APPROVAL SHEET

Title of Dissertation: VISUAL COMPUTATIONAL CONTEXT:
USING COMPOSITIONS AND NON TARGET
PIXELS FOR NOVEL CLASS DISCOVERY

Name of Candidate: Jeffrey Thomas Turner
Doctor of Philosophy, 2019

Dissertation and Abstract Approved: 

Tim Oates
Professor
Computer Science and Electrical Engineering

Date Approved: 6/14/2019

JT Turner Ph.D.

Washington, DC
Curriculum vitae

Research Interests

Deep Learning
Computer Vision
Machine Learning
Sabermetrics

Education

2017-2019 **Ph.D. in Computer Science**, *University of Maryland Baltimore County.* 3.47/4.
Advisor: Dr. Tim Oates

2013-2014 **M.S. in Computer Science**, *University of Maryland Baltimore County.* 3.47/4.
Advisor: Dr. Tim Oates
+Thesis track

2009-2013 **B.S. in Computer Science**, *University of Maryland Baltimore County.* GPA: 3.41/4. Major GPA: 3.84
+Dual Major in mathematics

Research Experience

2018-Pres **Clarifai**, *Tysons Corner, VA.*

2014-2018 **Knexus Research Corporation**, *National Harbor, MD.*
+Context based object detection in images.
+Neural network enhancements for part detection.
+Algorithms research for faster region proposal.

2014 **Autonomy Engine, LLC**, *Marriottsville, MD.*
+Energy Model Classification of voice tones.

2014 **Naval Research Laboratory**, *Washington, DC.*
+Neural architecture modifications for actions.

2013-2014 **CoRaL Lab**, *Baltimore, MD.*
+Time series analysis of EEG signals for seizure Classification using deep learning

2012 **National Institute of Standards and Technology**, *Gaithersburg, MD.*
+Video Interpolation of facial recognition and Integration with FFMPEG filters.

Teaching Experience

2013 **Undergraduate Teaching Assistant**, *CSEE Department, UMBC.*
+Led 2 lab sections of CMSC 201.

2012 **Undergraduate Lab Assistant**, *CSEE Department, UMBC.*
+ Assisted in lab for CMSC 201 and 202.

2011-2012 **Undergraduate Tutor**, *CSEE Department, UMBC.*
+Tutored CMSC 104 - 314.

2010-2011 **Undergraduate Grader**, *CSEE Department, UMBC.*
+Graded students projects for CMSC 201 and 202.

Professional Experience

2018 - Present
Clarifai
Senior Research Scientist
Tysons Corner, VA.

2014-2018

Knexus Research Corporation

Research Scientist

National Harbor, MD.

+Writing proposals and consulting on research.

+Development projects as a software engineer.

+Google administrator for company.

2012

UMBC Computer Science/Electrical Engineering Department,

Unix System Administrator

Baltimore, MD.

Daily maintenance tasks and updates to systems.

Development of visual SVN manager for faculty.

Publications

2018 **"Novel Object Discovery using Case-Based Reasoning and Convolutional Neural Networks"**, *accepted at ICCBR-2018, 1st author.*

2017 **"Using Deep Learning to Automate Feature Modeling in Learning by Observation"**, *accepted at FLAIRS-30*, 2nd author.

2017 **"Using Deep Learning to Automate Feature Modeling in Learning by Observation: A preliminary study"**, *accepted at AAAI-SS 2017*, 2nd author.

2016 **"SPARCNN: SPAtially Related Convolutional Neural Networks"**, *accepted at AIPR 2017, 1st author.*

2016 **"Keypoint Density Region Proposal for fine grained Object detection using regions with convolutional Neural network features"**, *accepted at AIPR 2017, 1st Author.*

2015 **"Convolutional Architecture Exploration for Action**

Recognition and Image Classification”, technical note NCARAI, 1st author.

2014 **“Comparing Raw Data and Feature Extraction for Seizure Detection with Deep Learning Methods”**, accepted at FLAIRS-27, 2nd author.

2014 **“Deep belief networks used on high resolution multichannel electroencephalography data for seizure detection”**, accepted at AAAI-SS 2014, 1st author.

2013 **“TIME SERIES ANALYSIS USING DEEP FEED FORWARD NEURAL NETWORKS”**, accepted masters thesis.

Languages

English	Native
Spanish	Moderate
American Sign Language	Moderate

Computer Skills

Advanced:	Linux OS, Python, Java, Caffe
Proficient:	C, numpy, scipy, Theano
Basic:	C++, Perl, Bash, Tensorflow

ABSTRACT

Title of Proposal: VISUAL COMPUTATIONAL CONTEXT: USING COMPOSITIONS AND NON-TARGET PIXELS FOR NOVEL CLASS DISCOVERY

JT Turner, Doctoral Candidate, 2019

Proposal guided by: Professor Tim Oates
Department of Computer Science

During the deep learning revolution in computer science that has occoured since 2006, two factors have pushed our ability to successfully learn from large-scale data sources: exponential growth in computational power and the size and degree of annotation of our datasets. Modern models loaded in the Graphics Processing Unit (GPU) can fill an entire 12 GB Video Random Access Memory (VRAM) graphics card cache; a training task achievable in weeks that would have taken centuries on CPUs from 10 years ago [1]. The standard computer vision dataset at the time - Mixed National Institute of Standards and Technology (MNIST) - consisted of $70,000$ 28×28 pixel grayscale images of 10 class labels. The more recent ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset contains over 15 million full color images with $1,000$ different class labels. During this time however, there has been little growth in contextual use in images. Context can be used to identify target objects that may be obfuscated in the input space as well as confirm or deny the existence of objects based on underlying parts. I use context in two main

ways to improve object detection and scene understanding. First, I use location and correlation between objects to infer difficult to see and obfuscated objects [2]. In my second study, I further support the necessity of non-target pixels by using background pixels of the image to aid in classification instead of only other objects in the scene. In addition I use case-based reasoning to detect novel objects that were not seen during training, and classify them with other visually similar objects based on their observable parts. I use this case-based reasoning model in conjuction with a CNN to demonstrate the ability to overcome shortcomings of a traditional deep learned network with case-based reasoning.

VISUAL COMPUTATIONAL CONTEXT: USING COMPOSITIONS AND NON-TARGET PIXELS FOR NOVEL CLASS DISCOVERY

by

JT Turner

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County to complete fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Tim Oates, Chair/Advisor
Dr. Michael Floyd
Professor Anupam Joshi
Professor Tinoosh Mohsenin
Dr. Kalyan Moy Gupta

Preface

Note from the author: This section has been unedited from it's original form...the unfiltered JT if you will. This is what I write like, and my voice. An extra thanks to my thesis committee for reading 150 pages of this.

The first time I heard about **DEEP LEARNING** was in the Spring of 2012 in Dr. Oates' machine learning class. There were some circles and some arrows that pointed to other circles.

When I started graduate school in May of 2013 I was tasked on a project for seizure detection, and I started in this crazy field using *theano*, and an *NVIDIA Quadro K2000M* GPU. The first time on this project when I checked the wikipedia page for Deep Learning, it was a stub. As the summer went on, and the graduate school semester started, I had become obsessed with the raw power of these algorithms; that I was able to start training, play Starcraft for 2 days, and when the model had finished training, it would be able to seemingly be able to transform raw signals into classifications of greater accuracy than SVMs. Normal classes had taken a back seat to my desire to use deep neural networks. Around this time, my current boss Matt Zeiler won the top five positions in Imagenet (Image classification superbowl), and my co-worker David Eigen won the Imagenet localization challenge. Deep learning was racheting up big, and probably wasn't a wikipedia stub by this point.

Images became the new focus in 2014 when I started student contracting at the Naval Research Lab (the connection which led me to meet the two externals on

the committee, my old boss and a large motivation in this work Kalyan Gupta, and my old coworker, friend, and Canadian Michael Floyd), and I became the image processing guy at Knexus Research Corporation. When I left Knexus in 2018 to work at Clarifai, everybody knew about Deep Learning. There were entire classes at Universities dedicated to these backpropogating brain benders. While in my masters thesis I thought that the 3 layer deep stacked restricted boltzmann machines that I used were deep, these days if you want to do convolution on edge devices you'll probably only use a 50 layer deep resnet, instead of the full 200 layers.

It's been a hell of a ride since I graduated with my masters. I got engaged. We got a cat. Then we got a dog. The Caps won the stanley cup, and the redskins were awful. I ate only ice cream for 27 days. In 6 more years when my dissertation is a mid point in my deep learning studies I can't imagine what shape the field will be in. Maybe neural networks will fall out of fashion like they did in the 90's for something new; quantum machine learning, or kitten powered support vector machines. Whatever the future brings, I'm excited to be on the bleeding edge of research in computer science.

Dedication

Dedicated to a happy rest of my life with Heather Wolf.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank Knexus Research Corp, in particular Dr. Kalyan Gupta, and Dr. Michael Floyd. As the first job I had after graduate school, Knexus exceeded my wildest dreams of what I thought the *real world* would be like. A big thanks to Kalyan for countless hours of kicking around ideas creating this thesis, and to Floyd for countless hours reading quasi-English drafts of papers.

I would also like to thank Dr. Tim Oates for dealing with me on (going on 10 years) since he first had his class rudely interupted by class clown comments in *Introduction to Compilers*. Tim has been instrumental in my progress in undergraduate, masters, and now doctorate studies. I would be nowhere near where I am now without Tim's help. I think I can bench more than him.

My family has always been supportive of my efforts in school, and out of school (even when efforts are not worthy of support). I've never felt short on love, support, or someone to argue whether or not a carb/meat combination is a sandwich. This includes my amazing new family of in-laws who ride motorcycles, paddle rivers, practice law, found companies, and more!

My friends are p**e**ople who surely deserve some **g**ratitude. **A**ll of my life, I have relie**d o**n the kindness a**n**d company of others to get me through life. You guys are **g**reat.

I said this when I got my masters because I figured there would be no way

that I would be getting *more* degrees from the school, but for the last time: **SO LONG, AND THANKS FOR ALL THE FISH!!**

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

> A computer would deserve to be called intelligent if it could deceive a human into believing it was human.
>
> –Alan Turing

Larger visual datasets are imperative to increasing performance in object detection and classification. The growth of computational power through Graphics Processing Units (GPUs) turns the duration of learning feature representations from an input of pixel matrices from lifetimes to weeks. There is an additional way to add information to a classification algorithm that does not replace past improvements, and does not claim superiority over these algorithms, but instead can be used modularly to supplement learning, and produce gains in representational knowledge. By using the same powerfully trained models and features, visual context can be used to detect more objects, reject background artifacts, and improve our representational knowledge of the scene.

1.1 State of The Art in Computer Vision

Note: This was published in June 2019. This will become obsolete quickly. By 2022 this should sound antiquated, and foreign a few years later.

1.1.1 Hardware and Models

Since 2012 in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [7], Convolutional Neural Networks (CNNs), a Deep Learning approach, have emerged as a promising technique that dramatically outperforms conventional approaches on classification accuracy. Evolving from the early work of Lecun [8], who primarily focused on image classification, CNNs can now achieve state-of-the-art performance on object detection tasks [9]. Although highly accurate they are still not suitable for real-time detection and classification applications. For instance, with Regional Convolutional Neural Networks (R-CNN) and Fast R-CNN [10] the object detection pipeline takes 2 seconds. With the usage of sophisticated region proposal techniques [11] [12] object detection can be done in fractions of a second with no statistically significant loss to detection accuracy. With the introduction of regional proposal networks into the CNN (Faster R-CNN [13], and masking R-CNN networks [14]), detection can be done almost realtime, and pixel level segmentation of objects can be achieved.

In addition to more computationally taxing and memory intensive CNNs being used for gains, the deep algorithms themselves have made large strides. Common techniques such as dropout [15] to prevent overfitting, batch normalization (BN) [16]

to improve gerneralization, or residual learning [17] can be attached modularly to existing networks to improve performance on given tasks. We claim that adding contextual information about parts and objects in an image will acheive similar gains in performance.

1.1.2 Data

Datasets have also increased in size since the boom of Deep Learning. Some of the datasets such as the Scene Understanding (SUN) dataset [18] or the ImageNet classification dataset [7] have over 1,000 object categories, while datasets such as the Microsoft Common Objects in Context (MS COCO) [19] and PASCAL Visual Object Challenge (VOC) [20] have over 1,000 instances for every target category. The 3 leveled hierarchy of the SUN dataset provides extra knowledge about the image that we are seeing, similarly to how the CUB-200 [21] or the subset of VOC Pascal Parts Dataset [22] give us information about the location and presence of parts inside the bounding boxes for a better knowledge representation of the objects comprising the greater scene.

1.2 Context and Novel Object Discovery.

Images are represented in computers as a 3-dimensional tensor or matrix of pixels representing the images height, width, and channel depth. Pixel matrix images contain a vast amount of raw data which is transformed into features, objects, scene descriptions, etc by CNNs. By learning on the features provided to us from

deep CNNs, we are able to learn features of the features themselves, and identify and recognize objects that would not have been detected had we only used the raw signal input. This type of learned features of features schema will move us closer to a human type vision system, where we are able to reason about and compare what we think we see with what we know we see. The ability to know that an obscured object on a table is much more likely a bottle than a flute because of the surrounding objects and features is a valuable type of knowledge representation to have. Furthermore, the usage of parts in objects will allow us to not only raise or lower our confidence in visual object detections, but it will also allow us to infer new types of objects from the parts that we see, or be able to better estimate the actions of objects detected. Imagine a system that has been trained on a variety of brass instruments: a trumpet, a tuba, and a sousaphone. Upon encountering an image containing a trombone, the system would be able to deduce from the lack of finger valves present in the other brass instruments that this is not one of the three it has seen previously. It would know however, that this instrument's coloration, long tubes, mouthpiece, and flared horn is much more similar to a brass instrument than it is a type of animal, so ideally it would be able to label the trombone as a new trumpet-tuba relative, instead of incorrectly assigning to an existing class.

Parts can also tell us the action that the object itself is taking. Imagine an image of a man asleep on a bed with a cat next to him licking it's paws clean. The configurations of the the body parts of the man's head, legs, torso, and arms may allow us to label this man as a sleeping man. Furthermore, the cat's configuration could tell us this is a grooming cat. This would be very different from a picture of a cat

sprinting across the room, as they are known to do, and a man standing in front of the stove stirring a stew. The fine-grained classifications here would be a running cat and a standing man.

1.3 Peer Reviewed Contributions

The remainder of this thesis is organized as follows: Chapter 2 is prior work and related work to the topic of contextual computation in images, as well as a brief history of computer vision, as well as a discussion on object relationships with neural networks, case-based reasoning, and few shot learning. Chapter 3 is an extensive review of Deep Neural Networks and their usage in computer vision. When discussing the region proposal and object detection component of CNNs, a large portion of my work *Keypoint density-based region proposal for fine-grained object detection using regions with convolutional neural network features* published at the *2016 Institute of Electrical and Electronic Engineers (IEEE) Applied Imagery Pattern Recognition (AIPR) Workshop* will be referenced. Chapter 4 will discuss the usage of pixels that are *not* the object we are currently trying to classify to assist our classification. In doing this we will reference two of my previous works, *SPARCNN: Spatially related convolutional neural networks* published at the *2016 IEEE AIPR Workshop*, and *Unknown Target Classification: A Contextual Classifier Study*, submitted for publication at *2019 British Machine Vision Conference (BMVC)*. Chapter 5 will introduce case-based Reasoning for Novel Object Classification. This chapter will reference my theoretical paper *Novel Object Discovery using Case-Based Reason-*

ing and Convolutional Neural Networks published at the *International Conference on case-based Reasoning (ICCBR) 2018*, and the practical application of my paper *NOD-CC: A Hybrid CBR-CNN Architecture for Novel Object Discovery*, accepted for publication and an oral talk at *ICCBR 2019*. Chapter 6 will be closing remarks on computational context, and its usage in computer vision, and what will hopefully be future works from myself and other scientists.

The contributions of this thesis to the field of Deep Learning and computer vision are the following:

- A method of region proposal that is demonstrably faster that existing region proposal techniques [23] without a loss in accuracy.
- A method of spatial and size-based object relation in images with multiple objects that were shown to boost recall and overall accuracy in multiple experiments.
- Experimentation on using non-target pixels in an image to infer classes withheld during training time, and provided evidence supporting the necessity or ordering non-target pixels in aiding classification.
- An algorithm created to discover and classify novel objects given feature embedding similarity and part similarity to other objects using case-based reasoning
- The novel object algorithm described above in conjunction with CNNs to guide decisions to create novel classes beyond what was only achievable with

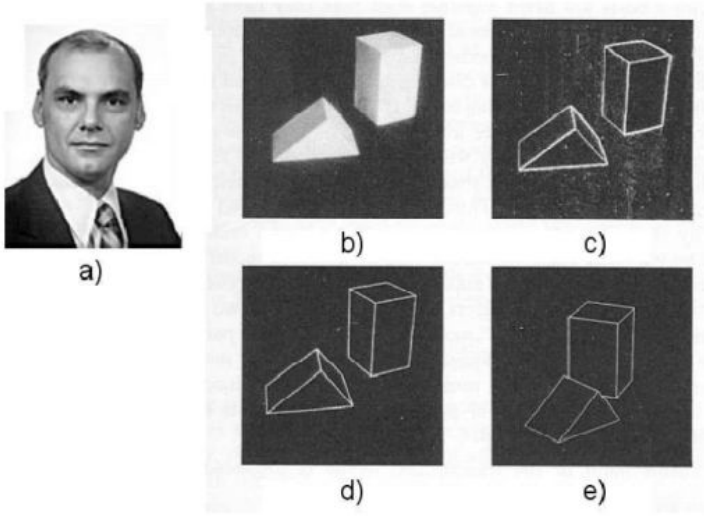
case-based reasoning.

Chapter 2: Literature Review

> Everyone points at a picture of a goat sitting in a grassy field and says *Look at the goat sitting in the grassy field*. Nobody cares about goats in grassy fields, that is not interesting.
>
> –Kalyan Gupta, 2015.

Here in Chapter 2, I provide context for the work done in this dissertation, and where it sits in the wide field of computer vision and case-based Reasoning (a technique used heavily in Chapter 5 to overcome shortcomings of CNN only techniques). In Section 2.1, I give a general history of the field of computer vision from it's inception in the 1960's to the neural networks of today. In Section 2.2, I discuss Recurrent Neural Networks which are another form of using spatial or temporal context for performance gains. In Section 2.3, I discuss existing work done in the field of Computational Context, and in Section 2.4, I provide an overview of CBR and its usages with respect to our purposes of novel object detection.

Figure 2.1: Figure from [3]. **a)** Lawerence Roberts, **b)** 3-d polyhedrals, **c)** Detected edges from polyhedrals using a gradient based edge detector, **d)** Machine representation of the polyhedrals, **e)** Internal shifted view of the polyhedrals.



2.1 Vision History

In the following sections, I give a little bit of insight on where the field of Computer Vision was, what started the Neural Network revolution, and lastly say where the field is now. I also dispel an infamous Computer Vision urban legend.

2.1.1 Ancient History (Pre 2012)

Computer vision became a topic of interest in the world shortly after computers did, with primitive solutions at first. In 1963, Lawerence Roberts published *Machine Perception of Three Dimensional Solids* [3], in which a system recognized 3 dimensional polyhedrals using 2×2 gradient operators over the image, and then reconstructed these to be viewed from different angles as shown in Figure 2.1.

While the geometric views had the advantage of using powerful edge detectors using the image gradient, the algorithms required a known shape of the object being detected which was not scalable to real world vision problems.

Three years later in 1966, it was said that Marvin Minsky told a summer student to *solve the computer vision problem*. As endearing and fun as this tale is, it was unfortunately not true, as Seymour Papert's original proposal paper [24] shows their task was different. The believed quote from Minksy was telling the students to *do something interesting* with a camera and a computer.

In the following few decades, there were other uses of gradient edge detectors and geometric recognition of objects [25] [26] [27] [28], which all struggled, as well, to recognize object categories, or objects with variation in shape or edge composition. Not to be lost in the 80's was the work of Fukushima developing *Neocognitron* [29], an artificial neural network in the geometric era of vision. Neocognitron used two different types of neurons, *s-cells* or simple cells, and *c-cells* or complex cells to learn letters and numbers on handwritten checks. The *s-cells* were responsible for edge and corner detection of the writing of the characters, while the *c-cells* learned features of the output of the simple features extracted as seen in Figure 2.2.

The 90's brought a proposed solution to the problem of object variability and rigidity of before by using *appearance Based Models*. These models used global image features such as color indexing [30], eigen vector and fisher vector linear projections [31], or appearance manifold clustering [32] to get more robust views of objects to classify them. Although these appearance based models were much more robust than the geometric constructions of decades before, they still struggled

Figure 2.2: Neocognitron Neural architecture of simple *s-cells*, and complex *c-cells*

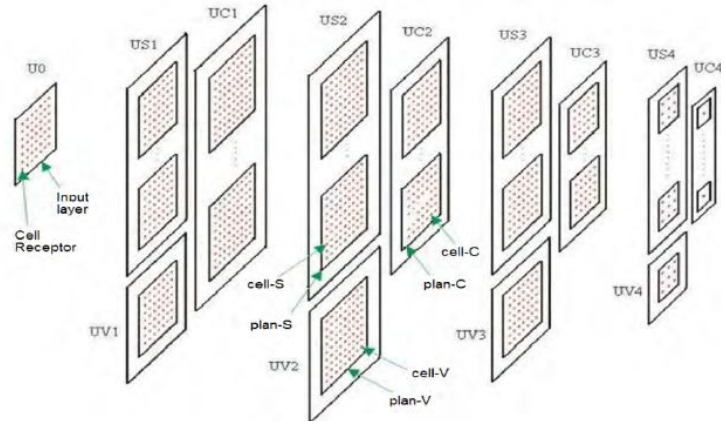
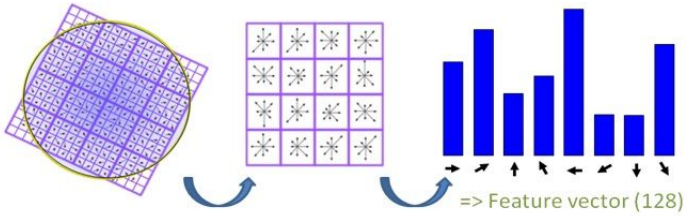


Figure 2.3: A diagram showing the direction of the gradient at many pixels evaluated in the neighborhood of the SIFT keypoint. The magnitude of these vectors were given to a histogram of gradients (HOG) to create a feature descriptor.



with object occlusion, variations, deformations, or situations where a global pattern could not be registered in the image.

At the end of the 90's leading into 2000, local image patterns and keypoint descriptive methods began dominating the computer vision world. In a computer vision paper with over $50,000$ citations at the publication of this work (mid 2019), David Lowe published *Distinctive image features from scale-invariant keypoints* [33], where he created feature vectors using histograms of gradients (HOG) taken at points in the image with large gradient shifts (Figure 2.3).

These features were distinctive, and could be used to match exemplars of a

prototype object to new instances of the object. Moreso, these HOG feature vectors were scale invariant vectors, and could easily be fed to a support vector machine for robust classification. The winning entry to the Imagenet Competition in 2011 used a gaussian mixture model to build a vocabulary learned on low level SIFT image features [7]. 2011 was the last year that the Imagenet challenge has been won by something other than a neural network.

2.1.2 The Neural Revolution (2012 - 2016)

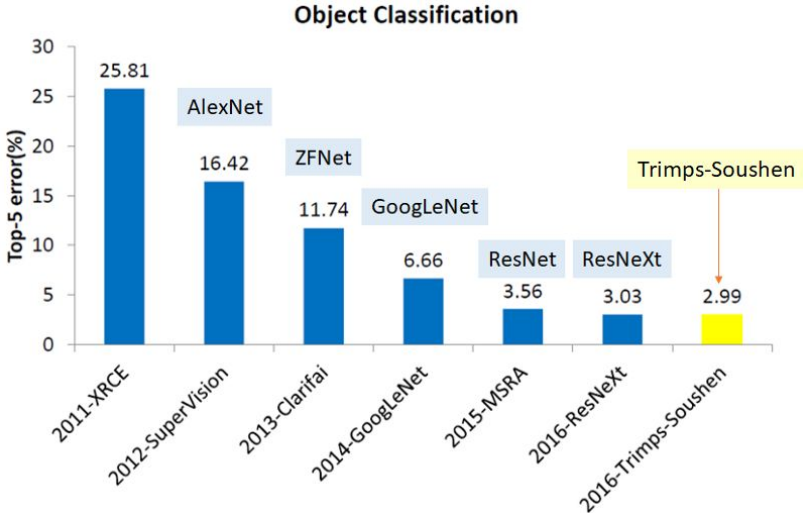
While the recent advances in utilization of SIFT keypoints and feature descriptors led to impressive performances on the Imagenet classification challenges (in 2011 XRCE got a classification error of 25.81%), they were nothing compared to the wave of neural networks about to take the world by storm. The results from 2011 to 2016 (the challenge was fundamentally changed to be more focused on video classification in its final year 2017, so classification results are not given) are shown in Figure 2.4.

We will briefly discuss the architectural significance of each of the networks from 2012 to 2015.

- 2012 **Alexnet-** Alex Krizhevsky's *Alexnet* [34] in 2012 was the first deep neural network, using convolutional filter layers, max pooling, and GPUs (for the first time in the Imagenet challenge) to reduce the classification error by 9.39 percentage points.

- **2013 ZF Net-** Matt Zeiler's 2013 *ZFnet* [35] introduced a deconvolutional

Figure 2.4: Object Classification scores from shallow beginnings in 2011 to residual networks of high cardinality in 2016.



layer to the network in order to better visualize the filters and activations from the network being learned. Using this method, Zeiler was able to slightly modify the hyper parameters of Alexnet to reduce the classification error by 4.68 percentage points.

- **2014 GoogLeNet-** In 2014, Szegedy, GoogLeNet [36] (also known as inception net) won the Imagenet challenge with far less parameters then seen before by the introduction of *inception modules*. By using filters of different sizes, and dimensional changing via 1×1 convolutional layers, inception was able to reduce classification error by 5.08 percentage points.

- **2015 Residual Network (Resnet)-** Up through 2014 the deepest neural networks were VGGNet [6] at 19 layers, and Inception [36] at 22 layers. Resnet [36] was 152 layers deep, and able to alleviate the vanishing gradient problem by using skip connections. Resnet was able to reduce classification error by 3.1

percentage points.

- **2016 Residual NeXtworks-** After Resnet had gone into triple digit layers it became difficult to go deeper, so resneXt [37] went *wider*. By splitting the channels into different groupings (instead of adding more layers), the same computational power is able to be achieved with fewer parameters, so they were able to add more discriminatory power with the same amount of memory. By doing this, they were able to reduce the classification error by .53 percentage points.

In 2017, the contest was more focused on video than image classification. The Imagenet challenge has not been run since, as classification and localisation boosts showed diminishing returns.

2.1.3 Modern Vision (2017 - Present)

Deep Learning changes since the end of imagenet have not been as significant as the changes in the 5 years since *Alexnet*. The algorithms have been applied to more domains and videos, and the data being collected in the public and private sectors is increasing. Since 2015, deep learning algorithms have surpassed human performance on the classification and localization tasks [36], and it's becoming nearly impossible to gain more information from the target object signals. Although there are consistently new algorithmic improvements in detection, classification, and video analysis, it is impossible to get the performance boosts seen in 2012 of 9.39 percentage points (since current error rates are less than 3%).

2.2 Recurrent Neural Networks

The primary method for attention-based visual models and scanning images in the literature is Recurrent Neural Network's (RNNs). The most common RNN used is Long Short-Term Memory (LSTM) [38]. At time step t, the LSTM recieves a hidden cell state c_{t-1}, and an input signal m_{t-1} to produce an output signal h_t. Three gates are used; a forget gate f_t, input gate i_t, and output gate o_t, with parameter matrices W_g and U_g given signal g, and parameter vector b_g given g such that (with σ being the nonsaturating linearity of implementers choice):

$$f_t = \sigma\big(W_f x_t + U_f h_{t-1} + b_f\big)$$

$$i_t = \sigma\big(W_i x_t + U_i h_{t-1} + b_i\big)$$

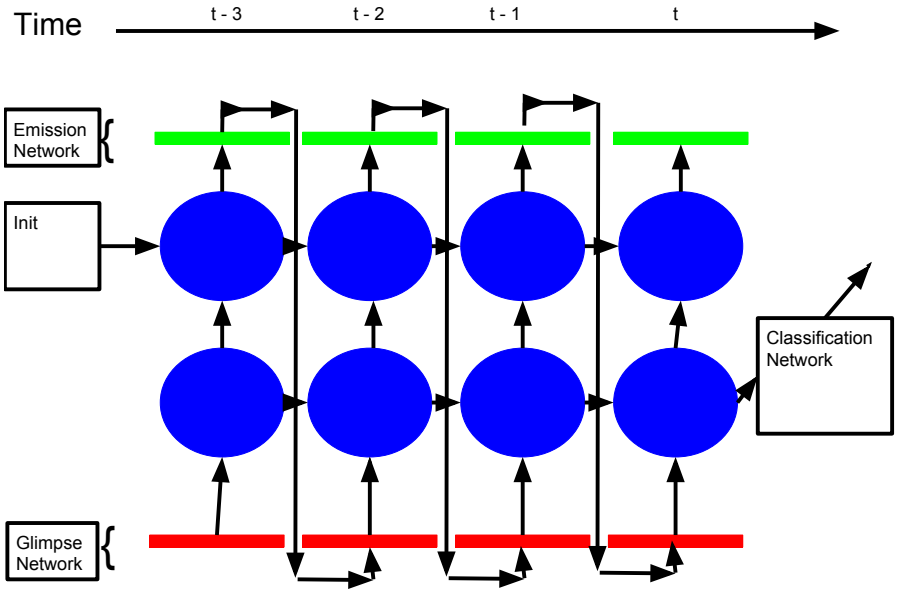
$$o_t = \sigma\big(W_o x_o + U_o h_{t-1} + b_o\big)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma\big(W_c x_t + U_c h_{t-1} + b_c\big)$$

$$h_t = o_t \circ \sigma(c_t)$$

LSTMs have more power to regulate how much and which parts of their previous inputs they remember, and the ability to forget them once a new input renders the old memory obsolete. [38]. In [4], two different LSTMs are used to guide object detection; the *emission network* E, and the *glimpse network* G. The role of the emission network is to provide coordinate information for what region of the image to process next given what it has seen previously, and what it is currently seeing. The purpose of the glimpse network is to use a powerful convolutional network on a region of the image to classify what object (if any) is present.

Figure 2.5: Visual attention method of [4]



Because RNNs need some initialization, the initial feature vector I_{fv} is created by processing a downsampled version of the initial image I. The first emission network E_0 produces output coordinates c_1 by using I_{fv} as input. The following emission networks E_n take as input the output of E_{n-1}, and the output of G_n, where G_n is the n^{th} glimpse network. The n^{th} glimpse network is the feature vector created by convolution of the c_n^{th} region of I. The network is shown in Figure 2.5.

The recurrent nature of the model allows some contextual information to be used in both deciding where to look next, and what to expect next. This technique is helpful in datasets such as Street View House Numbers (SVHN) [5] (Figure 2.6) where a single forward pass is sufficient and object sizes are generally fixed, but would struggle in a large search environment where target objects are not neatly aligned.

Figure 2.6: Sample images from SVHN [5]

2.3 Computational Context

The use of context in images is a field that has not garnered more than a small fraction of the research in the area. Bell et al. [39] use recurrent neural networks on multiple scales to scan the feature maps of the image for inter-object relationships to make further predictions on these relationships. This work leverages IRNNs [40] which are identity matrix initialized Rectified Linear Unit (ReLU) recurrent transitions on the feature mapped output space of a Fast-RCNN [10]. Although these were successful in raising the mAP of complex object detection systems, this system used target pixels from other objects in the scene (such that it would gain no advantage in a single target image), and used the stored feature map output from the network, so is not end-to-end trainable.

A different work using context for gains in detection accuracy is that of Turner

et al. [2], where a VGG16 feature extractor [6] was used in conjunction with a Fast-RCNN detection net to compute object correlation probabilities in training, and to apply these known relations to modify the probability of detection at inference time for increased accuracy. In removing biases from the model added by using training data, the work builds correlative probability modifiers using a bias reducing region proposal technique [41]. Although increasing accuracy and average precision on the VOC2012 dataset, this technique did not leverage Non Target Pixels (NTP)s explicitly, and requires a large annotated dataset to generate the necessary object correlation matrices for detection.

Another interesting approach using recurrent neural networks in the field is that of Minh et al. at Deep Mind [42] on attention based networks. This work uses a separate recurrent neural network called a *Glimpse Network* together with a *Glimpse Sensor* to sample an image patch or *chip* at multiple scales in order to guide the network. Although this does focus on NTPs similar to our work, it is using the recurrent model to shift the attention of the network to new locations to complete part of a larger image, such as discovering the full sequence of numbers on the Street View House Numbers (SVHN) or MNIST dataset.

2.4 Case-based Reasoning

Integrations of Deep Learning and CBR have seen increased interest recently, with many researchers exploring how the two approaches can benefit each other. In the domain of Human Activity Recognition (HAR), CNNs have been used for

feature extraction [43]. This work differs from our work in that it uses accelerometer data rather than image data, but similarly finds that reasonable results can be achieved with instance-based algorithms when features are automatically learned and extracted using CNNs. Instance-based retrieval in the HAR domain has also been used to find similar existing data that can be used to train a classifier for a new user [44]. Their system also uses CNN-extracted features and, like our work, is motivated to allow learning under limited data availability. However, their work is focused on classifier personalization rather that novel class identification (e.g., they do not detect new types of activities that have not been seen before). They have also examined how Siamese Neural Networks can be used to learn similarity functions [44], and such an approach could potentially be used in our algorithm to improve retrieval. Deep Learning has been combined with CBR to generate novel recipes that are both surprising and plausible [45]. However, this differs from our own work in that their system creates novel items rather than discovering previously unknown items.

Case-based reasoning has been used for a variety of image processing and computer vision tasks [46]. One application area that has seen particular interest is medical CBR (e.g., [47] [48] [49]), primarily due to the prevalence of medical imagery in patient files and the need to retrieve similar images to aid in diagnosis. However, unlike our work, the majority of CBR approaches rely on hand-crafted features rather than learned features (e.g., [50] [51] [52]). Additionally, while CBR systems are often used for image retrieval and classification, to the best of our knowledge none are able to detect novel object types (or, more generally, novel classes in non-

image systems). Some systems may be able to perform outlier detection (e.g., when no similar cases are retrieved) but do not attempt to learn novel object types from these outliers. For example, rather than attempt to generate a novel object label, a CBR system may present an input image to a domain expert for manual labelling. Although having human annotations is valuable, it is not always practical when a system is operating autonomously for long periods of time.

The most similar work to our own involves classifying webpages based on multimedia data (e.g., images) rather than only the contained text [52]. Like our approach, they use CNNs to perform feature extraction from images and use those features during case retrieval. The primary difference between their work and our own is that they only classify images into predefined classes, so no novel object discovery is performed. They do perform outlier detection, but that is to identify mislabeled or irrelevant images contained in a webpage rather than to detect novel webpage themes; outliers influence the case structure but do not modify the set of class labels.

As we mentioned previously, existing approaches to Computer Vision tend to focus on object classification (e.g., CNNs [34]) or detecting regions containing objects (e.g., R-CNNs [9] [10] [13]). These approaches rely on a predefined set of object types, with fewer works examining novel object discovery. Existing approaches for unsupervised object class discovery are similar to our own work in that they learn from images containing a single object type per image [53] [54] [55]. However, as we mentioned previously, the images we use in this work often contain multiple objects in each image but with only one of the objects labelled by human annotators. The

primary difference between these approaches and our work is that they perform offline object detection using the entire dataset. Our approach is both online and incremental; novel object types are detected at run-time based on the content of input images. To the best of our knowledge, no other approaches exist to allow online and incremental unsupervised object discovery. As we discussed previously, existing computer vision systems can only identify that an input image is unlikely to be of a known object type. They do not provide online labels for these unknown objects or learn from them (i.e., how to classify future images of that object type). However, our approach can perform such labelling and learning, and can learn after retaining only a single case.

Our algorithm learns in an unsupervised manner when no expert-annotated training cases are provided to it (e.g., as in our evaluation that starts with an empty case-base). As such, it has many similarities to clustering since it is grouping input images by assigning them generated class labels. Many traditional clustering algorithms, like K-means [56], divide data into a fixed number of partitions, whereas our approach dynamically creates new object types as necessary. Hierarchal clustering methods, like single-linkage [57], are able to dynamically increase the number of clusters created but do not cluster incrementally; the entire dataset must be provided as a batch. Incremental clustering algorithms have been developed, such as incremental k-means [56], that allow data points to be added sequentially rather than as a batch. However, even incremental clustering algorithms rely on comparing each data point to a set of cluster centroids. Our approach compares data points (i.e., input images) to any existing case in the case-base. This is important given

the two-stage retrieval process we use. Since retrieval is based on both an images feature vector representation and its observable parts, there can be a high degree of variability amongst cases of the same object type. For example, since the similarity thresholds used by our algorithm may be relatively low, cases of the same object type may not have highly similar feature vector representations (i.e., a medium feature vector similarity but high parts similarity). Similarly, cases of the same object type may have high feature vector similarity but only medium parts similarity. If only cases representing class centroids were retrieved, an input image could appear dissimilar to all of the centroids (i.e., treated as a novel object type) but would have been similar to one or more of the noncentroid cases. Additionally, unlike clustering algorithms, our algorithm can be used for both classification and novel class discovery. Without any labeled data, it performs classification based on its generated object type labels. However, if some cases are provided using labeled training data (i.e., some supervised learning was performed) the algorithm can either generate novel class labels or perform classification based on existing object type labels.

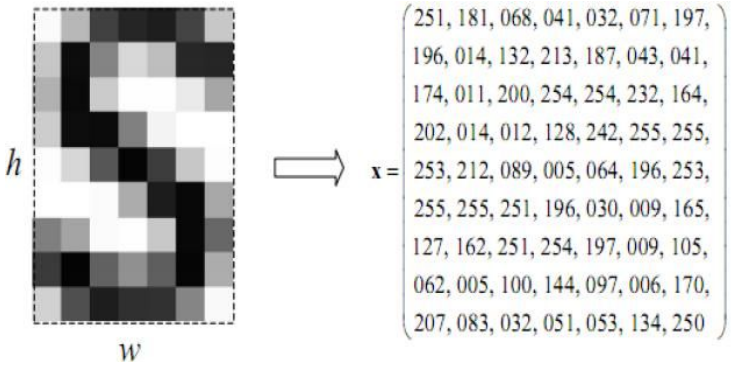
Chapter 3: Deep Neural Networks for Vision

> Marvin Minksy published a book called *Perceptrons* where he explained the problem with people stacking super deep linear networks. The book could have been called *You're all idiots.*
>
> –Tim Oates, 2012.

3.1 Introduction

All of the work that I present uses deep neural networks as the feature extractor for classification. Although the work I present does different things with these classified outputs or feature vectors used for making such an output, the featurization is always the same. In the remainder of this chapter, I define the problem of classification in Section 3.2, discussing the forward pass of the network in Section 3.2.1 used to produce the feature outputs from the network. I set up the detection problem in Section 3.3, and show original work done to speed up the detection pipeline given precomputed regions in Section 3.3.1.

Figure 3.1: **Left-** A human representation of the letter S, which we can process and classify in fractions of a second. **Right-** A machine representation of the same pixels, which are difficult to understand to humans and machines.



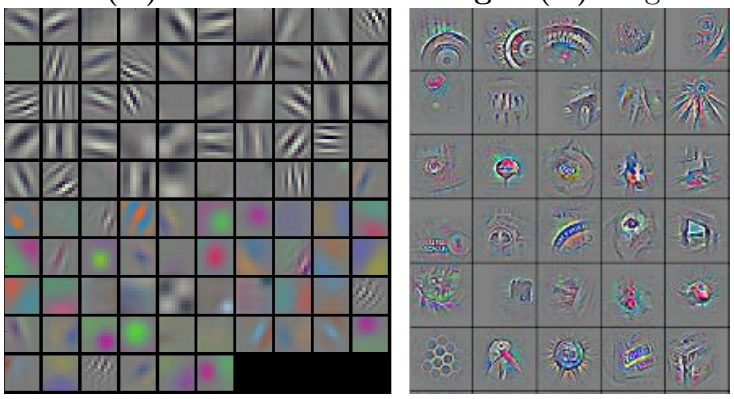
3.2 Classification

The classification problem for CNNs is as follows: given an image I with an associated class label c, find a function f such that $f(I) \rightarrow c$. Although this may seem easy to a *human* with our understanding of shapes, edges, parts, and compositions, consider the simple image with the corresponding greyscale matrix seen in Figure 3.1.

Although new networks, algorithms, layer variations, and combinations are released daily[1], we will focus our energies in understanding the forward and backwards pass on the following basic layers: Convolutional, Max Pooling, Saturating Non-Linear (we will use ReLU [40]), and fully connected.

[1] *Literally* daily. During the week of April 22 - 26 in 2019 during the writing of this section there were 222 computer vision papers published on arxiv.

Figure 3.2: **Left (A)-** Low level features **Right (B)-** High level features.



3.2.1 Forward Pass

In the forward pass of the CNN, the raw pixels are transformed into a class label that can be interpreted more easily by a human or a machine. In Figure 3.1 the desired class label output would be 'S'.

Convolutional Layers- Convolutional layers are the most important layers in a Convolutional Neural Network. These layers perform matrix multiplications over the input space of the image to extract features. In the earlier layers of the network, the convolutional neurons produce a strong activation to stimuli such as edges or curves, as seen in Figure 3.2 A. As later levels of the network perform convolutional operations on these features, complex shapes and patterns are seen exemplified in Figure 3.2 (B). These high-level features are more helpful to the final classification by the machine.

Given an image I and a convolutional filter of size $m \times n$ denoted $C_{m \times n}$, we produce an output feature map O as follows[2]:

[2]For simplicity's sake, we assume a single channel greyscale input image, and a single channel

$$O_{ij} = \sum_{i=0}^{m} \sum_{j=0}^{n} I_{ij} W_{ij}$$

making the back-propagation map of size $m \times n$:

$$\delta C_{m \times n} = \{\delta W_{ij} \forall i \in [0, m] \text{and} j \in [0, n]\}$$

where

$$\delta W_{ij} = \sum_{i=0}^{m} \sum_{j=0}^{n} I_{ij} \delta h_{ij}$$

where δh_{ij} is the error signal from the layer above. In a lower layer δW_{ij} would become δh_{ij}.

Max Pooling Layers- Max pooling layers are mapped over the input space with the aim of removing unnecessary output from the model. We do this for two reasons: (1) Max pooling layers reduce the number of parameters in later layers of the network. (2) Max pooling eliminates distractors from the signal (if we have 8 neurons saying this is a maine coon cat with high certainty, we can ignore the one that says it is a tugboat with a low probability). Pixels of the output map O are generated from the feature layer $F_{m \times n}$ from max pooling layer $M_{q \times r}$ as follows:

$$O_{i,j} = \begin{cases} F_{i,j} & F_{i,j} \geq F_{l,k} \forall l \in [0, q] \text{and} k \in [0, r] \\ 0 & \text{else} \end{cases}$$

Back-propagation of max pooling layers is somewhat simple because there is *no* error gradient in the non maximum activations. With output vector O, weight matrix W,

feature map. Non single channel images are tensor multiplications in 3 dimensions.

and input feature map F, back-propogation is defined as follows:

$$O_i = f(\sum_j W_{ij} F_j)$$

By the chain rule,

$$\Delta(F_j) = \sum_i \Delta O_i f' W_{ij}$$

However, since our function f is the max pooling function, the derivative is equal to the derivative of the identity function ($f' = 1$) on the maximum neuron output, and no derivative elsewhere, giving

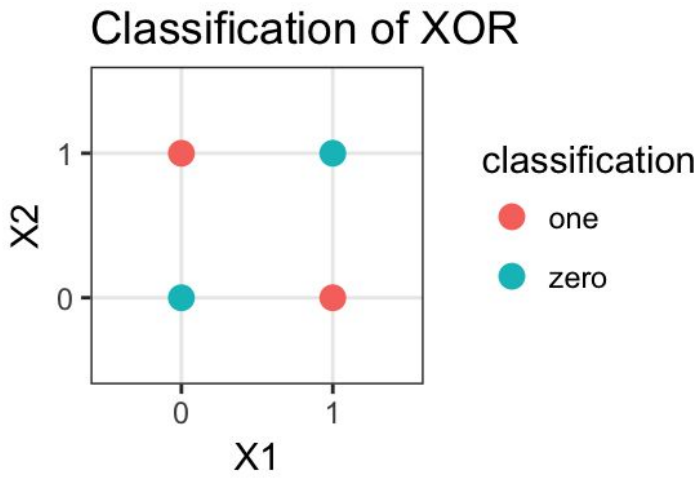
$$\Delta F = \begin{cases} \sum_i \Delta F_i W_i & \text{Max Neuron} \\ 0 & \text{else} \end{cases}$$

Rectified Linear Units (ReLU)- The ReLU layer has seen great use in recent years for the saturating non-linearity function used in a variety of deep neural networks [40]. Saturating nonlinearities serve the purpose of giving additional expressive power to the neural networks that are not obtainable in simple linear networks. A famous example of what linear networks are incapable of doing is the exclusive or (XOR) problem shown in Figure 3.3. Try as you may, a linear decision boundary cannot be drawn separating positive and negative classes in this space.

ReLU is a simple non-linear function that is easy to compute, easy to understand, and shows greater performance over traditional non-linearities [58]. The function is defined as:

$$\text{ReLU} = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Figure 3.3: The nemesis of linear classifiers.



As any Calculus I student will tell you, *the derivative is undefined at $x = 0$!* While technically correct, this is unimportant, as this is a usage for the theory of subgradients [59]. It has been shown in literature (I won't cite my entire bibliography here, but pick any paper that uses ReLU) to still facilitate faster, more robust learning. Any arbitrary value between 0 and 1 (inclusive) will do. The backpropogation is given as:

$$\frac{d}{dx}\text{ReLU} = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Fully Connected Layers- Fully connected layers often exist outside the scope of CNNs as powerful feature extractors, or dimensionality modifiers. While naively stacked fully connected layers with only linearities do not add information to networks [60], stacked, normalizing, properly trained deep neural networks can provide great boosts over non-deep solutions [1][3].

[3]Fully connected deep networks were the topic of my masters thesis in the context of seizure detection in 2014. For a more detailed explanation, please reference [61].

Given an input layer I (an $m \times 1$) vector, a hidden layer (which is a weight matrix $W_{m \times n}$ and a bias vector b of size $n \times 1$), we compute the Output layer O $(n \times 1)$ as follows:

$$O_j = \sigma((\sum_{i}^{m} W_{ij} I_i) + b_j)$$

where σ is a nonlinear function like ReLU. For the backwards pass, given a loss L at output layer Z,

$$\frac{\delta L}{\delta y_i^Z} = \frac{d}{dy_i^Z} L(y^Z),$$

we can compute the *partial* derivatives of the neurons with respect to the neuron input (x) at layer k.

$$\frac{\delta L}{\delta x_j^k} = \sigma'(x_j^k) \frac{\delta L}{\delta y_j^k},$$

giving us the errors at the previous layer

$$\frac{\delta L}{\delta y_i^k} = \sum w_{ij}^k \frac{\delta L}{\delta x_j^{k+1}}. \tag{3.1}$$

Finally, using Equation 3.1 we can calculate the gradient of the error with respect to the weights for update as follows:

$$\frac{\delta L}{\delta w_{ij}^k} = y_i^k \frac{\delta L}{\delta w_j^{k+1}}$$

The four layers listed above are a non-exhaustive list of layers used in modern neural networks, however they are meant as a sampling of the different types of layers used. Not included was a *normalizing* layer such as batch normalization [16] or dropout [15]. These layers reduce internal covariate shift of networks and reduce overfitting by jittering or adding noise to the data distribution similar to restricted boltzmann machines [62] or denoising autoencoders [63].

Using variations of these layers in known neural network architectures we are able to train the network to classify images. For some of the experiments, instead of using the softmax probability from the final layer of the neural network, we instead use the feature activation in one of the penultimate layers of the network as input to a CBR algorithm. The reasoning for this was will become clear in Chapter 5 (but briefly because the Case-Based Reasoning algorithm was not trained on *any* of the classes, restricting the feature vector power to an arbitrary number didn't make sense).

3.3 Detection

If classification (Section 3.2) can be thought of answering the question *what* is in the image, detection answers the question *where* is it in the image[4]. We have seen the framework for feature extraction and classification above, and detection is not that different as seen in Section 3.3.1. Before the convolutional feature maps are flattened to produce fully connected layers for classification, these high level feature maps (which are generally a much downsampled version of the original input) contain information about the relative location of objects in the image. Given these feature maps, and the groundtruth location of objects pixel locations in the image, we can train regression algorithms to detect and classify objects in the feature space. We'll explore how in Section 3.3.1, a published paper supported by the Office of Naval Research about speeding up the detection pipeline of CNNs in the maritime

[4]Technically it answers *what* and *where* simultaneously.

domain [41].

3.3.1 Keypoint Density Region Proposal (KDRP)

Note: Section 3.3.1 is mostly the published work of Turner et. al in 2016 AIPR in the paper ***Keypoint density-based region proposal for fine- grained object detection using regions with convolutional neural network features***. This work was supported by the **Office of Naval Research.**

Although recent advances in regional Convolutional Neural Networks (rCNNs) enable them to outperform conventional techniques on standard object detection and classification tasks, their response time is still slow for real-time performance. To address this issue, we developed a method for region proposal as an alternative to selective search, which is used in current state-of-the art object detection algorithms. We evaluate our Keypoint Density-based Region Proposal (KDRP) approach and show that it speeds up detection and classification on fine-grained tasks by 100% versus the existing selective search region proposal technique without compromising classification accuracy. KDRP makes the application of CNNs to realtime detection and classification practical.

Applications of image processing algorithms to Navy missions such as those involving intelligence surveillance and reconnaissance (ISR), maritime security, and force protection (FPr) require that they achieve high accuracy and respond in real time. Conventional approaches to image classification tasks include the use of keypoint descriptors and local feature descriptors [33], which are binned into histograms

and compared to other keypoints to match similarly featurized objects. For instance, the work of Felzenszwalb [64] on deformable part models and detection of parts gave rise to specialized part models that operate by transfer of likely locations [65], which achieved high classification and detection accuracy and speed on the fine-grained Caltech UCSD bird dataset [66]. Recently, Convolutional Neural Networks (CNNs), a deep learning approach, has emerged as a promising technique that dramatically outperforms conventional approaches on classification accuracy. Evolving from the early work of Lecun [8], who primarily focused on image classification, CNNs can now achieve state-of-the-art performance on object detection tasks [9]. Although highly accurate they are still not suitable for real-time detection and classification applications. For instance, with R-CNN and Fast R-CNN [64] the object detection pipeline takes 2 seconds.

Our goal in this section is twofold. First, we develop approaches to speed up R-CNNs so that they perform at sub-second levels, and second, we examine the effectiveness of our approach on fine-grained detection and classification tasks. Fine-grained classification tasks focus on visually similar but semantically different categories and present substantially larger interclass ambiguities, which complicate detection and classification problems. Therefore, we propose a new technique, called Keypoint Density-based Region Proposal (KDRP), to include as an integral element of the R-CNN detection and classification pipeline. The existing evaluations and applications of R-CNN have focused on standard coarse (rather than fine-grained) detection and classification tasks (e.g., PASCAL VOC 2007/2012).

However, naval applications often require fine-grained detection and classifi-

cation [67]. Therefore, we evaluate KDRP on representative fine-grained data sets, namely UEC-100 food and CUB200 birds. We found that KDRP provides a 100% speedup over Fast R-CNN without compromising detection and classification accuracy.

We structure the remainder of this section as follows: We provide background on the Fast R-CNN detection pipeline in Section 3.3.1.1. In Section 3.3.1.2, we describe our KDRP algorithm, which replaces step 2 in the Fast R-CNN detection classification pipeline. Section 3.3.1.3 describes our experiments, along with the time and accuracy results. We discuss these further in Section 3.3.1.4 and then conclude with future work plans.

3.3.1.1 Region of Interest Pooling

Here we describe the processing pipeline of Fast R-CNN [10], which we modify to reduce image processing response time. It includes the following steps:

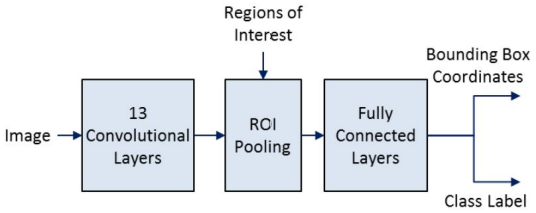
1. Region Proposal (via KDRP)
2. Feature Extraction (via a trained network)
3. Non-Maximum Suppression
4. Hypothesis Selection.

Our proposed KDRP algorithm (Section 3.3.1.2) replaces the state-of-the-art algorithm (selective search) in Step 1. We describe all four steps in the following paragraphs.

Region Proposal The purpose of region proposal is to generate enough regions such that there is a high probability that one of the regions r in the set of all regions proposed R will be an accurate region that bounds the ground truth object. Assuming we are given an image I with dimensions $w \times h$, a region is a subset $r \subset I$, with dimensions $w' \times h'$, where $0 < w' \leq w$, and $0 < h' \leq h$. The state-of-the-art approach for region proposal is selective search [23], which is an image segmentation method that uses multiple image scales and eight opponent color-spaces (consisting of one luminance channel and two color spaces stimulated by opponent cones in the retina) to generate regions that may contain target objects (in a true object detection task there can be any number from 0 to n objects detectable in the image). This is a time-intensive approach because it creates several thousand regions to increase the likelihood that at least one proposed region contains the ground truth object (to enable detection). As we explain later, because KDRP never infers bounding boxes and cannot modify the boundaries of a bounding box, the region that contains the object must be generated or detection will fail. The implementation of Fast R-CNN [10] uses selective search for region proposal. This is a shortcoming of the Fast R-CNN method, not one of KDRP.

Feature Extraction After region proposal a trained CNN generates features using image convolution, activation functions, and pooling. For a classification model trained on n classes, the output from the classification layer comprises $n+1$ probabilities. For the bounding box coordinates of these classes, there are $4n+4$ coordinates. The reason for adding one to the number of classes is that it allows a *background*

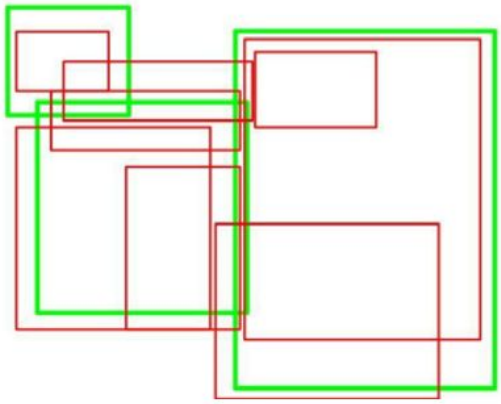
Figure 3.4: Feature extractor R-CNNs topology, modeled after the VGG16 architecture [6], with Regions of Interest (ROIs) and the final convolutional filters being given as input to the ROI pooling layer. The 13 convolutional layers, pooling layers, and Rectified Linear Unit (ReLU) layers are combined into one to increase the figures readability.



class. Each of the regions classification is the softmax of the $n+1$ probabilities, and each class corresponds to a 4-tuple of coordinates. The computationally expensive convolutional phase needs to only occur once; in this implementation the Regions of Interest (ROIs) that are detected are passed through the network at the same time as the image itself, and translated to the ROI pooling coordinates. After the ROI pooling layer has been computed, the only computations that need to be repeated multiple times are the multiplications between the fully connected layers, and dense matrix multiplications have been optimized to be extremely fast on GPUs. The network topology is shown in Figure 3.4.

Non-maxima Suppression Non-Maxima Suppression (NMS) is a technique for images containing an unknown number of objects. It combines likely detections of the same class together into one object if the detections overlap enough. NMS is very fast; it runs in $O(mcn^2)$ time, where m is the number of images processed, c is the number of classes, and n is the number of regions. A description of NMS is

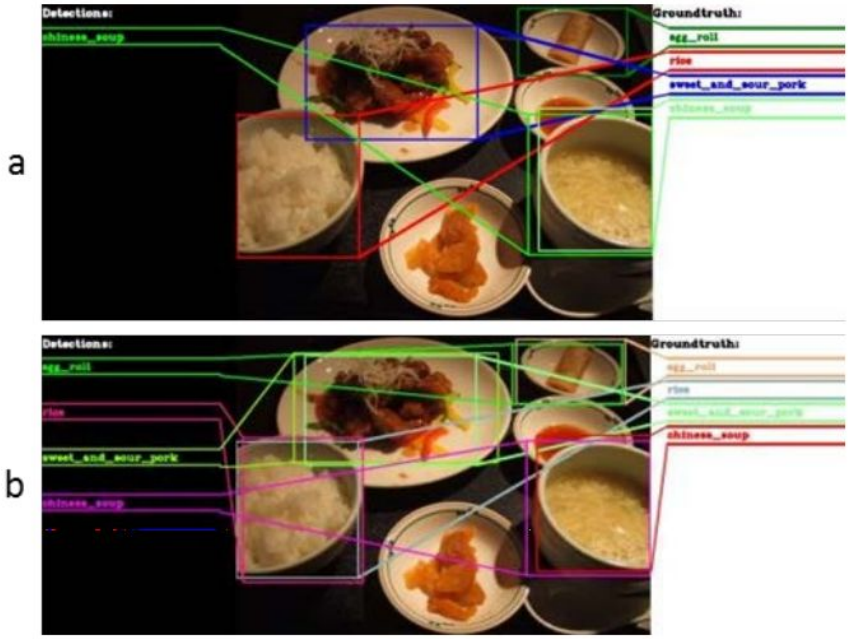
Figure 3.5: The three green regions are selected because they are the highest probability regions in the area that do not overlap at a fraction greater than α with a higher probability region. The red regions are suppressed because the regions they occupy were already occupied at a fraction greater than α by a higher probability region



given in [9] and an example is shown in Figure 3.5.

Hypothesis Selection Hypothesis selection assigns class labels to regions of an image that have been featurized, or disregards the region. Like NMS, it is very fast with respect to region proposal or featurization. Hypothesis selection can be undertaken in one of two ways. In the first method, the algorithm is told the number of objects to detect in the image (e.g., Figure 3.6-a shows a situation where it is told that there is one object in the image, when there are in fact 4). In the second (more realistic) method, the algorithm is not told how many objects (if any) are in the image, and it must determine which of its thousands of regions are valid detections using the confidence output from the R-CNN, as shown in Figure 3.6-b.

Figure 3.6: Comparison of two hypothesis selection methods, namely (a) top-k selection, or (b) selecting all detected objects greater than a known probability threshold. On the top, the system is told to select only one hypothesis (shown on the left) despite four ground truth objects being present. On the bottom, it is allowed to detect multiple objects, and all four ground truth objects are detected.



Function: $KDRP(I_{in}, k)$ ***returns*** R_{out}

1 $keypointCoords \leftarrow siftlike_kpDetect(I_{in})$;

2 $keypointMean \leftarrow mean(keypointCoords, I_{in})$;

3 $keypointStdDev \leftarrow sDev(keypointCoords, I_{in})$;

4 $R_{out} \leftarrow []$;

5 **while** $len(R_{out}) < k$ **do**

6 $r \leftarrow generateRandomRegion(I_{in})$;

7 $densityPct \leftarrow getPercentile(len(r), keypointMean, keypointStdDev)$;

if $binomialSuccess(densityPcte)$ **then**

8 $R_{out} += r$;

end

end

9 **return** r_{out};

Algorithm 1: Keypoint Density-based Region Proposal (KDRP) Algorithm

3.3.1.2 Keypoint Density Region Proposal Algorithm (KDRP)

Given the inefficiencies of selective search, we propose an alternative for region proposal called KDRP (Algorithm 1). It generates k arbitrary regions of an image in two steps:

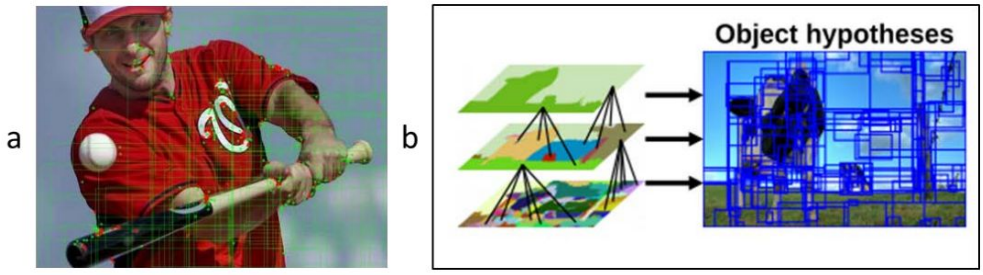
1. **Keypoint Generation:** Line 1 computes SIFT features, which correspond to areas of a large change in the image gradient. These areas are known to include distinguishing points of an image [33]. By focusing on regions with a

high density of these descriptive keypoints, there is a greater likelihood that one region accurately bounds the target object. KDRP uses standard keypoint detection algorithms to find and plot points of interest on the image. The default SIFT-like features we use in step 1 are determined by cross validation using permutations of 8 keypoint methods on the UEC food dataset and the CUB bird dataset. The greatest performance boost is due to Shi Tomasi features, Oriented Rotated Brief (ORB) features, and Solenoidal Tracker At Realivistic Heavy Ion Collider (STAR) features. This differs from the existing method of segmentation and nearest neighbor search of color segments, under the assumption that regions will contain strong corners and edges of the image, which is needed for recognition and detection.

2. **Region Hypothesis:** In line 2, a square window is slid over the given image using a uniform stride,and the density of key points from line 1 is sampled. After computing the mean and standard deviation of the region density, lines 5 through 8 stochastically generate a region (not of fixed width to height ratios), and examines how its density compares to the baseline just established. Regions in the n^{th} density percentile are retained for detection following a binomial sampling with an $n\%$ chance of success. Sampling in this manner has been shown to reduce bias [1]. This differs from traditional RCNN with selective search in the time required to produce the equivalent number of regions.

A major advantage of KDRP versus selective search is that the exact number of

Figure 3.7: KDRP example of Max Scherzer batting. Red keypoints are ORB features, while green keypoints are STAR features. Only 5% of the regions are shown for increased visibility (a), and color segmentation used to generate selective search regions is shown in (b).



regions to be used can be selected. The selective search implementation in MATLAB (as used in [9] [10]) has a fast and slow mode (number of opponent color spaces that are clustered making the speed difference), but the former cannot generate more regions than the slow mode. We conjecture that using relatively fast keypoint detection methods [33] will be faster than the selective search method that generates and clusters opponent color spaces. An example of regions generated with KDRP is shown in Figure 3.7-a, while a different image generated using selective search is shown in Figure 3.7-b.

3.3.1.3 Experimental Results

Our evaluation objective is to assess the performance advantages of KDRP versus selective search on fine-grained detection and classification tasks. Our metrics are accuracy and response time.

We evaluate the two region proposal algorithms on the following two datasets

in Table 3.3.1.3. Since both datasets include bounding box coordinates, they are suitable for detection and classification evaluation. They offer unique challenges for detection above and beyond being fine-grained:

- **UEC-100 food dataset:** This contains pictures of 100 categories of Japanese food, with bounding box annotations provided. UEC-100 has a variable number of ground truth objects per image, so we cannot hard code the algorithm to search for a pre-specified number of items. Some example categories are Miso Soup, Chinese Soup, Soba Noodle, and Udon Noodle.

- **CUB-200 bird dataset:** This contains pictures of 200 bird categories, with bounding box annotations provided. This presents a notoriously difficult classification task due to its many fine-grained categories. Example categories include American Crow, Yellow Bellied Warbler, Rock Warbler, and Baltimore Oriole.

Dataset	Train/Test	Number Classes	Instances Per Image
UEC 100	10205/2966	100	Variable $(0, n]$
CUB 200	5994/5794	200	1

Table 3.1: Dataset Attributes for UEC-100 and CUB-200

We compared our implementations of the following two region proposal algorithms:

1. Selective Search Region Proposal (SSRP)

2. Keypoint Density Region Proposal (KDRP)

To test the two alternative region proposal approaches, we kept the detection-classification pipeline identical across the approaches except for line 1 (in Algorithm 1). However, the differences in the region proposal approaches do affect the downstream processes of feature extraction, NMS, and hypothesis selection.

For feature extraction, we separately trained and fine-tuned the pre-existing ImageNet model VGG16 [6], modifying only the output layers of the classification and bounding box regression steps. No other layers were modified (fully connected or convolutional). Fine-tuning was performed over the course of 5,000,000 iterations with a base learning rate of .01, decreasing by a factor of 10 every 500,000 iterations. The momentum term was set to .9, and weight decay was set to .0005.

During training, proposed regions were generated by the SSRP and KDRP algorithms, respectively. Using [23], selective search was applied to generate region proposals. The number of regions per image varied, as it depends on color features of the image, and averaged to around 2,000 per image. The number of regions generated by KDRP was set to 2,250; this was the maximum number of regions that could be used without the total time exceeding our predetermined threshold of 1 second per image.

Featurization has no tunable parameters, NMS was applied for high probability windows with an overlap exceeding 30% (as done by [9]), and the threshold for selecting a region for hypothesis selection when there is an unknown amount of objects in the ground truth was set to $p > .88$. This value was determined through

cross validation of a small hold out set from training.

We used the following two measures to compare the algorithm's performance:

- **Time** (t): The response time taken to process a test image.
- **Detection accuracy** (a): We consider an object to be correctly detected if the intersection over union of the ground truth and bounding boxes is greater than 50%. Correctly identified ground truths are True Positives (TP), ground truths not covered by at least 50% of the detection box (or of the wrong class) are False Negatives (FN), and bounding boxes that do not correctly detect exactly one object are False Positives (FP). Average accuracy is defined as follows, for $x \in \{\text{SSRP, KDRP}\}$: $\mu a_x = \frac{TP}{TP+FP+FN}$

We tested the following two null hypotheses using standard $t-$tests:

- KDRPs pipeline is faster than SSRPs pipeline: $H_0 : \mu t_{KDRP} < \mu t_{SSRP}$, where μt_{KDRP} is the mean image processing time for the KDRP pipeline and μt_{SSRP} is the mean image processing time for selective search.
- There is no difference between the detection accuracies of the KDRP and SSRP pipelines. $H_1 : \mu a_{KDRP} = \mu a_{SSRP}$ where μa_{KDRP} is the mean accuracy for the KDRP pipeline and μa_{SSRP} is the mean accuracy for selective search.

The results of our evaluation are summarized in Table 3.2, which shows that KDRPs image processing response time is less than half of SSRPs for the two data sets (.96 seconds versus 1.96 seconds, and .968 versus 1.997 seconds). Therefore, we accept Hypothesis H_0.

Table 3.2: Mean time and detection accuracy of the KDRP and SSRP pipelines

Measures	Processing Time			Detection Accuracy		
Dataset	KDRP	SSRP	p	KDRP	SSRP	p
UEC 100	.960	1.960	<.0001	68.03	68.19	.546
CUB 200	.968	1.997	<.0001	66.24	65.18	<.0001

Table 3.3: Region proposal computation time as a percentage of total processing time

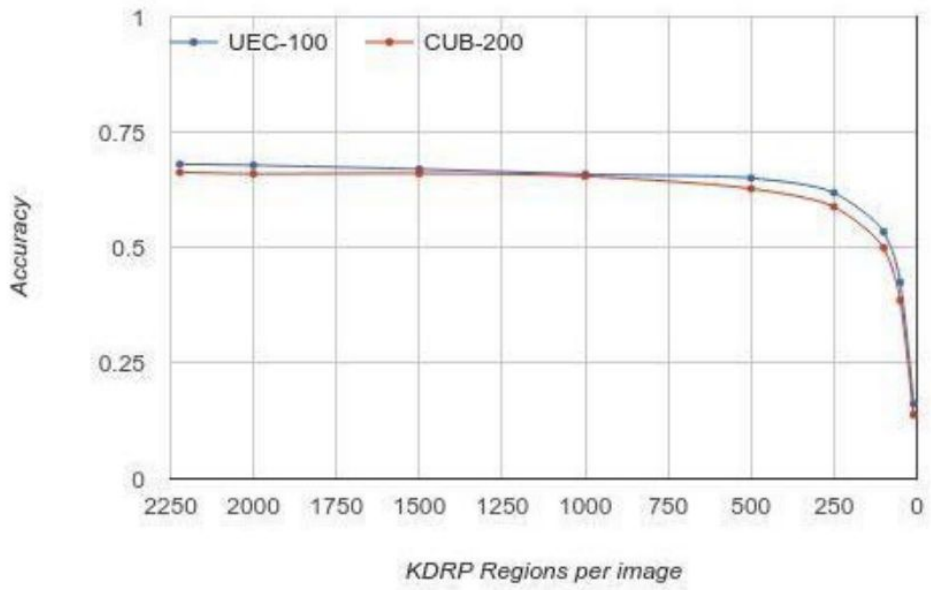
Dataset	KDRP	SSRP
UEC-100	63.0%	85.0%
CUB-200	62.5%	86.1%

Also, KDRPs detection accuracy does not differ significantly from SSRPs (68.03 versus 68.19, $p = .54$) for the UEC data set but is significantly higher for the CUB dataset (66.24 versus 65.18, $p = .00009$). Thus, Hypothesis H_1 is not supported.

We conclude that KDRP increases speed by roughly 100% versus SSRP without compromising detection accuracy in all cases (further investigation is warranted to why sometimes it does change accuracy). Furthermore, it reduces the overall processing time per image to under a second, which permits near real-time processing (a requirement for our target applications).

Earlier, we argued that region proposal is the most time-intensive step in the

Figure 3.8: Effect of proposing fewer regions per image on detection accuracy for UEC-100 and CUB-200



detection and classification pipeline. Table 3.3 shows that it is as high as 63% for KDRP and 86% for SSRP. Consequently our efforts in reducing the overall computation time were well motivated.

Unlike SSRP, KDRP provides the ability to control the number of regions proposed. We reported the results in Table 3.2 with a setting of 2250 proposals. We explored the relationship between the number of proposed regions and accuracy. The results of our study are shown in Figure 3.8. There is steady but small decrease in accuracy until reducing from 2250 to 500 proposals, after which there is a steep decrease. The overall computational time at 500 proposals for one image is .361 seconds for CUB-200 (region proposal accounts for 43% of total time), and .372 seconds for UEC-100 (45%).

3.3.1.4 Applications of KDRP

Navy applications that require detection and classification of objects in images demand high accuracy and real-time response. However, while deep networks promise high levels of accuracy, they cannot yet respond in real-time. In this section, we presented and evaluated a new approach called KDRP that modifies the current deep network approach to substantially reduce the response time without compromising accuracy. Furthermore, we evaluated our approach on fine-grained classification tasks that are relevant to naval decision making. Our goal was to develop a pipeline that places accurate bounding boxes around objects of interest in a dataset in under a second, so that it could be used in a low frame per second surveillance camera setting. We demonstrated that this goal is practically achievable with currently available hardware. In our future work, we will investigate methods for improving KDRP by adding different region aspect ratios or scales upon acceptance. The nature of Fast R-CNN networks makes additional convolutions take marginal time, and increasing the number of regions proposed can only increase accuracy.

Chapter 4: Contextual Visual Signals

> Please don't use this sentence out of context.
>
> –Kevin McPherson-Eckhoff, Easy Peasy.

4.1 Introduction

In this chapter, we explore two different meta-algorithms for leveraging spatial context to boost information gain. In Section 4.2 (which is my work from the paper *SPARCNN: SPAtially Related Convolutional Neural Networks* published in the *Applied Imagery and Pattern Recognition Workshop 2016*) we use co-located objects in the scene, and relative size and aspect ratio information to boost accuracy. In Section 4.3 (submitted as a full paper titled *Unknown Target Classification: A Contextual Classifier Study* currently awaiting decision on acceptance from the *British Machine Vision Conference 2019*) we explore using regions of non-target (or background) pixels of an object, and study the varying effects on object detection these pixels can cause.

4.2 Object Correlation (SPARCNN)

Note: Section 4.2 is mostly the published work of Turner et. al in 2016 AIPR in the paper ***SPARCNN: SPAtially Related Convolutional Neural Networks****.* This work was supported by the **Office of Naval Research.**

The ability to accurately detect and classify objects at varying pixel sizes in cluttered scenes is crucial to many Navy applications. However, detection performance of existing state-of-the-art approaches such as convolutional neural networks (CNNs) degrade and suffer when applied to such cluttered and multi-object detection tasks. We conjecture that spatial relationships between objects in an image could be exploited to significantly improve detection accuracy, an approach that had not yet been considered by any existing techniques (to the best of our knowledge) at the time the research was conducted. We introduce a detection and classification technique called Spatially Related Convolutional Neural Networks (SPARCNN) that learns and exploits a probabilistic representation of inter-object spatial configurations within images from training sets for more effective region proposals to use with state-of-the-art CNNs. Our empirical evaluation of SPARCNN on the VOC 2007 dataset [68] shows that it increases classification accuracy by 8% when compared to a classification schema that does not exploit spatial relations. More importantly, we obtained a higher performance boost of 18.8% when task difficulty in the test set is increased by including highly obscured objects and increased image clutter.

Evolving from the early work of [8], which primarily focused on image classification, CNNs can now achieve state-of-the-art performance on object detection

tasks [10]. Although CNNs have become adept at processing pixels to classify objects, and even computing bounding box targets based on the objectness score of the region, there is additional information about the object or objects in an image that we cannot discern from a low level pixel signal. In this paper, we present a new system for multiobject detection in images with clutter called Spatially Related detection with Convolutional Neural Networks (SPARCNN). SPARCNN includes the following three key features:

- It leverages and extends our previous state of the art region proposal technique called KDRP [41]; KDRP is a region proposal technique that uses density of high interest features to propose regions with higher likelihood for objects of interest.

- It recursively proposes regions based on where it has previously observed objects.

- It adjusts thresholds for object detection based on what objects have been detected with a sufficiently high confidence.

The rest of the section is organized as follows: Section 4.2.1 presents the contributions of the SPARCNN approach to the existing detection pipeline with a subsection devoted to each of the three features enumerated above. Section 4.2.2 presents the results of SPARCNN evaluation on the VOC 2007 dataset [68], and Section 4.2.3 concludes the paper with a discussion and outlines our planned future work.

4.2.1 Methodology

SPARCNN is designed to detect objects in a cluttered image with high accuracy. During training, two models are trained for use by SPARCNN; Fast R-CNN [10], and the Spatial Relation Model (SRM).

SPARCNN Training SRM captures the following attributes about a training dataset assuming that there are n classes, it stores the following information:

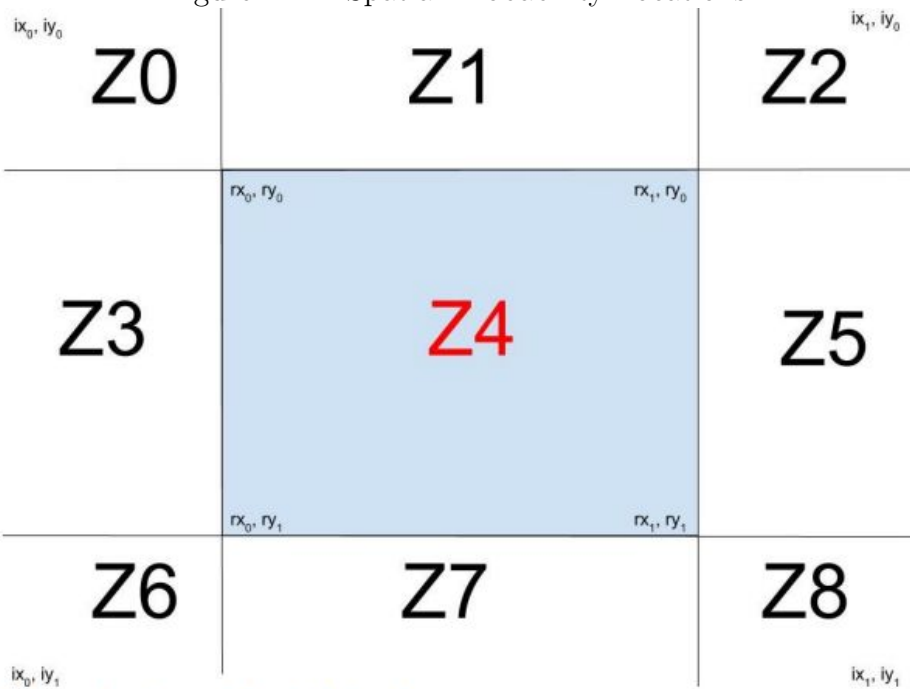
1. *Fraction of Class Label-* SRM sums all objects that are of a given class a, and creates an $n-$dimensional list of the probability of any given class

2. *Fraction of Images Present-* : Sums over all images I where there is at least one instance of an object of class a, and stores them in an $n-$dimensional list

3. *Conditional Probabilities-* Given a class a, and another class b, this is calculated for a given b as the probability of any given image containing a and b divided by the probability of an image only containing b. This is stored in an $n \times n$ matrix.

4. *Spatial Probabilities-* Given class a, and another class b, this is the normalized fraction of locations on the divided grid, as shown in Figure 4.1. The anchor object class a is defined to occupy 100% of z4 in the diagram, and the secondary object has its overlap with each other region calculated. For example, an object that is strictly above Z4 would increment (1 object * 1.00 overlap). This is done for every pair of objects in every image, and normalized, so the end result

Figure 4.1: - Spatial Probability Locations



is an $n^2 * 9$ sized lookup table, where any given entry is the normalized fraction where class b has occurred in relation to class a.

5. *Relative Sizes*- Given class a, and another class b, this is an $n^2 \times 2$ dimensional list of the mean relative pixel^2 sizes of $\frac{a}{b}$, as well as the standard deviation of the relative sizes.

6. *Aspect Ratios*- For any arbitrary class a, an $n \times 2$ table is calculated for the mean and standard deviation of the shorter side of the image over the larger side of the image. All aspect ratios will fall in the range of (0,1].

Region proposal is the only difference in the training of SPARCNN versus Fast R-CNN; the SRM is also trained. The Fast R-CNN models trained in [10] can still be used to process the image corpus.

Applying SPARCNN The application of SPARCNN varies from that of traditional Fast R-CNN, both in terms of region proposal and hypothesis selection. Both techniques leverage the SRM to search for and select objects that are overlooked by a simpler region proposal and convolution method. SPARCNN is a recursive method for object detection that proposes regions based on highly confident detections, and adjusts detection thresholds based on the objects in the image that we are confident of observing. Pseudocode for SPARCNN is shown in Algorithm 2.

Region proposal in SPARCNN is performed in three recursive tiers based on object size: large object, medium object, and small object.

Region proposal is an iterative and recursive process. Iteration is done using three ranges of window sizes; large (where the width of the region is between 40% and 99% of the width of the image, height of the region is with 40% and 99% the height of the image), medium (constrained similarly between 10% and 64%), and small region proposal (constrained between 2% and 16%).

Region Proposal Building on the work of KDRP [41], SPARCNN uses a keypoint density based approach for region proposal. As more objects are detected in an image, prior knowledge of cooccurring objects can be leveraged to improve the proposal of regions to search for additional objects nearby. Once KDRP has detected an object, it begins a new type of region proposal (the function genSRMregion in the pseudocode). Regions from the SRM are generated to be consistent with training set observations. Algorithm 3 initializes keypoints as a blank array, and its first loop is over every detected object in the image. For each object detected with

Function: *SPARCNNDetect*(I_{in}, SRM) ***returns*** $confirm_{obj}$

```
1  confirmed ← [];
2  for sizein[LARGE, MEDIUM, SMALL] do
3      firstLoop ← True;
4      nmsDetection ← None;
5      while nmsDetection ≠ Noneor firstLoop do
6          firstLoop ← False;
           if knownDetections == None then
7              regions ← getKDRPregion(size, img);
           end
           else
8              regions ← getKDRPregion(size, img, SRM, confirmed);
           end
9          newDetections ← sparcnnDetect(regions, SRM, confirmed);
10         nmsDetections ← nonmaxSupress(regions, confirmed);
11         confirmed ← nmsDetections;
       end
   end
12 return confirmed;
```

Algorithm 2: Pseudocode for SPARCNN detection

a sufficiently high probability, it loops through every class c observed in the training set, and counts the number of times n the detected object class and objects in class c co-occurred in the training set. Then using the spatial probability location grid of Figure 4.1, β (given $\beta = 15$ is a constant number to produce more keypoints and regions determined through cross validation) keypoints are randomly generated in the corresponding sector of the grid, and their (x, y) locations are recorded.

Function: *genSRMregion*($I_{in}, SRM, size, confirmedDetects$) ***returns*** $SRMregions_{out}$

```
keypoints ← [];
for detectioninconfirmedDetects do
    for classinsrm.classLabels do
        coOccur ← srm.getCoOccurances(det.class, class);
        keypoints+ = kdrpKeypointGen(srm.loc, det.cls, class);
        SRMregions_out = kdrpGenerate(size, num_reg =
          β ∗ coOccur, keypoints);
    end
end
return SRMregions_out;
```

Algorithm 3: SRM region generation pseudocode

For example, if objects of type person and dog co-occurred 73 times, SPAR-CNN would generate $1,095$ keypoints. Suppose that 30% of dogs were located below people (box Z7), 50% were found overlapping people (Z4), and 5% and 15% were

Figure 4.2: - Regions generated by SRM information

found to the left and right (Z3 and Z5 respectively). SPARCNN would mirror and split the grid on the central y axis (we assume that for everyday objects it does not matter if something is to the right or left; consider an object or video reflected about the y axis, it is easily understandable by humans both ways), $\frac{z_3+z_5}{2} = 10\%$, which yields new values for Z3 and Z5. SPARCNN would then distribute the 1,095 keypoints evenly in proportion to the spatial matrix (i.e., 329 keypoint locations under the person detection in Z7, 547 keypoint locations overlapping the person, and 109 keypoints generated on the left and right side of the person, respectively). This is done for every class, and then used as input to KDRP. Usually, KDRP operates by detecting binary patterns and keypoints in the changes of gradient of the image to generate regions [41], but these keypoint locations can be given directly to KDRP so it selects high keypoint regions instead of local binary patterns [33]. Using this method, SPARCNN is likely to generate the following regions looking for a dog given information on where a person is located as seen in Figure 4.2.

Hypothesis and Threshold Adjustment The confidence of the detections has, to this point in the SPARCNN process, been fixed; only different regions have been proposed than would have been proposed by a traditional selective search algorithm [23] or by a region proposal network [13]. Although region proposal techniques can be used to reduce time constraints [41], as long as the correct region is proposed by multiple region proposal techniques, they are unlikely to reduce detection accuracy. There are two ways to adjust hypothesis acceptance; by raising or lowering the threshold of probability needed for detection (T_p), or by raising or lowering the probability of the region that has been convoluted R_p. T_p is selected through multi fold validation to produce the maximum detection accuracy in all cases where $R_p \geq T_p$. In general, the amount that we want to change the probability is split between R_p, and T_p such that the probability of detection will be a positive number, but a number smaller than 1.00 (since this would make it impossible to identify an object). We used a tuning set to set a minimum signal strength needed for detection of .36 from the network. No matter what objects are around it, and if it matches the aspect ratio and relative size perfectly, positive detections cannot be set at lower values without spurious detections. The three types of evidence from the SRM can be used to influence threshold or detection confidence:

1. Object Aspect Ratio

2. Object Correlation

3. Object Relative Size

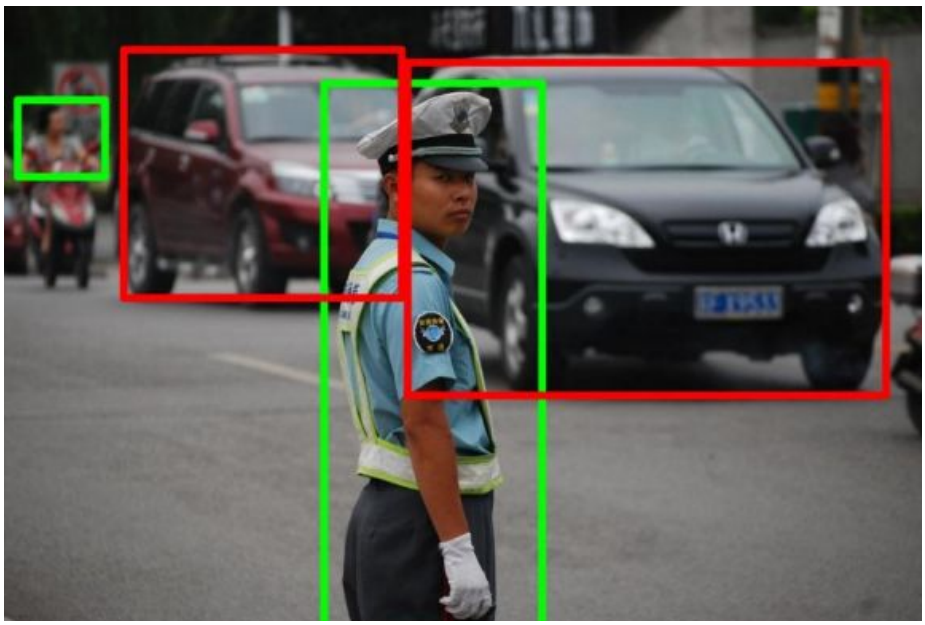
Object Aspect Ratio For each object in the training set, we record the ratio of the longest side to the shortest side of the object, along with its class c. We do not use a fixed height and width because, for example, a bottle (which is usually a little more than twice as long as its width) could be misidentified if it were laying on its side (in a bottle rack for example). After computing these ratios, we calculate the mean aspect ratio (Ac) and the standard deviation of the aspect ratios (Sc).Because aspect ratio may be noisy (e.g., there may be an oddly shaped water bottle, or perhaps a person has a square shape due to a kneeling posture), even if the aspect ratio matches the threshold, it might shouldn't necessarily be changed greatly. When applied, SPARCNN computes the aspect ratio and the Z-score (number of standard deviations away from the mean), and the region probability and threshold probability are adjusted as shown in Table 4.1.

Table 4.1: Aspect Ratio Evidence

Experimental value x in Z score	Classification	δR_p	δT_p
$-1 \leq x \leq 1$	Evidence For	+ .02	-.02
$-2 \leq x$ or $x \geq 2$	Neutral	0	0
$-3 \leq x$ or $x \geq 3$	Evidence Against	- .02	+ .02

Object Correlation The increase in SPARCNNs accuracy is primarily due to the use of object correlations to boost detections. SPARCNN creates a copy of the

Figure 4.3: Two people are visible; one is larger than the cars and one is much smaller



image, but instead of 3 pixel values at each (x, y) coordinate, it assigns a probability modifier for each class. After an object in class A is detected, then for every object in class B, SPARCNN, will update every pixel using the probability modifier to reflect changes in probability of all the classes. In the SRM, let the fraction of objects that occurred in the same spatial position with respect to A be S_A, and let $P_B = P(B|A)$. SPARCNN modifies the value needed for detection as follows: $T_p = T_p(S_A \times P_B)$. This ensures that objects that are conditionally codependent will lower the threshold, and the effect is even greater if they were in a previously detected spatial relation. Each pixel on the representation of an image is assigned a new threshold weight. To determine the threshold needed for any given region, SPARCNN sums all of the pixel value thresholds contained in that region, and averages them.

Table 4.2: Relitive Size Evidence

Experimental value x in Z score	Classification	δR_p	δT_p
$-1 \leq x \leq 1$	Evidence For	+ .02	- .01
Else	Inconclusive	0	0

Relative Object Size For every pair of objects in the training images, the relative size of every object is recorded. The means and standard deviations of the class wise pairs are computed. Much like aspect ratio, this is a weak evidence for object identification, as objects that are near or far from the camera may appear to be incorrect in object size, but actually be a real detection, as highlighted in Figure 4.3. This is considered weaker evidence than aspect ratio.

4.2.2 Experimental Evaluation

Objective And Hypothesis Our objective is to assess whether, by leveraging (1) spatial relationships between objects and (2) conditional probabilities as described in Chapter 3, SPARCNN would outperform neural networks using the same region proposal techniques and network topology. Our first hypothesis (H_1) is that by detecting additional objects, the increase in recall from previously overlooked objects will be greater than the false positives that arise from misidentifying objects, so we expect an increase in accuracy and F_1 measure.

$$H_1 : Accuracy_{SPARCNN} > Accuracy_{BASELINE} \tag{4.1}$$

Our second hypothesis H_2 is that even with the spurious false positives from SPARCNN, the added true positives will increase accuracy and F_1 measure enough such that the Area under the ROC curve (AUC) will be no less than the AUC of the baseline. Stated formally:

$$H_2 : AUC_{SPARCNN} = AUC_{BASELINE} \tag{4.2}$$

We test H_1 using an A/B Split test, and H_2 using a classwise paired t test. We used the very deep network full model VGG-16 [6] trained using Fast R-CNN [10]. We set hyper parameters and SRM evidence levels (Table 4.1 and Table 4.2) using 5-fold cross validation on a held-out data set. In this experiment, we compared two systems:

- Baseline: uses KDRP + Fast R-CNN without using SRM for region proposal or hypothesis selection.
- SPARCNN: uses the additional region proposals and hypothesis selection criteria, and undergoes hypothesis changes and detection threshold adjustment as described in Section 4.2.1.

Datasets We tested SPARCNN only with PASCAL VOC 2007 [69]. The dataset split and annotations were the same as used in [10], and dataset characteristics are given in Table 4.3.

PASCAL VOC 2007 also has a difficult flag that can be toggled *True* or *False*. An object in the image can be labeled as difficult for several reasons, most often

Table 4.3: PASCAL VOC 2007 characteristics

Characteristic	**Value**
Number of Classes	20
Class Distribution	Skewed (349 table vs. 8000 person)
Objects Per Image (px)	1 - 42
Target Object Size	44 - 248k
Train / Test Split	8539 / 1424

because it is cropped or partially shown in the image, as exemplified in Figure 4.4.

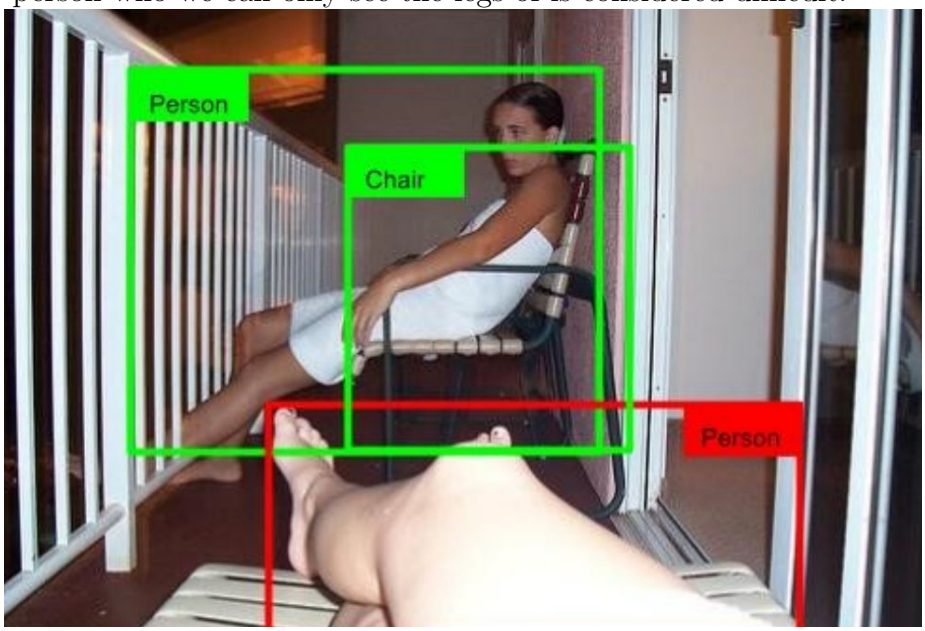
Evaluation Metrics We measured the algorithm using the following measures: accuracy, recall, precision, F_1 measure, and AUC. Three outcomes were recorded for each detection attempt/undetected object:

- *True Positive (TP)*: A true positive is recorded if the predicted bounding box has an intersection over union (IoU) of greater than 0.5, and is of the correct class.

- *False Positive (FP)*: A false positive is recorded for every detection that does not have an IoU of greater than 0.5 with a previously undetected object of the correct class.

- *False Negative (FN)*: A false negative is recorded if none of the system detections match the ground truth bounding box for IoU and class label.

Figure 4.4: The two green objects are not difficult because they are entirely visible, but the person who we can only see the legs of is considered difficult.



Using these definitions, the standard for the following four terms is defined as:

- Accuracy = $\frac{TP}{TP+FP+FN}$
- Recall = $\frac{TP}{TP+FN}$
- Precision = $\frac{TP}{TP+FP}$
- $F_1 = 2 * \frac{Precision*Recall}{Precision+Recall}$

Results Table 4.4 and Table 4.5 show the results (for the first four metrics) for two dataset conditions: (1) without and (2) with difficult annotations, where the boldfaced number indicates the system that significantly performed better.

For both datasets, SPARCNN outperformed Baseline on accuracy, recall, and F_1, but performed worse on precision. This is because SPARCNN adds detections

Table 4.4: PASCAL VOC 2007 Evaluation without difficult annotations

Metric	Baseline	SPARCNN	Change (Percentage)
Accuracy	45.96	**49.29**	7.27
Recall	51.72	**66.78**	29.12
Precision	**80.48**	65.3	-16.86
F_1 Measure	62.97	**66.04**	4.88

Table 4.5: PASCAL VOC 2007 Evaluation with difficult annotations

Metric	Baseline	SPARCNN	Change (Percentage)
Accuracy	39.89	**47.37**	18.75
Recall	42.92	**58.17**	35.53
Precision	**84.97**	71.84	-15.45
F_1 Measure	57.04	**64.29**	12.71

that would have been skipped due to lower confidence than the needed threshold. Although SPARCNN does this correctly more often than not (as evidenced by the higher accuracy and F_1 measure), it also creates additional false positives, which reduces precision. The A/B split testing for both the standard and difficult splits are statistically significant at a level of $\alpha = .05$, so we accept the hypothesis H_1.

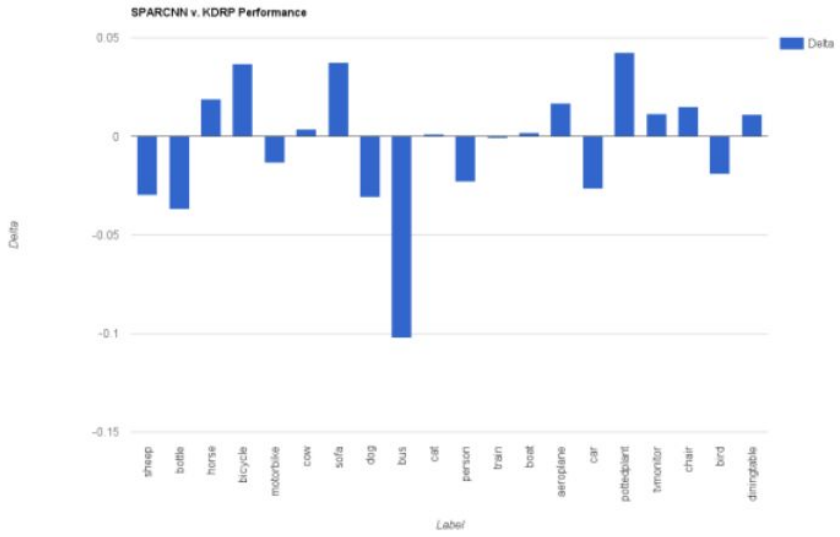
We also found that SPARCNN increases relative performance for difficult (i.e., cluttered, overlapping) scenes. The percentage change from the non-difficult to difficult dataset conditions, in comparison with Baseline, is more than double, and the F_1 measure increases nearly three-fold, while the percentage change in precision actually decreases. As more clutter and obfuscation of ground truth target objects are added to an image, fewer false positives result, which increases precision.

In the more commonly used metric for PASCAL VOC 2007 evaluation (AUC), there was no significant difference at a level of $\alpha = .05$ between the baseline (0.6474) and SPARCNN (0.6431). A classwise comparison is shown in Figure 4.5. Although for some classes SPARCNN does better, other times it does worse, and is often not statistically significant. Overall SPARCNN has less area under the ROC curve by a small amount.

4.2.3 Conclusion

Although SPARCNN did not outperform Baseline for AUC, this metric does not accurately highlight its improvements. Using the same cross validation scheme to select parameters for SPARCNN to use for detection threshold levels as the baseline

Figure 4.5: Classwise comparison of AUC for SPARCNN v. Baseline



algorithm, there exists a set of parameter settings for which SPARCNN significantly outperforms Baseline in terms of object recall, while also increasing accuracy and F_1.

This study warrants future work in possible improvements to SPARCNN so that it can be applied in real-time tasks that require instantaneous monitoring and detection. For example, we plan to use a different network topology that would propose regions automatically as part of convolution as seen in [13], using information about object semantics and what can physically exist, and using different trained networks on different sized objects for our large, medium, and small search regions.

4.3 Non-Object Correlation

While Section 4.2 dealt with correlated, spatial object correlations, this section will instead deal with Non-Target Pixels.

Note: Section 4.3 is mostly the submitted for publication work of Turner et. al in 2019 BMVC in the paper ***Unknown Target Classification: A Contextual Classifier Study***. This work was supported by the **Knexus Research Corporation.**

4.3.1 Introduction

Convolutional Neural Networks (CNN) have been the leading algorithm for creating of feature embeddings for object classification since 2012. Due to nearly a decade of improvements in deep neural network algorithms, Graphics Processing Unit (GPU) hardware, and megalithic sources of high quality human annotated data, these networks are now able to not only classify, but also calculate detection box locations of target objects in an image. They can even calculate pixelwise masks of these semantic segmentations with superhuman accuracy [70] [71] [72].

A weakness of these systems is their inability to use information beyond the target pixels. In this work, we aim to use context surrounding target objects to boost our classification accuracy, and gain insight into ways that we can exploit these non-target Pixels (NTP) in future networks.

Convolutional Neural Networks (CNNs) are becoming synonymous with computer vision. All-state-of-the-art classification and detection systems use these high powered algorithms. While the hardware, data, and algorithms used for this task has progressed greatly, the ways in which we use the data given have barely progressed since the first CNNs.

In 2012, Alex Krizhevsky's implementation [34] of a Multilayer Perceptron (MLP) geared towards computer vision in the form of a CNN won the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) by 10 and 15 percentage points (respectively) on classification and localization (respecitvely). CNNs cemented themselves as the new standard in image processing algorithms for discriminatory networks.

The ILSVRC ran for five years after *AlexNet* first won, with major advancements being made in 2013 with the Zeiler-Fergus model [35] introducing *deconvolutional networks*, in 2014 with the first inception network [36] introducing the tensor split-concatenate procedure *inception module*, and in 2015 with Microsoft Research introducing the Residual Network [17] using skip connections to reduce training difficulties caused by vanishing gradients.

Detection networks have advanced from their earliest forms of a sliding window approach [9] used on feature layers of the network [58] [10] [13] to the more modern single shot approaches [73] [74] [75] with algorithmic improvements for handling hard negative examples [76]. Although a handful of these networks have used context in the form of non-target pixels [17] [77], these works used context for boosting performance in classifying negative examples, In contrast, we consider an integrated system to study the effect of these NTPs on classification accuracy of objects.

The use of context in images is a field that has not garnered more than a small fraction of the research in the area. Bell et al. [39] use recurrent neural networks on multiple scales to scan the feature maps of the image for inter-object relationships to make further predictions on these relationships. This work leverages IRNNs [40]

which are identity matrix initialized Rectified Linear Unit (ReLU) recurrent transitions on the feature mapped output space of a fast-rcnn [10]. Although these were successful in raising the mAP of complex object detection systems, this system used target pixels from other objects in the scene (such that it would gain no advantage in a single target image), and used the stored feature map output from the network, so is not end to end trainable.

A different work using context for gains in detection accuracy is that of Turner et al. [2], where a VGG16 feature extractor [6] was used in conjunction with a Fast-RCNN detection net to compute object correlation probabilities in training, and to apply these known relations to modify the probability of detection at inference time for increased accuracy. In removing biases from the model added by using training data, the work builds correlative probability modifiers using a bias reducing region proposal technique [41]. Although increasing accuracy and average precision on the VOC2012 dataset, this technique did not leverage NTPs explicitly, and requires a large annotated dataset to generate the necessary object correlation matrices for detection.

Another interesting approach using recurrent neural networks in the field is that of Minh et al. at Deep Mind [42] on attention based networks. This work uses a separate recurrent neural network called a *Glimpse Network* together with a *Glimpse Sensor* to sample an image patch or *chip* at multiple scales in order to guide the network. Although this does focus on NTPs similar to our work, it is using the recurrent model to shift the attention of the network to new locations to complete part of a larger image, such as discovering the full sequence of numbers on

the Street View House Numbers (SVHN) or MNIST dataset.

The remainder of the Section is organized as follows: Section 4.3.2 shows implementations of context of NTPs, and demonstrates use cases, as well as providing a brief review on the convolutional architectures and algorithmic prerequisite to this work. Section 4.3.3 details the reasoning, methodology, and results of several theoretical and applied experiments regarding usage of NTPs for classification accuracy. Section 4.4 contains a concluding discussion of the work, and ideas and points to next steps for natural future extensions.

4.3.2 Computational Networks

Since the Imagenet challenge results in 2012 where Neural Networks took first place [34], CNNs have been recognized as the most accurate image/region classifier in computer vision. Classification in neural networks is done by passing the input image pixel matrix in through a variety of processing layers to produce feature maps. These processing layers are typically one of the following (Inception v3 [78] was used as the feature extractor for this work, and its layers are listed).

- *Convolutional Layer*- These rectangular (often square) layers serve as edge detectors from images and low level maps, and feature ensemblers from higher level maps. The shape of these filters dictates the dimensional depth of the output tensor.

- *Max Pooling*- These layers are helpful in size reduction of the feature maps, as well as suppressing irrelevant and noisy features from the maps.

- *Saturating Non-Linearity*- Non-linearities are needed to increase the expressive power of the networks beyond a simple single layer perceptron-type CNN. Because of vanishing gradient problems observed in deeper networks, our implementation leverages Rectified Linear Units (ReLU) [79].

- *Batch Normalization*- Batch normalization is used to increase stability in training deep neural networks, and reducing internal covariance shifts of the network [16].

The feature map outputs of these processing layers are generally given as 3-dimensional tensors ($w \times h \times c$), where strong activation in certain channels in the feature map corresponds to the presence of certain classes at position (x, y) in the image[1]. In the ultimate layers of the feature map, the tensor is flattened, and non-linear combinations of neuron activations in the feature map indicate which class labels are likely and unlikely.

Detection in CNNs is done in a multitude of ways, but they can all be seen as a variant of the original Regional Convolutional Neural Network R-CNN [9], where the features extracted from a subspace region of the original input space are used to serve as a prediction for that region. As enumerating every possible region in the input image for processing is exponential with respect to the input space, techniques for guided region selection were used [23] [41], and later replaced by networks that used automatic region proposal networks [13], or feature pyramid

[1]Because of convolution stride, and down sampling, and other position destructive operations in the network the feature map location is not the location on the input image.

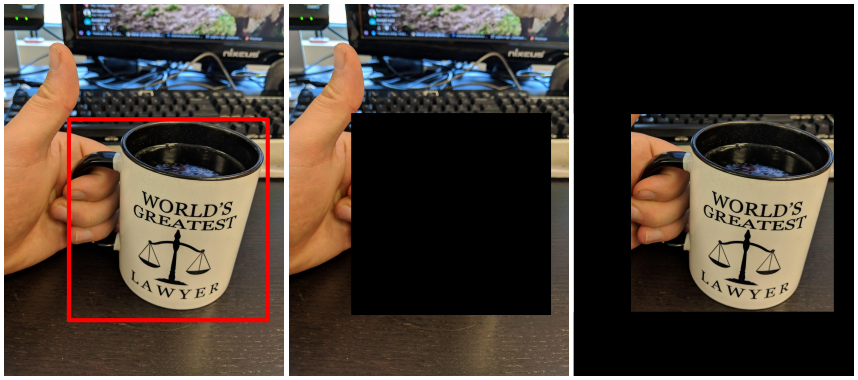


Figure 4.6: Three figures illustrating context, showing (left) a labeled coffee mug in a natural desk scene, (center) only the context of the coffee mug in the scene, and (right) only the coffee mug itself, with as much context as possible removed.

networks [75] to detect regions of interest automatically based on the *objectness* (quality of containing objects) of regions.

4.3.2.1 non-target Pixel Orderings

Here, we define **computational context** (differently from the normal operating systems definition of process data and resources) as the non-target parts of the image that are not the target pixels, as seen in Figure 4.6. In the figure, the coffee cup can be seen rather easily without the use of context that it is near a hand, a wooden desk, and a computer. Looking further into the image though, we see a grey long object that is not easily identified. By adding information (the grey object is sitting on a desk, the grey object is touching the bottom of the keyboard), we may be able to correctly deduce that this object is a gel wrist rest for a keyboard.

Signal v. Non Signal Given an input image I, and the set $\mathcal{D}$ of object class ground truths in the image set, and a rectangular bounding box, we will define set $\mathcal{T}(I)$ as the set of target pixels and $\mathcal{N}(I)$ as the set of non-target Pixels (NTPs). For each pixel value p_{ij} in the image space I, we apply the indicator function 4.3 to determine inclusion in $\mathcal{T}(I)$ where x_0 and y_1 represent the left and bottom boundaries respectively, and i represents the horizontal pixel location, while j represents the vertical pixel location:

$$\mathbf{1}_{\mathcal{T}} I = \begin{cases} 1 & \exists d \in \mathcal{D} : d_{x0} \leq i \leq d_{x1} \wedge d_{y0} \leq j \leq d_{y1} \\ 0 & \text{else} \end{cases} \tag{4.3}$$

While $\mathcal{T}(I)$ is well defined with multiple ground truth objects in the image, the definition of NTPs is more complicated in this scenario. Are the NTPs all of the area around the target object, regardless of whether it contains other target objects? If this is the case, then we need to define $|\mathcal{D}|$ sets of NTPs, and for $|\mathcal{D}| - 1$ of these sets, every pixel will belong to both the $\mathcal{T}(I)$ and $\mathcal{N}(I)$ sets.

For this work, we will assume $|\mathcal{D}| = 1$, so a subset of the data was taken such that only images with one object were used. Multiple object correlative contexts are explored in [2], and treating target pixels as NTPs is left for future work.

non-target Pixel Orderings Now that we have defined what a non-target pixel is, we define the three different orderings of NTPs: *Unordered regions* (uNTP), *Ordered regions* (oNTP), and *Absolute ordered pixels* (aNTP). For the uNTP and oNTP regions, a minimum region side parameter $\alpha = 70$ and a sampling rate pa-

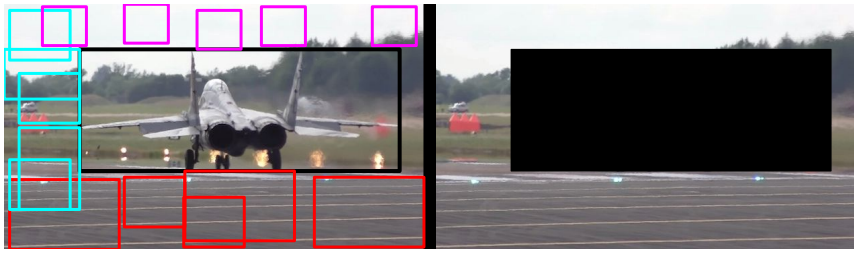


Figure 4.7: *Left*- Airplane class showing the uNTP and oNTP regions. Every region is a member of *airplane* in uNTP, while oNTP red regions are *airplane_south*, cyan is *airplane_lateral*, and magenta is *airplane_north*. This is shown with $\beta = 6$. The α parameter was set too high for small regions from the east, unlikely to contain information. *Right*- aNTP of the pixels surrounding the object.

rameter $\beta = 9$ were selected through a validation set, meaning that there must be at least 70 pixels between the edge of the image and the bounding box of the object on each side, and 9 random regions of a minimum size of 70 were taken from each side of the ground truth bounding box. Examples of all are given in Figure 4.7.

- *Unordered regions* (uNTP)- Unordered regions are random crops with a height and width minimum of α pixels such that no part of the region contains any target pixels. Any uNTP taken from an image with an object of class label c is also given label c.

- *Ordered regions* (oNTP) - Ordered regions are generated in the same manner as uNTP, except that they are given an additional label: *north*, *south*, or *lateral*. Lateral includes both east and west crops, as this has been shown to not affect the ability to comprehend natural images from data augmentation [34]. oNTP can be assigned one or two labels; it is assigned the label if over 50% of the

pixels are above or horizontally offset the object bounding box's y or x axis respectively. These regions surrounding an object of class c in direction d are labeled c_d.

- *Absolute ordered pixels* (aNTP) - Absolute ordering is the scenario where we leave the NTPs undisturbed and not partitioned, only censor out the target object pixels.

The usage of these regions vary by experiment, which will be detailed in Section 4.3.3.

Dataset and Model Creation A variety of experiments were run on this contextual problem, trying to gain insight into best practices for using context in classification and detection networks. All of the experiments used a subset of the PASCAL-parts VOC dataset [80], and 20 further datasets were created. For each of the twenty classes, a negative dataset was created for that class, where no instances of the class were seen in the training dataset, but were seen in the testing dataset. The networks were *not* pre-initialized on any larger dataset as to not give identifying information away implicitly (even though Imagenet may not contain the same $4,000$ *airplane* photos as VOC, the network would still have remembered filters of airplanes from seeing hundreds of thousands of them in initialization). For each of these 20 manufactured datasets, uNTP, aNTP, and oNTP regions were generated. To maximize the size of the test set, the unused training images from the dataset without the class were moved to testing. The 20 datasets have on average $5,717$ training images, and $5,823$ testing images, and are distributed near evenly.

An uninitialized inception-v3 [78] was used as the architecture of all of the networks trained. For each of the 20 datasets, we first trained a discriminative 20-way classifier on the aNTP regions of all of the training sets. Following convergence training on this dataset, the model is then trained on the $20-$way classification regions of uNTP. Following convergence on this dataset, the feature extractor is frozen with the fully connected layers and softmax layers still allowed to change and further trained on the $60-$way oNTP problem. For each of the datasets $c \in D$ a 19-way classifier was trained to be a discriminator of every class *except c*.

All experiments in Section 4.3.3 (except for the last counter-intutive introduction which will be explained at the time) are performed using combinations of the aNTP trained CNN (CNN_a), the uNTP CNN (CNN_u), the oNTP CNN (CNN_o), and one of the twenty CNNs $\{CNN_{\bar{c}} \forall c \in D\}$ where $CNN_{\bar{c}}$ is the CNN trained on every class *except c*.

4.3.3 Experimental Results

4.3.3.1 Theoretical Results

Effects of Ordering Our first experiment serves two purposes; first to show that NTPs have discriminative power greater than random chance, and second to show that more order leads to greater gains in accuracy. The results are shown in Table 4.6. For CNN_o, the cardinality of the class label was not considered in the classification (such that if the algorithm guessed *cat_south* when the ground truth class label was *cat _lateral* it was considered correct.

	Average	Recall	Precision
$CNN_u, \beta = 9$	28.59	18.62	26.38
CNN_a	38.73	26.39	30.14
$CNN_o, \beta = 9$	28.73	18.69	26.44

Table 4.6: Effects of ordering of regions on accuracy/PR metrics

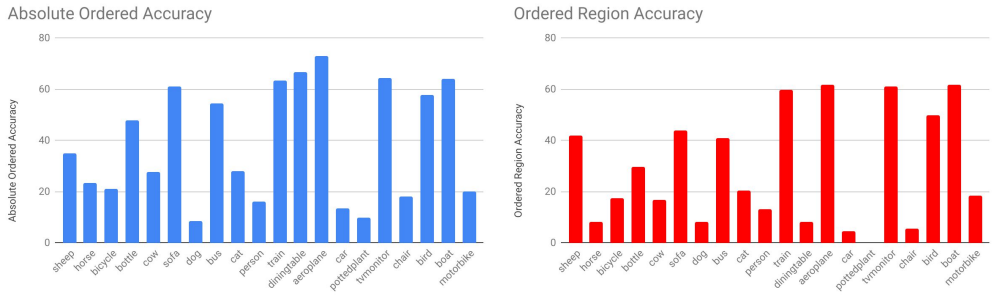


Figure 4.8: *Left-* Accuracy of prediction of censored class fromm CNN_a. *Right-* Accuracy of prediction of ordered regions of target objects using CNN_o.

These results support both claims; by using NTP, we get between a 5 to 8 times improvement in overall accuracy compared to random assignment (5% for 20 classes). Because of the increase in performance in accuracy, as well as precision and recall, CNN_o was used instead of CNN_u for the remainder of the work.

Gain by Class A second experiment run was using CNN_a and CNN_o on the 20 class absolute ordered NTP and the 20 class ordered NTP classification problem respecitvely [2]. For all 20 of the classes in the dataset, the algorithms made clas-

[2]There are 60 class ordered NTP, but only the target object not cardinality (North, South, Lateral) was considered, so 20-way classification.

sifications on the surrounding areas of the target pixels *using only the context and none of the target pixels.* The results are shown in Figure 4.8.

As we expected from our results in the previous Section **Effects of Ordering**, it is clear that the absolute ordered regions provide more contextual information than the ordered region pixels did (as shown before in Table 4.6), but also that different classes benefit to varying degrees by the usage of context. We expect classes with a high contextual accuracy (objects where seeing what is around the target pixels may be very important for classification) to be those that are often found in a unique context, such as an airplane against a sky background (66.7% accuracy on absolute ordered pixels), or a boat in a marine/dock background (64.20% accuracy on absolute ordered pixels). On the other hand, we expect objects with a *low* contextual accuracy to have contextual regions with non-unique objects and textures, like a dog (which can be seen in a variety of amibguous contexts on carpet, hardwood, or grass like many other objects having the low accuracy of 8.66%), or a potted plant which can literally be used as a decoration in virtually any natural setting (having a low accuracy of 9.76%).

4.3.3.2 Practical Results

Now that we have insight into the theoretical underpinnings of computational context applied to natural images, we can explore some of the ways in which these contextual neural networks can be used in conjunction with traditional classification neural networks to improve accuracy and recall of target objects using NTPs. We

give a natural usage of such a network combination next, and discuss appropriate use cases and metrics to measure performance gains from computational context.

Contextual Boosting To use context to augment classification, we need to devise an algorithm to analyze the classification output from the CNN, and based on these findings determine whether to use contextual classifier CNN_o to aid the classification. Since the probability of a vector belonging to the distribution of observed vectors using energy models [81] is of the form:

$$p(x) = \frac{e^{-E(x)}}{\sum_x e^{-E(x)}},$$

using the raw feature output from the CNN is infeasible as this problem quickly becomes intractable for vectors of sufficient length to describe natural visual phenomena. Instead, we should consider the point in a validation set where the mean harmonic mean (F_1) of the recall and precision is maximized.

$$\begin{aligned} F_1 &= 2 \times \frac{precision \times recall}{precision + recall} \\ &= 2 \times \frac{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}} \end{aligned}$$

Since we are adding an additional ground truth answer in the testing set of the contextual CNN that the plain CNN does not have, we are more likely to reduce the number of false positive labels by correctly guessing the unseen class and thereby increasing our precision. As we can see in Figure 4.9, the precision is maximized at $\lambda = 0.16$ for contextual network usage, while recall does not increase at this

	acc Range	avg acc	rec Range	avg rec	prec Range	avg prec
$CNN_{\bar{c}}$	87.74-89.09	88.40	81.97-84.28	83.13	75.45-87.46	83.78
$CNN_{\bar{c}} + CNN_o$	87.85-90.44	88.95	81.77-84.35	83.07	75.19-91.32	85.46

Table 4.7: $CNN_{\bar{c}}$- Control algorithm. This network was trained to classify on 19 different classes, but is tested on 20. CNN_o- This system was a control network ($CNN_{\bar{c}}$) that utilized the ordered contextual network CNN_o for detections under a probability threshold of $\lambda = .16$.

	avg acc	avg rec	avg prec	Median acc	Median rec	Median prec
CNN	87.36	86.67	78.92	90.48	88.60	82.62
CNN_T	93.16	94.34	87.74	95.20	95.41	90.84

Table 4.8: Contradicting the dissertation in a short table at the end, bold strategy Cotton.

point. When the CNN is used with an abstention parameter (where it references the majority output of randomly sampled ordered regions given to CNN_o), we see that the CNN_o system outperforms the control system by nearly half a percentage point in mean accuracy, and almost two points in average precision in Table 4.7.

4.3.3.3 A Confounding Concluding Experiment

With all of the evidence provided above, it seems that it should be a foregone conclusion that adding contextual information to networks aids in decision making. Table 4.8 shows that is not the case. In this table, CNN was trained against the 20

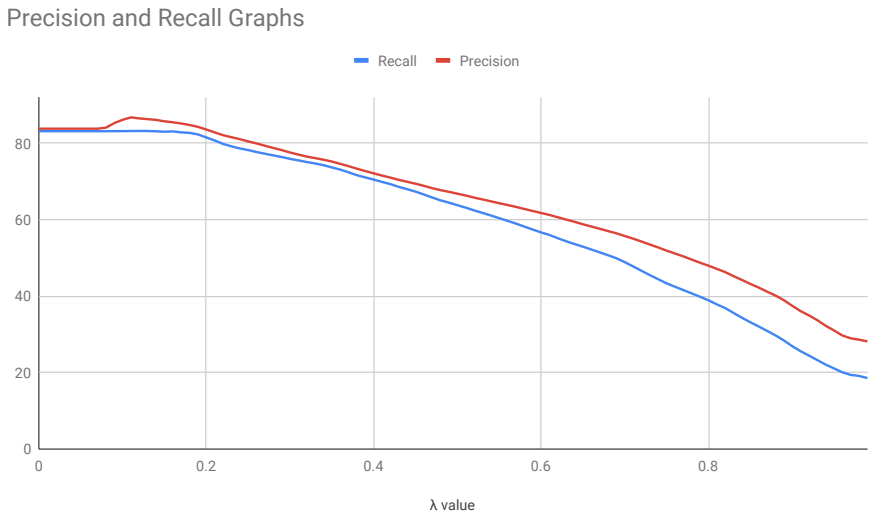


Figure 4.9: Accuracy of prediction of censored class fromm CNN_a.

class distribution of objects, while CNN_T was trained only on the cropped target pixels (i.e., no NTPs). If context adds power, then why does removing all context improve the average accuracy by nearly 5 points and the average precision by double that? We theorize this is because we were using context to add explainable power to an unexplainable target object; that adding known context could reveal unknown objects. When framed this way, Table 4.8 does not contradict our findings, but instead necesitates the usage of context, and looking beyond the target pixels when dealing with unknowns.

4.4 Contextual Power

In this section, we laid a theoretical groundwork for the usage of context in natural image scenes in dealing with unknowns. The main argument of the paper being that there are parts of the image (the NTP) that we are throwing out, because

they are not readily usable to increase the classification accuracy of our CNNs. And while we are *technically* correct in that sense, it's a very unsatisfying correctness.

We are moving past the age where we hand an image of a goat in a grassy field to a neural network and having it return the label "goat" and are impressed. While performance in this was made better and more accurate by adding wider filters [35], more layers in an inefficient way [6], more layers in a *very* clever way [17], and countless others we are still re-inventing the eyeball. Humans didn't become the dominant species on our planet by having the most sophisticated eyeballs that could see the most colors at the sharpest resolution (see *mantis shrimp*), we gnawed our way to the top *using* what we were seeing. By using context of what we were seeing, reasoning about what the signals we were observing meant. Knowing that if we see $100 laying on the ground, that how quickly and blindly we rush towards it should take the situation and context into account. Is it lying harmlessly near the television set? What if it's in the middle of a highway? What if it's under a grand piano attached to a rope and pulley, while the roadrunner holds on to the end of the rope?

Don't be the coyote. Use context.

Chapter 5: case-basedd Reasoning

> Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius and a lot of courage to move in the opposite direction.
>
> –E.F. Schumacher, *The Radical Humanist* 1973

Note: Section 5 is mostly the published work of the 2018 published ICCBR work ***Novel Object Discovery using Case-Based Reasoning*** *in Section 5.1 and Convolutional Neural Networks, and the submitted 2019 ICCBR work* ***NOD-CC: A Hybrid CBR-CNN Architecture for Novel Object Discovery*** *in Section 5.2.* Both works were supported by Knexus Research Corporation.

Cased based reasoning (CBR) allows us to apply the high powered algorithms that we have available at our disposal without enormous amounts of data. Having powerful classifiers and detectors is important, but we can do much more by extending upon them with more traditional AI techniques such as CBR.

5.1 Novel Object Detection Algorithm

5.1.1 Introduction

The development of Convolutional Neural Networks (CNNs) has resulted in significant improvements to object classification and detection in image data. One of their primary benefits is that they learn image features rather than relying on hand-crafted features, thereby reducing the amount of knowledge engineering that must be performed. However, another form of knowledge engineering bias exists in how objects are labelled in images, thereby limiting CNNs to classifying the set of object types that have been predefined by a domain expert. We describe a case-based method for detecting novel object types using a combination of an images raw pixel values and detectable parts. Our approach works alongside existing CNN architectures, thereby leveraging the state-of-the-art performance of CNNs, and is able to detect novel classes using limited training instances. We evaluate our approach using an existing object detection dataset and provide evidence of our approach's ability to classify images even if the object in the image has not been previously encountered.

Although CNNs greatly reduce the knowledge engineering required by removing the need for hand-crafted features, they do require knowledge about the types of objects that are present in the training images (i.e., an annotation of the object labels). This adds significant bias based on the types of objects that are used to annotate images. For example, an image of an office typically contains dozens of

visible objects but may only have labels for a small subset of those (e.g., humans, computers, desks) and treat the others as unlabeled background (e.g., books, pencils, papers). Thus, the CNN is only able to learn to classify objects that the domain expert felt were important enough to annotate. Similarly, the level of granularity of annotations can impact what a CNN learns. For example, the CNN will learn differently depending on if an image of a dog is labeled as *animal*, *dog*, or as the specific dog breed. These issues can become more significant when you have large datasets containing thousands or millions of annotated images, since it reduces the likelihood that a consistent annotation methodology was used on all images (e.g., different annotators, human error, timevarying methods of annotation). The annotated object types in training images restrict the potential classifications that a CNN can make when deployed; if an object type is not annotated in the training data, the CNN will be unable to classify that object type. For example, if a CNN is trained with images of *airplanes, boats, and houses,* an image of a *dog* would either be classified as one of those three object classes or not classified at all (i.e., if the confidence was too low).

We propose a case-based approach for novel object detection that uses a combination of raw pixel values and detectable object part information to identify when input images differ noticeably from known object types. Our approach is intended to be used in combination with existing CNN vision approaches and leverage their state-of-the-art performance while addressing some of their limitations. More specifically, our approach makes the following contributions: (1) a method to detect novel object types without prior knowledge of those types; (2) a method to identify vari-

ations in images of objects of the same type; (3) an approach that can be used in combination with existing CNN architectures; and (4) an approach that can be used even with small datasets and a single example of each object type. We believe the ability to operate using a small dataset is important given the large dataset requirements that are typically required by existing Deep Learning systems.

The remainder of the Chapter outlines our case-based novel object detection approach. Section 5.1.2 describes our method for novel object detection and how we leverage CNNs for this task. Section 5.1.3 describes our empirical evaluation using an existing object detection dataset. Section 5.1.4 discusses areas of future work and concluding remarks.

5.1.2 Cased Based Reasoning

Convolutional Neutral Networks perform supervised machine learning, so their ability to classify the presence of objects in images is directly related to the labeled training data they have available; they cannot detect the correct object type if no annotated training data exists with a label for that object type. If a CNN outputs the confidence in each known class label (i.e., the output of the fully-connected layers), it could, at best, label an input image as *unknown* if none of the possible class labels were above a confidence threshold. For example, if a CNN was trained to classify *airplanes, boats, and houses,* an image of a *dog* would either be classified as one of the three known classes (i.e., if the CNN output a high confidence for one of the classes) or as unknown (i.e., if none of the classes had a high confidence). If

several different novel objects are encountered, they would all be classified together into the generic *unknown* class, even if the objects were significantly different from each other. Returning to the example, images of *dogs, books, space stations*, and *humans* would all be classified together as unknown. One solution would be to retrain the CNN after each novel object type is detected. However, this is generally impractical as CNNs require both a large number of labeled training examples (i.e., more than a single training instance) and significant computational time to retrain the fully-connected layers.

We propose a case-based reasoning approach to detect the presence of novel object types and quickly learn from limited training data. Unlike CNNs, a CBR approach can learn using only a single training example and requires no training time. However, our approach does not propose to remove CNNs from the object classification process. Instead, our approach leverages the state-of-the-art performance of CNNs while providing capabilities that alleviate some of their limitations. For the remainder of this section we will largely present the CBR component in isolation, but will discuss how we integrate with existing CNN architectures at the end of this chapter.

Our CBR system encodes each image I_i as a case C_i. Each case is a triple containing the image's feature vector F_i, its set of observable parts P_i, and object label l_i:

$$C_I = \langle F_i, P_i, l_i \rangle \tag{5.1}$$

This representation assumes the availability of two functions: features and parts. The features function converts a raw image $I_i \in I$, where I is the set of all images, into a feature vector $F_i = \langle f_i^1, ..., f_i^n \rangle \in F$, where F is the set of all feature vectors (*features:* $I \rightarrow F$), composed of n feature values. For the *features* function, we use the convolutional and pooling layers from a CNN to perform this conversion, since they convert a raw image into a flat feature vector. This is essentially a version of the CNN with the fully-connected layers removed such that the CNN is only used for feature extraction. The *parts* function extracts a set of observable parts $P_i = \{p_i^1, ..., p_i^{m_i}\} \subseteq P$, where P is the set of all object parts, from image I_i. The number of observable parts in an image m_i is not fixed, so the size and contents of p_i is image-specific. In this section, we consider object parts to be low-level components that make up larger objects. For example, the parts of a dog could include its *legs, tail, torso, head,* and *ears*. Although the object parts provide more detail about an object, they are assumed to be generic such that the same parts can be part of numerous object types. Returning to the *dog* example, many mammals would share some or all of the same parts. However, even two instances of the same object may have different observable parts depending on what is visible in the image. In the dog example, the *dog*'s tail may not be visible depending on where it is facing or its legs may not be visible if the bottom of its body is occluded by another object. The *parts* function requires a separate vision system that can identify these generic object parts from visible images. However, as we will discuss later, while our CBR approach can leverage parts information, it is not strictly necessary for case retrieval. For example, if no parts extraction was possible, each case could contain

an empty set of parts ($P_i = \emptyset$) and rely only on the feature vector for retrieval. We assume each case has a single object label $l_i \in L$, where L is the set of all object labels. This assumes that each image will contain only a single object of interest. Such an assumption is valid for uncluttered images or, more realistically, when used as part of a Region-Based Convolutional Neural Network (R-CNN) [9]. R-CNNs use a region proposal stage to propose subregions of the input image and then classify those subregions individually. Thus, instead of the entire image being used as input to the CNN, each subregion is used as a distinct input to the CNN (i.e., the CNN is run multiple times) and each subregion is used to perform a single classification. In our work, the images stored in cases and used as input to the CBR system could be the image data from these subregions. Using this case representation, the feature vector and set of parts represent the *problem* and the object label is the *solution*.

When an input image is received, either a complete image or a proposed subregion from an R-CNN, object classification is performed using Algorithm 4. In addition to the input image I_{in}, the algorithm uses as input a case-based CB, number of nearest neighbors k, feature vector similarity threshold λ_f, and parts similarity threshold λ_p. A description of Algorithm 4 is given here: the algorithm starts by extracting the features and parts from the image (Line 1). If the case-based is empty (i.e., the CBR system has no training instances), a novel object label is generated using the *generateLabel* function (Line 2). We do not expect this function to generate an informative label based on knowledge of the image (e.g., *dog, cat, airplane, house*) but instead a unique label for the object type (e.g., *class1, class2, class3*). If the case-based is not empty, the top k most similar cases are retrieved from the

case-based (Line 4). The similarity only considers the feature vector similarity (e.g., using a similarity function based on the Euclidean distance between feature vectors), so no parts information is considered. Cases are only added to the top k if their similarity is above the feature vector similarity threshold λ_f, so it is possible for fewer than k cases to be retrieved. In some situations, no cases will be retrieved if none of the cases are similar to the input image (Line 5). In such a situation, the input image is assumed to be of a novel object type so a new class label is created for it (Line 6). The previous stages of the algorithm only considered the feature vectors when comparing the input image to cases. The remainder of the algorithm leverages the detectable parts information. The parts of the input image are compared to the parts of each of the top k nearest neighbors (Lines 7-12). The similarity function used (Line 10) is assumed to be a similarity function that calculates set similarity (e.g., Jaccard similarity). Similar to when comparing feature vector similarity, only cases with a parts similarity above the parts similarity threshold λ_p are retained (Line 11). If there were no cases above this threshold (Line 16), the input image is considered to be a novel object type so a novel label is generated. Otherwise, the label from the most similar case (based on parts similarity, with feature vector similarity used as a tiebreaker). Finally, a novel case is created and added to the case-based (Line 14) and the object label is returned (Line 15).

An existing label is only returned when there is a case that is similar to both the input images feature vector and its parts set. Thus, there are three situations where a novel object label, and therefore a new object class, are created: (1) when the case-based is empty; (2) when none of the cases have similar feature similarity;

Function: *classify*$(I_{in}, CB, k, \lambda_f, \lambda_p)$ ***returns*** l_{in}

---

1 $F_{in} \leftarrow features(I_{in}), P_{in} \leftarrow parts(I_{in}), l_{in} = \emptyset$;

2 $l_{in} = (CB = \emptyset)?generateLabel() : \text{None}$;

3 **else**

4 $topK \leftarrow retrieveTopK(F_{in}, CB, k, \lambda_f)$;

5 **if** $topK = \emptyset$ **then**

6 $l_{in} \leftarrow generateLabel()$;

end

7 **else**

8 $nn = \emptyset; nnSim = -1$;

9 **foreach** $C_i \in topK$ **do**

10 $sim \leftarrow partSim(P_{in}, C_i.P_i)$;

11 **if** $sim > nnSim$ **and** $sim > \lambda_p$ **then**

12 $nn = C_i; nnSim = sim$;

end

end

13 $l_{in} = (nn = \emptyset)?generateLabel() : nn.l_i$;

end

end

14 $CB \leftarrow CB \cup \langle F_{in}, P_{in}, l_{in} \rangle$;

15 **return** l_{in};

Algorithm 4: Object Classification using image features and parts

and (3) when there is at least one case with similar features but none of those cases have similar parts. As we mentioned earlier, although parts information is used in the algorithm, it is not strictly necessary. Assuming no parts information is available, the parts set of the input image and all cases will be empty. If the parts similarity function is designed to return maximal similarity when comparing two empty sets, all of the top k cases will be above λ_p and have an equal similarity value. Thus, as long as the top k cases are iterated over in order of descending feature vector similarity (Lines 9-12), the case with the most similar feature vector similarity will be selected as the nearest neighbor and have its label returned.

One of the primary benefits of this algorithm is that it is able to learn using only a single training instance. Once a novel class has been detected (Lines 2, 6, or 13), it is immediately added to the case-based and can be used to classify future input images. Similarly, this algorithm can be used even when no existing training data exists (i.e., an initially empty case-based). For example, this algorithm could be used from a cold-start to perform object classification without any labeled data. At such a time when sufficient data was collected and annotated, and sufficient time was available, a Convolutional Neural Network could be trained. Once a CNN is trained, the CBR algorithm could run in parallel to the CNN. Assuming the fully trained CNN has superior performance classifying known object types, the CBR system could defer classification for known object types and only interject when a novel class is detected or an input image is most similar to an object class that the CNN has not been trained on (i.e., a previously detected novel class). Thus, the CBR system can be used in situations where it has advantages over the CNN, and

defer in other situations.

5.1.3 Experimental Results

In this section, we evaluate the claim that our *case-based reasoning system can be used to detect and learn from novel object types.* Our evaluation tests the following hypotheses:

H1: Extracting a feature vector representation from images, using a CNN, provides sufficient information for a CBR algorithm to differentiate between object types.

H2: The addition of observable parts information improves object classification performance.

H3: Our CBR approach is able to detect novel object classes and learn from detected classes.

H4: Our CBR approach discovers finer-grained object classes than those provided by the datasets human annotators.

5.1.3.1 Data Set

The dataset we use for evaluation is the publicly available PASCAL-Part Dataset [80]. It is based on the dataset used for the *Visual Object Classes Challenge 2010*, a Computer Vision competition to recognize objects in realistic scenes. While the *Visual Object Classes Challenge 2010* dataset only contains the annotated

object types visible in each image, the PASCAL-Part Dataset contains additional annotations for the object parts that are visible in the image. The dataset contains 20 object types: *aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train*, and *tvmonitor*. Each object can have between 0 (*boat, chair, diningtable, sofa*) and 24 (*person*) object parts annotated. However, images of the same object type may have a different number of annotated parts due to object occlusion, object positioning, or annotator error. In addition to providing object part annotations, the *PASCAL-Part Dataset* has several properties that make it a suitable dataset for us to use. The images are realistic real-world images, so most images contain multiple objects (including objects from the 20 annotated object types as well as other unlabeled object types). The objects have varying locations, rotations, sizes, and scales. Additionally, images have different backgrounds (e.g., beach, indoors, forest) and lighting conditions. The annotated objects may be partially occluded, located partially outside the image, or incorrectly labeled by human annotators.

Our work is focused on detecting a single object type in each image, as we justified in the previous section, so we preprocessed the *PASCAL-Part Dataset* to extract only the images with a single annotated object. However, it should be noted that although each image only contains a single annotated object, many of them contain multiple visible objects. The additional objects are either objects that are not of the 20 labelled object types, or objects that have been omitted due to annotator error. After preprocessing, 4737 images remained (from an initial dataset size of 10,103).

The *features* function used in Algorithm 4 is a Convolutional Neural Network using the ResNet [17] architecture (i.e., how the various layers are connected). The CNN was pre-trained using the *ImageNet* dataset [7], a dataset containing hundreds of thousands of annotated images. This was performed to learn the filters (i.e., the image features) used by the convolutional layers of the CNN, and after training the fully-connected layers were removed. The output of the CNN is a feature vector of length 2048. Although *ImageNet* is a different dataset than the *PASCAL-Part Dataset*, pretraining a CNN on *ImageNet* learns many general-purpose image features (e.g., lines and shapes). Thus, it allows training a generic features function that can be used regardless of domain, and with significantly less time and computational effort than retraining the CNN for each new image dataset. However, it should be noted that due to the size and scope of *ImageNet*, there is likely some overlap with the objects contained in the *PASCAL-Part Dataset* (but none of the labels from *ImageNet* are used during our evaluation). The parts function in Algorithm 4 uses the ground-truth parts annotations provided by the dataset (i.e., assumes the presence of a perfect parts extractor). Although in real computer vision tasks the parts would need to be extracted using a separate vision system, we used the provided parts labels in order to remove error during our initial evaluations. Future work will examine how our CBR systems performance is influenced when parts are extracted using a more realistic *parts* function. Thus, each image in the preprocessed *PASCAL-Part Dataset* can be converted into our case representation using the *features* function, *parts* function, and object type annotation.

5.1.3.2 Classification Accuracy

Our initial set of experiments aims to evaluate the ability of our CBR algorithm to correctly classify the objects contained in images. More specifically, we examine the classification performance based on what information is used during case retrieval: *feature vector only, parts only,* or *both feature vector and parts.* Essentially, these experiments look to confirm that CBR can reasonably discriminate between the various object types and that reasonable data is contained in cases.

In the experiments, we use a variation of Algorithm 4 that does not attempt to identify novel classes; the label from the nearest neighbor is used even if that neighbor is dissimilar. This is achieved by using a non-empty case-based (avoiding Algorithm 4, Line 2), and setting $\lambda_f = \lambda_p = 0.0$ (avoiding Algorithm 4, Lines 7 and 14). The experiments used leave-one-out testing, such that each of the 4737 cases are used as input with the remaining 4736 cases used as the case-based. The accuracy is measured as the percentage of input cases that have a retrieved object type that is identical to their true object type (i.e., the solution portion of the case). The three variants we test are:

- **Feature Vector Only** We used $k = 15$ and an empty parts set for all images, thereby only basing similarity on the feature vectors. In practice, this is identical to using $k = 1$ since the case with the highest feature vector similarity will be selected given that there is no influence from parts similarity (i.e., all cases have empty parts sets). We used $k = 15$ to highlight that the various experiments were using similar parameter values.

- **Parts Only** We used $k = 4736$ so that the entire case-based was retrieved, regardless of feature vector similarity. All cases contained parts information. Thus, the most similar case is the case with the most similar parts.

- **Both Feature Vector and Parts** We used $k = 15$ and all cases contained parts information. Thus, the most similar case is the case with the most similar set of parts from amongst its 15-nearest neighbors (based on feature vector similarity).

Using only a single component of the case for retrieval resulted in lower performance, with a classification accuracy of 80.14% when only the feature vector is used and 88.79% when only the parts are used. The best performance was achieved when both the feature vector and parts were used for retrieval, with a classification accuracy of 91.13%. These results demonstrate that using CBR with only the feature vector provides reasonable classification performance (giving support for **H1**) but that performance can be increased by using both the feature vector and parts information (giving support for **H2**).

5.1.3.3 Novel Class Detection

The results in the previous subsection demonstrate the ability of our approach to be used to classify known objects in images. However, the primary motivation of our work is to detect and learn from novel object types in images. In this experiment, we use Algorithm 4 such that it can detect novel object types (i.e., the case-based may be initially empty, or either λ_f or λ_p are non-zero values). The experiment

starts with an empty case-based, and cases are randomly removed from the dataset and used as input to Algorithm 4. After each input, the algorithm stores a case in its case-based using the object classification it made for the image (i.e., Algorithm 4, Line 14). Thus, 4737 total inputs are provided to the algorithm, and after the n^{th} input the algorithm will have a case-based of size n. The evaluation was designed to simulate how the CBR system would start with no knowledge (i.e., an empty case-based) and incrementally learn based on its novel object detection capabilities. The parameters used are $k = 15, \lambda_f = 0.45, \lambda_p = 0.45$. The thresholds were selected to be relatively low such that they only exclude cases if they are significantly different than the input image. Similarly, the k value was selected such that a neighborhood of similar cases would be retrieved.

We use two metrics to evaluate the algorithms performance: *Class Purity* and *Class Count Divergence*. *Class Purity* measures, after all 4737 input images have been classified and added to the case-based, the percentage of images that are placed in a class where they share a true object type with the majority of other images in that class. Since our algorithm starts with no training data, all classes in the case-based are novel classes learned by the algorithm. Thus, we compare whether images with the same algorithm-generated object type classification have the same ground-truth object type classification. For example, the algorithm would be performing well if all images of *dogs* were given the same novel object classification of *class10* (or any other class label, as long as all *dogs* were given the same label). This metric calculates values between 0 and 1 (inclusive), with higher values being better.

Class Count Divergence measures how close the number of detected object

types is to the true number of objects types in the dataset. In our dataset, there are 20 true object labels. The motivation for using the metric is to penalize creating an unnecessarily large number of classes. For example, creating 4737 unique class labels would result in a perfect *Class Purity* score but each object label would be overfit to a single image. We use a curved function that has the maximal value when the number of predicted classes $class_{pred}$ is equal to the true number of classes $class_{true}$ and decreases as those values diverge:

$$\text{Class Count Divergence} = \frac{1}{(\frac{class_{true} - class_{pred}}{500})^2 + 1} \tag{5.2}$$

Similar to *Class Purity*, *Class Count Divergence* calculates values between 0 and 1 (inclusive), with higher values being better. The value 500 was selected for use in the Class Count Divergence based on the size of the dataset, such that the metric would be below 0.50 if the number of detected classes was larger than approximately 10% of the dataset size. Additionally, we report the Overall Performance of the algorithm as the harmonic mean of *Class Purity* and *Class Count Divergence*.

We repeated the experiment 25 times, and Table 5.1 shows a summary of the results. Based on the *Class Purity*, our approach does a reasonable job detecting novel object types and using those to classify images it encounters in the future. As a baseline, when input images are randomly assigned to 20 object types (i.e., no novel classes are learned), the *Class Purity* is 0.168. The majority of the mistakes made by the algorithm were to provide the same label to objects that are both physically similar and have similar parts. For example, many of the four-legged animals were

given the same label, especially in situations where they were small or occluded. Overall, occlusion had a significant impact on performance since it often resulted in very little of the object being visible (i.e., $< 10\%$) and no parts information being available. Even for humans, it was difficult to know that these highly obscured objects were the objects of interest. In fact, it was often the situation that unlabeled objects (i.e., not among the 20 annotated labels) were the most prevalent objects in images. By examining the learned class labels, we found that our algorithm was learning based on these unlabeled object types. However, given that the *Class Purity* metric only considers the 20 annotated object types, the metric is unable to quantify how well the algorithm was able to learn object types that were not annotated in the dataset. Overall, these results provide support for **H3**.

Table 5.1: Results of novel object type detection over 25 experimental runs

Metric	**Mean**	**Minimum**	**Maximum**	**Standard Deviation**
Class Purity	0.676	0.572	0.738	0.052
Object Types	121.7	111	133	6.2
Class Count Divergence	0.960	0.951	0.968	0.005
Overall Performance	0.792	0.717	0.838	0.036

5.1.3.4 Number of Object Types

The results in Table 5.1 show that our algorithm is learning approximately six times as many object types as are labeled in the dataset. This is reasonable per-

formance, considering that it would have created 4737 object types had each image been assigned its own label, but higher than anticipated. However, our qualitative examination of the classifications uncovered that the number of object types is not exclusively a result of algorithm error but primarily a result of learning finer-grained object types. For example, images annotated as *pottedplant* are largely divided by our algorithm into two distinct classes: one for images of *fully-grown plants* and one for *seedling plants*. To a human, there are clear and obvious distinctions between these two subsets of images, providing support that the algorithm learned a meaningful subdivision. Numerous other similar examples were found where the algorithm learned meaningful finer-grained object types, a selection of which include: *full-sized cars* vs. *go-karts* (both annotated as *car*), *people in water* vs. *babies* vs. *athletes* (all annotated as *person*), and *locomotives* vs. *subway trains* vs. *empty train tracks* (all annotated as *train*). However, although a significant number of the additional object types learned by our algorithm appear to be meaningful object types, it also learned less meaningful single image object types. Although some of those singleton object types are uninteresting or redundant, it learned several interesting singleton object types based on unusual images in the dataset: a sheep standing in a bus shelter, a train car with a picture of a dinosaur painted on it, an alpaca (incorrectly annotated in the *PASCAL-Part Dataset* as *sheep*), and a Ferris wheel. However, there were also situations where our algorithm erroneously subdivided object types, or performed divisions that a human would not deem as necessary (i.e., too fine-grained). This qualitative analysis provides partial support for **H4**, but a more detailed analysis will be necessary to definitively prove that our

algorithm is identifying meaningful object sub-types.

5.1.4 Algorithmic Proof of Concept

This section described a method for detecting novel object types in images using a combination of case-based reasoning and Convolutional Neural Networks. Our approach leverages the automated feature learning and extraction provided by CNNs while taking advantage of CBRs ability to perform incremental learning with relatively few training instances. A set of nearest neighbors are initially retrieved based solely on similarity between extracted image features, with subsequent retrieval based on the similarity between observable object parts. Although our approach leverages observable object parts during case retrieval, it can be used even if such information is unavailable. Additionally, since CBR is an instance-based learner, it does not abstract the object parts contained in images, thereby allowing them to be directly used during similarity calculation. If a CNN was to include object part information it would likely learn an abstraction of what parts exist in a class. For example, it would learn what parts are generally observable in images of *dogs*, possibly losing valuable information necessary to detect uncommon images, like a dog with most of its observable parts obscured by a costume it is wearing.

Our evaluation was performed using realistic images from the publicly available *PASCAL-Part Dataset.* The initial results demonstrated the ability of a CBR system to classify images using CNN-extracted feature vectors, and the performance improvement provided by including object parts information during retrieval. We

also provided evidence of our algorithms ability to be used to detect novel object types. Even when the algorithm had an empty initial case-based and no background knowledge about object types, it was able to detect novel object types and use them to classify subsequent images. One important finding of these experiments was that the algorithm appeared to learn finer-grained object types than those provided by the human dataset annotators, based on an initial qualitative analysis.

Several areas of future work remain. First, while we briefly discussed how our approach could be used in parallel with a full CNN (i.e., including fully-connected layers), we have not provided a full methodology to integrate them. In this paper, we focused on learning without an existing dataset, so it would not be possible to train a CNN in such a situation. However, if a subset of existing object types are known and have sufficient data, a full CNN could be used to classify those known types while our approach could handle novel object type detection. Second, our approach learns a flat object type hierarchy. Future work will examine how novel object types can be compared to existing types to determine relationships (e.g., a fully-grown plant is similar to a *seedling plant*) or to provide explanations (e.g., *I think this is different than a fully-grown plant because it doesnt have any leaves*). Third, we used ground truth parts information, but future work will detect both parts and object types. Finally, we plan to integrate our work with existing R-CNN architectures to allow learning with images containing multiple annotated objects.

5.2 Hybrid CNN-CBR Architecture

Section 5.1 laid the framework for a CBR system to be used with a CNN, but fell short of putting the hybrid architecture to usage to alleviate some of the problems faced by deep learning. In this section we introduce a hybrid CBR-CNN architecture for novel object discovery, and show promising experimental results from its usage.

5.2.1 Introduction

For supervised learning techniques, the human effort required to acquire, encode, and label a sufficiently large dataset may add such a high cost that deploying the algorithms is infeasible. Even if a sufficient workforce exists to create such a dataset, the human annotators may differ in the quality, consistency, and level of granularity of their labels. Any impact this has on the overall dataset quality will ultimately impact the potential performance of an algorithm trained on it. This paper partially addresses this issue by providing an approach, called **NOD-CC**, for discovering novel object types in images using a combination of Convolutional Neural Networks (CNNs) and Case-Based Reasoning (CBR). The CNN component labels instances of known object types while deferring to the CBR component to identify and label novel, or poorly understood, object types. Thus, our approach leverages the state-of-the-art performance of CNNs in situations where sufficient high-quality training data exists, while minimizing its limitations in data-poor situations. We empirically evaluate our approach on a popular computer vision dataset and show

significant improvements to objects classification performance when full knowledge of potential class labels is not known in advance.

We propose a method, called *Novel Object Discovery Using Convolutional Neural Networks and Case-Based Reasoning (NOD-CC)*, for object discovery and classification in images that leverages the high-end performance of CNNs while reducing its reliance on large sources of pre-labeled training data. Instead, NOD-CC attempts to classify an input image using a trained CNN, but can dynamically switch to using a case-based classification approach if the CNN isn't confident in its prediction. The primary motivation of this approach is that while CNNs require a large collection of training images of each object type to learn successfully, a CBR system can be used to learn using as few as one training instance. Thus, the CBR component can be used to discover novel object types and provide classification of those types until such time as there are sufficient training examples to retrain the CNN.

In our previous work [82] (Section 5.1), we demonstrated how CBR can leverage the automated feature extraction capabilities of CNNs, and perform novel object discovery and classification. In that work, which we will refer to as *Novel Object Discovery using Case-Based Reasoning (NOD-CBR)*, the convolutional layers of a CNN (i.e., the CNN architecture excluding the fully-connected neural network layers) are used to convert input images into a feature vector representation. That feature vector representation, and optionally any detectable *object parts* that are visible, is used to retrieve similar cases and determine if an object of that type has been encountered previously. NOD-CC significantly extends NOD-CBR and provides the following key contributions:

- A hybrid architecture that includes both the NOD-CBR system as well as a fully functional CNN (i.e., a CNN that performs object classification rather than purely feature extraction).

- An architecture that provides both the high-end performance of CNNs as well as the lazy, data-poor learning capabilities of CBR.

- A series of decision algorithms that can dynamically select whether to use the CNN or CBR components of our architecture to perform object classification.

- An online method for object classification, novel object discovery, novel object labeling, and learning.

- An empirical evaluation that demonstrates the utility of NOD-CC when the full set of object types is not known in advance.

The remainder of this chapter describes how NOD-CC combines Convolutional Neural Networks and Case-Based Reasoning to classify images while also performing novel object discovery. Section 5.2.2 describes our hybrid architecture that combines CNNs and CBR for object classification and discovery. Our empirical evaluation is presented in Section 5.2.3, and provides evidence to support our claims of the utility of NOD-CC. Finally, in Section 5.2.5 we summarize our findings and identify important future research directions.

5.2.2 NOD-CC Architecture

Our approach, Novel Object Discovery Using Convolutional Neural Networks and Case-Based Reasoning (NOD-CC), is a hybrid of two learning and classification methods (Figure 5.1). The Convolutional Neural Network component (labeled as CNN) is intended to classify images of object types for which sufficient training instances are available. Additionally, it converts raw images into feature vectors for use by the Case-Based Reasoning component (labeled as CBR). The CBR component is intended to learn from and classify object types that are not classifiable by the CNN. A meta-algorithm, labeled as Controller, determines whether the classification from the CNN or CBR component is used to provide final image classification. In the following sub-sections, we will provide details about each of the three primary components: CNN, CBR, and Controller.

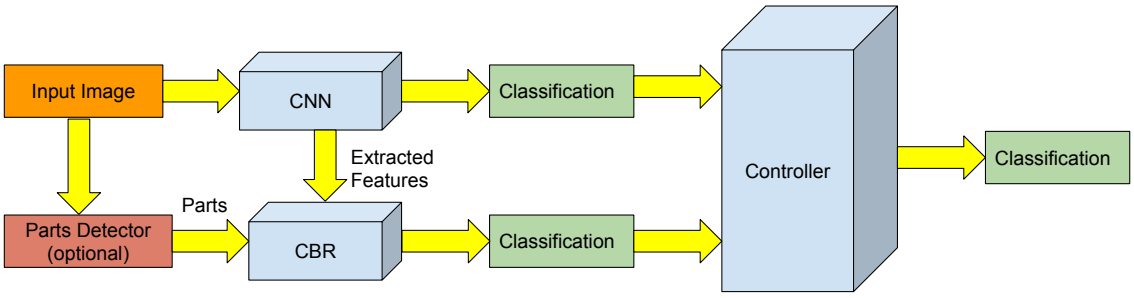


Figure 5.1: Architecture of the NOD-CC image classification system. The classifications are shown in green and are produced by the three decision algorithms shown in blue. The inputs to the decision algorithms are shown in yellow, the input image in orange, and the optional parts detector in red.

Although CNNs can achieve high accuracy when classifying objects in images,

their performance is dependent on the set of class labels (i.e., object types) contained in the training data. If the training data contains images labeled with the set of labels $\mathcal{L} = \{l_1, \ldots, l_n\}$, a CNN (and most other learning algorithms) will only be able to classify those n object types. Any images of objects with a label l_m (where $l_m \notin \mathcal{L}$) will either be misclassified as one of the labels in $\mathcal{L}$ or unclassified (i.e., the CNN will output a low confidence for all labels such that an *unknown* output is produced). This issue is particularly problematic for CNNs since they require a large set of example images labeled as l_m before they be accurately trained to predict that object type. CBR, on the other hand, likely does not have the same peak classification performance on massive image datasets but is capable of one-shot learning. Once a single image with label l_m is encountered, it can be stored as a case and reused to classify other instances of that object type.

For the CBR component of NOD-CC, we use our previous case-based novel object discovery approach, NOD-CBR [82]. NOD-CBR stores each training image $I_i \in \mathcal{I}$ (where $\mathcal{I}$ is the set of all images) as a case C_i in the case-based CB ($C_i \in CB$). Cases are encoded as triplets containing the feature vector representation of the image F_i, a set of detectable image parts P_i, and the ground truth object label l_i: $C_i = \langle F_i, P_i, l_i \rangle$. Using case-based reasoning nomenclature, the feature vector and parts set of the image are the *problem*, and the class label is the *solution*.

Recall from the previous subsection that the convolutional and pooling layers of the CNN component convert a raw input image into a feature vector $F_i = \langle f_1, \ldots, f_v \rangle \in \mathcal{F}$ (where v is an integer value defined by the CNN architecture and $\mathcal{F}$ is the set of all feature vectors). Thus, both the CBR component and the

fully-connected layers of the CNN component use an identical feature vector representation as produced by the convolutional and pooling layers mapping from images to features: $features : \mathcal{I} \rightarrow \mathcal{F}$.

Each case also contains the set of parts $P_i \subset \mathcal{P}$ that are detectable in the input image, where $\mathcal{P}$ is the set of all parts that may be detected. These parts are generic lower-level structures of an image, like *hands*, *feet*, *wheels*, or *wings*. Since parts are generic, different objects types can share parts (e.g., both *dogs* and *cats* have *legs*, *heads*, *ears*, *tails*). However, even images of the same object type may have different detectable parts based on variations in pose, occlusion, or photographic style. For example, in Figure 5.2, the cats do not have an identical set of detectable parts due to different poses and image framing. Our work assumes the presence of a parts extractor that returns the set of detected parts in an image: $parts : \mathcal{I} \rightarrow \mathcal{P}$. However, as we will discuss shortly, while our approach can leverage parts information it is not necessary for classification (i.e., it can classify using only the feature vector).

The NOD-CBR object discovery and classification algorithm is shown in Algorithm 4. While full details of the algorithm are described in our previous work [82], we will provide a brief overview of its reasoning process. Given an input image, the algorithm will extract the feature vector representation (i.e., from the CNN component) and the set of detectable parts (i.e., from the parts extractor). If the either the case-based is empty (Line 2), no cases are sufficiently similar to the input image's feature vector representation (Lines 4-6, based on a threshold λ_f), or there are cases with similar feature vectors but their detectable parts are not similar (Lines

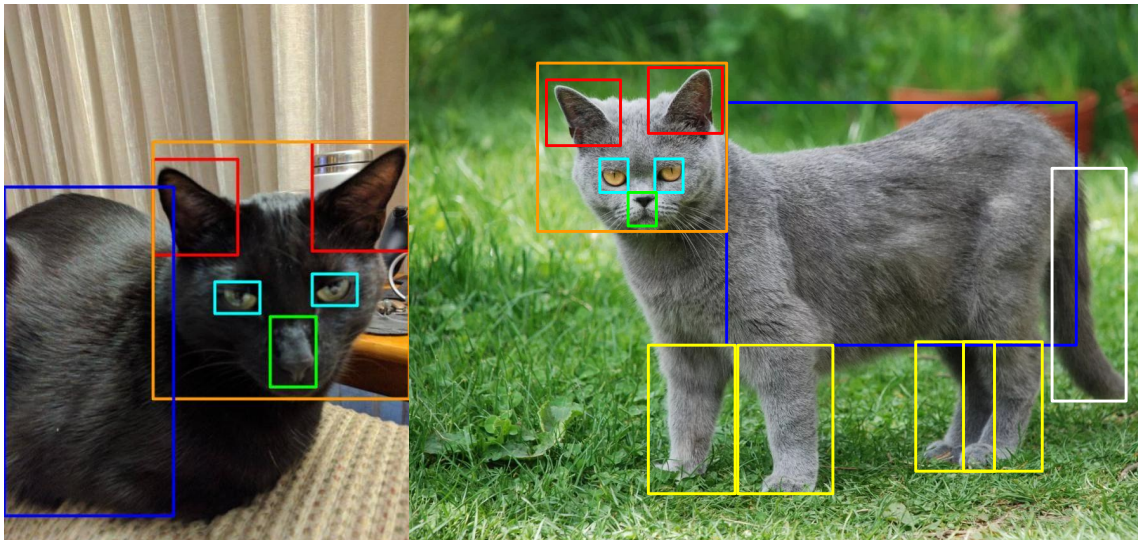


Figure 5.2: The variation in pose of the two cats, as well as the framing of the picture can drastically effect the observable parts. The cat on the left in the so-called *catloaf* position is hiding his legs under his torso, and the way the picture is framed does not show its tail, while the cat on the right has all major parts visible.

8-12, based on a threshold λ_p), then NOD-CBR generates a new label for the input image. In that situation, it believes the image to be of a newly discovered object type. Otherwise (Line 13), it uses the class label from the the most similar retrieved case. In all situations, a new case is retained and added to the case-based (Line 14).

An advantage of this approach is that it can start from a variety of initial case-based configurations: an empty initial case-based if no prior knowledge exists, a case-based containing cases for all images used to train the CNN, or a sampling of cases of each object type if the full training set is too large. It should also be noted that while the *generateLabel*() function in Algorithm 4 will generate a unique label for a newly discovered object type, it will likely not be a meaningful class label

(e.g., returning the label *object5849* rather than *lion*). However, images with newly generated labels (i.e., the newly discovered object types) could be presented to a human expert, either online or offline, to receive more meaningful object labels.

Controller Component The CNN component and CBR component both output a classification for the input image. However, there is no guarantee that they will predict the same object type. The role of the controller is to receive as input the predictions from both components and output a final predicted class label.

In our work, we use three different Controller strategies:

- **Always CNN:** The classification output by the CNN component is used regardless of the the CBR component's classification. This is equivalent to the CNN component operating in isolation.
- **Always CBR:** The classification output by the CBR component is used regardless of the the CNN component's classification. This is equivalent to the CBR component operating in isolation.
- **Conditional CBR:** The classification of the CNN component is used unless the CNN has low confidence in its prediction. This occurs when none of the class labels are above an abstention threshold λ_a. In situations where the CNN does not output a class label, the prediction of the CBR component is used.

5.2.3 Evaluation Standard

Our empirical evaluation demonstrates the image discovery and classification performance of NOD-CC when the complete set of object types that will be encountered at run-time is not known in advance. More specifically, the following hypotheses are evaluated:

H1: The CNN component will be unable to correctly classify any object types not present in the training set.

H2: The CBR component, NOD-CBR, will outperform the CNN component when the training images do not contain instances of all object types that may be encountered at run-time.

H3: NOD-CC will achieve higher classification performance than the CNN component alone when the training images do not contain instances of all object types that may be encountered at run-time.

H4: NOD-CC will achieve higher classification performance than NOD-CBR alone when the training images do not contain instances of all object types that may be encountered at run-time.

The dataset used is the same as in [82], and described in depth in Section 5.1.3.1.

Our current work is focused on classifying a single object type in each image. To facilitate this, we filtered the *PASCAL-Part* dataset to only the images that

contain a single class label, thereby reducing the dataset from 10,103 images to 4,737. While this may seem like a limitation of our approach, many computer vision applications first propose sub-regions of a cluttered image to classify (e.g., the *region proposal* stage of a Region-Based Convolutional Neural Network [9]), and then provide at most a single object label for each sub-region (i.e., a traditional CNN classification). Additionally, even though each image only contains a single labeled object, nearly all of the images contain a variety of unlabeled background objects. Since the size of the filtered *PASCAL-Part* dataset is quite small by Deep Learning standards, the CNN component used in our work, the Inception-v3 architecture, was pretrained on the much larger *Open Images v4* dataset [83] and then fine-tuned using the filtered *PASCAL-Part* dataset. It should be noted that there is no overlap between the images contained in the two datasets (i.e., pretraining on *Open Images v4* will not provide any images from the testing sets we use).

For our experiments, we used the filtered *PASCAL-Part* dataset to create 20 experimental datasets. The original dataset comes pre-partitioned into training and testing sets. For each of the 20 experimental datasets, 5 of the 20 object types were selected at random (such that no two experimental datasets used the same set of 5 object types). All images of the 5 selected object types were removed from the training set but left in the testing set. Thus, all testing sets contain images of all 20 object types, but the training sets only contained images of 15 object types. These experimental datasets were partitioned in advance, such that all experimental variations would work on an identical set of datasets.

Section 5.1 demonstrated the ability of NOD-CBR, when starting from an

empty case-based, to discover and classify classes. More specifically, we evaluated its ability to maximize class purity (i.e., provide the same generated label to images of the same object type) while minimizing the divergence in the number of discovered classes from the true number of classes (i.e., not over-partitioning the data). Given that we have previously demonstrated the efficacy of NOD-CBR on these tasks, our evaluation will measure the performance of our hybrid NOD-CC architecture's classification performance when class labels from the testing set are not present in the training set (i.e., novel object types are encountered at run-time).

For each testing image provided to NOD-CC, there are four possible ways in which the classification prediction of NOD-CC can align with the image's ground truth label, ordered from best to worst:

1. **Correct:** The class label predicted by NOD-CC matches the ground truth class label. This is the ideal situation and is considered to be a 100% match. C represents the percentage of training instances labeled correctly.

2. **Known Novel:** NOD-CC correctly predicts that the class label was not one of the class labels in its training set. Since a random guess would correctly predict a novel class 25% of the time (since 5 of 20 classes are not in the training set), we consider this to be a 25% match. KN represents the percentage of training instances labeled as known novel.

3. **Abstention:** NOD-CC does not have enough confidence in any of its potential predications, so it abstains from making a prediction. Since guessing a class label randomly would provide the correct prediction approximately 5%

of the time (since there are 20 classes), we consider an abstention to be a 5% match. Essentially, this prevents NOD-CC from being forced to provide a random guess to boost its accuracy and allows it to abstain when it is unsure. A represents the percentage of training instances that were abstained from labeling.

4. **Incorrect:** NOD-CC predicts a known class label (i.e., a class label present in the training set) but it does not match the ground truth class label. This is incorrect and considered to be a 0% match. I represents the percentage of training instances labeled incorrectly.

During each evaluation, each image in the testing dataset is used as input to NOD-CC and a comparison between the predicted class and ground truth class label is used to calculate our scoring metrics: accuracy (ρ_A), precision (ρ_P), recall (ρ_R), and F1 score (F_1). Although the precision, recall, and F1 score calculations use well-established equations, we use a modified accuracy function based on the previous discussions of the four ways NOD-CC's classification can align with the ground truth classification.

$$\rho_A = \frac{(C + .25 \times KN + .05 \times A) \times TP}{TP + FN} \qquad F_1 = 2\frac{\rho_P \times \rho_R}{\rho_P + \rho_R}$$

$$\rho_P = \frac{TP}{TP + FP} \qquad \rho_R = \frac{TP}{TP + FN}$$

For every class label in the dataset (all training classes unseen at training time are considered to be of a single class labeled as *Novel Class*), we compute the accuracy (ρ_A), precision (ρ_P), recall (ρ_R), and f-score (F_1). For each experimental

run (i.e., providing the testing instances from a single experimental dataset to Algorithm 4) the mean of each of the class-level metrics is computed. We further vary our experiments by randomizing the order in which testing instances are provided to Algorithm 4. This is important since it is a learning algorithm (i.e., new cases are stored) so the order of testing instances may impact performance. For each of the 20 experimental datasets, 20 random orderings were used. This resulted in 400 total experimental runs (20 datasets $\times$ 20 orderings) and the reported results are the averages of the metrics over all 400 runs.

5.2.4 Experimental Results

5.2.4.1 Always CNN Variant

As a baseline, we evaluated the **Always CNN** variant of NOD-CC (i.e., when the CBR component is ignored). The abstention parameter λ_a was determined through cross-validation on the entire dataset, such that the F_1 was maximized. Recall that the **Always CNN** variant is unable to learn online; it is only able to abstain from providing a label. Assuming a perfectly balanced set of classes, since the CNN is only trained on 15 classes with the remainder only appearing in the testing set, its maximum accuracy is bounded as: $max(\rho_A) = (\frac{15}{20} \times 100\%) + (\frac{5}{20} \times 5\%) = 76.3\%$. In reality, due to the imbalance of the datasets the true maximum accuracy was lower - 63.9% in our experiments. We report an additional metric, *Relative Mean Accuracy (RMA)*, that measures the fraction of $max(\rho_A)$ that was achieved. We also report the minimum (Min. ρ_A), maximum (Max. ρ_A), median

(Med. ρ_A) and standard deviation (σ ρ_A) of the accuracy (i.e., when examining each experimental run individually). The performance of **Always CNN** is shown in Table 5.2.

Table 5.2: Performance of the various NOD-CC configurations

Variant	ρ_A	ρ_P	ρ_R	F_1	RMA	Min. ρ_A	Max. ρ_A	Med. ρ_A	σ ρ_A
Always CNN	42.99	61.31	37.98	44.32	67.27	25.98	61.27	41.80	9.15
Always CBR w/ Parts	58.45	54.30	59.66	56.18	81.70	43.49	68.93	61.33	6.57
Always CBR w/o Parts	49.82	49.77	49.41	48.21	69.67	37.49	63.44	59.78	7.27
Conditional CBR w/ Parts	59.90	56.52	60.98	58.17	83.77	54.00	64.12	60.35	2.66
Conditional CBR w/o Parts	53.73	51.39	53.75	52.44	75.15	49.84	61.91	55.23	2.67

One item of note in these baseline results is that the precision is significantly higher than the recall. This is intuitive in a system that uses a threshold to determine confidence in classifications (i.e., λ_a); the system only provides classifications when it is confident in its predictions and thereby lowers the number of false positives. In these results, as expected, the **Always CNN** approach is never able to correctly

label unknown classes, providing evidence to support **H1**.

5.2.4.2 Always CBR Variant

As an additional control, we use the *Always CBR* variant of NOD-CC (i.e., the CNN always abstains, so only CBR is used). This variant was evaluated both with observable parts information (i.e., a parts detector component was available) and without. When parts are not available, Algorithm 4 only uses the image features during retrieval. For these experiments, the CBR component was initially given a case-based containing all training instances.

Always CBR has a higher theoretical maximum accuracy than **Always CNN** because it has the ability to label an image as a novel class rather than abstaining: $max(\rho_A) = (\frac{15}{20} \times 100\%) + (\frac{5}{20} \times 25\%) = 81.3\%$. Based on the class imbalance of the datasets, the true maximum accuracy was determined to be 71.5%. Similar to with *Always CNN*, this was used to calculate the RMA. The results are shown in Table 5.2.

Although the availability of detectable object part information is beneficial, **Always CBR** is able to outperform **Always CNN** even without parts. The only metric **Always CNN** performs better on is precision. As we mentioned previously, this is a result of the CNN algorithm being able to abstain, thereby lowering its false positive rate. Overall, the results demonstrate the benefits CBR can provide when the full set of object classes is not known in advance. Even considering the performance of these approaches relative to their maximum accuracy (i.e., RMA),

Always CBR still outperforms **Always CNN**. These results provide evidence to support **H2**.

5.2.4.3 Conditional CBR Variant

In this variant, we use both the CBR and CNN components (i.e., our full architecture). As described previously, the classification from the CNN is used unless the CNN abstains. If the CNN does abstain, the CBR component is used for classification. We use the same configurations (i.e., λ_a threshold and initial case-based) for the CNN and CBR components as described in the previous experiments. Similar to the **Always CBR** variant, we evaluate the **Conditional CBR** both with and without parts information. The results are shown in Table 5.2. Across all metrics, except precision, both variants of **Conditional CBR** outperform **Always CNN**. This demonstrates that the ability of CBR to dynamically detect and learn from previously unseen class types provides significant benefit to the CNN component. In situations where the CNN abstains, the CBR component is able to provide assistance. This provides support for **H3**.

When comparing **Always CBR** to **Conditional CBR**, the **Conditional CBR** variants outperform across all five core metrics (accuracy, precision, recall, f-score, and RMA). This includes both the variants that use parts information as well as those that do not. The results show that the **Conditional CBR** performance has fewer extreme results (i.e., minimums and maximums closer to the mean) and significantly lower standard deviation. This is beneficial because it provides both

improved performance as well as less uncertainty about the potential performance on an unknown dataset. Additionally, these results demonstrate the combination of both the CNN and CBR components are necessary for maximum performance; neither module is sufficient for novel object discovery on their own. These results provide support for **H4**.

5.2.5 CBR applications

In this section, I described NOD-CC, a hybrid architecture that uses Case-Based Reasoning and Convolutional Neural Networks to discover novel object types during the image classification process. NOD-CC leverages the automated feature extraction and image classification performance of CNNs while minimizing their requirement for large, pre-labeled training datasets by using CBR's instance-based learning capabilities. NOD-CC can be used with any CNN implementation so it is not tied to a specific CNN architecture, training methodology, or parameter selection. This is particularly important given the rapid advancement in the field of CNNs.

Additionally, NOD-CC can use detected object parts to further improve its performance, although it performs well even if such additional information is unavailable. We evaluated our approach on a publicly available image dataset and showed NOD-CC had improved performance over a CNN or CBR module alone. Our results demonstrated that NOD-CC was able to discover previously unknown classes of objects (i.e., not represented in training data), learn from a single instance

of the novel object type, and classify future instances of those objects. Additionally, NOD-CC performed these tasks without compromising the discriminatory power of the CNN.

Future work will involve using the WordNet hierarchy in conjunction with the hierarchical multi-class capabilities afforded by Inception-style architectures in order to perform hierarchical clustering of classes. Thus, a novel class could be placed in a hierarchy relative to known classes, possibly revealing a parent-child relationship. For example, if an image dataset contained labeled images of *balloons* and *baskets*, it could be learned that they are related to a newly discovered object type, an image of a *hot air balloon*. Similarly, textual relations between the known class labels could be used to generate a more semantically meaningful label for the novel object type (e.g., *balloon basket*). We also wish to investigate additional methods for using CBR for classification. Even in our dynamic approach described in this paper, we set an abstaining threshold λ_a for detection to be used unilaterally across all classes. There is an intuitive reason to believe that a CBR system (i.e., a meta-algorithm) for determining when to deploy a second CBR system (i.e., an image classifier) may be useful in this effort.

Chapter 6: Conclusion and Future Work

> The best minds of my generation are thinking about how to make people click ads...that sucks.
>
> –Jeff Hammerbacher, *Bloomberg Businessweek*, 2017.

This work is in no means a complete study of computational context and it's usages in computer vision; I can't even say that I know how much I don't know on this topic.

Possible future work (Figure 6.1) includes the following:

- Temporal Context- The usage of videos is an obvious extension, where based on what we had seen in the video before (and backwards if we are using a bidirectional system) could influence what we *think* we are seeing now. Similar to the SPARCNN work above, this could be leveraged to learn from our past experience in videos in order to predict future sequences. Imagine a system that was trained on horror films that began to recognize scenes where the killer is about to jump out before it happens, or a network trained on videos of hockey where the system can start predicting the puck in the net before it happens based on the sequences of movement it had seen before, and the

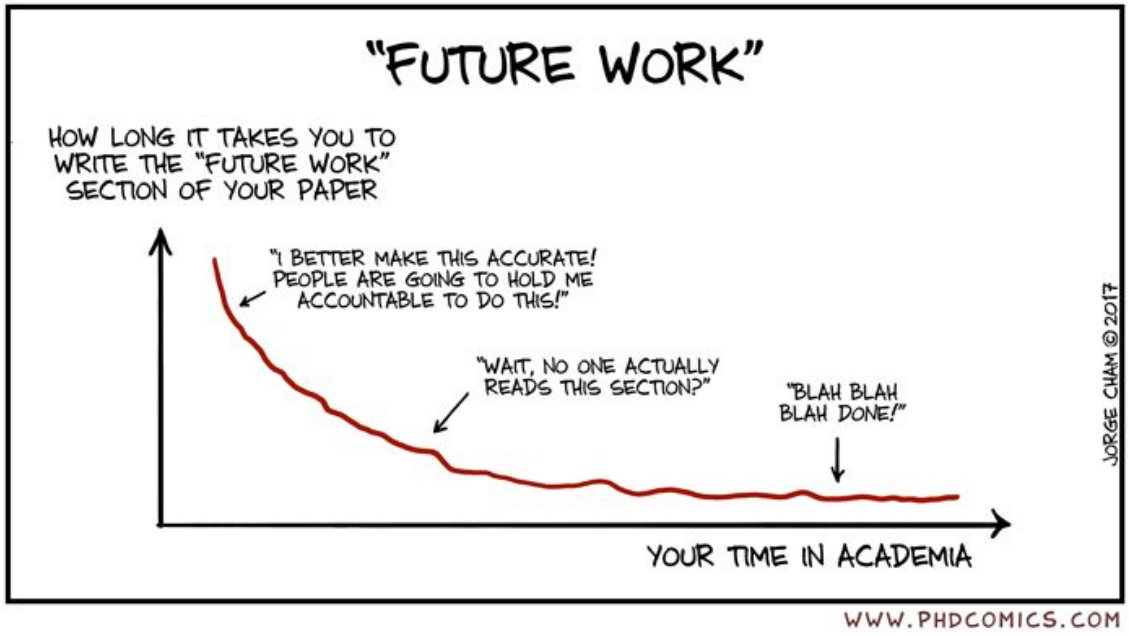


Figure 6.1: I've been in graduate school for 7 years, and I work in private industry. Credit to Jorge Cham at www.phdcomics.com.

position of players.

- Hierarchical Grouping- Using the novel object discovery algorithms of Chapter 5, it is possible to build relational networks of objects relationship to each other in the real world based upon the observable parts included in the objects as well as the feature vectors extracted from the networks.

- Multiple Object Contextual Scenes- Although the SPARCNN work dealt with object corelations, and the NTP work dealt with *non* object correlations, we are yet to fuse the two systems together into an integrated system that is able to use information from other objects in the image and background.

- Multiple Novel Object discoveries- In the novel object explorative domain, we

do not at the current time use more than one object to simplify the study, and focus on discovery based on the target pixels. In future work we can also leverage information from other objects present in the image (or even NTPs) to create more complex cases for a more interesting form of novel object discovery.

Appendix A: Appendix A: Pascal Parts Dataset Composition

Class Label: sheep (551 instances)

Part(65.34 %): muzzle, $\bar{x}, \sigma, min, max$: 0.65, 0.48, 0, 1

Part(85.84 %): head, $\bar{x}, \sigma, min, max$: 0.86, 0.35, 0, 1

Part(23.59 %): leg4, $\bar{x}, \sigma, min, max$: 0.71, 1.36, 0, 4

Part(8.89 %): horn, $\bar{x}, \sigma, min, max$: 0.15, 0.50, 0, 2

Part(27.95 %): tail, $\bar{x}, \sigma, min, max$: 0.28, 0.45, 0, 1

Part(91.83 %): torso, $\bar{x}, \sigma, min, max$: 0.92, 0.27, 0, 1

Class Label: bottle (776 instances)

Part(98.84 %): body, $\bar{x}, \sigma, min, max$: 0.99, 0.11, 0, 1

Part(86.47 %): cap, $\bar{x}, \sigma, min, max$: 0.86, 0.34, 0, 1

Class Label: horse (451 instances)

Part(99.11 %): torso, $\bar{x}, \sigma, min, max$: 0.99, 0.09, 0, 1

Part(94.68 %): head, $\bar{x}, \sigma, min, max$: 0.95, 0.22, 0, 1

Part(85.59 %): leg4, $\bar{x}, \sigma, min, max$: 2.84, 1.46, 0, 4

Part(56.10 %): tail, $\bar{x}, \sigma, min, max$: 0.56, 0.50, 0, 1

Class Label: bicycle (450 instances)

Part(97.11 %): wheel, $\bar{x}, \sigma, min, max$: 1.76, 0.49, 0, 2

Part(65.56 %): chainwheel, $\bar{x}, \sigma, min, max$: 0.66, 0.48, 0, 1

Part(89.33 %): handlebar, $\bar{x}, \sigma, min, max$: 0.89, 0.31, 0, 1

Part(66.44 %): saddle, $\bar{x}, \sigma, min, max$: 0.66, 0.47, 0, 1

Class Label: motorbike (457 instances)

Part(93.87 %): wheel, $\bar{x}, \sigma, min, max$: 1.61, 0.60, 0, 2

Part(44.42 %): headlight, $\bar{x}, \sigma, min, max$: 0.50, 0.62, 0, 3

Part(5.03 %): handlebar, $\bar{x}, \sigma, min, max$: 0.05, 0.22, 0, 1

Part(1.97 %): saddle, $\bar{x}, \sigma, min, max$: 0.02, 0.14, 0, 1

Class Label: cow (317 instances)

Part(70.98 %): muzzle, $\bar{x}, \sigma, min, max$: 0.71, 0.45, 0, 1

Part(83.60 %): head, $\bar{x}, \sigma, min, max$: 0.84, 0.37, 0, 1

Part(66.88 %): leg4, $\bar{x}, \sigma, min, max$: 2.02, 1.66, 0, 4

Part(21.14 %): horn, $\bar{x}, \sigma, min, max$: 0.37, 0.75, 0, 2

Part(21.45 %): tail, $\bar{x}, \sigma, min, max$: 0.21, 0.41, 0, 1

Part(87.38 %): torso, $\bar{x}, \sigma, min, max$: 0.87, 0.33, 0, 1

Class Label: sofa (397 instances)

Class Label: tvmonitor (545 instances)

Part(91.01 %): screen, $\bar{x}, \sigma, min, max$: 0.91, 0.29, 0, 1

Class Label: dog (1058 instances)

Part(96.31 %): torso, $\bar{x}, \sigma, min, max$: 0.96, 0.19, 0, 1

Part(98.11 %): head, $\bar{x}, \sigma, min, max$: 0.98, 0.14, 0, 1

Part(80.25 %): leg4, $\bar{x}, \sigma, min, max$: 2.28, 1.42, 0, 4

Part(39.60 %): tail, $\bar{x}, \sigma, min, max$: 0.40, 0.49, 0, 1

Class Label: bus (370 instances)

Part(81.08 %): wheel, $\bar{x}, \sigma, min, max$: 1.77, 1.14, 0, 5

Part(33.51 %): door, $\bar{x}, \sigma, min, max$: 0.45, 0.72, 0, 4

Part(44.86 %): fliplate, $\bar{x}, \sigma, min, max$: 0.45, 0.50, 0, 1

Part(61.89 %): headlight, $\bar{x}, \sigma, min, max$: 1.53, 1.67, 0, 8

Part(17.30 %): back, $\bar{x}, \sigma, min, max$: 0.17, 0.38, 0, 1

Part(6.76 %): roof, $\bar{x}, \sigma, min, max$: 0.07, 0.25, 0, 1

Part(94.86 %): window, $\bar{x}, \sigma, min, max$: 3.32, 2.68, 0, 19

Part(68.38 %): mirror, $\bar{x}, \sigma, min, max$: 1.05, 0.83, 0, 2

Part(67.03 %): front, $\bar{x}, \sigma, min, max$: 0.67, 0.47, 0, 1

Part(6.22 %): bliplate, $\bar{x}, \sigma, min, max$: 0.06, 0.24, 0, 1

Part(86.22 %): side, $\bar{x}, \sigma, min, max$: 0.86, 0.34, 0, 1

Class Label: cat (834 instances)

Part(96.64 %): torso, $\bar{x}, \sigma, min, max$: 0.97, 0.18, 0, 1

Part(98.92 %): head, $\bar{x}, \sigma, min, max$: 0.99, 0.10, 0, 1

Part(77.46 %): leg4, $\bar{x}, \sigma, min, max$: 1.86, 1.36, 0, 4

Part(41.97 %): tail, $\bar{x}, \sigma, min, max$: 0.42, 0.49, 0, 1

Class Label: person (5935 instances)

Part(94.44 %): head, $\bar{x}, \sigma, min, max$: 0.94, 0.23, 0, 1

Part(94.63 %): torso, $\bar{x}, \sigma, min, max$: 0.95, 0.23, 0, 1

Part(59.55 %): leg2, $\bar{x}, \sigma, min, max$: 1.04, 0.92, 0, 2

Part(87.01 %): arm, $\bar{x}, \sigma, min, max$: 1.43, 0.71, 0, 2

Class Label: train (379 instances)

Part(67.55 %): head, $\bar{x}, \sigma, min, max$: 0.68, 0.47, 0, 1

Part(67.81 %): coach, $\bar{x}, \sigma, min, max$: 1.17, 1.26, 0, 9

Part(53.83 %): headlight, $\bar{x}, \sigma, min, max$: 1.16, 1.27, 0, 5

Part(2.37 %): back, $\bar{x}, \sigma, min, max$: 0.02, 0.15, 0, 1

Part(14.25 %): roof, $\bar{x}, \sigma, min, max$: 0.22, 0.65, 0, 4

Part(59.10 %): front, $\bar{x}, \sigma, min, max$: 0.69, 0.76, 0, 9

Part(68.60 %): side, $\bar{x}, \sigma, min, max$: 1.26, 1.32, 0, 10

Class Label: aeroplane (549 instances)

Part(96.36 %): body, $\bar{x}, \sigma, min, max$: 0.96, 0.19, 0, 1

Part(51.91 %): wheel, $\bar{x}, \sigma, min, max$: 1.38, 1.66, 0, 8

Part(52.09 %): engine, $\bar{x}, \sigma, min, max$: 0.93, 1.10, 0, 6

Part(0.18 %): tail, $\bar{x}, \sigma, min, max$: 0.00, 0.04, 0, 1

Part(87.98 %): stern, $\bar{x}, \sigma, min, max$: 0.88, 0.33, 0, 1

Part(84.15 %): wing, $\bar{x}, \sigma, min, max$: 1.35, 0.74, 0, 2

Class Label: car (1337 instances)

Part(73.37 %): wheel, $\bar{x}, \sigma, min, max$: 1.33, 1.05, 0, 5

Part(25.43 %): door, $\bar{x}, \sigma, min, max$: 0.36, 0.66, 0, 3

Part(16.68 %): fliplate, $\bar{x}, \sigma, min, max$: 0.17, 0.37, 0, 1

Part(34.18 %): headlight, $\bar{x}, \sigma, min, max$: 0.56, 0.91, 0, 6

Part(34.55 %): back, $\bar{x}, \sigma, min, max$: 0.35, 0.48, 0, 1

Part(16.60 %): roof, $\bar{x}, \sigma, min, max$: 0.17, 0.37, 0, 1

Part(77.64 %): window, $\bar{x}, \sigma, min, max$: 1.39, 1.08, 0, 7

Part(48.32 %): mirror, $\bar{x}, \sigma, min, max$: 0.60, 0.68, 0, 2

Part(45.62 %): front, $\bar{x}, \sigma, min, max$: 0.46, 0.50, 0, 1

Part(16.08 %): bliplate, $\bar{x}, \sigma, min, max$: 0.16, 0.37, 0, 1

Part(80.25 %): side, $\bar{x}, \sigma, min, max$: 0.80, 0.40, 0, 2

Class Label: pottedplant (617 instances)

Part(92.87 %): pot, $\bar{x}, \sigma, min, max$: 0.93, 0.26, 0, 1

Part(98.87 %): plant, $\bar{x}, \sigma, min, max$: 0.99, 0.11, 0, 1

Class Label: table (480 instances)

Class Label: chair (1793 instances)

Class Label: bird (724 instances)

Part(91.71 %): head, $\bar{x}, \sigma, min, max$: 0.92, 0.28, 0, 1

Part(68.23 %): tail, $\bar{x}, \sigma, min, max$: 0.68, 0.47, 0, 1

Part(74.86 %): beak, $\bar{x}, \sigma, min, max$: 0.75, 0.43, 0, 1

Part(56.49 %): leg2, $\bar{x}, \sigma, min, max$: 0.97, 0.91, 0, 2

Part(95.17 %): torso, $\bar{x}, \sigma, min, max$: 0.95, 0.21, 0, 1

Part(35.50 %): wing, $\bar{x}, \sigma, min, max$: 0.57, 0.82, 0, 2

Class Label: boat (594 instances)

Bibliography

[1] JT Turner, Adam Page, Tinoosh Mohsenin, and Tim Oates. Deep belief networks used on high resolution multichannel electroencephalography data for seizure detection. 2014.

[2] JT Turner, Kalyan Moy Gupta, and David Aha. Sparcnn: Spatially related convolutional neural networks. In *Applied Imagery Pattern Recognition Workshop (AIPR), 2016 IEEE*, pages 1–6. IEEE, 2016.

[3] Lawrence Roberts. *Machine Perception of Three-Dimensional Solids*. 01 1963.

[4] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[5] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning.

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

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

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

[11] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.

[12] J. T. Turner, Kalyan Moy Gupta, Brendan Morris, and David W. Aha. Keypoint density-based region proposal for fine-grained object detection and classification using regions with convolutional neural network features. *CoRR*, abs/1603.00502, 2016.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

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

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

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

[18] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

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

[20] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[21] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[22] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1971–1978, 2014.

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

[24] Seymour A Papert. The summer vision project. 1966.

[25] David G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3):355–395, 1987.

[26] Olivier D Faugeras and Martial Hebert. The representation, recognition, and locating of 3-d objects. *The international journal of robotics research*, 5(3):27–52, 1986.

[27] W Eric L Grimson and Tomas Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *The international journal of robotics research*, 3(3):3–35, 1984.

[28] Daniel P Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[29] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[30] Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[31] Peter N Belhumeur, João P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):711–720, 1997.

[32] Hiroshi Murase and Shree K Nayar. Visual learning and recognition of 3-d objects from appearance. *International journal of computer vision*, 14(1):5–24, 1995.

[33] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

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

[35] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[37] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *CoRR*, abs/1512.04143, 2015.

[40] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[41] JT Turner, Kalyan Gupta, Brendan Morris, and David W Aha. Keypoint density-based region proposal for fine-grained object detection and classification using regions with convolutional neural network features. *arXiv preprint arXiv:1603.00502*, 2016.

[42] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[43] Sadiq Sani, Nirmalie Wiratunga, and Stewart Massie. Learning deep features for knn-based human activity recognition. 2017.

[44] Kyle Martin, Nirmalie Wiratunga, Sadiq Sani, Stewart Massie, and Jérémie Clos. A convolutional siamese network for developing similarity knowledge in the selfback dataset. 2017.

[45] Kazjon Grace, Mary Lou Maher, David C Wilson, and Nadia A Najjar. Combining cbr and deep learning to generate surprising recipe designs. In *International Conference on Case-Based Reasoning*, pages 154–169. Springer, 2016.

[46] Petra Perner, Alec Holt, and Michael Richter. Image processing in case-based reasoning. *Knowledge Eng. Review*, 20:311–314, 09 2005.

[47] Robert T Macura and Katarzyna J Macura. Macrad: Radiology image resource with a case-based retrieval system. In *International Conference on Case-Based Reasoning*, pages 43–54. Springer, 1995.

[48] Mojgan Haddad, Klaus-Peter Adlassnig, and Gerold Porenta. Feasibility analysis of a case-based reasoning system for automated detection of coronary heart disease from myocardial scintigrams. *Artificial Intelligence in Medicine*, 9(1):61–78, 1997.

[49] Gowri Allampalli-Nagaraj and Isabelle Bichindaritz. Automatic semantic indexing of medical images using a web ontology language for case-based image retrieval. *Engineering Applications of Artificial Intelligence*, 22(1):18–25, 2009.

[50] Petra Perner and Angela Bühring. Case-based object recognition. In *European Conference on Case-Based Reasoning*, pages 375–388. Springer, 2004.

[51] Alessandro Micarelli, Alessandro Neri, and Giuseppe Sansonetti. A case-based approach to image recognition. pages 443–454, 09 2000.

[52] Daniel López-Sánchez, Juan M Corchado, and Angélica González Arrieta. A cbr system for efficient face recognition under partial occlusion. In *International Conference on Case-Based Reasoning*, pages 170–184. Springer, 2017.

[53] Tinne Tuytelaars, Christoph H Lampert, Matthew B Blaschko, and Wray Buntine. Unsupervised object discovery: A comparison. *International journal of computer vision*, 88(2):284–302, 2010.

[54] Jun-Yan Zhu, Jiajun Wu, Yan Xu, Eric Chang, and Zhuowen Tu. Unsupervised object class discovery via saliency-guided multiple class learning. *IEEE transactions on pattern analysis and machine intelligence*, 37(4):862–875, 2015.

[55] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2027–2034, 2014.

[56] Bryant Aaron, Dan E Tamir, Naphtali D Rishe, and Abraham Kandel. Dynamic incremental k-means clustering. In *2014 International Conference on Computational Science and Computational Intelligence*, volume 1, pages 308–313. IEEE, 2014.

[57] James MacQueen et al. Some methods for classification and analysis of multivariate observations. 1967.

[58] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[59] Philip Wolfe. Note on a method of conjugate subgradients for minimizing nondifferentiable functions. *Mathematical Programming*, 7(1):380–383, 1974.

[60] Marvin Minsky and Seymour Papert. *Perceptrons - an introduction to computational geometry*. MIT Press, 1987.

[61] J.T. Turner. *Time Series Analysis Using Deep Feedforward Neural Networks*. 2014.

[62] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[63] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[64] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[65] Christoph Goring, Erik Rodner, Alexander Freytag, and Joachim Denzler. Nonparametric part transfer for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2489–2496, 2014.

[66] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[67] Brendan Morris, David W Aha, Bryan Auslander, and Kalyan Gupta. Learning and leveraging context for maritime threat analysis: Vessel classification using exemplar-svm. Technical report, NAVAL RESEARCH LAB WASHINGTON DC NAVY CENTER FOR APPLIED RESEARCH IN , 2012.

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

[69] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[70] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2018.

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

[72] Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Ali Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. In *Advances in Neural Information Processing Systems*, pages 6965–6975, 2018.

[73] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[75] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[76] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[77] Jiang Liu, Chenqiang Gao, Deyu Meng, and Wangmeng Zuo. Two-stream contextualized cnn for fine-grained image classification. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[78] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[79] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[80] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[81] Yann LeCun, Sumit Chopra, and Raia Hadsell. A tutorial on energy-based learning. 2006.

[82] J. T. Turner, Michael W. Floyd, Kalyan Moy Gupta, and David W. Aha. Novel object discovery using case-based reasoning and convolutional neural networks. In Michael T. Cox, Peter Funk, and Shahina Begum, editors, *Case-Based Reasoning Research and Development*, pages 399–414, Cham, 2018. Springer International Publishing.

[83] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.