for anchor ice. This simply means that the Full UNet predicted a number of anchor ice pixels that is closer to the actual number of anchor ice pixels than predicted by the Previous UNet. This does not necessarily mean that the location of those anchor ice pixels on the image are correct; which would be reflected more by the precision or $IoU$ measure. Looking at the *Anchor Ice Precision* for the Full and Previous UNet, these values are much closer, signifying that the increased *Anchor Ice Recall* score for the Full UNet does not clearly indicate better performance. As a result we can comfortable consider the Full and Previous UNet similar in performance.

Continuing to look at the reported scores from Table A.1, we see similarities to the trends shown in Table 5.1. Similar to our experiments, the Small DSC LBC UNet generally slightly outperforms the Full UNet, while MobileNetV3 (LR-ASPP) underperforms relative to the Full UNet. We also notice that the Small DSC LBC UNet performs the best in all precision measurements, which as noted, accounts for both true-positives and false-positives, unlike the recall measurements which only account for true-positives. Although the scope of our study is not purely performance but also efficiency, it is nice to see that our Small DSC LBC UNet has an added benefit of exceptionally high precision. In the context of sediment transport models, this can be helpful in avoiding over estimates due to incorrectly high anchor ice concentration predictions.

Table A.1: Metric comparison of the Small DSC LBC UNet and MobileNetV3 (LR-ASPP) with the UNet tested in previous work using the same training and testing split. Recall and precision for the two ice types correspond to class specific $pA$ and $fwIoU$. Ice+Water recall and precision correspond to $mPA$ and $mIoU$ respectively for the all classes. The frequency weighted (fw) equivalents for Ice+Water recall and precision correspond to $pA$ and $fwIoU$ respectively for all classes.

| **Model** | **Anchor Ice Recall** | **Frazil Ice Recall** | **Ice+Water Recall** | **Ice+Water Recall (fw)** |
|---|---|---|---|---|
| Previous UNet [87] | 73.75 | 84.27 | 85.13 | 88.69 |
| Full UNet | **80.03** | 83.86 | **87.22** | 89.35 |
| Small DSC LBC UNet | 74.18 | **87.28** | 86.84 | **90.57** |
| MobileNetV3 (LR-ASPP) | 77.35 | 80.22 | 84.32 | 86.99 |
| **Model** | **Anchor Ice Precision** | **Frazil Ice Precision** | **Ice+Water Precision** | **Ice+Water Precision (fw)** |
| Previous UNet [87] | 54.89 | 71.17 | 73.19 | 81.73 |
| Full UNet | 53.69 | 70.87 | 72.87 | 82.96 |
| Small DSC LBC UNet | **56.59** | **73.17** | **74.78** | **84.70** |
| MobileNetV3 (LR-ASPP) | 48.50 | 65.82 | 68.29 | 79.54 |

# Glossary

**DSC LBC** Our novel convolution block which has the architecture of a depthwise separable convolution, with the depthwise filters swapped with those of a local binary convolution. 2, 3, 40, 41, 43, 49, 51, 52, 55, 57, 58, 60–62, 96, 97

**Full** Prefix to the name of a UNet which denotes that the UNet has four upsampling and four downsampling layers, as well as 64, 128, 258, and 512 embedding dimensions in the first four layers. The convolution variant is not described by this prefix and can usually be determined by the word that immediately follows 'Full' in the network name. 41, 43, 49, 51, 54, 55, 57, 58, 96, 97

**HSV** Hue, Saturation, Value colour model for an image. There are three channels to the image, hue is measured in degrees from 0 to 360, while value and saturation are both measured on a scale of 0 to 100 percent. 22

**LC-FCN** Fully Convolutional Network with a localization-based counting loss. vii, 27–30, 64–73, 79, 80

**RGB** Red, Green, Blue additive colour model for an image. There are three channels to the image, each being coded on 256 levels from 0 to 255. 22, 35

**Small** Prefix to the name of a UNet which denotes that the UNet has two upsampling and two downsampling layers, as well as 32 and 64 embedding dimensions in the first two layers. The convolution variant is not described by this prefix and can usually be determined by the word that immediately follows 'Small' in the network name. 41, 43, 49, 51, 52, 55, 57, 58, 60, 96, 97