DEEP HETEROGENEOUS SUPERPIXEL NEURAL NETWORKS FOR IMAGE ANALYSIS AND FEATURE EXTRACTION

A Dissertation presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

ZHANGWEI (ALEX) YANG

Dr. Grant Scott, Dissertation Supervisor

JULY 2021

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

DEEP HETEROGENEOUS SUPERPIXEL NEURAL NETWORKS
FOR IMAGE ANALYSIS AND FEATURE EXTRACTION

presented by Zhangwei (Alex) Yang,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Grant Scott

Dr. Derek Anderson

Dr. Jeffrey Uhlmann

Dr. Jianlin Cheng

Dr. Lincoln Sheets

ACKNOWLEDGMENTS

I would first like to thank my advisor Dr. Grant Scott and my committee members, Dr. Derek Anderson and Dr. Jeffery Uhlmann, Dr. Jianlin Cheng and Dr. Lincoln Sheets, for their continual support of my researh and study, for their patience, encouragement and immense knowledge. Their questions and insights has contributed the result of this Ph.D. dissertation. Dr. Scott has helped me throughout my graduate research, including my Master's thesis and this dissertation.

I would like to thank my fellow graduate students and co-workers from Dr. Scott's High-Performance Data-Intensive Computing Systems Laboratory, from Dr. Davis's Geospatial Intelligence Laboratory (CGI), from Dr. Anderson's Mizzou INformation and Data FUsion Laboratory (MINDFUL) and from Dr. Scott's Mizzou Data Science and Analytics Program (MUDSA). We often brainstorm ideas and publish papers together. Notably, Charlie Veal and Alex Hurt have participated in many of the conversations that has helped steer the research into this interesting direction. Thanks to also Sam Kreter, Dr. Grant Scott's former research assistant, who previsously helped me set up equipment for collecting depth estimation training datasets, which we have continued to use in further research described in this dissertation. These laboratories have all contributed computational resources to the research.

Last but not least, I would like to thank my friends and family for supporting me throughout my Ph.D. study. It would not have been possible without each one of you!

Contents

List of Tables

Table Page

List of Figures

List of Algorithms

Algorithm Page

ABSTRACT

Lately, deep convolutional neural networks are rapidly transforming and enhancing computer vision accuracy and performance, and pursuing higher-level and interpretable object recognition. Superpixel-based methodologies have been used in conventional computer vision research where their efficient representation has superior effects. In contemporary computer vision research driven by deep neural networks, superpixel-based approaches mainly rely on oversegmentation to provide a more efficient representation of the imagery data, especially when the computation is too expensive in time or memory to perform in pairwise similarity regularization or complex graphical probabilistic inference. In this dissertation, we proposed a novel superpixel-enabled deep neural network paradigm by relaxing some of the prior assumptions in the conventional superpixel-based methodologies and exploring its capabilities in the context of advanced deep convolutional neural networks. This produces novel neural network architectures that can achieve higher-level object relation modeling, weakly supervised segmentation, high explainability, and facilitate insightful visualizations. This approach has the advantage of being an efficient representation of the visual signal and has the capability to dissect out relevant object components from other background noise by spatially re-organizing visual features. Specifically, we have created superpixel models that join graphical neural network techniques and multiple-instance learning to achieve weakly supervised object detection and generate precise object bounding without pixel-level training labels. This dissection and the subsequent learning by the architecture promotes explainable models, whereby the human users of the models can see the parts of the objects that have led to recognition. Most importantly, this neural design's natural result goes beyond abstract rectangular bounds of an object occurrence (e.g., bounding box or image chip), but instead approaches efficient parts-based segmented recognition. It has been

tested on commercial remote sensing satellite imagery and achieved success. Additionally, We have developed highly efficient monocular indoor depth estimation based on superpixel feature extraction. Furthermore, we have demonstrated state-of-the-art weakly supervised object detection performance on two contemporary benchmark data sets, MS-COCO and VOC 2012. In the future, deep learning techniques based on superpixel-enabled image analysis can be further optimized in accuracy and computational performance; and it will also be interesting to evaluate in other research domains, such as those involving medical imagery, infrared imagery, or hyperspectral imagery.

Chapter 1

INTRODUCTION

Even without salient inflection points, our daily life has already been transformed by the availability of artificial intelligence (AI) technologies such as facial recognition, object recognition, smart assistants, self-driving vehicles, as well as its vast and profound implications in areas of medicine, cybersecurity, and business. The landscape of computer vision has drastically changed since the resurgence of neural networks. Neural networks have empowered advanced AI systems in a variety of the aforementioned fields, and in particular computer vision. Neural networks, now popularized as deep learning models, have achieved state-of-the-art results in areas such as object recognition and localization [1, 2, 3, 4, 5], semantic segmentation [6, 7, 8, 9, 10], RGB-D [11] and 3-D data analytics [12], and more. Object detection, within the field of computer vision, is especially important as deep neural networks push forward to empower the next generation of AI and autonomous devices that must see and understand our natural world. Contemporary robust and highly accurate object detection models rely on data augmentation and transformation-invariant features. CNNs are inherently limited to these geometric transformations by the built-in convolutional kernels and network architectures [13]. Creating a high-level object detector is a demanding objective because the capability to recognize objects based on high-level features such as their appearance, known components and hierachical relations can provide a more generalizable abstraction of the object. Moreover, as we dive deeper

into the various approaches that provide faster, more accurate and more precise predictions, natural questions arise as to why and how are these neural networks able to perform these tasks with such great success; can we show what is involved in these neural networks' decision-making; can this learned knowledge by neural networks be broken down in a way that is comprehensible and explainable to humans? To address these objectives of creating a high-level object detector, weakly supervised learning is showing promise. Specifically, the weakly supervised approach does not provide complete data, e.g. pixel-level image class labels or instance labels, during training. Instead, this type of model will be designed to learn imagery objects with the absence of spatially precise labels, to dissect material data from noise and a achieve a higher level of scene understanding. That is, to achieve the true human-like behavior of recognizing objects and learning the salient parts and features of the objects that lead to successful and robust recognition.

One of the forerunners [13, 14, 15], that is actively approaching these areas of concern, is the ideology of superpixel-enabled image analysis. As a representation of imagery, the superpixel is a computer vision research area with much potential for exploration. The term *superpixel* was initially introduced by Ren and Malik in 2003 [16] and is used to describe a group of pixels similar in color or other low-level properties. In the contemporary research of deep neural networks, superpixel pooling is one of the primary recognized techniques to reduce the computational intensity when necessary, e.g., Markov Random Field (MRF), Conditional Random Field (CRF) or other types of Probabilistic Graphical Model (PGM) inference. From a deep learning perspective, these inference models translate into a sophisticated top layer, which can make the loss function rather computationally expensive. As reported in the literature, such as [11, 17], it is feasible to adapt superpixel features to many deep convolutional neural network (DCNN) architectures by deriving superpixel features from convolutional maps, making a potentially very expensive computation

feasible. Although the superpixel is not entirely a new concept in computer vision, its usage has been primarily confined within the scope of dimensionality reduction and low-level image segmentation, also known as *oversegmentation* within the literature. However, in this dissertation, we will focus on the strength of this ideology, its spatial organization capability, and novel paradigms that can be derived from it, which in hope, pushes image analysis to a higher cognitive level in both computational function and explainability of network decisions for humans.

A notable trend in the deep learning literature is that object detection and semantic segmentation are frequently treated as entirely different tasks. When dealing with imagery, object recognition and localization requires region-level class labels. These labels utilize meta information, such as rectangular bounding boxes, to represent the region of interest in which the object occupies. Common models that rely on the labeled regions include FasterRCNN [4], YOLO [5], SSD [18], and CenterNet [15]. While this approach is compact and efficient, it does not provide a precise boundary or contour of the object. And its predictive performance also involves a typical trade-off between the precision and the confidence threshold. In other words, we cannot identify or interpret the reasons behind these predictive decisions, especially regarding the interior of the bounding box. It is impossible to be certain that the neural network is focusing on the most critical aspects of the data and deriving its predictions with sound logic, being on the same reasoning level as humans; thus it is difficult to judge if the networks are providing reliable service to humans.

In a change of perspective, a *Multiple-Instance Learning* (MIL) based segmentation method proposes the challenge for the network to learn segmentation only with image class labels by learning from instances or occurrences of the object across the imagery dataset. MIL models are designed to learn from labeled "bags" containing unlabeled "instances", with the objective being to learn to predict the labels for both bags and instances. By presenting the MIL model a series of labeled bags, the MIL

model discovers and identifies the instances as well. When this technique is applied to image segmentation, it can create a Weakly-Supervised Object Detector (WSOD) model, that learns segmentation from only image class labels. With the MIL specific approach, we may adopt the term *Multiple-Instance Segmentation* (MIS) for this novel and high-level object detection methodology. In the case of the superpixel feature maps, they can be viewed as a perfect example of MIL bags, where the entire image, or the labeled rectangular region of interest (RoI) is the MIL bag, and the superpixels are unlabeled instances. Therefore, the superpixel feature map can fit into this framework very well and create novel WSOD models, which not only classify images, but also provide additional image analysis insights and achieve object detection from a training dataset that only labels entire images, or very coarsely labeled image RoI.

Finally, the highly interpretable nature of such a paradigm leads to the topic of explainability. The additional insight that a superpixel based high-level object detector provides can include object appearance, known components and hierarchical relations. The model architecture can be designed in a way such that this information is stored in different partitions of the model and can help with the analysis and diagnosis of the model decision-making. Therefore, we believe that the superpixel-based DNN ideology aligns with the current demanding research objective and has many novel and interesting potentials.

1.1 CONTRIBUTIONS

In this dissertation, we proposed a novel heterogeneous superpixel DNN paradigm, which creates an opportunity for high-level object detection, as well as various other deep learning computer vision tasks. This approach has the advantage of being an efficient representation of the visual signal and has the capability to spatially re-organize visual features and dissect out background noise from relevant object components.

Such re-organization preserves high-level visual features thanks to transfer learning from the start-of-the-art DCNN backbone, while generating precise pixel-level segmentation due to the usage of superpixels, which has a high recall of edges. We term this technique *heterogeneous superpixel* to distinguish it from the conventional usage of superpixels, which only emphasizes oversegmentation and serves as dimensionality reduction only. To this end, a GPU implementation of the superpixel feature extraction is created to perform such operations, capable of delivering a high-frame rate, and it is differentiable. This dissection and the high-level image analysis, such as using multiple-instance learning, promote explainable models, whereby the human users of the models can easily visualize the parts of the objects that have led to recognition. Most importantly, the result of this design goes beyond abstract rectangular bounds of an object occurrence (e.g., bounding box or image chip), but instead approaches efficient parts-based segmented recognition and object proposal learning. The predictive performance of these deep heterogeneous superpixel models has been demonstrated in solving specific image-to-image regression, segmentation and weakly-supervised object recognition problems.

1.2 OUTLINE

This dissertation first discusses contemporary advancement in computer vision using deep learning and existing challenges. Then it brings up the aspects of superpixel segmentation that are not adequately investigated in the contemporary literature. We then introduce several paradigms that can be derived from the superpixel-enabled image analysis ideology, which eventually leads to high-level object recognition and explainable models. Chapter 1 has introduced the problem domain and relevant background. Chapter 2 will cover the computer vision problem domains and task types that contemporary research predominantly focuses on, and the state-of-the-art achievements. Then we introduce the ideology of superpixel neural networks and

their potential in addressing some of the existing issues and challenges. Chapter 3 will cover the concept and utility of superpixels in traditional computer vision research, as well as mathematical notions around superpixels that the rest of this dissertation will be based upon. This will be followed by detailed discussions on the algorithms for generating superpixels. Chapter 4 will describe the superpixel feature extraction in detail, which is a technique that superpixel-based neural networks rely on heavily. This chapter discusses how superpixel feature extraction will bring together the efficient superpixel representation and popular deep neural network models. And then further explore how this notion can usher in novel approaches to high-level object detection. Chapter 5 will describe the deep heterogeneous superpixel networks' special role in solving the precise weakly-supervised object detection (WSOD) problem. A novel deep heterogeneous superpixel neural network architecture is proposed to solve this problem that targets precise object detection without pixelwise fully annotated training data. Chapter 6 will provide several successful examples, in which superpixel-enabled deep neural network architecture provides additional insights on the image datasets from various problem domains, while keeping up with the state-of-the-art model predictive performance. Chapter 7 will summarize the main discoveries, breakthroughs, results achieved and future directions with superpixel neural networks using deep heterogeneous superpixel neural networks.

Chapter 2

RELATED WORK

This chapter will cover the computer vision problem domains and task types that contemporary research predominantly focuses on, the taxonomy of deep convolutional neural networks, the state-of-the-art achievements, their roles and contributions in contemporary computer vision, as well as their potential limitations. Then, it will detail some of the trending research demands: high-level object recognition, weak supervision, explainable AI. Finally, we introduce the ideology of superpixel neural networks and their potential in addressing some of the existing issues and challenges.

2.1 IMAGE CLASSIFICATION NETWORKS

Image classification requires models that can assign an input image a label from a pre-defined set of categories, also known as *classes*. In the state-of-art techniques of constructing deep convolutional neural networks, many different structures have been proposed [19]. Most popular ones include AlexNet [20], VGG network [21], Deep Residual Network (ResNet) [22], Inception [23], DenseNet [24], and NASNet [25]. Each of the aforementioned neural architectures have brought novel techniques in architecture, training, and related optimizations to the field of contemporary neural network using different building blocks.

The AlexNet [20] and VGG [21] networks are exemplary networks that stack up convolutional layers with small filter kernels and max pooling layers in order to spa-

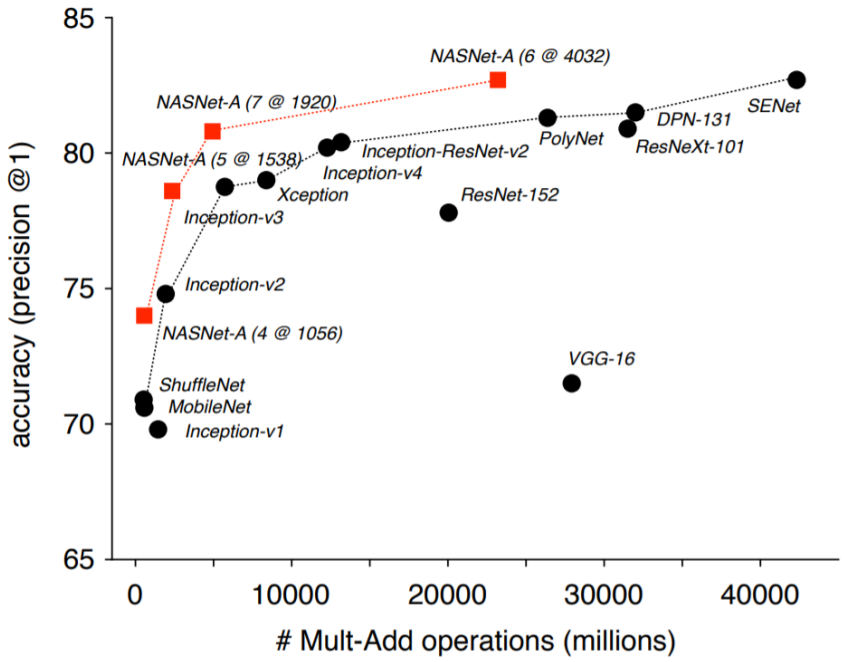


Figure 2.1: State-of-the-Art model precision versus computational demand, evaluated on the CIFAR-10 [26] and ImageNet [20] datasets. Figure from NASNet [25]. https://arxiv.org/pdf/1707.07012.pdf

tially aggregate the 2-D image signal and achieve high-accuracy image classification. Convolution is a very standard 2-D spatial signal processing technique in computer vision, and these deep convolutional networks have marked the ground-breaking success in creating intelligent non-handcrafted computer vision systems using learnable *kernels*. In essence, these kernels are localized signals, which are used to filter the input signal, hence also known as *convolutional filters*. The fundamental mechanism involves shifting the signals spatially across each other and applying an inner product, as shown in Eq. (2.1)

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

where "$*$" is the commonly used operator for convolution. The resulting signal function $f * g$ has higher numerical values (also known as *response* or *activation*) where the corresponding local signal from the input matches the kernel. Inspired by this,

CNNs essentially learn and maintain a large collection of these kernels to be able to accomplish much more sophisticated computer vision tasks. However, it is important to note that the practical implementation of convolutional layers in a CNN deviates from the the notion of convolution in several ways, as shown in Eq. (2.2),

$$(f *_{2D} g)_{mn} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} f_{ij} g_{m+i,n+j} \ , \tag{2.2}$$

where we use a different symbol "$*_{2D}$" to notate the typical 2-D "convolution" operation that exists in CNNs and (I, J) would be the kernel size. Ironically, it is, in fact, a 2-D discrete cross-correlation instead of convolution, due to the plus operators used in the kernel indices $m+i, n+j$ instead of minus operators. This change is valid because the convolution is found to be equivalent to a cross-correlation flipped along all spatial dimensions. Since the kernels will be learnable, this flipping would only introduce computational overhead without gain. Without flipping, the convolutional kernel and the input signal will be oriented in the same direction, which leads to easier human interpretation and cache-friendliness. Due to the fact that the kernel is a localized signal, and in a discrete case, it is mostly likely a very small patch of $(3 \times 3$ or $5 \times 5)$ pixels, the inner product is very compact, as opposed to the improper integral in Eq. (2.1) that spans across the entire real domain. Finally, from the neural network perspective, the CNN layer is a specialized shared neuron layer, whose back-propagation algorithm is already known.

When using a CNN for inference, the convolutional layer can be viewed as a function with a constant kernel g, as shown in Eq. (2.3),

$$\mathcal{F}_g(x) = x *_{2D} g \ . \tag{2.3}$$

The ResNet [22] and DenseNet [24] have proposed and promoted the residual connection technique, as shown in Fig. 2.4, to facilitate even deeper architectures

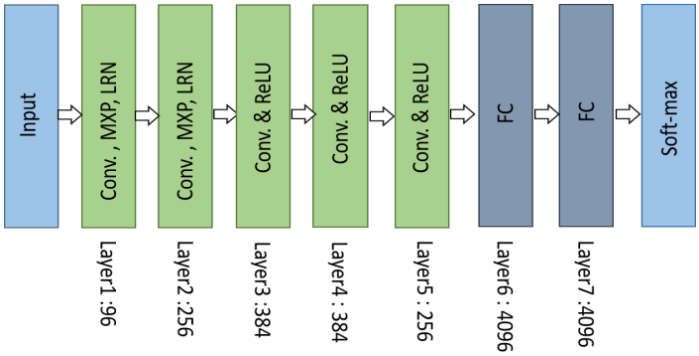


Figure 2.2: AlexNet Architecture. Figure from [20].

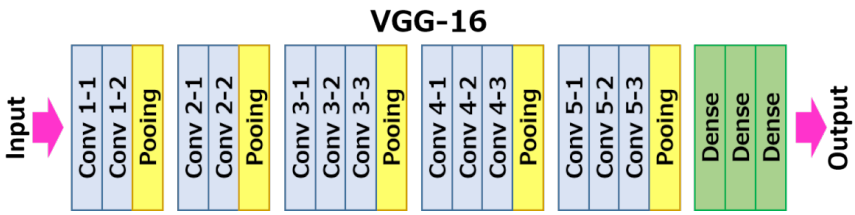


Figure 2.3: VGG-16 Architecture. Figure from [21].

that achieve higher accuracy. The primary incentive to build the model this way is to mitigate the well-known *Vanishing Gradient* problem that causes difficulties in DCNN training. Specifically, when the artificial neural network is deep, the error signals which are propagated back to the bottom layers are so small that it prevents the training from progressing. Theoretically, they also resemble *Power Series*,

$$f(x) = \sum_{n=0}^{\infty} a_n (x-b)^n \ ,$$

or *Taylor Series*,

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x-x_0)^n \ ,$$

which bring together signals from different orders, which has allowed these the deep neural networks to be effective approximators of the class probability distribution functions. For example, the $\mathcal{F}(x) + x$ in Fig. 2.4 is a mathematical representation of a residual network building block [22]. The $\mathcal{F}(\cdot)$ represent convolutional kernels

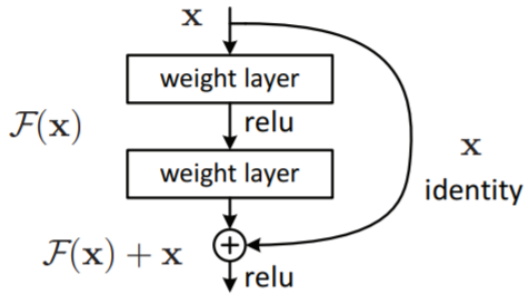


Figure 2.4: ResNet Building Block [22].

that are essentially an assorted collection of 2-D derivative and weighted average operators. As these building block are stacked up, that formula would compound as $\mathcal{F}_{n+1}(\mathcal{F}_n(x) + x) + \mathcal{F}_n(x) + x$, etc., and include various orders of derivatives.

The Inception [23] and NASNet [25] architectures have further explored non-linear ways of wiring of the neural network layers. Both the Inception network and the NASNet are motivated by reducing the computational cost of DCNN. The Inception network introduced some 1×1 convolution as a way of dimensionality reduction in each building block as the convolutional map is fed into the subsequent 3×3 and 5×5 convolutional layers. However, the NASNet, as its name Neural Architecture Search (NAS) suggests, goes one step futher and has proposed to search for the optimal building block architecture based on a smaller scale dataset, such as CIFAR [26]. The NASNet has even taken the computational efficiency into account to derive the optimal neural network architecture, which has the minimum number of parameters with the maximum accuracy. Fig. 2.1 from NASNet [25] provides a comparison among popular DCNN models in terms of precision and computational demand in million multiply-adds.

Each of the aforementioned architectures accomplish image classification and have been evaluated on the ImageNet [20] dataset. VGG16 was able to achieve 92.3% top-5 accuracy on 1000-class ImageNet dataset; ResNet 50 achieved 93.3% accuracy; Inception v3 achieved 96.4% accuracy and NASNet achieved 96.2% accuracy in the

same metric. The best-performing models are released by the authors to facilitate further research. Additionally, much of the value has come through the adaption of the networks, leveraging the already trained feature extraction phases, along with the application to a new domain (versus ImageNet ground photos) – commonly referred to as *transfer learning*. Transfer learning allows models to generalize better by leveraging existing labeled data of some related problem domain, and transfer this knowledge gained in solving the source problem to solving the problem of interest. Recall that a typical DCNN architecture for image classification be divided into two parts: firstly, a stack or ensemble of layers of convolution and/or pooling and batch normalization; secondly, a stack of dense layers which resembles *Multi-layer Perceptron* (MLP). The first phase of feature extraction can be viewed as a learnable visual feature extraction component, whereas the second phase of image classification learns and handles the class feature encoding and the decision-making for the final class prediction. Between the two parts, there can be a global pooling or flattening layer, that will transform the 2-D image feature into a spatially 1-D vector representation. In practice, it is not common for researchers to train a deep convolutional network from scratch because of limited number of data samples available. On the other hand, the neural network layers in the feature extraction phase of a pre-trained convolutional network can be re-used as a good initial point of training because even different problems may have overlapping collections of features that are beneficial. Additionally, incorporating these pre-trained models are very beneficial to building large-scale image recognition based models even with relatively small dataset. In order to transfer these feature filters (feature extraction phase) learned from pre-training on ImageNet (or similar datasets), the fully-connected layers (classification phase) would be replaced with a new set of layers appropriate for the new problem domain.

2.2 IMAGE SEGMENTATION NETWORKS

Image segmentation is the process of dissecting a image into multiple smaller parts, also known as *segments*. The goal of this process is to transform the image into a representation that is more useful to process and analyze. *Semantic segmentation* tackles the task of assigning each pixel a class label. By doing so, the image will be segmented into many meaningful components with clear boundaries. Pascal VOC [2] and MS COCO [3] are very popular datasets to train and evaluate semantic segmentation networks. A very successful approach to semantic segmentation is the fully convolutional neural network (FCN). The FCN [6] provides a model that takes input of arbitrary sized images and produce a same-sized output. The main architecture difference between a image classification network and a semantic segmentation network is that the latter typically uses a transposed convolutional layer. This terminology is rooted in the Toeplitz matrix representation of the convolution; where this operation can be interpreted as the transposed Toeplitz matrix [27]. A transposed convolutional layer [28] [29] [30] is also known as "fractionally strided convolution" or "back-prop convolution" or "dilated convolution" [7] or "deconvolution" [8, 31], by which it is better known, nevertheless, is an unfortunate misnomer following [6] and [8, 31] because the actual deconvolution operation would be referring to the inverse of convolution, an ill-posed operation [32].

Fig. 2.5 is an illustration of the transposed convolution computation. From the visual perspective, Fig. 2.5 shows the reason that is is also called the dilated convolution or fractionally strided convolution, due to the internal zero paddings. In [7], the authors have compared dense up-sampling and hybrid dilated convolution for semantic segmentation. This work has specifically scrutinized the dilated convolution from the validity and effects of padding zeros values internally. And the hybrid dilated convolution aims to address the loss of receptive field from these internal zero-

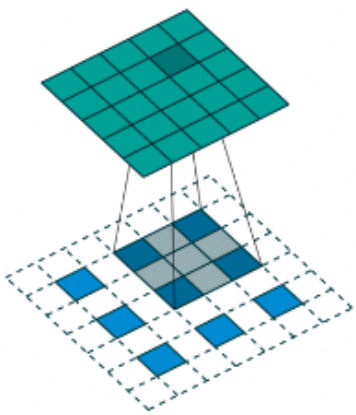


Figure 2.5: Transposed convolutional layer, also known as "deconvolution". Figure from Convolultion Arithemtic `https://github.com/vdumoulin/conv_arithmetic`

padding by carefully adjusting the kernel sizes such that they collectively provide a complete coverage of the input image. Last but not least, in practice, these operations, both convolution and transposed convolution, are often accelerated by the Fast Fourier Transform (FFT) [33, 34, 35] and are implemented as auto-correlation in lieu of convolution in DCNNs.

The state-of-the-art image semantic segmentation network architectures include RefineNet [9], U-Net [10], etc. Both U-Net [10], as shown in Fig. 2.6, and RefineNet [9] have adopted the residual connection technique from ResNet [22], and constructed a popular encoder-decoder structure, where the decoder fuses equivalent resolution features using residual connections. Conditional Random Field (CRF) is typically used among those semantic segmentation networks as regularization to enhance smoothness. In Chapter 4 we will introduce superpixel feature extraction's capability to facilitate residual connections across arbitrary feature resolutions and circumvent upsampling.

2.2.1 Instance Segmentation Networks

Instance segmentation is a concept that is built upon semantic segmentation, where, in addition, each individual object from a pre-defined set of classes should be distinctly

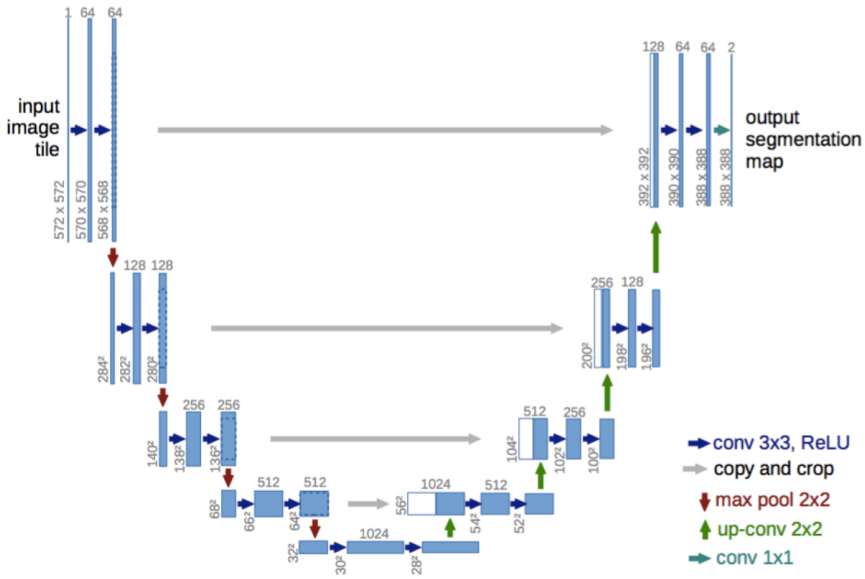


Figure 2.6: U-Net Architecture [10]

identified, where each occurrence is known as an *instance*. Specifically, this provides the capability to tell one instance from another even if they are from the same class, to count the number of instances, and to obtain precise boundary for each instance.

For instance, an object proposal based discriminative convolutional network, DeepMask [36], is trained jointly with two objectives: given an image patch, the first part of the system outputs a class-agnostic segmentation mask, while the second part of the system outputs the likelihood of the patch being centered on a full object. At test time, the model is efficiently applied on the whole test image and generates a set of segmentation masks, each of them being assigned with a corresponding object likelihood score. The model yields significant improvements over prior object proposal algorithms to achieve instance segmentation, with higher object recall using fewer proposals. DeepMask can locate objects within input images, but cannot describe them and their boundaries.

Later research SharpMask [37] has made further improvements upon DeepMask by augmenting the low-level layer using high-level feature as a way of refinement. The network first outputs a coarse "mask encoding" in the feed-forward pass, then refines

Figure 2.7: DCNN Popular Tasks Illustration, figure from [39].

this mask encoding in a top-down pass utilizing features at successively lower layers. The resulting bottom-up/top-down architecture is capable of efficiently generating high-fidelity object masks.

FastMask [38] takes advantage of hierarchical features in deep convolutional neural networks to segment multi-scale objects in one shot. On the MS COCO benchmark [3], the proposed FastMask outperforms all state-of-the-art segment proposal methods in average recall being 2–5 times faster. With a slight trade-off in accuracy, FastMask can segment objects in near real-time ($\sim$13 fps) with 800×600 resolution images, demonstrating its potential in practical applications.

2.3 SUPERPIXEL NEURAL NETWORKS

Prior research in the field of deep learning has used superpixels mainly to reduce the computational intensity. Superpixel pooling is one of the primary recognized techniques of dimensionality reduction for deep convolutional networks. This method allows perceptually homogeneous regions of pixels to be grouped together and represented more concisely [40], and then aggregated according to such regions.

There are several well-known algorithms to generate the superpixel segmentation, such as the Felzenszwalb and Huttenlocher's Algorithm [41], Watershed [42], Quick Shift [43] and SLIC [40, 44]. In contemporary computer vision research, Felzenszwalb's Algorithm and Watershed are recognized as the best in computational ef-

ficiency, and Quick Shift robust to local variations. However, the greatest concern has been their tendency to straddle multiple objects [45, 46]. In practice, the SLIC algorithm is frequently chosen especially in the context of deep learning [11, 47, 48], due to its balance between the computational speed and the conservative estimation of the homogeneous regions, especially concerning edge recall. The gSLIC algorithm [44] remains the most utilized algorithm in deep learning when superpixel is involved, which heavily leverages general purpose GPU (GPGPU) implementations to achieve acceleration. Despite all of this, other intriguing superpixel segmentation algorithms exist, such as the Entropy Rate Segmentation [46], NCut [49, 50] and GraphCut [51], and they may emerge as alternative choices as we evolve the superpixel neural network research and rethink many conventional assumptions that no longer hold true thanks to the tremendous contribution and impact from deep learning.

Since it is feasible to adapt superpixel features to many DCNN architectures [11, 17] to extract features from convolutional maps, there have been numerous applications of superpixels in DCNN. In the work of Liu, *et al.* [11] RGB-D data is used to train a single frame monocular vision depth estimator based on efficient superpixelated [52] CNN features and conditional random field (CRF) inference, which has involved expensive computation such as pairwise similarity metrics in their regularization terms. This computation would have been impractical without extracting convolutional features using superpixels that drastically reduced the dimensionality of the similarity metrics. More interestingly, however, in some literature it is reported that not only have superpixels made the computation more efficient, but there has been model performance benefits as well. For example, in the work of Mostajabi, *et al.* [53] superpixels were used to exploit statistical structure in the image and in the label space and to avoid complex and expensive inference. More recently, in the work of Wu, *et al.* [17] superpixel based analysis is incorporated to perform highly efficient cloud image segmentation. The local consistency of the segmentation was

preserved and a faster DCNN learning was reported. The challenge with processing superpixel features and learning a model that makes inferences based on superpixels is to fully exploit the superpixel features with some capable mathematical machinery; as opposed to performing naive aggregations too early and losing a significant amount of information. The superpixel feature map has an interesting representation, which provides opportunities for developing novel DCNN architectures that can process the imagery in a efficient way and derive extra insightful information from the dataset. In the meantime, such superpixel based DCNN architectures are facing new issues, such as each superpixel segmentation of the image being completely different, and there is no specific role assigned to the neurons handling the superpixels, and this makes the superpixel feature space difficult to learn especially with high dimensionality.

Chapter 3

IMAGERY REPRESENTATION AND DIMENSIONALITY REDUCTION

This chapter will cover the concept and utility of superpixels in traditional computer vision research, as well as mathematical notions around superpixels that the rest of this dissertation will be based upon. This will be followed by detailed discussions on the algorithms for generating superpixels.

3.1 MATHEMATICAL REPRESENTATION

The superpixel representation of imagery arises to address two primary concerns: (1) the pixels cannot directly represent natural entities, e.g., objects in an image; (2) the large amount of pixels brings about a significant computational challenge [54] in complex computer vision algorithms. Superpixels are typically used to group perceptually homogeneous regions in order to approximate and represent the image by such groupings. Herein, we will consider the mathematical formulation of images and superpixel and examine their properties and relations.

3.1.1 The Image Function

First of all, the *image function* needs to be re-introduced from conventional image analysis [55], because to precisely characterize these superpixel concepts and prove a few theorems, it requires a general and versatile definition of the image instead of the

Table 3.1: Symbols and Notation from Chapter 3

s.t.	"subject to"
dim	the dimensionality of a topological space
dom	the domain of a function
$\varnothing$	diameter
px	pixel count along a horizontal or vertical line; or a unit of length
px^2	pixel count in a region; or a unit of area
$\varnothing$ 4px	a superpixel size that is equivalent to a 4×4 square pixel patch
16px^2	a superpixel size that contains 16 pixels in the region enclosed thereof Note: it is true that $\varnothing\ x\,\text{px} \equiv x^2\,\text{px}^2$.
$p = (i, j)$	a tuple is a group of quantities and they may be named
$x(i, j)$	for a single parameter function x, the parenthesis around the single tuple parameter may be omitted, because it does not cause ambiguity i.e. $x((i, j)) \equiv x(i, j)$ if function x only has a single parameter
$x(p) = i^2$	assuming $p = (i, j)$, named components of a tuple parameter may be directly referenced in a function definition to simplify the formula

popular matrix approach. A few notations that will be used throughout the chapter are shown in Tab. 3.1.

A gray-scale image function maps each image point to a scalar quantity, by which its corresponding *brightness* is encoded [55]. This brightness may consist of a variety of other optical quantities, which are intentionally abstracted away here for simplicity of the discussion and the mathematical capturing of the concept of an image. As imaging sensors produce the image by means of projecting the 3-D scene to a 2-D plane. The image function inherently maps a 2-D topology to real values, as shown in Eq. 3.1,

$$x : V \to \mathbb{R} \quad \text{s.t.} \quad \dim V = 2\ . \tag{3.1}$$

As images are digitized for computer processing, it is common in the contemporary image processing and computer vision literature to assume that this 2-D topology has to be a subset of the 2-D Cartesian coordinate system $\mathbb{R}^2$, or even the integer

coordinate system $\mathbb{Z}^2$, such as

$$\text{dom } x = \left\{(i,j), 1 \leq i \leq m, 1 \leq j \leq n, m \in \mathbb{Z}^+, n \in \mathbb{Z}^+\right\} \subseteq \mathbb{Z}^2 \ . \tag{3.2}$$

Those assumptions also lead to the prominent matrix representation of an image, as shown in Eq. 3.3,

$$x(i,j) \equiv X_{ij} \in \mathbb{R} \ . \tag{3.3}$$

However, the rest of this dissertation does not necessarily adopt these assumptions unless expressly stated, because these are mostly irrelevant constraints, and the matrix representation is often found inconvenient for superpixel image analysis.

3.1.2 Image Feature Space

The *codomain* of an image function is commonly referred to as the feature space, e.g. a set of possible brightness values. In the case of a gray-scale image function, the feature space is the set of real numbers $\mathbb{R}$. As an extension, a color or multi-channel image function $\boldsymbol{x} : V \to \mathbb{R}^K$ maps image points to the feature space $\mathbb{R}^k$, where K is the feature space dimensionality. For instance, this could be the RGB color space or the CIE L*a*b color space. In most circumstances, this extension can be trivially viewed as a collection of gray-scale image functions u_i instead,

$$\boldsymbol{x}(p)_i = u_i(p) \ , u_i : V \to \mathbb{R} \ , 1 \leq i \leq K \ , \tag{3.4}$$

so we will focus on gray-scale image functions.

3.1.3 Superpixel

In the language of topology, a superpixel is a closed set on the image domain. Practically speaking, a superpixel can be thought of as a group of pixels that form a

Table 3.2: Properties of a Superpixel

	s is a closed set on V
	s is connected
$s \subseteq V$	s is a subset of V
$\dim s = 2$	s is a 2-D topological subspace of V
$\lvert s \rvert \in \mathbb{R}^+$	the area of a superpixel can be measured
$D : (V, \mathbb{R}^k) \to \mathbb{R}^+$	there exists a distance metric D on the combined space of $(V, \mathbb{R}^k)$
$x : S \to \mathbb{R}$	the image function applies to superpixels,
	where $\dim S = 2, S = \{s \subseteq V\}$

connected polygonal region. Apart from the topological properties of a superpixel, namely being 2-dimensional, closed and connected, it is also an important bridge that connects to the image feature space. Therefore, taken into account the feature space of the image, it is not only a closed set from the image domain, which are within the local vicinity on the 2-D topology V, but also may share proximity in other feature spaces, such as color, at the same time. This is hinting at the distance metric across both the 2-D topological space and the feature space, that is necessary to define and generate superpixels. The specific approach will be discussed in Sect. 3.2.2. Tab. 3.2 summarizes important properties of the superpixel, s.

3.2 SUPERPIXEL SEGMENTATION

The superpixel segmentation is a process that breaks the image down into a superpixel representation. The segmentation itself can be formulated as simultaneously finding a function $c : V \to S$ that surjectively maps pixels to superpixels, and a collection S of these superpixels, as

$$S = \left\{ s \subseteq V \middle| V = \bigcup_j s_j \ , s_i \cap s_j = \varnothing \right\}, \tag{3.5}$$

given

$$\forall p_i \in V, \exists j : c(p_i) = s_j \in S \ . \tag{3.6}$$

And when distance metric $D : (V, \mathbb{R}^k) \rightarrow \mathbb{R}^+$ is well maintained, and values are assigned to superpixels through the superpixel-based image function $x_c : S \rightarrow \mathbb{R}$, it will approximate the original image function x,

$$\exists M \in \mathbb{R}^+, \forall p \in V : |(x_c \circ c)(p) - x(p)| < M \ , \tag{3.7}$$

where M is the error bound.

Therefore, the segmentation c bridges the two different representations of an image, x and x_c, and $(x_c \circ c)$ would be a superpixel image reconstruction. This is worthwhile to note that, because $\mathrm{dom}(x_c \circ c) = \mathrm{dom}\ x = V$, the reconstructed image $(x_c \circ c)$ shares the exact same 2-D topology as x, which is advantageous for visualization and cross-resolution feature aggregation. Such implications and the methodology for the assignment of x_c will be further explored in Chapter 4.

For practical purposes, each superpixel in a segmentation can be labeled with an unique integer from interval $[1, |S|]$. And the segmentation c can be represented as a function $V \rightarrow [1, |S|] \subseteq \mathbb{Z}$. In a matrix representation of image x, as V only consists of matrix indices, c is practically a same-shape 2-D integer array, where each pixel is labeled with the superpixel index. Such is the implementation for computer algorithms in Sect. 3.2.2.

For example, in the work of [56] and [57], the assumption was that an image is composed of small homogeneous regions (superpixels) and the graphical model composed of nodes defined on superpixels was considered. In the scope of DCNN architecture, the usage of superpixels is mainly limited to dimensionality reduction. Fig. 3.1 shows a few examples of superpixel segmentation. It visually demonstrates the effects of the two most dominant superpixel parameters: size and compactness –

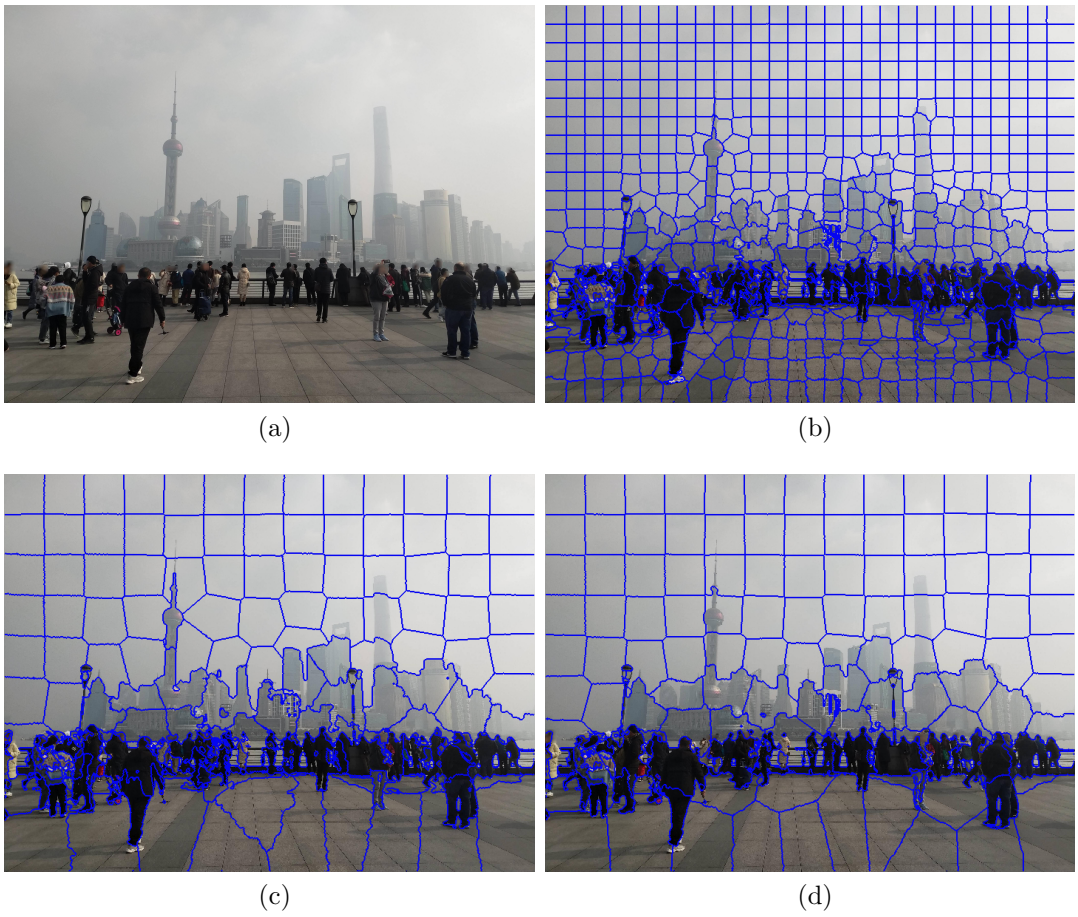


Figure 3.1: A demonstration of Superpixel segmentation. (a) input RGB image; (b) superpixel segmentation size = ⌀ 25px, compactness = 0.6; (c) size = ⌀ 54px, compactness = 0.6; (d) size = ⌀ 54px, compactness = 1.8. Picture taken at the Bund, Shanghai, Jan 2020.

these parameters and their use in the superpixel generation algorithm will be discussed in Sect. 3.2.2. For instance, Fig. 3.1 (b) and (c) have used the same compactness (0.6) with different superpixel sizes (⌀ 25px and ⌀ 54px). As a result, (b) produces more superpixels than (c) and is able to capture more details, such as parts of the skyscrapers, in the fog and with low contrast, on the right of each picture. On the other hand, Fig. 3.1 (c) and (d) have used the same superpixel size (⌀ 54px) with different compactness (0.6 and 1.8). As a result, (c) puts more weight on color proximity and produces superpixels that better align with natural object boundaries.

3.2.1 Oversegmentation

The superpixel segmentation, as in the process which generates a superpixel representation or structure of the image, is also known as *superpixelation*. Superpixelation is usually considered computationally and representationally more efficient [55] in the processing of imagery, although it can also be expensive to compute the actual superpixel groupings in the first place. In the domain of superpixel-enabled DCNNs, it is traditionally a common practice to generate an oversegmentation [58]. This means that most of the important boundaries in the image are found and very few pixels are lost from the pixel grid to superpixel mapping. In essence, the superpixelation is a type of image segmentation that breaks down the image into visually consistent pieces. And, theoretically, it is more acceptable to have those pieces smaller than the natural entities in the image, than overly large pieces which can obliviate too many details from the image. This practice is known as *oversegmentation*. Therefore, an oversegmentation breaks images into homogeneous areas and may create more superpixels than necessary while retaining edges that exist in a image, as those edges provide important structural information about the image objects.

The trade-off worth considering in superpixelation is number of superpixels versus effect of dimensionality reduction. If more superpixels were created, it would become more likely that important edges are preserved. On the other hand, dimensionality reduction would become less effective once the number of superpixels is increased. Thus, the goal is to minimize the number of superpixels while preserving important edges that provides key information in images.

The number of superpixels can be chosen so that the resulting superpixel size can align well with the natural entities in the imagery. This provides a way of controlling the interest focus for a superpixel-enabled DCNN. For example, the superpixel DCNN can use very small superpixel sizes to focus on various types of texture details, or slightly larger superpixels to focus on some of the natural entity components.

Chapter 4 will further explore this topic.

3.2.2 Superpixelation via Clustering

One common algorithm for superpixelation is Simple Linear Iterative Clustering (SLIC) [40], which is a relatively simple and efficient method to decompose an image into homogeneous regions, based on k-means clustering. The image would be first divided into a grid with each tile center prepared to initialize a k-means. Then after refining the clusters with Lloyd's algorithm, the segmentation would be completed [59].

From a high-level view, the SLIC algorithm initializes a grid of superpixels then refines the boundaries to follow the edges based on the clustering algorithm. So these numbers would only be an approximate number of superpixels because SLIC algorithm will try to generate the number of superpixels as specified for the input image, but in practice, the number may vary slightly depending on whether the algorithm decides to merge or split some superpixels as appropriate [58].

The superpixel segmentation will determine the number of extracted features and affect the structure of the subsequent classifier network; as such, the parameters used to generate superpixels become new hyperparameters for the neural network architecture. Superpixels are generated by way of the SLIC superpixel algorithm. It is the most reasonable choice because the effects of different superpixel segmentation methods have extremely marginal differences in this model, while they vary drastically in their time-efficiency (e.g., GPU acceleration versus standard CPU implementations).

This algorithm first converts the image's color into CIE L*a*b color space, then clusters the data in the 5-D feature space composed of the concatenation of CIE L*a*b and the 2-D pixel (spatial) coordinate system, i.e., $p = (l, a, b, x, y) \in \mathbb{R}^5$. The color proximity and spatial proximity are weighted in the form of a compactness factor m. The distance measure $D : \mathbb{R}^5 \times \mathbb{R}^5 \to \mathbb{R}^+$ used in the SLIC algorithm is shown in

Eq. (3.8), where ρ is the sampling interval, which controls the grid geometry upon initialization; $d_c : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^+$ is color proximity; $d_s : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^+$ is spatial proximity [40];

$$D(p_1, p_2) = \sqrt{d_c^2(p_1, p_2) + \left(\frac{d_s(p_1, p_2)}{\rho}\right)^2 m^2} \ , \tag{3.8}$$

$$d_c(p_1, p_2) = \sqrt{(l_1 - l_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2} \ , \tag{3.9}$$

$$d_s(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \ . \tag{3.10}$$

The number of superpixels $|S|$ will control the granularity of the superpixel segmentation, and the compactness factor m will control the superpixel shape irregularity that is achievable based on weight associated to spatial proximity.

Alg. 3.1 describes this procedure of generating superpixels using K-means clustering. Initially, the algorithm will convert the image into CIE L*a*b* color space. Then create a grid that covers the entire input image, each block having the same size as what the superpixel size would be estimated to as. This grid will serve as the initialization for the K-means cluster prototypes, which represent superpixels, according to the distance metric defined in Eq. 3.8. Then a standard K-means alternate optimization will proceed to refine these clusters, and in turn, refine the superpixels' position and shape. The "UpdateCluster" step will optimize superpixel prototype positions given a pixel-to-superpixel assignment. And the "FindClusterAssociation" step will optimize the superpixel assignment given a set of superpixel prototype centers. The algorithm comes to a stop as these two steps have been iterated alternately a fixed number of times. Finally, the "EnforceConnectivity" post-processing step is performed to mitigate stray pixels and improve the quality of the superpixel segmentation quality.

Algorithm 3.1 SLIC Superpixel Segmentation using K-means

procedure InitClusters(x, $superpixelSize$)
  $numPixel \leftarrow |\text{dom } x|$
  $numSuperpixel \leftarrow numPixel/superpixelSize$
  Initialize c_i and μ_j to make the segmentation a grid
procedure UpdateClusters(x)
  for $i : (0, numSuperpixel)$ **do**
    $\mu_j \leftarrow \frac{\sum_i [c_i=j] x(i)}{\sum_i [c_i=j]}$
procedure FindClusterAssociation(x)
  for $i : (0, numPixel)$ **do**
    $c_i \leftarrow \text{argmin}_j D^2(x_i, \mu_j)$
procedure SLIC($imInput$, $superpixelSize$, $nIter$)
  ConvertColor($imInput$) ▷ RGB to Lab color conversion
  InitClusters(x, $superpixelSize$)
  FindClusterAssociation($imInput$)
  for $i : (0, nIter)$ **do**
    UpdateClusters( )
    FindClusterAssociation($imInput$)
  EnforceConnectivity($imInput$)

3.2.3 GPU Acceleration

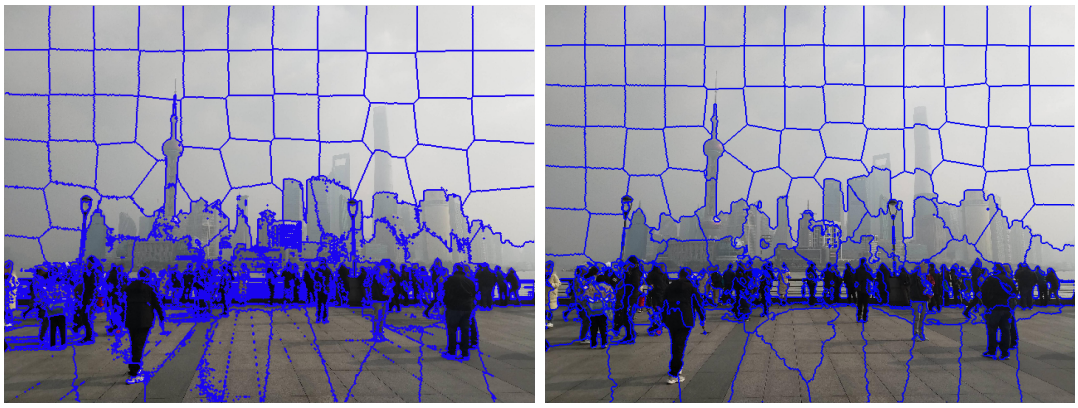
The state-of-the-art SLIC algorithm that is implemented on GPU can achieve 83× speedup compared to a sequential program according to [44]. Same as K-means, this alternately optimizes the pixel assignment and the superpixel cluster center. The GPU algorithm is shown in Alg. 3.2, where the "<>" that follows directly after the function call name designates the way that parallel threads are to be launched.

Alg. 3.2 creates one thread per pixel and assigns the pixel to the best cluster prototype, which represent a superpixel, in the aforementioned 5-D feature space. Then the algorithm updates the cluster prototype according to the feature of the pixels assigned to the cluster. These two steps will repeat for a fixed number of iterations. Typically the number of iterations is a very small number such as 5 or 3. Finally, the algorithm does an extra step to eliminate stray pixels, by examining a 2×2 neighborhood of each pixel and enforce connectivity. In essence, this step resembles

Algorithm 3.2 SLIC GPU Algorithm

```
procedure SLIC(imInput, nIter)
    src ← DeviceLoad(imInput)
    ConvertColor<perPixel>(src)
    InitClusters<perSuperpixel>( )
    FindClusterAssociation<perPixel>(src)
    for i : (0, nIter) do
        UpdateClusters<perSuperpixel>( )
        FindClusterAssociation<perPixel>(src)
    EnforceConnectivity<perPixel>(src)
    DeviceSync( )
```

mathematical morphology, in an effort to improve the effectiveness of pixel grouping and form high-quality superpixels. As a result, much fewer stray pixel artifacts will appear in the final superpixel segmentation, as shown in Fig. 3.2. Example of this from Fig. 3.2 can be seen among the crowd in the image, where many tiny superpixels have been generated in (a) without connectivity enforcing, but in (b) the superpixels are able to better capture the contour of each person.



(a) (b)

Figure 3.2: gSLIC Post-processing [44] (a) without connectivity enforcement (b) enforcing connectivity

Based on our testing, the algorithm can achieve over 100 fps with 720p an image stream on an NVIDIA Turing architecture GPU. The resulting superpixel segmentation is also more stable and deterministic compared to the OpenCV CPU SLIC

implementation. Additionally, since it generates a more predictable resulting number of superpixels, it is the best fit for neural network based machine learning, which typically has a fixed model structure.

3.3 DIMENSIONALITY REDUCTION IN DNNS AND PGMS

A popular area of DCNN inference that requires the superpixel representation to reduce the computational burden is the advanced Markov Random Fields (MRF) and Conditional Random Fields (CRF) graphical inference models. These models are versatile in addressing various inference demands, especially advanced image-to-image prediction, with unparalleled predictive accuracy. However, without a well informed neighbor vertex set structure, it is common to resort to the computationally intense pairwise features. This is often prohibitively computationally expensive to do on a pixel level, as the partition function or the normalization factor may imply a complex vector space integration and the solution can amount to a cubical-time algorithm. The energy function shown in Eq. (3.11) is one such function. It is only made feasible using a superpixel representation,

$$\begin{aligned} E(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{p} (y_p - z_p)^2 - \frac{1}{2} \sum_{p,q} R_{pq} (y_p - y_q)^2 \\ &= \boldsymbol{y}^T A \boldsymbol{y} - 2\boldsymbol{z}^T \boldsymbol{y} + \boldsymbol{z}^T \boldsymbol{z} \ , \end{aligned} \tag{3.11}$$

where $\boldsymbol{y}$ is a prediction and $\boldsymbol{z}$ is the latent variable and the R matrix incorporates a pairwise similarity metric. This is an example where the MRF [60] can be used to formulate the image-to-image inference as a maximum a posteriori (MAP) solution, which a special case of the Bayes framework [61]. And this is called a MAP-MRF framework, advocated by Geman, et al., since 1984, upon which many statistical image analysis problems were formulated. The MRF consists of an undirected graph, a set of random variables satisfying Markov properties that define how the probability

distribution would factorize in a neighborhood of the graph. One notable variant of MRF is a conditional random field (CRF), in which the response is conditioned upon observations, instead of trying to capture the distribution over the observations as well. Therefore, in the work of [56], deep learning is incorporated and the depth inference framework is based on CRF with a closed form analytical solution, in order to combine these theories with recent successful achievements from deep learning to create a model with better predictive performance and generalization ability.

These graphical models represent the state-of-the-art techniques to address many highly complex image-to-image prediction problems. And this is also as far as we have explored in the field of Superpixel DCNNs in the contemporary computer vision research.

Chapter 4

SUPERPIXEL FEATURE SPACE AND FEATURE EXTRACTION

Superpixel feature extraction is a technique that superpixel-based neural networks rely on heavily. This chapter discusses how superpixel feature extraction will bring together the efficient superpixel representation and popular deep neural network models. And then further explore how this notion can usher in novel approaches to high-level computer vision tasks, especially object detection.

Table 4.1: Symbols and Notation from Chapter 4

cod	the codomain of a function, as per the function definition
ran	the range of a function, which contains actual values the function can generate
$x\|_s$	image function x restricted to subset $s \subseteq \text{dom } x$
h_x	the histogram of image function x, $h_x : \text{cod } x \to \mathbb{Z}^+$
$H(X)$	information entropy of a random variable X
$H(x)$	information entropy of an image, based on the image function x
$H(x\|s)$	superpixel local entropy, see Sect. 4.3
E	the expectation operator
	$\text{ran } x \equiv x(\text{dom } x) \subseteq \text{cod } x$
	$\text{ran } x\|_s = x(s)$ is how restriction affects the range
	$\text{cod } x\|_s = \text{cod } x$ the restriction does not alter codomain
$h_{x\|s}$	the histogram of image function $x\|_s$; s can be viewed as a mask

4.1 HOMOGENIZATION

Homogenization is the process where the superpixel-based image function x_c is determined, i.e. it is where each superpixel is assigned a value of brightness, color or feature vector. The ultimate goal is to achieve an intelligent assignment for each superpixel, such that it is representative of the image object that it would capture, in an effort to promote high-level object detection. Based on the superpixel reconstruction image described in Eq. 3.6, given a superpixel segmentation $c : V \to S$, $x_c : S \to \mathbb{R}$ should be chosen such that $x_c \circ c$ should approximate $x : V \to \mathbb{R}$, the image function which works on some pixel p,

$$x(p) = (x_c \circ c)(p) + \epsilon(p) \ , \tag{4.1}$$

where ϵ is the difference between the two images. The optimal x_c, in the mean-squared-error (MSE) sense, can be derived as follows. Since c is surjective, $\epsilon^2(p)$ can be minimized by considering the subproblem within each superpixel,

$$\forall p_i \in s_j : \sum_i \epsilon^2(p_i) = \sum_i \left(x(p_i) - x_c(s_j)\right)^2 \ . \tag{4.2}$$

Setting the derivative with respect to $x_c(s_j)$ to 0,

$$\frac{\partial \sum_i \epsilon^2(p_i)}{\partial x_c(s_j)} = -2 \sum_i x(p_i) + 2x_c(s_j)|s_j| = 0 \ , \tag{4.3}$$

leads to

$$x_c(s_j) = \frac{\sum_i x(p_i)}{|s_j|} = \frac{\sum_{p \in s_j} x(p)}{|s_j|} \ . \tag{4.4}$$

Much like dimensionality reduction at the first glance, the superpixel feature extraction works by aggregating pixels and assigning one value to each superpixel. This one value per superpixel can be a single number, a color, or a vector of deep neural features. To distinguish this procedure from decision level aggregations that may

occur at later stages of the model, we have adopted the term *homogenization*. In contemporary deep learning research, there exists superpixel pooling, which is an operation that is similar to max-pooling or average-pooling except for the shape of the region they operate on. Technically, homogenization and superpixel pooling are two sides of the same coin, but the two terms bear different connotations and biases. The superpixel pooling layer is utilized in deep convolutional networks where the value for each output pixel is determined by computing the mean value of the region corresponding to that pixel from the input image, then normalized by the area of the region. For superpixel pooling, the goal is to reduce amount of information from convolutional maps generated by a deep convolutional network based on how superpixels are segmented. However, this operation also has the benefit of transforming the image representation in a way that is more advantageous to capture high-level objects for better scene understanding, especially when these high-level object features are well encoded.

4.2 SUPERPIXEL HOMOGENEITY AND RECEPTIVE FIELD

Conventionally, it is common to assume that the superpixel segmentation must be applied only where the encapsulating region is homogeneous [40, 54]. Sect. 3.2.1 has also covered how conventional superpixel usage strongly favors over-segmentation to avoid loss of low-level visual features, such as for preservation of edges. This is also known as the superpixel homogeneity assumption.

However, it is shown [47] that the superpixel homogeneity assumption is not strictly necessary when superpixels are used to highlight various parts of an image object as a means to higher-level image analysis. This idea is inspired by the way popular DCNNs construct and incorporate their visual *receptive fields* [62]. First of all, it is common sense that the DCNN needs to respond to large enough areas in the image in order to capture and process the information about image objects.

The region of the input space that affects a particular unit of the neural network is known as the *receptive field*. The image objects can typically vary in their apparent size due to perspective projection as the image is taken; however, in contrast, the **plausible receptive field** of neuron units in a CNN is very predictable. This means that the CNN must have its way of adapting the effective receptive field to image objects. There are a few popular ways of altering the size of the receptive field, such as stacking convolutional layers, sub-sampling or pooling, and dilated convolution.

In the case of stacking convolutional layers, the receptive field is increased by basing the input of a convolutional layer on the output of another. Because the receptive field is relative to the neuron unit, feeding one convolution layer's output to another as input will, in effect, enlarge the receptive field of the following layer. Therefore, the plausible receptive field can be accounted for by recursively back-tracing the region of input pixels that can affect the output pixel values. However, the **effective receptive field** also depends on the convolutional filter weights because they cause the input pixel to contribute differently to the output, for which, in the work of [62] there is a detailed analysis. This effective receptive field is analogous to the *foveal vision*, where a high level of visual acuity and clarity is attained in a human eye thanks to high-density region of cone cells, versus the *peripheral vision*. As a result, almost all popular DCNNs predominantly use 3×3 filter kernel size, while still being able to perform object detection, localization and segmentation in imagery datasets.

Based on this intuition, superpixel sizes must also significantly affect neuronal receptive fields, which counters the conventional idea of determining superpixel sizes solely based on input image homogeneity. It becomes apparent that the homogeneity assumption should no longer hold in the context of DCNNs, as it imposes unwarranted constraints. In Sect. 4.3, the effect of incorporating heterogeneous superpixels into the DCNN will be quantitatively analyzed. Furthermore, the properties of heterogeneous

superpixels will be explored to better understand what they have to offer for DCNNs.

4.3 LOCAL ENTROPY AND SUPERPIXEL SIZE

As adjusting superpixel size introduces a trade-off between neuronal receptive field size and information loss, it is useful to measure the local *information entropy* to reflect the impact of applying superpixel homogenization. The concept of *entropy* has roots in thermodynamics and statistical mechanics, where it quantifies the "chaoticness" of a physical system. In Shannon Information Theory, it is commonly used as a way to measure the information content, known as the *information entropy*. Its original definition is shown in Eq. 4.5,

$$H(X) \equiv \sum_{k=1}^{n} p(x_k) \log_2 \frac{1}{p(x_k)} = -\sum_{k=1}^{n} p(x_k) \log_2 p(x_k) \geq 0 \ , \tag{4.5}$$

where $H(\cdot)$ is the entropy operator, X is a random variable that takes on value $\{x_1, x_2, ..., x_n\}$, and $p(x_k) = P(X = x_k) \in (0, 1]$ is the probability that X takes value $x_k, 1 \leq k \leq n$. Random variable values that are associated with 0-probability will not contribute to the entropy. The base-2 logarithm suggests that the information entropy is measured in binary *bits*. The probability density $p(x_k)$ needed to calculate the entropy is often estimated using a histogram $h_f : \text{cod } f \to \mathbb{Z}^+$, given some function $f : \cdot \to \{x_1, x_2, ..., x_n\}$ that actually generates the random value for X, as shown in Eq. 4.6,

$$p(x_k) \approx h_f^*(x_k) \equiv \frac{h_f(x_k)}{\sum_{z \in \text{cod } f} h_f(z)} \ , \tag{4.6}$$

where the histogram h_f, as computed by Alg. 4.1, is normalized to a *probability mass function* (PMF), and h_f^* is the short notation for the PMF, such that $\sum_z h_f^*(z) = 1$.

By this estimation, we can extend the meaning of information entropy to image function x,

Algorithm 4.1 Computing Histogram

```
procedure HISTOGRAM(samples)
    for all z ∈ cod f do                ▷ Assume bins are determined by cod f
        h_f(z) ← 0
    for all z ∈ samples do
        h_f(z) ← h_f(z) + 1
    return h_f
```

$$H(x) \equiv -\sum_{z \in \operatorname{ran} x} h_x^*(z) \log_2 h_x^*(z) \ , \tag{4.7}$$

and also find the superpixel local entropy, substituting $x|_s$ for x,

$$H(x|s) = -\sum_{z \in \operatorname{ran} x} h_{x|s}^*(z) \log_2 h_{x|s}^*(z) \ . \tag{4.8}$$

Theorem 4.3.1 (Decomposition Theorem)**.**

$$\begin{aligned} H(x) &= H(\mathcal{S}(c)) + \mathit{MSLE}(x) \\ &= -\sum_{s \in S} P(c(\Pi) = s) \log_2 P(c(\Pi) = s) + \sum_{s \in S} \frac{|s|}{|V|} H(x|s) \\ &= -\sum_{s \in S} \frac{|s|}{|V|} \log_2 \frac{|s|}{|V|} + \sum_{s \in S} \frac{|s|}{|V|} H(x|s) \ . \end{aligned} \tag{4.9}$$

where $H(\mathcal{S}(c))$ is the superpixel assignment entropy, i.e. information entropy of any pixel belongs to one of the superpixels, which can also be described in terms of the probability that a random pixel $\Pi \in V$ also belongs to superpixel s; and this probability is found to be the ratio of the superpixel area to the area of the entire image. The second term is the *Mean Superpixel Local Entropy* (MSLE), which is an average of local entropies weighted by superpixel area. This theorem describes the relation between the image entropy $H(x)$, superpixel assignment entropy $H(\mathcal{S}(c))$ and the superpixel local entropies $H(x|s)$. Namely, the total image entropy consists of the entropy from the superpixels and within each superpixel. When the superpixel

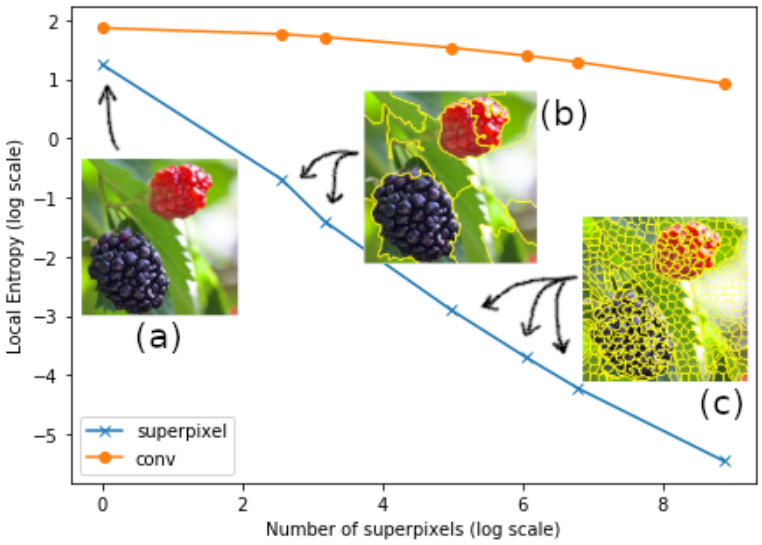


Figure 4.1: MSLE (log scale): The blue line (cross-points) indicates the mean of the superpixel entropy, while the orange line (round-dot-points) indicates a local entropy calculated similar to convolution where the local neighborhood is defined by a square window.

segmentation is generated with an appropriate distance metric, the superpixel assignment entropy $H(\mathcal{S}(c))$ should become the dominant term in the decomposition, while the MSLE would decrease effectively.

Therefore, the loss of information can be measured by comparing the image entropy and the MSLE. Because after homogenization, the information within each superpixel will be lost as it will only be assigned one value, the lower MSLE translates to the less total information loss by homogenization. To evaluate this effect, we can construct superpixel segmentations with various levels of superpixel sizes and observe how the MSLE is affected. The MSLE, in this example, is measured by creating a 256-bin hue histogram with respect to each superpixel region. It has been compared with the local entropy associated with the same-area receptive field of a convolutional operation, which characterizes the information content as received by typical CNNs. The convolution receptive field sizes have been adjusted by aligning the convolutional kernel size to the superpixel size according to pixel area. From

Fig. 4.1, it is evident that, in the context of superpixel segmentation, the MSLE exhibits a strong anti-correlation with respect to the number of superpixels, as shown in Eq. 4.10,

$$\mathrm{MSLE}(x) = \sum_{s \in S} \frac{|s|}{|V|} H(x|s) \approx |S|^{-1} \cdot H(x) \ . \tag{4.10}$$

Empirically it can be found that Eq. 4.10 approximately describes the graph shown in Fig. 4.1 across a broad range of number of superpixels evaluated. The multiplication factor $|S|^{-1}$ has shown the effect of reducing the entropy of superpixel feature organization; and it is strongly dependent on the number of superpixels used in image resolution. By contrast, the convolutional local entropy decreases very minimally as the receptive field is increased, which also explains why contemporary DCNNs all prefer very small kernels to enable a more compact encoding of the visual features.

4.4 HETEROGENEOUS SUPERPIXEL FEATURES

Now that a high-level picture of the capabilities of the heterogeneous superpixel has been painted through an information-theoretical analysis, in this section, we will discuss some of the novel techniques that can be applied to constructing superpixel-enabled DNNs and incorporating heterogeneous superpixel features.

4.4.1 DCNN Feature Inheritance

From the superpixel local entropy analysis (Sect. 4.3), we have learned that the information loss would be minimal if we incorporate heterogeneous superpixel in DNNs. However, an intelligent assignment of features is still required to fully harness the power of superpixel DNN. A good starting point without the effort of feature engineering from square-one would be to inherit it from state-of-the-art DCNNs.

A general structure of an image classification DCNN can typically be divided

into two parts: firstly, a stack or ensemble of layers of convolution and/or pooling; secondly, a stack of dense layers which resembles *Multi-layer Perceptron* (MLP). The first part can be viewed as a learnable visual feature extraction component, whereas the second part learns and handles the class feature encoding and the decision-making for the final class prediction. Between the two parts, there will be a global pooling or flattening layer, that will transform the 2-D image feature into a spatially 1-D vector representation.

As mentioned, the superpixel feature extraction requires a 2-D topology as input. This makes any DCNN feature extraction intermediate output, $\boldsymbol{o}$, also known as *convolutional maps*, an admissible input. The collection of convolutional maps $\boldsymbol{o} : V' \to \mathbb{R}^K$ fits in the definition of multi-channel image function, where V' is a 2-D topological space of the convolutional map, as distinct from V, K is the number of feature channels. For simplicity of the discussion, we will consider a single channel convolutional map $o : V' \to \mathbb{R}$ first.

The superpixel segmentation is computed using the original input image in order to maximize the resolution and preserve as much image detail as possible. Consequently, the DCNN convolutional maps will have much less resolution than the superpixel segmentation. In contemporary DCNN-based computer vision [6, 7, 8, 9, 10], it is very common to apply *differentiable bilinear up-sampling* or *transposed convolution* [6, 31] to reconcile the resolution difference. However, this conflict can be addressed by associating each pixel in the convolutional map to a tile of pixels from the superpixel segmentation [56]. This approach's validity in DCNN architectures is based on the analysis in Sect. 4.3.

4.4.1.1 Tiled Image Function to Reconcile Resolutions

Suppose we can cover the input image with a grid of tiles. The precise association between the image pixels and these tiles can be found as a function $c_0 : V \to T$ that

Table 4.2: Tiling Concepts from 0-Iteration Superpixel Segmentation

Tile	**Description**	**Superpixel**
T	the set of tiles	S
$t_j \in T$	a tile that belongs to the tile set	$s_j \in S$
$c_0(p_i) = t_j \in T$	pixel-tile association	$c(p_i) = s_j \in S$
$x_{c_0} : T \to \mathbb{R}$	tiled image function	$x_c : S \to \mathbb{R}$
$x_{c_0} \circ c_0 : V \to \mathbb{R}$	tiled "reconstruction"	$x_c \circ c : V \to \mathbb{R}$
τ	tile interval	ρ
$x_{c_0} \circ t : \mathbb{Z} \to \mathbb{R}$	a map from tile index to value assignment only used in Eq. (4.20)	

surjectively maps pixels to tiles, where T represents the set of these tiles. In fact, this tiling can be obtained without extra computation, as it has already made an appearance in Alg. 3.1 as the initial grid of superpixel segmentation, as the result of the "InitClusters" procedure. In other words, if we label the intermediate segmentation result in the Alg. 3.1 main iterations as $\{c_0, c_1, ..., c_n\}$, where n is the fixed number of iterations, this grid of tiles is described by c_0. Therefore, a tiled image function $x_{c_0} : T \to \mathbb{R}$ serves to represent the up-scaled image that reconciles the resolution difference. The term "up-scale" is distinct from "up-sample" in a way that the former is in the form of pixel association, without any interpolation as an up-sampling algorithm would incur. Table 4.2 shows the conceptual mapping between the tiling and the superpixel definitions.

As a result, based on the concepts defined in Tab. 4.2, the mapping between tiles and superpixels can be derived. To achieve DCNN feature inheritance, we require a solution to ϕ_o that re-assigns x_c in terms of the convolutional feature map o,

$$\begin{aligned} x_c(s_j) &= \frac{\sum_{t_q \in T} \sum_{p \in (s_j \cap t_q)} (x_{c_0} \circ c_0)(p)}{|s_j|} \\ &= \frac{\sum_q |s_j \cap t_q| \cdot x_{c_0}(t_q)}{|s_j|} \\ &= \frac{\sum_q |s_j \cap t_q| \cdot \phi_o(q)}{|s_j|} \ . \end{aligned} \tag{4.11}$$

The multiplier $|s_j \cap t_q|$ is accounting for repeated pixel value assignment as the effect of homogenization, and replaces the inner summation.

4.4.1.2 Mapping Pixels to Tiles

As shown in Table 4.2, Eq. (4.12) describes the function c_0 which maps image pixels to tiles,

$$\forall p_i \in V, \exists j : c_0(p_i) = t_j \in T \ . \tag{4.12}$$

Without loss of generality, suppose these are $\tau \times \tau$ sized square tiles. In the context of a 2-D digitized $m \times n$ image, i.e. constraining

$$\begin{aligned} V &= \left\{(i,j), 1 \leq i \leq m, 1 \leq j \leq n, m \in \mathbb{Z}^+, n \in \mathbb{Z}^+\right\} \subseteq (\mathbb{Z}^+ \times \mathbb{Z}^+) \ , \\ V' &= \left\{(i,j), 1 \leq i \leq \lfloor m/\tau \rfloor, 1 \leq j \leq \lfloor n/\tau \rfloor, m \in \mathbb{Z}^+, n \in \mathbb{Z}^+\right\} \subseteq (\mathbb{Z}^+ \times \mathbb{Z}^+) \ , \end{aligned} \tag{4.13}$$

the tiling function $c_0 : V \rightarrow T$ can be implemented by introducing a coordinate transformation $\zeta_{mn} \circ \delta_\tau : \mathbb{Z}^+ \times \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$,

$$\forall p \in t_q : q = (\zeta_{mn} \circ \delta_\tau)(p) \ , \tag{4.14}$$

given

$$\begin{aligned} \zeta_{mn}(i,j) &= i \times n + j \ , \\ \delta_\tau(i,j) &= (\lfloor i/\tau \rfloor, \lfloor j/\tau \rfloor) \ . \end{aligned} \tag{4.15}$$

Now the tile assignment function c_0 can be decomposed into

$$c_0 = t \circ \zeta_{mn} \circ \delta_\tau \ , \tag{4.16}$$

given $t(q) \equiv t_q \in T$.

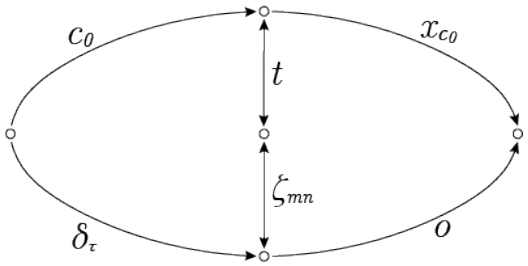


Figure 4.2: Tile Up-scaling $c_0 : V \rightarrow T$ Decomposition

Lemma 4.4.1. *The coordinate transform implies the tile value assignment* $x_{c_0}(t_q)$ *is determined by the pixel values* $(x_{c_0} \circ c_0)(p)$*,*

$$\begin{aligned} \forall p \in t_q : (x_{c_0} \circ c_0)(p) &= (x_{c_0} \circ (t \circ \zeta_{mn} \circ \delta_\tau))(p) && \textit{(Eq. 4.16)} \\ &= (x_{c_0} \circ t)((\zeta_{mn} \circ \delta_\tau)(p)) \\ &= (x_{c_0} \circ t)(q) && \textit{(Eq. 4.14)} \\ &= x_{c_0}(t(q)) \\ &= x_{c_0}(t_q) \ . \end{aligned}$$

Lemma 4.4.2. *Additionally, the inverse of* ζ_{mn} *exists,*

$$\zeta_{mn}^{-1}(q) = (q/n, q \ mod \ n) \ .$$

4.4.1.3 The Imagery Up-scaled Feature Map

Definition 4.4.2.1 (Up-scaled DCNN Feature Map)**.**

$$\forall p \in t_q : (x_{c_0} \circ c_0)(p) \equiv (o \circ \delta_\tau)(p) \tag{4.17}$$

This assignment to the imagery object $(x_{c_0} \circ c_0)$ is valid due to t and ζ_{mn} both being bijective,

$$
\begin{aligned}
x_{c_0}(t_q) &\equiv (o \circ \zeta_{mn}^{-1})(q) \quad t_q \in T \\
(x_{c_0} \circ t \circ \zeta_{mn})(p) &= o(p) \qquad\qquad p \in V' \\
(x_{c_0} \circ (t \circ \zeta_{mn} \circ \delta_\tau))(p) &= (o \circ \delta_\tau)(p) \qquad p \in V
\end{aligned}
$$

As a result of Lemma 4.4.1 and Eq. (4.17), the solution to DCNN feature inheritance ϕ_o can be determined by a dense assignment of the up-scaled feature map $o \circ \delta_\tau$, as shown in Eq. 4.18,

$$
\begin{aligned}
\phi_o(q) &= \sum_q (x_{c_0} \circ t)(q) \\
&= \sum_{p \in t_q} (x_{c_0} \circ c_0)(p) \\
&= \sum_{p \in t_q} (o \circ \delta_\tau)(p) \ ,
\end{aligned}
\tag{4.18}
$$

and Eq. (4.11) can be completed as,

$$
x_c(s_j) = \frac{\sum_q \left(|s_j \cap t_q| \cdot \left(\sum_{p \in t_q} (o \circ \delta_\tau)(p) \right) \right)}{|s_j|} \ . \tag{4.19}
$$

However, by introducing a functional association, we can avoid obtaining a copy of the tiling or any up-scaling altogether. Instead, we find an appropriate substitution value directly from the convolutional map o.

Theorem 4.4.3 (Up-scaling Substitution Theorem)**.** *Subject to the tiling constraint Eq. 4.13, the tile value is determined by the convolutional feature map,*

$$
x_{c_0} \circ t = o \circ \zeta_{mn}^{-1} \ . \tag{4.20}
$$

Proof: Considering any pixel p within each tile t_q,

$$
\begin{aligned}
\forall p \in t_q : (x_{c_0} \circ t)(q) &= (x_{c_0} \circ c_0)(p) && \text{(Lemma 4.4.1)} \\
&= (o \circ \delta_\tau)(p) && \text{(Eq. 4.17)} \\
&= (o \circ (\zeta_{mn}^{-1} \circ \zeta_{mn}) \circ \delta_\tau)(p) \\
&= (o \circ \zeta_{mn}^{-1})((\zeta_{mn} \circ \delta_\tau)(p)) \\
&= (o \circ \zeta_{mn}^{-1})(q) && \text{(Eq. 4.14)}
\end{aligned}
\tag{4.21}
$$

As a result, the solution to DCNN feature inheritance ϕ_o can be efficiently determined by $\phi_o(q) = (o \circ \zeta_{mn}^{-1})(q)$, and Eq. 4.11 can be completed as,

$$
\begin{aligned}
x_c(s_j) &= \frac{\sum_q |s_j \cap t_q| \cdot x_{c_0}(t_q)}{|s_j|} \\
&= \frac{\sum_q |s_j \cap t_q| \cdot (o \circ \zeta_{mn}^{-1})(q)}{|s_j|} \\
&= \sum_q \frac{|s_j \cap t_q|}{|s_j|} (o \circ \zeta_{mn}^{-1})(q) \ .
\end{aligned}
\tag{4.22}
$$

Finally, in DCNNs, the superpixel feature map $\boldsymbol{x}_c : S \to \mathbb{R}^K$ and convolutional feature map $\boldsymbol{o} : V' \to \mathbb{R}^K$ are both multi-channel image functions, where K is the number of image channels. They can be thought of as a collection of functions, $\{x_c^1, x_c^2, ...x_c^K\}$ and $\{o^1, o^2, ...o^K\}$. Therefore,

$$
x_c^k(s_j) = \sum_q \frac{|s_j \cap t_q|}{|s_j|} (o^k \circ \zeta_{mn}^{-1})(q) \, , k \in [1, K] \ . \tag{4.23}
$$

4.4.2 Cross-resolution Feature Integration

The requirement for feature aggregation across feature maps from different resolutions is not only a practical issue, in the research of many popular semantic segmentation networks, such as U-Net [10] and RetinaNet [63], it can also be shown that a more robust model can be created to recognize imagery objects in various scales thanks to

such a feature aggregation.

Consider a set of N convolutional feature maps $\{\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_N\}$ that have different resolutions $(m_1, n_1), (m_2, n_2), ..., (m_N, n_N)$ and numbers of channels $K_1, K_2, ..., K_N$. As an extension to Eq. (4.22), the superpixel feature map can be derived from the set of convolutional feature maps as the following,

$$\boldsymbol{x}_c^r(s_j) = \sum_q \frac{|s_j \cap t_q|}{|s_j|} \boldsymbol{o}_r(\zeta_{m_r,n_r}^{-1}(q)) \, , r \in [1, N] \, , \tag{4.24}$$

where $\boldsymbol{x}_c^r : S \to \mathbb{R}^{K_r}$ is the superpixel feature generated from $\boldsymbol{o}_r$, and all of the $\boldsymbol{x}_c^r, r \in [1, N]$, can be concatenated together to form a vector function $\boldsymbol{x}_c : S \to \mathbb{R}^{\mathcal{K}}$, where $\mathcal{K} \equiv \sum_{i=1}^{N} K_i$. Therefore, the superpixel feature extraction is capable of associating feature from multiple feature maps with different resolutions, thanks to theorem 4.4.3 and the function $\zeta_{m_r,n_r}^{-1} : [1, |S|] \to V_r'$ which establishes the direct access to convolutional feature map pixel, from V_r', regardless of the resolution. The issue of putting these features together is reduced to vector concatenation, which generates a feature space that encodes visual features that encompass various receptive field sizes.

4.4.3 Optimizations

Recall Eq. (4.23),

$$x_c^k(s_j) = \sum_q \frac{|s_j \cap t_q|}{|s_j|} (o^k \circ \zeta_{mn}^{-1})(q) \, ,$$

is in the form of a dot-product over index q, we can rewrite the equation using linear algebra.

Definition 4.4.3.1 (Superpixel Feature Matrix)**.**

$$\chi_{jk} = x_c^k(s_j) \, .$$

Definition 4.4.3.2 (Superpixel Frequency Matrix)**.**

$$A(S,T)_{jq} = |s_j \cap t_q| \ . \tag{4.25}$$

Definition 4.4.3.3 (Flattened Convolutional Features)**.**

$$O_{qk} = o^k(\zeta_{mn}^{-1}(q)) = o^k(q/n, q \ mod \ n) \ .$$

Now Eq. (4.23) is transformed into,

$$\chi = \nu_2(A(S,T)) \cdot O \ , \tag{4.26}$$

where ν_2 represents normalization over the second axis, such that $\sum_q \nu_2(A_{jq})$ is an $|S|$-vector of ones. This is beneficial because: (a) there are many sophisticated ways to accelerate linear algebra computation; and (b) learning frameworks like TensorFlow [64] can automatically compute the gradient of variables in a matrix multiplication. And all that is left is the computation of the *Superpixel Frequency Matrix* $A(S,T)$, which is a $|S| \times |T|$ matrix.

Optimizing Normalization

Firstly, the normalizer is an $|S|$-vector which can be obtained using a matrix-vector product with a $|T|$-vector of ones. Then an element-wise division can be computed to derive $\nu_2(A(S,T))$, as shown in Eq 4.27,

$$\nu_2(A(S,T))_{jq} = \frac{A(S,T)_{jq}}{\Big(A(S,T) \cdot \vec{1}(|T|)\Big)_j} \ , \tag{4.27}$$

Both operations can leverage cuBLAS to be accelerated on GPU. The normalizer and the vector of ones will require extra space allocation, but $\nu_2(A(S,T))$ can be

computed in the space of $A(S,T)$.

Optimizing Superpixel Frequency Matrix

The computation of $A(S,T)$ is the most essential part in the superpixel feature extraction, and it is the most complex because the superpixels can appear to be any arbitrary closed and connected region. The only way to compute this is by brute-force, and using a parallel GPU program helps greatly.

$A(S,T)$ represent the pairwise overlapping area between superpixels and tiles, and we can specify it using the set notation,

$$\begin{aligned} A(S,T)_{jq} &= |s_j \cap t_q| \\ &= |\{p \in V | c(p) = j\} \cap t_q| \\ &= |\{p \in t_q | c(p) = j\}| \qquad (t_q \subseteq V) \\ &= \sum_{p \in t_q} [c(p) = j] \ . \end{aligned} \tag{4.28}$$

Approach 1: SIMD

The tiling can be handled as a block matrix, where summation over $p \in t_q$ can be expressed as summation over each matrix block. Considering the shape of a multi-dimensional array is determined by stride, which defines number of elements to skip before obtaining the next element along certain axis, it is possible to reshape the segmentation suggested by c into a rank 6 tensor, tiling all 3 axes, by a mere change of metadata. Then this summation can be computed by broadcasting the equal predicate and then summing over last 3 tile axes. Alg. 4.2 is a vectorized implementation of equation 4.28 that aims to utilize *Single Instruction Multiple Data* (SIMD) as means to accelerate the computation, limitation being the high space complexity of $O(|S| \times |V|)$. The time complexity is high mainly due to broadcasting the equal predicate.

Algorithm 4.2 Overlapping area between superpixels and tiles

procedure SUPERPIXELFREQ(segments, shapeConvMap, nSegments)
  $subsample \leftarrow \lfloor segments.shape / shapeConvMap \rfloor$
  $w \leftarrow$ RESHAPE($segments$, $(1,) + segments.shape$)
  $z \leftarrow$ ARANGE($nSegments$)
  $w \leftarrow w ==$ RESHAPE(z,($nSegments, 1, 1$)) ▷ broadcasting equal predicate
  $w \leftarrow$ VIEWASBLOCK(w, $(1,) + subsample$) ▷ w is rank 6 tensor now
  return SUM(w, $axis = (3, 4, 5)$)

Approach 1: CUDA Kernel on Output Space

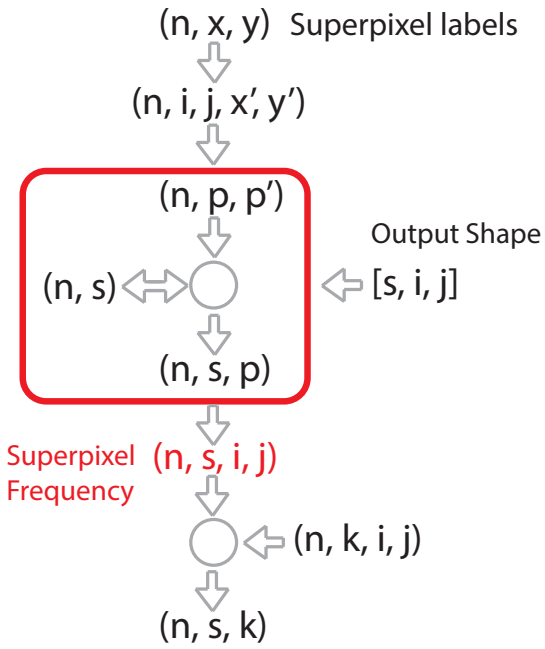


Figure 4.3: Dataflow in Superpixel Feature Extraction. $[s, i, j]$ is a tuple that contains exactly those 3 numbers, to specify the output shape. (n, s) is the only extra space allocation in this CUDA implementation of superpixel feature extraction, storing the normalizing factors $\nu_2(A(S, T))$.

To cut down the space allocated for storing the result of broadcasting the equal predicate, it can be evaluated in parallel and only reside in streaming processor's register as a local variable. Namely, $[c(s) = j]_{|S|\times|V|}$ will no longer reside in memory, and not all of its values will be present at the same time. This reduces space complexity down to $O(|S| \times |T|)$. Alg. 4.3 has provided a parallel algorithm to compute equation 4.28. Additionally, Fig. 4.3 has shown a dataflow diagram that illustrates shape

changes of data during superpixel feature extraction. A CUDA implementation of the superpixel area computation as described in Alg. 4.3 is highlighted in a red box from data perspective of view. Advantages of this parallel implementation described in Alg. 4.3 are threefold in space, time and extensibility.

Algorithm 4.3 Overlapping area between superpixels and tiles

procedure SUPERPIXELAREA(segments, shapeConvMap, nSegments)
  Infer output space A_{jq} from $shapeConvMap, nSegments$
  $sum \leftarrow 0$
  if $(j, q) \in$ output space **then** ▷ thread ID (j, q) is assumed
    for all $p \in t_q$ **do**
      if $c(p) == j$ **then**
        $sum \leftarrow sum + 1$
  $A_{jq} \leftarrow sum$

Name	Shape	Description
Input (segments)	(N, H, W)	TensorFlow tensor
Block view	$(N, H/\tau, W/\tau, \tau, \tau)$	CUDA (in-place)
Flattened superpixel view	$(N, (H/\tau) \times (W/\tau), \tau^2)$	(in-place)
L_1 norms	$(N, \lvert S\rvert)$	CUDA
Flattened superpixel frequency	$(N, \lvert S\rvert, (H/\tau) \times (W/\tau))$	(in-place)
Superpixel frequency	$(N, \lvert S\rvert, H/\tau, W/\tau)$	TensorFlow tensor

Table 4.3: Data structures in superpixel feature extraction

Comparing Alg. 4.3 to Alg. 4.2, a rank 6 tensor got removed, which used to store information of $[c(s) = j]$ with shape (superpixels, tile-row, tile-col, 1, offset-row, offset-col), where "offset" are coordinates relative to each tile, which is bijectively associated to t_q. The space complexity of this tensor is $O(|S| \times |V|)$.

From an extensibility point of view, this CUDA implementation of superpixel feature extraction would enable TensorFlow shape inference[64], as opposed to deciding shape of data structures inside superpixel feature extraction layers before any training or prediction due to technical difficulties in expressing superpixel completely with matrix operations; which in turn, would constrain shape of input images while

requiring the part that cannot be easily computed with matrix operations to be pre-processed and persisted on disk. Therefore, by implementing Alg. 4.3 with CUDA, the model is able to compute superpixel feature extraction directly from input image and superpixel segments and adapt to input image resolution.

Approach 2: CUDA Kernel on Input Space

Recall the tensor $[c(s) = j]_{|S|\times|V|}$ that was optimized away. Not only does it take a lot of space, it is also very sparse because assuming a uniform distribution $P(s_j) = 1/|S|$,

$$\begin{aligned} P(c(s) = j) &= \frac{\mathrm{E}|s|}{|V|} \\ &= \frac{\sum_j |s_j| P(s_j)}{|V|} \\ &= \frac{\sum_j |s_j| \frac{1}{|S|}}{|V|} \\ &= \frac{1}{|S|} \frac{\sum_j |s_j|}{|V|} \\ &= \frac{1}{|S|} , \end{aligned} \tag{4.29}$$

where $|S|$ is number of superpixels. To put this into perspective, if 600 superpixels were typically generated, only 16.7% of the space would be non-zero. And if input images were in resolution 1280x768, then around 563 MiB of memory would be occupied per image with around 469 MiB of memory filled with just zeros. Alg. 4.3 fixes this problem by not storing the result of $[c(s) = j]$.

However, this sparsity has also suggested that *atomics*, i.e. atomic read-modify-write operations on GPU, can be efficiently utilized to handle the output space race condition, as shown in Alg 4.4. This implementation further optimizes time efficiency, because it will not loop over zero-valued pixels. It also simplifies the program.

Algorithm 4.4 Overlapping area between superpixels and tiles

procedure SUPERPIXELAREA(segments, shapeConvMap, nSegments)
  Infer input space $(m \times n)$ from *segments*
  if $p = (i, j) \in$ input space **then** ▷ thread ID (i, j) is assumed
    $z \leftarrow c(p)$
    $q \leftarrow \zeta_{mn}(\delta_\tau(p))$
    ATOMICADD(A_{zq}, 1)

4.5 NOVEL ARCHITECTURE AND INFERENCE MODELS

This chapter has provided a comprehensive description of the computation of the superpixel feature extraction. There are two levels of algorithmic optimization in this particular operation that lead to its high performance. Firstly, this operation is equivalent to an up-scaling, followed by an irregular average pooling, but our design of the operation has fused the two steps mathematically, whose correctness is provided by the Up-scaling and Substitution Theorem 4.4.3. Secondly, the design is implemented using GPGPU parallel programming, which make it possible to keep up with over 50 fps frame rate of real-time image processing. The fused computation is an especially noteworthy feature, which also provides a way to circumvent the up-sampling operation, as a very common operation that appears in image segmentation networks. In addition, our approach is done in a way such that the information loss is minimal from the information-theoretical point of view, as discussed in Sect. 4.3.

This means that to aggregate features across feature maps from different layers or receptive field scales of the neural network is now feasible in this entirely new approach. The requirement for feature aggregation across different resolutions is not only a practical issue in creating novel neural architectures, but also an important approach to create a more robust model can be created to recognize imagery objects in various scales. And this has been demonstrated in many segmentation networks, such as U-Net [10], RetinaNet [63].

The aggregation can be done in two different ways. Fig. 4.4 is an illustration of

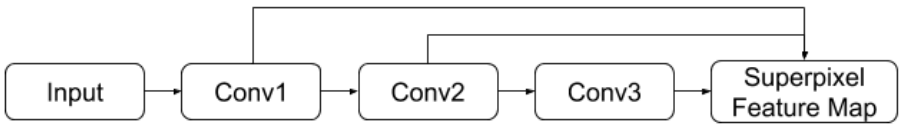


Figure 4.4: Superpixel Feature Integration across Scales

superpixel feature integration across various scales, whereas Fig. 4.5 is an illustration of superpixel feature integration across various superpixel sizes. Both can be described in a grouped superpixel feature extraction formula with a final concatenation along the two respective dimensions.

Consider a group of N convolutional feature maps $\{\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_N\}$ that have different resolutions $(m_1, n_1), (m_2, n_2), ..., (m_N, n_N)$ and numbers of channels $K_1, K_2, ..., K_N$ and different superpixel sizes, resulting in superpixel sets $S_1, S_2, ..., S_N$. The grouped superpixel feature map can be derived from the set of convolutional feature maps as the following,

$$\boldsymbol{x}_c^g(s_j^g) = \sum_q \frac{|s_j^g \cap t_q|}{|s_j^g|} \boldsymbol{o}_g(\zeta_{m_g,n_g}^{-1}(q)) \, , g \in [1, N] \, , s_j^g \in S_g \ , \tag{4.30}$$

where $\boldsymbol{x}_c^g : S_g \to \mathbb{R}^{K_g}$ is the superpixel feature generated from $\boldsymbol{o}_g$.

In the case of feature integration across scales,

$$S_g = S \, , g \in [1, N] \ ,$$

and all of the $\boldsymbol{x}_c^g, g \in [1, N]$, can be concatenated together to form a vector function $\boldsymbol{x}_c : S \to \mathbb{R}^{\mathbb{K}}$, where $\mathbb{K} \equiv \sum_g K_g$.

Therefore, the superpixel feature extraction can associate features from multiple feature maps with different resolutions using vector concatenation. This generates a feature space that encodes visual features that encompass multiple receptive field sizes.

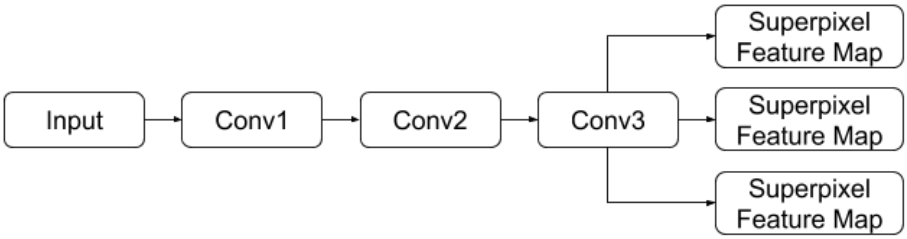


Figure 4.5: Superpixel Feature Integration across Superpixel Sizes

In the case of feature integration across superpixel sizes,

$$K_g = K \, , g \in [1, N] \, ,$$

and all of the $\boldsymbol{x}_c^g, g \in [1, N]$, can be concatenated together to form a vector function $\boldsymbol{x}_c : \mathbb{S} \rightarrow \mathbb{R}^K$, where $\mathbb{S} \equiv \prod_g S_g$, as the vector space concatenation of all $S_g, g \in [1, N]$.

By jointly considering superpixel feature maps from different superpixel sizes, this generates a feature space that encodes a variety of superpixel feature space organization.

Chapter 5

WEAKLY-SUPERVISED LEARNING AND HIGH-LEVEL OBJECT DETECTION

Contemporary robust and highly accurate object detection models rely on data augmentation and transformation-invariant features. CNNs are inherently limited to model geometric transformations due to the built-in convolutional kernels and network architecture [13]. Whereas, creating a high-level object detector is a demanding research objective that leads to more generalizable abstraction of the object. In this chapter, we primarily discuss high-level object detection with weakly-supervised learning that we derived from the superpixel methodologies.

5.1 WEAKLY SUPERVISED OBJECT DETECTION (WSOD)

A notable trend in the deep learning literature is that object detection and semantic segmentation are typically treated as completely different tasks, when in fact these two visual tasks are very similar in nature. When dealing with imagery, object detection requires region-level class labels. These labels utilize meta information, such as rectangular bounding boxes, to represent the region of interest (ROI) in which the object occupies. Common models that rely on the labeled regions include Faster R-CNN [4], YOLO [5], SSD [18], and CenterNet [15]. While this approach is compact and efficient, it does not provide a precise boundary or contour of the object. On the other hand, the state-of-the-art semantic segmentation models (e.g. Fully Convolutional

Table 5.1: Symbols and Notation from Chapter 5

$\mathbb{R}^+$	The set of positive real numbers with also zero.
$\mathbb{I}_K$	The $K \times K$ identity matrix.
$\mathbf{1}_K$	The column vector in $\mathbb{R}^K$ that is filled with ones.
$\lvert\cdot\rvert$	The cardinality of a set, also known as the number of elements for finite sets.
$\lVert\cdot\rVert$	The magnitude or the L^2-norm of a vector; or the Frobenius norm on matrices.
V_0	The entire pixel set of an image.
V_1	The initial set of heterogeneous superpixels as pixel sets. Additionally, $s = \lvert V_1\rvert$ is the number of superpixels.
V_2	The set of object proposals as pixel sets.
$\boldsymbol{C_0} \in \mathbb{R}^{\lvert V_0\rvert \times \lvert V_1\rvert}$	The assignment matrix of pixels into superpixels.
$\boldsymbol{C_1} \in \mathbb{R}^{\lvert V_1\rvert \times \lvert V_2\rvert}$	The assignment matrix of superpixels into object proposals.
$\boldsymbol{X_0} \in \mathbb{R}^{\lvert V_0\rvert \times K}$	The DCNN features spatially flattened.
$\boldsymbol{X_1} \in \mathbb{R}^{\lvert V_1\rvert \times M}$	The superpixel features.
$\boldsymbol{X_2} \in \mathbb{R}^{\lvert V_2\rvert \times M}$	The object proposal features.
$\boldsymbol{M}_i^{jk}$	The element at the j-th row and the k-th column of the matrix $\boldsymbol{M}_i$.
Bold lower case letters represent a vector, e.g. $\boldsymbol{u} \in \mathbb{R}^n$.	
Bold upper case letters represent a matrix, e.g. $\boldsymbol{W} \in \mathbb{R}^{m \times n}$.	
Non-bold lower case letters represent a scalar or a function.	
Non-bold upper case letters represent a set.	

Object Detection Models		Ground Truth Granularity		
		Image	Region	Pixel
Prediction Granularity	Image	Classification	N/A	N/A
	Region	WSOD	Object Detection	N/A
	Pixel	Precise WSOD	Precise WSOD	Semantic Segmentation

Figure 5.1: Object Detection Models. The popular fully supervised DCNN image object detection tasks are on the diagonal line, whereas the weakly-supervised models are below the diagonal line.

Network (FCN) [6], U-NET [10]) are able to produce a precise object boundaries, but they also require pixel-level annotated training datasets. By comparing and contrasting these two popular paradigms, we can see that not only is it theoretically desirable to create a more universal model that bridges the gap between different imagery deep learning task domains, but it also presents as a significant advantage if image classification datasets can be used for segmentation network training, based on the fact that there are far more image-level annotated image datasets, such as [65, 66, 67], than large-scale ROI-level or pixel-level annotated datasets [1, 2, 3] available, as the former are far less expensive to acquire. And the number of these coarsely-labeled datasets will grow faster than that of the finely-labeled datasets. Fig. 5.1 has shown a classification of various types of object detection models by granularity, among which WSOD models characterized by the weak supervision are below the diagonal line.

Therefore, despite the challenges, many researchers are motivated to explore the methods for the Weakly Supervised Object Detection (WSOD) problem that targets training an object detector using coarsely annotated datasets, i.e. with image-level or ROI-level annotations. As a WSOD model, it will not have sufficient data to learn from each individual training data sample directly, but more often the hidden causal relation between the specific image components and their object classes can only be inferred from many examples in the training data, presented to the model over time. This leads to DCNN architectures whose top-level inference layers incorporate

Multiple Instance Learning (MIL), such as [68, 69]. Furthermore, atop this intrinsic ambiguity from the WSOD problem formulation, we would like to derive precise object boundary. Although that may seem paradoxical, by jointly considering low-level and high-level features, the object boundary, the hidden object-class causal relation from MIL, and the object class probability density can be all related. Specifically, the object boundary can be retained through a low-level oversegmentation with high edge recall, while given the object proposals, the high-level hidden relation between image components and their class labels can be resolved using MIL. In this chapter, we present the heterogeneous superpixel, as a learnable feature extraction technique to reorganize DCNN features and reconcile the low-level and high-level image features to generate object proposals. These heterogeneous superpixels are initialized from a low-level oversegmentation and evolved and integrated according to high-level features from the DCNN as well as proposal generation feedback from the MIL layers.

WSOD aims to achieve object detection and localization with lower training data requirements, specifically, forgoing pixel-level class labels. This will make coarsely annotated datasets available for training in object detection, localization, segmentation, and related task settings, not just image classification. This type of model construction has gained traction in the most recent computer vision research because of its practicality, as well as its potential of deriving explainable AI (XAI) algorithms, which has piqued researchers' interest. Traditionally, there are several well-known models that can achieve this goal, including objectness with iterative conditional random field (CRF) localization [70], Bayesian latent models [71], and most notably the MIL models [68, 69, 72, 73]. With the recent advancement of DCNNs, a WSOD model can also be created from off-the-shelf feature extractors with transfer learning [72, 73, 74, 75, 76, 77]. They have all demonstrated that the DCNN features can significantly boost the performance over hand-crafted features.

Early work [69, 78, 79] typically involved human interaction and was not fully

automated, while the recent methods [47, 68, 80, 81] favor end-to-end trainable network architectures, as they have the advantage of being jointly optimized to facilitate a streamlined error back-propagation. These models share something in common, as they generate proposal regions and evaluate the probability density across object classes. This quantity can be seen as a proposal classification confidence score or a soft relational mapping [47] between the proposal and image object vector encoding. For the purpose of MIL, the proposal scores are then aggregated into an image-level classification score [68] to formulate the final objective function, suitable for image-level supervised training data.

With regard to end-to-end WSOD models that leverage DCNN-derived features, it is a typical strategy to associate the convolutional features with specific candidate regions and train a WSOD detector based on these candidate regions with local features. Such candidate regions are also known as the *proposals*. MIL is typically used as the top-level WSOD detector [68, 80, 81]. In an object detector, rectangular bounding boxes are typically used to represent the region of interest of the object, so the proposals generated for it are also rectangles. For example, Tang et al. [68, 80] built the WSOD model upon rectangular proposals and use RoI pooling layer [82] and Spatial Pyramid Pooling (SPP) [83] layer to aggregate features. The latter shares certain resemblance with the RoI align technique in Faster-RCNN [4]. However, unlike [4, 82] or [83], the WSOD models [68, 80] do not require labeled RoIs in the training data. Additionally, [68] has found that the treatment of proposals as MIL bags has shown superior performance compared to other models that make use of spatial regulariser [81] to achieve proposal ranking and spatial consistency, i.e. spatially adjacent proposals will have a consistent score.

In this dissertation, building on these existing achievements in WSOD, in order to create a WSOD model that can derive precise object boundaries instead of rectangular bounding boxes, we assimilate these these prior experiences where still applicable,

such as transfer-learning DCNN features [74, 75, 76, 77], using MIL-based proposal scoring [68, 69, 72, 73, 47] and constructing an end-to-end trainable model; but we explore object proposals generated using superpixels [47, 84], for the reasons that will be covered in Sect. 5.1.2. DCNN-based heterogeneous superpixels play an important role in weakly-supervised precise boundary generation. And we adapt and extend this technique in order to create a precise WSOD detector as a step further from rectangular bounding box based approaches.

Sect. 6.2 has previewed an example of superpixel-based multiple-instance learning model, whereby object segmentation is learned from merely image-level class labels, as a positive side-effect of the relational modeling. This has demonstrated that based on high-level features such as their appearance, known components and hierarchical relations, it is possible to create a high-level object detector by learning from not fully supervised or incomplete training data. This capability to recognize objects based on high-level features is a demanding research objective in the state-of-the-art semantic or instance segmentation and localization models. Such a high-level object detector can provide a more generalised abstraction of the imagery objects, which has the benefit of enabling few-shot object detection, leveraging partially labeled data and promoting model explainability. As the training data does not allow for a typical fully supervised model, i.e. not all prediction elements have associated training label to learn from, this approach falls into the category of weakly-supervised learning, specifically Weakly-Supervised Object Detection (WSOD).

Precise bounding box segmentation is a challenging problem. Most models have either taken the fully supervised approach or still involve human interaction and are not fully automated. Approaches that attempt to derive precise boundary from bounding boxes or accomplish related tasks include graph-based interactive image segmentation [78, 85], MIL-based interactive image segmentation [**MILCut** , 14], semantic edge detection and boundary thinning [79] (supervised), FCN-based fusion

network [86], deformable convolution [13] based architectures that feature spatial support visualization.

5.1.1 An Early Attempt with Capsule Networks

Sabour *et al.* [14] put forward an ideology known as *capsules* to improve a model's knowledge of how its observed features relate to one another in a hierarchical manner. Those capsules occupy a specific partition of the model that is clearly associated with a certain class of imagery objects. This is another approach where models are designed with the built-in concept of the imagery objects, which are encoded and stored explicitly within the model. In addition, a routing mechanism is required, that only activates the relevant partition, the capsule, of the model to perform the object recognition. With only a few layers of these capsules, the model can emulate the relationship between low level features (i.e., the superpixels) and higher level features (i.e., the imagery objects). In theory, this can be a better approach because the usage of different transformation matrices $\mathbf{W}_{ij}$ in each capsule with routing suggests that each imagery object gets to have its own independent collection of features in their respective isolated feature spaces. Interested readers can find more information about capsules and routing in [14] and [87].

When accounting for k channels in the superpixel feature map, superpixel features form a matrix $[y_j^k]$. The superpixel features are treated as if they were lower-level capsule features $(\mathbf{u}_j)_k = y_j^k$ and use dynamic routing to interface with the capsule layer that follows,

$$\mathbf{v}_j = \psi\left(\sum_i c_{ij}\mathbf{W}_{ij}\mathbf{u}_i\right), \tag{5.1}$$

where $\mathbf{v}_j$ are capsule outputs per class, which are entity instantiating vectors [14]; and $\psi(\cdot)$ is the squash function [14] that monotonically affects only the magnitude of the k_1-vector, whose exact form (see squash function [14]) is irrelevant here.

Figure 5.2: Object vector space. Object detection encoders often generate vector representation of the objects in a feature space that is well organized with respect to their semantics so that there is a viable decision boundary. Figure from DeltaEncoder [88]

Sect. 6.2 has provided a description of our experiments that has demonstrated the preliminary success with this approach. However, we also notice a few shortcomings in the usage context of superpixel feature maps, such as the ability to converge drastically deteriorates as more layers are added and the routing response is not significantly strong to a fully functioning level. These issues will require further investigation for this model to scale up to more object classes.

5.1.1.1 Representation of Parts and Objects

Consider the representation of an imagery objects, which involves an *object feature space*, U, a finite set of possible objects $\boldsymbol{u}_i \in U$, that are encoded as vectors. The vector $\boldsymbol{u}_i$ represent the i$^{\text{th}}$ object in U, where $1 \leq i \leq |U|$. In addition, a larger object may consist of smaller objects, i.e. there is a hierarchical relation.

The object classes defined in an image classification task can be encoded in this fashion using a hidden layer vector obtained from a DCNN.

The heterogeneous superpixel can also provide vector-encoded objects. Recall

Eq. (5.7),

$$x_c^k(s_j) = \sum_q \frac{|s_j \cap t_q|}{|s_j|}(o^k \circ \zeta_{mn}^{-1})(q) \ ,$$

which describes the superpixel features, can also be viewed as an instance of this object representation,

$$\boldsymbol{y}_j \equiv \boldsymbol{x}_c(s_j) = \sum_q \frac{|s_j \cap t_q|}{|s_j|}(\boldsymbol{o} \circ \zeta_{mn}^{-1})(q) \ , \tag{5.2}$$

given a set, $Y \subseteq \mathbb{R}^{K_0}$, of superpixel features $\boldsymbol{y}_j \in \mathbb{R}^{K_0}$ with K_0 feature channels. Therefore, the superpixel image function $\boldsymbol{x}_c$, along with the superpixel segmentation (c, S), also provides a collection of vector-encoded objects $\boldsymbol{y}_j \in Y \subseteq \mathbb{R}^{K_0}$ to be studied. As superpixel induced objects, they also have the property of being localized through the superpixel segmentation, i.e. they can be spatially pinned to a 2-D coordinate associated with the superpixel they are derived from, which is very meaningful for visualization. Finally, it can useful to introduce intermediate or latent objects in a hierarchical relational model as well.

As the object feature space, U, is a vector set that contains prototypes of objects, the probability density of occurance w.r.t an object is a function $\mathcal{U} \to [0, 1]$, that sums up to one over $\mathcal{U}$, where the vector space $\mathcal{U}$ is a superset of U that will contain vectors that are not exactly one of the object prototypes. Typically, this vector space can be $\mathbb{R}^K$ which is a real vector space that uses K feature channels to encode objects. As a special arrangement, the direction of the vectors can be used to distinguish between objects, and the magnitude of the vectors can be regarded as "instantiation strength". This approach is popularized by Capsule Networks [14, 87], known as instantiation vectors, whose probability of occurance is associated with the magnitude of the vector $\|\boldsymbol{u}_i\|$ directly, through a monotonically increasing function, as shown in Eq. (5.3), the "squash" function [14],

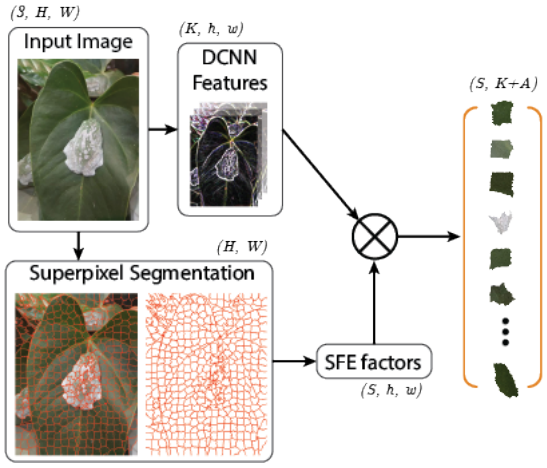


Figure 5.3: Superpixel Segmentation and Superpixel Feature Extraction (SFE). The input image has resolution $(W \times H)$ with 3 color channels. The DCNN features typically will have a smaller resolution $(w \times h)$ but more feature channels (K). A (s, h, w)-tensor can be generated, as the SFE factors, to match the resolution and transform these features in a differentiable manner, where s represents the number of superpixels in the segmentation. The SFE will also generate extra A features that are statistical attributes related the the superpixel and the DCNN features being homogenized.

$$\boldsymbol{f}(\boldsymbol{u}_i) = \frac{\|\boldsymbol{u}_i\|^2}{1 + \|\boldsymbol{u}_i\|^2} \cdot \frac{\boldsymbol{u}_i}{\|\boldsymbol{u}_i\|} \ . \tag{5.3}$$

5.1.2 Deep Superpixel Neural Networks (DSpNN)

Superpixel is regarded as an interesting way of creating a tessellation of the image, i.e. a disjoint partitioning, and this focuses on grouping perceptually homogeneous regions of pixels together to be represented more concisely [40]. Typically, the intensity values of these grouped pixels will be aggregated to achieve dimensionality reduction. We will also refer to this aggregation as *homogenization*, specifically, to distinguish this from feature aggregation or fusion that will occur in later stages in a deep superpixel neural network (DSpNN), by which, broadly, we are referring to deep neural networks that take advantage of the superpixel image representation. A coarse classification of them will be discussed in this section to further clarify our motivation.

Prior research in the field of deep learning has primarily leveraged the superpixel

representation as a way of reducing the computational intensity. For example, in some of the CRF-based, very computationally complex models [11, 47, 70], this has proven critical to the success and practicality of the model. Applying homogenization to DCNN features is also known as superpixel pooling, after which the data representation will be changed profoundly. Fig. 5.3 has illustrated the essential elements in this data transformation. This demands creative ways [84] to further exploit such data if used as alternative high-level image analysis, as this dissertation will discuss and highlight, as opposed to just dimensionality reduction. Some well-known algorithms to generate a superpixel segmentation have been detailed in Chapter 2.

There are three most prominent motivations to incorporate the superpixel representation into a deep neural network: speed, precision and spatial analytical capability. The high computational efficiency of the superpixel representation is primarily derived from homogenization, where redundant pixels are eliminated, leaving the subsequent inference algorithm far fewer data to process. For example, in [11], a single-image monocular depth estimation model was created based on a superpixel oversegmentation and a CRF depth inference model. Despite the fact that an analytical solution has been found, the computational complexity would still have been too high to be practical to carry out on a pixel level. Preserving natural object boundaries facilitates the superpixel's ability to dissect the interesting pixels from irrelevant ones with precision, in contrast to rectangular ROIs. In [47] ROI bounding box data is converted into superpixels. The joint consideration of spatial and feature proximity provides the superpixel representation with a unique vantage point to spatially correlate both low-level and high-level image features and perform analysis in a precise and adaptive fashion.

These three advantages of superpixel motivates our interest in superpixel proposal generation. In the realm of precisely capturing the irregular object boundary, we can at least compare with two other popular techniques such as adaptive segmentation

[89, 90], and *attention*. In more general terms, there exist model-free methods and model-based (or learnable) methods. The flexibility of the former are often limited by the information available at its own processing stage and cannot adapt to the downstream task, while the latter can be unstable and prone to generate imprecise object boundaries. Therefore, there are limitations to generating the superpixel segmentation either before or after the DCNN features are extracted, respectively. Heterogeneous superpixel is a technique devised to address this issue by joining these features, and it is better poised to target WSOD with the goal of precise object boundary. Our approach for object proposal generation utilizes graph-theoretical clustering, whose relevant background will be further discussed in Sect. 5.1.3.

We propose that the criteria of an admissible superpixel segmentation should be the following, for the purpose of generating better object proposals in DSpNN.

1. Each heterogeneous superpixel should not overlap more than one object. They can be heterogeneous with respect to low-level features, but they are conservative with respect to object boundaries, whose accuracy relies on high-level features.

2. A superpixel segmentation should optimize edge recall. Although edges are not sufficient to derive object boundaries, they are necessary to separate objects and provide accurate low-level features to be related later.

3. A superpixel segmentation should contain as few superpixels as possible. Ideally, one heterogeneous superpixel should be sufficient as representing an object proposal.

4. The segmentation algorithm should not significantly affect the inference latency of the neural network.

5.1.3 Graph Neural Networks (GNN)

Graph Neural Networks (GNN), which are currently most popular in prediction of social network or molecular properties, extend the regular task domain of neural works to arbitrary topologies and irregular structures, and they are used for processing graphs, as represented by $G = (V, E)$, where V is the set of vertices, and E is the set of edges. In neural network architectures, a graph is often characterized by the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|V|^2}$ and the feature matrix $\boldsymbol{X} \in \mathbb{R}^{|V| \times K}$. There exist parallel concepts in GNN that follow CNN closely. This leads to Graph Convolutional Networks (GCN) [91], where the equivalent of convolution [92, 93] and pooling [89, 94] operators are developed specifically. In our case, GNN is only one of the many valid approaches [89, 95, 96] towards processing the binary relation between superpixels in order to generate object proposals for the WSOD model. In this dissertation, we have to focus on joining low-level and high-level features to create a novel precise WSOD model using heterogeneous superpixel, and leave the ablation study of the effects of specific graph algorithms to the future research. However, we will explore the min-cut layer [89] from GCN in order to generate object proposals based on heterogeneous superpixels for the purpose of the WSOD model construction, as it will be detailed in Sect. 5.2.2.

5.2 METHODOLOGY

5.2.1 Proposed Network Architecture

Our proposed network architecture, Deep Heterogeneous Superpixel Network (DHSN), is shown in Fig. 5.4, which has two parallel streams. The first stream is a typical DCNN classification network which contains a transferable DCNN backbone network as feature extractor, followed by a global 2-D pooling layer and several task specific dense layers in the MLP component. The second stream has the Superpixel Feature

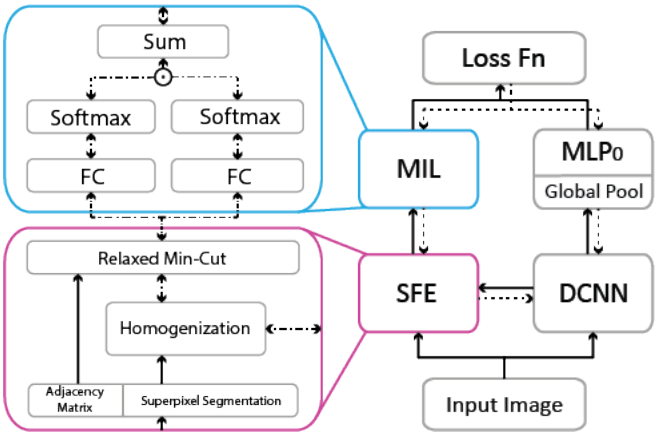


Figure 5.4: Deep Heterogeneous Superpixel Network (DHSN). On the right side is the overall architecture of DHSN, which is based on a transfer-learned DCNN and heterogeneous superpixel feature extraction (SFE) and multiple-instance learning (MIL). On the left side are more detailed illustrations of SFE and MIL, which facilitate superpixel-level image analysis for WSOD and model explainability purposes. The novel adaptive SFE design in our methodology not only bridges the transfer-learned DCNN and MIL to achieve WSOD, but also joins high-level and low-level features to generate precise object proposals using the heterogeneous superpixel technique.

Extraction (SFE) module, which reorganizes the DCNN feature according to superpixels, i.e. transforming the (h, w) axes into the s axis. At the same time, it can optionally include one shared layer of neurons to reduce and optimize the superpixel feature space, transforming the K axis into the M axis. A represents the number of superpixel attributes that will be generated from the SFE. They can be used for data fusion purposes in the future research. The SFE will be further discussed in Sect. 5.2.2. Following the SFE, superpixel-based object proposals are then generated. These object proposals will be processed using Multiple Instance Learning (MIL) for the purpose of uncovering the instance level object classification that is otherwise obscured by typical DCNN top layers, including global 2-D pooling and dense layers. The MIL network functions by generating a proposal classification score through a learnable instance classifier, then aggregates scores to reconcile with the image classification based loss function. By doing so, this MIL network architecture can correlate these irregularly-shaped object proposals with image classes and further break down

the causal relation between the image and the label given in the training data.

This entire proposed architecture is end-to-end differentiable, and the matrix formulation of SFE makes the backward propagating gradient path easier to implement. Table **??** has provided mathematical notations that will be use in the following sections.

5.2.2 Heterogeneous Superpixel

Contrary to the popular usage of superpixel, herein much larger superpixel sizes are explored as a way of providing object proposals. The aim of heterogeneous superpixel is to achieve a precise but weakly-supervised object segmentation. As discussed, to accomplish this, both low-level and high-level features need to be jointly considered. Heterogeneous superpixels are created by reorganizing and localizing the high-level DCNN features according to the oversegmentation generated from the low-level features, and plays a key role in achieving this goal; whereas the conventional homogeneous superpixel primarily groups perceptually homogeneous regions and reduces the dimensionality of the data. In this subsection, we discuss in detail the implications and utility of feature extraction using heterogeneous superpixels.

5.2.2.1 Initial Segmentation

Superpixels are generated using the SLIC superpixel algorithm. As discussed in Sect. 5.1.2 on related works, it is chosen due to its good balance between computational efficiency and accuracy. Additionally, its superpixel generation is more conservative, i.e. it is less prone to include multiple objects in the same superpixel, compared to Felzenszwalb's Algorithm, Watershed, and Quick Shift. In the usage of preparing superpixels to generate object proposals, this property is of paramount importance. This algorithm first converts the image's color into CIE L*a*b color space $\mathbb{C}$, then clusters the data in the 5-D feature space $\mathbb{F} = \mathbb{C} \times \mathbb{R}^2$, which is generated

by the concatenation of the CIE L*a*b color space $\mathbb{C}$ and the 2-D pixel coordinate (spatial) system $\mathbb{R}^2$. The color proximity and spatial proximity are weighted using a compactness factor m. The distance measure $d : \mathbb{F}^2 \to \mathbb{R}^+$ used in the SLIC algorithm [44] is shown in Eq. (5.4), where λ_c and λ_s and are color and spatial proximity normalizers;

$$d\begin{pmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 \end{pmatrix} = \sqrt{\lambda_c \|\boldsymbol{c}_1 - \boldsymbol{c}_2\|^2 + \lambda_s m^2 \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}, \\ \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{x}_1 \end{pmatrix} \in \mathbb{F}, \quad \begin{pmatrix} \boldsymbol{c}_2 \\ \boldsymbol{x}_2 \end{pmatrix} \in \mathbb{F} \tag{5.4}$$

The two most important parameters of the algorithm include the expected superpixel size, which can control the granularity of the superpixel segmentation, and the compactness factor and will control the superpixel shape irregularity that is achievable. Heterogeneous Superpixels are initially generated from such a segmentation algorithm with the expected superpixel size set to a value comparable to the objects or components of the objects, from which object proposals can be derived later. Our architecture has allowed a relatively high superpixel local entropy as an admissible segmentation so as not to overfit low-level features, but to facilitate the opportunity to jointly optimize for weakly-supervised object detection.

5.2.2.2 SFE Factors

The SFE factors play an important role in spatially reorganizing DCNN features and the reconciliation of feature map resolutions. In essence, this is a soft membership assignment of the pixels to the superpixels. As shown in Fig. 5.3, the SFE factors are organized in the form of a multi-dimensional array with shape (s, h, w), shere s is the number of superpixels inferred from the expected superpixel size, and h and w

match the height and width of the DCNN feature map respectively. The membership is typically assigned to the area occupancy rate, as shown in Eq. (5.5), generally,

$$\boldsymbol{\tau}_{\text{SFE}}(S,T)^{ij} = \frac{|S_j \cap T_i|}{|S_j|},\ j \in [1,|S|],\ i \in [1,|T|]\ . \tag{5.5}$$

where S and T are two sets of pixel sets representing two different segmentations or partitioning of the entire pixel set of the image, S_j is some pixel set in the partitioning S, T_i is some pixel set in the partitioning T, and the superpixel feature assignment direction is from T to S. In the specific case of the initial heterogeneous superpixel segmentation, the assignment matrix $\boldsymbol{C}_0$ is shown in Eq. (5.6),

$$\boldsymbol{C}_0 = \boldsymbol{\tau}_{\text{SFE}}(V_1, \varphi_{\text{UPSCALE}}(V_0))\ , \tag{5.6}$$

where $\varphi_{\text{UPSCALE}}(V_0)$ represent the hypothetically up-scaled convolutional map tiles, which would have aligned the feature map resolution to the segmentation resolution, but computationally this will be implemented by an inverse coordinate transformation during the homogenization, detailed in Sect. 5.2.2. This matrix $\boldsymbol{C}_0$ is a sparse quantity where non-zero elements occur at the probability of $1/s$. While the forward pass of SFE does not have to rely on this quantity, but it will make the backward pass much easier to implement, now that the SFE can be formulated as a matrix multiplication. This matrix formulation of the SFE factor is also consistent with the assignment matrix used in the famous max-flow min-cut problem [50], which will appear again in Sect. 5.2.2.

5.2.2.3 Homogenization

As discussed, a superpixel segmentation algorithm provides a partitioning of a 2-D image with irregular region shapes. Thus a superpixel-segmented image can be represented as a series of superpixel intensity values. The resulting superpixel-segmented

image is typically stored in a 1-D data structure with multiple channels. A 2-D spatial representation can be obtained by filling superpixel intensity values back into the segmentation. This reorganization retains the association between vectors from the feature space and their locality within the image.

In order to match the higher resolution superpixel segmentation with the lower resolution convolutional map of extracted DCNN features, we associate each pixel of the convolutional feature map to a tile of pixels in the superpixel segmentation. This is a conceptual up-scaling of the convolutional feature map, while practically computation of the pixels can be directly addressed without up-scaling or up-sampling the convolutional feature map. This association will induce the up-scaled convolutional map tiles $\varphi_{\text{UPSCALE}}(V_0)$ in Eq. (5.6).

Then the SFE factors can be applied to extract superpixel features, as shown in Eq. (5.7) in matrix form, or equivalently Eq. (5.8) in the element-wise expression,

$$\boldsymbol{X}_1 = \boldsymbol{C}_0^T \cdot \boldsymbol{X}_0 \ , \tag{5.7}$$

$$\boldsymbol{X}_1^{jk} = \sum_q \frac{|(V_1)_j \cap (\varphi_{\text{UPSCALE}}(V_0))_q|}{|(V_1)_j|} \cdot (\boldsymbol{o}^k \circ \zeta_{mn}^{-1})(q) \ , \tag{5.8}$$

where ζ_{mn}^{-1} is an inverse coordinate transformation, and $(\boldsymbol{o}^k \circ \zeta_{mn}^{-1})$ serves as the hypothetically up-scaled convolutional map at the k-th channel in $m \times n$ resolution. For the purpose of WSOD, we can also reduce the dimensionality of the features and parameterize this operation using a shared fully connected layer $\boldsymbol{\varphi}_{\text{FC0}} : \mathbb{R}^{|V_1| \times K} \to \mathbb{R}^{|V_1| \times M}$,

$$\begin{aligned} \boldsymbol{\nu}_0(\boldsymbol{X}) &= \boldsymbol{X}\boldsymbol{W}_0 + \boldsymbol{1}_{|V_1|}\boldsymbol{b}_0^T, \\ \boldsymbol{X}_1 &= \boldsymbol{\varphi}_{\text{FC0}}(\boldsymbol{C}_0^T\boldsymbol{X}_0; \Theta_0 = \{\boldsymbol{W}_0, \boldsymbol{b}_0\}) \\ &= \phi_{\text{FC0}}(\boldsymbol{\nu}_0(\boldsymbol{C}_0^T\boldsymbol{X}_0)) \ , \end{aligned} \tag{5.9}$$

where $\boldsymbol{\nu}_0 : \mathbb{R}^{|V_1| \times K} \to \mathbb{R}^{|V_1| \times M}$ is the induced local field of the shared fully connected

layer, Θ_0 is holding the set of parameters of the fully connected layer, including weights $\boldsymbol{W}_0 \in \mathbb{R}^{K \times M}$ and biases $\boldsymbol{b}_0 \in \mathbb{R}^M$, and ϕ_{FC0} is the activation function ReLU. The gradient backpropagation of SFE is shown in Eq. (5.10),

$$\begin{aligned}
\frac{\partial l_{\text{TOTAL}}}{\partial \boldsymbol{X}_0} &= \boldsymbol{C}_0 \cdot \left(\frac{\partial \epsilon}{\partial \boldsymbol{X}_1} \odot \frac{\partial \phi_{\text{FC0}}}{\partial \boldsymbol{\nu}_0} \right) \cdot W^T + \epsilon_{\text{MLP0}}, \\
\frac{\partial l_{\text{TOTAL}}}{\partial \boldsymbol{W}_0} &= (\boldsymbol{C}_0^T \boldsymbol{X}_0)^T \left(\frac{\partial \epsilon}{\partial \boldsymbol{X}_1} \odot \frac{\partial \phi_{\text{FC0}}}{\partial \boldsymbol{\nu}_0} \right), \\
\frac{\partial l_{\text{TOTAL}}}{\partial \boldsymbol{b}_0} &= \left(\frac{\partial \epsilon}{\partial \boldsymbol{X}_1} \odot \frac{\partial \phi_{\text{FC0}}}{\partial \boldsymbol{\nu}_0} \right)^T \mathbf{1}_{|V_1|},
\end{aligned} \tag{5.10}$$

where ϵ_{MLP0} is the error contribution through the first stream of the architecture, which are the top-layers of the original DCNN architecture, including the global 2-D pooling layer or flatten layer and the dense layers, the notation $\odot$ represents the element-wise product, and all the partial derivatives preserve the layout of the matrix or the vector that they are taken with respect to.

5.2.2.4 Superpixel Proposals

The heterogeneous superpixel based object proposals are generated with two key things to consider. One is the initial superpixel size, which has already relaxed the conventional superpixel homogeneity assumption, the other is a graph-theoretical refinement technique using min-cut [89], which extends the heterogeneous superpixel to match potential object boundaries.

The model has allowed a relatively high superpixel local entropy in an admissible segmentation. The mean local entropy is measured by creating a 256-bin hue histogram with respect to each superpixel region. From Fig. 4.1, it is evident that, in the context of superpixel segmentation, the mean local entropy exhibits a strong anticorrelation with respect to the number of superpixels. Superpixels are viewed as disjoint pixel sets S_i and the entire image as the pixel set V and $H(\cdot)$ as the entropy operator over some pixel set, as shown in Eq. (5.11),

$$0 \leq \sum_{S \in V_1} \frac{|S|}{|V_0|} H(S) \approx \frac{H(V_0)}{|V_1|} \leq H(V_0) \ . \tag{5.11}$$

Empirically it can be found that Eq. (5.11) approximately describes the graph shown in Fig. 4.1 across a broad range of number of superpixels evaluated. The multiplication factor s^{-1} has shown the effect of reducing the entropy of superpixel feature organization; and it is strongly dependent on the number of superpixels used in image resolution.

Finally, the relaxed min-cut is applied to further evolve the heterogeneous superpixel into object proposals. This technique is primarily used in GNNs in contemporary deep learning research [89] as a graphical pooling layer. Specifically, an assignment matrix $\boldsymbol{C_1}$ will be learned using a Multi-Layer Perceptron (MLP) $\boldsymbol{\varphi}_{\mathrm{MLP1}} : \mathbb{R}^{|V_1| \times K} \rightarrow \mathbb{R}^{|V_1| \times |V_2|}$, holding a set of parameters Θ_1, as

$$\boldsymbol{C_1} = \boldsymbol{\varphi}_{\mathrm{MLP1}}(\boldsymbol{X}_1; \Theta_1) \ , \tag{5.12}$$

where $\boldsymbol{X}_1$ are the initial heterogeneous superpixel features. Two loss function terms can be added to approximate the relaxed min-cut problem, including a *cut loss* l_c and an *orthogonality loss* l_0, as shown in Eq. (5.13).

$$l_c = -\frac{\mathrm{tr}(\boldsymbol{C}_1^T \boldsymbol{A}_1 \boldsymbol{C}_1)}{\mathrm{tr}(\boldsymbol{C}_1^T \boldsymbol{D}_1 \boldsymbol{C}_1)} \ , \ l_o = \left\| \frac{\boldsymbol{C}_1^T \boldsymbol{C}_1}{\|\boldsymbol{C}_1^T \boldsymbol{C}_1\|} - \frac{\boldsymbol{I}_{|V_2|}}{\sqrt{|V_2|}} \right\| \ , \tag{5.13}$$

where, $\mathrm{tr}(\cdot)$ is the matrix trace operator, $\boldsymbol{A}_1$ and $\boldsymbol{D}_1$ are the adjacency matrix and the degree matrix of the graph derived from the initial heterogeneous superpixel segmentation. The orthogonality loss serves as a regularizer to adjust the assignment in the form of cluster uniformity.

To define the adjacency matrix, a distance metric between all pairs of superpixels is required because the relaxed min-cut allows continuous edge weights in order to

reduce computational complexity. As the adjacency matrix will determine the actual topology of the graph, it will have significant impacts on the resulting object proposal. In principle, the weights should be assigned according to high-level features that can represent imagery objects. Therefore, it is reasonable to build the distance metric upon SFE features, which are derived from DCNN features. For this purpose, we use the Euclidean distance between SFE feature vectors. A jointly optimized learnable SFE will also be able to also optimize the distance metric to the extent attainable by warping the SFE feature space.

Finally, the object proposals are generated by simultaneously applying the assignment matrix $\boldsymbol{C}_1$ to the adjacency matrix and the feature matrix of the heterogeneous superpixels,

$$\begin{aligned} \boldsymbol{A}_2 &= \boldsymbol{C}_1^T \boldsymbol{A}_1 \boldsymbol{C}_1 - \boldsymbol{I}_{|V_2|}\mathrm{diag}(\boldsymbol{C}_1^T \boldsymbol{A}_1 \boldsymbol{C}_1), \\ \boldsymbol{X}_2 &= \boldsymbol{C}_1^T \boldsymbol{X}_1 \end{aligned} \tag{5.14}$$

where $\boldsymbol{A}_2$ is the adjacency matrix (without self-loops) of object proposals. This adjacency matrix will be used later, in Sect. 6.4.4, to retrieve the imagery object precisely.

5.2.3 Multiple-instance Learning

By combining convolutional neural networks with multiple instance learning method, Multiple Instance Detection Network (MIDN) has become the most popular method to address the WSOD problem and been adopted as the initial model in many works [97]. With Multiple-Instance Learning (MIL), the model receives "bags" containing "instances", where only the bags are labeled, but the instances, even though being part of learning objective, are not labeled. In a binary example, a bag may be labeled as either positive or negative. A bag may be labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least

one instance in it which is positive. By presenting the model a collection of these bags to learn from, the model is expected to create an internal concept of these instances and identify these instances as well as the bags and label them. Coupled Multiple Instance Detection Network (C-MIDN) [97] is one of the state-of-the-art models to address this problem. Specifically, there is a pair of MIDNs, which work in a complementary manner. The localization information of the MIDNs is further coupled to obtain tighter bounding boxes and localize multiple objects. Finally this model obtains 53.6% and 50.3% mAP on the challenging PASCAL VOC 2007 and 2012 [2] benchmarks respectively, which significantly outperform the prior works.

In the case of the superpixel feature maps, they can be viewed as a perfect example of MIL bags, where the entire image, or the labeled rectangular ROI is the MIL bag, and the superpixels are unlabeled instances. This formulation is a novel approach towards creating a fully automated WSOD model.

The purpose of the MIL in our proposed architecture is to generate proposal scores for each object proposal and eventually determine the precise object boundary. The specific MIL method used is based on a proposal cluster learning [68] approach. Upon receiving object proposals, the network splits into two streams, where one stream will apply softmax across object classes and the other will apply softmax across object proposals. Therefore, this network generates two estimated probability distribution densities, respectively. The former establishes the probability density of a given proposal identified as one of the object classes, whereas the latter establishes the probability density of the presence of a given object class indicated by one of the object proposals. Then the two are combined using an element-wise product to generate the proposal score $\boldsymbol{Y}_1$, which indicates the confidence that the object proposal in question

is the target as described by the image-level class label, as shown in Eq. (5.15),

$$\begin{aligned}
\boldsymbol{N}_{3.1} &= \boldsymbol{\nu}_{\text{FC2.1}}(\boldsymbol{X}_2; \Theta_{2.1}), \\
\boldsymbol{N}_{3.2} &= \boldsymbol{\nu}_{\text{FC2.2}}(\boldsymbol{X}_2; \Theta_{2.2}), \\
\boldsymbol{Y}_1^{ij} &= \frac{\exp \boldsymbol{N}_{3.1}^{ij}}{\sum_k \exp \boldsymbol{N}_{3.1}^{ik}} \cdot \frac{\exp \boldsymbol{N}_{3.2}^{ij}}{\sum_k \exp \boldsymbol{N}_{3.2}^{kj}}, \\
\boldsymbol{y}_2 &= \boldsymbol{Y}_1^T \mathbf{1}_{|V_2|},
\end{aligned} \tag{5.15}$$

where $\boldsymbol{\nu}_{\text{FC2.1}}, \boldsymbol{\nu}_{\text{FC2.2}} : \mathbb{R}^{|V_2| \times M} \rightarrow \mathbb{R}^{|V_2| \times C}$ are the local induced fields of two distinct fully connected layers, resulting in matrices $\boldsymbol{N}_{3.1}$ and $\boldsymbol{N}_{3.2}$ respectively, C is the number of object classes, and all elements of $\boldsymbol{y}_2$ already belong to the range $[0, 1]$, as shown in the inequalities of Eq. (5.16),

$$\begin{aligned}
0 \leq \boldsymbol{y}_2^i &= \sum_j \frac{\exp \boldsymbol{N}_{3.1}^{ij}}{\sum_k \exp \boldsymbol{N}_{3.1}^{ik}} \cdot \frac{\exp \boldsymbol{N}_{3.2}^{ij}}{\sum_k \exp \boldsymbol{N}_{3.2}^{kj}} \\
&\leq \sum_j \frac{\exp \boldsymbol{N}_{3.1}^{ij}}{\sum_k \exp \boldsymbol{N}_{3.1}^{ik}} = \frac{\sum_j \exp \boldsymbol{N}_{3.1}^{ij}}{\sum_k \exp \boldsymbol{N}_{3.1}^{ik}} = 1,
\end{aligned} \tag{5.16}$$

which means the commonly used multi-class cross-entropy loss is appropriate to apply without further ado, except for preventing its values from reaching zero or one exactly. Refer to Table **??** for notations used here.

Compared to [68], our proposed architecture has the advantage of precise object proposals, as opposed to the more popular rectangular bounding boxes. From the information-theoretical point of view, the DCNN features are spatially re-organized by SFE and has a denoising effect. From the signal processing point of view, due to the improved spatial precision, SFE features does not account for the context by itself, instead, this is derived from the convolution operations in the DCNN network. By reorganizing the feature space using SFE, we are afforded the opportunity to move towards precise WSOD, and using a jointly optimized learnable SFE has the potential to improve the feature space for the purpose of WSOD.

5.2.4 Total Loss Function

The proposed architecture is constructed in a way that the two streams are parallel to each other. This design decision is made because considering the optimization of Eq. (5.13) is highly non-convex and prone to a local minimum convergence, a two stream architecture can effectively single this out as one of the terms in the error backpropagation, as opposed to a factor resulted by the invocation of the chain rule. Otherwise, had the WSOD layers been organized in serial, the classification DCNN could drastically sacrifice in accuracy due to the challenge of non-convex local minimum convergence. This is a common issue as observed in the routing of capsule nets [14, 87], objection relation mapping [84], Proposal Cluter Learning (PCL) [68], etc. As a result of the two-stream design, the amount of error backpropagation can also be controlled by a regularizing weight in the loss function. In [68], a weighted loss strategy is adopted across object classes, and it is demonstrably beneficial for model training.

5.3 A REFLECTION ON BACKGROUND NOISE

As discussed in Sect. 6.2 and [47], one of the interesting effects that a superpixel models create is that rectangular bounding boxes can be refined into more precise boundaries, even with only very minimal effort. It is primarily done by computing a spatial join between the superpixel polygonal regions and the rectangular bounding boxes known from the training data. As an approximation, it is also feasible to substitute the superpixel centroids for the superpixel regions. It is an approach successfully applied in [47] as data labeling refinement in the data pre-processing. Nevertheless, this image processing method has provided a good example of carrying out more effective image analysis by dissecting out the background noise. To clarify, the concept of *background noise* here only refers to the part of the image that belongs

to the background and also gets mixed into the convolutional filter receptive field, which is inevitable due to the square geometry of the convolutional filters. That is to acknowledge that the background does provide information and context generally, but only some of the background area, that gets mixed up with foreground from the convolutional filter receptive field, are considered background noise here, because their contribution is suspected to be redundant and counterproductive.

In contemporary computer vision research, especially those that are deep learning based, people have come up with many ways to increase noise tolerance of the model and reduce over-fitting by presenting more artificial noise or distortion. However, provided some noise are feasible to be removed, why would it not take the precedence over having to work with all that noise, which by definition, should contain no useful information to the model?

Therefore, this result has also prompted a potentially more effective alternative paradigm, where we focus more on the useful information and disregard the useless, and among the useful information, there is also a higher level of organization, compared to the most typical square-shaped DCNN receptive fields. This paradigm is not limited to image pre-processing. When built into the model with tunable parameters during training, this can allow adaptive noise dissection and feature space organization, which presents as a dynamic element that affects the DCNN receptive field without altering the DCNN architecture, and potentially enables more time and power efficient models as they intelligently narrow down the focus.

Chapter 6

APPLICATIONS, EXPERIMENTS AND RESULTS

The heterogeneous superpixel feature space, described in Chapter 4, provides a foundation for high-level object detection based on superpixel-level image analysis methods. This chapter provides several successful examples, in which superpixel-enabled deep neural network architecture provides additional insights on the image datasets from various problem domains, while keeping up with the state-of-the-art model predictive performance.

6.1 MONOCULAR RELATIVE DEPTH ESTIMATION

6.1.1 Problem Description

We propose a method to perform monocular relative depth perception using a passive visual sensor [98]. Specifically, the proposed method makes depth estimation with a superpixel based regression model based on features extracted by a deep convolutional neural network. We have established and conducted an analysis of the key components required to create a high-efficiency pipeline to solve the depth estimation problem with superpixel-level regression and deep learning. The key contributions of our method compared to prior works are as follows. First, we have drastically simplified the depth estimation model while attaining near state-of-the-art prediction performance, through two important optimizations: the idea of the depth estimation model is completely based on superpixels that very effectively reduces the dimension-

ality; additionally, we exploited the scale-invariant mean squared error loss function which incorporates a pairwise term with linear time complexity. Additionally, we have developed optimizations of the superpixel feature extraction, that leverage GPU computing to achieve real-time performance (over 50fps during training) Furthermore, this model does not perform up-sampling, which avoids many issues and difficulties that one would otherwise have to deal with. To perpetuate future research in this area we have created a synchronized multiple-view depth estimation training dataset that is available to the public.

6.1.1.1 Stereo Vision and Depth Dataset

Related work such as [56] and [99] primarily use the NYU indoor dataset [100]; while the Make3D [101] and KITTI [102] outdoor datasets are generally focused on the robotic vision field as a benchmark. All three utilize laser sensor for acquiring a depth map. However, strong sunlight can cause infrared interference and make depth information more noisy. Therefore, in this work we developed a dataset captured by stereo camera, thereby facilitating research that explores passive depth sensing.

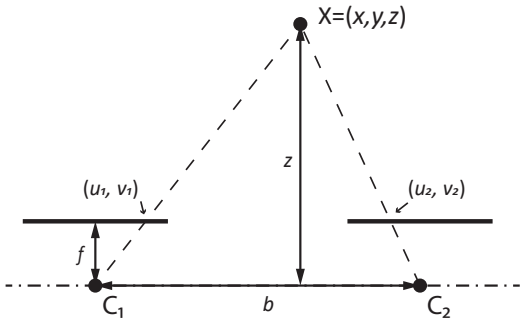


Figure 6.1: Stereo cameras rectified configuration. Disparity $d = u_2 - u_1$

Our training images are collected using a stereo camera, which generates depth maps via binocular vision techniques. The camera has two "eyes" separated by 12 cm which captures a high-resolution video of the scene and estimates depth by comparing the displacement of pixels between the *left* and *right* images. Then the distance to the

physical point corresponding to each pixel in the image is triangulated. The depth map aligns with the left image and encodes depth ranging from 0.7 m to 20 m in a 8-bit representation [103]. This dataset is made publicly available at https://ieee-dataport.org/open-access/indoor-stereo-vision-and-depth.

The stereo vision configuration, as shown in Fig. 6.1, is referred to as a rectified configuration, where the *left* and *right* image planes coincide and optical axes of the two cameras are oriented in parallel. Therefore, depth of each pixel can be computed using Eq. (6.1) [55].

$$z = \frac{bf}{d} \tag{6.1}$$

In Eq. (6.1), z is the physical depth, b is distance of between two cameras, f represents focal length, and $d = u_2 - u_1$ is disparity measured on two different images for the same physical point.

Uncertainty of the triangulation can be found by taking derivative of depth w.r.t. disparity, as shown in Eq. (6.2), which means that the depth resolution would decrease quadratically over the z-distance.

$$\frac{\partial z}{\partial d} = -\frac{bf}{d^2} \tag{6.2}$$

According to [103], it may have a loss in accuracy of 1% of the distance in the near range, and up to 9% in the far range. The uncertainty of disparity can potentially be affected by outlier measurements on homogeneous and texture-free surfaces.

We captured training images from six different indoor locations. We trained and validated the model with images captured from left camera and the computed depth map; the right camera images were not included this study, but they are available in the public dataset. All images are 1280×720 pixels in resolution. Figure 6.3 shows sample images from dataset; and Table 6.1 provides additional details.

Table 6.1: Dataset

Part	Description	Samples	
		Used	Total
Part 1	Library	137	137
Part 2	Book Shelves	90	107
Part 3	Conference Rm.	383	403
Part 4	Cafe	338	350
Part 5	Study Area	288	293
Part 6	Hallway	504	513

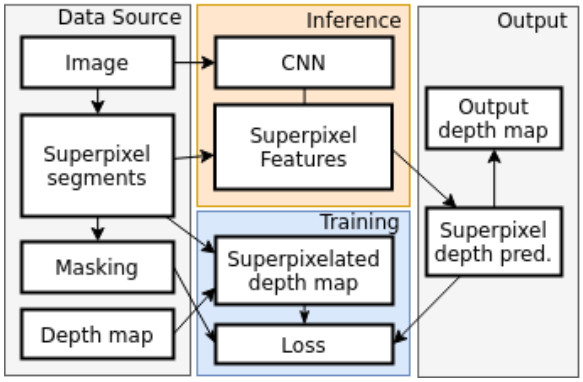


Figure 6.2: Monocular depth estimation model architecture: the raw data required are an RGB visual image and the corresponding depth map. The masking is necessary to condition the loss function for each sample as each one may contain a different number of superpixels.

6.1.2 Model Architecture

The proposed network architecture is shown in Fig. 6.2.This is a monocular view depth estimation model that requires an RGB visual image and the corresponding depth map for training. The superpixel segmentation is derived from the visual image and can be either processed on-the-fly or pre-processed for repeated iterations of training. The masking stage is necessary to condition the loss function for each training sample, as each one may contain a different number of final superpixels, as discussed in detail in the Training Loss Function Sect.

This model contains trainable weights in the convolutional feature extraction that

are implemented as the convolutional blocks from VGG-16; and the depth regression component that is based on a shared MLP. The convolutional feature maps will be reorganized into superpixel-based features, which is a non-parametric component of the system. The model contains 44M trainable parameters in total, and these parameters are trained on the dataset acquired as described in Sect. 6.1.1. The layers leveraged from VGG-16 network have all layers initialized with weights from ImageNet [20], which originally allowed the VGG-16 network to perform image recognition tasks. The Adam optimizer has been utilized here in place of SGD optimizer and has shown a superior training efficiency.

6.1.2.1 Training Loss Function

The training loss function is based on the scale invariant mean squared error [104] [105], which has incorporated both the direct difference in log-scale and a pairwise term to improve continuity of the depth map. The computational complexity of the loss function is linear time with respect to the number of superpixels. During training, each sample will have a different number of superpixels, therefore the loss function should only account for the data that represent valid superpixels from the fixed size prediction vector. The same should apply to any error metric as well. Additionally, there is a linear-time pairwise regularization implemented in Eq. (6.4), which reduces the loss if the error occurs in the same direction and increases the loss if the error is in an opposite direction [104].

$$d_s = y_s^{\text{pred}} - \log z_s^{\text{true}} = \log z_s^{\text{pred}} - \log z_s^{\text{true}} \tag{6.3}$$

$$\mathcal{L} = \frac{1}{S_k} \sum_{s=1}^{S_k} d_s^2 - \frac{\lambda}{S_k^2} \left(\sum_{s=1}^{S_k} d_s \right)^2 \tag{6.4}$$

6.1.3 Results

The key motivation of this research is to develop an efficient methodology to advance monocular depth estimation a step closer towards real-time applications. In this section, the experiments and results from our method will be discussed in detail. As mentioned, we have the depth estimation model predict $y_s^{\text{pred}} = \log z_s^{\text{pred}}$, therefore $z_s^{\text{pred}} = \exp y_s^{\text{pred}} > 0$. Suppose S_k is number of superpixels generated for sample k and α is the relative accuracy threshold, some metrics commonly used in prior works [56] [99] [104] for evaluating depth estimation are:

- Average relative error (rel)

$$\frac{1}{S_k} \sum_{s=1}^{S_k} \frac{\left| z_s^{\text{true}} - z_s^{\text{pred}} \right|}{z_s^{\text{true}}} \tag{6.5}$$

- Root mean squared error (rms)

$$\sqrt{\frac{1}{S_k} \sum_{s=1}^{S_k} \left(z_s^{\text{true}} - z_s^{\text{pred}} \right)^2} \tag{6.6}$$

- Scale-invariant root mean squared error (si-rms)

$$\sqrt{\frac{1}{S_k} \sum_{s=1}^{S_k} d_s^2 - \frac{1}{S_k^2} \left(\sum_{s=1}^{S_k} d_s \right)^2} \tag{6.7}$$

- Average $\log_{10}$ error ($\log_{10}$)

$$\frac{1}{S_k} \sum_{s=1}^{S_k} \left| \log_{10} z_s^{\text{true}} - \log_{10} z^{\text{pred}}{}_s \right| = \frac{1}{S_k \log 10} \sum_{s=1}^{S_k} |d_s| \tag{6.8}$$

- Relative depth threshold accuracy (acc)

$$\frac{1}{S_k}\sum_{s=1}^{S_k}\left[\max\left(\frac{z_s^{\text{true}}}{z_s^{\text{pred}}}, \frac{z_s^{\text{pred}}}{z_s^{\text{true}}}\right) < \alpha\right] \tag{6.9}$$

Due to the superpixelated depth estimation formulation, each sample may consist of different number of superpixels, consequently, we must pay special attention that the loss function and metrics are dependent on samples, namely the exact amount of superpixels S_k each has. Among these metrics, the *si-rms*, $\log_{10}$, and accuracy can be considered scale-invariant because they either compare depth values using a ratio or compare in a logarithmic scale. On the other hand, the *rel* and *rms* metrics are directly measuring the differences between predicted depth maps and superpixelated ground truth.

6.1.3.1 Evaluation

We have evaluated our model performance on the dataset detailed in Sect. 6.1.1. A difference to note is that our metrics are completely based on superpixels and the accuracy in Table 6.2 has been compensated by an expectation value of the error introduced by the superpixelation process relative to the pixel-based metrics. However, other metrics are not adjusted because we would likely be introducing some complex approximation calculation without necessarily improving clarity. For reference, as the ground-truth depth map is processed into the target superpixel depth map, around 8% of relative accuracy error incurred in the process.

As Table 6.2 shows, our method has achieved a comparable performance to prior research, while we have significantly simplified a depth estimation model to optimize inference and training time. This is largely due to the superpixel feature extraction and an efficient choice of loss function. The superpixel feature extraction has allowed us to compare images in a computationally effective way when constructing the loss

Table 6.2: Model Evaluation

Method	Error metrics			Accuracy
	rel	log10	rms	$\alpha = 25\%$
Liu *et al.* [56]	0.213	0.087	0.759	65%
Laina *et al.* [99]	0.127	0.055	0.573	81%
Our method	0.393	0.078	0.342	83%

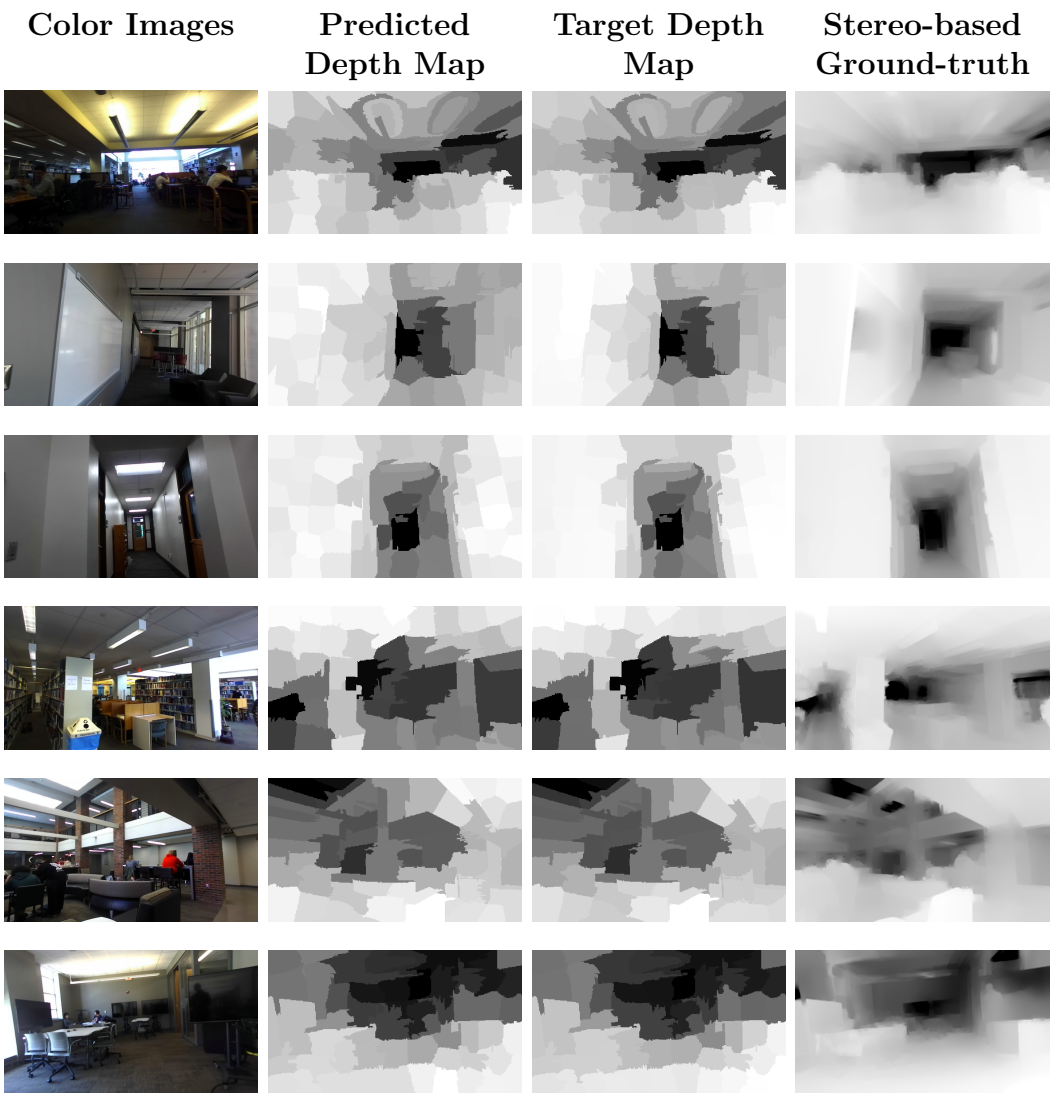


Figure 6.3: Examples of depth map prediction from all 6 scenes. They are, from top to bottom, library, study area, hallway, book shelves, cafe and a conference Room. From left to right, they are input images, predicted depth maps, superpixelated ground truth and multiple-view based depth map ground truth.

function by comparing the relatively small amount of superpixels instead of requiring an up-sample of the convolutional feature maps back to the input image resolution. The convolutional feature maps used in the model is as small as $(12, 6)$, which are reorganized into a superpixel representation in order to compare against the superpixelated ground truth in the loss function. The scale-invariant loss function has incorporated both the direct difference in log-scale and a pairwise term to improve continuity of the depth map.

6.1.3.2 Per-scene Analysis

We have also examined our per-scene metrics as shown in Table 6.3 (all metrics are superpixel based). From this analysis we can see that the model accuracy is dependent on the complexity of the scene, where a key trade-off is between more precise pairwise terms in the loss function and the inference speed. Qualitatively, the complex scenes include the cafe and the book shelves, and the spacious scenes include the cafe (again), the library and the conference room. Logically, these can be the more challenging places for the model to make depth predictions as their depth varies discontinuously in a wide range, but according to the per-scene metrics the model has handled most of the scenes equally well thanks to the scale-invariant depth estimation objective function. Here we see the cafe and the study area as the lower performing subsets. The hallway has the least amount of furniture and the structure of the scene is simpler; and the model has had the best performance in this kind of relatively close range depth estimation situation.

Apart from the complexity of the scene, another known issue that affects the depth estimation model is the presence of light sources. For example, the fluorescent light in the scene that appeared in the first row of Fig. 6.3 has created a color gradient in the input color image as well as the ground-truth depth map. In this case, not only has the model been trained with inaccurate ground-truth because the ground truth

Table 6.3: Subset Analysis

Scene	Error metrics				Accuracy	
	rel	rms	si-rms	log10	25%	10%
Library	0.393	0.326	0.245	0.052	92%	81%
Book Shelves	0.394	0.320	0.246	0.056	**93%**	81%
Conference Rm.	0.368	0.314	0.251	0.063	92%	80%
Cafe	0.445	0.373	0.264	0.063	90%	76%
Study Area	0.412	0.356	0.272	0.071	89%	77%
Hallway	0.333	0.362	0.238	0.061	**93%**	**83%**

was generated through stereo-based disparity map, but also it is suspected that the model can be influenced by such condition even with accurate ground-truth because the depth estimation is formulated as a regression from visual features that ultimately came from the color images. Fig. 6.3 shows examples our depth estimation. As the loss function of the model is constructed based on superpixels, the ground truth depth maps have to be superpixelated as well.

6.1.3.3 Various superpixel sizes

Fig. 6.4 visually demonstrates how size of the superpixel would affect prediction performance. As mentioned, the superpixel size is an important factor to trade-off between representation efficiency and information loss. The smaller superpixels used, the lower superpixel local entropy becomes, which justifies the appropriate level of compression. On the other hand, the larger superpixels used, the more efficient the imagery representation, but also the more information loss. In addition, as the superpixel size decreases, the pixels that each superpixel contain become more homogeneous and the regression network will be exposed to a more complex feature space, meanwhile the resolution difference between the convolutional features and superpixel features increases, which makes the predicted depth map less stable and more susceptible to noise.

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

Figure 6.4: Depth estimation with various superpixel sizes. The superpixel size is a factor that will affect the prediction performance.

6.1.3.4 Superpixel feature extraction optimization

As one of the primary goals of this research to accelerate monocular depth estimation, the computational efficiency is also evaluated in terms of frame rates on a workstation. We have used a Nvidia GTX 1070 graphics card, that has 1920 CUDA cores and 8GB of video memory, as an accelerator to train the deep neural network. The hardware platform also includes Intel i5-7600K CPU. At 384×216 resolution, the network performs depth inference at 125 fps and can be trained on the dataset at 65 fps. The superpixel feature extraction, which has reorganized the feature space into a more efficient representation for subsequent neural layers, does not slow down the network architecture in the first place, including the superpixel segmentation. With the GPU accelerated superpixel segmentation [44], which is rated at 250 fps and our own CUDA implementation of superpixel feature extraction operates in excess of 100 fps. This superpixel feature extraction layer is highly optimized and is able to process images on-the-fly without noticeable slowdown relative to the speed of the rest of the architecture. Even though the primary reason that we have implemented the superpixel feature extraction in CUDA is that such operation is very infeasible to express using just linear algebra, we have also measured the computational speedup from such an optimization.

We have performed a comparison of speed of the superpixel feature extraction step. Fig. 6.5 has shown that the CUDA optimized superpixel feature extraction is

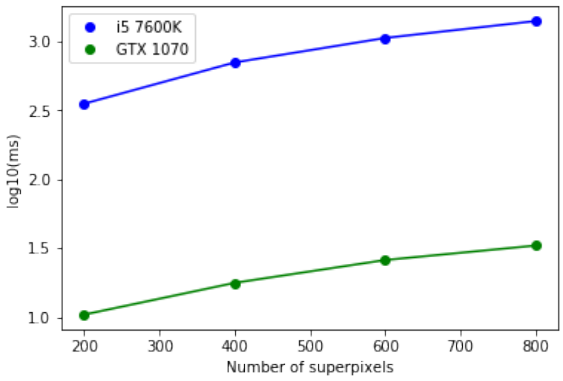


Figure 6.5: CUDA based superpixel feature extraction performance versus CPU, showing 1.5 orders of magnitude improvement.

consistently 1.5 orders of magnitudes faster.

6.2 MULTIPLE-INSTANCE LEARNING AND OBJECT DETECTION

6.2.1 Problem Description

Object recognition and localization are predominantly approached with supervised learning using labeled bounding boxes. Fully automated segmentation without supervised training data remains an intrinsically ambiguous and challenging problem. Here, we present a novel imagery relational analysis paradigm that aims to simultaneously perform object classification and segmentation by dissecting parts of objects out of the image and then explore the relation between parts and the whole object using multiple instance learning. We leverage superpixel feature extraction to spatially reorganize the feature space and to provide the neural network with a substructure consisting of images as high-level parts and objects, instead of texture, edges or geometrical primitives where image analysis is traditionally focused on. As the superpixels retain the association between the visual feature space with parts of the objects in an image, the architecture also moves towards higher explainability. We compare the results among our superpixel-based multiple-instance learning (MIL) networks with different hyperparameter settings and against a baseline deep

neural model. The proposed model has demonstrated preliminary success in the effort of bridging image object recognition with enhanced localization and provides one possible approach to the fundamental problem of simultaneously solving object classification and segmentation without complete supervision.

6.2.2 Model Architecture

The proposed *Deep Heterogeneous Superpixel MIL Network* (DHS-MIL), as shown in Fig. 6.9, consists of a DCNN, superpixel feature extraction, and a *Relational Analysis* (RA) component. The aim of the model is to learn what specific parts of the image are contributing to the image class label from multiple instance learning of a feature space that is partitioned and reorganized by superpixels. Intuitively, as the key component of the model, the RA is an ensemble of many one-vs-all MIL models, which are presented with superpixel-segmented feature bags. The MIL is constrained such that the positive bags contain at least one positive instance, the negative bags contain entirely negative instances, and are therefore satisfied for each class-specific model within RA. To facilitate such relational learning, these feature bags are obtained through the superpixel feature extraction, which reorganizes the feature space such that the visual features are reassigned to superpixels. These superpixel features are essentially inherited from the DCNN convolutional feature maps and dependent on how well the backbone DCNN is able to encode and characterize the visual aspects of the imagery. The superpixel feature extraction maintains the spatial association so that individual superpixel contribution can be mapped to the object detection and create object segmentations.

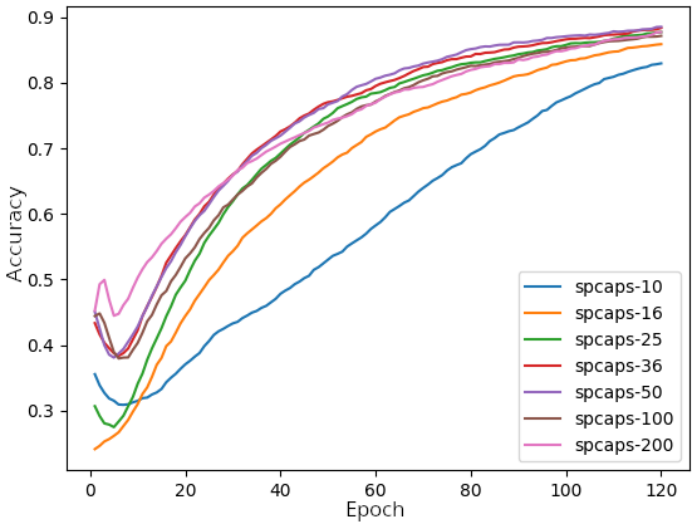
In consideration of the RA framework and popular state-of-the-art neural networks in classification, it can be implemented in the form of shared *multi-layer perceptron* (MLP) or a Capsule Network [14]. The superpixel masking is used to handle the slight variation of the number of superpixels from the superpixel segmentation. Class

masking is also used to selectively train these capsules. Only the image-level class labels are supplied and the trained model is designed to perform entity segmentation and classification through the RA model without pixel-level or superpixel-level segmentation labels.

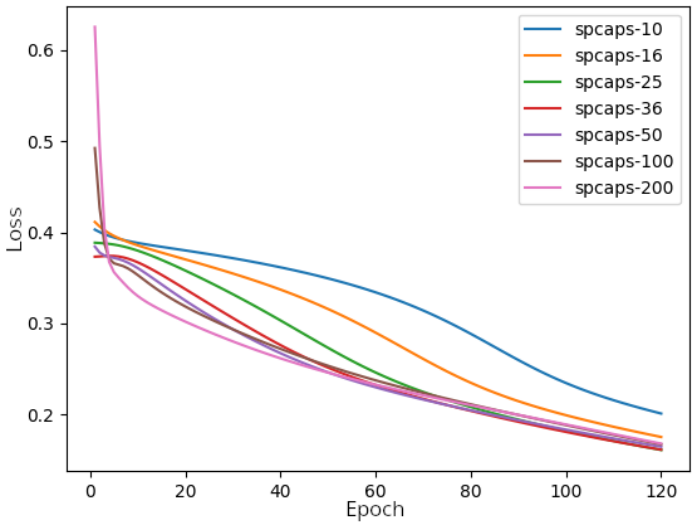
6.2.3 Results

From Fig. 6.6 and Table 6.4, one can see that the use of very large superpixel sizes, where the image content within the superpixels are no longer just monotone texture (see also 6.4b and 6.4c, etc.), can still produce a comparable predictive performance, as the curves for models *spcaps-16* through *spcaps-200* in Fig. 6.6 show. Here, the superpixel sizes are intentionally varied in order to allow the neurons to learn the superpixel features from different receptive field sizes, thereby changing the scale of the local context as the superpixel feature extraction reorganizes the feature space and readjusts the error back propagation accordingly during training.

As mentioned, the superpixel sizes are controlled indirectly by constraining the approximate number of superpixels per image upon superpixel segmentation. Fig. 6.6 shows the model convergence characteristics over 120 epoch of training by comparing the model configured to use 10, 16, 25, 36, 50, 100 and 200 superpixels per image sample respectively. The general behavior that is observed is that with higher number of superpixels per image, the model appears to achieve slightly higher training accuracy after 120 epochs of training and also exhibits slightly more over-fitting. However, as shown in Table 6.4, all 7 models have generalized consistently well on this dataset and have a consistent validation accuracy at around 88%. In Fig. 6.6a, the gap that is seen between the model with 10 versus the higher number of superpixels can be attributed to the extreme violation of the superpixel homogeneity assumption (i.e., large heterogeneous superpixels) by using very large superpixel sizes. Overall, it has been confirmed that the superpixel sizes are an important hyperparameter for train-



(a)



(b)

Figure 6.6: Comparison of training accuracy and loss across different superpixel sizes, which are indirectly constrained by specifying the approximate number of superpixels per image. Using 16-component vectors to represent entities, the general trend appears to be that the accuracy increases as smaller superpixel sizes are utilized. It can also be seen that the superpixel size affects convergence speed especially with smaller superpixel sizes. The relation between superpixel size and prediction accuracy is not always monotonic, as training metrics on the 64-component class vector experiments, in Tab. 6.4 has turned out to be a counterexample.

ing this style of part-whole recognition model, as they imply the scale of the neural receptive fields.

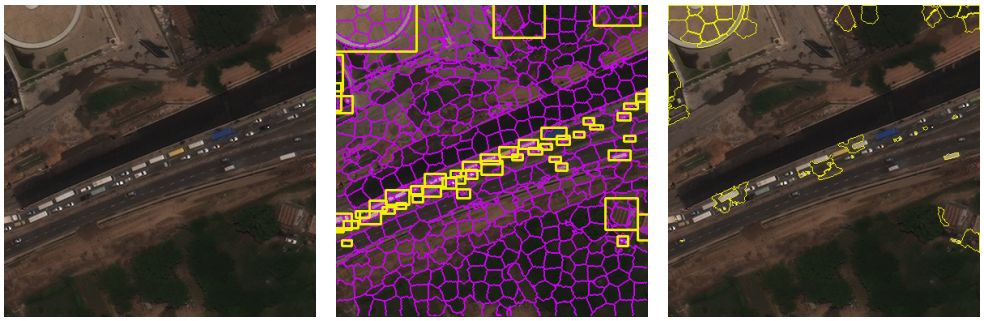
6.3 REMOTE SENSING OBJECTION LOCALIZATION WITH HETEROGENEOUS SUPERPIXEL FEATURE

6.3.1 Problem Description

Object detection and localization within high-resolution remote sensing imagery (HR-RSI) is a challenging task for a variety of reasons, such as the complexity and clutter of the image scene and the compactness of the intermixed object classes. Even the most comprehensive training datasets cannot adequately account for the rich diversity and complexity of anthropogenic objects and their contextual settings in large-scale HR-RSI collections. Recent approaches using deep learning techniques include bounding box approaches (e.g., YOLO), object nomination then recognition (e.g., R-CNN), and post-detection object localization of deep neural network detections. Herein, we propose a novel technique that leverages heterogeneous superpixels and deep neural feature extraction to classify the superpixel segmentation through relational analysis [47]. In this research, we demonstrate the validity of this approach for object detection and localization, as well as its suitability for identifying the irregular shapes of objects (as opposed to a bounding box). Experiments are performed using a sub-set of the xView benchmark dataset with a goal of spearheading future techniques in cluttered scene object recognition that allows deep feature extractors to have more focus on the target objects instead of the surrounding area or nearby object pixels.

6.3.2 Geospatial Imagery in Superpixels

Geospatial imagery and HR-RSI have their own unique challenges when it comes to object detection. Notably, the imagery objects are mostly captured from an overhead view or with a slight off-nadir angle [106] [107]; and the imagery objects of interest



(a) (b) (c)

Figure 6.7: An example of heterogeneous superpixels: (a) sample input image; (b) heterogeneous superpixels (magenta) and object bounding boxes (yellow); (c) derived superpixel labels. Heterogeneous superpixels are meant to highlight parts of imagery objects, but not necessarily be homogeneous in texture. In some cases, this approach can also alleviate noise introduced by the background compared to rectangular bounding boxes. For example, the tower in the top left corner is now more accurately labeled as the imagery is converted to superpixel representation.

often have a very small spatial extent [108] and are typically densely clustered. In contrast to large prominent objects in ImageNet [109] or COCO [3], the imagery typically is in a high resolution and the training sample frequency varies drastically from class to class. In the domain of remote sensing, the resolution of the imagery is typically defined by *ground sample distance* (GSD), which measures the physical distance between pixel centers. The xView [107] dataset used in this research for evaluation, for example, is reported to be 0.3 m GSD.

A superpixel representation of the imagery provides the ability to efficiently handle the high resolution and also associate the training inputs and the feature maps across different resolutions. The superpixel sizes are chosen so that their areas are 6 to 30 px^2 to match the size of the imagery objects from small cars to buildings because compared to daily photo object detection, even heterogeneous superpixels have to be considerably smaller to detect objects from HR-RSI. In this preliminary research, we present an evaluation of deep heterogeneous superpixel feature based models with a selection of object classes from the xView [107] dataset (detailed in Sect. 6.3.2) that

do not vary drastically in sizes. Provided the superpixel sizes are carefully chosen to represent parts of the imagery objects and not to under-segment, the edges of the superpixels will align with the edges of the objects, thus alleviating the error introduced by rectangular bounding boxes. As such, in a superpixel representation, the accuracy threshold is only based on classification probability instead of the intersection over union (IoU) metric.

This heterogeneous superpixel representation of parts of objects not only has provided localized DCNN features with tight and crisp boundaries but has also reorganized the feature space in a much more interpretable manner for object detection. The imagery can be processed as a collection of localized 1-D vectors, instead of 2-D signals, thanks to heterogeneous superpixel. By computing the superpixel-wise feature contribution to the class prediction, a spatial probability map can be generated that represents the relations between superpixels and the object recognized, which makes it very feasible to segment the object of interest.

Fig. 6.7 is a visualization of how heterogeneous superpixel feature extraction works spatially. The extracted superpixel features will be derived from the DCNN feature maps that are spatially aligned with the superpixel, although in different resolutions. Then, the superpixel will obtain a label according to the intersecting labeled bounding box. Afterward, the relational analysis will learn to encode imagery object classes and the relation between superpixels and class objects in vector representation, so that it can make superpixel-level predictions.

The processing resolution chosen was 256×256, mainly limited by the DCNN as a pre-trained DCNN as a feature extractor. As the smaller sized chips are derived from xView data, a 25% overlap is considered. A raster-based superpixel segmentation and feature extraction algorithm are then used on these image chips to obtain a superpixel feature map.

6.3.2.1 xView Dataset

The xView [107] challenge dataset was released by DIUx in 2018. Some samples of the imagery dataset are shown in Fig. 6.8. The dataset has only 150 scenes, but those scenes encompass over a million total objects, each with their own bounding box metadata. The training portion of the dataset has just over 600,000 objects belonging to 60 classes, resulting in an average number of samples of 10,030.97 samples/class. While there are a plethora of samples available, the dataset is heavily unbalanced, with a range of samples per class ranging from as low as 17 and as high as 316,765 objects/class.

An object label will be assigned from the xView metadata by ranking the bounding boxes which contain the centroid of the superpixel by area in ascending order. A smaller area suggests that the bounding box is more specific, and consequently, the superpixel will be primarily labeled the same as the smallest box. The label multiplicity can also be considered as a confidence factor for the superpixel label. As such, the geospatial satellite imagery data will be converted into appropriately labeled superpixels, associated with various types of DCNN features.

On average, for each image from the xView dataset, 20K to 300K superpixels will be generated, depending on the constraint of the approximate size of superpixels. Proportional to the bounding box density, a fraction of these superpixels will be matched with some bounding boxes from xView metadata and be labeled. For each DCNN feature extractor and superpixel size class, we generate 2.38M to 9.5M superpixels that match any bounding box from the xView data for object recognition. Although, as a preliminary evaluation, only a small fraction of this data was used.

6.3.3 Model Architecture

Considering the many advantages that a deep heterogeneous superpixel based model brings to address some of the challenges, particularly for HR-RSI, briefly covered

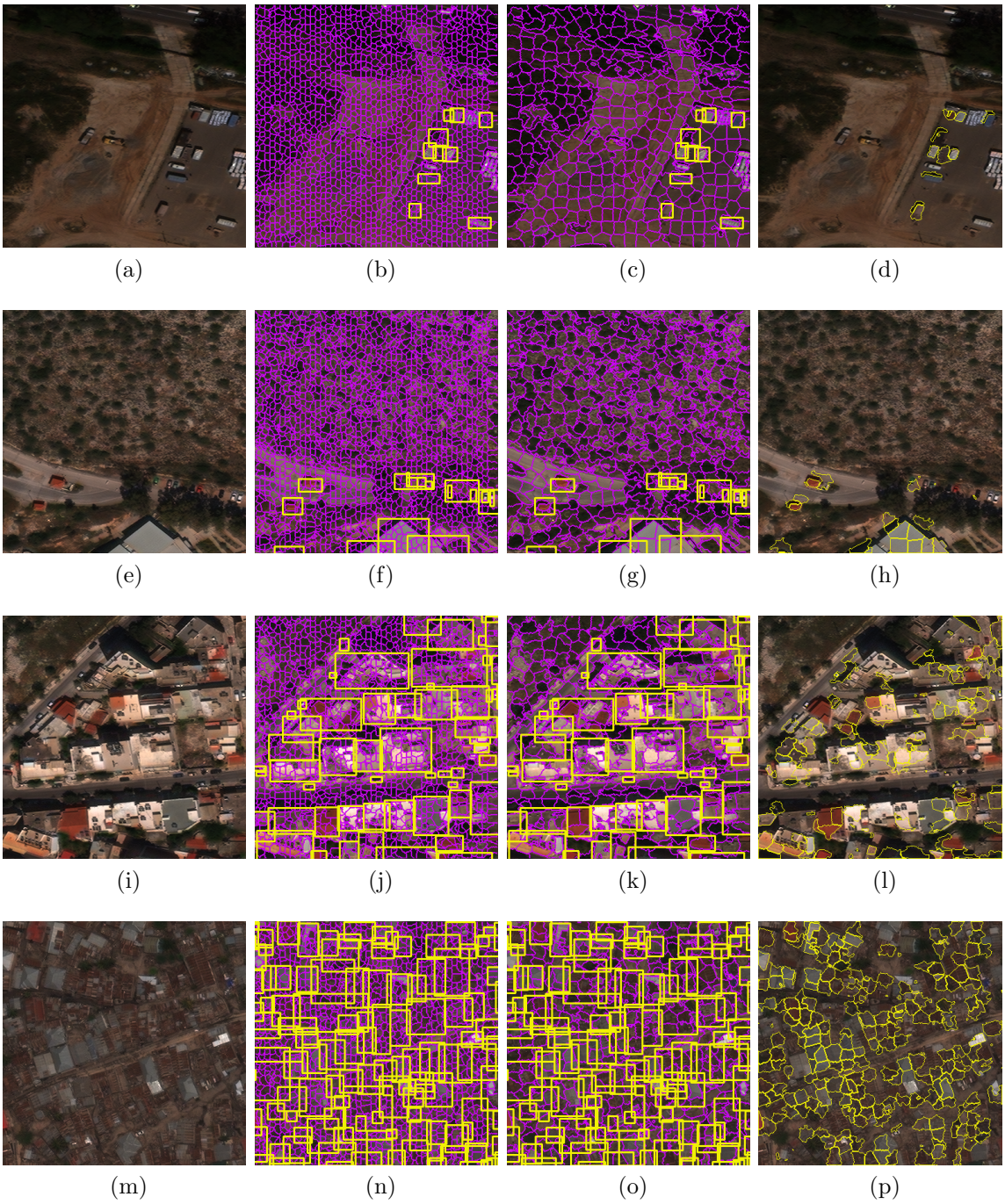


Figure 6.8: Examples of xView imagery: the first column is sample input images; the second and the third column are superpixel (magenta) segmented images with $8px^2$ and $18px^2$ as their expected superpixel sizes respectively, along with bounding boxes (yellow) from the xView dataset; the fourth column are derived labeled superpixels for training. Columns 2-4 are visualizations of the training data from which the neural network are trained on object oriented 1-D vectors on a superpixel basis, as opposed to any of these 2-D images as visualized.

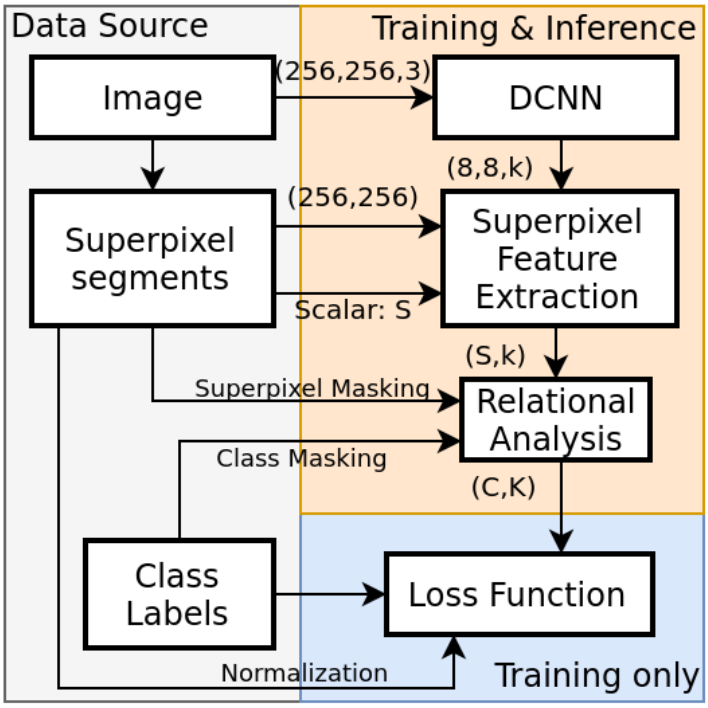


Figure 6.9: Deep Heterogeneous Superpixel Network: The superpixel feature extraction replaces the spatial dimensions of the convolutional feature maps with superpixels. The relational analysis can learn and associate extracted features to object classes to make predictions for the superpixels. The superpixel masking is used to handle the slight variation of the number of superpixels from the superpixel segmentation, and to selectively train these capsules. Only the class labels are supplied, and the trained model performs entity segmentation and classification through the part-whole relation modeling without pixel-level or superpixel-level segmentation labels.

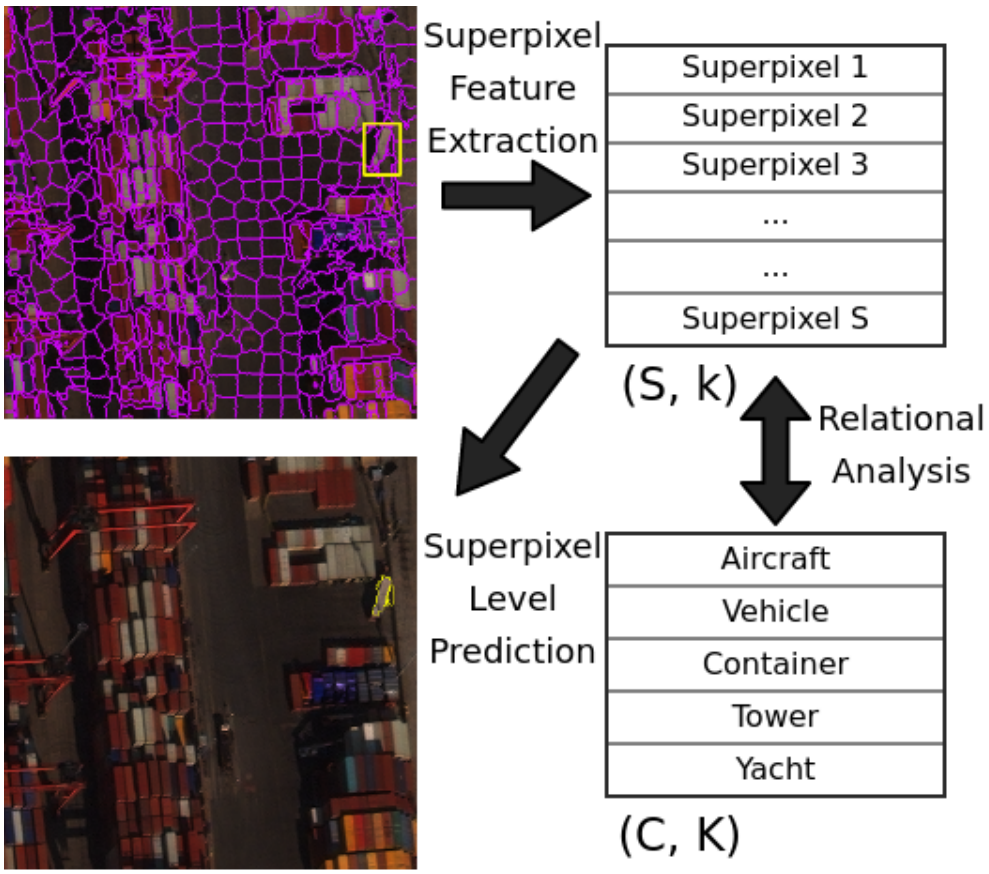


Figure 6.10: Feature Space Reorganization using Superpixels: After superpixel feature extraction, the feature space is converted from 2-D convolutional feature maps $(8, 8, k)$ to a object-oriented feature space (S, k), where parts and class objects are represented as 1-D vectors, which facilitates the higher level relational analysis, in order to learn to associate parts with object classes and make superpixel-level predictions. Magenta polygons are the irregular shaped superpixels, and the yellow rectangle is the object bounding box.

in Sect. 6.3.1, a novel neural network architecture, Deep Heterogeneous Superpixel Network, is evaluated, whose structure is also shown in Fig. 6.9. In order to evaluate the effectiveness of image object recognition based on superpixel relational analysis, comparisons are set up between a DCNN model with shared MLP and Capsules. The superpixel feature extraction constructs a reorganized feature space in order to facilitate the part-whole relational analysis while maintaining the spatial association. Therefore, one can back-project the contribution brought by an individual superpixel once the model is trained and an entity segmentation is obtained. The evaluated models share a common DCNN to generate the base convolutional feature maps, but the superpixel size is varied across different models, which was the hyperparameter that has demonstrated the greatest impact on the model accuracy and speed of convergence during training. We have evaluated VGG-16 [21] and ResNet-50 [22] as DCNN feature extractors. As one can see that the superpixel feature space reorganization is based on visual features, various kinds of DCNNs can be utilized as visual feature extractors. In this research, VGG-16 and ResNet-50 were chosen for two main reasons: (1) to not over complicate the exploration of the ideology of superpixel feature based networks, direct experimentation on very large scale dataset at this stage to facilitate better inspection of the data processing; and (2) if the new feature spaces are a better internal representation for object recognition, then the network should not require as a deep neural network architecture as most popular deep learners [14]. The authors of VGG [21] and ResNet [22] have released their best-performing models to facilitate further research.

The pre-trained DCNN network weights are used as the initial state of the network. This technique is known as transfer learning, which allows models to learn faster by leveraging existing weights from a related problem domain, and transfer this knowledge gained in solving the source problem to solving the problem of interest.

As the DCNN feature extractors generate 8×8 visual features with k channels, the superpixel feature extraction maps the 8×8 visual features to the superpixels according to Eq. (4.4). The superpixel feature extraction requires convolutional features and a superpixel segmentation, and the resulting superpixel feature map will have the same amount of channels as the convolutional feature maps. As a result, the superpixel feature map is a 2-D data structure ($S \times k$) per image sample, but neither axis is a spatial dimension of the image; instead, they are superpixels S and channels k because the spatial axes of the visual features are merged into one and can be restored given the superpixel segmentation.

Lastly, the relational analysis layer is implemented as 4 capsules with classes encoded as a 16-D vector, where the probability of the entity existing in the image is represented by the magnitude of the class vector [14]. To carry out the relational analysis, the $S \times k$ superpixel feature map is interpreted as S low-level capsules, which encode low-level features as a k-vector.

6.3.4 Results

6.3.4.1 Training and Evaluation Procedure

Using *Adam optimizer* [110] with a learning rate of 1×10^{-4}, various Deep Heterogeneous Superpixel Networks were trained with different superpixel sizes, 8px^2, 18px^2 and 24px^2 , for 30 epochs. All the models are initialized with pre-trained weights from ImageNet [21]. The 30-epoch limit does not necessarily drive the models to a learning saturation point, but the goal is to set up a comparable training setting between different models, such that they all converge in a relatively stable manner. The size of the superpixels is important because it defines the scale of image parts whose features will be examined by the subsequent relational analysis, which currently can be shared MLP or Capsules [14].

For each image, we can derive predictions per each superpixel as described in Fig. 6.9 model architecture, and choose the class with max confidence level as the final prediction. Following [111] and [108] the mean average precision metric (mAP) is used for comparison, and the speed is calculated by converting chip size into km^2 according to the GSD and then multiplying by the inference frame rate (around 65fps).

6.3.4.2 State-of-the-art Comparison

Table 6.4 shows the predictive and computational performance along with some state-of-the-art models [4] [111]. To clarify, although we attempt to compare with state-of-the-art models, due to the architectural differences, the mAP metric used is fundamentally different from these popular models. The mAP in this research refers to superpixel level average precision, whereas the mAP used in [111] [4] [108] is based on the intersection over union (IoU) threshold, which is a popular metric to evaluate rectangular bounding box accuracy.

From Tab. 6.4, it is evident that the DCNN feature extractor can make a difference for the classification; where the **DCNN** is the feature extraction component and the **RA** is the relational analysis component of the proposed architecture. As the superpixel feature extraction can be viewed as a type of spatial aggregation or reorganization, it will directly inherit from the extracted DCNN features. The ResNet-50 based features performed better with the same training conditions. With regard to the relational analysis, it is shown that the neural network can benefit from the separated feature mapping and routing using Capsules compared to a shared MLP.

Additionally, we measured the inference rate in square kilometers per second of imagery that can be processed with the various approaches. From the computational performance perspective, it is still relatively fast among existing models. YOLO and YOLT [111], being much more straightforward network architectures in terms of

Table 6.4: Training Results

Model variations		Performance	
DCNN	**RA**	mAP	km^2/s
VGG16	Shared MLP	0.23	0.38
Res50	Shared MLP	0.46	0.26
VGG16	Capsules	0.27	0.22
Res50	Capsules	0.52	0.18
Res101	Faster RCNN [108]	0.23	0.09
N/A	YOLT [108]	0.68	0.44

computational complexity, are the only slightly faster per square kilometer than our method.

6.4 WEAKLY-SUPERVISED OBJECT DETECTION

6.4.1 Problem Description

As mentioned in Chapter 5, Weakly-supervised Object Detection (WSOD) aims to create an object detector using coarsely annotated datasets. This could be image-level or ROI-level annotations, while the desired output is a pixel-level segmentation. It is desired because then many coarsely annotated datasets can also become available for training in object detection, location, segmentation, and other tasks beyond image classification. Meanwhile there is an increasing demand in promoting model explainability to achieve higher understanding of the models' behavior from a human's perspective, because in some applications it can be important when incorporating the prediction into the decision-making. Apart from the aforementioned goals, creating an end-to-end trainable model has always presented a technical obstacle until MIL-based neural networks became successful and prominent.

6.4.2 Model Architecture

To solve the WSOD problem, we created a novel neural network architecture utilizing heterogeneous superpixel feature extraction, relaxed min-cut and MIL that based on Proposal Cluster Learning [68]. It is an end-to-end trainable architecture. The details of the architecture can be found in Chapter 5 and specifically Fig. 5.4.

6.4.2.1 Experiment Setup Details

We first compare the differences between the effectiveness of superpixel proposals based on transfer-learned separate stage model and that of a transfer-learned jointly optimized WSOD model. As mentioned in Sect. 5.2.4, the proposed architecture is a two stream network. One stream is a typical DCNN network, the other is specialized in WSOD based on heterogeneous superpixel object proposals. The goal of comparing and contrasting those methods can allow us to understand the benefits of being able to jointly optimize the end-to-end trainable model. For the DCNN stream, we transfer pre-trained weights from ImageNet [1], and alter the top layers to the appropriate number of classes according to the dataset. The pretrained DCNN network weights are used as the initial state of the network. This technique, known as transfer learning, can allow the models to learn faster by leveraging existing weights from a related problem domain, and transfer this knowledge gained in solving the source problem to solving the problem of interest. The input image will be center-cropped to 224×224 pixels. As the DCNN feature extractors generate 6×6 visual features with 512 channels, the superpixel feature extraction maps the 6×6 visual features to the superpixels according to Eq. (5.8). And the resulting superpixel feature map will have the same amount of channels as the convolutional feature maps.

We experiment with the Resnet-34 [22] and the Inception [112] networks because they have achieved best results from prior research [11, 47, 113, 114]. For the WSOD

stream, the heterogeneous superpixel will be derived from the DCNN intermediate output just before the 2-D global pooling or flattening, depending on the type of layer used in the original DCNN. Therefore, the DCNN features serve as a base for creating localized high-level object proposal features. Whose output have a very small spatial extent such as 6×6 from the Resnet-34. Then we follow the PCL WSOD model [68] to set up training hyperparameters because our MIL network in the WSOD stream resembles the basic configuration of the PCL network. Specifically, we use SGD optimization and set the learning rate to 1×10^{-3} and reduce it to 1×10^{-4} towards the last 1/4 progress of the training. This learning rate setting is high because the significant amount of layers and parameters are transferred from a pre-trained DCNN. The momentum is set to 0.9.

We used Dlib [115] for neural network architecture, gSLIC [44] for initial superpixel segmentation, cuBLAS for GEMM computation and custom-built CUDA kernels for heterogeneous SFE and image conversion, annotation and processing to implement the entire architecture, which has been developed and experimented with on NVIDIA GTX 1650, 1080, Tesla P100 GPUs. Some parts of the model with graph-theoretical algorithms working on sparse and light data structures are running on CPU. The datasets we use for evaluation are VOC 2012 [2] and MS-COCO [3]. They are large-scale datasets typically used for image classification and segmentation tasks. Additionally, prior WSOD research [68, 116, 117] has also used them for validation of their methods. The results are reported in Tables 6.5 and 6.7 for the two datasets, respectively.

6.4.3 Results

The naming convention of the experiments are as follows. The "HS-" prefix represents our heterogeneous superpixel based WSOD architecture, which is followed by the DCNN backbone network, whether the two stream archtecture is independently

Table 6.5: AP (%) results on VOC 2012 dataset.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	mAP
WSDNN [81]	64.0	57.9	36.4	8.1	12.6	53.1	40.5	28.4	6.6	35.3	35.3
Jie et al. [116]	60.8	54.2	34.1	14.9	13.1	54.3	53.4	58.6	3.7	53.1	38.3
PCL-VGG16 [68]	58.2	66.0	41.8	24.8	27.2	55.7	55.2	28.5	16.6	50.1	40.6
PCL-FRCNN [68, 82]	69.0	**71.3**	56.1	30.3	27.3	55.2	57.6	30.1	8.6	56.6	44.2
HS-Res34-0-128	80.3	57.0	62.2	35.8	47.0	77.2	75.2	60.0	29.4	80.8	58.2
HS-Res34-1-128	75.1	47.3	**82.7**	50.7	58.4	77.2	67.1	82.3	19.7	84.5	64.6
HS-Inception-0-128	87.1	54.0	79.1	79.5	76.8	79.0	75.3	87.6	29.8	67.6	67.6
HS-Inception-1-128	**87.2**	50.8	64.3	**87.8**	**81.8**	**82.7**	70.6	**93.2**	**35.7**	83.2	**73.3**
HS-Res34-0-64	81.1	59.0	53.9	55.6	44.1	70.9	70.1	77.1	27.0	75.4	57.4
HS-Res34-1-64	75.6	51.3	54.6	46.7	50.8	77.4	69.6	69.3	26.6	77.0	58.5
HS-Inception-0-64	77.7	60.6	66.4	84.5	72.5	75.4	**78.3**	57.8	31.0	**85.6**	64.8
HS-Inception-1-64	87.0	41.9	77.1	65.5	68.7	80.0	76.4	75.6	33.5	80.6	67.5

Table 6.6: AP (%) results on VOC 2012 dataset.

Method	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
WSDNN [81]	34.4	49.1	42.6	62.4	19.8	15.2	27.0	33.1	33.0	50.0	35.3
Jie et al. [116]	8.3	43.4	49.8	69.2	4.1	17.5	43.8	25.6	55.0	50.1	38.3
PCL-VGG16 [68]	17.5	28.6	49.7	70.5	7.1	25.7	47.5	36.6	44.1	59.2	40.6
PCL-FRCNN [68, 82]	18.4	43.9	64.6	71.8	7.5	23.0	46.0	44.1	42.6	58.8	44.2
HS-Res34-0-128	15.7	76.7	74.1	**67.8**	36.8	37.5	50.0	63.7	68.6	68.9	58.2
HS-Res34-1-128	41.2	70.4	89.8	63.2	47.8	37.6	**73.5**	66.7	**80.2**	76.5	64.6
HS-Inception-0-128	29.9	68.4	84.5	53.9	69.3	41.2	70.0	65.9	76.5	77.0	67.6
HS-Inception-1-128	**47.0**	**86.3**	91.5	57.3	**81.4**	**50.2**	71.8	73.4	73.6	**95.9**	**73.3**
HS-Res34-0-64	22.9	58.1	79.9	61.0	28.2	30.2	49.1	59.0	74.5	71.4	57.4
HS-Res34-1-64	22.1	80.4	78.9	62.9	34.5	35.8	65.6	55.2	68.4	68.4	58.5
HS-Inception-0-64	25.8	86.2	78.5	60.5	55.4	42.0	51.4	59.6	62.7	85.0	64.8
HS-Inception-1-64	39.6	79.6	**92.3**	55.8	61.6	**50.2**	56.5	**75.1**	70.4	83.1	67.5

trained "0" or jointly trained "1", and finally the SFE layer feature dimensionality. As mentioned, for the comparison of the DCNN backbone network, from which the SFE features are derived, we compare the Resnet-43 network and the Inception network. For SFE feature dimensionality, we compare the difference between 64 and 128 neuron units in the fully connected layer attached to the SFE.

6.4.3.1 Datasets and Evaluation Metrics

We evaluate our method on the PASCAL VOC 2012 dataset [2] and the MS-COCO [3] dataset. Only image-level annotations are used to train our WSOD models. The PASCAL VOC 2012 dataset has over twenty thousand images with 20 object classes. The MS-COCO dataset has 80 object classes. The mean of average precision (mAP) is the most popular evaluation metric for image segmentation as well as WSOD models. The average precision (AP) is calculated by approximating the area under the recall-precision curve. Then the mAP is, by definition, the mean of the AP across object classes. It is also often acceptable to simply refer to both of the metrics as AP, as their meaning can be inferred in the context, i.e. single class or multiple classes, without ambiguity. Per the standard PASCAL [2] criterion, in a typical bounding box based model, the groundtruth bounding boxes are to be compared with the predicted bounding boxes, thresholded at a minimum of 0.5 intersection over union (IoU), also known as the *Jaccard Index*,

$$\delta_{\text{JACCARD}}(A, B) = \frac{|A \cap B|}{|A \cup B|} \ . \tag{6.10}$$

Whereas the MS-COCO [3] dataset has proposed to evaluate across a range of IoU thresholds from 0.5 to 0.95 at each 0.05 step. This is also known as the 101-point AP. However, with precise WSOD where bounding boxes are replaced with more precise boundaries, continuing to make the AP measurements between groundtruth bounding boxes and predicted precise object boundaries would be neither reasonable nor convenient. Therefore, we emphasize that the AP measurement in this work, although follows the same definition and principles, it is a comparison between precise groundtruth and precise predicted boundaries, instead. That is to say, the AP metric is analytically consistent, according to the *Jaccard Index* definition, as shown in Eq. (6.10), as a similarity metric between sample sets, but conceptually different

from the usual rectangular IoU [4, 5], which is something to keep in mind when interpreting the results numerically. To compare precise ground truth against precise predictions entails that the pixel-level segmentation masks are still a requirement to compute the metric, but prohibited in any loss function terms for the purpose of WSOD. Practically speaking, for datasets where only bounding boxes are available, this can still be approximated by intersecting the bounding boxes with the superpixel segmentation [47],

$$\delta_{\text{JACCARD}}(A, B) \approx \delta_{\text{JACCARD}}\left(\bigcup V_1|_{\hat{A}}, B\right) \ , \tag{6.11}$$

where A is the unavailable precise ground truth, $\hat{A}$ is the available bounding box ground truth, B is the predicted precise object region, $V_1|_{\hat{A}} = \left\{S \in V_1 | S \subseteq \hat{A}\right\}$ represents a restricted superpixel set, and the unbound union notation $\bigcup V$ is equivalent to $\bigcup_{S \in V} S$, without explicitly calling out the individual element sets within that are subject to the union. This approximation, Eq. (6.11), holds assuming the superpixel segmentation V_1 has high edge recall, low oversegmenation, and the bounding box $\hat{A}$ is adequately tight, to achieve this approximation. As mentioned in Sect. 5.1.2, in the context of heterogeneous supeprixel, these already are existing assumptions. The relation between Eq. (6.11) and $\delta_{\text{JACCARD}}(\hat{A}, B)$ is not trivial, as both the numerator and the denominator in the Jaccard Index in the latter will decrease. When there are only image-level object class labels, this metric is not at all informative because,

$$\begin{aligned} \hat{A} = V_0 \Rightarrow \bigcup V_1|_{\hat{A}} = \bigcup V_1 = V_0, \\ \delta_{\text{JACCARD}}\left(\bigcup V_1|_{\hat{A}}, B\right) = \delta_{\text{JACCARD}}\left(V_0, B\right) = \frac{|B|}{|V_0|} \ , \end{aligned} \tag{6.12}$$

which reveals nothing about $\delta_{\text{JACCARD}}(A, B)$, also indicating that a meaningful AP metric is unattainable. That is to say, practically speaking, if a dataset only contains image-level labels, we can train a WSOD model, although it would be challenging to

Table 6.7: AP (%) results on MS-COCO dataset (continued).

Method	mAP@0.5	mAP@[0.5,0.95]
Ge et al. [117]	19.3	8.9
PCL-VGG16 [68]	19.4	8.5
PCL-FRCNN [68, 82]	19.6	9.2
HS-Res34-0-128	19.3	8.3
HS-Res34-1-128	21.1	9.5
HS-Inception-0-128	23.7	11.2
HS-Inception-1-128	**25.2**	**11.9**
HS-Res34-0-64	18.7	7.1
HS-Res34-1-64	18.5	8.0
HS-Inception-0-64	21.0	8.7
HS-Inception-1-64	23.9	10.2

quantitatively verify.

6.4.3.2 Jointly optimizing the two streams

Unlike the prior research PCL [68] that has a similar base MIL architecture, our model is designed as a two-stream network due the non-convex optimization that exists in the graph algorithm mentioned in Sect. 5.2.4. The weighted loss function can help adjust and re-route gradient backpropagation as needed. From Tables 6.5 and 6.7, we can compare the variations of the architecture that are having the two streams trained independently or jointly. To train independently, the gradient backpropagation from SFE to the DCNN is cut off. Since both streams will finish with a distinct set of dense layers and categorical cross entropy loss, they both have existing gradient backpropagation paths to tune their parameters respectively. To train jointly, we send a small amount of the gradient contribution (10%) from the WSOD stream back into the DCNN, through SFE, as shown in Eq. (5.10). This is because the main task of the DCNN stream is to still accurately classify images, but not to overly accommodate feature space construction for WSOD purposes.

The results have shown that from a independently trained two-stream architecture, the WSOD stream can still achieve good results. In this scenario, the WSOD stream makes the proposal-level classification decision solely based on existing information from a pre-trained DCNN. And we have always known such localized decision-level information exists in the trained DCNN layers, such as from [118, 119, 120], etc., research that explores the attention in the CNNs. From our proposed architecture, comparing the models that are independently trained and jointly trained, we see an overall performance boost by jointly training the model. However, from the per-class metric in Table 6.5, we can also see that this is not necessarily consistent for each class, and the deviation is still high due to difficulty in training convergence. Moreover, this performance boost is also dependent on the DCNN backbone architecture and other factors and it is not trivial to come up with such expectation without experimenting with some specific architecture. Comparing the two datasets, we can still see a noticeable overall performance boost, but as the number of the classes increases in the MS-COCO dataset, as shown in Table 6.7, the AP also drops drastically.

6.4.3.3 Comparing DCNN Backbone Networks

Among these experiments, we are comparing two different types of DCNN backbone networks, the Resnet-34 [22] network and the Inception [112] network. Overall, the Inception network produces higher WSOD performance. Upon a closer look into the per-class metric, in Table 6.5, this performance boost also varies from class to class. For example, in the "boat" object class, there is a significant difference between the two network architectures. The same can also be observed in the "person" object class. Meanwhile, we see some performance regression in the "mbike" and "bike" object classes. As mentioned, the proposed architecture is highly non-linear and discrete, i.e. a mistake in the early superpixel segmentation, or proposal generation stages, can lead to chaotic results in the subsequent MIL analysis. As the heteroge-

neous superpixel feature space is inherited and built upon the DCNN features, the choice of the backbone DCNN architecture naturally has great impact on the model's overall performance. The experimental data has shown the general principle that the more recent and sophisticated models can derive a higher level of abstraction of the object classes and, therefore, lead to better heterogeneous superpixel and object proposal generation performance. This is also crucial for the MIL proposal scoring subnetwork to achieve the goal of WSOD. Our proposed architecture has provided a pathway to bring together and build upon these state-of-the-art deep neural network models.

6.4.3.4 Adaptive SFE

As shown in Eq. (5.9), for the purpose of WSOD, we have experimented with a form of adaptive SFE by attaching a shared fully connected layer after the SFE and computing the appropriate gradient backpropagation to make this layer learnable. The shared fully connected layer not only serves as dimensionality reduction, but it also makes the SFE feature space adaptable to suit the problem type suggested by the subsequent network layers. The most important hyperparameter is the number of neuron units used in this shared fully connected layer, which corresponds to the dimensionality of the heterogeneous superpixel features. However, from our current experiments detailed in Table 6.5, we did not observe a clear connection between the dimensionality of the SFE feature space and the WSOD AP performance on a per-class basis. There are substantial improvements in the overall mAP metric from the experiments done on both the VOC 2012 and MS-COCO dataset. Since there exist many hyperparameters, such as the initial superpixel size and compactness, dimensionality of the SFE layer output, and the number of min-cut clusters for generating the object proposals that will all affect the model prediction outcome, we believe further research is needed to alternately study these factors before we can further better utilize the adaptive SFE to

achieve a more predictable outcome in specific cases. Among these hyperparameters, in this dissertation, we will further discuss the effects of superpixel sizes through visual examples in Sect. 6.4.4.

6.4.3.5 State-of-the-Art Comparison

Overall there are significant improvements compared to prior research. The overall precision is improved by these heterogeneous superpixel based deep neural networks. On the VOC 2012 dataset, the heterogeneous superpixel approach has more distinct improvements. This is consistent with the fact that the VOC 2012 dataset has fewer number of object classes, as a greater number of classes will typically require a higher feature space dimensionality and more parameters in the MIL portion of the model to make an accurate object proposal scoring. Other factors that contribute to the numerical difference are also due to the difference in the algorithm of AP measurement due to the special requirements of the precise WSOD problem, specifically, that the prediction needs to achieve a higher granularity than the training data. This has been discussed in detail from Sect. 6.4.3.

Our approach bears a similar ideology from PCL models [68] in the MIL part of the model, which works by evaluating object proposals and providing classification individually. Other alternatives neural network architecture to fulfill the purpose of MIL have also been outlined in [68], although not investigated in the same study. However, our proposed model targets obtaining precise object boundaries, and therefore we utilized the superpixel approach. Especially, the heterogeneous superpixel technique is utilized to exploit DCNN features, and the results have shown that these models can greatly benefit from joining together these state-of-the-art techniques, including DCNN, heterogeneous superpixel and multiple-instance learning (MIL).

6.4.4 Qualitative Assessment

To obtain the detected object boundary, we select the object proposal according to the highest proposal score $\boldsymbol{Y}_1$ in Eq. (5.15), and then trace back to the SFE superpixels according to the min-cut assignment matrix $\boldsymbol{C}_1$.

By comparing the WSOD segmentation performance across various classes on the VOC 2012 dataset, it has been shown that the average superpixel size is still playing a significant role in adapting to the spatial extent of the imagery object and affects the effectiveness of precisely capturing the object in the generated object proposals. As suggest by prior research [84], if the heterogeneous superpixel size is too small, the outcome can be susceptible to perturbations from the predicted convolutional feature maps; on the other hand, if the heterogeneous superpixel size is too large, then the tolerance in local appearance can be too high to maintain the precision of the object boundary. To achieve the best outcome, further exploration to find a way to fine-tune the superpixel sizes based on the training data and each specific object class can be very beneficial. To this end, we have also visually compared the effects of different initial superpixel segmentation sizes. From Fig. 6.11, we can observe some pros and cons of using large and small superpixel sizes respectively.

For example, comparing Fig. 6.11f and Fig. 6.11g, heterogeneous superpixels can present as overly under-segmented occasionally, as shown in Fig. 6.11f when the object size is too small and the superpixel compactness factor outweighs edge retainment, causing the under-segmentation. When a smaller superpixel size is used, as shown in Fig. 6.11g, the overall precision can be improved, at the cost of spurious inclusion of irrelevant areas per each object, or false positive regions. Secondly, visually comparing Fig. 6.11n and Fig. 6.11o, both supeprixel sizes had a similar level of the false positive regions included. Similarly, comparing Fig. 6.11b and Fig. 6.11c or Fig. 6.11j and Fig. 6.11k, no improvements are visually observable by using a smaller superpixel size, while the computational burden has definitely increased. Thirdly, by comparing

the results at different superpixel sizes in Fig. 6.12, some limitations can also be seen.

"People" are among some of the challenging object classes for the proposed model. The surrounding environment in the background and various clothing styles can be make it difficult to initially dissect objects with superpixels. As a result, the derived superpixel features may not well represent the imagery objects in the first place, which creates a challenge for the subsequent MIL network. With the example of the bikes in Fig. 6.12j and Fig. 6.12k, some false positive areas seem inevitable in both cases. Technically, there are also false positive areas in between the wheel spikes, which are almost never considered as false positive areas, to our knowledge, in contemporary computer vision. However, in the context of precise WSOD, whether these regions should be counted or whether it is useful to include or exclude these regions can be an open question. As such, we conclude that the large-sized heterogeneous superpixels does provide proper object boundaries in many scenarios, to be associated with DCNN features later on for object proposal scoring, with the benefit of lowering the computational burden.

This dissertation has presented a novel neural architecture, which comprises state-of-the-art DCNN, heterogeneous superpixel and multiple-instance learning (MIL), to achievel the goal of precise weakly-supervised object detection (WSOD). We have proposed and demonstrated the adaptive superpixel feature extraction (SFE) based on heterogeneous superpixels. The heterogeneous superpixel is an ideal technique to relate low-level and high-level image features and align DCNN features to precise object boundaries and subsequently generate object proposals. We have evaluated this approach for the purpose of precise WSOD, where it is possible not only to detect and localize objects, but also to obtain precise object boundaries without strongly (or explicitly) supervised training data in that regard. Among the heterogeneous supeprixel based approaches that we have evaluated, they have demonstrated overall superior average precision (AP) on the VOC 2012 dataset and the MS-COCO dataset.

The per-class AP has a high variability, suggesting it is a superior approach especially in certain scenarios where the heterogeneous superpixel generates more proper object proposals.

We have also discussed the possible causes of such predictive performance variability with visual examples as a qualitative assessment. Overall, the benefits of heterogeneous superpixel outweighs its limitations in our proposed approach. Some of the limitations may be addressed by improving the adaptive SFE layer to incorporate feedback to fine-tune the generated superpixel sizes, and generating a better representation at the superpixel feature space. The heterogeneous superpixel is a versatile technique that allows WSOD models to be constructed upon existing state-of-the-art DCNNs, and provides object proposals for MIL, and generates insightful visualizations for better model explainability and guidance for further improvements. Our proposed model has achieved high performance in solving precise WSOD, and has presented a new paradigm to approach this problem using the heterogeneous superpixel technique. This may also be beneficial to future research where precise object segmentation or boundaries are needed without full pixel-wise labeled training data.

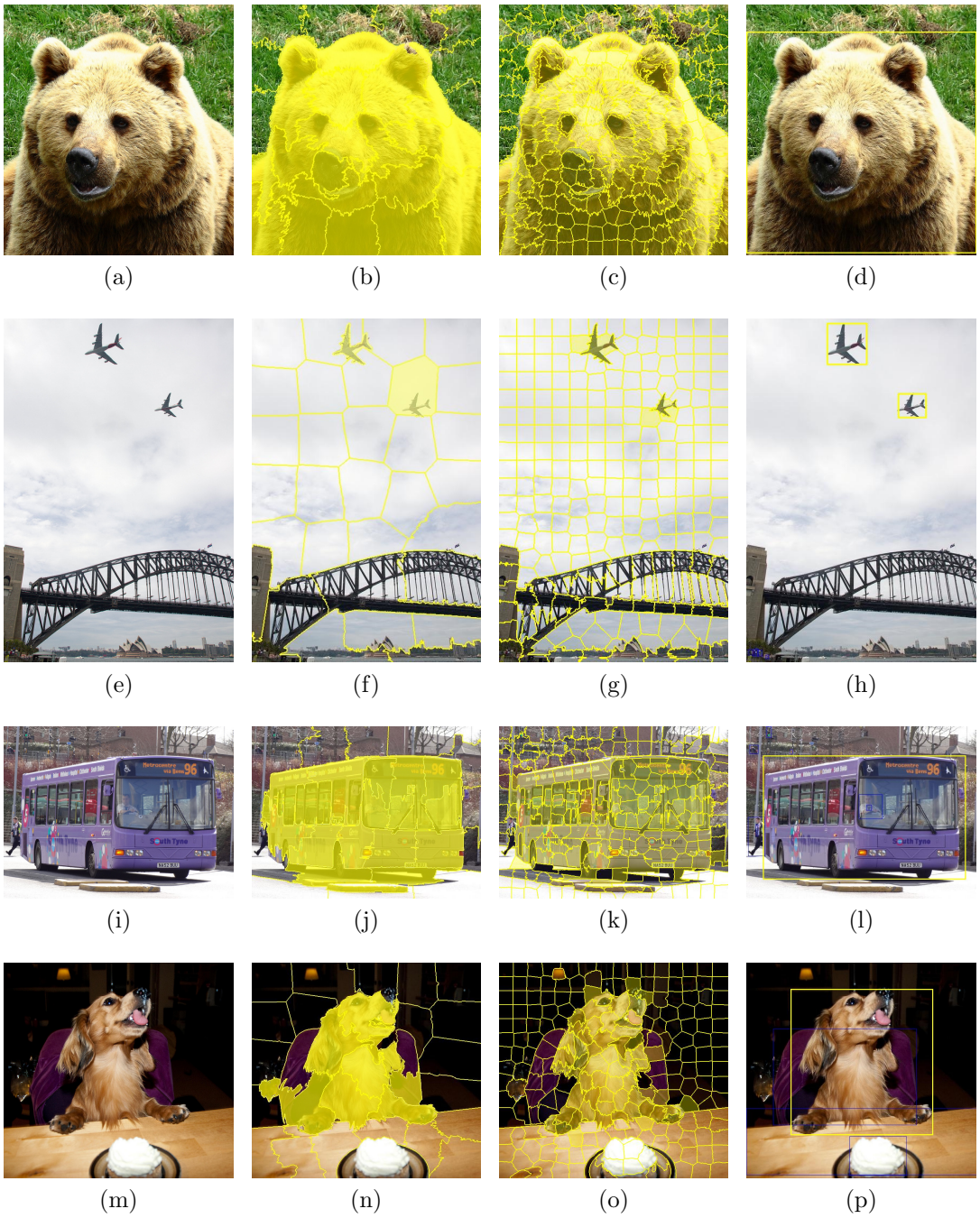


Figure 6.11: Supixel-wise contribution to the entity feature vector visualized: (a), (e), (i), and (m) are images from the VOC 2012 data set, followed by the 128-superpixel contribution maps in (b), (f), (j), and (n), then the 64-superpixel contribution maps in (c), (g), (k), and (o); finally, for reference, the ground-truth bounding boxes are shown in (d), (h), (l) and (p).

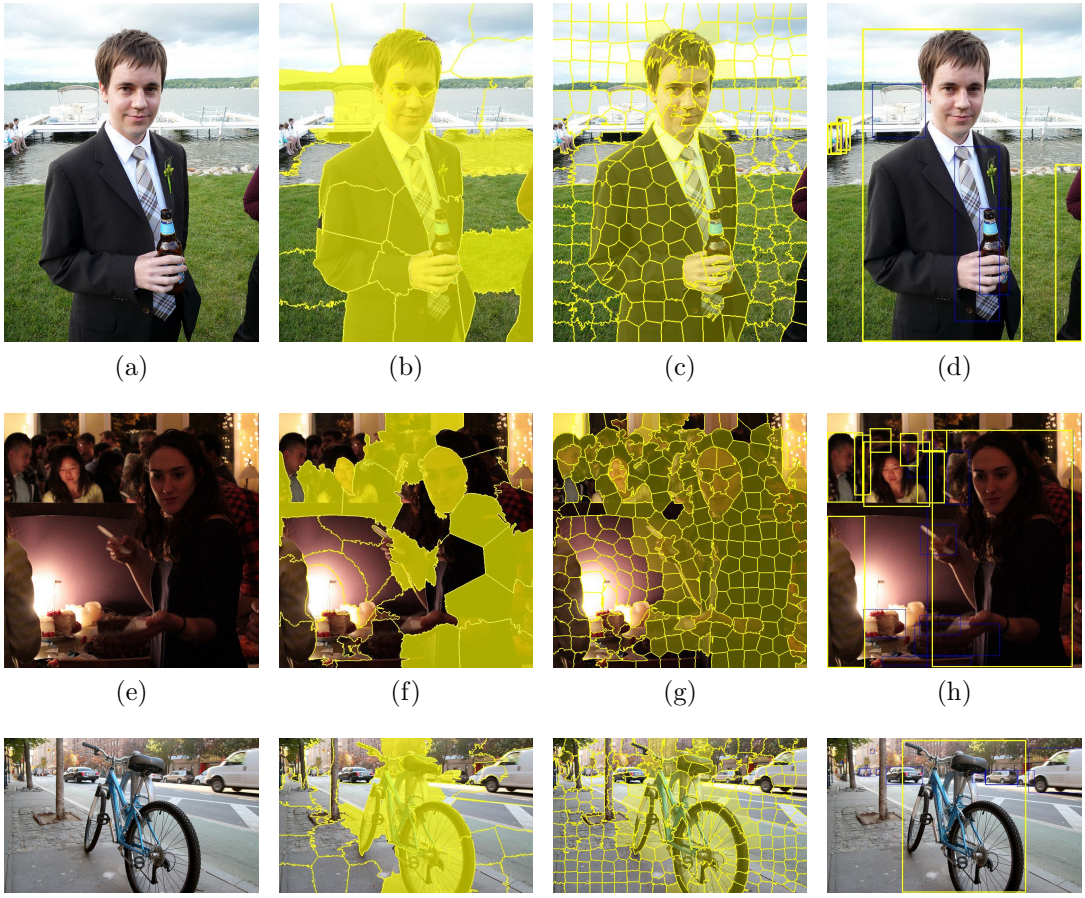


Figure 6.12: Supixel-wise contribution to the entity feature vector visualized: (a), (e), and (i) are images from the VOC 2012 data set, followed by the 128-superpixel contribution maps in (b), (f), and (j), then the 64-superpixel contribution maps in (c), (g), and (k); finally axis aligned bounding boxes can derived, as shown in (d), (h), and (l).

Chapter 7

CONCLUSION

In this dissertation, a novel superpixel-enabled deep neural network paradigm is proposed and the concept of heterogeneous superpixel is created to leverage the results state-of-the-art DCNNs. This new paradigm has enabled superpixel-level image analysis, that breaks the convention of its usage that were limited to dimensionality reduction in computer vision. We have developed the theoretical foundation for Superpixel Feature Extraction (SFE) in the context of deep learning, and discovered that to spatially correlate both low-level visual image features and high-level DCNN features is an effective technique to achieve high-precision, especially in a weakly-supervised task setting. Such as usage of SFE requires expanding the concept of superpixel to groups of pixels that are no longer perceptually homogeneous, i.e. heterogeneous superpixel. GPU-accelerated SFE techniques are developed to enable end-to-end fully differentiable training and real-time inference. The heterogeneous superpixel is a versatile technique that allows WSOD models to be constructed upon existing state-of-the-art DCNNs, and provides object proposals for MIL, and generates insightful visualizations for better model explainability and guidance for further improvements. It has inspired novel heterogeneous superpixel DNN architectures (DHSN) that have demonstrated high accuracy, precision and performance.

Specifically, results have been reported in this dissertation with applications in single monocular depth estimation, remote sensing, and precise WSOD. Despite the

apparent complexity of the architecture, a GPU implementation of the SFE is feasible in real-time inference in an advanced CRF model, as demonstrated in the single monocular indoor relative depth estimation example. DHSN's result goes beyond abstract rectangular bounding-boxes of an object occurrence, but instead approaches efficient parts-based segmented recognition. It has been tested on commercial remote sensing satellite imagery and achieved success. In some of the early attempts, superpixel-based DNN models are created that joins capsule networks to achieve multiple-instance learning and generate precise object boundaries without pixel-level training labels. In further efforts, DHSN is combined with relaxed min-cut technique from graph convolutional networks (GCN) as well as multiple-instance learning (MIL), and it has also demonstrated state-of-the-art performance in solving precise Weakly-Supervised Object Detection (WSOD) problems, according to the evaluation on MS-COCO and VOC 2012 datasets. The DHSN architecture promotes explainable models, whereby the human users of the models can see the parts of the objects that have led to recognition by visualizing the superpixel-wise contribution.

In the future, it will also be interesting to evaluate DHSN in other domains of research, such as those that involve medical imagery, infrared imagery or hyperspectral imagery. This paradigm will also be beneficial to future research where precise object segmentation or boundaries are needed without pixel-wise fully annotated training data. Last but not least, this can also help satisfy the demand of visually explainable models in many applications.

Appendix A

SUMMARY OF SYMBOLS AND NOTATIONS

Table A.1: Summary of Symbols and Notation

dim	the dimensionality of a topological space
dom	the domain of a function
$\varnothing$	diameter
px	pixel count along a horizontal or vertical line; or a unit of length
px^2	pixel count in a region; or a unit of area
cod	the codomain of a function, as per the function definition
ran	the range of a function, which contains actual values the function can generate
$x\|_s$	image function x restricted to subset $s \subseteq \text{dom } x$
h_x	the histogram of image function x, $h_x : \text{cod } x \to \mathbb{Z}^+$
$H(X)$	information entropy of a random variable X
$H(x)$	information entropy of an image, based on the image function x
$H(x\|s)$	superpixel local entropy, see Sect. 4.3
E	the expectation operator
$h_{x\|s}$	the histogram of image function $x\|_s$; s can be viewed as a mask
$\mathbb{R}^+$	The set of positive real numbers with also zero.
$\mathbb{I}_K$	The $K \times K$ identity matrix.
$\mathbf{1}_K$	The column vector in $\mathbb{R}^K$ that is filled with ones.
$\|\cdot\|$	The cardinality of a set, also known as the number of elements for finite sets.
$\|\| \cdot \|\|$	The magnitude or the L^2-norm of a vector; or the Frobenius norm on matrices.

BIBLIOGRAPHY

[1] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012, 2012. *URL http://www. image-net. org/challenges/LSVRC*, 2012.

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

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In: *ECCV*. Springer. 2014, 740–755.

[4] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *NIPS*. 2015, 91–99.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: unified, real-time object detection (2015). *arXiv preprint arXiv:1506.02640*, 2015.

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In: *CVPR*. 2015, 3431–3440.

[7] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, 1451–1460.

[8] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In: *ICCV*. 2015, 1520–1528.

[9] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: *CVPR*. 2017, 1925–1934.

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

[11] F. Liu, C. Shen, G. Lin, and I. Reid. Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *TPAMI*, 38(10), Oct. 2016.

[12] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris. Deep learning advances in computer vision with 3D data: A survey. *ACM Computing Surveys (CSUR)*, 50(2):20, 2017.

[13] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. In: *CVPR*. 2019, 9308–9316.

[14] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In: *NIPS*. 2017, 3856–3866.

[15] X. Zhou, D. Wang, and P. Krähenbühl. Objects as Points. *arXiv preprint arXiv:1904.07850*, 2019.

[16] X. Ren and J. Malik. Learning a classification model for segmentation. In: *null*. IEEE. 2003, 10.

[17] L. Wu, J. He, M. Jian, J. Zhang, and Y. Zou. Fast cloud image segmentation with superpixel analysis based convolutional networks. In: *Systems, Signals and Image Processing (IWSSIP), 2017 International Conference on*. IEEE. 2017, 1–5.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In: *ECCV*. Springer. 2016, 21–37.

[19] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In: *Int. Conf. Learn. Representations*. 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 2818–2826.

[24] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. CoRR abs/1608.06993 (2016). *arXiv preprint arXiv:1608.06993*, 2016.

[25] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. CoRR abs/1707.07012 (2017). *arXiv preprint arXiv:1707.07012*, 2017.

[26] A. Krizhevsky and G. Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[27] G. Strang. A proposal for Toeplitz matrix calculations. *Studies in Applied Mathematics*, 74(2):171–176, 1986.

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

[29] *Convolution arithmetic tutorial*. URL: http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html.

[30] *The transpose of conv2d*. URL: https://www.tensorflow.org/api_docs/python/tf/nn/conv2d_transpose.

[31] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In: *In Computer Vision and Pattern Recognition (CVPR)*. 2010, 2528–2535.

[32] E. W. Weisstein. *Deconvolution*. URL: http://mathworld.wolfram.com/Deconvolution.html.

[33] M. Mathieu, M. Henaff, and Y. LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

[34] W. H. Press, S. A. Teukolsky, B. P. Flannery, and W. T. Vetterling. *Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing*. Cambridge university press, 1992.

[35] V. Podlozhnyuk. FFT-based 2D convolution. *NVIDIA white paper*, 32, 2007.

[36] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In: *Advances in Neural Information Processing Systems*. 2015, 1990–1998.

[37] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In: *European Conference on Computer Vision*. Springer. 2016, 75–91.

[38] H. Hu, S. Lan, Y. Jiang, Z. Cao, and F. Sha. Fastmask: Segment multi-scale object candidates in one shot. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 991–999.

[39] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, and P. H. Torr. Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35(1):37–52, 2018.

[40] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *TPAMI*, 34(11):2274–2282, Nov. 2012.

[41] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

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

[43] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In: *European conference on computer vision*. Springer. 2008, 705–718.

[44] C. Y. Ren, V. A. Prisacariu, and I. D. Reid. gSLICr: SLIC superpixels at over 250Hz. *arXiv preprint arXiv:1509.04232*, 2015.

[45] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In: *European conference on Computer vision*. Springer. 2010, 211–224.

[46] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In: *CVPR 2011*. IEEE. 2011, 2097–2104.

[47] A. Yang, J. A. Hurt, C. T. Veal, and G. J. Scott. Remote Sensing Object Localization with Deep Heterogeneous Superpixel Features. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, 5453–5461.

[48] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz. Superpixel sampling networks. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, 352–368.

[49] X. Ren and J. Malik. Learning a classification model for segmentation. In: *null*. IEEE. 2003, 10.

[50] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[51] A. P. Moore, S. J. Prince, and J. Warrell. "Lattice Cut"-Constructing superpixels using layer constraints. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, 2117–2124.

[52] M. Schuurmans, M. Berman, and M. B. Blaschko. Efficient semantic image segmentation with superpixel pooling. *arXiv preprint arXiv:1806.02705*, 2018.

[53] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In: *CVPR*. 2015, 3376–3385.

[54] D. Stutz. Superpixel segmentation: an evaluation. In: *German conference on pattern recognition*. Springer. 2015, 555–562.

[55] M. Sonka, V. Hlavac, and R. Boyle. Image processing, analysis, and machine vision. In: third. pp. 545-553. CL Engineering, 2008. Chap. 11.

[56] F. Liu, C. Shen, G. Lin, and I. Reid. Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10), Oct. 2016.

[57] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *18th Int. Conf. Mach. Learn.* 2001, 282–289.

[58] A. Rosebrock. *Segmentation: A SLIC Superpixel Tutorial using Python*. July 2014. URL: http://www.pyimagesearch.com/2014/07/28/a-slic-superpixel-tutorial-using-python/.

[59] A. Vedaldi. *Simple Linear Iterative Clustering (SLIC)*. 2010. URL: http://www.vlfeat.org/api/slic.html.

[60] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2010, 1253–1260.

[61] S. Z. Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009.

[62] W. Luo, Y. Li, R. Urtasun, and R. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In: *Advances in neural information processing systems*. 2016, 4898–4906.

[63] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. 2017, 2980–2988.

[64] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[65] G. Chaladze and L. kalatozishvili. Linnaeus 5 Dataset for Machine Learning, 2017.

[66] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – Mining Discriminative Components with Random Forests. In: *European Conference on Computer Vision*. 2014.

[67] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017. arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[68] P. Tang, X. Wang, S. Bai, W. Shen, X. Bai, W. Liu, and A. Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):176–191, 2018.

[69] J. Wu, Y. Zhao, J.-Y. Zhu, S. Luo, and Z. Tu. Milcut: A sweeping line multiple instance learning paradigm for interactive image segmentation. In: *CVPR*. 2014, 256–263.

[70] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.

[71] Z. Shi, T. M. Hospedales, and T. Xiang. Bayesian joint modelling for object localisation in weakly labelled images. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):1959–1972, 2015.

[72] R. G. Cinbis, J. Verbeek, and C. Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):189–203, 2016.

[73] X. Wang, Z. Zhu, C. Yao, and X. Bai. Relaxed multiple-instance SVM with application to object discovery. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, 1224–1232.

[74] M. Shi, H. Caesar, and V. Ferrari. Weakly supervised object localization using things and stuff transfer. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, 3381–3390.

[75] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. *arXiv preprint arXiv:1403.1024*, 2014.

[76] C. Wang, W. Ren, K. Huang, and T. Tan. Weakly supervised object localization with latent category learning. In: *European Conference on Computer Vision*. Springer. 2014, 431–445.

[77] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with convex clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, 1081–1089.

[78] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In: *ACM transactions on graphics (TOG)*. Vol. 23. 3. ACM. 2004, 309–314.

[79] D. Acuna, A. Kar, and S. Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In: *CVPR*. 2019, 11075–11083.

[80] P. Tang, X. Wang, Z. Huang, X. Bai, and W. Liu. Deep patch learning for weakly supervised object classification and discovery. *Pattern Recognition*, 71:446–459, 2017.

[81] V. Kantorov, M. Oquab, M. Cho, and I. Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In: *European Conference on Computer Vision*. Springer. 2016, 350–365.

[82] R. Girshick. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. 2015, 1440–1448.

[83] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[84] A. Yang, C. T. Veal, D. T. Anderson, and G. J. Scott. Recognizing Image Objects by Relational Analysis Using Heterogeneous Superpixels and Deep Convolutional Features. *arXiv preprint arXiv:1908.00669*, 2019.

[85] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: *CVPR*. 2018, 859–868.

[86] Y. Hu, A. Soltoggio, R. Lock, and S. Carter. A fully convolutional two-stream fusion network for interactive image segmentation. *Neural Networks*, 109:31–42, 2019.

[87] G. E. Hinton, S. Sabour, and N. Frosst. Matrix capsules with EM routing. *6th International Conference on Learning Representations, ICLR*, 2018.

[88] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, A. Kumar, R. Feris, R. Giryes, and A. Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In: *Advances in Neural Information Processing Systems*. 2018, 2845–2855.

[89] F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral Clustering with Graph Neural Networks for Graph Pooling. In: *Proceedings of the 37th international conference on Machine learning*. ACM. 2020, 2729–2738.

[90] R. Uziel, M. Ronen, and O. Freifeld. Bayesian Adaptive Superpixel Segmentation. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 8470–8479.

[91] T. N. Kipf and M. Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: `1609.02907 [cs.LG]`.

[92] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[93] M. Defferrard, X. Bresson, and P. Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. 2017. arXiv: 1606.09375 [cs.LG].

[94] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. *Hierarchical Graph Representation Learning with Differentiable Pooling*. 2019. arXiv: 1806.08804 [cs.LG].

[95] B. Fritzke. A growing neural gas network learns topologies. In: *Advances in neural information processing systems*. 1995, 625–632.

[96] X. Xie, G. Xie, X. Xu, L. Cui, and J. Ren. Automatic image segmentation with superpixels and image-level labels. *IEEE Access*, 7:10999–11009, 2019.

[97] Y. Gao, B. Liu, N. Guo, X. Ye, F. Wan, H. You, and D. Fan. C-MIDN: Coupled Multiple Instance Detection Network With Segmentation Guidance for Weakly Supervised Object Detection. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 9834–9843.

[98] A. Yang and G. J. Scott. Efficient Passive Sensing Monocular Relative Depth Estimation. In: *2019 IEEE Applied Imagery Pattern Recognition (AIPR)*. IEEE. 2019.

[99] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE. 2016, 239–248.

[100] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In: *Eur. Conf. Comput. Vis.* 2012, 746–760.

[101] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, May 2009.

[102] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.*, 32:1231–1237, 2013.

[103] *Depth from Stereo Introduction*. URL: `https://www.stereolabs.com/documentation/overview/depth-sensing/introduction.html`.

[104] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In: *Advances in neural information processing systems*. 2014, 2366–2374.

[105] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman. Learning the Depths of Moving People by Watching Frozen People. *arXiv preprint arXiv:1904.11111*, 2019.

[106] A. Van Etten, D. Lindenbaum, and T. M. Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.

[107] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord. xView: Objects in Context in Overhead Imagery. *arXiv preprint arXiv:1802.07856*, 2018.

[108] A. Van Etten. Satellite Imagery Multiscale Rapid Detection with Windowed Networks. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, 735–743.

[109] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger, 1097–1105.

[110] K. Diederik and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[111] A. Van Etten. You only look twice: Rapid multi-scale object detection in satellite imagery. *arXiv preprint arXiv:1805.09512*, 2018.

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

[113] K. Ye, M. Zhang, A. Kovashka, W. Li, D. Qin, and J. Berent. Cap2det: Learning to amplify weak caption supervision for object detection. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, 9686–9695.

[114] Z. Ren, Z. Yu, X. Yang, M.-Y. Liu, Y. J. Lee, A. G. Schwing, and J. Kautz. Instance-aware, context-focused, and memory-efficient weakly supervised object detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, 10598–10607.

[115] D. E. King. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[116] Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu. Deep self-taught learning for weakly supervised object localization. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 1377–1385.

[117] W. Ge, S. Yang, and Y. Yu. Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 1277–1286.

[118] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[119] K. Chen, J. Wang, L.-C. Chen, H. Gao, W. Xu, and R. Nevatia. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*, 2015.

[120] H. Zheng, J. Fu, T. Mei, and J. Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In: *Proceedings of the IEEE international conference on computer vision*. 2017, 5209–5217.

VITA

Zhangwei (Alex) Yang

Alex Yang received a Bachelor of Engineering degree (2012–2016) in Computer Science East China University of Missouri, Shanghai, China. He then earned the Master of Science degree (2016–2017) and the Ph.D. degree (2018–2021) in Computer Science from the University of Missouri at Columbia. His research interests include superpixel-enabled deep neural networks, their application in computer vision tasks, and GPU accelerated scientific computing. During his Ph.D., he worked on various superpixel deep neural architectures and developed a novel approach towards superpixel-level image analysis to solve challenging computer vision problems, such as weakly supervised object detection. Due to a strong interest in general-purpose GPU computing, he also interned at Nvidia Corporation with the Deep Learning Library team as a software engineer. After graduating with the Ph.D. degree, he continued working at Nvidia Corporation, developing world-leading software that powers deep neural networks with unprecedented performance on the latest GPU architectures.