Top-Down Selection in Convolutional Neural Networks

Mahdi Biparva

A dissertation submitted to the Faculty of Graduate Studies
in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy

Graduate Program in
Electrical Engineering and Computer Science
York University
Toronto, Ontario, Canada

September, 2019

Abstract

Feedforward information processing fills the role of hierarchical feature encoding, transformation, reduction, and abstraction in a bottom-up manner. This paradigm of information processing is sufficient for task requirements that are satisfied in the one-shot rapid traversal of sensory information through the visual hierarchy. However, some tasks demand higher-order information processing using short-term recurrent, long-range feedback, or other processes. The predictive, corrective, and modulatory information processing in top-down fashion complement the feedforward pass to fulfill many complex task requirements. Convolutional neural networks have recently been successful in addressing some aspects of the feedforward processing. However, the role of top-down processing in such models has not yet been fully understood. We propose a top-down selection framework for convolutional neural networks to address the selective and modulatory nature of top-down processing in vision systems. We examine various aspects of the proposed model in different experimental settings such as object localization, object segmentation, task priming, compact neural representation, and contextual interference reduction. We test the hypothesis that the proposed approach is capable of accomplishing hierarchical feature localization according to task cuing. Additionally, feature modulation using the proposed approach is tested for demanding tasks such as segmentation and iterative parameter fine-tuning. Moreover, the top-down attentional traces are harnessed to enable a more compact neural representation. The experimental achievements support the practical complementary role of the top-down selection mechanisms to the bottom-up feature encoding routines.

Dedication

To my lovely wife, for her unconditional love and sacrifices ...

To my mother, for her loving heart and prayers ...

To my father, for being so generous and considerate ...

And to the one I am desperately seeking to visit one day ...

Acknowledgements

I am speechless to express how grateful I am to all the people who made this thesis possible. My sincere appreciation goes to my supervisor, Prof. John Tsotsos, whose knowledge and patience aided me considerably during my PhD journey. I am grateful to my supervisory committee, Prof. Richard Wildes and Prof. Marcus Brubaker, for their supports and insightful comments. I have been fortunate to be supervised by these wonderful people, and without them all, I could not have finished this thesis.

I am extremely grateful to all my close friends and colleagues who made this journey much more enjoyable and helped me through difficult times with their sincere supports and encouragement. My special thanks go to Calden Wloka, Markus Solbach, Toni Kunic, Mohammad Hosein Amini, Sajad Shirali-Shahreza, and Ali Mehrkish for their endless help and support during my PhD journey. My kind thanks go to all my previous teachers and instructors.

My deepest appreciation goes to my lovely wife, Najmeh Bordbar, who has always been next to me throughout the most difficult and challenging moments of my PhD program. Nothing I could ever do would repay you Najmeh for all sacrifices you have done for me. Without your love and support, this thesis would not have happened.

I would like to express my eternally sincere gratitude to my father, Rasool, my mother, Zohreh, my sister, Zahra, and my brothers, Mohsen and Mohammad Amin, who remotely far from home supported me throughout these years. I am also thankful to my father-in-law, Kamal, my mother-in-law, Zohreh, and my sister-in-law, Maedeh for their kind and sincere supports. I would be forever indebted for their unconditional love and encouragement. Thank you all for all the sacrifices you made to help me to complete this PhD.

Last but not least, I am grateful of the blessings that the God has bestowed upon me in this wonderful life, the opportunity to live long and healthy to see, think, and learn what, why, and how he has created in this amazing world.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vision has long been recognized as one of the major input modalities for the human brain with striking physiological and psychophysical capabilities enabling sensation, perception, cognition, and action. Vision consumes a large portion of the human brain processing resources in comparison with other modalities such as audition and olfaction. This underscores the critical role that vision plays in the definition of an intelligent system. The human brain receives a high volume of visual sensory data, and as a result, has developed specialized and complicated information processing machinery to support complex decision situations.

The scientific community has conducted a large amount of inter-disciplinary research in different academic fields such as neuroscience, cognitive science, and computer science to not only find answers to questions but also discover unknown aspects of the brain information processing system. Among various cognitive capabilities such as learning, memory, reasoning, and planning, attention plays a critical role. Attention enables us to selectively concentrate on an aspect of the input stream while ignoring others. Due to the large amount of sensory data received at any moment by the human brain, attention is effectively engaged as a crucial component in the efficiency and speed of the entire processing pipeline [6, 7, 8].

The ability to formulate spatial relationships and functional interactions of object categories is an integral component of any intelligent machine. Different generic recognition tasks such as object classification, localization, detection, and segmentation are collectively important to accomplish short- and long-range task objectives for an intelligent vision system. The recent success of machine vision systems is impactd by the improvement in the performance of such recognition tasks [9, 10].

The current dominant approaches to model visual recognition tasks are mainly inspired from the feed-forward pass of information processing in the visual cortex. Information flows from the early stages of sensory data reception to an intermediate representation and finally the top semantic-encoding levels. While the feed-forward pass plays a central role in forming a visual representation, this has been shown in various experimental studies to be incomplete. It is accepted that not only does information flow forward from the bottom to the top of the visual hierarchy, but also top-down connections propagating information in the reverse direction are widely established throughout the visual hierarchy. [11, 12, 13, 14]

In human and machine vision systems, two directions of information processing flows are commonly recognized: a data-driven or feed-forward direction (Bottom-Up), and a reverse direction (Top-Down) that has a predictive, controlling or modulatory role. The sensory input data is processed in the Bottom-Up (BU) pathway and sequentially transformed into high-level semantic information such that some task criterion is satisfied. The Top-Down (TD) direction provides a route for knowledge, goals, priorities and context to be included in relevant processing stages throughout the visual system. We use feed-forward, bottom-up, and data-driven interchangeably in this document to refer to the parametric multi-layer transformation of the input data to the output semantic information. In the following, we introduce the contributions of the thesis in sequential order. Chapter 2 will provide the basics of the neural network machinery and then overview the related approaches to visual attention modeling in neural networks for visual recognition tasks.

Chapter 3 - Top-Down Selection: Visual attention is one of the sources that activates the modulatory top-down processing. The goal is that depending on task requirements, some level of the visual hierarchy needs to be modulated or tuned according to the high-level semantic information computed at the top of the hierarchy. The propagation, formulation, and operation that jointly develop the systematic modulation form the essence of Top-Down visual attention. The major goal of the thesis is to investigate, explore, and formulate in a systematic approach a top-down selection mechanism for convolutional neural networks to facilitate attentional modulations. The critical element of an attentional modulation mechanism is the computation of the selection patterns based on which network responses at multiple layers are tuned. Top-Down selection is among the set of mechanisms that jointly define the visual attention framework proposed by [7]. The role of selection mechanisms is to determine the most important sub-set of the network processing units and parameters according to some task criteria. We present a novel Top-Down attention framework with hierarchical selection mechanisms for convolutional neural networks in Chapter 3 and perform experimental evaluation on the task of object localization.

Chapter 4 - Priming Neural Networks: Visual task priming is an early tuning process before the feedforward information flow in the visual hierarchy. The objective is to tune the visual hierarchy to be prepared for the expected stimulus and thus enable the visual hierarchy to optimally process it. One purpose of visual priming is to help detection of unnoticeable scene elements under severe and misleading visual conditions such as contextual noise and camouflaged objects. We propose a top-down mechanism in the convolutional neural network framework to mimic the process of priming in the context of object detection and segmentation in Chapter 4. This implicit top-down mechanism shapes the bases of the subsequent chapters in which we introduce the explicit top-down selection mechanisms for related visual tasks.

Chapter 5 - Object Segmentation Using Selective Attention: Despite recent success of purely feed-forward models, several aspects of performance degradation in Bottom-Up (BU) networks have been uncovered. Research on visual confusion and adversarial attacks [15, 16, 17, 18, 19] have revealed the vulnerability of data-driven feedforward networks. Furthermore, signal interference issues within multi-layer hierarchical representations are well studied and reported in the literature [20, 21, 22]. Some of these convolutional neural network problems might be due to signal interference issues within such data-driven hierarchical representation. Additionally, these models are sensitive to small input perturbations and are easily fooled for arbitrary final label predictions. In Chapter 5, we propose to extend the TD selection mechanism for the task of semantic segmentation. We test the hypothesis that a convolutional neural network augmented with a TD modulatory and controlling mechanism can achieve better data generalization and be more robust against out-of-distribution perturbations for object segmentation. The attention-driven feature modulation is built on top of the proposed TD selection mechanisms for object segmentation. We experimentally validate the observation that the modulation of the BU features initiated by TD selection improve the benchmark performance metrics in comparison with the baseline model on benchmark dataset.

Chapter 6 - Attention for Compact Neural Representation: The widespread usage of mobile platforms with improved video-recording capabilities have demanded applications with intelligent visual features to be able to process large amounts of data instantly. Neural network compression based on some form of sparsity over the parameter space may provide a route to this goal. The idea is to prune redundant network connections and consequently leave the influential connections intact to maintain network inference accuracy while reducing the redundancy for the sake of a minor compromise of performance loss. Our proposed attentional framework in neural networks is extended to investigate the hypothesis whether such top-down mechanisms are informative to drive the pruning of neural networks. We develop an attention-driven connection pruning approach for the convolutional neural network framework in Chapter 6 and show the parameter reduction is competitive with the baseline approaches.

Chapter 7 - Contextual Interference Reduction: Contextual interference with the foreground target objects is one of the main shortcomings of current neural networks. Due to the dense hierarchical parametrization of convolutional neural networks, "cross talk" of the foreground and the background representation is inevitable [21]. The category label prediction using convolutional networks relies on feature extraction performed uniformly across the input image. Consequently, there is no explicit notion of contextual interference reduction in such models. In Chapter 7, we propose a systematic approach to shift learned neural representations towards the foreground target objects in order to achieve a higher degree of representation dis-entanglement for object classification. We define a selective fine-tuning of neural networks using a unified bottom-up and top-down framework. A gating mechanism of hidden activities is defined in the iterative feedforward pass. An attention-augmented loss function is introduced that permits the network parameters to be fine-tuned for a number of iterations. The fine-tuning using the iterative pass helps the network to reduce the contextual representation emphasis. Therefore, the label prediction relies more on the target object representation and consequently achieves a higher degree of robustness to the background changes. The experimental evaluations on a modified MNIST dataset reveal not only that the results are improved but also a higher degree of robustness to background additive noise is obtained.

To conclude, the significance of the research is two fold: First, we present a unified network with both BU and TD information processing pathways. A novel TD selection mechanism using multiple computational stages is introduced. Second, we conduct research to gain more reliable insights towards the computational role of a TD pass in the conjunction with a BU pass for different visual tasks such as object localization, detection, and segmentation. The significant role of the proposed TD selection mechanism is demonstrated for different tasks with respect to baseline models and compared under different input distortion scenarios.

Chapter 2

Background

In this chapter, we review the literature related to the thesis. We first define the terminology used in the thesis in Sec. 2.1. We then introduce the bases and foundations of the neural network framework for different visual recognition tasks such as object classification, detection, and localization in Sec. 2.2. We define the terminologies, computational elements, and modeling approaches to develop neural network models for such tasks. In Sec. 2.3, previous visual attention modeling attempts and approaches will be reviewed.

2.1 Introduction

Object recognition tasks have been heavily studied in various research disciplines ranging from psychology, cognitive science, and neuroscience, to computer science. In computer science, the primary goal is to develop a computational machinery based on a solid understanding of different visual tasks. The secondary goal has been to develop machine vision systems that compete with human performance; this has proven to be very challenging. In our definition, we refer to object recognition as a general task that involves two different types of tasks; object instance recognition and object class recognition. The first type is a matching problem such that previously seen object instances have to be identified under some variable imaging conditions and partial occlusions based on a bank of visited exemplars. Image alignment and registration processes also are often required. For example, recognizing the face of a particular person under different conditions is defined as object instance recognition.

The second type, also known as category-level or generic object recognition, aims to recognize instances of learned categories. The apparent difference is that in the latter, the target instances are unseen by the recognition model and the generalization capability is important. However, in the former, a particular instance is known and the goal is to identify it under various conditions. Following Perona's [23] definitions, we refer to the object instance recognition as the Verification task and the object class recognition as either Detection or Classification. Both detection and classification can be augmented with the object localization task. Object localization is defined as the task of spatially localizing a target object in the input image [23]. In other words, the answer to the question of "where is a target object?" satisfies the localization task requirement.

Perona [23] defines object classification as: given an image patch, what object category label, from a set of predefined categories, best represents the patch content? The objective is very straightforward in the sense that the output is produced as the predicted label of the given image patch. The assumption of this task obviously is that the patch should reflect the object and the context it resides in. Depending on how much context surrounds objects in input images, localization could be non-trivial.

Object recognition tasks performed by the human visual system in a real-life scenario is even more challenging. An object in a real-world scene is often perceived by a human observer in a cluttered environment with lighting and shading variations. The detection of an instance of some object category in a noisy cluttered environment with partial occlusion makes the task of object detection more complicated. A detection model predicts whether instances of some particular object category exist in the input images. Following the requirements defined for the detection tasks [24, 25], object detection always requires the localization of the detected instances. However, localization for object classification is explicitly mentioned if required. The single-object localization task is one of the challenges defined for the ImageNet benchmark dataset [24] and it explicitly requires location prediction on top of object classification.

Visual variations are transformations in the spatial domain that make recognition very challenging. All types of visual variations can be divided into two distinct groups: object variations and image variations. Object variations work within a category delimiting different instances based on visual cues such as color, texture, shape, pose, and size. However, image variations are caused by different lighting, place, atmospheric (weather), illumination, and viewpoint conditions. Object variations can change one instance of a category to another instance while image variations always keep the instance identity intact. The variation by the viewpoint condition is separate from the object pose variation. It is due to the fact that viewpoint variation is caused by the external observer while the object pose variation is produced by the structural variation of the object itself. Therefore the former is purely independent from the nature of object categories while the latter is inherent in object categories. Nonetheless, recovering the pose of an object is usually preceded by an estimation of the viewpoint parameters.

Object categories can further be distinguished according to their structural configurations. Horse, dog, chair, book categories are instances of deformable or structured object categories. On the other hand, amorphous or unstructured categories do not have constant shape or size such as cloud, sky, grass categories. Consequently, they can be described in terms of local appearances based on color and textural patterns. They are called things and stuff semantic categories respectively in [26].

The first and most critical step towards solving object recognition problems generally is the choice of visual representation for input images. Moving from low-level raw representation of gray-scale, color, gradient, local shape and texture cues to a mid-level feature representation is a very challenging task. Objects could be represented in the 2D image domain based on explicit shape and form cues, which rely on the boundary depiction of the objects distinguishing them from the background, or based on the visual appearance of the object surface. Following each would lead to either shape-based models or appearance-based models respectively.

2.2 Object Recognition with Convolutional Networks

By defining object classification as the task of predicting the category label of an input image, the crucial part is to learn a visual representation suitable for invariant object classification. Variability in this task ranges from object to image variations. Robustness to such variations is the key aspect of a reliable representation. Dealing with object location variation within the image frame is very important. It could be, however, overcome by framing the object of interest in the center of the input image or by injecting very slight shift invariance into the representation model so then the object location shift would be tolerated within the image frame. A visual hierarchy is defined as a systematic organization of multiple levels of feature extraction, grouping, selection, and integration into a unified framework. The objective of the hierarchy is to be able to represent input data according to some task requirements and specifications.

In the neural network modeling paradigm, object classification methodologies consist of two main components: the visual representation and the discriminative classifier. Visual representations are categorized based on the depth of visual representations ranging from shallow to deep representations. The intuition is that the depth of the visual representation would help to better encode the semantic and appearance factors. Various attempts to visual representation modeling for object recognition can be differentiated by the consideration of the architecture criteria such as the depth of the representation defined by the number of filtering, sub-sampling, pooling, and non-linearity layers. The type of operation at each layer in addition to the intrinsic parameter settings is another aspect of representation modeling. Whether the hierarchy parameters are defined to be hard-wired or learned in a supervised or unsupervised manner matters when characterizing one approach from another. Lastly, the modeling framework that imposes the objective criterion onto the underlying representation plays an important role. The classification paradigm characterizes the objective terms, the optimization routines, and the parameter updating procedures.

A visual representation is defined as a transformation function $\phi(I)$ that encodes the input image I to some form of vectorial representation in a high-dimensional space. A robust nonlinear representation is capable of projecting the input space to a feature space such that visual tasks can be performed using a linear decision machine. Manifold learning of a visual representation hypothesizes that a robust visual representation transforms a tangled high-dimensional input space into a feature space such that the input data samples lie on linearly separable manifolds.

Convolutional Neural Networks have been studied from different perspectives ranging from a biological point of view in neuroscience to a computational point of view in computer science. In analogy to the brain, the basic operation of weight sharing implemented in convolutional neural networks can be regarded as the representation of a particular salient feature over the retinal topography in one visual cortical area. The basic selectivity of neurons achieved through their receptive field profile can be regarded as the convolution operation in such networks. [27] is one of the first attempts to propose a computational model with convolutional connectivities. It was, however, formulated to suit unsupervised learning problems. A more extensive realization of convolutional networks in the computer vision application of isolated character recognition was appeared in [28, 29] purposefully oriented for supervised learning problems. The network architecture named LeNet-5 shows how a well-implemented convolutional visual hierarchy with the representation learned through a particular optimization procedure can be successful in a real computer vision application.

The breakthrough of convolutional networks in large-scale object recognition competitions started with the inspirational work of [30], which is commonly referred to as AlexNet. The trained convolutional network was one of the largest and well-implemented networks to date with the best performance results on popular benchmark datasets. The highly GPU-optimized implementation of convolutional networks has been very highly effective. The overall computation throughout the hierarchy of convolutional networks is such that local parallel units can be utilized to achieve a major processing speedup in both the learning and inference stages. GPUs are designed to have many local parallel processing units that can be assigned to perform the computation required for multiple nodes simultaneously in a convolutional network. The use of Rectified Linear Units (ReLU) [31] as a novel non-linearity and a regularization method called Dropout [32] to reduce over-fitting in the fully-connected layers are among the important breakthroughs in the advances of neural networks.

2.2.1 The Basic Building Blocks

Convolutional networks are comprised of a modular combination of different types of layers on top of each other beginning from an input layer and ending with a score function based on which the loss function is measured according to the label of the input data.

Convolutional networks are comprised of various stages of processing consecutively processed in a cascade manner on top of each other forming the visual hierarchy. The convolution operation through a particular kernel profile is the essence of feature selectivity. A non-linear activation function is applied in the next stage to compute the output of model neurons (hidden units) at each layer. This is usually followed by another type of layer called pooling and sub-sampling layers to impose some level of gradual shift invariance.

Convolutional Layers: The first type of layer is the convolutional layer, the core building block of convolutional networks. By analogy to the receptive field selectivity of the neurons in visual cortex, the basic operation of weight sharing implemented in convolutional networks leads to the representation of a particular type of feature over the retina topography in a visual cortex area. In other words, the basic selectivity of neurons through their receptive field profile can be regarded as the convolution operation in such networks. The convolution operation is the means through which the feature selectivity is locally applied. Convolutional layers are characterized by design choices such as kernel size, number of input feature channels, number of output feature channels, stride, and border padding. Fully-connected (FC) layers are 1×1

layers inspired by MLP networks. Hidden nodes in FC layers are connected to all of the hidden units in the previous layer. A matrix multiplication added with a bias offset is needed to compute activities in a FC layer. The output feature channels of the last FC layer is equal to the number of categories in classification problems. It learns to encode for the class scores, which are arbitrary real-valued numbers. There is no activation function after the last FC layer but rather the logistic regression (for binary classification problems) or softmax layer (for multi-label classification problems) to generate predictive probability values.

Non-linearity Layers: The other layer type, mostly used after the convolution layer, is the non-linear activation function. The idea is based on the biological inspiration of real neurons for which the firing rate of neurons is limited between zero and a positive clamping value [33]. Once the output maps are computed by a convolutional layer, a transfer function is applied to map the input to a proper neural response range. *Sigmoid* and $Tanh$ are the most common functions used in the early days. Rectified Linear Units (ReLU) [31] is simply a linear function such that the values lower than a threshold are set to zero. To overcome the shortcomings of ReLU such as the zero gradients for value lower than zero, different ReLU variants have been proposed such as the Parametric ReLU (PReLU) [34], the Leaky ReLU (LReLU) [35].

Hidden Normalization Layers: One modeling inspiration from the real neuronal mechanisms observed in neuroscience research is inhibition processes such as lateral inhibition. There are different attempts to address the need for a similar mechanism in convolutional neural networks. Local Response

Normalization (LRN) [30] and Local Contrast Normalization (LCN) [36] are among the first attempts to investigate the role of normalization layer in neural networks. They implement a similar idea with a subtle difference in terms of the scope of normalization and also whether it is just divisive or also subtractive too. Such normalization is applied after non-linearity layers in certain layers to stabilize training and improve generalization. Unlike these two methods with small normalization scopes, Batch Normalization (BN) layer [37] is proposed with a more global scope of normalization to mainly deal with improper parameter initialization and lack of training consistency, and it usually comes after ReLU layers. Since the batch concept is not always present, Layer [38], Instance [39], and Group [40] normalization layers are proposed to avoid exploiting directly the batch dimension. The overall core idea is to collect statistics across the input hidden tensors according to some grouping approach (e.g. batch dimension in BN), and then use them to normalize the input hidden tensors in a divisive and subtractive manner followed by some parametric scaling and addition. [41] recently proposes a synchronized BN layer that collects statistics and updates coefficient parameters across multiple GPUs when the mini-batch size is high (e.g. 128) and each GPU holds one input sample due to a large network or input data size.

Pooling Layers: Pooling layers are utilized as an attempt to inject small local shift invariance into the overall representation by the gradual pooling mechanism. Average and Max pooling [42] are the two popular widely used pooling layers, which spatially pool information across a 2D window of hidden units on a feature map. Properties such as kernel size and sub-sampling stride are important. In contrast to the spatial pooling approaches, Maxout [43] is regarded as a pooling operation over the inter-channel dimension. Other variants such as Probabilistic Maxout (Probout) [44], Probabilistic Max-Pooling [45], p-norm pooling [46], and parametric p-norm pooling [47] are proposed in the hope of improving their invariance properties. Network in Network (NIN) [48] is proposed to incorporate a higher degree of complexity in the profile selectivity of hidden units in convolutional networks. It replaces a simple weighted summation performed by convolution with a Multi Layer Perceptron (MLP) motivated layer to add more complexity for feature encoding. NIN is respected as a cross-channel parametric pooling and is extended by [49] into a new architecture using Inception modules. The idea is to use a set of 1×1 convolution filter banks to reduce the number of input feature maps into a lower more computationally affordable number for the subsequent $k \times k$ filter banks. A particular incarnation of the inception modular architecture is called GoogLeNet and introduced in [49]. Residual networks [50] are proposed as a way of overcoming the obstacle of losing gradient information in convolutional networks using skip connections. Inspired by the Spatial Pyramid Kernel Matching [51], Spatial Pyramid Pooling (SPP) [52] is defined to overcome the variable image size issue by a pooling layer that outputs fixed-length feature maps.

Training and Testing Protocol: A popular learning algorithm for convolutional neural networks is to minimize a loss function over the training set. This is a non-linear optimization problem that is done using iterative gradient descent optimization updates. The goal is to update parameters with the gradients that minimize the loss function. Basically a best direction based on which a particular parameter should be changed is the one given by the first partial derivative of the loss function with respect to the parameter. The chain rule is a systematic approach to analytically compute error gradients of network parameters. It is interchangeably referred to as Backpropagation in the neural network community [53]. Having the input data propagated into the feed-forward layer throughout the network at the inference phase, error gradients of the weight parameters are analytically computed according to an objective function and systematically propagated backward from one layer to the next. Once gradients are computed, the gradient optimization algorithm updates the weight parameters and repeats these steps for the next set of training samples.

Network Regularization: As the capacity of a network in terms of free parameters increases, the model tries to memorize the training data set rather than to learn the underlying data distribution. This is famously known as over-fitting and will lead to low generalization performance at the testing phase. In order to avoid over-fitting, various regularization methods have been proposed such as the weight-decay approach (L1- or L2-norm). It is imposed on the objective loss function at the learning phase to regularize weight parameters. Early stopping is also one of the early proposals. Dropout [32, 54] is proposed to decrease the co-adaption that emerges during the training phase between hidden units by randomly setting hidden units to zero. Adaptive Dropout [55] uses an auxiliary network to learn the probability based on which dropout mask is generated for each hidden uni. DropConnect [56] set the connection weights rather than the hidden units to zero .

Network Representation Visualization: Understanding the hierarchical representation of convolutional neural networks plays a critical role. [57, 58] propose to define pooling switches recorded in the feedforward pass and are used to project back to the top layer activities to the input layer using the Deconvolution layers. [59] proposes a gradient-based visualization method that uses automatic differentiation of the loss function with respect to the input layer. The idea of activation maximization simply is to maximize the classification score of a specific class label penalized with some regularization term such as L2-norm term for the input image. The regularization terms act as image priors that restrict the search process in the input image space to those that can resemble well natural images. In addition to the hand-designed natural image priors [60, 61, 62]such as Gaussian blur and α-norm, [63] proposes a learned prior based on Generative Adversarial Networks (GAN) [64]. This learned prior provides high-quality input image search results and intermediate hidden activity visualizations.

2.3 Visual Attention in Deep Learning for Object Recognition

Attention in humans helps to concentrate and tune the brain's computational resources to fulfill task requirements within a particular time frame. Object detection in a large context is a task that inherently demands a form of processing concentration. In a real life scenario, object detection is regarded as finding instances of a particular category in a noisy, cluttered, and complex visual environment. The task of finding a car on the street among many different irrelevant object categories and then consequently be able to localize it, is an example of the goal approached in object detection with localization. One question that emerges at this point is whether the localization is by itself a task separate from the detection task. Is it that first an object is detected out of noisy context and then, upon the requirement of the task, is localized? Or is it that detection is performed on a localized portion of input image, and thus, localization is achieved a priori to detection? These two extreme points form the two sides of the spectrum of approaches for object detection. We call them late- and early-localization approaches respectively.

2.3.1 Early-Localization: Hypothesizing for Objectness

Early-localization measures some generic definition of objectness from local and pictorial cues in an image, and then outputs an importance map of the regions that are most likely to contain category objects. Objectness indicates how likely a category object exists across image regions [65, 66]. A subtle question is what is the best metric to measure objectness, and what differentiates object categories for which ground-truth labels are provided from the unlabeled ones. A detection system utilizes the objectness measurement to pick the most salient regions to attend to for category prediction. Refinement over background regions and pre-trained visual representation seems necessary to help early-localization approaches beat the state-of-the-art in object detection. It is worth mentioning that the weak early-localization approach is equivalent to the brute-force sliding window approach to output importance maps over which candidate regions containing objects are returned. On the other hand, a strong early-localization approach achieves the 100% recall accuracy with a number of bounding box proposals equivalent to the number of ground truth bounding boxes. In other words, the highest level of recall accuracy is achieved with the least number of proposals using the strong approach.

A measure of objectness is provided over the entire high resolution image using a class-agnostic algorithm in order to model a level of attention for the classifier that predicts category labels. Objectness models work in the bottom-up fashion without utilizing any form of top-down task knowledge. They rather collect and integrate pictorial and structural information locally from different tracks of visual processing to find regions with a high measurement of objectness.

Approaches for object proposal generation can be categorized into three distinct paradigms. The first one is harnessing the image pictorial structure locally and globally to merge the super-pixels into a hierarchy [67, 68, 69, 70]. Cutting through the hierarchy at some specific level provides a number of bounding box proposals. We will explain this approach in more detail in Sec. 2.3.1.2. The second paradigm is measuring objectness of boxes through a learning method. It is intrinsically statistically data-driven [66, 71, 72, 73]. Using a pre-trained visual representation, objectness is learned in a class-generic manner. This is similar to the simultaneous detection and segmentation methodology. Further information is given in Sec. 2.3.1.3. Lastly, based on the classic figure/ground criteria, the third paradigm uses segmentation algorithms which are used to partition input images spatially into distinct regions [74, 75, 76]. The bounding boxes enclosing the partitions are proposed for object classification. This approach is expanded in more detail in Sec. 2.3.1.3.

2.3.1.1 Metrics to Measure Objectness

Objectness detection algorithms are interchangeably referred to as bounding-box proposal algorithms. The intuition is to use an objectness measure and other factors to confidently propose regions in boxes that most likely span the entire extent of the objects in an image. Three evaluation measurements are commonly characterized in the performance comparison of different algorithms. First, the recall rate is defined as the accuracy of hitting correct ground-truth bounding boxes from the set of proposals regardless of the false positive rate. There is always a trade-off between false negative and false positive rates. Accounting for one would impact the other. Therefore a good cutting-point threshold is always cross-validated. However, region proposal algorithms are mostly evaluated based on the recall rate. Precision is left over to object classification algorithms. The main criterion is to increase the recall as much as possible.

The second evaluation metric is the number of proposals to achieve a particular level of recall. A powerful reliable objectness detector is recognized based on the number of proposal boxes. Apparently, as the number increases, the object classification module takes more time and gradually shifts towards the brute-force sliding window search mode. Hence, a decent close-to-optimal objectness detector is the one that while maintaining a low number of proposals, hits the highest recall rate of one.

Third, the size accuracy of the proposals measures objectness detection performance. Tightness of the proposals could be relative and gets refined in the recognition step according to the relative shift invariability of robust classifiers. In practice, recall rate is measured as the total number of proposed bounding boxes that have overlaps of more than some threshold (mostly 0.5) with the ground truth bounding boxes.

2.3.1.2 Harnessing the Pictorial Cues in a Hierarchy

Selective Search (SS) [67] is inspired by the segmentation community to use local cues to separate figures from ground. SS combines the best of segmentation with a selective search over various locations in the image. Complementary grouping criteria and invariant color spaces are used to diversify the search over the entire space for targeting better regions. SS attempts to use segmentation to narrow down the large search space over locations, aspect ratios, and sizes. Rather than the common goal of proposing a strong segmentation strategy to partition regions apart, SS uses various strategies to extract knowledge from various aspects of an image ranging from shape, color, curvature, texture. In this regard, there is huge similarity in the representation space of SS with saliency prediction algorithms. These image clues are grouped systematically in a bottom-up manner to generate good object locations using a diverse set of strategies. A fast graph-based algorithm [68] is used to initialize SS. Then a greedy grouping algorithm is iteratively used to construct the hierarchy of regions until the entire image is grouped into one region. Grouping is based on the feature similarities of all the neighboring regions and merging of the two most similar ones. This leads to a tree of regions with leaves as the initialized regions and the root as the region covering the entire image. The similarity based on a variety of complementary measures are constrained to be fast such that the measures can be propagated through the hierarchy so then at each level the similarity can be computed from the measures of the previous level rather than the measure from image pixels.

One of the diversification strategies in SS is the utilization of various color spaces with different invariance properties. The second strategy is to use four similarity measures between regions: color, texture, region size, cross-region similarity for combination. The combination of these four measures of similarity is used to diversify the search. The third strategy is to generate different starting regions via varying the threshold of the graph-based over-segmentation algorithm. Different ordered sets of proposals are generated according to the diversification strategies. Then regions are extracted for proposal based on their overall ranking.

A multi-scale hierarchical segmentation and object hypothesis generation system in a unified framework called Multi-scale Combinatorial Grouping (MCG) is proposed in [69]. First, a fast normalized cuts algorithm is proposed. A set of local contour cues are extracted as the input to the algorithm. Then, a high-performance hierarchical segmentation approach that leverages effective use of multi-scale information is employed. Finally, exploring the combinatorial space of possible object candidates, regions are combined efficiently to account for the accurate proposals. Two steps are taken to reduce the number of candidates while keeping the quality. One is through combinatorial grouping problem formulation and the other through training a random forest regressor from the low-level bottom-up features to predict the overlap of the region with the ground truth.

MCG differs from SS in focusing on multi-scale information rather than color spaces to generate various hierarchies to diversify the object search. Moreover, pixel accuracy region extraction is more considered in MCG in contrast to SS. MCG outputs regions whereas SS outputs directly bounding-boxes. However, a normalized evaluation measure which consists of the overlapping area, the number of proposals, and the execution time has to be employed for a fair comparison of different algorithms. Object hypothesis proposals are not meant for accurate categorization result but speed and smaller accurate number of candidates for an optimal object recognition system. It is stated that MCG is marginally better than SS in terms of the amount of the overlap of the proposals with the ground truth.

The Edge Box algorithm is proposed to generate bounding boxes containing objects using edges [70]. Edges are directly used rather than segmented regions, as an informative representation in a non-hierarchical fashion to measure if an object is enclosed in a box. The efficiency of computing edge maps and the sparse representation have made them a promising approach. The number of completely enclosed contours indicates the likelihood of an object in a box. Contour straddling is considered as the sign of a partially enclosing object. Thus such contours are removed during the process of scoring the completeendres2010categoryly enclosed contours. The issue of how to search the space over various positions, scales, and aspect ratio values is addressed using the sliding window search strategy. The candidate boxes are ranked and non-maximum suppression is used for the final proposal generation.

2.3.1.3 Data-Driven Object Hypothesis Generation

[66] proposes to train a classifier to detect a well-defined generic object class. Then, at inference time, boxes are randomly sampled across the entire image and the classifier is applied. Those boxes detected with high score are proposed as the object hypothesis. The classifier would have to be able to discriminate among well shaped objects from an amorphous background.

Closed boundary, different appearance from the immediate surrounding, and uniqueness are the characteristics for which four measures based on image pixels are proposed: multiscale saliency (uniqueness), color contrast (different appearance), edge density (closed boundary), and super-pixel straddling. Regardless of pixel information inside the window, the probability of a window of particular size and location is learned.

A training set of positive and negative windows are generated randomly. The Bayesian framework is used to maximize the posterior probability of the objectness given the cue measures. Once the posterior distribution is fit to the training set, the posterior predictive probability over all random windows are calculated. Those with highest probability are selected to propose.

Representation learning using deep convolutional networks can also be used for the object proposal generation task. [71] trains a variant of convolutional networks to output a fixed number of bounding boxes with some confidence scores for each input image in a class-agnostic paradigm. The network has two output layers with a fixed number of candidate outputs; one predicts the bounding box locations and the other outputs the confidence score. They use a transformation of the representation extracted from the last hidden layer to calculate the corresponding output value. The objective function was modified to include the terms appropriate for this formulation.

In order to exploit the use of parallel processing using graphics processing units (GPU) and decrease the computational time of region proposals for objects, [72] proposes Region Proposal Network (RPN) to optimize the deep visual representation of the detection network end-to-end with a relative cost function for object proposal confidence measure and bounding coordinates in an alternative learning paradigm. Obviously, the visual representation learned for the detection task can be shared for the task of object proposal generation. In this paradigm, both tasks are alternatively optimized.

Following the common practice in the community, on top of the last convolutional layer, a fully-connected mini-network is attached. This small network is slid spatially over the whole last layer. Two ultimate fully-connected layers implementing correspondingly the proper loss function terms are defined: classification and regression layers. Each mini-network predicts for a predefined number of anchor boxes. The classification layer predicts the binary decision of objectness or non-objectness and the regression layer outputs four coordinate values for each anchor.

The classification score function can be analyzed locally by sampling the regionally-masked input image [73]. The intuition is that masking out a region of the input image overlapping an instance of a specific class would cause a significant drop of the classification score. This intuitive method can be used for object proposal generation and also in the localization task of object recognition.

An agglomerative hierarchical clustering of regions based on the amount of drop, the feature and size similarity, and spatial vicinity is employed to derive a unified saliency map for automatic localization. The cluster is initialized with super-pixels. The pairwise clustering is based on the following criteria: similar large drop, similar emerged representations, cover as large of the image as possible encouraging small regions to merge, and spatial adjacency. These terms are weighted, combined and at each iteration the pair that maximize the sum is combined. The stopping criterion is where one region is left. The feature similarity is the histogram intersection of the representation of the last fully-connected layer.

2.3.1.4 Figure-Ground Segmentation for Objectness Measure

The classic approach to object segmentation is used to generate object hypotheses for bounding box proposals. The output of such segmentation-based approaches partitions regions with which bounding boxes are proposed. [74] segments input images into figure-ground partitions using a multiple-constrained parametric min-cuts method, and then learns to group them using the likelihood that a partition contains a complete object. A ranked list of such figures is then generated to be refined as the object hypotheses. Features are extracted following Gestalt psychology principles to approach visual grouping using properties such as proximity, similarity, and good continuation. Graph, region, and Gestalt properties are the three feature sets based on which the segments are represented. A random forest regressor is used to learn the importance of these features to regress the largest overlap with the ground truth object. The predicted overlap is used to rank segments such that the similar regions are in adjacent positions.

[75] have used output segments as input to a categorization module to score each segment based on the category membership function. These scores are used to refine and merge segments into one region representing one object. Such an approach can be seen as the bottom-up/top-down segmentation framework rather than object localization due to the high emphasis on the accuracy of the object segmentation. The score function is a support vector regression machine to predict the union-over-segmentation of the region with the ground truth. [76] follows the same approach towards object hypothesis proposals with some changes in details. One difference is the use of learned affinity functions to guide segmentation. The ranking model is defined as a structural optimization problem.

All three approaches attempt to generate objectness hypotheses using combinations of various visual cues, the locality of feature extraction, and the type of visual representation. Regardless of how they approach visual representation for objectness, all of them precede the classification module. In other words, object detection using objectness measure is a rapid and passive approach to localization. Once the set of proposals are generated, there is no need to revisit the proposal module down the detection pipeline.

2.3.2 Late-Localization: Top-Down Attention

The late localization paradigm, on the other hand, aligns more with the biological studies of visual attention such that localization is fulfilled upon a request based on the task requirements. Visual attention modeling plays an important role in the completion of various visual tasks. Particularly in object recognition, feature-based attentional modulation can bias one aspect of objects to facilitate the recognition task. However, visual attention can also be used for object localization. Covert and overt attention are two different operating modes that are distinguished based the role of eye movements in attentional engagement. The process of attending to a region in the input visual field without eye movement defines covert attention. In other words, the visual representation for the input data is used to address the localization requirement while the input data is not changed.

Due to the rapid fall-off of the spatial acuity from the central regions of the wide field of view in humans, localization of objects in the periphery is challenging. Consequently, the several field of view changes by the human observer including the body and eye movements to bring objects from periphery into the central field for accurate localization is covered by the definition of overt attention. The role of eye movements is to bring ambiguous regions of the field of view into the highest level of acuity for further inspection. The set of mechanisms for both covert and overt visual attention is outlined in a unified framework in Selective Tuning [7].

Despite recent successes and development of the early-localization approaches using engineering advances in deep learning, one fundamental question still remains not fully answered in this paradigm. What is the reference frame in which objectness is measured? Is there a robust approach that can reliably predict objectness regardless of object and image variations? If there is not, the task of early-localization using object proposal generation reduces to the brute-force sliding window paradigm which is inefficient and slow. In the real-life scenarios of object category detection, one obvious issue is how to locate the object-centered reference frame properly over the visual input data with high resolution and wide field of view. Tsotsos [77] analyzes the computational complexity of recognition problems. He proves that visual search problems are inherently NP-complete. Therefore, according to the available computational resources of the brain and given the usual ordinary time frames, it is computationally impossible to find answers to such complex and dynamic problems in the high-dimensional input space. Consequently, the brain essentially devises approximations to harness the task knowledge to decrease the complexity and bound the problem domain to possible regions [7].

2.3.2.1 Top-Down Approaches for Localization

Having learned the bottom-up visual representation in a feedforward manner, the question is how to unify the top-down attentional modulation in one framework. The characteristics of the representation and the computational basis of the attention modeling are two key aspects based on which different frameworks arise. In a classical approach, [78] has proposed a shallow representation with two processing streams namely shape and color cues formulated in the bag-of-words framework. Recognition of object categories is pursued by training a category-specific shape model and then uses the model predictions for attentional top-down modulation to bias the weight parameters of the shape stream for the enhancement of the recognition performance of the model.

[79], on the other hand, utilizes a Bayesian modeling approach to unify the two streams addressing the “What” and “Where” problems in a visual recognition system. It is stated that some of the attentional phenomena such as bottom-up and pop-out effects, multiplicative modulation of neuronal tuning curves, and shift in contrast responses are predicted naturally in this modeling approach. Bayesian inference is followed to show that within this framework, computational attention could answer the “what is where” question.

A generative model is defined using the Bayesian inference rules. The location and category of objects are defined by stochastic variables that describe the scene layout of an input image. Some assumptions are made to factorize the joint probability of the generative model. For instance, it is assumed that recognition is modeled for one object at a time. The other assumption is that object and location variables are independent. Additionally, features at all locations are modeled using latent variables such that they are independent from each other so their probability factorizes into a simple form for ease of probability inference. Given the probabilistic graphical model defining the factorization of the generative model, translation invariance, spatial attention, feature-based attention, and feature pop-out are formulated based on Bayes rules of inference. It is empirically verified using the conducted experiments that the expected behavior of the model highlights some aspects of the attentional mechanisms.

[80] proposes to use Deep Boltzmann Machines (DBM) [81] as the machinery on which the visual representation is learned. This particular representation is then used to model covert object-based attention by the weight sharing commonly practiced in convolutional networks. The aim is to implement DBM such that the model weights are shared locally over feature maps at each layer and also have depth such that the local integration of information over the receptive field gradually evolves in higher layers.

[80] proposes a covert attention mechanism implemented on the learned representation of a DBM. The goal is to study the role of the covert suppressive attention to retain the recognition performance of the model while increasing the level of surrounding noise in the background. The top-down projection deterministically reconstructs the activities at each hidden layer. The top-down reconstruction in generative models is used to tackle occlusion and missing parts. In the feed-forward inference, each hidden layer probability is measured using a sigmoid function given the weighted sum of the lower layer input. However, the recurrent inference considers both the top-down and bottom up connections. The reconstruction in recurrent inference then begins from the top layer and descends to the early layers. It is observed experimentally that for non-cluttered object samples, single feedforward inference is sufficient. However, the representation of the cluttered object needs to be disambiguated through recurrent inference.

The object category detection task deals with searching for an instance of a category within a large visual field with lots of clutter in the environment. The question that arises is whether a purely-feedforward hierarchical representation would suffice to solve the problem? Feedback connections are studied very well in physiological and psychophysical experiments on the brain. Evidence supports the hypothesis that feedback connections play important roles in all level of the visual hierarchy [12, 11]. [22] conducted a set of psychophysical experiments to show hierarchical pooling models of object recognition would fail to account for the crowding scenario using human data. In the crowding experiment, populating the target with the surrounding neighboring distractors would deteriorate the target discriminability. The pooling models predict that as the number of distractors increases, the target discriminability would suffer more. However, experimentally the opposite is observed. [22] concludes that the same way low-level processing determines high-level processing, high-level processing determines low-level processing. In other words, global integration of information over large parts of the visual field along with iterative recurrent processing consisting of both feedforward and feedback are necessary and sufficient conditions for the object recognition task.

Knowing that a recognition system needs to consider principled ways to recall noisy, occluded, and missing data in a top-down manner, [82, 83] propose a selective attention model inspired extending the original multi-layer neural network model of Neocognitron [27], which is basically an example of an associative memory model. The goal is to demonstrate that not only can the information flow in a bottom-up manner in such associative models but also it can reciprocally flow in a top-down manner. Therefore, at an intermediate layer, there are both incoming connections from the layer below and also outgoing connections from the layer to the layer below. The former helps recognition and the latter aids reconstruction and occlusion recall. It is called auto-associative memory models.

The bottom-up connections of the selective attention model are first learned using a self-organizing learning method, in which there is no supervising to guide the weighting of the connections between two neurons. Next, the weights of the top-down connections are adjusted according to the weight values of their corresponding bottom-up connections. Once the learning is over, the connection weights are not modifiable anymore. The bottom-up connections will be deactivated if the top-down counterparts are not activated through the course of time. The selective attention model operates in a simultaneous recognition and segmentation mode. After a number of inference and reconstruction iterations, the pattern is not only correctly recognized and but also segmented from the context it resides in.

In addition to the associative recall functionality of the selective attention model, where noise and partial occlusion can be tackled, the model can also deploy attention to one part of the input data in case of seeing two visible stimuli simultaneously. Deploying the top-down attention of a selected top node (mostly the highest responding node) routes through the visual hierarchy to reach to the segment of the input data at which the known stimulus pattern have been appeared. This is reminiscent of a localization procedure since deployment of the selective attention to one region of the input stimulus narrows down the processing throughout the entire hierarchy.

Importantly, the top-down flow of processing follows the same trace of routing as the bottom-up processing. The selection mechanism is such that the top-down processing brings into the focus of attention the bottom-up processing by gain modulating directly the activities of neurons falling inside the routing pass and implicitly suppressing the remaining neurons. Therefore, the top-down modulation globally affects the entire hierarchy regardless of the routed pass.

A framework for visual attention seems to be important to close the loop for reaching the state of being able to solve the visual task of object recognition [84]. [21, 7] propose attention as a set of mechanisms for which the aim is to adaptively tune search processes essential to succeed on different visual tasks. Selective Tuning [21, 7] is a computational dynamical framework that accounts for various mechanisms known to be required to perform visual search approximately similar to what is observed physiologically and psychophysically in the primate brain visual cortex. Task priming, bottom-up inference, top-down selection, and iterative passes are the main binding stages defined in [7] over the course of the processing time measured from the onset of stimulus. In this dissertation, we develop and highlight the roles of these four computational stages in convolutional neural networks for different recognition tasks. Unlike the processing units in convolutional neural networks, the visual hierarchy in the Selective Tuning model is comprised of neuronal dynamical computation units which are tuned to model the firing behavior of neurons in the human visual cortex [85, 86].

As the Selective Tuning model might echo some resemblance with the selective attention model [87], one of the main divergences of these two models from each other is the type of attentive modulation employed in each. The latter uses the facilitatory mechanism of gain modulation to deploy top-down attention over the entire hierarchy, whereas the former uses the inhibitory mechanism of surround suppression to deploy top-down attention to a particular region of interest rather than the entire hierarchy leaving the remaining parts unaffected. In other words, the non-attended regions of the input stimulus do not disappear from the bottom-up flow of processing.

The other difference stems from the type of competition employed in each of the two approaches. The Winner-Take-All process [88, 87] used in the selective attention model selects the maximum responding neuron in each competition. The output of the competition in WTA is always a single value resembling the maximum of the competing neurons. On the other hand, the Selective Tuning model implements a dynamic competition derived from a parametric variant of the WTA process termed θ-WTA [7]. θ is a task specific parameter that defines the margin based on which the winners are determined. Unlike WTA, multiple neurons are returned as the output of the θ-WTA process according to the value of the θ parameter. Consequently, the θ-WTA process does not suffer from the convergence issue in the cases that there is more than one maximum value.

2.3.2.2 Overt Attention Modeling Using Learning Methods

Attention has a controlling role to decide where to look for gathering information to resolve the recognition task difficulties. Similarly, learning methods are used to mimic human behavior with visual attention capabilities. For this matter, a recognition system is composed of several components such as a visual representation, a classifier, and an attentional controller modules.

Based on the hypothesis that visual attention is a systematic approach of dealing with the massive amount of sensory stimulus, [89] proposes to utilize a generative model based on the deep learning framework. On one side, generative models can deal with occlusion and missing parts more easily than discriminative models. Furthermore, prior knowledge such as lighting can be incorporated in the forms of structured latent variables. On the other side, they suffer from proper scalability to process moderate size of input data in contrast to powerful discriminative models such as convolutional networks. This is the motivation behind studying the possibility of visual attention modeling with generative models. In other words, an attentional framework is proposed to infer the region of interest over the large input image on which the generative model is deployed to recognize object labels. The attention module in this work is inspired by Dynamic Routing [90, 91].

Gaussian Restricted Boltzmann Machines (GRBM) are used as the core recognition model. Stacking layer-wise GRBMs on top of each other would lead to a Gaussian Deep Belief Network (GDBN). Such a multi-layer architecture is used as the representation and classification model. The dynamic routing approach has a 2D similarity transformation with four parameters, two for position, one for rotation, and one for scale to characterize the canonical image (region of interest) over the big image. This canonical image is the input to the GDBN. The joint distribution of the patch window, and the transformation parameters can be formulated such that it easily factorizes into the conditional distribution, that models the top-down generative process of attention by a

Gaussian distribution, and the two priors.

[92] proposes an approach to define a learnable module to implement the idea of dynamic routing that fits within the framework of convolutional networks. The module brings into account the capability of learning spatial transformations based on which conceptually invariance representations to translation, scale, rotation, and warping can be achieved. Since this module is defined to be differentiable, its parameters can be learned during the typical learning procedure of convolutional networks without any extra supervision. Beside having the capacity of bringing the intermediate representation to a more understandable pose, the spatial transformer module can be seen as an implementation of selective spatial attention that dynamically selects the best part of the representation to fulfill inference in the subsequent layers. However, attention is always driven by the task knowledge from higher layers while in this approach, the parameters are learned and fixated in the feed forward manner.

The spatial transformer module consists of three collaborative sub-modules: localization network, grid generator, and the sampler. The localization network is a neural network with a few layers that takes the feature map at layer below as the input, and produces spatial transformation parameters as the outputs. The grid generator is responsible for calculating the output coordinates at which the input feature maps has to be sampled. These coordinates are produced using the spatial transformation parameters derived from the localization network. The sampling sub-module receives both the input feature map and the sampling grid to generate the output feature map. It is achieved by sampling the input feature map at the generated sampling coordinates using a predefined sampling kernel. The spatial transformer is purposefully designed to be thoroughly differentiable so then it can naturally integrate into any part of convolutional networks.

There are some attempts in the recent literature to bring into consideration the fact that the retina resolution falls off the farther it gets from the fovea. In such a paradigm, the problem of object recognition changes inherently with new highlighted issues. The very first one is how to decide on the sequence of fixations. Thus the task of visual object recognition changes to a combination with decision making processes over sequential data. Secondly, there must be some kind of visual short-term memory to accumulate and integrate information from different glimpses such that the objects get identified and noise surrounding them discarded.

A third-order RBM is proposed to learn and accumulate features over glimpses [93]. A glimpse is the set of features extracted from the retinal input according to [93]. Factored higher-order RBMs were first introduced in [94] in the attempt to learn spatial transformations from two images. The connections are between the visible units representing glimpses, the hidden units for accumulated features, and position-dependent units to gate connections between visible and hidden units. A retinal transformation is defined to map visual information of the input image according to the retina eccentricity fall-off property. The classification problem simply changes to the prediction of the image label from a few glimpses rather than the whole high resolution image.

Following the same paradigm for modeling the sequences of fixations over the retinal transformation of the input image, [95] proposes to use a neural feedforward auto-regressive model in the replacement of RBM. The argument to support such replacement is primarily due to the intractability of the gradient estimations for RBMs. On the other hand, the proposed model has a feed-forward architecture to target deep structures for the hope of better representation learning [96].

The entire modeling approach using generative models shows how to unify a fixation controller with the retinotopic representation learning under one framework. Despite the attempt to reconcile the heavy computational demand of generative models by retinotopic transformation and fixation controller as means of attentional modeling, the experimental setups and results shed serious doubt on the applicability and competence of this direction with the fast trending developments of convolutional networks. However, the ideas of the fixation controller and retinotopic transformation can be adapted to a discriminative modeling paradigm.

As an attempt to model the eye movement requirement in overt attention to collect extra information across the visual field, [97] proposes a multi-fixation model to integrate visual information over several glimpses. The goal is to reduce the uncertainty of the predicted category of the input image by the proposed multi-fixation mechanism. Processing large size images for invariant object recognition are the main issue that is addressed by learning where to look next sequentially.

The computational complexity of convolutional networks is non-linear in the number of pixels of the input image. Inspired by the observation that processing of very large and detailed visual scene images cannot be approached easily with such computationally demanding models, an attention-based neural network framework is proposed in [98]. Attention is thus seen as a control mechanism to reduce the search complexity of the visual processing regardless of the task in hand [98]. The Reinforcement learning framework is fully employed to conceptualize the visual attention modeling for object recognition as an action/reward concept of an agent interacting with an environment. The idea simply is to provide a general attention-based process to be applied to different tasks.

Following the same attention modeling paradigm, [99] shows how multiple object recognition can be achieve as the consequence of this approach. The idea is to learn to localize and recognize by deploying attention simultaneously for multiple objects while there is only class label available at training time. Briefly, the model has five sub-networks working altogether. The glimpse network is a three layer convolutional network to represent the input glimpse patch. Its output is fed into the recurrent network which in essence is acting as a short-term memory. The output of the recurrent network is sent to the emission network where the decision to where the glimpse network needs to look is made. The location of the first glimpse is chosen by a separate network called Context network. There is a classification network where the category label prediction is generated from the current state of the recurrent network.

The argument to compete with convolutional network using networks that learn over sequential data using fewer parameters and less computation is getting popular in recent papers. However, the object recognition task is very hard using such shallow representations unless attention modeling is combined with a deep representation. An integration of discriminative and generative processing pathways is proposed to model both the what and the where problems in object recognition using the auto-encoder paradigm [100]. It creates a coupled convolutional network in the feedforward pathway with a Deconvolution network in the backward pathway. The idea simply is to harness the most out of the reconstruction from where to learn better features to encode the what problem. Thus it is different from any usage of the where problem for object localization.

The learning procedure is achieved through an end-to-end optimization of the objective function with the reconstruction penalization of the hidden layers. A compositional loss function to account for both what and where problems consisting of three terms is defined. One is the negative log-likelihood for discrimination, the second is the reconstruction at the input level, and the third is the reconstruction at the middle levels of the feature maps. The benefits gained with this middle-level reconstruction penalty term are the perseverance of the correspondence with a particular unit in a feature map to measure the reconstruction error and the enforcement of the middle layer to participate in the reconstruction penalization rather than only the two end layers.

In addition to the feedforward role that the pooling layers commonly have, they further collect the pooling position switches to provide deterministic reconstruction in the feedback pathway. In contrast, to the sampling method of contrastive divergence algorithm to train RBMs, the gradient descent of the objective function through the popular backpropagation algorithm is enough to train such architectures.

Chapter 3

Top-Down Selection for Localization

The work in this chapter has been published previously as the following:

Mahdi Biparva and John K. Tsotsos, “STNet: Selective Tuning of Convolutional Networks for Object Localization”, in *The IEEE International Conference on Computer Vision Workshop on Mutual Benefits of Cognitive and Computer Vision (MBCC)*, 2017

and is presented here with no further changes and modifications.

3.1 Abstract

Visual attention modeling has recently gained momentum in developing visual hierarchies provided by Convolutional Neural Networks. Despite recent successes of feed-forward processing on the abstraction of concepts from raw images, the inherent nature of feedback processing has remained computationally controversial. Inspired by the computational models of covert visual attention, we propose the Selective Tuning of Convolutional Networks (STNet). It is composed of two streams of Bottom-Up and Top-Down information processing to selectively tune the visual representation of convolutional neural networks. We experimentally evaluate the performance of STNet for the weakly-supervised localization task on the ImageNet benchmark dataset. We demonstrate that STNet not only successfully surpasses the state-of-the-art results but also generates attention-driven class hypothesis maps.

3.2 Introduction

Inspired by physiological and psychophysical findings, many attempts have been made to understand how the visual cortex processes information throughout the visual hierarchy [101, 102]. It is significantly supported by reliable evidence [12, 22] that information is processed in both directions throughout the visual hierarchy: The Bottom-Up (BU) pass in a hierarchical visual representation is defined as the transformation of the sensory input data into high-level abstract semantic information through a cascade of feature extraction layers. In a convolutional neural network, the input data is passed through a series of processing layers, each composed of convolutional, pooling, and non-linearity, from the bottom to the top of the neural network. Parametric layers in the BU pass contains connection weight parameters that are optimized during the training phase to minimize the task loss function. The representational power of the BU pass is defined relative to the benchmark datasets on which the model is trained.

The Top-Down (TD) pass, on the other hand, is based on the reverse flow direction. The pass begins from an initial signal at the top of the hierarchy, and it goes all the way down to the early low-level TD layers. The TD pass does not go through any parametric layers such as convolutional layers. However, connection weights are computed locally at each TD layer using not only the localized receptive field activities but also the network kernel filters. The local computation has three stages: selection as a result of neural competition emerges, grouping highlights important nodes, and the final normalized propagation update output gating nodes.

In recent years, while the learning approaches have matured, various models and algorithms have been developed to present a richer visual representation for various visual tasks such as object classification and detection, semantic segmentation, action recognition, and scene understanding [103, 104]. Regardless of the algorithms used for representation learning, most attempts benefit from BU processing paradigm, while TD processing has very rarely been targeted particularly in the computer vision community. In recent years, convolutional neural networks, as a BU processing structure, have shown to be quantitatively very successful on the visual tasks targeted by popular benchmark datasets [30, 49, 50, 105].

Attempts in modeling visual attention are attributed to the TD processing paradigm. The idea is using some form of facilitation or suppression, the visual representation is selected and modulated in a TD manner [7, 106]. Visual attention has two modes of execution [6, 107]: Overt attention attempts to compensate for the lack of visual acuity throughout the entire field of view in a perception-cognition-action cycle by the means of an eye-fixation controller. In nutshell, the eye movement keeps the highest visual acuity at the fixation while leaving the formed visual representation intact. Covert attention, on the other hand, modulates the shaped visual representation, while keeping the fixation point unchanged.

We strive to account for both the BU and TD processing in a novel unified framework by proposing STNet, which integrates attentive selection processes into the hierarchical representation. STNet has been experimentally evaluated on the task of object localization. Unlike all previous approaches, STNet considers the biologically-inspired method of surround suppression [77] to selectively deploy high-level task-driven attention signals all the way down to the early layers of the visual hierarchy. The qualitative results reveal the superiority of STNet on this task over the performance of the state-of-the-art baselines.

We propose to use the Selective Tuning (ST) computational model of visual attention [21, 7] as the basis for the TD selection processes that complements the BU processes of a typical convolutional neural network. In both BU and TD passes, the information at one particular layer is acquired from the previous layer, processed at the layer, and then sent to the next layer in the hierarchy.

The flow direction, though, is different for each pass, and the previous and next layers are accordingly defined. In the TD pass, for a layer L_k, the previous layer is the top layer L_{k+1} and the next layer is the bottom layer L_{k-1} within the hierarchy while the reverse is true for the BU pass. The BU pass benefits from learnable connection weight parameters for feature transformation such as convolutional kernels while the TD pass has no such type of weight parameters. Instead, there are adaptive threshold parameters that rule the selection process properties, which are determined as the result of competitions between input values at each TD layer. We hereafter call processing units in the BU and TD passes hidden and gating units respectively. Next, we explain the input, the attention selection process, and the output of TD layer.

Every TD layer expects two input arguments: one is the hidden activities at the layer below, which are retrieved via a localized Receptive Field (RF), and the other is the kernel weights of the BU layer. Given these two arguments, the input to the attention selection process is determined by the Hadamard matrix product of the kernel weights with the localized RF activities for a particular gating unit. The product result is referred to as the Post-Synaptic (PS) activities. PS activities are localized for each gating unit at particular layers, spatial locations, and feature maps. The output of the selection process is also localized. Therefore, local output values are written to their corresponding units in the next TD layer.

Given the localized PS activities to the selection process at a particular TD layer, we define a cascade of three computational stages. The first one is proposed to reduce noise interference by pruning redundant PS activities. The selection pattern is the result of competitions between PS units such that the value of the top gating activity is either retained or increased. The second stage is defined to group and select the most important input units. For a convolutional layer, spatial contiguity matters while for a fully-connected layer, statistical importance is measured. The final stage is defined to normalize the selected local PS units such that they sum to one, and then propagate the activity of the top gating unit proportional to the normalized selected PS activities to the gating units of the next layer.

The initial gating activities at the hierarchy top layer starts with the execution of the TD pass. Consequently, a number of TD layers are sequentially processed, and the TD pass terminates at some early layer. The gating activities of the final TD layer are retrieved and post-processed to produce the desired task output. This proposed formulation of a TD selection mechanism for convolutional neural networks is called STNet [108].

Various approaches based on error gradient propagation with respect to the input variable are dominant for object localization and representation visualization [1, 109, 2]. These approaches use dense TD traversal to early layers and at a late post-processing stage, perform selection of salient regions for object localization. STNet, on the other hand, is inherently selective throughout the visual hierarchy. This property implies sparse representation and faster TD traversal while improving the localization accuracy.

We evaluate the performance of the proposed network on the weakly-supervised object localization task [59, 110]. In this task, the ultimate goal is to localize objects within the input image with no extra parameter fine-tuning. STNet outperforms the state-of-the-art methods on this task [108]. The evaluation is measured using the Intersection-over-Union (IoU) metric. If the IoU of the proposed box with the ground truth box is greater than some threshold value (0.5), that prediction is marked a successful prediction. The localization accuracy is then the average number of successful predictions across the unseen validation set.

3.3 Related Work

Various attempts have been made to model an implicit form of covert attention on convolutional neural network s for representation visualization and weakly-supervised object localization. [59] proposes to maximize the class score over the input image using the backpropagation algorithm for the visualization purposes. [58] introduces an inverted convolutional neural network to propagate backward hidden activities to the early layers. Harnessing the superiority of global *AVERAGE* pooling over global *MAX* pooling to preserve spatial correlation, [1] has defined a weighted sum of the activities of the convolutional layer feeding into the global pooling layer. Recently, an explicit notion of covert visual attention has gained interest in the computer vision community [109, 2] for the weakly-supervised localization task. Having interpreted *ReLU* activation and *MAX* pooling layers as feedforward control gates, [109] proposes feedback control gate layers which are activated based on the solution of an optimization problem. Inspired closely by Selective Tuning model of visual attention, [2] formulates TD processing using a probabilistic interpretation of the *Winner-Take-All* (WTA) mechanism. In contrast to all these attempts that the TD processing is densely deployed in the same fashion as BU processing, we propose a highly sparse and selective TD processing in this work.

The localization approach in which the learned representation of the visual hierarchy is not modified is commonly referred to as weakly supervised object localization [59, 110, 1, 109, 2]. This is in contrast with the supervised localization approach in which the visual representation is fine-tuned to better cope with the new task requirements. Additionally, unlike the formulation for the semantic segmentation task [111, 112, 113], bounding box prediction forms the basis of performance measure. We evaluate experimentally STNet in this paradigm and provide evidence that selective tuning of convolutional neural networks better addresses object localization in the weakly-supervised regime.

3.4 Model

3.4.1 STNet

An integration of the conventional bottom-up processing by convolutional neural networks with the biologically-plausible attentive top-down processing in a unified model is proposed in this work. STNet consists of two interactive streams of processing: The BU stream has the role of forming the representation throughout the entire visual hierarchy. Information is very densely processed layer by layer in a strict parallel paradigm. The BU pathway processes information at each layer using a combination of basic operations such as convolution, pooling, activation, and normalization functions. The TD stream, on the other hand, develops a projection of the task knowledge onto the formed hierarchical representation until the task requirements are fulfilled. Depending on the type of the task knowledge, the projections may be realized computationally using some primitive stages of attention processing. The cascade flow of information throughout both streams is layer by layer such that once information at a layer is processed, the layer output is fed into the next adjacent layer as the input according to the hierarchical structure.

Any computational formulation of the visual hierarchy representing the input data can be utilized as the structure of the BU processing stream as long as the primary visual task could be accomplished. Convolutional neural networks trained in the fully supervised regime for the primary task of object classification are the main focus of this chapter. Having STNet composed of a total of L layers, the BU processing structure is composed of $\forall l \in \{0, \ldots, L\}, \exists \, \mathbf{z}^l \in \mathbb{R}^{H^l \times W^l \times C^l}$, where $\mathbf{z}^l$ is the three dimensional feature volume of hidden nodes at layer l with the dimension of width W^l, height H^l and C^l number of channels.

3.4.2 Structure of the Top-Down Processing

Based on the topology and connectivity of the BU processing stream, an interactive structure for the attentive TD processing is defined. According to the task knowledge, the TD processing stream is initiated and consecutively traversed downward layer by layer until the layer that satisfies task requirements is reached. A new type of node is defined to interact with the hidden nodes of the BU processing structure. According to the TD structure, gating nodes are proposed to collectively determine the TD information flow throughout the visual hierarchy. Furthermore, they are very sparsely active since the TD processing is tuned to activate relevant parts of the representation.

The TD processing structure consists of $\forall l \in \{0, \dots, L\}, \exists \, \mathbf{g}^l \in \mathbb{R}^{H^l \times W^l \times C^l}$, where $\mathbf{g}^l$ is the three dimensional (3D) gating volume at layer i having the exact size of its hidden feature volume counterpart in the BU processing structure. We define the function $RF(z)$ to return the set of all the nodes in the layer below that falls inside the receptive field of the top node according to the connectivity topology of the BU processing structure.

Having defined the structural connectivity of both the BU and TD processing streams, we now introduce the attention procedure that locally processes information to determine connection weights of the TD processing structure and consequently the gating node activities at each layer. Once the information flow in the BU processing stream reaches the top of the hierarchy at layer L, the TD processing is initiated by setting the gating node activities of the top layer as illustrated in Fig. 3.1. Weights of the connections between the top gating node g_L and all the gating node in the layer below within the $RF(g_L)$ are computed using the attentive selection process. Finally, the gating node activities of layer $L - 1$ are determined according to the connection weights. This attention procedure is consecutively executed layer by layer downward to a layer at which the task requirements are fulfilled.

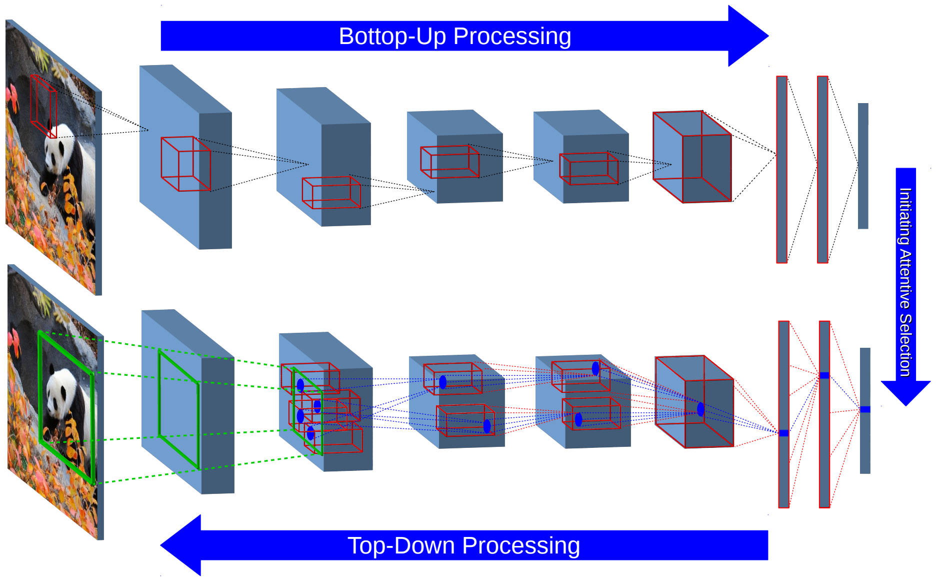


Figure 3.1: STNet consists of both BU and TD processing streams. In the BU stream, features are collectively extracted and transferred to the top of the hierarchy at which label prediction is generated. The TD processing (bottom), on the other hand, selectively activate part of the structure using attention processes. Figure schematically illustrates AlexNet architecture. The middle blue boxes represent hidden and gating activity tensors on the BU and TD pathways. The last three squares represent fully-connected layers. Receptive fields are schematically depicted with small red boxes. The blue circles illustrate the selection regions that the information propagates to in the TD stream.

3.4.3 Stages of Attentive Selection

Weights of the connections of the BU processing structure are learned by the backpropagation algorithm [53] in the training phase. For the TD processing structure, however, weights are computed in an immediate manner using the deterministic and procedural selection process from the Post-Synaptic (PS) activities. We define $\forall\, g^l_{w,h,c} \in \mathbf{g}^l$, $PS(g^l_{w,h,c}) = RF(z^l_{w,h,c}) \odot k^l_c$, where $PS(g)$ is the Hadamard product (element-wise) of two similar-size matrices, one representing the receptive field activities, and the other, the kernel at channel c and layer l.

The selection process has three stages of computation. Each stage processes the input PS activities and then feeds the selected activities to the next stage. In the first stage, noisy redundant activities that interfere with the definition of task knowledge are pruned away. Second, among the remaining PS activities, the most informative group of activities are marked as the winners of the selection process at the end of the second stage. In the third stage, the winner activities are normalized. Once multiplicatively biased by the top gating node activity, the activities of the bottom gating nodes are updated consequently. Fig. 3.2 schematically illustrates the flow sequence beginning from passing PS activities from the BU stream to the attention process and then propagating weighted activities of the top gating node to the lower layer. Fig. 3.3 demonstrates the different computational building blocks of the attention process module in detail. Different computational operations in the BU and TD processing streams along with passing PS activities to the TD stream are shown in the figure.

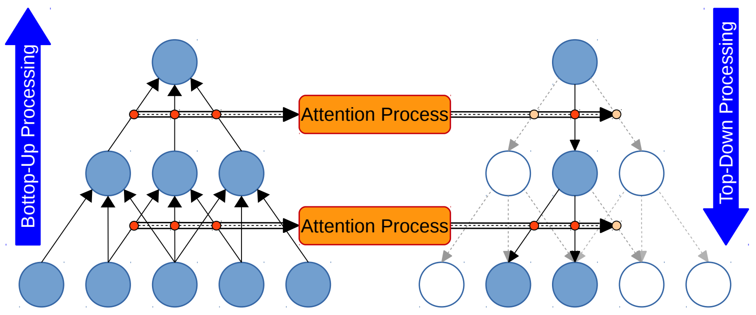


Figure 3.2: Schematic Illustration of the sequence of interactions between the BU and TD processing streams using the three-stage attention process.

Stage 1: Interference Reduction

The main critical issue to accomplish successfully any visual task is to be able to distinguish relevant regions from irrelevant ones. Winner-Take-All (WTA) is a biologically-plausible mechanism that implements a competition between input activities. At the end of the competition, the winner retains its activity, while the rest become inactive. The Parametric WTA (P-WTA) using the parameter θ is defined as $P\text{-}WTA(PS(g), \theta) = \{s \mid s \in PS(g), s \geq WTA(PS(g)) - \theta\}$. The role of the parameter θ is to establish a safe margin from the winner activity to avoid under-selection such that multiple winners will be selected at the end of the competition. It is critical to have some near optimal selection process at each stage to prevent the under- or over-selection extreme cases.

We propose an algorithm to tune the parameter θ for an optimal value at which the safe margin is defined based a biologically-inspired approach. It is biologically motivated that once the visual attention is deployed downward to a part of the formed visual hierarchy, those nodes falling on the attention trace

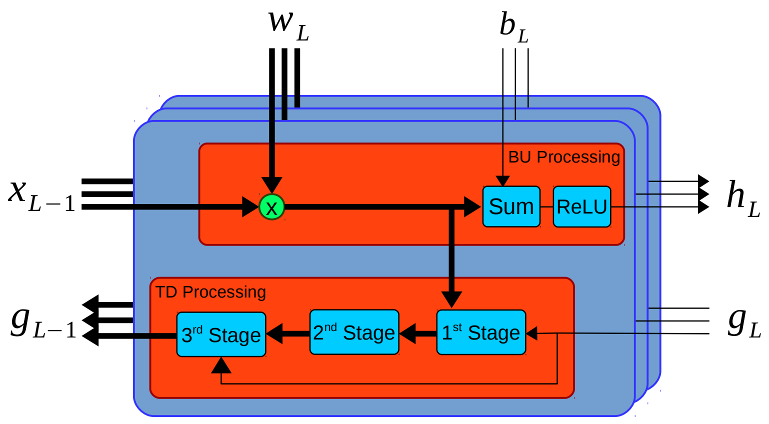


Figure 3.3: Modular diagram of the interactions between various blocks of processing in both the BU and TD streams. The arrow direction shows the flow of the information to each computational block. The layers schematically represent that the BU and TD processing is done on feature maps with spatial and channel dimensions. Thick arrows represent vector values while thin arrows represent scalar values

will eventually retain their node activities regardless of the intrinsic selective nature of attention mechanisms [114, 7]. In analogy to this biological finding, the Activity Preserve (AP) algorithm optimizes for the distance from the sole winner of the WTA algorithm at which if all the PS activities outside the margin are pruned away, the top hidden node activity will be preserved.

Algorithm 1 specifies the upper and lower bounds of the safe margin. The upper bound is clearly indicated by the sole winner given by the WTA algorithm, while the lower bound is achieved by the output of the AP algorithm. Consequently, the P-WTA algorithm returns all the PS activities that fall within this range specified by the upper and lower bound values. They are highlighted as the winners of the first stage of the attentive selection process defined by the set W^{1st}. We will refer to the set of winners at the end of the 1st and 2nd stage with W^{1st} and the set W^{2nd} terms respectively. Basically, W^{1st} returned from P-WTA algorithm, contains those nodes within the receptive field that most participate in the calculation of the top node activity. Therefore, they are the best candidates to initiate the attentive selection processes of the layer below. The size of the set of winners at this point, however, is still large. Apparently, further stages of selection are required to prohibit interference and redundant TD processing caused by the over-selection phenomenon.

Algorithm 1 Parametric WTA Optimization

```
1: $NEG(PS) = \{s \mid s \in\ PS(g), s \leq 0\}$
2: $POS(PS) = \{s \mid s \in\ PS(g), s > 0\}$
3: $SUM(NEG) = \sum_{n_i \in NEG(PS)} n_i$
4: $buffer = SUM(NEG)$
5: $i = 0$
6: while $i \leq |POS(PS)|$, $buffer < \epsilon$ do
7:     $buffer+ = SORT(POS(PS))[i]$
8:     $i+ = 1$
9: end while
10: return $SORT(POS(PS))[i-1]$
```

Stage 2: Similarity Grouping

In the second stage, the ultimate goal is to apply a more restrictive selection procedure in accordance with the rules elicited from the task knowledge. Grouping of the winners according to some similarity measure serves as the basis of the second stage of the attentive selection process. Two modes of selection at the second stage are proposed depending on whether the current layer of processing has a spatial dimension or not: Spatially-Contiguous(SC) and Statistically-Important(SI) selection modes respectively. The former is applicable to the Convolutional layers and the latter to the Fully-Connected(FC) layers in a typical convolutional neural network.

There is no ordering information between the nodes in the FC layers. Therefore, one way to formulate the relative importance between nodes is using the statistics calculated from the sample distribution of node activities. SI selection mode is proposed to find the statistically important activities. Based on an underlying assumption that the node activities have a Normal distribution, the set of winners of the second stage is determined by $W^{2nd} = \{s|\, s \in W^{1st}, s > \mu + \alpha * \sigma\}$, where μ and σ are the sample mean and standard deviation of W^{1st} respectively. The best value of the coefficient α is searched over the range $\{-3, -2, -1, 0, +1, +2, +3\}$ in the second stage based on a search policy meeting the following criteria: First, the size of the winner set W^{2nd} at the end of the SI selection mode has to be non-zero. Second, the search iterates over the range of possible coefficient values in a descending order until $|W^{2nd}| \neq 0$. Furthermore, an offset parameter O is defined to loosen the selection process at the second stage once these criteria are met. For instance, if α is $+1$ at the end of the SI selection mode, the loosened α will be -1 for the offset value of 2. The effects of loosening SI selection mode is experimentally demonstrated in Sec. 3.5.

Convolutional layers, on the other hand, benefit from stacks of two dimensional feature maps. Although the ordering of feature maps in the third dimension is not meant to encode for any particular information, 2D feature maps individually highlight spatial active regions of the input domain projected into a particular feature space. In other words, the spatial ordering is always preserved throughout the hierarchical representation. With the spatial ordering and the task requirement in mind, SC selection mode is proposed to determine the most spatially contiguous region of the winners based on their PS activities.

SC selection mode first partitions the set of winners W^{1st} into groups of connected regions. A node has eight immediate adjacent neighbors. A connected region R_i, therefore, is defined as the set of all nodes that are recursively in the neighborhood of each other. Out of all the number of connected regions, the output of the SC selection mode is the set of nodes W^{2nd} that falls inside the winner connected region. It is determined by the index i such that $\hat{i} = \arg\max_i \alpha * (\sum_{r_j \in R_i} PS_{r_j}(g)) + (1-\alpha) * (|R_i|)$, where $PS_{r_j}(g)$ is the PS activity of node r_j among the set of all PS activities of the top node g. It shows the overall strength of the winner region. The value of multiplier α is cross-validated in the experimental evaluation stage for the best balance between the strength and size of the winner regions. $|R_i|$ is the total number of nodes in the winner connected region i and shows the size of region. $\alpha = 1$ implies a selection policy that only relies on the strength of the connected regions while $\alpha = 0$ only counts the size of the regions into account. Lastly, SC selection mode returns the final set of winners $W^{2nd} = \{s|\ s \in\ R_{\hat{i}}\}$. The argument that W^{2nd} could better address task requirements in comparison to W^{1st} is experimentally supported in 3.5.

Having determined the set of winners W^{2nd} out of the set of all nodes falling inside the receptive field of the top node $RF(g)$, it is straightforward to compute values of the both active and inactive weight connections of the TD processing structure. The inactive weight connections have value zero. In Stage 3, the mechanism to set the values of the active weight connections from

Architecture	L_{prop}	O_{FC}	O_{Bridge}	α	δ_{post}
ST-AlexNet	pool1	3	3	0.2	μ_A
ST-VGGNet	pool3	2	0	0.2	μ_A
ST-GoogleNet	pool2/3x3_s2	0	-	0.2	μ_A

Table 3.1: Demonstration of the STNet configurations in terms of the hyper-parameter values. L_{prop} is the name of the layer at which the attention map is calculated. O_{FC} and O_{Bridge} are the offset values of the SI selection mode at the fully-connected and bridge layers respectively. α is the trade-off multiplier of the SC selection mode. δ_{post} represents the post-processing threshold value of the attention map.

W^{2nd} will be described.

Stage 3: Attention Signal Propagation

Gating nodes are defined to encode for attention signals using multiple level of activities. The top gating node propagates the attention signal proportional to the normalized connection weights to the layer below. Having the set of winners W^{2nd} for the top gating node g, $PS_{W^{2nd}}(g)$ is the set of PS activities of the corresponding winners. The set of normalized PS activities is defined as $PS_{norm} = \{\hat{s}|\ s \in\ PS_{W^{2nd}}(g),\ \hat{s} = s/\sum_{s_i \in PS_{W^{2nd}}(g)} s_i\}$. Weight values of the active TD connections are specified as follows: $\forall\, i \in W^{2nd}, w_{ig} = PS^i_{norm}$, where w_{ig} is the connection from the top gating node g to the gating node i in the layer below, and PS^i_{norm} is the PS activity of the winner node i.

At each layer, the attentive selection process is performed for all the active top gating nodes. Once the winning set for each top gating node is determined and the normalized values of the corresponding connection weights to the layer below are computed, the winner gating nodes of the layer below are updated as follows: $\forall i \in \{1, \ldots, |\mathbf{g}^l|\}, \forall j \in \{1, \ldots, |W^{2nd}_i|\}, g^{l-1}_j += w_{ji} * g^l_i$. The updating rule ensures that the top gating node activity is propagated downward such

Model	AlexNet	VGGNet	GoogleNet
Oxford[59]	-	-	44.6
CAM[1]	48.3[1]	48.1[1]	48.1[2]
Feedback[109]	49.6	40.2	38.8
MWP[2]	41.7[1]	40.6[1]	38.7
STNet	**40.3**	**40.1**	**38.6**

Table 3.2: Comparison of the STNet localization error rate on the ImageNet validation set with the previous state-of-the-art results. The bounding box is predicted given the single center crop of the input images with the TD processing initialized by the ground truth category label. (1) Results calculated using the publicly published code by [1, 2]. (2) Based on the result reported by [2]. Otherwise, the results are reported by the reference work cited on the left.

that it is multiplicatively biased by weight values of the active connections.

3.5 Experimental Results

Top-down visual attention seems necessary for the completion of sophisticated visual tasks for which only Bottom-Up information processing is not sufficient. This implies that tasks such as object localization, visual attribute extraction, and part decomposition require more processing time and resources. STNet, as a model benefiting from both streams of processing, is experimentally evaluated on object localization task in this work.

STNet is implemented using Caffe [115], a library originally developed for convolutional neural network s. AlexNet [30], VGGNet(16) [59], and GoogleNet [49] are the three convolutional neural network architectures that are applied to define the BU processing structure of STNet. The model weight parameters are retrieved from the publicly available convolutional neural network repository of Caffe Model Zoo in which they are pre-trained on ImageNet

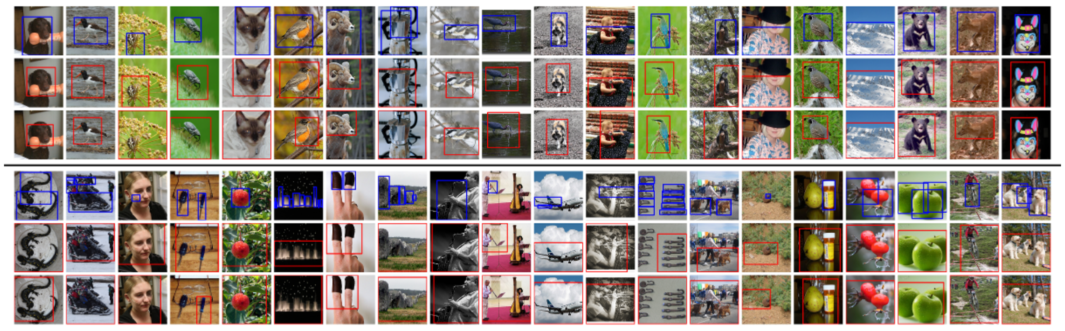


Figure 3.4: Illustration of the predicted bounding boxes in comparison to the ground truth for ImageNet images. In the top section, STNet is successful to localize the ground truth objects. The bottom section, on the other hand, demonstrates the failed cases. The top, middle, and bottom rows of each section depict the bounding boxes from the ground truth, ST-VGGNet, and ST-GoogleNet respectively.

2012 classification training dataset [116]. For the rest of the paper, we refer to STNet utilized with AlexNet as the base architecture of the BU structure as ST-AlexNet. This similarly applies to VGGNet and GoogleNet.

3.5.1 Implementation Details

Bounding Box Proposal: Having an input image fed into the BU processing stream, a class specific attention map for category k at layer l is created. It is a resultant of the TD processing stream initiated from the top gating layer with the one-hot encoding of category k. Once the attention signals are completely propagated downward to layer l, the class specific attention map is defined by collapsing the gating volume $\mathbf{g}^l \in \mathbb{R}^{H^l \times W^l \times C^l}$ at the third dimension into the attention map $A^l_k \in \mathbb{R}^{H^l \times W^l}$ as follows: $A^l_k = \sum_{i \in C^l} g^l_i$, where C^l is the number of gating sheets at layer l, and g^l_i is a 2D gating sheet. We propose to post-process the attention map by setting all the small collapsed values below the sample mean value of the map to zero.

We propose to predict a bounding box from the thresholded attention map $\hat{A}_k^l$ using the following procedure. Apparently, the predicted bounding box is supposed to enclose an instance of the category k. If layer l is somewhere in the middle of the visual hierarchy, $\hat{A}_k^l$ is transformed into the spatial space of the input layer. In the subsequent step, a tight bounding box around the non-zero elements of the transformed $\hat{A}_k^l$ is calculated. Nodes inside the RF of the gating nodes at the boundary of the predicted box are likely to be active if the TD attentional traversal further continues processing lower layers. Therefore, we choose to pad the tight predicted bounding box with the half size of the accumulated RF at layer l. We calculate accurately the accumulated RF size of each layer according to the intrinsic properties of the BU processing structure such as the amount of padding and striding of the layer.

Search over Hyperparameters: There a few number of hyperparameters in STNet that are experimentally cross-validated using one held-out partition of the ImageNet validation set. It contains 1000 images which are selected from the randomly-shuffled validation set. The grid search over the hyperparameter space finds the best-performing configuration for each convolutional neural network architecture.

The SI selection mode is experimentally observed to perform more efficiently once the offset parameter O is higher than zero. The offset parameter has the role of loosening the selection process for the cases under-selection is very dominant. Furthermore, we define the bridge layer as the one at which the 3D volume of hidden nodes collapses into a 1D hidden vector. SI selection procedure is additionally applied to the entire gating volume of the bridge layer in order to prevent the over-selection phenomenon. Except GoogleNet, the other two architectures have a bridge layer. Further implementation details regarding all three architectures are given in the supplementary material in Sec. A.1.

Hyperparameters such as the layer at which the best localization result is obtained, the multiplier of the SC selection mode, and the threshold value for the bounding box proposal procedure are all set by the values obtained from the cross-validation on the held-out partition set for all three convolutional neural network s. Having the best STNet configurations given in Table 3.1, we measure STNet performance on the entire ImageNet validation set.

3.5.2 Weakly Supervised Localization

The significance of the attentive TD processing in STNet is both quantitatively and qualitatively evaluated on the ImageNet 2015 benchmark dataset for the object localization task. The experimental setups and procedures have been considerably kept comparative with previous works.

Dataset and evaluation: Localization accuracy of STNet is evaluated on the ImageNet 2015 validation set containing 50,000 images of variable sizes. The shortest side of each image is reduced to the size of the STNet input layer. A single center crop of the size equal to the input layer is then extracted and sent to STNet for bounding box prediction. In order to remain comparative with the previous experimental setups for the weakly supervised localization task [109, 2], the ground truth label is provided to initiate the TD processing. A localization prediction considers to be correct if the Intersection-over-Union (IoU) of the predicted bounding box with the ground truth is over 0.5.

Quantitative results: STNet localization performance surpasses the previous works with a comparative testing protocol on the ImageNet dataset. For all three BU architectures, Table 3.2 indicates that STNet quantitatively is on par with previous state-of-the-art approaches [59, 1, 109, 2] in two of the three cases considered, while in the third case modestly outperforms the previous best by 1.4%. The results imply that not only has the localization accuracy improved but also fewer nodes are active in the TD processing stream. The finding is in sharp contrast to all the previous approaches that densely propagate down information in the TD stream. STNet, on the other hand, is hierarchically selective intrinsically. This helps sparse processing in the TD stream and consequently implies faster processing speed.

Comparison with Previous Works: One of the factors distinguishing STNet from other approaches is the selective nature of the TD processing. In gradient-based approaches such as [59, 110, 117, 60], the gradient signals, which are computed with respect to the input image rather than the weight parameters, are deployed densely downward to the input layer. This approach suffers from an unconstrained propagation of gradient signals throughout the visual hierarchy. As a result, a good localization can be obtained through a harsh final thresholding. Deconvnet [58] is proposed to reverse the same type and extent of processing as the feedforward pass originally for the purpose of visualization. The Feedback model [109] defines a dense feedback structure that is iteratively optimized using a secondary loss function to maintain the label predictability of the entire network. Similarly, attention signals are densely propagated through positive weight connections biased by the normalized PS activities in the MWP model [2, 118]. Additionally, MWP suffers from the lack of the three-stage attentive selection process and leave the object localization to the last stage at which strong thresholding is necessary to obtain reliable bounding box predictions. In contrast, the TD structure of STNet remains fully inactive except for a small portion that leads to the attended region of the input image. We empirically verify that not only has the localization accuracy been improved in STNet, but also on average around 0.3% of the TD structure is active. This implies comparative localization results can be obtained with faster speed and less wasted amount of computation in the TD processing stream. Furthermore, it is worth noting that ST-AlexNet localization performance is very close to the two other high capacity models despite the shallow depth and simplicity of the network architecture.

Qualitative Analysis: The qualitative results provide insights on the strengths and weakness of STNet as illustrated in Fig. 3.4. Investigating the successful and failed cases, we are able to identify two extreme scenarios: under-selection and over-selection scenarios. The under-selection scenario is caused by the inappropriate learned representation or improper configuration of the TD processing, while the over-selection scenario mainly is due to either multi-instance or what we call *Correlated Accompanying Object* cases. A large bounding box enclosing multiple objects is proposed as a result of over-selection. Neither streams of STNet are tuned to systematically deal with

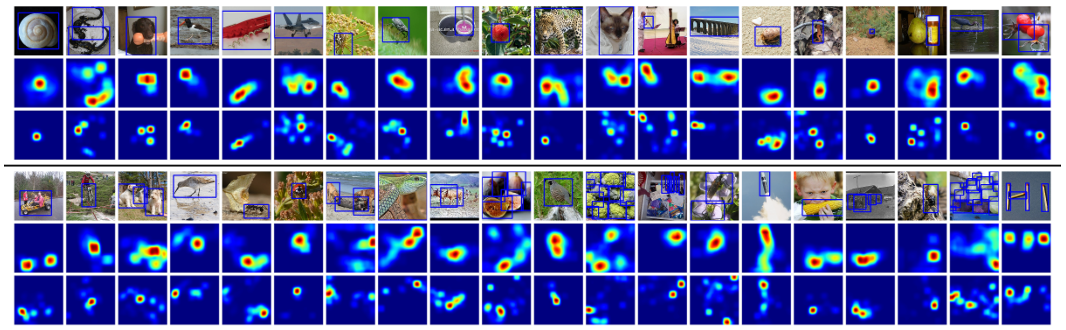


Figure 3.5: Demonstration of the attention-driven class hypothesis maps for ImageNet images. In both top and bottom sections, rows from top to bottom represent ground truth boxes on RGB images, the CH map from ST-VGGNet, and the CH map from ST-GoogleNet respectively.

these extreme scenarios.

3.5.3 Class Hypothesis Visualization

We show that gating node activities can further be processed to visualize the salient regions of input images for an activated category label. Following a similar experimental setup to the localization task in Table 3.1, an attention-driven *Class Hypothesis* (CH) map is created from the transformed thresholded attention map. We simply increment by one the pixel values inside the accumulated RF box centered at each non-zero pixel of the attention map. Once iterated over all non-zero pixels, the CH map is smoothed out using a Gaussian filter with the standard deviation $\sigma = 6$. Fig. 3.5 qualitatively illustrates the performance of STNet to highlight the salient parts of the input image once the TD processing stream is initiated with the ground truth category label. Further details regarding the visualization experimental setups are given in the supplementary material in Sec. A.1.

Comparison of convolutional neural networks: We observed in Sec. 3.5.2 that the localization performance of the ST-GoogleNet surpasses both ST-AlexNet and ST-VGGNet. The qualitative experimental results using CH maps in Fig. 3.5 further shed some light on the inherent nature of this discrepancy. Both AlexNet and VGGNet benefit from a coherently increasing RF sizes along the visual hierarchy such that at each layer all hidden nodes have a similar RF size. Consequently, the scale at which features are extracted coherently changes from layer to layer. On the other hand, GoogleNet is always taking advantage of intermixed multi-scale feature extraction at each layer. Additionally, 1x1 convolutional layers act as high capacity parametrized modules by which any affine mixture of features could be computed. In the TD processing, we treat such layers as regular fully-connected layers in all experiments.

Context Interference: The learned representation of convolutional neural networks strongly relies on the background context over which the category instances are superimposed for the category label prediction of the input image. This is expected since the learning algorithm does not impose any form of spatial regularization during the training phase. Fig. 3.6 depicts the results of the experiment in which we purposefully deactivated the second stage of the selection process at FC layers. Furthermore, the winner with the highest PS activity is remained active among all winners and the rest are set inactive at the end of the first stage of FC layers such that there is always one winner at each layer. Deactivating the second stage on the convolutional layers deteriorates the capability of STNet to sharply highlight the salient regions

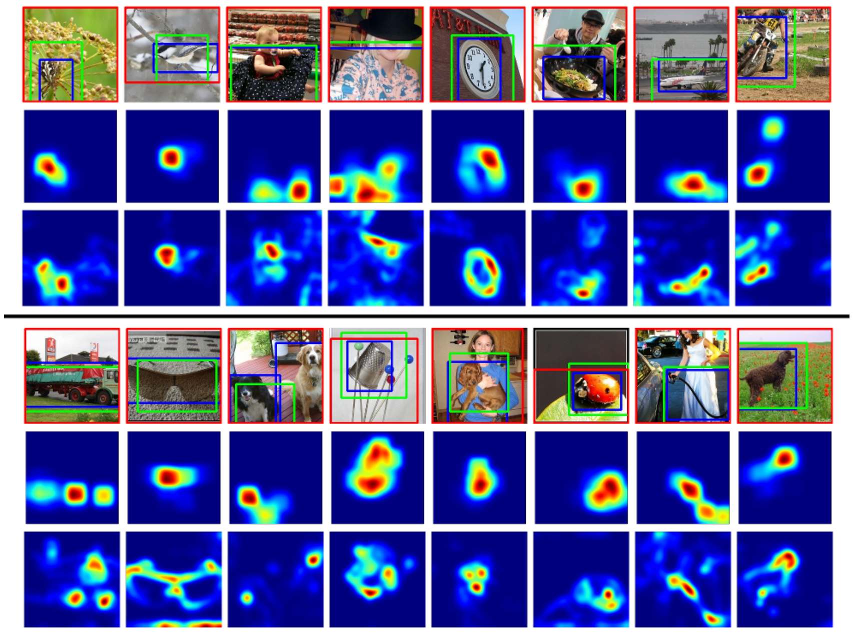


Figure 3.6: The critical role of the second stage of selection is illustrated using CH visualization. In the top row of each section, images are presented with boxes for the ground truth (blue), full-STNet predictions (green), and second-stage-disabled predictions (red). In the second and third rows of each section, CH maps from the full and partly disabled STNet are given respectively.

relevant to objects in the TD processing stream. The results implies that the learned representation heavily relies on the features collected across the entire image regardless of the ground truth. The SC mode of the second stage helps STNet to visualize the coherent and sharply localized confident regions. The CH visualization demonstrates the essential role of the second stage to deal with the redundant and distracting context noise for the localization task.

Correlated Accompanying Objects: The other shortcoming of the learned representation emphasized by CH visualization is that the BU processing puts high confidence on the features collected from the regions belong-

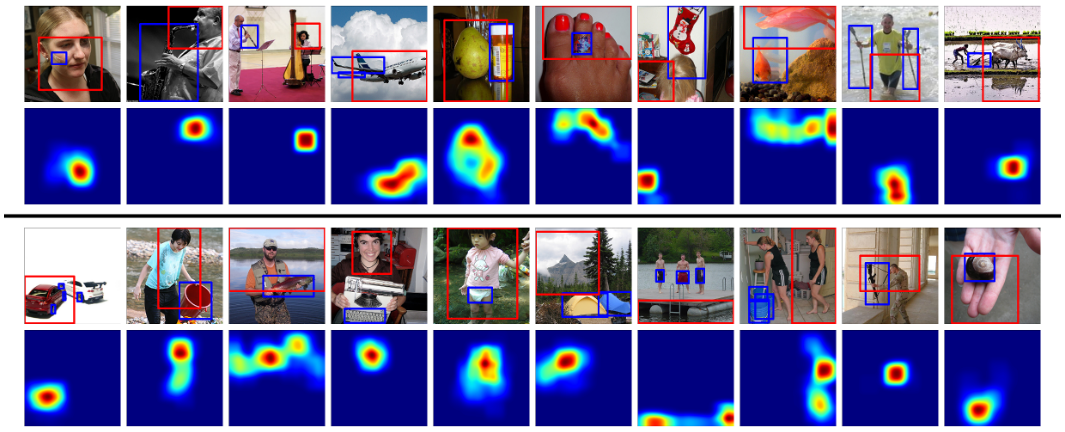


Figure 3.7: We demonstrate using ST-VGGNet the confident region of the accompanying object highly correlating with the true object category. The top row of each section contains images with the ground truth (blue) and predicted (red) boxes. CH maps highlight the most salient regions in the bottom row of each section.

ing to correlated accompanying objects. They happen to co-occur extremely frequently with the the ground truth objects in the training set on which convolutional neural network s are pre-trained. Similar to the previous experiment, the modified version of the first stage for FC layers is used, while the convolutional layers benefit from the original 3-stage selection process. Figure 3.7 reveals how STNet misleadingly localize with the highest confidence the accompanying object that highly correlates with the ground truth object. As soon as the visual representation confidently relates the correlated accompanying object with the true category label, over-selecting for the bounding box prediction will be inevitable. The multi-instance scenario and such cases are the two sources of the over-selection phenomenon in the localization task. We credit these two sources of over-selection to the pre-trained representation obtained from the unconstrained backpropagation learning algorithm.

3.6 Conclusion

We proposed an innovative framework consisting of the Bottom-Up and Top-Down streams of information processing for the task of object localization. We formulated the Top-Down processing as a cascading series of local attentive selection processes each consisting of three stages: First inference reduction, second similarity grouping, and third attention signal propagation. We demonstrated experimentally the efficiency, power, and speed of STNet to localize objects on the ImageNet dataset supported by the quantitative results that are on par with the state-of-the-art. Class Hypothesis maps are introduced to qualitatively visualize attention-driven class-dependent salient regions. Having investigated the difficulties of STNet in object localization, we believe the visual representation of the Bottom-Up stream is one of the shortcomings of this framework. The significant role of the selective Top-Down processing in STNet could be foreseen as a promising approach applicable in a similar fashion to other challenging computer vision tasks.

Chapter 4

Priming in Neural Network

The work in this chapter has been published previously as the following:

Amir Rosenfeld, Mahdi Biparva, John K. Tsotsos, "Priming Neural Networks", in *The IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Mutual Benefits of Cognitive and Computer Vision (MBCC)*, 2018

and is presented here with minor changes and modifications.

The contributions to this work are distributed among conceptualization, formulation, coding, and documentation. Amir Rosenfeld and Mahdi Biparva contributed equally to the conceptualization and abstraction of the novel idea in this work. The modeling and formulation of a possible implementation of the novel idea is equally contributed by the first two authors. Mahdi Biparva contributed 20% to the coding of the proposed formulation and conducting experimental evaluations while Amir Rosenfeld gets the credit for 80% contribution to this part. Lastly, Mahdi Biparva contributed 30% to the documentation of the work and the rest is done by Amir Rosenfeld.

4.1 Abstract

Visual priming is known to affect the human visual system to allow detection of scene elements, even those that may have been near unnoticeable before, such as the presence of camouflaged animals. This process has been shown to be an effect of top-down signaling in the visual system triggered by the said cue. In this paper, we propose a mechanism to mimic the process of priming in the context of object detection and segmentation. We view priming as having a modulatory, cue dependent effect on layers of features within a network. Our results show how such a process can be complementary to, and at times more effective than simple post-processing applied to the output of the network, notably so in cases where the object is hard to detect such as in severe noise. Moreover, we find the effects of priming are sometimes stronger when early visual layers are affected. Overall, our experiments confirm that top-down signals can go a long way in improving object detection and segmentation.

4.2 Introduction

Psychophysical and neurophysiological studies of the human visual system confirm the abundance of top-down effects that occur when an image is observed. Such top-down signals can stem from either internal (endogenous) processes of reasoning and attention or external (exogenous) stimuli- i.e. cues - that affect perception (cf. [7], Chapter 3 for a more detailed breakdown). External stimuli having such effects are said to *prime* the visual system, and potentially have a profound effect on an observer's perception. This often results

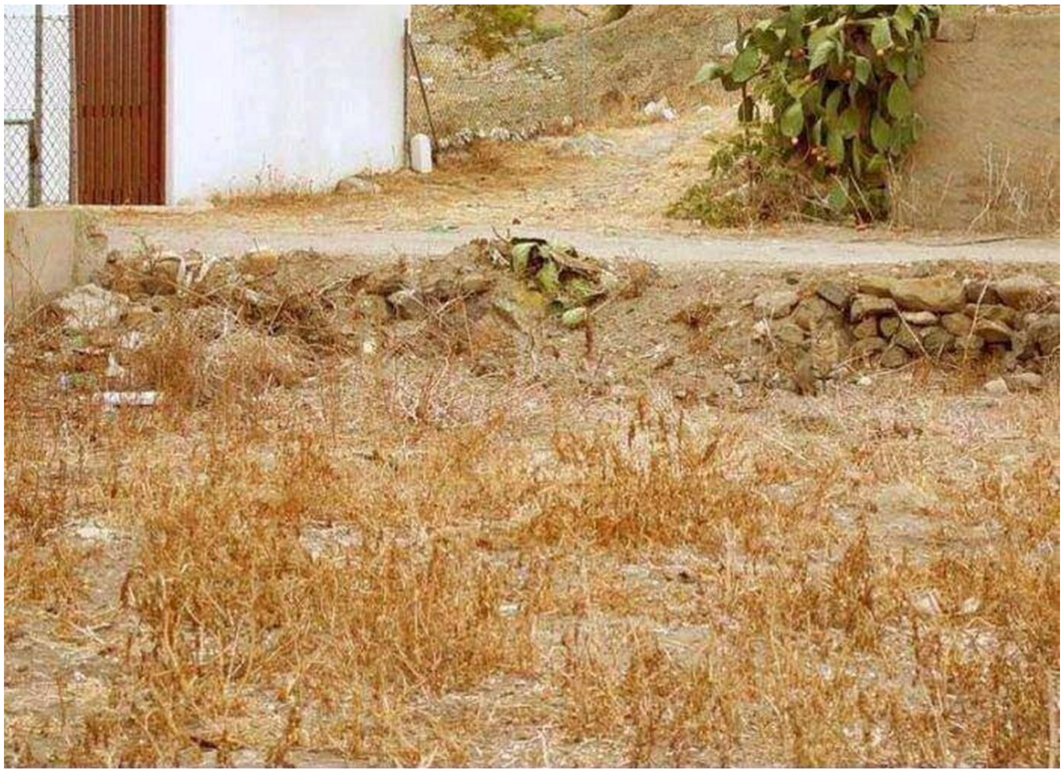


Figure 4.1: Visual priming: something is hidden in plain sight in this image. One is unlikely to notice it without a cue for what it is (for an observer that has not seen this image before). Once a cue is given, perception is modified to allow successful detection. See the supplementary material in Sec. A.2 for the full answer.

in an “Aha!” moment for the viewer, as he/she suddenly perceives the image differently; Fig. 4.1 shows an example of such a case. We make here the distinction between 3 detection strategies: (1) *free viewing*, (2) *priming* and (3) *pruning*. Freely viewing the image, the default strategy, likely reveals nothing more than a dry grassy field near a house. Introducing a cue about a target in the image results in one of two possibilities. The first, also known as priming, is modification to the computation performed when viewing the scene with the cue in mind. The second, which we call pruning - is a modification to the decision process after all the computation is finished. When the task is to detect objects, this can mean retaining all detections that match the cue, even very low confidence ones and discarding all others. While both are viable ways to incorporate the knowledge brought on by the cue, priming often highly increases the chance of detecting the cued object. Viewing the image for an unlimited amount of time and pruning the results is less effective; in some cases, detection is facilitated only by the cue. We claim that priming allows the cue to affect the visual process from early layers, allowing detection where it was previously unlikely to occur in free-viewing conditions. This has also recently gained some neurophysiological evidence [119].

We propose a mechanism to mimic the process of visual priming in deep neural networks in the context of object detection and segmentation. The mechanism transforms an external cue about the presence of a certain class in an image (e.g., “person”) to a modulatory signal that affects all layers of the network. This modulatory effect is shown via experimentation to significantly improve object detection performance when the cue is present, more so than a baseline which simply applies post-processing to the network's result. Furthermore, we show that priming early visual layers has a greater effect than doing so for deeper layers. Moreover, the effects of priming are shown to be much more pronounced in difficult images such as very noisy ones.

We investigate the modulatory role of the priming mechanism to support the argument that a TD mechanism properly gates the flow of hidden activities in the BU pass. As a result, the gating mechanism incorporates robustness against noise inference and remain more resilience against such visual disturbance. The priming mechanism motivates the effectiveness of a TD gating mechanism for object segmentation and supports the hypothesis that interference robustness is achievable.

4.3 Related Work

Context has been very broadly studied in cognitive neuroscience [120, 121, 122, 123, 124, 125, 126] and in computer vision [127, 128, 129, 130, 131, 132, 133]. It is widely agreed [134] that context plays crucial role for various visual tasks. Attempts have been made to express a tangible definition for context due to the increased use in the computer vision community [129, 130].

Biederman et al. [120] hypothesizes object-environments dependencies into five categories: probability, interposition, support, familiar size, and position. Combinations of some of these categories would form a source of contextual information for tasks such as object detection [130, 134], semantic segmentation [135], and pose estimation [136]. Context consequently is the set of sources that partially or collectively influence the perception of a scene or the objects within [137].

Visual cues originated from contextual sources, depending on the scope they influence, further direct visual tasks at either global or local level [129, 130]. Global context such as scene configuration, imaging conditions, and temporal continuity refers to cues abstracted across the whole scene. On the other hand, local context such as semantic relationships and local-surroundings characterize associations among various parts of similar scenes.

Having delineated various contextual sources, the general process by which the visual hierarchy is modulated prior to a particular task is referred to as visual priming [7, 138]. A cue could be provided either implicitly by a contextual source or explicitly through other modalities such as language.

There has been a tremendous amount of work on using some form of top-down feedback to contextually prime the underlying visual representation for various tasks [123, 124, 125, 126]. The objective is to have signals generated from some task such that they could prepare the visual hierarchy oriented for the primary task. [134] proposes contextual priming and feedback for object detection using the Faster R-CNN framework [72]. The intuition is to modify the detection framework to be able to generate semantic segmentation predictions in one stage. In the second stage, the segmentation primes both the object proposal and classification modules.

Instead of relying on the same modality for the source of priming, [139, 140] proposes to modulate features of a visual hierarchy using the embedding of the language model trained on the task of visual question answering [141, 142]. In other words, using feature-wise affine transformations, [140] multiplicatively and additively modulates hidden activities of the visual hierarchy using the top-down priming signals generated from the language model, while [134] appends directly the semantic segmentation predictions to the visual hierarchy. Recently, [135] proposes to modulate convolutional weight parameters of a neural network using segmentation-aware masks. In this regime, the weight parameters of the model are directly approached for the purpose of priming.

Although all these methods modulate the visual representation, none has specifically studied the explicit role of category cues to prime the visual hierarchy for object detection and segmentation. In this work, we strive to introduce a consistent parametric mechanism into the neural network framework. The proposed method allows every portion of the visual hierarchy to be primed for tasks such as object detection and semantic segmentation. It should be noted that this use of priming was defined as part of the Selective Tuning (ST) model of visual attention [21]. Other aspects of ST have recently appeared as part of classification and localization networks as well [108, 2], and our work explores yet another dimension of the ST theory.

4.4 Approach

Assume that we have some network N to perform a task such as object detection or segmentation on an image I. In addition, we are given some cue $h \in \mathcal{R}^{\mathrm{n}}$ about the content of the image, where n varies depending on the cue based on which the priming is performed. We next describe pruning and

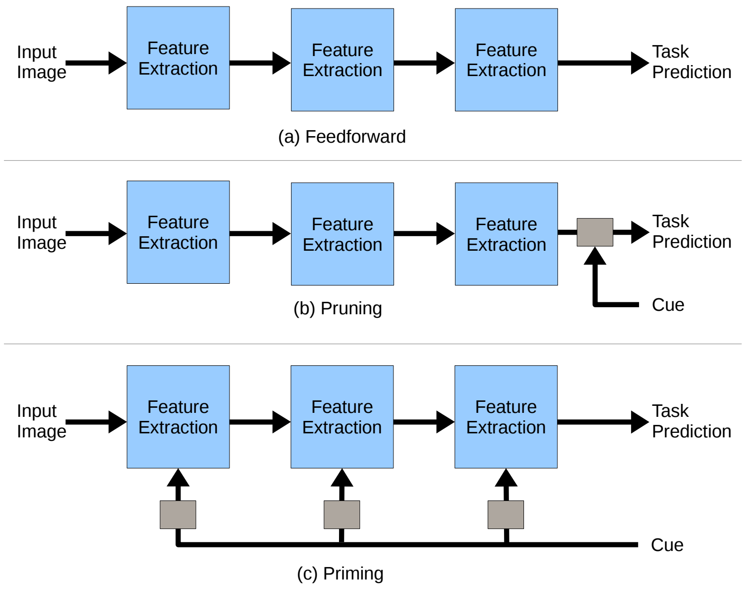


Figure 4.2: A neural network can be applied to an input in either an unmodified manner (*top*), pruning the results after running (*middle*) or *priming* the network via an external signal (cue) in image to affect all layers of processing (*bottom*).

priming, how they are applied and how priming is learned. We assume that h is a binary encoding of them presence of some target(s) (e.g, objects) - though this can be generalized to other types of information. For instance, an explicit specification of color, location, orientation, etc, or an encoded features representation as can be produced by a vision or language model. Essentially, one can either ignore this cue, use it to post-process the results, or use it to affect the computation. These three strategies are presented graphically in Fig. 4.2.

Pruning. In pruning, N is fed an image and we use h to post-process the result. In object detection, all bounding boxes output by N whose class is different than indicated by h are discarded. For segmentation, assume N outputs a score map of size $C \times H \times W$, where C is the number of classes learned by the network, including a background class, H and W are the height and width of the output score map. We propose two methods of pruning, with complementary effects. The first type increases recall by ranking the target class higher: for each pixel (x,y), we set the value of all score maps inconsistent with h to be $-\infty$, except that of the background. This allows whatever detection of the hinted class to be ranked higher than other which previously masked it. The second type simply sets each pixels which was not assigned by the segmentation the target class to the background class. This decreases recall but increases the precision. These types of pruning are demonstrated in Fig. 4.8 and discussed below.

Priming Our approach is applicable to any network N with a convolutional structure, such as a modern network for object detection, e.g. [5]. To enable priming, we freeze all weights in N and add a parallel branch N_p. The role of N_p is to transform an external cue $h \in \mathcal{R}^{\mathrm{n}}$ to modulatory signals which affect all or some of the layers of N. N_p has P parametric layers which are set according to the priming specification and the number of layer in N. Given the external cue h, N_p gate the appropriate coefficient weights to modulate in information propagation in N. Namely, let L_i be some layer of N. Denote the output of L_i by $x_i \in \mathcal{R}^{C_i \times H_i \times W_i}$ where C_i is the number of feature planes and H_i, W_i are the height and width of the feature planes. Denote the j_{th} feature plane of x_i by $x_{ij} \in \mathcal{R}^{H_i \times W_i}$. N_p modulates each feature plane x_{ij} by applying the function

$$f_{ij}(x_{ij}, h) = \hat{x}_{ij}. \tag{4.1}$$

The function f_{ij} always operates in a spatially-invariant manner - for each element in a feature plane, the same function is applied. In other words, The function f_{ij} multiplicatively modulates the feature units on each feature plane j at layer i regardless of the spatial correlations between features. The intuition is that convolutional kernel filters are taking care of feature encoding in the spatial domain and the priming mechanism is only responsible to bias for critical feature planes. This implies the complementary role of the priming mechanism to the feature encoding in neural networks. Obviously, the spatially-invariant formulation cannot prime spatially feature planes. Currently, the assumption is that feature transformation in neural networks is capable of separating appearance features of objects into separate planes so then once a plane is primed, we implicitly target a particular spatial regions. Other than that, currently the limitation of the priming mechanism is that there is no explicit spatial priming mechanism. We define the function f_{ij} specifically using a simple residual function, that is

$$\hat{x}_{ij} = \alpha_{ij} \cdot x_{ij} + x_{ij}, \tag{4.2}$$

where the coefficients $\alpha_i = [\alpha_{i1}, \ldots, \alpha_{ic_i}]^T$ are determined by a linear transformation of the cue:

$$\alpha_i = W_i * h, \tag{4.3}$$

where $W_i \in \mathcal{R}^{C_i \times C}$ such that C_i is the number of feature planes at layer i and C is the number of target classes that N learns to predict. the coefficient

α_{ik} specifies how much the feature units on feature plane k are multiplicatively modulated. N_p has the set of learnable parameters $\{W_1, \ldots, W_P\}$ where W_i is the tensor of coefficient parameters that are learned during the network training using the stochastic optimization algorithm. The parameters basically specify that given the top-down signaling cue h, how the feature planes need to be adjusted to better accommodate for the task requirements. An overall view of the proposed method is presented in Fig. 4.3.

Types of Modulation The modulation in eq. 4.2 simply adds a calculated value to the feature plane. We have experimented with other types of modulation, namely non-residual ones (e.g, purely multiplicative), as well as following the modulated features with a non-linearity (ReLU), or adding a bias term in addition to the multiplicative part. The single most important dominant ingredient to reach good performance was the residual formulation - without it, training converged to very poor results. The formulation in eq. 4.2 performed best without any of the above listed modifications. We note that an additive model, while having converged to better results, is not fully consistent with biologically plausible models ([21]) which involve suppression/selection of visual features, however, it may be considered a first approximation.

Types of Cues The simplest form of a cue h is an indicator vector of the object(s) to be detected, i.e, a vector of 20 zeros and 1 in the coordinate corresponding to "horse", assuming there are 20 possible object classes, such as in Pascal [143]. We call this a *categorical* cue because it explicitly carries semantic information about the object. This means that when a single class k is indicated, α_i becomes the k_{th} column of W_i.

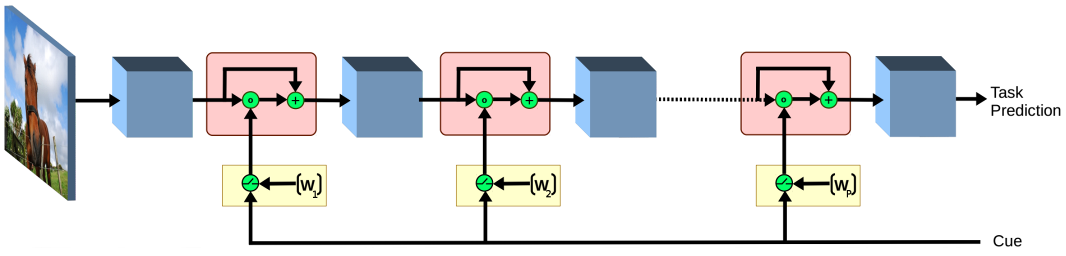


Figure 4.3: Overall view of the proposed method to prime deep neural networks. A cue about some target in the image is given by an external source or some form of feedback. The process of priming involves affecting each layer of computation of the network by modulating representations along the path. At the top, the stack of layers in N are schematically illustrated by the blue blocks. At the bottom, the coefficient parameters W_i in N_p are illustrated by the yellow blocks.

4.4.1 Training

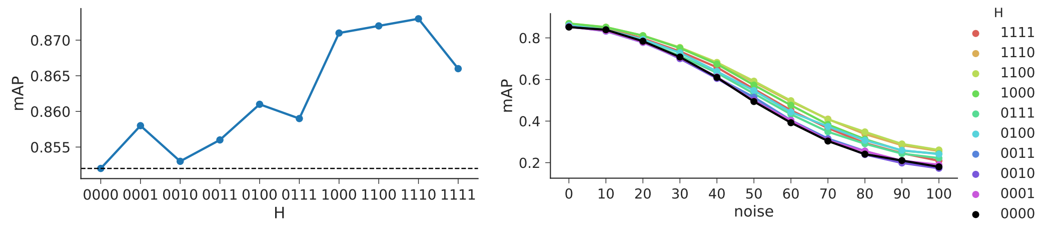
To learn how to utilize the cue, we freeze the parameters of our original network N and add the network block N_p. During training, with each training example (I_i, y_i) fed into N we feed h_i into N_p, where I_i is an image, y_i is the ground-truth set of bounding boxes and h_i is the corresponding cue. The output and loss functions of the detection network remain the same, and the error is propagated through the parameters of N_p. Fig. 4.3 illustrates the network. N_p is very lightweight with respect to N, as it only contains parameters to transform from the size of the cue h to at most $K = \sum_i k_i$ where k_i is the number of output feature planes in each layer of the network.

Multiple Cues Per Image. Contemporary object detection and segmentation benchmarks [25, 143] often contain more than one object type per image. In this case, we may set each coordinate in h to 1 iff the corresponding class is present in the image. However, this tends to prevent N_p from learning to modulate the representation of N in a way which allows it to suppress irrelevant objects. Instead, if an image contains k distinct object classes, we duplicate the training sample k times and for each duplicate set the ground truth to contain only one of the classes. This comes at the expense of a longer training time, depending on the average number k over the dataset.

4.5 Experimental Results

We evaluate our method on two tasks: object detection and object class segmentation. In each case, we take a pre-trained deep neural network and explore how it is affected by priming or pruning. Our goal here is not necessarily to improve state-of-the-art results but rather to show how use of top-down cues can enhance performance. Our setting is therefore different than standard object-detection/segmentation scenarios: we assume that some cue about the objects in the scene is given to the network and the goal is to find how it can be utilized optimally. Such information can be either deduced from the scene, such as in contextual priming [134, 144] or given by an external source, or even be inferred from the task, such as in question answering [141, 142].

Our experiments are conducted on the Pascal VOC [143] 2007 and 2012 datasets. For priming object detection networks we use pre-trained models of SSD [5] and yolo-v2 [145] and for segmentation we use the FCN-8 segmentation network of [146] and the DeepLab network of [4]. We use the YellowFin optimizer [147] in all of our experiments, with a learning rate of either 0.1 or 0.01 (depending on the task). Additional qualitative experimental results are



(a) (b)

Figure 4.4: (a) Performance gains by priming different parts of the SSD objects detector. Priming early parts of the network causes the most significant boost in performance. Black dashed line shows performance by pruning. (b) Testing variants of priming against increasing image noise. The benefits of priming become more apparent in difficult viewing conditions. The x axis indicates which block of the network was primed (1 for primed, 0 for not primed).

also provided in Sec. A.2.

4.5.1 Object Detection

We begin by testing our method on object detection. Using an implementation of SSD [5], we apply a pre-trained detector trained on the trainval sets of Pascal 2012+2007 to the test set of Pascal 2007. We use the SSD-300 variant as described in the paper. In this experiment, we trained and tested on what we cal PAS$^{\#}$: this is a reduced version of Pascal-2007 containing only images with a single object class (but possibly multiple instances). We use this reduced dataset to test various aspects of our method, as detailed in the following subsections. Without modification, the detector attains a mAP (mean-average precision) of 81.4% on PAS$^{\#}$(77.4% on the full test set of Pascal 2007). Using simple pruning as described above, this increases to 85.2%. This large boost in performance is perhaps not surprising, since pruning effectively removes all detections of classes that do not appear in the image. The remaining errors are those of false alarms of the “correct” class or mis-detections.

4.5.1.1 Deep vs Shallow Priming

We proceed to the main result, that is, how priming affects detection. The SSD object detector contains four major components: (1) a pre-trained part made up of some of the layers of vgg-16 [148] (a.k.a the “base network” in the SSD paper), (2) some extra convolutional layers on top of the vgg-part, (3) a localization part and (4) a class confidence part. We name these part *vgg*, *extra*, *loc* and *conf* respectively.

To check where priming has the most significant impact, we select different subsets of these components and denote them by 4-bit binary vectors $s_i \in \{0,1\}^4$, where the bits correspond from left to right to the vgg, extra, localization, and confidence parts. For example, $s = 1000$ means letting N_p affect only the earliest (*vgg*) part of the detector, while all other parts remain unchanged by the priming (except indirectly affecting the deeper parts of the net). We train N_p on 10 different configurations: these include priming from the deepest layers to the earliest: 1111, 0111, 0011, 0001 and from the earliest layer to the deepest: 1000, 1100, 1110. We add 0100 and 0010 to check the effect of exclusive control over middle layers and finally 0000 as the default configuration in which N_p is degenerate and the result is identical to pruning. Fig 4.4 (a) shows the effect of priming each of these subsets of layers on PAS$^{\#}$. Priming early layers (those at the bottom of the network) has a much more pronounced effect than priming deep layers. The single largest gain by priming a single component is for the *vgg* part: 1000 boosts performance from 85% to

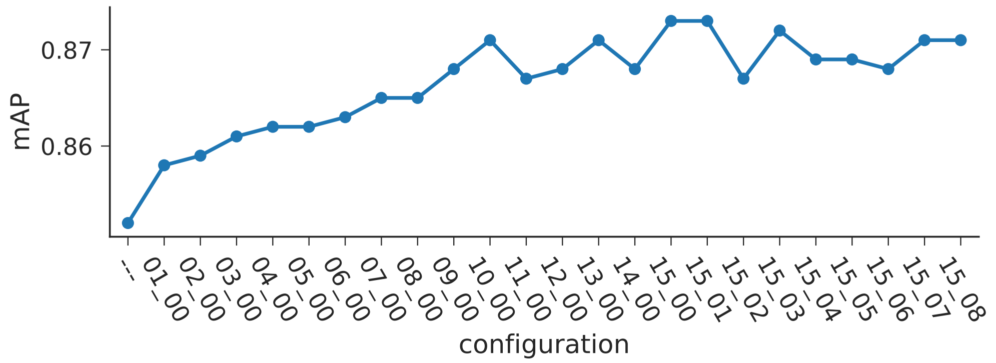
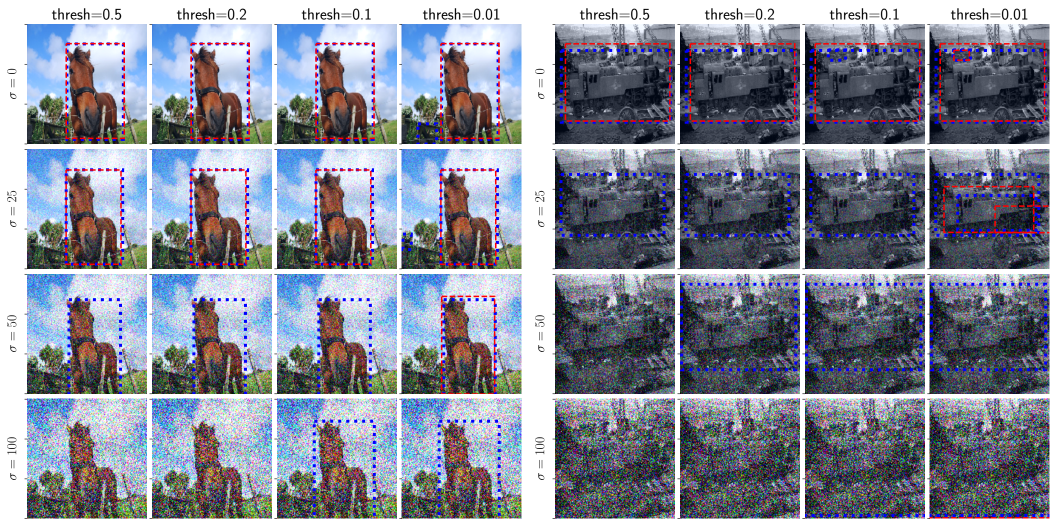


Figure 4.5: Effects of early priming: we show how mAP increases when we allow priming to affect each layer in turn, from the very bottom of the network. Priming early layers has a more significant effect than doing so for deeper ones. The numbers indicate how many layers were primed from the first and second blocks of the SSD network, respectively.

87.1%. A smaller gain is attained by the *extra* component: 86.1% for 0100. The performance peaks at **87.3**% for 1110, though this is only marginally higher than attained by 1100 - priming only the first two parts.

4.5.1.2 Ablation Study

Priming the earliest layers (*vgg*+*extra*) of the SSD object detector brings the most significant boost in performance. The first component described above contains 15 convolutional layers and the second contains 8 layers, an overall total of 23. To see how much we can gain with priming on the first few layers, we checked the performance on PAS$^{\#}$ when training on the first k layers only, for each $k \in \{1, 2, \ldots 23\}$. Each configuration was trained for 4000 iterations. Fig. 4.5 shows the performance obtained by each of these configurations, where i_j in the x-axis refers to having trained the first i layers and the first j layers of



(a) (b)

Figure 4.6: Priming vs. Pruning. Priming a detector allows it to find objects in images with high levels of noise while mostly avoiding false-alarms. Left to right (a,b): decreasing detection thresholds (increasing sensitivity). Top to bottom: increasing levels of noise. Priming (blue dashed boxes) is able to detect the horse (a) across all levels of noise, while pruning (red dashed boxes) does not. For the highest noise level, the original classifier does not detect the horse at all - so pruning is ineffective. (b) Priming enables detection of the train for all but the most severe level of noise. Decreasing the threshold for pruning only produces false alarms. We recommend viewing this figure in color on-line.

the first and second parts respectively. We see that the very first convolutional layer already boosts performance when primed. The improvement continues steadily as we add more layers and fluctuates around 87% after the 15th layer. The fluctuation is likely due to randomness in the training process. This further shows that priming has strong effects when applied to very early layers of the network.

4.5.1.3 Detection in Challenging Images

As implied by the introduction, perhaps one of the cases where the effect of priming is stronger is when facing a challenging image, such as adverse imaging conditions, low lighting, camouflage, noise. As one way to test this, we compared how priming performs under noise. We took each image in the test set of Pascal 2007 and added random Gaussian noise chosen from a range of standard deviations, from 0 to 100 in increments of 10. The noisy test set of $\text{PAS}^{\#}$ with variance σ is denoted $\text{PAS}^{\#}{}_{\text{N}(\sigma)}$. For each σ, we measure the mAP score attained by either pruning or priming. Note that none of our experiments involved training with corrupted images - these are only used for testing. We plot the results in Fig. 4.4 (b). As expected, both methods suffer from decreasing accuracy as the noise increases. However, priming is more robust to increasing levels of noise; the difference between the two methods peaks at a moderate level of noise, that is, $\sigma = 80$, with an advantage of **10.7**% in mAP: 34.8% compared to 24.1% by pruning. The gap decreases gradually to 6.1% (26.1% vs 20%) for a noise level of $\sigma = 100$. We believe that this is due to the early-layer effects of priming on the network, selecting features from the bottom up to match the cue. Fig 4.6 shows qualitative examples, comparing priming versus pruning: we increase the noise from top to bottom and decrease the threshold (increase the sensitivity) from left to right. We show in each image only the top few detections of each method to avoid clutter. Priming allows the detector to find objects in images with high levels of noise (see lower rows of a,b). In some cases priming proves to be *essential* for the detection: lowering the un-primed detector's threshold to a minimal level does not increase the recall of the desired object (a, 4th row); in fact, it only increases the number of false alarms (b, 2nd row, last column). Priming, on the other hand, is often less sensitive to a low threshold and the resulting detection persists along a range thereof.

4.5.2 Cue Aware Training

In this section, we also test priming on an object detection task as well as segmentation with an added ingredient - multi-cue training and testing. In Sec. 4.5.1 we limited ourselves to the case where there is only one object class per image. This limitation is often unrealistic. To allow multiple priming cues per image, we modify the training process as follows: for each training sample $< I, gt >$ containing object classes $c_1, \dots c_k$ we split the training example for I to k different tuples $< I_i, h_i, gt_i >, i \in \{1 \dots k\}$, where I_i are all identical to I, h_i indicate the presence of class c_i and gt_i is the ground-truth gt reduced to contain only the objects of class c_i - meaning the bounding boxes for detection, or the masks for segmentation. This explicitly coerces the priming network N_p to learn how to force the output to correspond to the given cue, as the input image remains the same but the cue and desired output change together. We refer to this method multi-cue aware training (CAT for short), and refer to the unchanged training scheme as regular training.

4.5.2.1 Multi-Cue Segmentation

Here, we test the multi-cue training method on object class segmentation. We begin with the FCN-8 segmentation network of [146]. We train on the training split of SBD (Berkeley Semantic Boundaries Dataset and Benchmark) dataset [149], as is done in [150, 4, 151, 146]. We base our code on an unofficial PyTorch[1] implementation[2]. Testing is done of the validation set of Pascal 2011, taking care to avoid overlapping images between the training set defined by [149] [3], which leaves us with 736 validation images. The baseline results average IOU score of 65.3%. As before, we let the cue be a binary encoding of the classes present in the image. We train and test the network in two different modes: one is by setting for each training sample (and testing) the cue so $h_i = 1$ if the current image contains at least one instance of class i and 0 otherwise. The other is the multi-cue method we describe earlier, i.e , splitting each sample to several cues with corresponding ground-truths so each cue is a one-hot encoding, indicating only a single class. For both training strategies, testing the network with a cue creates a similar improvement in performance, from 65.3% to 69% for regular training and to 69.2% for multi-cue training.

The main advantage of the multi-cue training is that it allows the priming network N_p to force N to focus on different objects in the image. This is illustrated in Fig. 4.7. The top row of the figure shows from left to right an input image and the resulting segmentation masks when the network is cued with classes *bottle*, *diningtable* and *person.* The bottom row is cued with *bus*, *car*, *person.* The cue-aware training allows the priming network to learn how to suppress signals relating to irrelevant classes while retaining the correct class from the bottom-up.

[1]`http://pytorch.org/`

[2]`https://github.com/wkentaro/pytorch-fcn`

[3]for details, please refer to `https://github.com/shelhamer/fcn.berkeleyvision.org/tree/master/data/pascal`

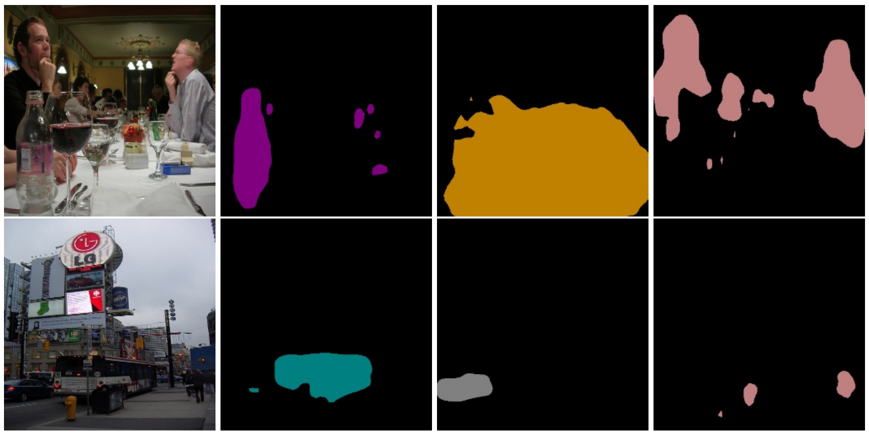


Figure 4.7: Effect of priming a segmentation network with different cues. In each row, we see an input image and the output of the network when given different cues. Top row: cues are respectively bottle, dining table, person. Bottom row: cues are respectively bus, car, person. Given a cue (e.g, *bottle*), the network becomes more sensitive to bottle-like image structures while suppressing others. This happens not by discarding results but rather by affecting computation starting from the early layers.

Types of Pruning. As mentioned in Sec. 4.4, we examine two types of pruning to post-process segmentation results. One type removes image regions which were wrongly labeled as the target class, replacing them with background and the other increases the recall of previously missed segmentation regions by removing all classes except the target class and retaining pixels where the target class scored higher than the background. The first type increases precision but cannot increase recall. The second type increases recall but possibly hinders precision. We found that both types results in a similar overall mean-IOU. Figure 4.8 shows some examples where both types of pruning result in segmentations inferior to the one resulting by priming: post-processing can increase recall by lowering precision (first row, column d) or increase precision by avoiding false-detections (second and fourth row, col-

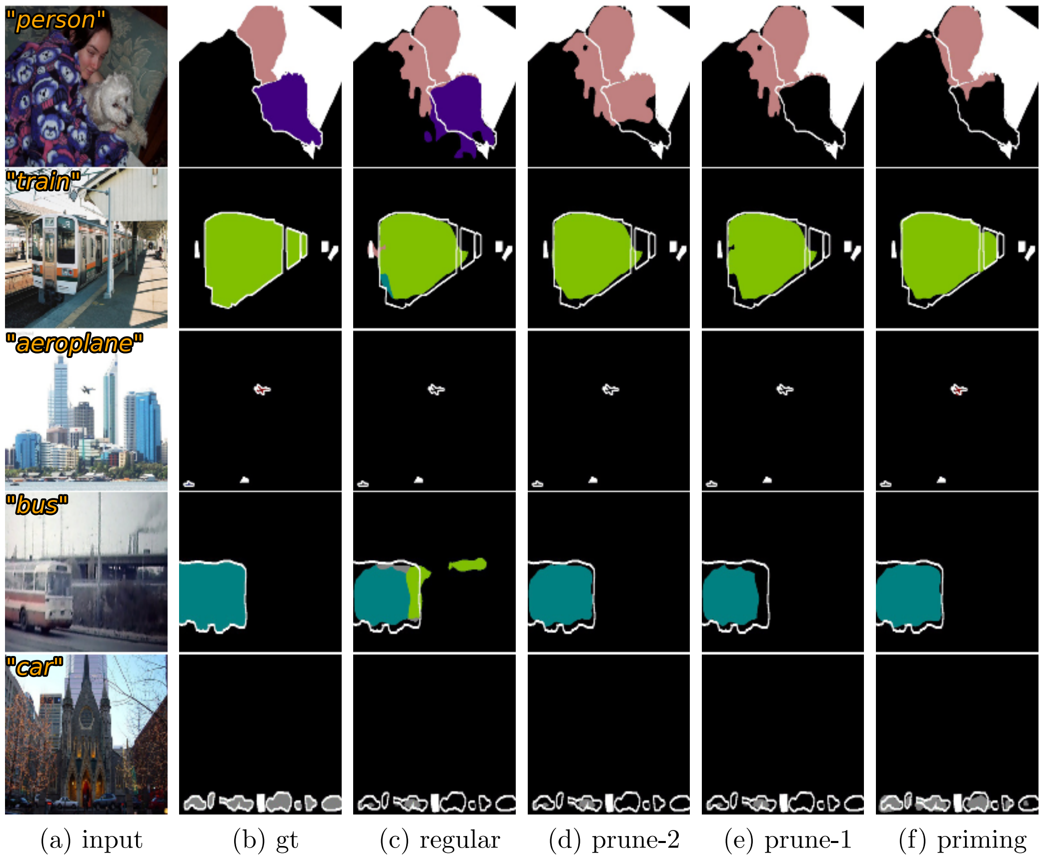


Figure 4.8: Comparing different methods of using a cue to improve segmentation: From left to right: input image (with cue overlayed), ground-truth (all classes), unprimed segmentation, pruning type-2, pruning type-1, and priming. In each image, we aid the segmentation network by adding a cue (e.g, “plane”). White regions are marked as “don’t care” in the ground truth.

umn e), priming (column f) increases both recall and precision. The second, and fourth rows missing parts of the train/bus are recovered while removing false classes. The third and fifth rows previously undetected small objects are now detected. The person (first row) is segmented more accurately.

DeepLab. Next, we use the DeepLab [4] network for semantic-segmentation with ResNet-101 [50] as a base network. We do not employ a CRF as post-processing. The mean-IOU of the baseline is 76.3%. Using Priming, increases this to 77.15%. While in this case priming does not improve as much as in the other cases we tested, we find that it is especially effective at enabling the network to discover small objects which were not previously segmented by the non-primed version: the primed network discovers 57 objects which were not discovered by the unprimed network, whereas the latter discovers only 3 which were not discovered by the former. Fig. 4.9 shows some representative examples of where priming was advantageous. Note how the bus, person, (first three rows) are segmented by the primed network (last column). We hypothesize that the priming process helps increase the sensitivity of the network to features relevant to the target object. The last row shows a successful segmentation of potted plants with a rather atypical appearance.

4.5.2.2 Multi-Cue Object Detection

We apply the CAT method to train priming on object detection as well. For this experiment, we use the YOLOv2 method of [145]. The base network we used is a port of the original network, known as YOLOv2 544x544. Trained

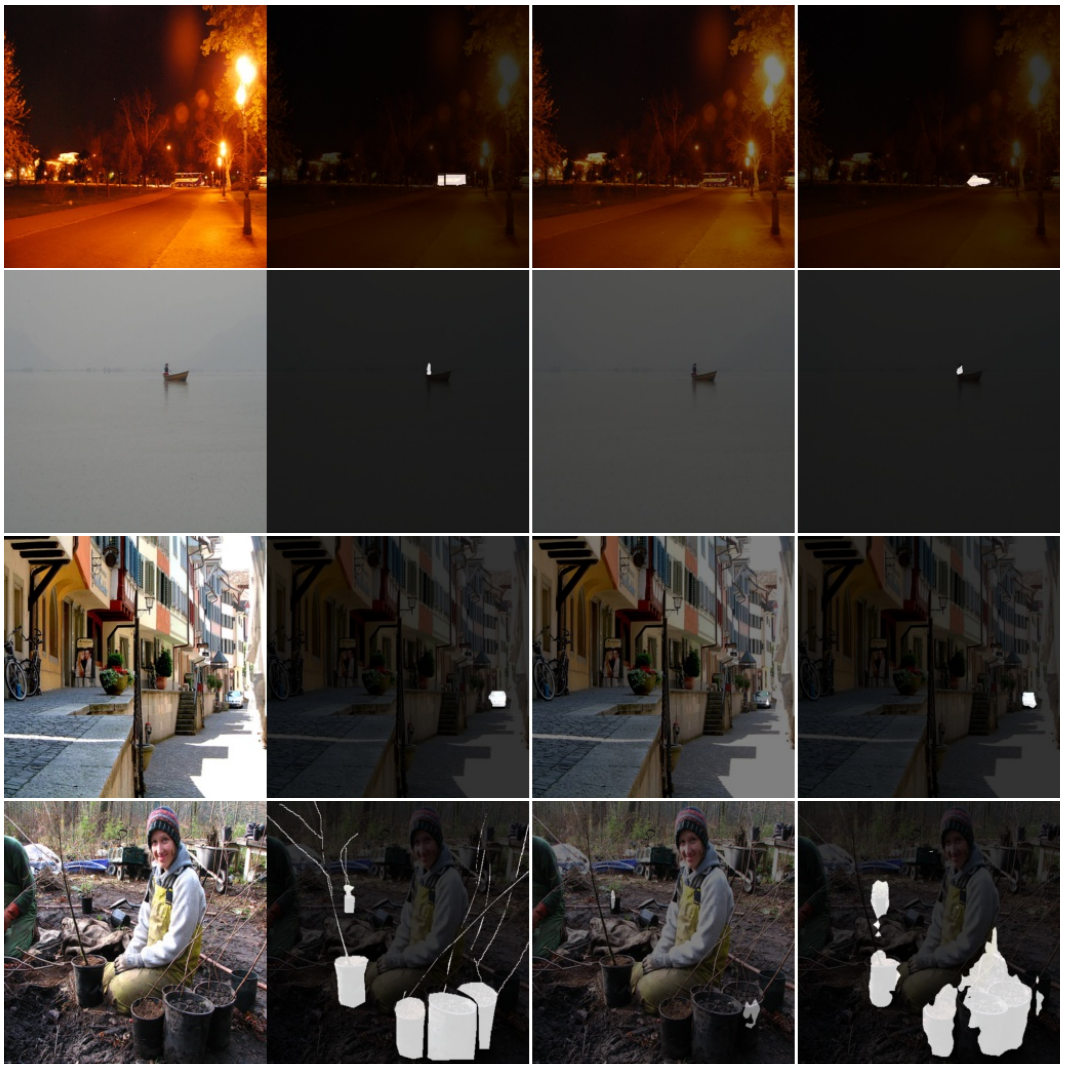


Figure 4.9: Priming a network allows discovery of small objects which are completely missed by the baseline method or ones with uncommon appearance (last row). From left to right: input image, ground-truth, baseline segmentation [4], primed network.

on the union of Pascal 2007 and 2012 datasets, it is reported by the authors to obtain 78.6% mAP on the test set of Pascal 2007. The implementation we use[4] reaches a slightly lower 76.8%, with a PyTorch port of the network weights released by the authors. We use all the convolutional layers of DarkNet (the base network of YOLOv2) to perform priming. We freeze all network parameters of the original detection network and train a priming network with the multi-cue training method for 25 epochs. When using only pruning, performance on the test-set improves to 78.2% mAP. When we include priming as well, this goes up to 80.6%,

4.6 Conclusion

We have presented a simple mechanism to prime neural networks, as inspired by psychological top-down effects known to exist in human observers. We have tested the proposed method on two tasks, namely object detection and segmentation, using two methods for each task, and comparing it to simple post-processing of the output. Our experiments confirm that as is observed in humans, effective usage of a top-down signal to modulate computations from early layers not only improves robustness to noise but also facilitates better object detection and segmentation, enabling detection of objects which are missed by the baselines without compromising precision, notably so for small objects and those having an atypical appearance.

[4] `https://github.com/marvis/pytorch-yolo2`

Chapter 5

Object Segmentation Using Selective Attention

5.1 Abstract

Convolutional neural networks model the transformation of the input sensory data at the bottom of a network hierarchy to the semantic information at the top of the visual hierarchy. Feedforward processing is sufficient for some object recognition tasks. Top-Down selection is potentially required in addition to the Bottom-Up feedforward pass. It can, in part, address the shortcoming of the loss of location information imposed by the hierarchical feature pyramids. We propose a unified 2-pass framework for object segmentation that augments Bottom-Up convolutional neural networks with a Top-Down selection network. We utilize the top-down selection gating activities to modulate the bottom-up hidden activities for segmentation predictions. We develop an

end-to-end multi-task framework with loss terms satisfying task requirements at the two ends of the network. We evaluate the proposed network on benchmark datasets for semantic segmentation, and show that networks with the Top-Down selection capability outperform the baseline model. Additionally, we shed light on the superior aspects of the new segmentation paradigm and qualitatively and quantitatively support the efficiency of the novel framework over the baseline model that relies purely on parametric skip connections.

5.2 Introduction

In both human and machine vision systems, two directions of information flow have been commonly considered, a data-driven or feedforward direction (Bottom-Up), and a reverse direction (Top-Down) that has a predictive, controlling or modulatory role. In the Bottom-Up (BU) pathway, the sensory input data is processed and sequentially transformed into high-level semantic information such that some task criterion is satisfied at the inference phase, while during the training phase, the error gradient signals are calculated according to a loss function and gradients are propagated down to the early layers for the updating of the network's weight parameters. Convolutional neural networks model the BU pathway for visual tasks such as object classification, detection, and segmentation.

The TD pathway, on the other hand, inherently characterizes modulatory and controlling roles and leverages selection mechanisms. TD selection approaches have been used for tasks such as object localization, object segmentation, and network visualization. Selective Tuning [21, 7] is a computational model of visual attention, and is an early attempt to establish the applicability of a TD selection pass along with a BU feedforward pass for basic visual tasks in dynamical networks. In chapter 3 we have shown that Selective Tuning of convolutional neural networks (STNet) succeeds to formulate a neural network framework with a selective TD mechanism and is evaluated for the object localization task [108]. STNet validates the effective role of the TD selection pass to localize relevant regions covering the object features. In this chapter, we investigate the role of TD selective attention in convolutional neural networks and the feature modulation of BU hidden activities for the task of object segmentation. We attempt to complement the STNet-for-localization formulation in this chapter with feature modulation of BU hidden activities. This chapter strives to examine whether feature modulation derived from a TD selective pass is successful for the demanding object segmentation task.

The purpose behind TD feature modulation is to select and modify the data interpretations represented by the hidden BU activities. Therefore, for a subsequent processing stage such as object segmentation, the TD selection patterns modulate the BU feature encoding activities. As a result, the subsequent segmentation process will benefit from both of the densely-encoded bottom-up flow of information and the sparsely-selective top-down patterns of modulation.

Dominant approaches for object segmentation are generally densely parametric in a fully-convolutional manner. Multiple up-sampling convolutional layers are leveraged to predict up-scaled category maps at the final segmentation layer. Approaches such as skip connections, multi-level feature augmentation, and discriminative attention are proposed to produce fine-grained segmentation. All such approaches enforce up-sampling of predicted score maps of a pre-trained network through a number of parametric layers in a purely feedforward manner. While they have reached a promising performance level, we show the up-sampling part does not need to be densely parametric. Rather, a successful hierarchical TD selection through the BU network can be helpful to modulate rich feature maps for segmentation map predictions.

We propose the Selective Segmentation Network (SSN) to systematically study the use of the TD selection pass for the object segmentation task. SSN consists of the BU pass that utilizes a typical convolutional neural network with multiple layers of feature extraction. To deal with the multi-instance and multi-scale issues in the experimental evaluation, we define a controlling module called Loose Spatial Detection (LSD) that examines high-level semantic information and loosely predicts the locations and scales at which TD attention needs to be activated. Once important locations and scales are determined, attention signals are set and the TD selection pass is activated. The TD selection mechanism has three stages of processing at each layer that relies on two stages of local competitions and one stage of normalization for gating activity propagation. Gating activities at each layer are computed and the selection is passed to the lower layer until some early layer in the visual hierarchy is reached.

The TD pass systematically computes selection patterns over relevant hidden features of the BU network. We develop our investigation around the hypothesis that the TD selection patterns are reliable as a source of hidden feature modulation for segmentation. We propose to have the information flow of hidden feature activities modulated by the TD gating activities into the segmentation pipeline for the final output predictions. We study the effect of three types of modulation at different stages of computation on the prediction performance of SSN. The segmentation pipeline forms a cascade of parametric blocks each consisting of a number of processing units. Each block performs operations such as input feature modulation, channel reduction, and parametric feature fusion. At each layer the modulated input information from the BU and TD passes is integrated into the segmentation pipeline and then using parametric transformation is passed to the layer below. After a few layers, the final segmentation maps are predicted at the bottom of the visual hierarchy.

SSN benefits from a multi-task formulation. We define two loss functions: one at the top of the visual hierarchy where the LSD module outputs the label predictions for attention signal initialization and one at the bottom of the visual hierarchy where the segmentation output maps are generated. The former loss measures the capability of the BU network and LSD module to jointly predict the starting points for TD pass initialization. The latter loss measures the performance of SSN to predict segmentation maps.

Learning in SSN has two phases: in the first phase, the pre-trained parameters of the BU network and the uniformly initialized parameters of the LSD modules are jointly fine-tuned using the first loss function before being loaded into the complete SSN framework. In the second phase, SSN is loaded with the parameters obtained in the first phase, and the entire segmentation network is trained with the two loss functions using the Stochastic Gradient Descent (SGD) algorithm.

SSN is qualitatively and quantitatively evaluated on the visual task of semantic segmentation. Semantic segmentation [152, 153, 25, 154] is the task of predicting pixel-level category labels given a pre-defined list of object classes. Unlike the object localization task that outputs a number of bounding boxes enclosing category instances, semantic segmentation returns a segmentation map containing a category label at each pixel.

We illustrate that SSN improves the baseline model for the evaluation performance on three benchmark datasets: Pascal VOC, CamVid, and Horse-Cow datasets. We conduct ablation studies to shed light on aspects of feature modulation using TD selection such as feature entanglement and noise interference. Experimental results reveal that the modulatory nature of the TD pass helps to untangle the underlying feature representations to some degree and improves the results of the segmentation metrics under input perturbation scenarios such as additive noise and box occlusion.

5.3 Related Work

There are two main classes of previous research that are related to our own, and a brief overview will be provided under the headings of Semantic Segmentation and Top-Down Approaches.

Semantic Segmentation: The Fully-Convolutional Network (FCN) [146] has been a pioneer to introduce convolutional encoder-decoder networks that benefit from parametric skip connections and fractionally-strided convolutions to gradually up-sample in a parametric fashion the output of the label prediction layer at the top of a regular convolutional neural network. The architecture implements an information bottleneck using the encoding network which is basically an extension of a multi-layer classifier and the decoding network that up-samples the semantic label predictions of the encoding network into the output segmentation map. The FCN approach has been extended with novel approaches for performance improvements [155, 156, 157, 158, 152, 3]. These approaches are mainly involved with modifications and extensions such as novel network architectures (*e.g.* residual networks [50]), addition of extra parametric layers and dense connectivity, multi-scale augmentation, and multi-level supervision to improve the segmentation prediction accuracy specially for small and fine-detailed objects. These approaches have shown success in terms of the evaluation performance metrics. We attempt to study the complementary role of Top-Down selection to such encoder-decoder frameworks. Furthermore, unlike the FCN model, the skip connections are no longer densely merged into the decoding segmentation pipeline. The modulation of hidden activities using the TD activities is introduced in this work in the hope of achieving higher performance results and robustness to out-of-distribution input perturbations.

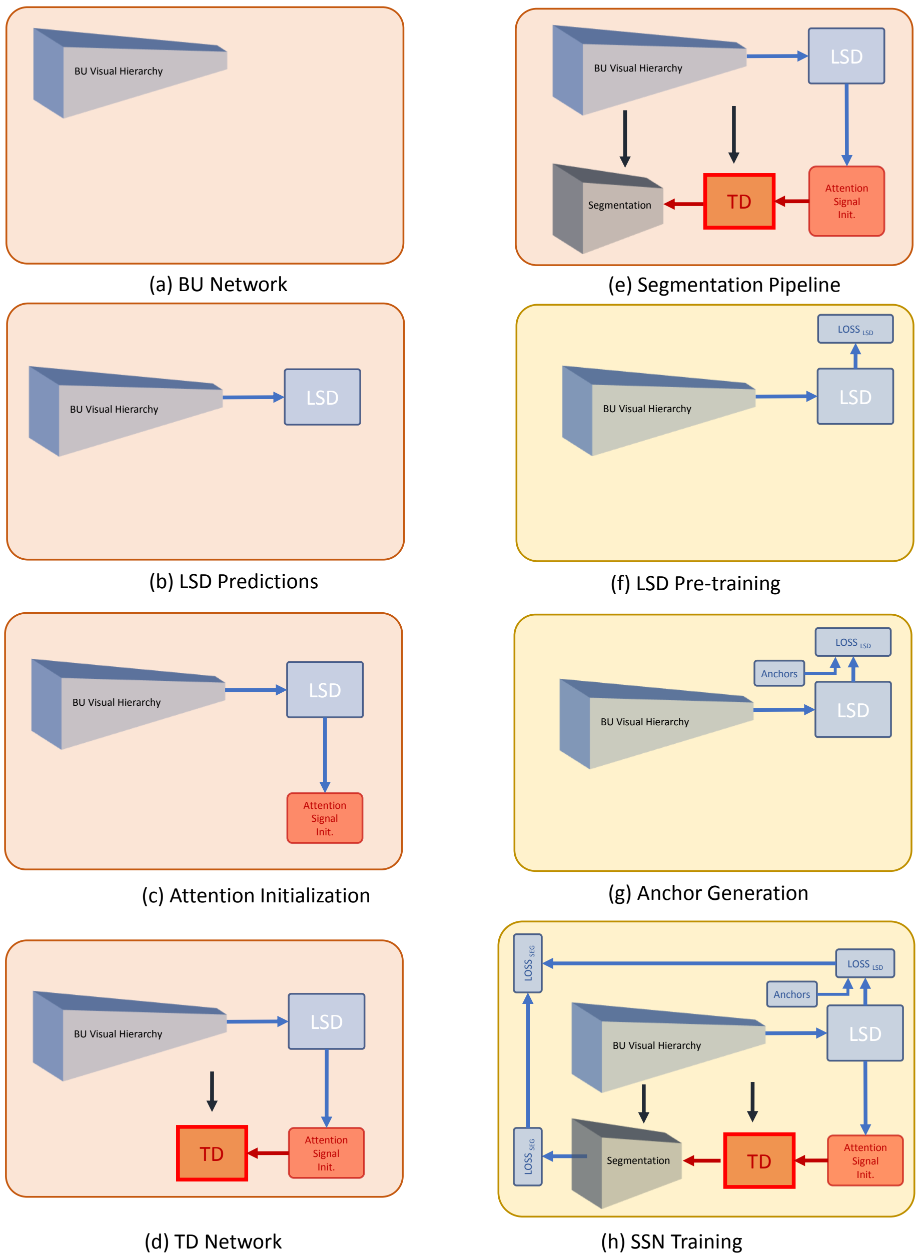


Figure 5.1: Illustration of the modular information flow of the Selective Segmentation Network (SSN) at each processing stage of the inference and learning phases. The stages in orange belong to the inference phase at which given some unknown test image, the predicted segmentation outputs are returned. The stages in yellow represent the learning phase at which SSN parameters are learned. The text provides details for each of the figure panels.

5.4 Selective Segmentation Network

The Selective Segmentation Network (SSN) consists of three major processing units: the BU network, the TD network, and the segmentation network: 1) the BU representation is the core visual hierarchy that consists of multiple parametric layers, 2) the TD selection mechanism produces gating activities using attentional traces throughout the visual hierarchy, and 3) the attentive segmentation network modulates hidden activities with gating activities at a number of different levels and merges them into a unified representation with gradual up-sampling for the final segmentation prediction. We are going to provide a procedural model overview in Sec. 5.4.1 explaining different stages of processing in SSN at the inference and learning phases briefly. In the subsequent sections, each stage will be explained with mathematical formulation and implementation details.

5.4.1 Method Overview

Fig. 5.1 demonstrates the computational stages of SSN at the inference and learning phases. We first begin with the stages involved in the inference phase and then move on to the learning phase. In the inference phase the information flows into SSN sequentially as follows. In Fig. 5.1 (a), the BU feedforward feature representation is defined by a convolutional neural network. The input image is transformed by multiple feature extraction layers into semantic information for some particular task prediction. Details are given in Sec. 5.4.2. In Fig. 5.1 (b), the LSD module is defined to deal with the multi-instance and multi-scale issues in semantic segmentation. The objective of LSD is to predict the top output units at which TD selection mechanisms need to be activated. Details are given in Sec. 5.4.3. In Fig. 5.1 (c), based on the prediction scores returned by LSD, the attention signal initialization unit determines the positions and scales to which attention must be deployed. We propose three different initialization strategies for which the details are given in Sec. 5.4.4. In Fig. 5.1 (d), TD selection begins from the initialization signal and traverses downward in a layer-by-layer manner. TD selection at each layer produces gating activities representing feature importance across spatial positions and channels. They influence the flow of hidden activities into the segmentation network. Details are given in Sec. 5.4.5. In Fig. 5.1 (e), the last stage of the inference phase is the segmentation prediction. At this point, both of the BU hidden and TD gating activities are produced for the input image and are the input to the segmentation pipeline. It has multiple levels of feature modulation, channel reduction, feature fusion, and spatial up-sampling. Further details are given in Sec. 5.4.6.

In the learning phase, we deal with the optimization of the SSN parameters for semantic segmentation given the new input data domain and task requirements. In the first learning stage as depicted in Fig. 5.1 (f), LSD pre-training is defined to adapt the feature representation of the BU network according to the LSD layers prior to the segmentation stage. In Fig. 5.1 (g), in order to optimize the BU and LSD parameters, proper target variables need to be produced from the provided ground truth bounding boxes. The LSD pre-training and anchor generation are explained in detail in Sec. 5.4.7. In Fig. 5.1 (h), the full SSN model is trained in the multi-loss setting using the LSD and segmentation loss functions. The former sits at the top of the hierarchy while the latter is at the bottom. The ground truth segmentation mask is the target variable used for the segmentation loss function. Details for the multi-loss setting are given Sec. 5.4.8.

Fig. 5.2 illustrates different parts of SSN in the learning phase all together, the information flow from one processing unit to another one, and the outputs at the top and the bottom of the hierarchy in more details. The input and output at each stage are labeled using the notations developed in the subsequent sections. It also depicts the joint loss function for the training of the entire network in an end-to-end manner using the SGD optimization algorithm. In the following, we explain the sequence of computational stages for the inference and learning phases in more detail.

5.4.2 Bottom-Up Feature Encoding

Definition: Information processing in SSN begins with the BU network that encodes the low-level input sensory data into high-level output semantic information at the top of the network. The BU network consists of multiple layers of feature extraction such as convolutional layers, non-linear transfer functions, and pooling layers. The spatial resolutions of the output maps throughout the network are gradually decreased while the feature channel size is increased. BU layers are defined according to a pre-defined network architecture such as AlexNet [30] or VGG-16 [59]. Part (a) of Fig. 5.1 illustrates the BU feature encoding as the first stage in the inference phase.

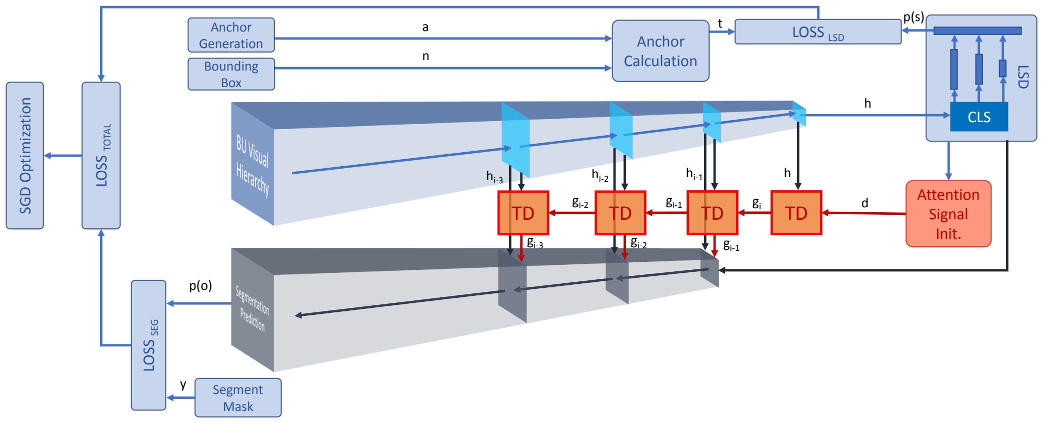


Figure 5.2: Illustration of SSN consisting of multiple parts such as the feed-forward BU representation, the classification LSD module, the TD selection network, and the up-sampling segmentation pipeline. Arrows show the information flow from one part to another part at the learning phase. The input and output at each stage are labeled using the variables which are defined in the subsequent sections.

Formulation: The training set $D = \{(x_i, y_i)\}_{i=1}^{N}$ contains N samples such that each sample consists of an input image $x \in \mathbb{R}^{3\times H\times W}$ and the ground truth y. The ground truth $y = (y^B, y^S)$ contains the bounding box annotations y^B of the category instances in the input image and the segmentation target mask $y^S \in \mathbb{R}^{1\times H\times W}$, where H is the input image height, W is the input image width. A pixel element on the segmentation mask at the vertical and horizontal position (h, w) has a category label $y^B_{hw} \in \{0, 1, \ldots, K-1\}$ for K different category labels including the background label 0. The BU pass consists of a multi-layer feedforward convolutional network

$$h = f(x; W_{BU}), \tag{5.1}$$

where f is a cascade of neural network layers, such as convolutional and pooling

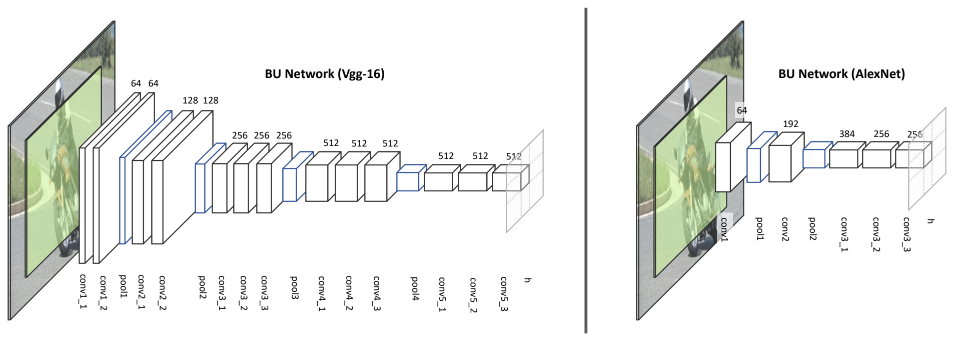


Figure 5.3: The BU network defined using the AlexNet and VGG-16 convolutional neural network architectures on the right and left respectively. The green box over the input image is the total receptive field size of a unit on the top feature map h. Blue boxes are pooling layers and the black boxes are convolutional layers with ReLU activation functions. Since the total receptive field size is smaller than the input image size, the top feature maps have size of greater than 1.

layers, and is parameterized with the set of connection weights W_{BU} depending on the underlying network architecture, x is the input image to the network, and $h \in \mathbb{R}^{C_f \times H_f \times W_f}$ is the hidden activity map at the top of the network with feature channel size C_f and spatial size $H_f \times W_f$. In a nutshell, the BU pass gradually transforms the raw input data using a number of parametric layers into a high-level semantic information for label predictions.

Implementation Details: In this work, we use AlexNet [30] and VGG-16 [59] network architectures to define the ordering, connectivity and parametrization of the BU layers. The former has a smaller number of layers while the latter has more layers with parameters. In this work, we use AlexNet as a proof-of-concept network due to the simplicity of the architecture and fewer number of parameters. We first begin experimenting with SSN using AlexNet and then later extent to VGG to validate the experimental evaluation results

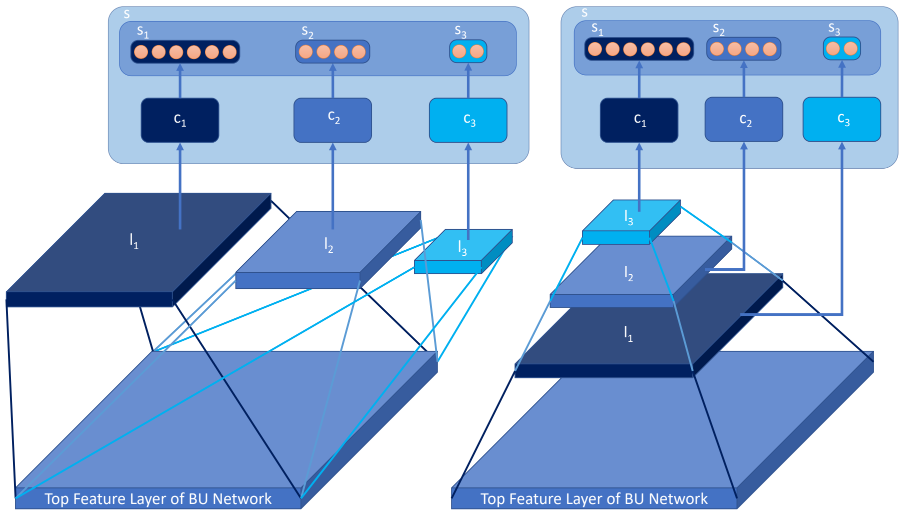

s
s1
s2
s3
c1
c2
c3
l1
l2
l3
Top Feature Layer of BU Network
s
s1
s2
s3
c1
c2
c3
l3
l2
l1
Top Feature Layer of BU Network

Figure 5.4: Parallel (right) and Sequential (left) architecture approaches to design the Loose Spatial Detection (LSD) module. Each shade of blue represents a group of layers with the intermediate layers l_i, the output prediction layers c_i and the output score maps s_i. The top feature layer is the last layer of the BU network that outputs the feature maps h. The layer connectivity of the parallel and sequential choices along with the spatial size reduction from one group to another is depicted schematically.

and demonstrate the generalization to a larger network. As illustrated in Fig. 5.3, the BU network based on the two architectures has the following set of layers respectively: {conv1, pool1, conv2, pool2, conv3_1, conv3_2, conv3_3} and {conv1_1, conv1_2, pool1, conv2_1, conv2_2, pool2, conv3_1, conv3_2, conv3_3, pool3, conv4_1, conv4_2, conv4_3, pool4, conv5_1, conv5_2, conv5_3}. The feature channel of the hidden maps at each layer are given in Fig. 5.3. The last layer of the BU network is Conv3_3 and Conv5_3 in the two architectures respectively. For the input image size 320×320, the hidden activity output h has the spatial size 20×20 in the both architectures.

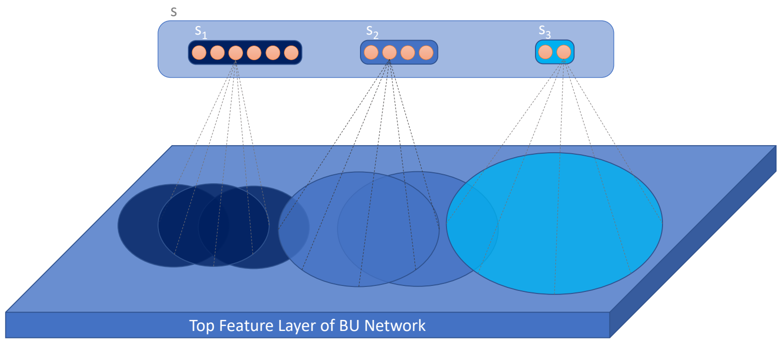


Figure 5.5: The receptive field size of three LSD groups over the input feature map h. The shades of blue represent the receptive field and the output score map of a particular group of LSD layer.

5.4.3 Loose Spatial Detection

Motivation and Objective: The BU network is initially loaded with a network trained for object classification on Imagenet benchmark dataset [24]. The definition of object classification is to recognize one single category instance in the input image. Thus, there is always one instance of one category in the input image. Consequently, classification models are required to return one single label output for an input image. The output unit apparently has a total receptive field as large as the input image size so then the entire image is covered. In object detection [24] and semantic segmentation [153], on the other hand, this is no longer the case and the input image is defined to have larger spatial size and may contain multiple instances of different object categories at various spatial locations and scales. Namely, the input data domain in these two tasks has multi-instance and multi-scale characteristics. As a result, the multi-instance and multi-scale aspects must be addressed by detection and segmentation models. Object detection approaches such as FRCNN [72] and SSD

[5] devise sliding-window approaches on the feature embedding space at the top of the visual hierarchy to produce label predictions at all possible spatial positions. The multi-scale issue in SSD [5] is addressed by defining multiple classification output layers at different levels of the visual hierarchy such that each has a wider receptive field and consequently covers a larger portion of the input image and is capable of predicting larger objects.

In this work, we also need to address the following aspects of the input data for semantic segmentation. SSN needs to be able to trigger TD selection for the positions and scales at which there are category instances. Following the same modeling approach as in SSD, we propose to deal with the multi-instance and multi-scale characteristics in semantic segmentation using the Loose Spatial Detection (LSD) module. It triggers the activation of the TD network at different positions and scales. LSD is a controlling unit that determines whether TD selection needs to start from an output unit at a particular position and scale.

Definition: LSD contains C groups of parametric layers. In each group, there are a number of convolutional and pooling layers and the last output prediction layer has a particular total receptive field size and output spatial size. The total receptive field size of a layer is a spatial span in the input image space that a unit on the output maps of the layer covers. The output of a layer has a 2D spatial size which is calculated according to the hyperparameter settings of the layer. Settings such as kernel filter size, marginal padding of the input activities to the layer, and sub-sampling rate (stride) specifies the spatial size of the output map. For instance, a layer with kernel filter size

5 has a wider total receptive field size in comparison to the kernel size 3. We define the combination and ordering of LSD layers in each group such that the total receptive field size of the units on the output prediction maps increases from the first group to the next while the output map spatial size decreases respectively. This is achieved using a combination of convolutional and pooling layers with appropriate kernel size, stride, and padding values. Fig. 5.5 schematically demonstrates the outputs of an LSD with three groups such that an node in the output score map s_1 has a smaller receptive field size while the number of nodes is larger. As we move to the next two groups, the receptive field size increases and the number of nodes in the output maps decreases. As illustrated in Fig. 5.1 (b), the LSD module is used right after the end of the BU pass of information processing in the second stage of the inference phase.

Formulation: h, the output of the final BU layer at the top of BU network, is fed into LSD

$$s = c(h; W_{LSD}), \tag{5.2}$$

where $s \in \mathbb{R}^{K \times A}$ is the output score map with A units such that at each unit, predictions for K category labels are produced, and W_{LSD} is the set of LSD weight parameters. LSD module $c(h; W_{LSD}) = \{(l_i, c_i)\}_{i=0}^{C}$ adds C extra groups of layers on top of the BU network where l_i is the set of intermediate parametric layers in group i and c_i is the final output prediction (classification) layer returning the output score map s such that $s = \{s_i\}_{i=0}^{C}, s \in \mathbb{R}^{K \times A}, A = \sum_{i=0}^{C} |s_i|$. $|s_i|$ is the total number of output units returned by the classification layer c_i. We refer to the last output layers in LSD as output prediction, discrimination, or classification layers interchangeably.

We propose to experiment with two possible approaches to define the connectivity of layers in the LSD module as illustrated in Fig. 5.4: the sequential and parallel architecture designs depicted in the right and left parts respectively.

As the name of the two design choices imply, the LSD output predictions are computed using a combination of parallel or sequential groups of layers. In the former, the C groups process the input feature maps h in a disjoint and parallel manner while in the latter, a group with a larger receptive field sits on top of the other with a smaller receptive field. Additionally, in the sequential case the parametric feature representation is shared among all of the other underlying groups by passing the output of one group as the input to the next one. In the parallel design, on the other hand, each group maintains a separate feature representation on top of the input feature maps h to produce the output predictions. So the feature representation throughout one group is not shared with the layers in another group.

In the sequential design, the set of intermediate layers l_i not only pass information to the set of intermediate layers l_{i+1} of the next group but also feeds into the classification layer c_i to output label score maps s_i. Each unit in s_i returns the confidence scores for $K + 1$ category labels. The category with the highest score is basically the category for which the TD selection needs to begin at this unit. Each set of intermediate layers $l_i = \{u_i, o_i\}$ contains a convolutional layer u_i with kernel size 1×1 followed by a convolutional layer o_i

with kernel size 3×3. In the parallel design, on the other hand, each set l_i only feeds into the classification layer c_i and contains a number of convolutional and pooling layers.

The parallel and sequential LSD types are schematically demonstrated in Fig. 5.4 for three groups of layers. The output h of the top feature layer in the BU networks is fed into the groups of layers. Boxes in different shades of blue represent the set of intermediate layers l_i for $i = \{0, 1, 2\}$. They output information into the final prediction layers c_i for the prediction of the score maps s. The output maps s_i are returned by the prediction layer c_i such that $|s_i| < |s_{i+1}|$ the output score map size of the first group is smaller than the second and the second smaller than the third group.

Implementation Details:

LSD has three groups of layers each of which has a set of intermediate layers followed by a final prediction layer. We define the number of groups C to be three as it is sufficient to fully cover the small, medium, and large category objects. The three sets of intermediate layers respectively consist of {c1x1, c1x1}, {c3x3-p2-d2, c1x1, c3x3-p1}, and {m3x3-s2, c3x3-p2-d2, c1x1, c3x3-p1}. c, s, p, d, m stands for a convolutional layer, stride, padding, dilation, and max pooling values respectively. c3x3-s2-p2-d2 defines a convolutional layer with the kernel size 3x3, stride 2, marginal padding 2, and dilation 2. For the sake of brevity, the default values of s1, p0, and d1, are ignored. Both of the sequential and parallel LSD predictors have layers with the same set of settings and hyperparameters. They only differ in terms of the ordering and connectivity of the groups with respect to each other. The input to LSD is taken from the intermediate layer conv5 and conv5_3 in AlexNet and VGG respectively. The output score map s is used by the attention initialization unit in the inference phase and the LSD loss function in the learning phase.

5.4.4 Attention Initialization

Definition: Once the LSD module is finished computing the output prediction tensor s, we need to determine the set of elements for which the TD selection mechanisms need to be activated in the third stage of the inference phase as illustrated in part (c) of Fig. 5.1. We propose to experiment with three different initialization strategies described in the following. The attention initialization module receives the LSD output tensor s and produces an initialization signal according to one of the three strategies. The TD selection pass is initialized by an input attention signal d such that $d = \{d_i | d_i \in \mathbb{R}^K\}_{i=0}^{A}$. d contains the same number of elements as s does and is initially a tensor of zero elements. The initialization strategy determines the category for which the TD selection mechanism will be activated for a particular element d_i. This is achieved by the one-hot encoding representation described as follows. There might be elements for which there is no TD selection activated.

Ground Truth Strategy: This strategy sets the elements of the attention signal d to one according to the ground truth anchor labels which are used for LSD prediction training described in 5.4.7:

$$d^{GT} = \{d_{ij} = 1 | j = t_i\}_{i=0}^{A}, \tag{5.3}$$

where t_i holds a category label for which the anchor i is determined to have the highest IoU with a ground truth bounding box of an object of the category. The goal of this strategy is to measure the performance of the LSD module to activate the TD pass according to the ground truth target values rather than the LSD output confidence scores.

Top-1 Strategy: This strategy is the most straightforward approach to initialize the attention signal. It finds the category label of the maximum output score value and then set the signal value for the category label to one:

$$d^{top-1} = \{d_{ij} = 1 | j = \text{argmax}_k s_{ik}\}_{i=0}^{A}, \tag{5.4}$$

in which $s_i \in \mathbb{R}^K$ is the LSD score element returned by the LSD module. The reliability of LSD is verified when the final segmentation performance of SSN initialized using this strategy is close to the ground truth strategy. It implies LSD has learned to determine the units for which TD selection is essential to be activated so then the segmentation network benefits from a rich set of modulated features.

Thresholding Strategy: There is no purpose in imposing the TD selection process in spatial regions where there is little confidence that a target is present (this would be a false alarm). Therefore, in order to reduce redundant TD selection imposed by false alarms, we threshold the maximum confidence scores using a cross-validated thresholding value $\theta^{attention}$:

$$d^{\theta} = \{d_{ij} = 1 | j = \text{argmax}_k s_{ik}, s_{ij} > \theta^{attention}\}_{i=0}^{A}. \tag{5.5}$$

It reduces the redundancy in TD pass and consequently lowers the interference imposed by misleading noisy features for the segmentation pipeline. Additionally, this strategy reduces the processing time required for the completion of the TD selection pass and hence the overall SSN processing time decreases. s contains the probability confidence values ranging from zero to one. We cross-validate a range of thresholding value $\theta^{attention}$ and find that $\theta^{attention} = 0.9$ is tight enough to improve the segmentation accuracy and maintain the TD selectivity. This value is used during the experimental evaluation.

5.4.5 Top-Down Selection

Definition: The TD network computes the selection patterns through which the gating of the BU activities into the segmentation network is performed. The TD selection mechanisms are activated using the attention signal initialization module. Once the initialization signal tensor is set, the TD selection network starts processing the BU hidden activities to compute the gating activities at each layer and then the selection is passed to the layer below. This process continues until the TD pass stops at some early layer of the visual hierarchy. The information flow in the inference phase from the initialization module to the TD network is illustrated in part (d) of Fig. 5.1. Further details are given in Fig. 5.2 by depicting the TD selection mechanism at each layer and the information flow from one layer to another layer in the TD network. The gating information flows from the top layers to the intermediate and early layers in the TD pass. It is shown in STNet model [108] that the TD gating activities are sufficiently representative for object localization, and we hypothesize that they are reliable to select features for object segmentation. We experimentally support the hypothesis and show that the gating activities indeed improve the segmentation accuracy over the baseline model without a similar gating mechanism.

Formulation: We follow STNet [108] formulation for the TD selection pass. The TD pass begins from the elements d_i, which are set to one by the initialization module. Those that are zero will not participate in the TD traversal. We define the TD network as

$$g = n(d, h, W_{BU}, W_{LSD}), \tag{5.6}$$

in which $n = \{n_i\}_{i=J}^{V} \,|\, g_i = n_i(g_{i+1}, h_i, w_i)$ is a set of sequential TD layers called one after each other, d is the input attention signal, h is the set of the BU hidden activities at all layers, and W_{BU} and W_{LSD} are the set of kernel parameters of the BU network and the LSD module respectively. At every TD layer, g_{i+1} and g_i are the input and output gating maps respectively, h_i is the hidden activity map passed from the BU layer i, and w_i is the kernel filter weights of the BU convolutional layer. J is the penultimate layer in the visual hierarchy such that $g_{J+1} = d, h_J = s$ and V is the level at which the TD pass ends.

The selection mechanism n_i is implemented by three computational stages. All three stages are performed in the local scope of the receptive field of a node. The computation in the stages is based on the element-wise multiplication of the hidden activities falling inside the receptive field and the kernel filter weights. We call this set Post-Synaptic (PS) activities hereafter. The first stage takes care of noise interference reduction by running a competition among PS activities, and determining the set of winners. It has an adaptive thresholding mechanism to implement a local competition between PS activities. The second stage performs grouping and selection of the winners according to spatial and statistical criteria for the convolutional and fully-connected layers. Lastly, the third stage normalizes PS activities of the selected group such that they sum to one and propagates the gating activity proportional to the normalized PS values to the localized gating units in the layer below. The gating maps at each layer represent the selection patterns that will be used for feature modulation in the segmentation pipeline.

Implementation Details: The TD selection is computed at each layer of the visual hierarchy, gating activities are determined and the selection is passed to the next layer, which is below the current one. Layer by layer selection is executed until a particular stopping layer is met. The *pool1* and *pool3* layers are the stopping layer V in AlexNet- and VGG-based BU networks respectively.

Unlike STNet model, SSN does not have any fully-connected layers. All the fully-connected layers are replaced with the convolutional layers that have kernel size 1×1. We refer to 1×1 convolutional layers as collapsed convolutional layers hereafter. Collapsed convolutional layers are technically fully-connected layers that are applied over two-dimensional feature maps rather than a one-dimensional feature vector. To address this requirement, we implemented the TD selection stages for collapsed convolutional layers using the stages for fully-connected layers.

The TD pass has one hyperparameter at each layer in the second selection stage while the first and the last stages do not have any hyperparameter. The second stage of the TD pass in STNet for the collapsed convolutional layers has a statistically-motivated thresholding value that determines how tight or loose the selection is. We replace it with the Winner-Take-All (WTA) mechanism since the nature of the object segmentation in this work is different from the object localization STNet was developed for. For the typical convolutional layers, there is a fusion factor that determines how much emphasis should be given to the spatial contiguity or the total activity strength. In STNet, it is called α. We experimentally choose to set $\alpha = 0.2$.

5.4.6 Segmentation Prediction

Definition: The BU representation and TD selection integrate into a unified pipeline in the segmentation network. The segmentation network is defined to learn the spatial and feature correlations of the modulated feature activities by parametric up-sampling of the feature planes for the final segmentation predictions. Hidden activities in the BU layers represent input sensory data according to an optimization policy such that an objective loss function is minimized. However, the spatial resolution is reduced along the visual hierarchy due to the gradual increase in the receptive field sizes and sub-sampling rates. Rather than adding parametric sub-sampling layers in a brute-force manner to generate segmentation predictions similar to FCN-based approaches, TD selection fills the gap between the spatial acuity and semantic richness by computing selective gating activities at each layer of the hierarchy. These

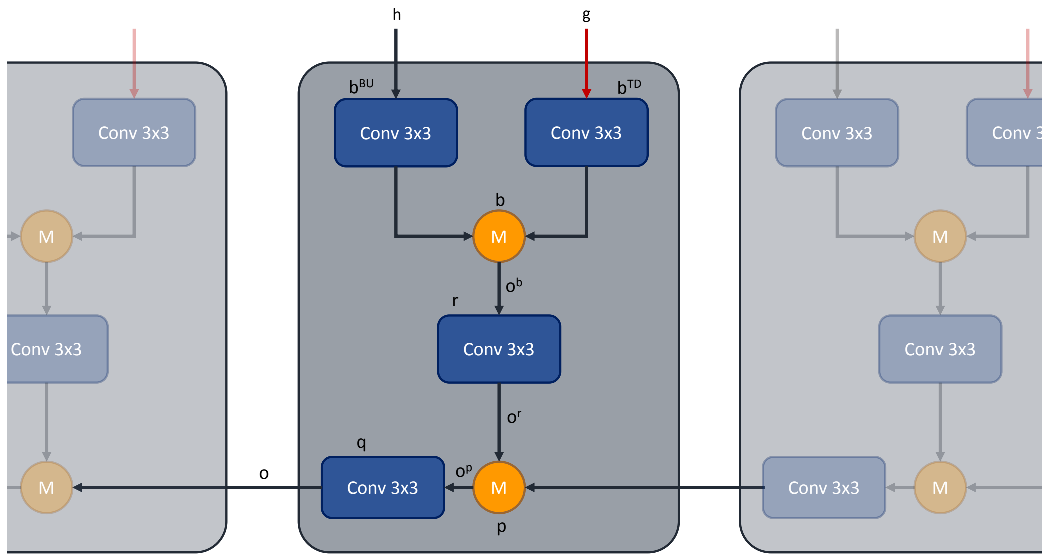


Figure 5.6: Illustration of the segmentation network with different parametric and modulation nodes. Each block receives the hidden (blue) and the gating (red) activity inputs. The selective gating units modulate the hidden units at the first node M. At each layer, after input fusion, information is integrated into the main segmentation pipeline using the second modulation node M. We conduct experiments on three different types of modulations: addition, multiplication, and concatenation. The layer label subscript i is neglected for the sake of brevity.

activities are used to modulate hidden activities in the spatial and feature dimensions at multiple levels. Part (e) of Fig. 5.1 schematically illustrates the information flow from the BU network for feature encoding to the LSD for multi-instance and multi-scale label predictions. Attention initialization and TD network produce the selection patterns at multiple levels of the hierarchy. Lastly, the segmentation network produces the segmentation output maps through a number of parametric layers. This is the last stage of the inference phase and the segmentation output map is used to predict the category label of each pixel of the input image.

Fig. 5.2 shows that the segmentation network has a number of processing layers. Each layer receives two inputs one from the corresponding BU layer and one from the corresponding TD layer. The inputs are transformed, fused, and up-sampled using a number of parametric building blocks at each layer. All of these block are necessary to form a representation given the inputs for the segmentation prediction.

Formulation: Once the two BU and TD passes are completed, Information is passed to the segmentation network

$$o = m(h_i, g_i; W_{seg}), \tag{5.7}$$

in which $m = \{(b_i, r_i, p_i, q_i)\}_{i=0}^{M}$ consists of M segmentation layers, W_{seg} is the set of segmentation weight parameters, h_i and g_i are the hidden and gating activities at layer i respectively, and o is the segmentation output map which will be used in the segmentation loss function in Sec. 5.4.8.

The BU hidden activities h_i and the TD gating activities g_i have two different data distributions, one is densely active and the other sparsely active due to the selective nature of the TD mechanisms. Therefore, the parametric layers $o_i^b = b_i(b_i^{BU}(h_i, W_{seg}^{BU}), b_i^{TD}(g_i, W_{seg}^{TD}))$ are used to learn an appropriate transformation of the two inputs before the TD modulation of BU activities. Three types of modulation are defined for the fusion of the TD and BU activities: $b(u, v) = u \diamond v | \diamond \in \{\oplus, \odot, \ominus\}$, $\oplus$ tensor summation, $\odot$ tensor multiplication, $\ominus$ tensor concatenation.

After the modulation unit b_i, there is a parametric layer $o_i^r = r_i(o_i^b; W_{seg}^r)$, a concatenation layer $o_i^p = p_i(o_i^r, o_{i-1}) \,|\, p(u, v) = u \ominus v$ to fuse the incoming

Network		level 1	level 2	level 3
AlexNet	name	conv3_3	pool2	pool1
	input	256	192	64
	b	128	96	48
	r	128	96	48
	q	96	48	32
VGG	name	conv5_3	pool4	pool3
	input	512	512	256
	b	384	256	128
	r	384	256	128
	q	256	128	64

Table 5.1: The output channel size of computational units in the segmentation layers is given for AlexNet and VGG at three different levels. b, r, q are the units defined in 5.4.6.

information at layer i into the information at layer $i+1$ in the segmentation pipeline, and lastly another parametric layer $o_i = q_i(o_i^p; W_{seg}^q)$ followed by a spatial up-sampling layer. The modulation type $\diamond$, the number of segmentation levels M, and other hyper-parameters are cross-validated in the experimental evaluation in Sec. 5.5. All these building blocks at a segmentation layer are illustrated in Fig. 5.6 with details such as the information flow from one computational unit to another one, the name and the computation type of each unit.

Implementation Details: The output channel size of each computation block at a segmentation layer is given in Table 5.1. All of the convolutional layers in the segmentation network have the kernel size of 3x3 with stride 1, padding 1, and dilation 0 unless otherwise mentioned. As illustrated in Fig. 5.6, h_i and g_i are the two inputs to the segmentation layer i. They are first fed into b^{BU} or b^{TD}, which are 3×3 convolutional layers, to reduce the feature channel size for a compact feature representation. Next, the outputs are fused into one feature tensor by the modulation operation b_i which can be tensor concatenation, addition, or multiplication. The output of the modulation unit is sent into the convolutional layer r_i to further reduce the feature redundancy before getting fused into the main segmentation pipeline. The feature tensor o_i^r at this point is merged into the main segmentation pipeline using the feature concatenation unit p_i along the feature channel dimension. The concatenation of o_i^r is with the segmentation activities o_{i+1} passed from the segmentation layer $i + 1$. At the first level, the LSD label predictions are used for the concatenation. This helps the error gradient signals computed using the segmentation loss function to reach to the LSD module and consequently flow downward through the BU visual hierarchy. This brings faster optimization convergence during the training phase. Next, the convolutional layer q_l reduce the feature channel size further while the output spatial size is increased using a bilinear up-sampling layer by the factor of 2. o_i is the output of the segmentation layer i and the input tensor for the next segmentation layer $i - 1$. The details on the number of segmentation levels and the output channel size of each computational block is given in the Table 5.1. We follow this layer definitions in the experimental evaluation in Sec. 5.5.

5.4.7 LSD Pre-training

Definition:

The learned feature representation of the pre-trained convolutional neural network loaded on the BU network needs to be adapted to the new data domain and visual task of semantic segmentation. The BU network parameters are initialized by the parameters of a convolutional neural network pre-trained on the Imagenet [24] dataset for the task of object classification. The input images in Imagenet contain one single instance of an object category. The label prediction output of the classification network has the size of $K \times 1 \times 1$ since the input image size is smaller than the overall network receptive field. As a result, the loss function is measured only on one set of label predictions for all categories. SSN, on the other hand, deals with input images that may contain multiple instances of the pre-defined labeled categories. Additionally, due to the larger size of the input images, the network returns output maps with spatial sizes greater than 1. This shift of domain and task requires a preliminary stage of fine-tuning of the BU network using the loss function defined on the LSD module.

The BU network needs to learn to accommodate for the new task and domain requirements in this first stage of the learning phase as is depicted in part (f) of Fig. 5.1. We define the objective function $\mathcal{L}_{\mathcal{LSD}}(\hat{p}, t) = \frac{1}{N_D} \sum_i^A \mathcal{L}_D(\hat{p}_i, t_i)$ on top of the LSD module to minimize the loss of the prediction distribution $\hat{p}$ for the target label t. $\hat{p}$ is computed using the Softmax transfer function from the output score maps s. The SGD optimization algorithm updates the BU and LSD weight parameters using the gradient signals computed by the backpropagation algorithm.

We will fine-tune the pre-trained BU network following a class-specific approach inspired from the SSD object detection model [5]. For an input image x_i, there is a set of bounding box annotations y_i^B in the dataset D which needs to be utilized to generate appropriate target labels for the training of the LSD module.

Unlike the dominant object detection models, SSN only needs to loosely know if a LSD output unit is required for activating a TD selection mechanism or not. Therefore, it is not needed in SSN to have multiple scale, aspect ratio, and offset value predictions at each output unit. The LSD module in SSN only requires to output category label predictions which are harnessed later on by the attention initialization unit for the activation of a number of TD selection mechanisms. Part (g) of Fig. 5.1 demonstrates the role of the anchor processing unit for the computation of the LSD loss function.

Formulation: For the training of LSD, we generate target labels $t = \{t_i\}_{i=0}^{A}$ to fine-tune the weight parameters of the BU network and the LSD module. We follow an anchor generation approach commonly practiced for object detection in [5, 72]. We define an anchor a_{ij} for each LSD output unit j at the group layer i, and calculate its box coordinates from the total receptive field size at that level. We then propose to set the target labels according to the following policy:

$$t_{ij} = \begin{cases} k & IoU(a_{ij}, y_k^B) > \theta^{pos} \\ 0 & IoU(a_{ij}, y_k^B) < \theta^{neg} \end{cases},$$

where $k \in \{1, \dots K\}$. the target t_{ij} is assigned to a category label for which the Intersection-over-Union (IoU) metric of the corresponding anchor box a_{ij} with a ground truth box g_k for the category k is over the positive threshold value $\theta^{[pos]}$. It is zero if the IoU metric is below the the negative threshold value

$\theta^{[neg]}$. We always ensure that there is at least one anchor set for a ground truth bounding box.

Implementation Details: Following the experimental setting in [72] and preliminary experimental results, we choose to set $\theta_{pos} = 0.5$ and $\theta_{neg} = 0.3$ in the experimental evaluation phase. A target label has the category label of the box overlapping with the corresponding anchor if the IoU value of the box with the anchor is above θ_{pos}. It has the background label zero if the IoU of the two is below θ_{neg}. We set the otherwise to the don't-care label value 255. Once all of the target labels are set for the bounding boxes annotations of the mini-batch samples, we randomly keep a maximum number of 128 target labels per mini-batch samples such that the ratio between the negatives and positives is at most 1:3 while the rest are set to 255. We set the element-wise cross-entropy loss function $\mathcal{L}_{\mathcal{LSD}}$ to exclude the loss terms of the output units for which the target labels are set to 255.

Given an input image, the LSD output units are computed using a feed-forward pass and the target labels are determined using the ground truth bounding boxes. We fine-tune the parameters of the BU network and the LSD module using the SGD optimizer with the initial learning rate 10^{-3}, momentum 0.9, weight decay 0.0005, and batch size 4 for 15 epochs. We evaluate the performance of LSD using similar segmentation metrics namely the mean pixel and the mean IoU performance metrics. Once the LSD pre-training is converged, we train the SSN model using the multi-loss function described in Sec. 5.4.8.

5.4.8 Multi-loss Training

Definition: SSN training using the multi-loss function is the last stage of the learning phase as illustrated in part (h) of Fig. 5.1. The BU network and LSD module parameters are loaded with the converged set of parameters in the LSD pre-training stage. The optimization algorithm considers two loss functions at the opposite ends of the visual hierarchy. The SSN parameters of the converged model is used in the inference phase for the segmentation prediction of unknown test images.

Formulation: SSN has output layers at the two ends of the visual hierarchy. The first receives the LSD score maps s as inputs at the top of the visual hierarchy and outputs a discrete probability distribution $\hat{p}(s) = \{\text{Softmax}(s_{i0}, \ldots, s_{iK-1})\}_{i=0}^{A}$ over K categories including the background. On the other side at the bottom of the hierarchy, segmentation output map o is fed into another output layer and returns similarly a discrete probability distribution $\tilde{p}(d) = \{\text{Softmax}(d_{i0}, \ldots, d_{iK-1})\}_{i=0}^{H\times W}$, where H and W are the height and width of the input image. The overall multi-loss objective function is a weighted sum of the LSD loss $\mathcal{L}_D$ and the segmentation loss $\mathcal{L}_S$ terms

$$\mathcal{L}_{Total}(\hat{p}, t, \tilde{p}, y) = \frac{1}{N_D} \sum_i \mathcal{L}_D(\hat{p}_i, t_i) + \alpha \frac{1}{N_S} \sum_i \mathcal{L}_S(\tilde{p}_i, y_i), \tag{5.8}$$

in which both of the loss functions $\mathcal{L}_D$ and $\mathcal{L}_S$ are defined using the element-wise negative log likelihood (NLL) function for the true target labels t_i and the segmentation mask y_i^S respectively, and $N_D = A$ and $N_S = H \times W$.

Implementation Details: The output of the last up-sampling layer is the input of the final segmentation prediction module. This module simply consists of one 3x3 and one 1x1 convolutional layers: the former keeps the feature channel size intact, and the other reduces it to the number of category labels K. The output of this module is the confidence scores used by a Softmax layer to produce multinomial probability values. Similar to LSD pre-training procedure, we use an element-wise cross-entropy loss function to optimize the set of all of the weight parameters of SSN $W_{SSN} = \{W_{seg}, W_{LSD}, W_{BU}\}$.

Since we have a multi-loss objective function, the error gradient signals are propagated from the two ends of SSN: the first loss term propagates error signals from the top of the visual hierarchy all the way to the input layer, while the second loss propagates the error signals from the bottom of the visual hierarchy. The error signals measured from the segmentation loss propagate into the BU network according to the modulatory patterns generated by the TD gating activities. This has an important impact on the underlying representation of the BU network. While the LSD loss keeps the representation fidelity of the BU network, the second loss term updates the parameters of the hierarchical transformation for a more robust and adapted segmentation prediction.

5.5 Experimental Results

We evaluate the performance of SSN on object segmentation to support the role of a top-down selection mechanism in neural network approaches. Semantic segmentation is the task that is defined to predict segmentation masks of

Model	Parallel		Sequential	
	m Accuracy	m IoU	m Accuracy	m IoU
AlexNet	54.3	39.8	52.6	39.1
VGGNet	66.7	55.6	67.1	53.6

Table 5.2: Parallel and Sequential LSD performance results on the Pascal VOC 2012 validation set once the BU network is fine-tuned on the extended Pascal dataset.

a pre-defined number of semantic categories [152, 153, 25, 154]. Challenging benchmark datasets such as PASCAL VOC [159], the Cambridge-driving Labeled Video (CamVid) [160], and Horse-Cow Parsing [161] datasets are used for experimental evaluation of SSN.

SSN is implemented using PyTorch[1] [162], an open source deep learning platform which is well-known for its automatic differentiation engine. The TD pass is integrated into the main implementation using the open-source CUDA library provided by[2] STNet [108]. LSD pre-training begins with the Imagenet pre-trained models provided by the PyTorch Model Zoo repository. The core architecture of the BU network in all of our experiments are defined based on either AlexNet [30] or VggNet [59] networks.

5.5.1 Semantic Segmentation

Dense image labeling such as semantic segmentation requires pixel-level predictions. In this section, we first measure the performance of a variety of SSN configurations on predicting accurately object segmentation of various categories of the input images. Later, we provide the comparison with the

[1]https://pytorch.org/
[2]https://github.com/mbiparva/stnet-object-localization

SSN Variant	mean Accuracy	mean IoU
SSN-GT	53.7	41.9
SSN-MAX	51.9	40.4
SSN-THD	53.5	41.4
SSN-THD-CAT	50.6	40.2
SSN-THD-ADD	53.5	41.4
SSN-THD-MUL	54.1	42.1

Table 5.3: Comparison of different variants of SSN using AlexNet on PASCAL VOC valid 2012. We use mean pixel accuracy and mean IoU metrics to report the performance. CAT, MAX, THD, ADD, and MUL stands for concatenation, top-1, and thresholding, additive, and multiplicative.

Network	Model	Mean Accuracy	Mean IoU
AlexNet	FCN	-	39.8
	SSN	54.1	42.1
	FCN++	-	48.0
	SSN++	55.8	49.4
VGGNet	FCN	-	56.0
	SSN	64.6	58.7
	FCN++	75.9	62.7
	SSN++	76.8	64.3

Table 5.4: Comparison of SSN with the baseline model on PASCAL VOC validation set using mean IoU metric. SSN++ is trained on the extended training set.

Method	mean IoU (%)
FCN [146]	62.7
DeepLab [163]	64.2
G-FRNet [3]	68.7
DeepLab-ASPP [152]	68.9
SSN(ours)	64.3

Table 5.5: Comparison of SSN with the state-of-the-art on PASCAL VOC 2012 valid set. All methods use VGGNet as the backbone network.

baseline model and discuss the aspects the superiority originates from.

Dataset and Evaluation: A popular and challenging benchmark dataset for semantic segmentation is PASCAL VOC 2012 [159, 143]. The dataset contains 1464 training and 1449 validation sample images. Each image may have a number of instances of 21 object categories (including the background category). To enable comparisons with previous works, the training set is expanded with extra labeled data [149]. We measure the segmentation performance using the mean pixel accuracy and mean IoU metrics [146]. In the training phase, we resize sample images to have the smallest side of 320 and then take a random crop of size 320x320. In the evaluation phase, we resize images to have the largest side of 320 and pad the smallest side of the RGB image the mean pixel values and the segmentation mask with don't care pixel values. We follow this strategy for all the experiments in this work.

Quantitative Results: We first fine-tune the Imagenet pre-trained AlexNet and VggNet networks using the LSD training protocol on the extended Pascal VOC 2012 dataset. The performance results are given in Table 5.2 using the two evaluation metrics. The performance metric results using the parallel LSD outperforms the sequential one for both of the convolutional neural network model. Since the problem in LSD formulation is merely a classification task, the parallel LSD benefits from the separation and independence of label predictors and consequently survives over-fitting to the training set. We stick to the parallel LSD configuration for the rest of experimental evaluation.

We report the performance of different setups of SSN using the AlexNet architecture in Table 5.3. First, we measure the effect of the attention signal initialization strategy on the SSN performance. Apparently, the GT initialization strategy is superior over the other two since it minimizes redundancy and noise interference imposed by TD selection from false alarm units. Mean IoU of 40.4 in the max strategy increases to 41.4 in the thresholding strategy. This indicates that the false alarm interference is reduced by thresholding units with prediction confidence below the $\theta^{attention}$ of 0.90.

Next, we investigate the effect of the modulation types of the segmentation network. The selective nature of the TD gating activities is supported since the multiplicative modulation outperforms the other two types. This is reminiscent of the surround suppression phenomenon in human vision predicted in the Selective Tuning model of visual attention [7]. Using the multiplicative modulation, BU hidden units that are not selected by the TD gating units do not participate in the computation of the segmentation network and therefore information flow is blocked or reduced in such units. This underlines how a hierarchical selective mechanism can dynamically suppress redundancy in the visual representation and lead to more robust task predictions. We use the thresholding initialization strategy and multiplicative modulation for the rest of experiments.

Comparison with the baseline: The best variant of SSN is trained on PASCAL VOC 2012 training set and the extra data of [149] using both of the network architectures. The performance is reported on the PASCAL VOC 2012 validation set in Table 5.4. Samples in the validation set are excluded from the extended training set. The SSN performance is compared with the baseline model of FCN [146]. The first goal is to introduce a well-established

TD selection mechanism and highlights the aspects that lead to improvements over the FCN model. FCN introduced dense and parametric skip connections from early layers for parametric up-sampling of label scores of a classifier for segmentation. In all cases in Table 5.4, SSN improves the mean IoU results over FCN for both of the architectures.

The quantitative results verify the modulatory role of the TD selection mechanism in SSN. SSN benefits from an architecture that employs TD selection to begin from high-level semantic layers and traverse to intermediate-level feature representations. The TD traversal outputs selection patterns at each layer that highlight important regions and features along the visual hierarchy. The results emphasize that a systematic hierarchical gating of the information flow from the early feedforward layers into the segmentation pipeline has positive impact on the evaluation metrics.

Comparison with the state-of-the-art: We compare the best segmentation performance of SSN on the PASCAL VOC 2012 validation set with the performance of the state-of-the-art methods in Table 5.5. SSN outperforms FCN by 1.6% and is par with DeepLab which benefits from features such as multi-scale prediction method and Atrous (strided) convolutional layers with large field of views. The multi-scale method concatenate feature maps form the early layers with the network's last layer feature map. This helps the last feature maps to gain extra information to compromise for the lost of information imposed by the hierarchical feature encoding. SSN, on the other hand, does not benefit from multi-scale skip connections but rather relies on the TD selection mechanism to route through the network and highlights the important features for segmentation. G-FRN and DeepLab-ASPP are by approximately 4% more accurate in predicting semantic segmentation in comparison with SSN. This is mainly due to the fact that SSN does not benefit from the features such as the stage-wise supervision in [3] and the Atrous Spatial Pyramid Pooling (ASPP) in [152]. The former provides strong supervision at multiple-levels of the feature hierarchy using different loss functions. This facilitates the error gradient propagation throughout the network hierarchy. ASPP employs parallel branches with different atrous rates at the top fully-connected layers to cover a wide range of filed of views. SSN uses none of these features and this explains the reason it falls behind these two methods.

Additional Experimental Evaluation: To further support experimentally the role of the TD mechanism for the attentive segmentation framework of SSN, we compare performance of SSN and FCN on two challenging datasets: CamVid and Horse-Cow datasets. CamVid has 701 frames of urban driving extracted from high resolution video recordings. Following [164, 165, 166], we consider 11 large semantic categories, down-sample images by a factor of two (i.e. 480x360), and split them into the training (367), validation (100), test sets (233). Horse-Cow part parsing dataset contains semantic labeling of four body parts (head, leg, tail, body). Following [161], we split the dataset into 294 training and 227 test images. We follow the resizing and padding protocol introduced for PASCAL dataset.

The results in Table 5.6 reveals the efficiency of the TD selection to obtain segmentation robustness is consistent across different datasets. SSN improves on the mean IoU metric values of FCN for CamVid and Horse-Cow

Network	Model	Mean Accuracy	Mean IoU
CamVid	FCN	66.4	57.0
	DeepLab-LargeFOV*	-	61.6
	G-FRNet[3]	-	68.0
	SSN	73.9	64.7
Horse-Cow	FCN	77.3	63.1
	DeepLab-LargeFOV*	-	62.7
	G-FRNet[3]	-	68.1
	SSN	77.2	65.2

Table 5.6: Comparison of SSN with the baseline and state-of-the-art on two additional segmentation benchmark datasets: CamVid and Horse-Cow. The results are reported on the test sets. Note that the DeepLab-LargeFOV* results are taken from[3].

datasets and is on par on the mean accuracy metric values. We further compare the performance of SSN on these two benchmark datasets with DeepLab-LargeFOV [163] and G-FRNet [3]. SSN outperforms DeepLab on the Horse-Cow and CamVid datasets. However, similar to the results of the PASCAL VOC dataset, SSN cannot compete with G-FRNet on these two datasets due to the extra design features G-FRNet benefits from. The evaluation results on these dataset are consistent with the previous experiments on the PASCAL VOC dataset. This reveals that the role of TD selection mechanism generalizes across different benchmark datasets for semantic segmentation.

Qualitative Results: We qualitatively compare SSN with the baseline FCN model on PASCAL, Cam-Vid, Horse-Cow part parsing datasets in figures 5.7, 5.8, 5.9 respectively. The selectivity and modulatory role of the TD processing on the BU processing for the lateral connections is clearly depicted in cases the small object instances in the far distance are missed by FCN to be segmented successfully. Additionally, SSN is capable of predicting the shape of

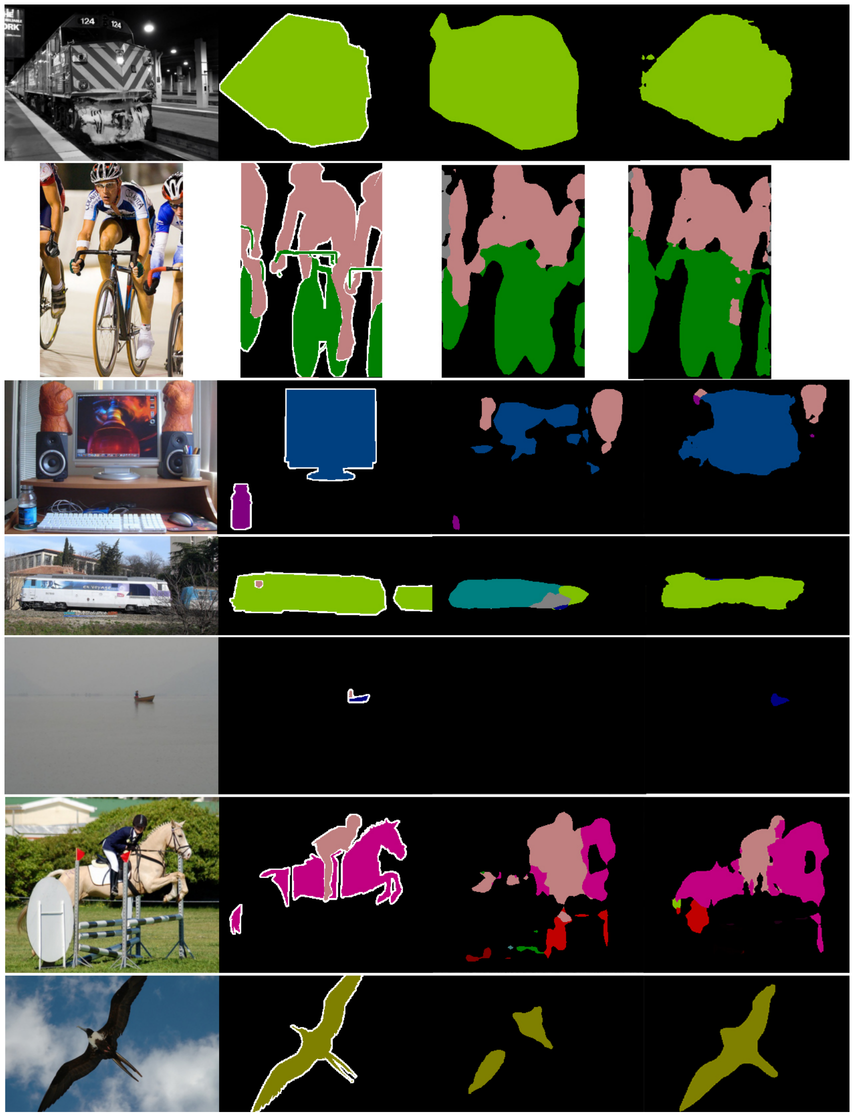


Figure 5.7: Comparison of the segmentation predictions of SSN with FCN on Pascal dataset. From left to right: RGB images, ground-truth, FCN predictions, SSN predictions.

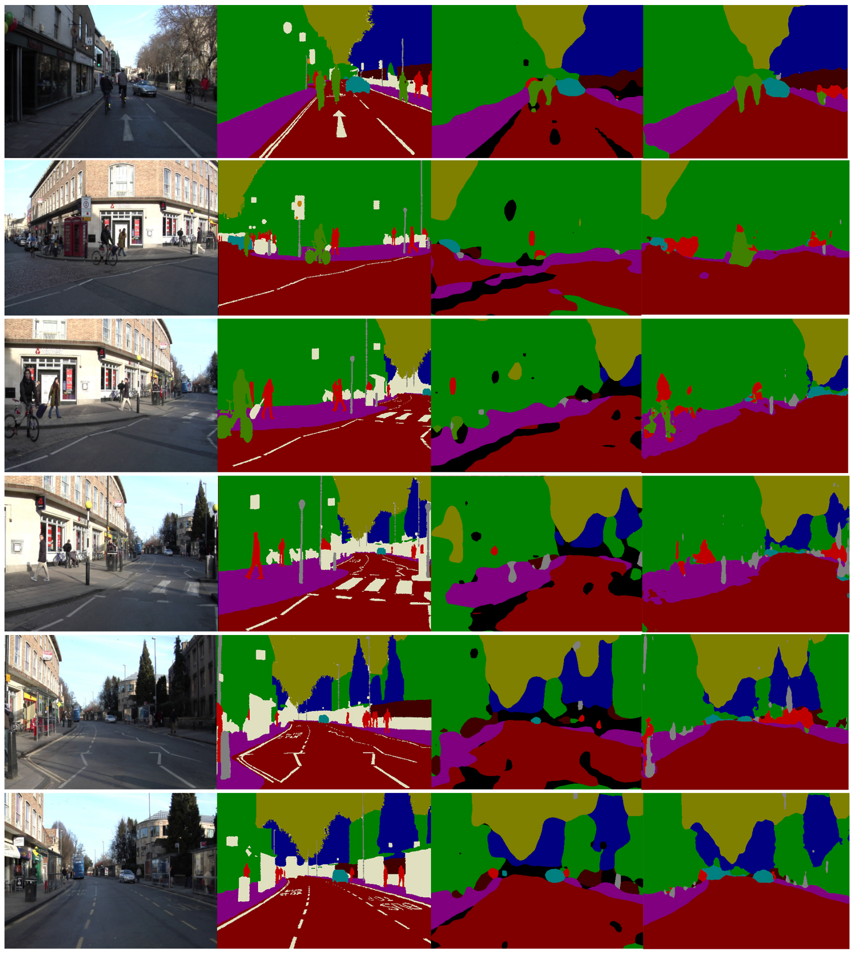


Figure 5.8: Comparison of the segmentation predictions of SSN with FCN on CamVid dataset. From left to right: RGB images, ground-truth, FCN predictions, SSN predictions.

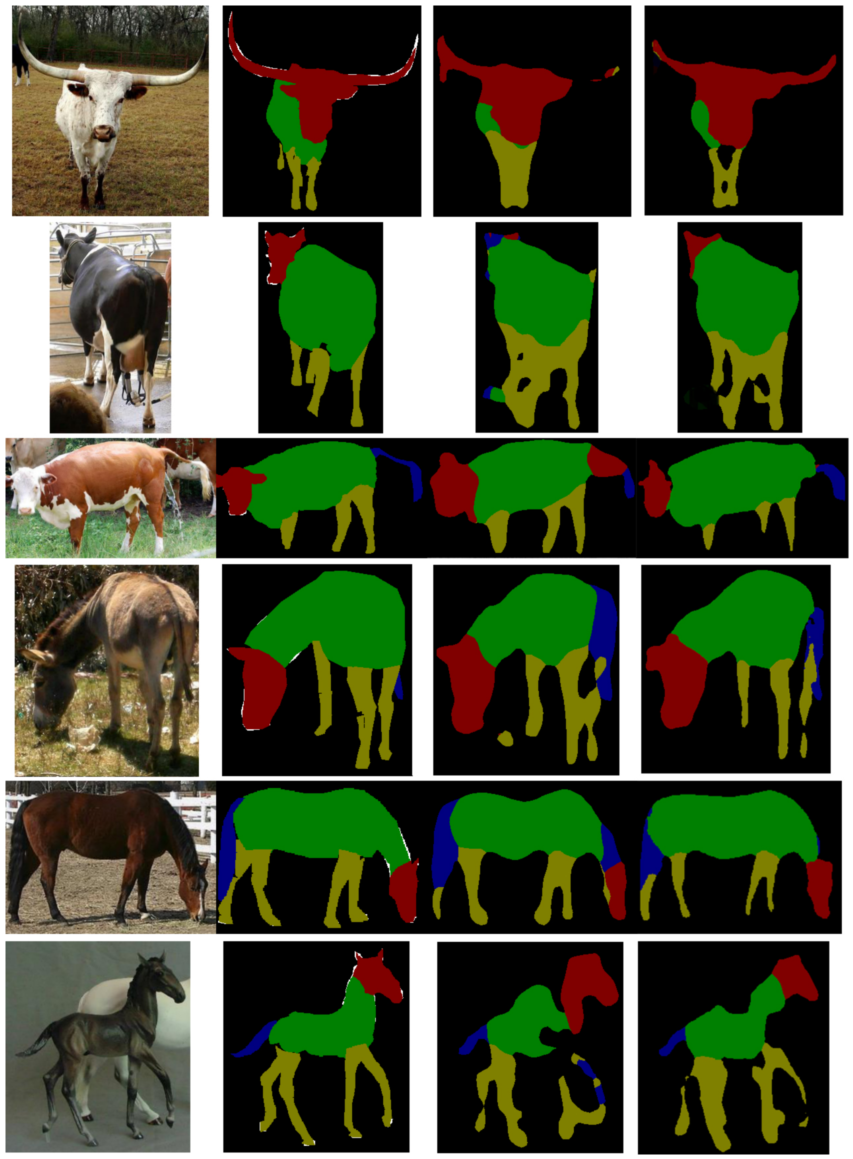


Figure 5.9: Comparison of the segmentation predictions of SSN with FCN on Horse-Cow part parsing dataset. From left to right: RGB images, ground-truth, FCN predictions, SSN predictions.

the segment masks and filling in the large regions in comparison to the FCN results. These aspects reveals the role of the TD selection in SSN to route relevant information from the BU pathway into the segmentation network.

5.5.2 Ablation Studies

We conduct further experiments to highlight aspects of SSN and emphasize the critical role of the TD selection. First, we show the performance deteriorates if either of the TD and BU inputs are blocked from feeding into the segmentation network. In the upper part of Table 5.7, performance drops from 42.1 for SSN with the segmentation network benefiting from the two inputs (BU and TD activities) to 40.2 for only BU hidden inputs and to 39.7 for only the TD gating inputs. Both inputs have complementary roles for the segmentation performance of SSN. The BU input benefits from the parametric distributed representation while the TD input has a predictive selective characteristic. Once these two jointly are in place, the segmentation performance of SSN is superior to the either once individually used.

Additionally, we emphasize the role of the number of levels of processing in the segmentation network on the SSN performance in the lower part of Table 5.7. The segmentation pipeline with one level has the least performance accuracy while as the number of levels increases, the segmentation performance improves. This finding underlines that SSN benefits from the selectivity of the TD mechanisms on the high-level to the intermediate layer representations of the BU network for accurate object segmentation. The intermediate layers have fine details while the top layers have coarse structures. The relevance of

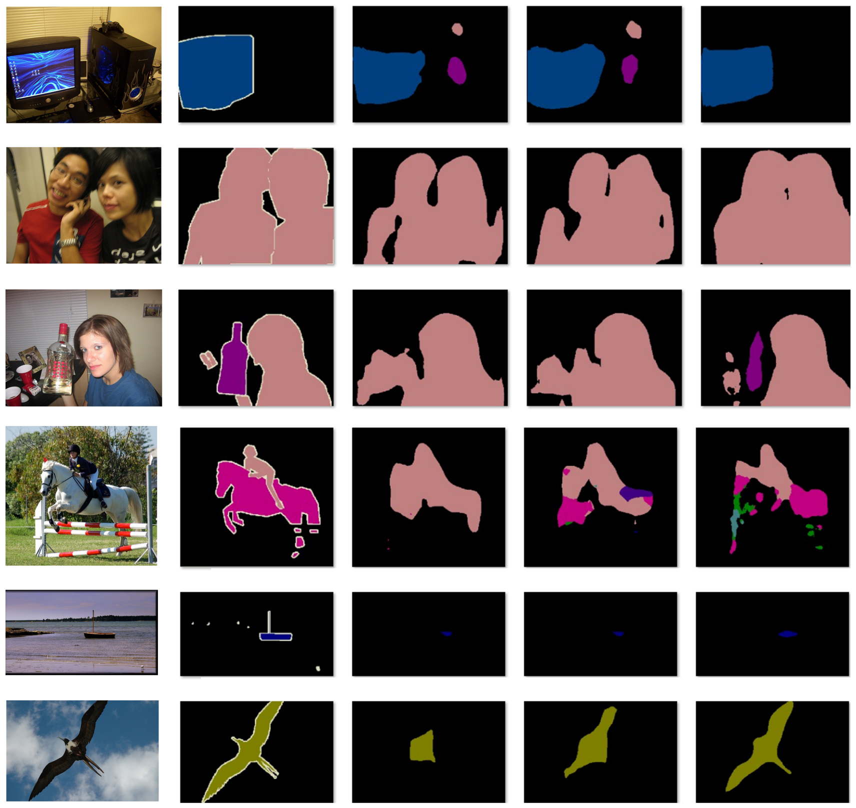


Figure 5.10: Demonstrating the role of the number of levels of TD and BU modulation on the segmentation prediction. From left to right: RGB images, ground-truth, SSN with 1 level of modulation, SSN with 2 levels of modulation, and SSN with 3 levels of modulation respectively.

Model	mean Accuracy	mean IoU
SSN	54.1	42.1
SSN-BU	51.3	40.2
SSN-TD	49.4	39.7
SSN-1	51.5	40.7
SSN-2	52.9	41.6
SSN-3	54.1	42.1

Table 5.7: Ablation Studies on the TD modulatory role, the error signal propagation, number of gating layers into the segmentation pipeline using AlexNet on the Pascal VOC 2012 validation set.

the selectivity of the lower layer hidden activities imposed by the TD gating activities is supported by the results in this experiment. This is in line with the hypothesis that the TD selection process is capable of activating units in the spatial and channel dimensions for the modulation of the BU features for the dense pixel-level labeling task of semantic segmentation. In Fig. 5.10, it is demonstrated that as the number of levels of modulation increases, the predictions become more accurate. False positive predictions are corrected and the shape of segmentation regions becomes more accurate. For instance, in the bird example, the shape of the segmentation for the bird becomes close the the ground truth once we have 3 levels of modulation in SSN. The qualitative results additionally highlight the modulatory role of the TD selection on the segmentation performance results.

5.5.3 Noise Interference Robustness

We test the robustness of SSN against two types of visual confusion: noise interference and partial occlusion. We conduct experiments to study the effects of the additive uniform noise, salt-and-pepper noise, and box occlusion

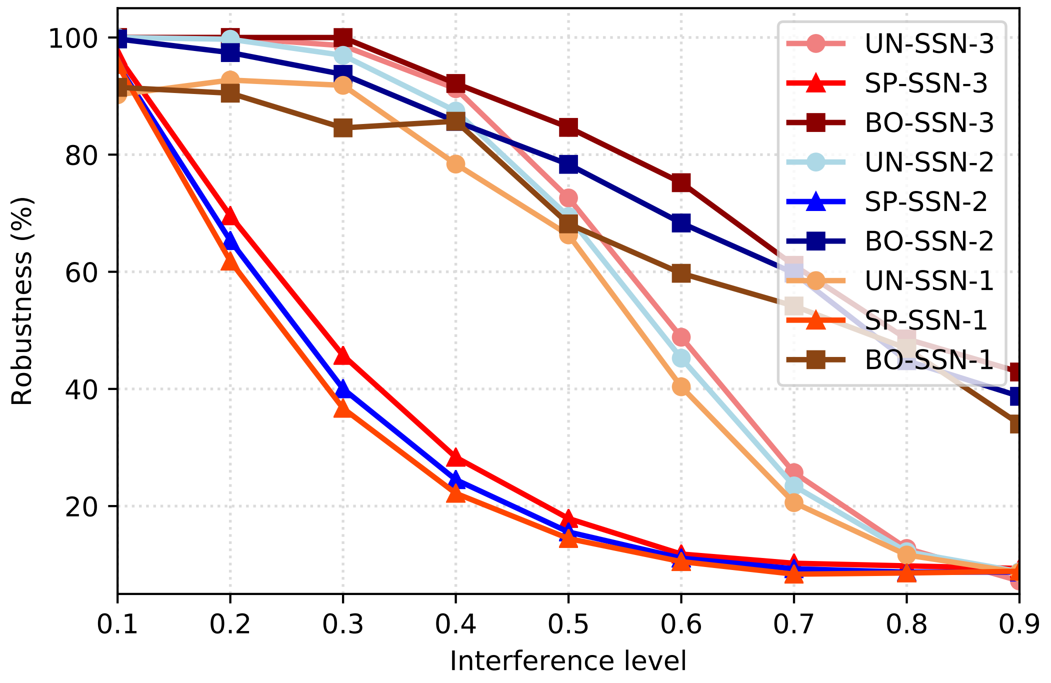


Figure 5.11: Robustness of SSN is measured at different interference levels (σ) for the uniform (UN), salt-pepper (SP), and box occlusion (BO) types. σ determines the bandwidth of the uniform noise ($255 \times \sigma$), the probability of having salt-pepper noise at a location, and the length of the occlusion box ($\sigma \times min(h_{image}, w_{image})$) respectively. The modulatory role of the depth of the TD selection is demonstrated for SSN-k with inputs at k number of levels into the segmentation pipeline.

on SSN performance. Fig. 5.11 illustrates the role of the gating mechanisms derived by TD selection at the early layers to gain increased level of robustness on perturbed data samples. SSN with three levels of attentive segmentation obtains the highest degree of robustness compared with smaller number of levels. This is consistent across not only the two types of additive noise interference, but also partial box occlusion. Figures 5.12, 5.13, and 5.14 illustrates qualitatively how three levels of noise degrades the segmentation performance of SSN. The figures present the experimental setups that we used to study the

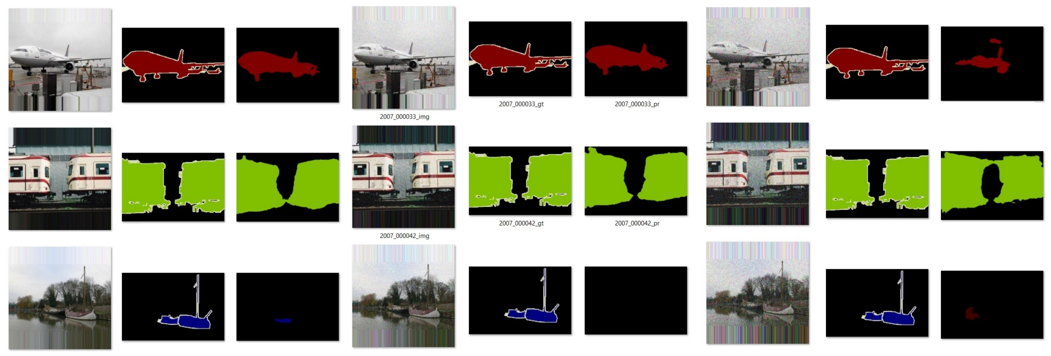


Figure 5.12: Three different levels of uniform noise is added to the RGB images. From left to right the noise level is 0.25, 0.45, 0.65 respectively.

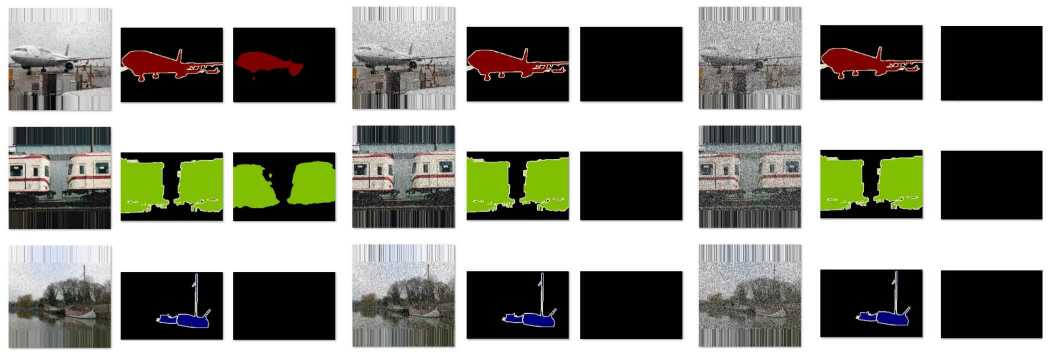


Figure 5.13: Three different levels of salt-pepper noise is added to the RGB images. From left to right the noise level is 0.25, 0.45, 0.65 respectively.

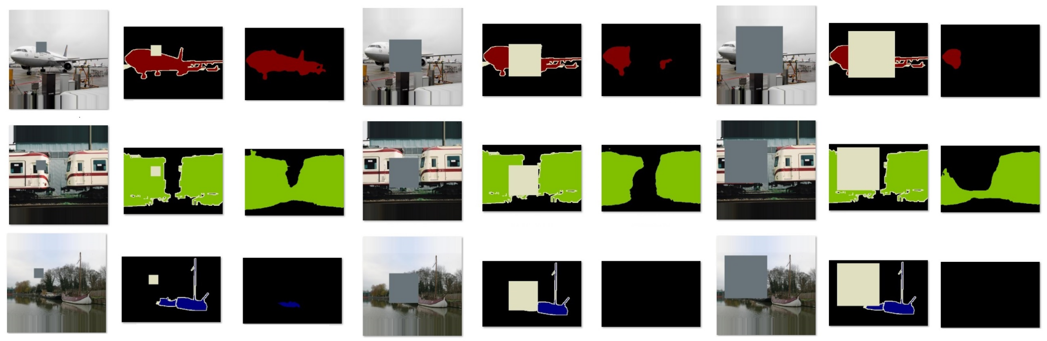


Figure 5.14: Three different levels of box-occlusion noise is added to the RGB images. From left to right the noise level is 0.25, 0.45, 0.65 respectively.

noise interference robustness of SSN with 3 different modulation levels.

5.6 Conclusion

The Top-Down selective attention is a well-known processing component of the human vision system. We introduce a unified Bottom-Up and Top-Down framework that not only benefits from a feedforward representation but also a backward selective modulation mechanism for the task of object segmentation in this chapter. We define a parametric semantic controller to predict for the activation of TD mechanisms at the top of the visual hierarchy. We demonstrate how the TD gating activities modulate the BU activities for object segmentation through different stages of the information processing. The experimental evaluation results supports the role of the TD selection to improve the baseline performance results.

Chapter 6

Attention for Compact Neural Representation

6.1 Abstract

Deep neural networks have evolved to become power demanding and consequently difficult to apply to small-size mobile platforms. Network parameter reduction methods have been introduced to systematically deal with the computational and memory complexity of deep networks. We propose to examine the ability of attentive connection pruning to deal with redundancy reduction in neural networks as a contribution to the reduction of computational demand. In this chapter, we describe a Top-Down attention mechanism that is added to a Bottom-Up feedforward network to select important connections and subsequently prune redundant ones at all parametric layers. Our method not only introduces a novel hierarchical selection mechanism as the

basis of pruning but also remains competitive with previous ad hoc methods in experimental evaluation. We conduct experiments using different network architectures on popular benchmark datasets to show high compression rate is achievable with negligible loss of accuracy.

6.2 Introduction

The human brain receives a tremendously large amount of raw sensory data at every second. How the brain deals efficiently and accurately with the amount of input data to accomplish short- and long-range tasks is the target of various research studies. [77, 21] analyze the computational complexity of visual tasks and suggest the brain employs approximation solutions to overcome some of the difficulties presented due to the vast amount of input sensory data.

Neural networks have been successful on various computational tasks in vision, language, and speech processing. Such networks are defined using a large number of parameters arranged in multiple layers of computation. Despite achieving good performance on benchmark datasets, parametric redundancy is known to be widespread, hence not suitable for real-time mobile applications. Tensor processing, memory usage, and power consumption of mobile devices are limited and consequently neural networks must be accelerated and compressed for such mobile applications [167]. Model compression reduces the number of parameters and primitive operations and consequently improve the computation speed at inference phase [167].

Moreover, due to the over-fitting phenomenon, over-parameterized models suffer from low generalization and therefore must be regularized. Such models learn dataset biases very quickly, memorize data distribution, and consequently lack proper generalization. One way to regularize parametric models is by imposing sparsity-imposing terms and consequently pruning a number of parameters to zero and keeping a sparse subset of them [168, 169].

Neural network compression is defined as a systematic attempt to reduce parametric redundancies in dense and multi-layer networks while maintaining generalization performance with the least accuracy drop. Parametric redundancy in such networks is empirically investigated in [170]. Various neural network compression approaches such as weight clustering [171, 172, 173], low-rank approximation [170, 174], weight pruning [175, 176, 177, 171, 178, 179, 180], and sparsity via regularization [181, 169, 182] are introduced to reduce parameter redundancy for lower computational and memory complexity. Pruning methods have shown to be computationally favorable while relying on straightforward heuristics and ad hoc approaches to schedule and devise pruning patterns. These compression approaches rely on defining some measure of importance based on which a significant subset of weight parameters are kept and the rest are pruned permanently.

STNet [108] introduces a selective attention approach in convolutional neural networks for the task of object localization. STNet leverages a small portion of the entire visual hierarchy to route through all parametric layers to localize the most important regions of the input images for a top label category. The selection process is hierarchical and provides a reliable source of weight pruning. The experimental results of STNet for object localization reveal that a sparse subset of the network units and weight parameters are sufficient for a successful localization result. We propose a novel attentive pruning method based on STNet to achieve compact neural representation using Top-Down selection mechanisms. Following [108], we define a neural network to benefit from two passes of information processing, Bottom-Up (BU) and Top-Down (TD). The BU pass is data-driven. It begins from raw input data, goes through multiple layers of feature transformation, and finally predicts abstract task-dependent outputs. On the other hand, the TD pass is initialized from high level attention signals, goes through selection layers, and outputs kernel importance activities. The importance activities are computed by three variable inputs in the TD selection pass: one is the hidden responses, the other is the kernel filters, the last is the top attention signals. We show that all of the three sources of TD selection are crucial for strong network pruning.

Attentive pruning relies on kernel importance activities to decide on pruning patterns. We feed neural networks with input data, and then activate TD selection to output kernel importance activities at every layer. These activities are accumulated and scheduled to generate pruning patterns. We evaluate the attentive pruning method using various network architectures on benchmark datasets. The competitive evaluation results reveal the selective nature of the TD mechanisms over the kernel filters. This complements the importance of the gating activities for visual tasks such as object localization and segmentation.

6.3 Related Work

Low-rank approximation, weight pruning, imposing sparsity through regularizers, and weight clustering are various approaches to reduce the number of learnable parameters in convolutional neural networks and other training-based systems. [170, 174] seek to find low-rank approximation of the weight kernel tensors by singular value decomposition. They achieve minimal performance degradation on large networks. These techniques are also computationally expensive and slow. [171, 172, 173] propose to cluster weights into a smaller number of groups that are representative of the original network. [181, 169, 182] investigate the role of the sparsity-induced regularization terms such as L1-norm on the loss function.

Weight pruning methods have been investigated from the early days of neural networks [183]. [179, 180] propose to prune the network connections based on the second-order derivatives of the loss function. The high computational complexity of computing the Hessian matrix on larger networks is a shortcoming of such methods. On the other hand, magnitude-based pruning [175, 176, 178] is fast, simple, and competitive with the baseline approaches. [177] recently proposes to compute the second derivatives of a reconstruction error minimization loss at each layer based on which the pruning is performed. All of these pruning methods use a retraining phase once a number of weights are set inactive for the subsequent steps. The number of pruning phases and the number of retraining iterations after a pruning phase to regain the same level of performance generalization are two important aspects of these methods. We study the role of the selective attention introduced in STNet [108] for the task of weight pruning in neural networks. The selective attention mechanism defines stages of local competitions based on which a selection pattern over the network connections is generated at each layer. We propose the algorithm using which these patterns be harnessed for the determination of the weight pruning. The experimental evaluation reveals a competitive compression ratio and error rate after retraining with the baseline methods.

6.4 Attention Drives Weight Pruning

We define neural networks with Bottom-Up (BU) and Top-Down (TD) information passes. The former transforms input data into high-level semantic information. On the other hand, the TD pass begins from class predictions and computes the kernel importance responses at each layer. The TD selection process relies on three main sources of information. We propose to compute the important connections that the TD attentional traces use to route through the visual hierarchy. In this work, we introduce a novel approach such that the pruning mechanism relies on the accumulated kernel importance responses while the baseline models solely consider kernel filters of the feedforward pass. The kernel importance responses are computed using the local competitions that receives three variable inputs: the kernel weights, the hidden activities, and the gating activities. The kernel weights are learned in the pre-training phase. The hidden activities represent the hierarchical feature representation of the underlying layers for some specific input data. Therefore, the calculated kernel importance responses take into consideration not only the kernel

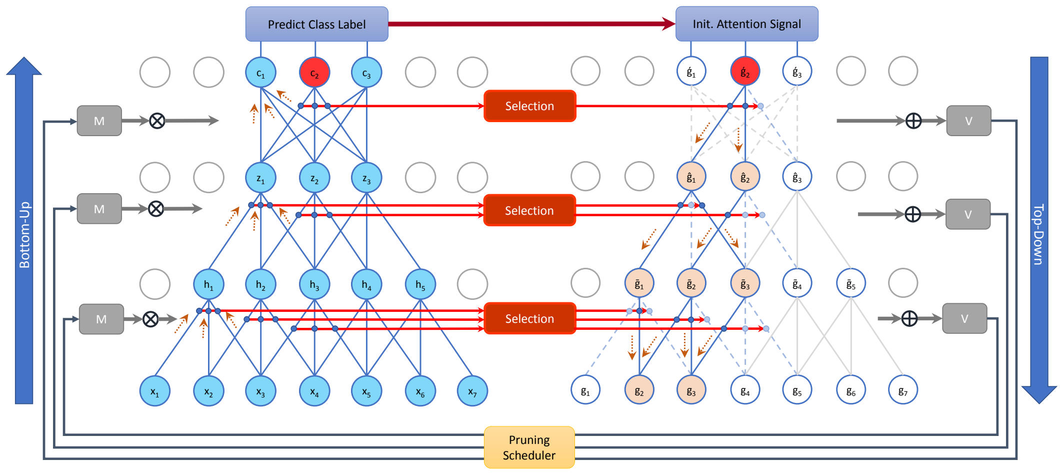


Figure 6.1: Schematic illustration of the proposed method for connection pruning that leads to the reduction of the number of network parameters. On the left side, a toy multi-layer feedforward network is shown. On the right, the corresponding TD networks is given. At each layer, once the active connections $\tilde{w}$ are computed using the TD selection mechanisms, they are additively accumulated into the persistent buffer V; subsequently, the mask tensor M is scheduled to get updated after a number of iterations. The feedforward pass is always additively modulated with the mask tensors M. The arrows show the direction of information flow.

weights but also the input hidden activities. The baseline pruning methods only rely on the magnitude thresholding while the proposed method generalizes the baselines to include the hidden activities. Furthermore, the kernel importance responses are category-specific. The important subset of weight parameters are determined not only for some specific input but also some particular label category. The TD selection pass starts from a category initialization signal. Consequently, all the TD selection mechanisms are informative of that particular category initialization. The category-specific nature of the attentive pruning method further reduces the non-relevant pruning and therefore, speeds up the convergence in the retraining phase.

6.4.1 Method Overview

Figure 6.1 demonstrates schematically the information flow at different computational stages of the proposed method. First, given some input x at the bottom of the visual hierarchy shown on the left part of Fig. 6.1, the feature extraction is done using the parametric layers and the output hidden activities h, z, c are computed at each layer until the top score layer is reached and the BU pass ends. The Predict Class Label block is a multi-class transfer function such that $softmax()$ outputs the class probability prediction given the input data. Then, the attention signal initialization determines the label category for which the TD pass (shown on the right side) will be activated. Once the attention signal is set, the selection mechanism within the local receptive field of the initialized category node is activated. According to the competition result, a number of important outgoing connections on the TD layer are activated. Then, the gating activity of the category node proportional to the activated connection weights propagates downward to the next gating layer. This is illustrated by the outgoing solid (activated) and dashed (deactivated) connections from $\acute{g}$ to $\hat{g}$ in Fig. 6.1. At this stage, the kernel importance responses V for the top layer are updated with the activated connection patterns in an additive manner. This layer-wise computation continues at all of the subsequent lower layers until the TD selection pass ends at the input layer and returns the gating activities g. The kernel importance accumulation is iterated for a number of randomly-chosen input samples until a pruning phase is set by the scheduling strategy depicted by the yellow Pruning Scheduler module at the bottom of the figure. The pruning mask M, then, is updated based on the so-far-accumulated kernel importance responses. The pruning masks are initially set to one, meaning no kernel weight is pruned before the first scheduled mask update. Over different pruning phases, they start to gradually become zero.

6.4.2 Notations

A multi-layer neural network $f : \mathbb{R}^{H\times W} \to \mathbb{R}$ is at the core of the BU pass. The training set $D = \{(x_i, y_i)\}_{i=1}^{N}$ has N samples such that the input data is $x \in \mathbb{R}^{H\times W}$an input image with height H and width W and $y \in \{0, 1, \dots, K-1\}$ is the ground truth label for K different classes. We define the BU pass as a feedforward network

$$h = f(x; W), \tag{6.1}$$

in which $f = \{f_j\}_{j=0}^{L}$ is a network with L layers, x and h are the input and output of the network, and $W = \{W_j\}_{j=0}^{L}$ is the set of network parameters at L layers. We define the feature transformation $h_l = f(h_{l-1}; W_l)$ at layer l such that $f(x; W) = W^T x$ is a linear transformation $f : \mathbb{R}^M \to \mathbb{R}^N$ of the input x by the weight matrix $W \in \mathbb{R}^{M\times N}$ for fully-connected layers. The convolutional layers apply convolutions using the kernel filter W.

The BU network output h is fed into a Softmax transfer function $\hat{y} = softmax(h)$ to compute the multinomial probability values $\hat{y}$. The cross-entropy loss function is used with the Stochastic Gradient Descent (SGD) optimization algorithm to update network parameters.

6.4.3 Top-Down Processing

The role of the Top-Down (TD) pass is to traverse downward into the visual hierarchy by routing through the most significant weight connections of the network. TD pass begins from an initialization signal $d \in \mathbb{R}^K$ generated according to the ground truth label y. It traverses down layer by layer by selecting through network connections

$$g = t(d, h, W), \tag{6.2}$$

in which $h = \{h_i\}_{i=0}^{L}$ is the set of BU hidden activities, network parameters $W = \{W_i\}_{i=0}^{L}$, and $g = \{g_i\}_{i=0}^{L}$ is the set of kernel importance responses at L layers. The attention signal is initialized based on the ground truth label of the input image. It sets the signal unit for the category label corresponding to the ground truth to one and keep the rest zero

$$d = \{d_{j=y} = 1, d_{j \neq y} = 0, \}.$$

The TD pass at each layer computes the importance responses using three computational stages defined in STNet [108]. the first stage takes care of noise interference reduction, the second one perform grouping and selection, and the last stage normalizes and propagates the gating activities to the next layer.

6.4.4 Kernel Importance Maps

The output hidden activities at a layer are computed by a linear multiplication of the kernel weight matrix and the input hidden activities $h = W^T x$ for fully-connected layers. The extension to convolutional layers is straightforward using convolution operations and are ignored for the sake of brevity. Each output unit h_i receives a weighted sum of the input vector x according to the weight parameter vector w_i .

The TD selection mechanism $a^{l-1} = t(g_j^l, x^l, w_j^l)$ for the output unit j operates on the input hidden activities x^l, the weight parameters w_j^l connecting all the input units to unit j, and the input gating unit g_j^l. The selection mechanism t is only executed for non-zero g^l units. The output of the selection mechanism contains two entities $a^{l-1} = \{g^{l-1}, \tilde{w}^l\}$, where g^{l-1} is the output gating activities which is the source of TD selection at the layer below, and $\tilde{w}^l$ is the kernel importance responses at layer l. Hereafter, we drop l for sake of notation brevity. The kernel importance responses are accumulated for all N samples in the training set and used in selective pruning for network compression.

We categorize all the previous pruning approaches as class-agnostic pruning methods since they determine the connections to prune regardless of the target categories they are interested in. Our proposed attentive pruning method, however, is class-specific since the TD pass begins from some class hypothesis signal and routes through the network hierarchy accordingly. Therefore, the computed kernel responses are representative of the subset of the network connections that are most important for the true category label predictions.

Additionally, network parameters are trained according to the input data distribution. The BU information flows into the network hierarchy by measuring the numeric relation between an input unit x_i and a connection weight w_{ji} that connects the input unit i to the output unit j. If both the input and the weight have high activity, the output will have high value too. Being motivated by this insight, we show that the TD selection process produces kernel importance maps by considering both of the input and the kernel weights. Kernel importance is measured based on whether the input units and the kernel weights are both positively related. This generalizes the previous works in which the kernel weights are individually considered for connection pruning.

6.4.5 Attentive Pruning

We define an attentive pruning method using the kernel importance responses $\tilde{W}$. The importance responses $\tilde{W}^t$ at iteration t is accumulated into a persistent buffer $V^t = V^{t-1} + \tilde{W}^{t-1}$. The binary pruning mask m defines the pattern using which the kernel weights are permanently pruned. The function r determines the pruning mask m. r is a thresholding function that sets the binary values of m:

$$r(u;a) = \begin{cases} 0 & u \leq a \\ 1 & a < u \end{cases}, \tag{6.3}$$

where $a = m(u) + \lambda\,\sigma(u)$. λ is a multiplicative factor, $m(u)$ is the mean, and $\sigma(u)$ is the standard deviation of the input u. We set the binary mask m_l at layer l by setting $u = V_l^t$ and $m_l = r(u;a)$. We run the BU and TD passes for a number of iterations after which the attentive pruning starts to compress the network parameters. Once the set of mask binary tensors $m = \{m_i\}_{i=0}^{l}$ are determined after each pruning phase, the feedforward BU pass is modified using the binary pattern in the mask tensors:

$$h = m \odot W^T x, \tag{6.4}$$

where $a \odot b$ is the element-wise (Hadamard) product of a with b.

Fig. 6.2 illustrates the BU and TD interactions in detail. At the layer i for instance, the TD selection mechanism receives the three inputs: the hidden activities h_{i-1}, the kernel weights w_i, and the gating activities g_i. Once the selection is completed, the kernel importance maps $\tilde{w}$ are set for the downward gating activity propagation. Additionally, they are used to additively update the persistent buffer V_i. The pruning pattern of the kernel weights w_i is determined according to the binary pruning mask M_i. The mask is updated according to the pruning scheduler unit. Once the scheduler set the updating on, the thresholding function r updates the mask binary elements given the input persistent tensor. This procedure is applied to every layer the pruning is defined to be applied.

6.4.6 Retraining Strategy

At every iteration, using the samples in the mini-batch, we have sequentially the following computational stages: a feedforward BU pass, attention signal initialization, a TD pass, and an updating of the persistent buffer V using

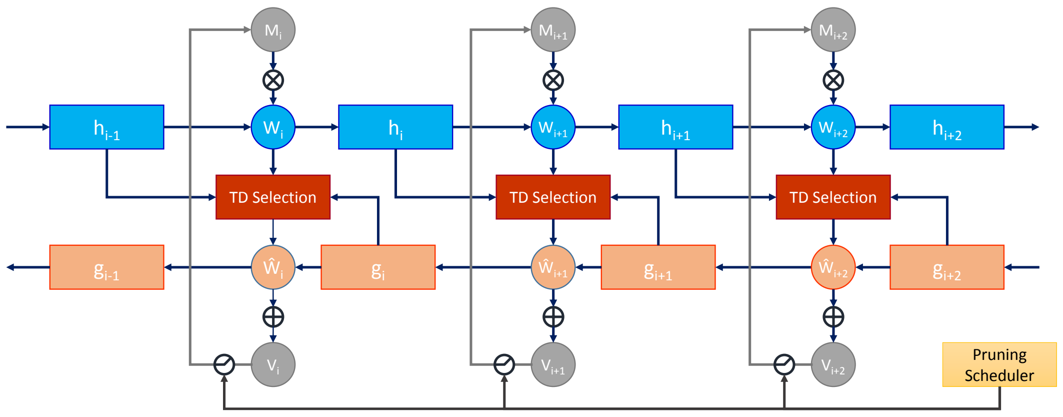


Figure 6.2: Detailed demonstration of different stages of computation of the BU and TD passes for selective connection pruning. At each layer, the inputs to the TD selection unit, the active connections $\tilde{w}$, the additive accumulation into the persistent buffer, and the multiplicative mask of the BU kernel weight are depicted.

the kernel importance responses $\tilde{W}$. After a number of initial iterations to accumulate kernel importance responses into the persistent buffer, we start pruning the network connections for several times. The network is retrained from the first occurrence of pruning onward. This helps the network retain its level of accuracy for label prediction over multiple stage of connection pruning. Retraining is inevitable due to the high pruning rate of the network weight parameters. The network needs some iterations to shift its representational capability for a high level of label prediction accuracy. The retraining allows the adaptation to the reduced parameter space. It follows the exact optimization settings used for the pre-training of the network prior to the network compression.

Model	Top-1 error	Parameters	Compression
LeNet-300-100-reference	3.3%	267K	-
LeNet-300-100-pruned	3.8%	5.2K	58×
LeNet-5-reference	2.1%	83K	-
LeNet-5-pruned	3.2%	4.7K	102×

Table 6.1: LeNet error rate and compression ratio on MNIST dataset using the attentive connection pruning.

6.5 Experimental Results

In this section, we conduct experiments to evaluate the compression ratio of the attentive pruning method. The compression ratio is defined as the ratio of the total number of mask units over the total number of the non-zero mask units (active connections). We use the Pytorch deep learning framework [1] [162] to implement the model for the experiments of this work. The layers of the TD pass are implemented using the code provided by [2] STNet [108]. We choose the learning rate 10^{-3}, momentum 0.9, weight decay 0.0005, and mini-batch size 64 for the SGD optimizer unless otherwise mentioned. We follow the network pruning protocol and experimental setup established in [175, 176, 177] in this work to evaluate the compression performance of the proposed approach. The goal is to achieve a high compression ratio while maintaining classification generalization performance with negligible performance compromise. Harsh pruning of network connections mostly degrades the network capability to recover high level of category label prediction and this results into a collapsed network with an improper representation capacity. We monitor the performance of the networks throughout the experimental

[1]https://pytorch.org/

[2]https://github.com/mbiparva/stnet-object-localization

Method	Network	Dataset	Error-degradation	Compression Ratio
Han et al. [175]	LeNet-300-100	MNIST	0.19%	12×
Guo et al. [176]	LeNet-300-100	MNIST	0.23%	56×
Dong et al. [177]	LeNet-300-100	MNIST	0.20%	66×
Ours	LeNet-300-100	MNIST	0.50%	58×
Han et al. [175]	LeNet-5	MNIST	0.09%	12×
Guo et al. [176]	LeNet-5	MNIST	0.09%	108×
Dong et al. [177]	LeNet-5	MNIST	0.39%	111×
Ours	LeNet-5	MNIST	0.90%	102×

Table 6.2: Comparison of the Compression ratio of the proposed method with the baseline approaches using LeNet-300-100 and LeNet-5 network architectures on MNIST. Error degradation is the difference between the original error and the error at the end of the retraining phase.

evaluation on the held-out validation set and report the compression ratio and performance after pruning on the test set.

6.5.1 The MNIST Dataset

One of the popular datasets widely used in the machine learning community to experimentally evaluate novel methods is MNIST dataset. It contains gray-scale images of handwritten digits and is used for category classification.

We define the BU network for the MNIST dataset according to two classic network architectures: LeNet-300-100 [29] and LeNet-5 [29]. The former consists of three fully-connected layers with output channel sizes of 300, 100, 10 successively and contains 267K learnable parameters. We train it for 10 epochs to obtain the reference model for the BU network. Lenet-5, on the other hand, has two convolutional layers at the beginning. Similarly, it is trained for 10 epochs. It has 431K learnable parameters.

After the first epoch that the persistent buffers are accumulated, we start pruning the network connections for the next 7 consecutive epochs. The multiplicative factor λ is set to the following values $[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0]$. We continue retraining the network for 25 epochs after the final pruning stage. We follow this pruning protocol for both of the LeNet architectures. Error rate and the compression ratio of the networks are given in Table 6.1. The results confirm the selectivity nature of the TD mechanism in the parameter space of the BU network. According to the experiment results, the kernel importance responses are shown a reliable source of connection pruning using LeNet architectures on MNIST. The proposed model is capable of reducing the number of kernel weights 58 and 102 times for LeNet300 and LeNet-5 respectively for negligible performance accuracy drops.

Comparison with the state-of-the-art: We compare the compression performance of the proposed attentive pruning mechanism with the baseline approaches on MNIST in Table 6.2. The experimental results reveal that the proposed approach outperforms two of the baseline approaches [175, 176] using the LeNet architectures while remain competitive with [177]. It should be noted that [177] uses the computationally expensive second order derivatives of a layer-wise error function to derive the pruning policy while we only rely on the important kernel responses derived from the TD selection mechanisms. [177] exhaustively relies on second-order derivatives at each layer while we chose to determine kernel responses in a hierarchical manner. However, the proposed method can outperform [175, 176] that use magnitude pruning of weight parameters. This supports the role of the TD selection mechanisms to determine the most important parameters of neural networks as the source of a pruning procedure.

6.5.2 The CIFAR Dataset

CIFAR-10 dataset [184] contains RGB images of the same size and scale as MNIST dataset. The dataset consists of natural images of 10 semantic categories for object classification. In comparison with MNIST, the goal is to benchmark classifier performance on a higher level of complexity using CIFAR-10. We evaluate the performance of the proposed method using three network architectures on this dataset: LeNet-5, CifarNet and AlexNet. CifarNet[3] [184] is a multi-layer network with three convolutional layer and two fully-connected layers. It has larger number of parameters than LeNet-5. AlexNet [30] has 5 convolutional and 2 fully-connected layers.

We empirically choose a slightly different pruning and re-training policy for the CIFAR-10 dataset since it has a lot more complexity and care must be taken for connection pruning. First, we change the mini-batch size to 16. The multiplicative factor λ is set only to 0.5. However, unlike the MNIST pruning protocol, we prune layers individually. We observed in the preliminary experiments that this approach helps maintain the label prediction accuracy with the minimal performance compromise while keep the compression ratio high. This policy helps the network to maintain its representation capability for the classification task and avoid deteriorating learning collapses. We first accumulate the kernel importance responses in the persistent buffer for one epoch. Next, for every 4 epochs, we prune the connections of one layer starting from the first parametric layer at the bottom to the last one at the top of the BU network. Once the pruning of the last layer is done, we continue re-training

[3]https://code.google.com/archive/p/cuda-convnet/

Model	Top-1 error	Parameters	Compression
Lenet-5-reference	38.2%	83K	-
Lenet-5-pruned	39.4%	8.3K	10×
CifarNet-reference	30.4%	84K	-
CifarNet-pruned	31.1%	7.6K	11×
AlexNet-reference	23.5%	390K	-
AlexNet-pruned	24.8%	13K	29×

Table 6.3: LeNet and CifarNet error rate and compression ratio on CIFAR-10 dataset using the attentive connection pruning.

of the pruned network for 40 epochs and then report the compression ratio in Table 6.3. For all of the three networks, the attentive pruning method is able to maintain the reference network error rate and achieve high compression ratio.

6.6 Conclusion

We propose a novel pruning method to reduce the number of parameters of multi-layer networks. The attentive pruning method relies not only a feedforward feature representation pass but also a selective top-down pass. The TD pass computes the most important parameters of the kernel filters according to a selected category label. Additionally, the hidden activities at each layer participate in the stages of the TD selection mechanism. This ensures both the top semantic information and input data representation play roles in the stages of kernel importance computation. We evaluate the compression ratio of the proposed method on two classification datasets and show the improvement on three popular network architectures. The network achieves a high compression ratio with minimal compromise of generalization performance.

Chapter 7

Contextual Interference Reduction

7.1 Abstract

The issue of the contextual interference with the foreground target objects is one of the main shortcomings of the hierarchical feature representations such as convolutional neural networks. Due to the dense hierarchical parametrization of convolutional neural networks and the utilization of convolution and subsampling layers in the feedforward manner, foreground and background representations are inevitably mixed up and visual confusion is eminent. Tsotsos et al. [21] refers to this as the Crosstalk issue in neural networks. Feedforward neural networks trained for object classification have shown successful application of localization through Top-Down mechanisms. Despite the success of localizing objects in the cluttered natural images using such feedforward net-

works, the context has still significant role in the final label prediction [185, 108, 19]. Additionally, research studies have revealed evidence on widespread visual confusion on convolutional neural networks [16, 185, 17, 19, 186]. A systematic approach to shift learned neural representations from the emphasis on the contextual regions to the foreground target objects can help achieve a higher degree of representation disentanglement. We propose a selective fine-tuning approach for neural networks using a unified bottom-up and top-down framework. A gating mechanism of hidden activities imposed by Top-Down selection mechanisms is defined in the iterative feedforward pass. An attention-augmented loss function is introduced during which the network parameters are fine-tuned for a number of iterations. The fine-tuning using the iterative pass helps the network to reduce the reliance on the contextual representation throughout the visual hierarchy. Therefore, the label prediction relies more on the target object representation and consequently achieve a higher degree of robustness to the background changes. The experimental evaluations on a modified MNIST dataset reveals not only that the results are improved but also a higher degree of robustness to the background perturbation using additive noise is obtained.

7.2 Introduction

Iterative feedforward and feedback processes are recognized to play important roles in the information processing of the human brain [12, 187]. To this end, the Selective Tuning model [7, 21, 188] defines multiple computational stages in artificial dynamical networks such as the preliminary stage of visual task priming, the early stage of bottom-up neuronal encoding, the selective stage of top-down attention, and finally the re-interpretation and iterative bottom-up passes.

Feedforward neural networks currently suffer from different vulnerabilities such as visual confusion [108, 19, 185], and adversarial attacks [16, 17, 186] due to the unconstrained and data-driven nature of the training method in such networks. Semantic objects of unlabeled categories are confusingly mixed up with the representation of labeled categories. [108] demonstrates the cases in which the top-down localization leads to the selection of unlabeled object categories with high co-occurrence to labeled categories. Similar types of visual confusion is reported for object detection in [19]. Human and machine robustness against input distortions is also studied [185]. It is revealed that even in the gist representation provided by the feedforward feature encoding, humans are still competent to deal with input noise distortions while neural networks fall behind. The neural networks, as highly parametric learning machines, are strongly prone to overfitting to the data distribution of the benchmark datasets and consequently achieve low generalization to unseen and distorted data samples. Addition of extra regularization terms to appropriate objective functions [189] and sparsification of gradients [190, 191, 17] are two approaches to improve robustness against visual vulnerabilities and generalization performance.

We suggest that implicit concentration of the learning method potential on target objects can help to reduce the contextual interference in neural networks. Since the spatial extent of objects is gradually lost within the visual hierarchy in neural networks (the Blurring problem defined in [7]), a TD selection mechanism is essential to constraint the focus of the learning method on relevant spatial regions and feature channels. We hypothesize that training a neural network with iterative BU passes driven from TD attentive mechanisms will achieve a more robust representation and improve the localization and categorization prediction metrics.

STNet [108] introduces a unified framework with BU and TD passes. The framework has shown success for tasks such as object localization, object segmentation, and compact neural representation in chapters 3, 5, and 6 respectively. Building on top of this two-pass framework, we propose a novel iterative framework that benefits from selection patterns generated in the TD pass for the modulation of the feature extraction layers in the iterative BU pass. We show that using a novel multi-loss objective function, the network learns to concentrate the focus of attention on the relevant aspects for feature representation. This helps the network to escape unreliable local minima in which the localization accuracy is low and the context has been utilized wrongly for label prediction. We demonstrate a notion of overfitting when a network is trained to predict category labels while unable to localize objects accurately using the learned representation. The proposed augmented loss function, derived from the iterative framework, has an implicit regularization impact on the entire learning algorithm. The experimental evaluation reveals that not only the localization but also classification accuracy rates are improved. The ablation studies demonstrate that the proposed model achieves a higher degree of robustness to the contextual perturbation and hence verifies the attentive capability to focus on relevant encoding aspects.

7.3 Selective Attention for Network Fine-Tuning

We define a neural network framework that consists of the Bottom-Up (BU) feature representation and the Top-Down (TD) modulatory selection. The BU network is a regular multi-layer feature extraction model. Having defined a training set $D = \{(x_i, y_i)\}_{i=1}^{N}$ of N number of input image $x \in \mathbb{R}^{H \times W}$ and ground truth category labels $y \in \{0, \ldots, K-1\}$ for K categories, a mini-batch of training samples are fed into the BU network for category label prediction:

$$s = f(x; W), \tag{7.1}$$

where x is the set of input images, W is the set of BU network parameters, and s is the output confidence scores of all classes. After multiple-layers of parametric feature transformation f, the confidence score s is returned to a softmax probability distribution $p = softmax(s)$ for multinomial category label predictions. $f = \{f_i\}_{i=1}^{L}$ is a multi-layer neural network with L layers. It contains the set of feature transformation functions f_i such that $h_i = f_i(h_{i-1}; w_i)$. The hidden activities of the previous layer h_{i-1} is the input and h_i is the output of the layer. It is worth mentioning that $h_0 = x$ and $h_L = s$.

Following STNet [108], the TD pass starts from a top initialization signal and ends at the bottom of the visual hierarchy. It contains a selection mechanism at every layer consisting of 3 stages of computation: 1) noise interference reduction, 2) grouping and selection 3) normalization and propagation. We define the TD network

$$g = u(d, H, W), \tag{7.2}$$

where $u = \{u_i\}_{i=1}^{L}$ is a set of selection layers, $d \in \mathbb{R}^K$ is the initialization signal, and $H = \{h_i\}_{i=1}^{L}$ is the set of the BU hidden activities. $d = \delta_{iy}$ is defined using Kronecker delta. It is a non-zero vector with all elements zero except the one at the ground truth label y. Particularly, at layer l, the selection layer $g_{l-1} = u(g_l, h_{l-1}, w_l)$ gets the gating activities g_l, the hidden activities at the previous layer h_{l-1}, and the kernel filter parameters w_l. It outputs the gating activities g_{l-1} at the end of the selection stages.

We try to shift the visual representation of the BU network to concentrate on the feature channels and spatial regions of the target object in the foreground rather than the context in the background. Using the TD pass initialized from the ground truth category labels, the gating activities at each layer are selective for the subset of features that are significantly important for the category label predictions. During the selective fine-tuning phase, the network learns to focus on the network parameters that are gated by the TD pass.

7.3.1 Iterative Feedforward Pass

Having defined the feedforward BU and the selective TD passes, we define the iterative BU pass using the gating activities computed in the TD pass. For a mini-batch of samples, the BU pass is first activated, the hidden activities are computed, and the output label prediction is returned. Next, the initialization signal is set using the ground truth label, and then the TD pass is triggered to begin. The gating activities are computed layer by layer until the TD pass stops at the input layer. Then, we define the iterative BU pass consisting of L layers similar to the initial feedforward pass such that at the layer i, the gated hidden activities t_i are

$$t_i = \alpha * \tilde{h}_i \odot \tilde{g}_i + \beta * \tilde{h}_i, \tag{7.3}$$

where $a \odot b$ is the Hadamard product of a with b, $\tilde{h}_i$ is the input hidden activities, and $\tilde{g}_i = n(g_i)$ is the normalized gating activities using the function n such that $\tilde{g}$ has a minimum and maximum activities of zero and one respectively. α and β are the multiplicative factors to control the numeric level of the hidden and gating activities respectively. They are set to one unless otherwise mentioned. Having t_i computed, the output hidden activities at layer $i+1$ is computed

$$\tilde{h}_{i+1} = f(t_i; w_{i+1}). \tag{7.4}$$

Using the confidence score output $\tilde{s} = f(x; W)$, the multinomial probability prediction of the iterative pass is $\tilde{p} = softmax(\tilde{s})$. We propose an attention-augmented loss function with two terms $\mathcal{L}_F$ and $\mathcal{L}_S$:

$$\mathcal{L}_T(p, \tilde{p}, y) = \frac{1}{N}\sum_i \mathcal{L}_F(p_i, y_i) + \alpha\frac{1}{N}\sum_i \mathcal{L}_S(\tilde{p}_i, y_i), \tag{7.5}$$

where p and $\tilde{p}$ are the class probabilities using the first and iterative BU passes respectively, and y is the ground truth class label. α is the factor that defines the emphasis on either term. It is set to one unless otherwise stated. $\mathcal{L}_F$ and $\mathcal{L}_S$ are the cross-entropy loss functions for the true target labels y_i and the probability predictions p_i and $\tilde{p}_i$ of the first and iterative feedforward passes respectively. The cross entropy loss function $\hat{\mathcal{L}}$ is defined as:

$$\hat{\mathcal{L}}(p_i, y_i) = -\sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y_{(i)} = k\} \log p(y_{(i)} = k \mid x_{(i)}; W), \tag{7.6}$$

where the indicator function $1\{a = b\}$ is one if $a = b$ and zero otherwise. $p(y_{(i)} = k \mid x_{(i)}; W) = p_i^k$ is the softmax prediction probability of class k given the input sample x_i and the network parameter W. The first term in the definition of $\mathcal{L}_T$ maintains the representational fidelity to the pre-trained BU network while the second term enforces the concentration of the learning algorithm on the TD attention traces. This encourages the network to learn to separate the representations of the background context from the foreground target objects. This hypothesis is examined in experimental evaluation and the observations supporting the role of attention to untangle the representation are demonstrated in Sec. 7.4.

The Stochastic Gradient Descent (SGD) optimization method is used for the training of the neural network. The error gradients are computed using the loss function and propagated backward to the input layer. The weight gradients are accumulated using the computation graphs generated in the first and iterative BU passes. They each contribute separately to the accumulation of gradients to update weight parameters at each SGD updating iteration. Importantly, the error gradients through the iterative BU pass are back-propagated

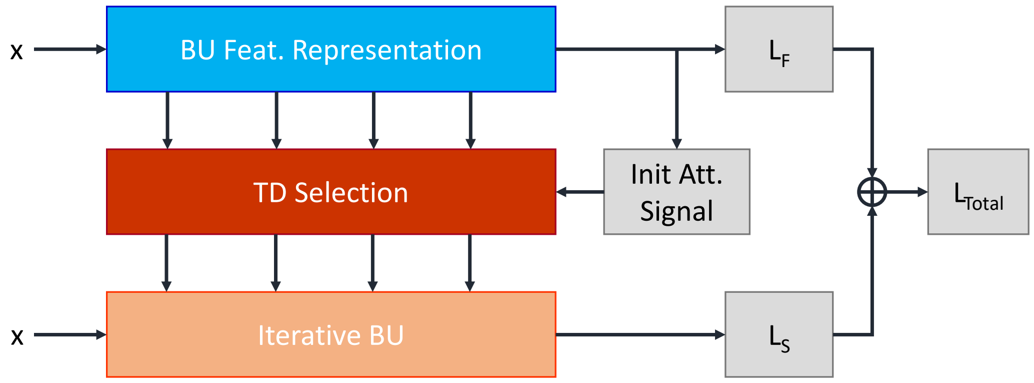


Figure 7.1: The TD network modulates the BU feature representation in the iterative BU pass. The total loss is defined as the weighted sum of the loss of the first and second BU passes.

according to the gating patterns that impacted the feedforward information flow in the iterative BU pass. This gating mechanism helps the optimization algorithm focus on the spatial regions and feature channels that most contributed to the prediction of the input samples at the first pass. The gradient signals are masked at each layer according to the selection patterns formed by the gating activities. Over various updating iterations, the network learns the representation using which a higher degree of robustness to contextual perturbation is obtained.

Figure 7.1 depicts the flow of the information from the BU feature representation into the TD selective attention block. Once the TD pass ends at the end of the visual hierarchy, the iterative BU pass is started given the same mini-batch of input data. The iterative feedforward pass has modulatory units that change the information flow according to the gating activity responses. The iterative pass, therefore, forms a visual representation with an emphasis on the attended regions and feature channels. The confidence score outputs of

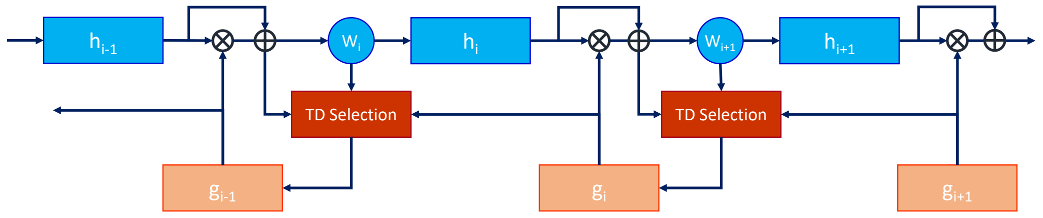


Figure 7.2: The gating activities at each layer modulate the hidden activities in the iterative BU pass.

the two feedforward passes define the $\mathcal{L}_F$ and $\mathcal{L}_S$ loss terms that are combined in Eq. 7.5 to define the total loss function $\mathcal{L}_T$. Once the loss value is computed, the computation graph is used in the SGD optimization algorithm to calculate the parameter gradients of the entire network. The SGD optimization algorithm aims to minimize $\mathcal{L}_T$ in the fine-tuning phase. This basically means that the negative log-likelihood functions derived from the confidence score outputs at the end of the two feedforward passes needs to be reduced. This further implies that the learned representation needs to maintain the class probability prediction capability at a high level of accuracy in the two feedforward passes. Not only does the first feedforward pass is important similar to a regular fine-tuning approach, but also the emphasis to the important aspects of the learned representation is increased by the attentive TD gating mechanisms in the iterative feed forward pass. Fig. 7.2 provides in detail the information flow in the BU pass, the TD pass, and the modulatory interaction of the TD pass with the iterative BU pass. At each layer, the gating activities g_i modulates the hidden activities h_i in the second BU pass. The result then is passed to the parametric transformation function.

7.4 Experimental Results

We evaluate the proposed selective fine-tuning of neural networks on a modified MNIST dataset called Wide-MNITS (WMNIST). MNIST is a handwritten digit classification dataset. The gray-scale image samples in the dataset contain handwritten digits of category zero to nine. We pre-train the BU network on WMNIST for 15 epochs before the evaluation of the proposed method. Once, the BU network is selectively fine-tuned for a number of epochs, we measure the robustness of the final network to the background noise perturbation. The experimental results reveal that the attention-augmented loss function improves the accuracy rate while obtain stronger robustness to noise perturbation.

7.4.1 Implementation Details

We define two choices of convolutional neural network architecture for the BU network: LeNet-5 [29] or AlexNet [30]. The TD network is defined by extending the implementation of STNet [108] for object localization to consider the new requirements of the iterative BU pass. We define the BU and TD framework in PyTorch deep learning framework [1] [162]. The dynamic graph engine in Pytorch allows the active gating of the hidden activities in the iterative pass to be systematically implemented. The SGD optimization method uses the learning rate 10^{-3}, momentum 0.9, weight decay 0.0005, and mini-batch size 64 unless otherwise mentioned for the pre-training and fine-tuning phases. Having the pre-trained BU network loaded, using the selective fine-

[1]https://pytorch.org/

Model	classification	localization
LeNet-5-reference	94.0%	96.4%
LeNet-5-sft	**97.5%**	**99.1%**
AlexNet-reference	97.1%	98.2%
AlexNet-sft	**99.3%**	**99.8%**

Table 7.1: The classification and localization rates of the selective fine-tuned network on the WMNIST dataset.

tuning method, we update the network parameters for 15 epochs and then report the accuracy metric in Table 7.1.

7.4.2 Wide MNIST Dataset

The experimental evaluation is designed to examine the role of the background context for the category label prediction of the foreground target object. The role of the background representation is explicitly highlighted by considering a relatively large context in the input data distribution.

Dataset and Evaluation: MNIST dataset contains 28×28 gray-scale digit images. We increase the size of images by expanding the background context such that images have the size 64×64. We additionally randomize the location of digits in images. In addition to the ground truth labels, while expanding image samples based on the aforementioned protocol, we also extract the tightest bounding box around the digit shape. We use both types of ground truth to measure the performance of the proposed method using the 0-1 classification and the IoU (0.5) localization accuracy rates.

Quantitative Results: The evaluation result for the LeNet-5 and AlexNet using the classification and localization metrics are reported in Table 7.1. The

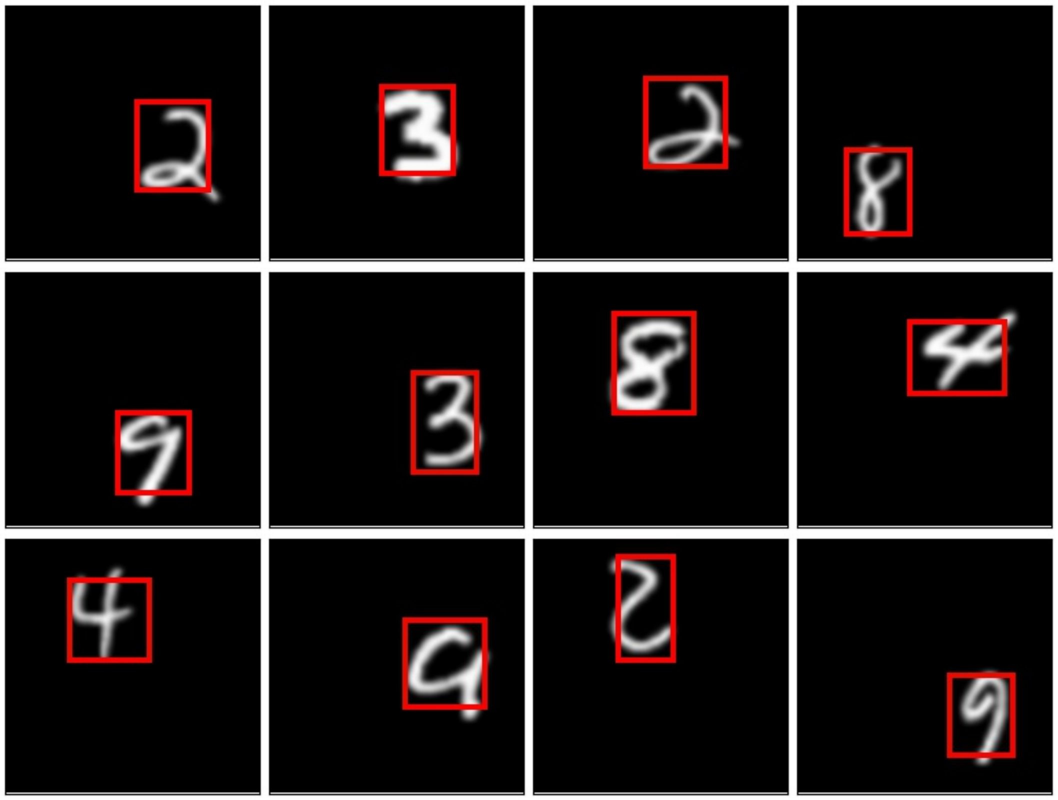


Figure 7.3: Illustration of sample digit images in the WMNIST dataset. The red boxes are the predicted bounding boxes using the LeNet-5 BU pass for feature encoding and the TD selection pass for object localization.

selectively fine-tuned neural networks report improved performance results. The results underline the role of the TD selective pass on network parameter optimization using the attention-augmented loss function. Not only the localization but also the classification results are improved once the network is fine-tuned using the proposed approach. Fig. 7.3 illustrates sample images with the predicted bounding boxes as the means of object localization using the LeNet-5 network architecture. The bounding boxes are predicted using the localization approach presented in STNet [108]. Since the gating activities at the input layer are used for box predictions and the input images are gray-

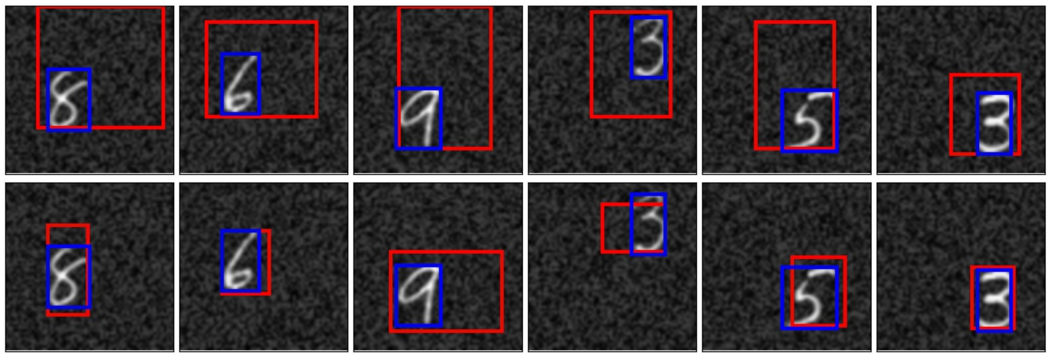


Figure 7.4: Demonstration of the effect of the additive uniform noise in the background and the comparison of the localization performance of the the LeNet-5 reference model (top) with the selective fine-tuned model (bottom). The ground truth and predicted boxes are depicted with the blue and red boxes respectively. The additive noise is taken from a uniform distribution with a lower and upper bounds of 0 and 100 respectively.

scale, we only need to find a tight enclosing box around all of the non-zero gating units. We do not use any pruning strategy to remove units with small gating values.

Ablation Analysis: We study further the role of the selective fine-tuning method on the separation of the foreground from the background representations. We use additive uniform noise in the background to study the impact of the context interference on the target object classification and localization predictions. We gradually increase the upper bound of the uniform noise function to measure the robustness of the reference and fine-tuned models in sever situations. Fig. 7.5 demonstrates the amount of classification robustness obtained using the selective models over the reference models for different levels of background additive noise. For both LeNet-5 and AlexNet network architectures, the selective fine-tuning brings a significant level of robustness to the reference models. This result indicates that during selective fine-tuning,

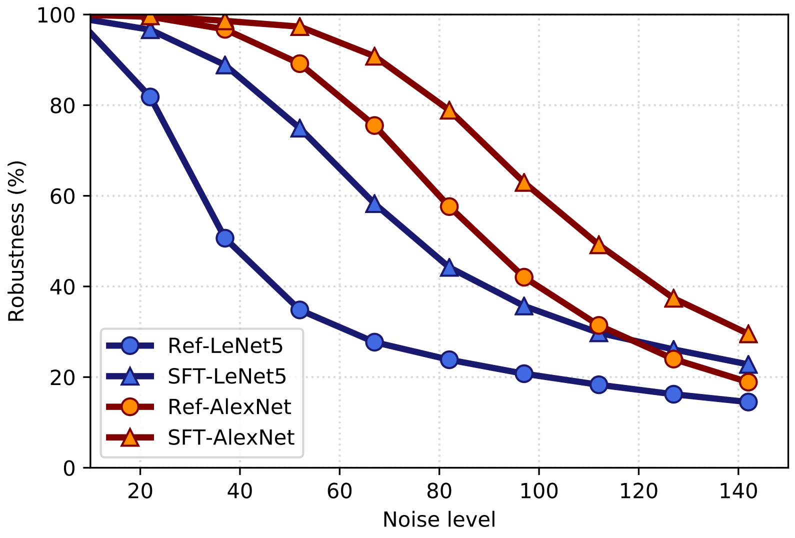


Figure 7.5: The effect of the additive noise distortion in the background on the classification accuracy rate. Ref and SFT refer to the reference and selectively fine-tuned models respectively. The vertical axis represents the robustness of the fine-tuned network at different noise levels. Robustness is calculated by the ratio of the accuracy rates of the noisy images over the clean images. The horizontal axis indicates the maximum amount of pixel intensity the uniform distribution may add to the background pixels.

the network learns to focus further on the features encoding of the foreground target objects and blocking contextual interference. In addition to the classification task, Fig. 7.6 reveals the localization accuracy is also maintained over different levels of additive noise using the proposed method. Fig. 7.4 qualitatively illustrates the cases the reference model fails to deal with the background noise. It underlines the fact that in the reference model the representation of the background context is entangled with the foreground target object. This explains why a simple form of contextual perturbation quickly destroys the localization and classification performance of the reference model. The selective fine-tuning approach, however, obtains a higher degree of robustness in such

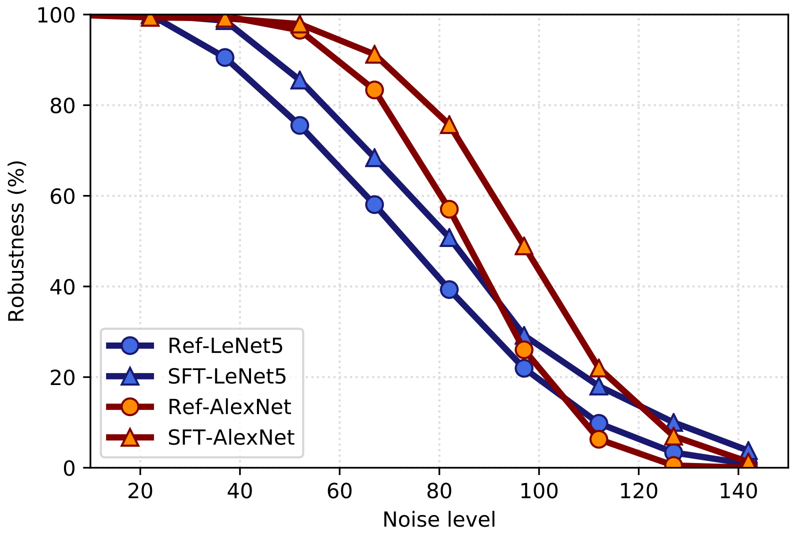


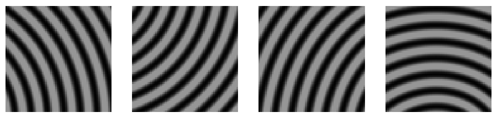
Figure 7.6: The effect of the additive noise in the background on the localization accuracy rate. Ref and SFT refer to the reference and selective fine-tuned models respectively. The horizontal axis indicates the maximum amount of pixel intensity the uniform distribution may add to the background pixels. Robustness is calculated by the ratio of the accuracy rates of the noisy images over the clean images.

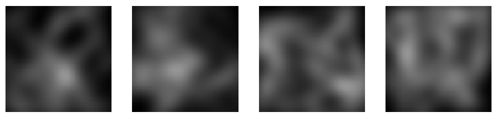
sever cases. Apparently, the network learns through the selectivity of the TD attention to concentrate on the foreground representation.

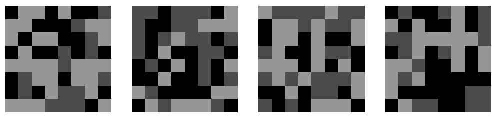
We further experiment with different types of noise generation functions to validate the generalization achieved by the selective fine-tuning approach. We choose four different noise sources based on which we choose to perturb the background regions as follows: (1) Grating: this is the radial grating method with a center coordinate randomly chosen for every input image. (2) MoG: this is a mixture of K Gaussian distributions such that each Gaussian has a random center coordinate and orientation. We choose K=50 since it provides smooth and irregular noise patterns. (3) Squares: this generates a K×K grid of squares with random pixel intensity values. We choose K=8 since it generates large enough square blocks that distinguishes them from random uniform noise patterns. (4) RLines: this generates K short line segments with random center coordinates and orientations. We choose K=100 to cover the entire background regions with enough number of noise patterns. Figure 7.7 illustrates four random samples generated by these noise generation methods.

These noise methods have chosen such that they cover a variety of shape patterns from small scale to large scale with different line structures and curvatures. We would like to measure the sensitivity of the reference and fine-tuned networks on the samples perturbed with the background noise generated by these methods. Similar to the experiment with the random uniform noise, we report the robustness results to these four noise methods on the LeNet-5 and AlexNet networks for the classification and localization evaluation metrics in Fig. 7.8 and Fig. 7.9 respectively.

The results reveal that the generalization against contextual noise achieved by the proposed fine-tuning method is persistent across all of the four noise sources for classification and localization. The robustness for Grating and Squares is less than for RLines due to the larger scale of noise patterns. Similar to the uniform noise patterns, RLines have small scale random elements. Both Grating and Squares show consistent robustness gain once the selective fine-tuning is used. MoG is the only method that benefits from smooth and continuous shape patterns. The proposed method still provides slight robustness gain in comparison with the reference networks. Though, the gap is small for AlexNet, we observe improvement for LeNet-5. The qualitative results for this experimental evaluation setup is illustrated in Fig. 7.10 and Fig. 7.11 for


(a) Grating


(b) MoG


(c) Squares

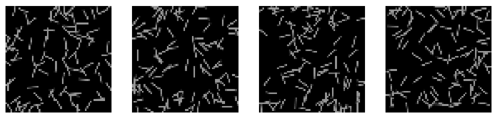
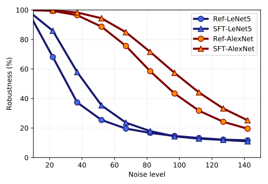
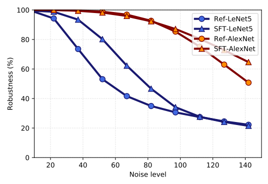

(d) RLines

Figure 7.7: Random samples generated by the four noise methods: (a) Grating: radial grating with random centers, (b) MoG: Mixture of Gaussians, (c) Squares: squares with random intensity values, and (d) RLines: short lines with random centers and orientation.

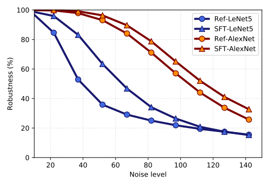
LeNet-5 and AlexNet respectively. They show the predicted bounding boxes of the reference and the proposed fine-tuned networks on the perturbed input samples using the four different noise methods. The results support the hypothesis that the TD selective gating method is capable of focusing the learning capacity of the network on the important aspects so then the prediction performance is less affected by the contextual perturbations. The qualitative results illustrates the cases in which the reference network fails to predict the class labels and bounding boxes accurately due to the background noise disturbance. On the other hand, the selective fine-tuned counterpart provides a more robust representation and it maintains prediction performance despite significant background noise patterns. This generalizes across all four different noise methods for both LeNet-5 and AlexNet networks.



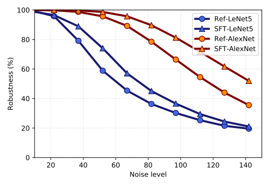
(a) Grating



(b) MoG

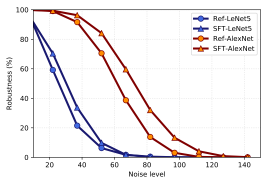


(c) Squares

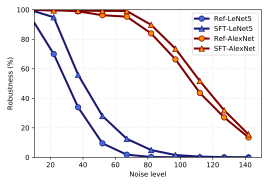


(d) RLines

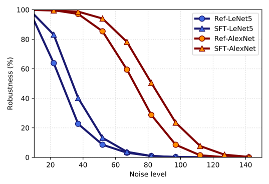
Figure 7.8: Comparing the effect of different methods of generating contextual noise perturbation on the classification accuracy. From left to right: (a) Grating: radial grating with random centers, (b) MoG: Mixture of Gaussians, (c) Squares: squares with random intensity values, and (d) RLines: short lines with random centers and orientation. The vertical axis represent the classification robustness metric, and the horizontal axis represent the maximum pixel intensity the noise adds to the background.



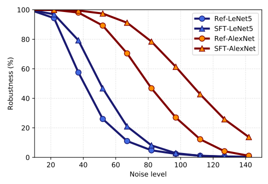
(a) Grating



(b) MoG

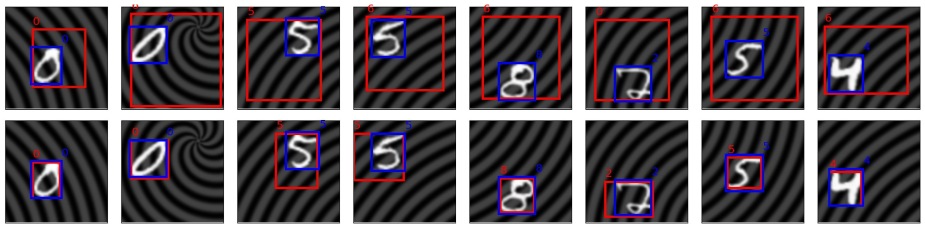


(c) Squares

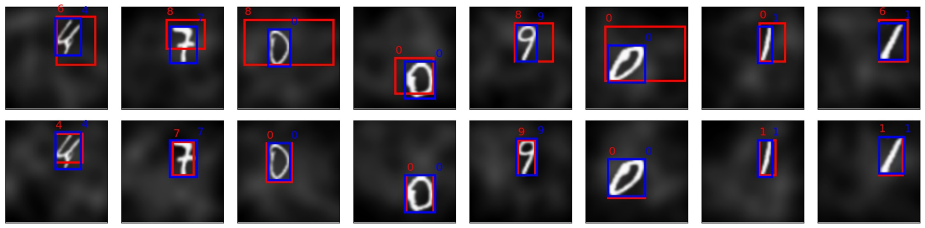


(d) RLines

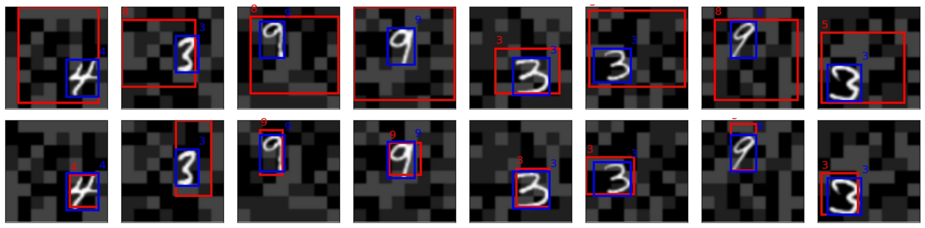
Figure 7.9: Comparing the effect of different methods of generating contextual noise perturbation on the localization accuracy. From left to right: (a) Grating: radial grating with random centers, (b) MoG: Mixture of Gaussians, (c) Squares: squares with random intensity values, and (d) RLines: short lines with random centers and orientation. The vertical axis represent the localization robustness metric, and the horizontal axis represent the maximum pixel intensity the noise adds to the background.



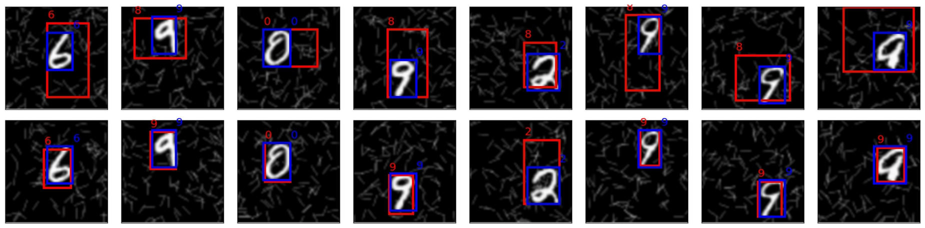
(a) Grating



(b) MoG

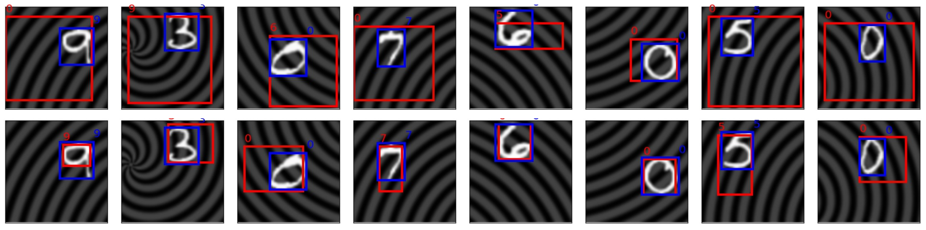


(c) Squares

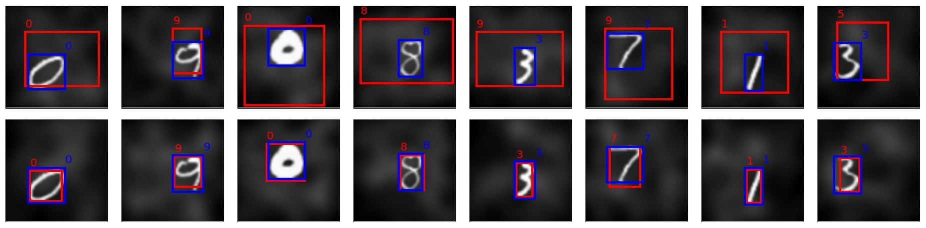


(d) RLines

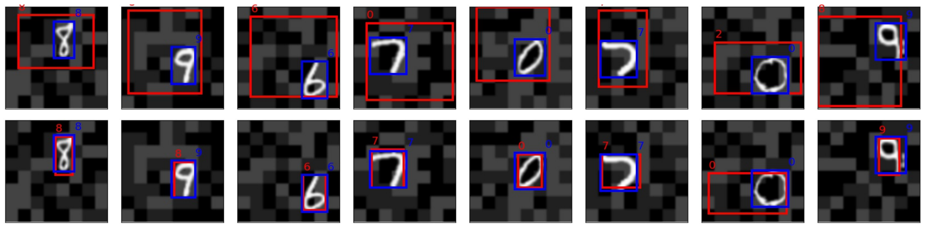
Figure 7.10: Comparison of the label and bounding box predictions of the LeNet-5 reference and fine-tuned networks once the background regions is perturbed with four different types of noise methods. In each section, the top and bottom rows represent predictions from the reference and selective fine-tuned networks. The ground truth and predicted bounding boxes are illustrated with blue and red boxes respectively. The ground truth and predicted labels are shown at the top-left and top-right of their corresponding box respectively.



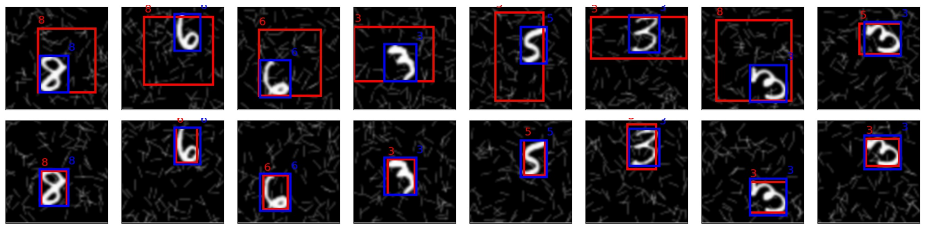
(a) Grating



(b) MoG



(c) Squares



(d) RLines

Figure 7.11: Comparison of the label and bounding box predictions of the AlexNet reference and fine-tuned networks once the background regions is perturbed with four different types of noise methods. In each section, the top and bottom rows represent predictions from the reference and selective fine-tuned networks. The ground truth and predicted bounding boxes are illustrated with blue and red boxes respectively. The ground truth and predicted labels are shown at the top-left and top-right of their corresponding box respectively.

7.5 Conclusion

Attention helps humans to learn in distracting and interfering situations. The selective nature of attentional processes are very well established in human vision studies. We propose a selective learning method for neural networks that has TD attentive mechanisms. We define an iterative feedforward pass using the modulation of the first feedforward pass with the TD gating activities. We experimentally test the impact of the background context when the network is trained with the proposed method. The evaluation results on a modified MNIST dataset indicate that the selection mechanism indeed constrains the learning capacity of the network on relevant aspects of the visual representation for target semantic abstractions. Over time, the network parameters converge to the state that has reduced contextual interference and improved robustness against distortions such as additive noise perturbations of the background regions. The qualitative and quantitative results support the role of the selective fine-tuning using the iterative feedforward pass and an augmented loss function.

Chapter 8

Conclusions and Future Directions

In this thesis, we define a novel Top-Down (TD) selection formulation over a visual hierarchy represented by multi-layer neural networks. We strive to model the TD selection formulation in a principled approach according to the target visual tasks. We hypothesize that the use of the TD attentive mechanisms will play effective roles in deep neural networks. We develop an implementation of the proposed TD selection mechanism for various visual tasks such as object localization and segmentation. We shed light on the critical aspects of such a hierarchical selection mechanism through various experimental setups and ablation studies in several visual tasks. The selective and modulatory roles of the proposed TD mechanism on the network feedforward responses are investigated. We speculate future research directions to further elucidate the complementary role of a TD selection mechanism for Bottom-Up (BU) feedforward networks.

8.1 Summary of Contributions

Chapter 3: We first test the hypothesis that whether TD processing is capable of spatially localizing objects in natural input images using convolutional neural networks. We investigate the role of hierarchical selective processing that begins from semantic task signals and gradually descends to lower layers. The question is whether such TD processing is capable of routing successfully through the visual hierarchy, activating important attentional traces, and finally reaches to the location where the instance of an object category can be found. We propose a unified BU and TD framework called STNet that is successful in localizing objects. We test STNet in the weakly-supervised object localization experimental setting on the ImageNet 2015 validation dataset using the IoU (0.5) evaluation metric. The proposed model is on par with the state-of-the-art using VGGNet, and GoogleNet with the localization error rates of 40.1, 38.6 respectively and outperforms the state-of-the-art using AlexNet with 40.3. STNet leverages the power of the multi-layer feature encoding by convolutional neural networks in the BU pass. The BU pass is unified with a novel TD selection approach formulating the TD selection as a cascading series of local attentive selection processes each consisting of three computational stages: (1) inference reduction, (2) similarity grouping, and (3) attention signal normalization and propagation. These computational stages are encapsulated in a new custom layer that defines the building-block of the

TD network in STNet. The first stage performs a type of local competition implemented using an adaptive thresholding policy. The second stage defines either a spatial or statistical local grouping strategy. The third stage selects the final winners of the second stage and propagates normalized gating activities of the top nodes to lower layer nodes.

Chapter 4: We hypothesize that visual task priming, as a form of TD processing, has a critical role in demanding scenarios such as detection of scene elements that are nearly unnoticeable. We test whether there is a systematic approach to alter, tune, and prime the computational routines of the visual hierarchy derived by convolutional neural networks. The experimental results on object detection and object segmentation on the PASCAL VOC 2007 and 2012 datasets reveal that the proposed approach with the TD task cuing mechanism is capable of improving the baseline results. The priming of the DeepLab semantic segmentation model leads to an improvement of the mean IoU metric from 76.3% to 77.15% using ResNet-101 as the base network. For object detection, once the YOLOv2 model is equipped with the priming mechanism, the baseline performance of 76.8% mAP is improved to 80.6% using priming. The priming mechanism also addresses demanding situations such as detection under heavy noise distortion. For Gaussian noise with standard deviation of 80, the priming approach obtains 34.8% in mAP compared to the baseline result of 24.1%. The proposed method adds novel modulation units to the feedforward layers to weight feature planes according to the priming cues. The modulation units have weight parameters that are learned in a preliminary training phase. In the test phase, the task cuing mechanism signals the modulation units to tune the feedforward information flow by weighting the hidden activities organized in the feature planes according to the learned parameters.

Chapter 5: We hypothesize that the feature selectivity of the TD mechanisms proposed in STNet for object localization and task priming are also useful for the tasks that require pixel-level semantic label predictions such as semantic segmentation. We examine the role of the proposed TD attention to modulate BU hidden activities for object segmentation prediction. The question is whether the selection patterns generated by the TD mechanisms are reliable to gate features into segmentation layers at multiple levels of the visual hierarchy. The Selective Segmentation Network (SSN) is evaluated experimentally on three challenging semantic segmentation benchmark datasets: PASCAL VOC 2012, CamVid, and Horse-Cow Parsing datasets. Using the mean IoU metric, SSN defined by the base network of VGGNet improves a baseline results of 62.7%, 57.0%, and 64.1% to 64.3%, 57.0%, and 65.2% on the three datasets respectively. Additionally, SSN gains further robustness to uniform, salt-pepper, and box-occlusion additive noise functions. First, we re-define the unified BU and TD information processing framework proposed in STNet to handle the requirements of object segmentation. Second, the modulation units are defined to gate BU feature maps using TD activities at multiple layers. A segmentation network is defined to receive modulated feature activities and fuse them into a parametric up-sampling pipeline for the segmentation output prediction. Additionally, a network block is defined to predict the activation of TD selection mechanisms at different location in a multi-scale manner. Lastly, a multi-loss function is defined to optimize the network parameters for segmentation and TD activation predictions.

Chapter 6: We hypothesize that the TD processing is capable of selecting not only important feature activities for localization and segmentation but also the critical kernel weight parameters for network pruning and representation redundancy reduction. The research question is whether the accumulation of the kernel importance responses is a reliable source of generating pruning patterns to reduce the number of active kernel parameters. The experimental results on MNIST and CIFAR-10 datasets using popular base networks reveals that the TD selection mechanisms can be leveraged for compact neural representations. On MNIST, the attentive network pruning approach achieves the compression ratio of 58$\times$ and 102$\times$ using LeNet-300-100 and LeNet-5 respectively. On CIFAR-10, the compression ratio is 10$\times$ and 29$\times$ using LeNet-5 and AlexNet respectively. We define a novel task-driven pruning method by TD selection mechanisms. The activated connections in the TD selection traversals are pooled and stored in persistent buffers using which periodic pruning proceeded by retraining phases are scheduled for a number of iterations. Unlike the ad hoc baseline pruning approaches that determine pruning masks based on purely kernel weights, the proposed method relies on values derived from not only the kernel weights but also the hidden feature activities in addition to the task-driven TD gating activities.

Chapter 7: We hypothesize that fine-tuning of TD-attention-augmented neural networks using iterative feedforward passes improves the label prediction and localization performance in sample images with large background context. Moreover, this systematically shifts the focus of the final learned representation away from the features encoding the background context for the sake of a more disentangled target object representation. We investigate how much robustness the neural networks gain once the learning is constrained by the TD attentional traces. We train and test the proposed method on a modified version of the MNIST dataset. We randomly zero-pad sample images in MNIST such that the size of images is 64×64 and the handwritten digits are randomly located in the input images. The baseline label prediction and localization results improves from 94.0% and 96.4% to 97.5% and 99.1% for LeNet-5 and from 97.1% and 98.2% to 99.3% and 99.8% for AlexNet respectively. Furthermore, the network fine-tuned using the attentive approach gains significant robustness against contextual perturbation under various levels of additive noise to the background regions. Using the proposed TD processing, an iterative Bottom-Up pass of feature encoding is defined such that in the first pass, the features are encoded using the core multi-layer neural network. The TD pass produces the hierarchical gating activities using which the second iterative pass will be modulated. In the iterative pass, the modulation units weight the hidden feature activities in the channel and spatial dimensions. The additive skip connections are used to stabilize the learning process. A two-pass loss function is defined by adding the loss term measured in the first pass with the one in the iterative pass. We update network parameters based on the accumulated gradient signals generated by the multi-loss function.

8.2 Future Directions

The hierarchical TD selection approach proposed in this work opens up wide varieties of future research directions. In chapter 3, we introduce a novel multi-stage selection process in neural networks that conduct competitions on local activities retrieved from the receptive fields of hidden units. Based on different task knowledge, other selection rules can be investigated to satisfy the task requirements. For instance, for object localization, computational stages to formulate shape priors can defined to enforce characteristics such as object aspect ratio, object parts, and appearance attributes.

Further, the formulation of the selection stages using parametric methods can be investigated in the future. The selection process could be formulated by parametric density estimation models such as the mixture of Gaussian models to learn parametric selection strategies.

Additionally, a soft-selection mechanism can be tested instead of a parametric winner-take-all mechanism. In this case, all the nodes in the receptive field will participate in the top-down propagation of gating activities rather than a small number of them.

The other future direction is that the first stage of the selection process can be replaced with a histogram-based strategy rather than an adaptive thresholding currently being used. This will increase the processing speed of the top-down pass since there is no need to find the thresholding value in the first stage using the sorting and cumulative-sum algorithms.

Tasks related to video understanding requires spatio-temporal feature representations that encode features not only in the spatial but also the temporal dimensions. Temporal localization in tasks such as action detection is a very critical requirement. One straightforward future direction is to investigate the role of the TD selection pass over the spatio-temporal feature hierarchy for action localization.

Recently, tasks such as visual question answering and visual caption generation move beyond classical modeling of object recognition problems and require a multi-modal setting. In such settings, the goal is to align and match the feature representation of one input modality with another one for the fulfillment of the task requirements. For instance, in visual caption generation, the language model needs a selection mechanism that collects the feature encoding across the spatial dimensions at salient regions to generate captions best describe the input image. The variant of the TD selection approach potentially can be utilized to route information according to multi-stage computational mechanisms and select relevant features at multiple levels. The object localization capability of the TD mechanisms can provide relevant features for caption generation.

Finally, multi-task learning in dynamic environments plays a critical role in the development of an intelligent system. The end goal is to use a core feature hierarchy that can be utilized for the predictions of multiple relevant tasks. Our proposed TD selection mechanism can be extended in this scenario to implement a gating mechanism that alters and tunes the base representation according to the task cuing. For instance, if the multi-task setting is to predict category labels and attributes of object categories, then the role of the TD selection mechanism is to tune the base network for the attribute prediction of different categories. For instance, the visual hierarchy should be tuned for attribute predictions of car instances differently from bus instances.

These are the most important future directions that will provide more insights about the underlying characteristics of top-down processing along with the bottom-up processing in convolutional neural networks. Further investigation of attentive processes in neural networks can help improve our understanding of the representational characteristics in such models derived using learning approaches. Additionally, attention mechanisms can provide dynamic processing in neural networks which is required for active perception in vision systems.

Chapter 9

Bibliography

[1] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[2] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-down neural attention by excitation backprop," in *European Conference on Computer Vision*, pp. 543–559, Springer, 2016.

[3] M. A. Islam, M. Rochan, N. D. Bruce, and Y. Wang, "Gated feedback refinement network for dense image labeling," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4877–4885, IEEE, 2017.

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European Conference on Computer Vision*, pp. 21–37, Springer, 2016.

[6] M. Carrasco, “Visual attention: The past 25 years,” *Vision Research*, vol. 51, no. 13, pp. 1484–1525, 2011.

[7] J. K. Tsotsos, *A computational perspective on visual attention.* MIT Press, 2011.

[8] S. Frintrop, E. Rome, and H. I. Christensen, “Computational visual attention systems and their cognitive foundations: A survey,” *ACM Transactions on Applied Perception (TAP)*, vol. 7, no. 1, p. 6, 2010.

[9] X. Zhang, Y.-H. Yang, Z. Han, H. Wang, and C. Gao, “Object class detection: A survey,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 10, 2013.

[10] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, “Salient object detection: A benchmark,” *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5706–5722, 2015.

[11] C. D. Gilbert and M. Sigman, “Brain states: top-down influences in sensory processing,” *Neuron*, vol. 54, no. 5, pp. 677–696, 2007.

[12] C. D. Gilbert and W. Li, “Top-down influences on visual processing,” *Nature Reviews Neuroscience*, vol. 14, no. 5, pp. 350–363, 2013.

[13] F. Baluch and L. Itti, "Mechanisms of top-down attention," *Trends in Neurosciences*, vol. 34, no. 4, pp. 210–224, 2011.

[14] R. Desimone and J. Duncan, "Neural mechanisms of selective visual attention," *Annual Review of Neuroscience*, vol. 18, pp. 193–222, Jan. 1995.

[15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[16] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

[17] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *arXiv preprint arXiv:1801.00553*, 2018.

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644*, 2016.

[19] A. Rosenfeld, R. Zemel, and J. K. Tsotsos, "The elephant in the room," *arXiv preprint arXiv:1808.03305*, 2018.

[20] J.-M. Jolion and A. Rosenfeld, "A pyramid framework for early vision: Multiresolutional computer vision," 1994.

[21] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo, “Modeling visual attention via selective tuning,” *Artificial Intelligence*, vol. 78, no. 12, pp. 507 – 545, 1995. Special Volume on Computer Vision.

[22] M. H. Herzog and A. M. Clarke, “Why vision is not both hierarchical and feedforward,” *Frontiers in Computational Neuroscience*, vol. 8, p. 135, 2014.

[23] P. Perona, *Visual recognition circa 2008*, pp. 55–68. Cambridge university press, 2009.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, pp. 211–252, sep 2015.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[26] H. Caesar, J. Uijlings, and V. Ferrari, “Coco-stuff: Thing and stuff classes in context,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1209–1218, 2018.

[27] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 202, 1980.

[28] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[31] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” *International Conference on Machine Learning*, no. 3, pp. 807–814, 2010.

[32] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.

[33] H. R. Wilson, *Spikes, decisions, and actions: the dynamical foundations of neurosciences*. Oxford University Press, 1999.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv preprint arXiv:1502.01852*, 2015.

[35] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, vol. 30, 2013.

[36] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *IEEE International Conference on Computer Vision*, pp. 2146–2153, IEEE, 2009.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[38] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[39] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[40] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[41] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," in *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6181–6189, 2018.

[42] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, pp. 1019–1025, 1999.

[43] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[44] J. T. Springenberg and M. Riedmiller, "Improving deep neural networks with probabilistic maxout units," *arXiv preprint arXiv:1312.6116*, 2013.

[45] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *International Conference on Machine Learning*, pp. 609–616, ACM, 2009.

[46] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 215–219, IEEE, 2014.

[47] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," in *Machine Learning and Knowledge Discovery in Databases*, pp. 530–546, Springer, 2014.

[48] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[49] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[51] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2169–2178, IEEE, 2006.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *arXiv preprint arXiv:1406.4729*, 2014.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., DTIC Document, 1985.

[54] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[55] J. Ba and B. Frey, “Adaptive dropout for training deep neural networks,” in *Advances in Neural Information Processing Systems*, pp. 3084–3092, 2013.

[56] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, “Regularization of neural networks using dropconnect,” in *International Conference on Machine Learning*, pp. 1058–1066, 2013.

[57] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *IEEE International Conference on Computer Vision*, pp. 2018–2025, IEEE, 2011.

[58] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision*, pp. 818–833, Springer, 2014.

[59] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” *arXiv preprint arXiv:1312.6034*, 2013.

[60] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” *arXiv preprint arXiv:1506.06579*, 2015.

[61] A. Mahendran and A. Vedaldi, “Visualizing deep convolutional neural networks using natural pre-images,” *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, 2016.

[62] A. Nguyen, J. Yosinski, and J. Clune, "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks," *arXiv preprint arXiv:1602.03616*, 2016.

[63] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems*, pp. 3387–3395, 2016.

[64] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[65] B. Alexe, T. Deselaers, and V. Ferrari, "What is an object?," in *IEEE Computer Vision and Pattern Recognition*, pp. 73–80, IEEE, 2010.

[66] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[67] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[68] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[69] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 328–335, IEEE, 2014.

[70] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European Conference on Computer Vision*, pp. 391–405, Springer, 2014.

[71] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2155–2162, IEEE, 2014.

[72] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[73] L. Bazzani, A. Bergamo, D. Anguelov, and L. Torresani, "Self-taught object localization with deep networks," in *IEEE Winter Conference on Applications of Computer Vision (wacv)*, pp. 1–9, IEEE, 2016.

[74] J. Carreira and C. Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3241–3248, IEEE, 2010.

[75] F. Li, J. Carreira, and C. Sminchisescu, "Object recognition as ranking holistic figure-ground hypotheses," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1712–1719, IEEE, 2010.

[76] I. Endres and D. Hoiem, "Category independent object proposals," in *European Conference on Computer Vision*, pp. 575–588, Springer, 2010.

[77] J. K. Tsotsos, "Analyzing vision at the complexity level," *Behavioral and Brain Sciences*, vol. 13, no. 03, pp. 423–445, 1990.

[78] F. S. Khan, J. van de Weijer, and M. Vanrell, "Modulating shape features by color attention for object recognition," *International Journal of Computer Vision*, vol. 98, no. 1, pp. 49–64, 2012.

[79] S. Chikkerur, T. Serre, C. Tan, and T. Poggio, "What and where: A bayesian inference theory of attention," *Vision Research*, vol. 50, no. 22, pp. 2233–2247, 2010.

[80] D. P. Reichert, P. Series, and A. J. Storkey, "A hierarchical generative model of recurrent object-based attention in the visual cortex," in *International Conference on Artificial Neural Networks*, pp. 18–25, Springer, 2011.

[81] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

[82] K. Fukushima, "A hierarchical neural network model for associative memory," *Biological Cybernetics*, vol. 50, no. 2, pp. 105–113, 1984.

[83] K. Fukushima, "A neural network model for selective attention in visual pattern recognition," *Biological Cybernetics*, vol. 55, no. 1, pp. 5–15, 1986.

[84] D. B. Walther and C. Koch, "Attention in hierarchical models of object recognition," *Progress in Brain Research*, vol. 165, pp. 57–78, 2007.

[85] A. L. Rothenstein and J. K. Tsotsos, "Attentional modulation and selection–an integrated approach," *PloS one*, vol. 9, no. 6, p. e99681, 2014.

[86] O. J. Avella Gonzalez and J. K. Tsotsos, "Short and long-term attentional firing rates can be explained by st-neuron dynamics," *Frontiers in neuroscience*, vol. 12, p. 123, 2018.

[87] K. Fukushima, "Artificial vision by multi-layered neural networks: Neocognitron and its advances," *IEEE Transactions on Neural Networks*, vol. 37, pp. 103–119, 2013.

[88] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," in *Matters of intelligence*, pp. 115–141, Springer, 1987.

[89] Y. Tang, N. Srivastava, and R. R. Salakhutdinov, "Learning generative models with visual attention," in *Advances in Neural Information Processing Systems*, pp. 1808–1816, 2014.

[90] C. H. Anderson and D. C. Van Essen, "Shifter circuits: a computational strategy for dynamic aspects of visual processing," *Proceedings of the National Academy of Sciences*, vol. 84, no. 17, pp. 6297–6301, 1987.

[91] B. A. Olshausen, C. H. Anderson, and D. C. Van Essen, "A neurobiological model of visual attention and invariant pattern recognition based on

dynamic routing of information," *The Journal of Neuroscience*, vol. 13, no. 11, pp. 4700–4719, 1993.

[92] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

[93] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *Advances in Neural Information Processing Systems*, pp. 1243–1251, 2010.

[94] R. Memisevic and G. E. Hinton, "Learning to represent spatial transformations with factored higher-order boltzmann machines," *Neural Computation*, vol. 22, no. 6, pp. 1473–1492, 2010.

[95] Y. Zheng, R. S. Zemel, Y.-J. Zhang, and H. Larochelle, "A neural autoregressive approach to attention-based recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 67–79, 2014.

[96] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

[97] M. Ranzato, "On learning where to look," *arXiv preprint arXiv:1405.5488*, 2014.

[98] V. Mnih, N. Heess, A. Graves, *et al.*, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.

[99] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[100] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, "Stacked what-where auto-encoders," *arXiv preprint arXiv:1506.02351*, 2015.

[101] E. A. DeYoe and D. C. Van Essen, "Concurrent processing streams in monkey visual cortex," *Trends in Neurosciences*, vol. 11, no. 5, pp. 219–226, 1988.

[102] D. J. Kravitz, K. S. Saleem, C. I. Baker, L. G. Ungerleider, and M. Mishkin, "The ventral visual pathway: an expanded neural framework for the processing of object quality," *Trends in Cognitive Sciences*, vol. 17, no. 1, pp. 26–49, 2013.

[103] A. Andreopoulos and J. K. Tsotsos, "50 years of object recognition: Directions forward," *Computer Vision and Image Understanding*, vol. 117, no. 8, pp. 827–891, 2013.

[104] A. L. S. Dickinson, *Object categorization: computer and human vision perspectives*, ch. The evolution of object categorization and the challenge of image abstraction, pp. 1–37. Cambridge University Press, 2009.

[105] R. Girshick, "Fast r-cnn," in *IEEE International Conference on Computer Vision*, December 2015.

[106] K. Lee and H. Choo, "A critical review of selective attention: an interdisciplinary perspective," *Artificial Intelligence Review*, vol. 40, no. 1, pp. 27–50, 2013.

[107] E. Kowler, "Eye movements: The past 25years," *Vision Research*, vol. 51, no. 13, pp. 1457–1483, 2011.

[108] M. Biparva and J. K. Tsotsos, "Stnet: selective tuning of convolutional networks for object localization.," in *ICCV Workshops*, pp. 2715–2723, 2017.

[109] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. S. Huang, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *IEEE International Conference on Computer Vision*, pp. 2956–2964, December 2015.

[110] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - weakly-supervised learning with convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[111] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *IEEE International Conference on Computer Vision*, pp. 1796–1804, 2015.

[112] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *IEEE International Conference on Computer Vision*, ICCV '15, (Washington, DC, USA), pp. 1742–1750, IEEE Computer Society, 2015.

[113] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1713–1721, 2015.

[114] J. H. Reynolds and R. Desimone, "The role of neural mechanisms of attention in solving the binding problem," *Neuron*, vol. 24, no. 1, pp. 19–29, 1999.

[115] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia*, pp. 675–678, ACM, 2014.

[116] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, 2009.

[117] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5188–5196, 2015.

[118] S. Adel Bargal, A. Zunino, D. Kim, J. Zhang, V. Murino, and S. Sclaroff, "Excitation backprop for rnns," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1440–1449, 2018.

[119] M. V. Bartsch, K. Loewe, C. Merkel, H.-J. Heinze, M. A. Schoenfeld, J. K. Tsotsos, and J.-M. Hopf, "Attention to color sharpens neural pop-

ulation tuning via feedback processing in the human visual cortex hierarchy," *Journal of Neuroscience*, vol. 37, no. 43, pp. 10346–10357, 2017.

[120] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz, "Scene perception: Detecting and judging objects undergoing relational violations," *Cognitive psychology*, vol. 14, no. 2, pp. 143–177, 1982.

[121] I. Biederman, *On the semantics of a glance at a scene.* 1981.

[122] A. Oliva and A. Torralba, "The role of context in object recognition," *Trends in cognitive sciences*, vol. 11, no. 12, pp. 520–527, 2007.

[123] E. Tulving, D. L. Schacter, *et al.*, "Priming and human memory systems," *Science*, vol. 247, no. 4940, pp. 301–306, 1990.

[124] G. S. Wig, S. T. Grafton, K. E. Demos, and W. M. Kelley, "Reductions in neural activity underlie behavioral components of repetition priming," *Nature neuroscience*, vol. 8, no. 9, pp. 1228–1233, 2005.

[125] t. E. Palmer, "The effects of contextual scenes on the identification of objects," *Memory & Cognition*, vol. 3, no. 5, pp. 519–526, 1975.

[126] A. Hollingworth, "Does consistent scene context facilitate object perception?," *Journal of Experimental Psychology: General*, vol. 127, no. 4, p. 398, 1998.

[127] C. Galleguillos and S. Belongie, "Context based object categorization: A critical survey," *Computer vision and image understanding*, vol. 114, no. 6, pp. 712–722, 2010.

[128] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert, "An empirical study of context in object detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1271–1278, IEEE, 2009.

[129] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin, "Context-based vision system for place and object recognition," in *null*, p. 273, IEEE, 2003.

[130] A. Torralba, "Contextual priming for object detection," *International journal of computer vision*, vol. 53, no. 2, pp. 169–191, 2003.

[131] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, pp. 1–8, IEEE, 2007.

[132] J. Yao, S. Fidler, and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 702–709, IEEE, 2012.

[133] K. P. Murphy, A. Torralba, and W. T. Freeman, "Using the forest to see the trees: A graphical model relating features, objects, and scenes," in *Advances in neural information processing systems*, pp. 1499–1506, 2004.

[134] A. Shrivastava and A. Gupta, "Contextual priming and feedback for faster r-cnn," in *European Conference on Computer Vision*, pp. 330–348, Springer, 2016.

[135] A. W. Harley, K. G. Derpanis, and I. Kokkinos, "Segmentation-aware convolutional networks using local attention masks," in *IEEE International Conference on Computer Vision*, pp. 5038–5047, 2017.

[136] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4733–4742, 2016.

[137] T. M. Strat, "Employing contextual information in computer vision," *DARPA93*, pp. 217–229, 1993.

[138] M. I. Posner, M. J. Nissen, and W. C. Ogden, "Attended and unattended processing modes: The role of set for spatial location," *Modes of perceiving and processing information*, vol. 137, p. 158, 1978.

[139] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. Courville, "Modulating early visual processing by language," *arXiv preprint arXiv:1707.00683*, 2017.

[140] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI Conference on Artificial Intelligence*, 2018.

[141] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433, 2015.

[142] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning," *arXiv preprint arXiv:1612.06890*, 2016.

[143] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[144] H. Katti, M. V. Peelen, and S. Arun, "Object detection can be improved using human-derived contextual expectations," *arXiv preprint arXiv:1611.07218*, 2016.

[145] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[146] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[147] J. Zhang, I. Mitliagkas, and C. Ré, "YellowFin and the Art of Momentum Tuning," *arXiv preprint arXiv:1706.03471*, 2017.

[148] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[149] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, “Semantic contours from inverse detectors,” in *International Conference on Computer Vision*, 2015.

[150] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” *arXiv preprint arXiv:1612.01105*, 2016.

[151] J. Dai, K. He, and J. Sun, “Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1635–1643, 2015.

[152] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[153] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[154] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for

semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016.

[155] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *IEEE International Conference on Computer Vision*, pp. 1520–1528, 2015.

[156] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *International Conference on Learning Representations*, 2015.

[157] Y. Wang, J. Liu, Y. Li, J. Yan, and H. Lu, "Objectness-aware semantic segmentation," in *ACM International Conference on Multimedia*, pp. 307–311, ACM, 2016.

[158] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Ieee Conference on Computer Vision and Pattern Recognition*, pp. 3640–3649, 2016.

[159] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: a retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, jun 2014.

[160] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[161] J. Wang and A. L. Yuille, "Semantic part segmentation using compositional model combining shape and appearance," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1788–1797, 2015.

[162] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 2017.

[163] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062v4*, 2016.

[164] A. Kundu, V. Vineet, and V. Koltun, "Feature space optimization for semantic video segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3168–3175, 2016.

[165] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[166] P. Sturgess, K. Alahari, L. Ladicky, and P. H. Torr, "Combining appearance and structure from motion features for road scene understanding," in *British Machine Vision Conference*, BMVA, 2009.

[167] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *Internation Conference on Learning Representation*, 2016.

[168] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 2074–2082, Curran Associates, Inc., 2016.

[169] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *European Conference on Computer Vision*, pp. 662–677, Springer, 2016.

[170] M. Denil, B. Shakibi, L. Dinh, N. De Freitas, *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, pp. 2148–2156, 2013.

[171] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations*, 2016.

[172] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International Conference on Machine Learning*, pp. 2285–2294, 2015.

[173] E. Park, J. Ahn, and S. Yoo, "Weighted-entropy-based quantization for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5456–5464, 2017.

[174] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, pp. 1269–1277, 2014.

[175] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[176] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances In Neural Information Processing Systems*, pp. 1379–1387, 2016.

[177] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Advances in Neural Information Processing Systems*, pp. 4857–4867, 2017.

[178] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

[179] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, pp. 598–605, 1990.

[180] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in neural information processing systems*, pp. 164–171, 1993.

[181] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 806–814, 2015.

[182] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, pp. 2074–2082, 2016.

[183] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[184] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[185] S. Dodge and L. Karam, "Can the early human visual system compete with deep neural networks?," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2798–2804, 2017.

[186] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[187] J. Hupé, A. James, B. Payne, S. Lomber, P. Girard, and J. Bullier, "Cortical feedback improves discrimination between figure and background by v1, v2 and v3 neurons," *Nature*, vol. 394, no. 6695, p. 784, 1998.

[188] J. K. Tsotsos, A. J. Rodríguez-Sánchez, A. L. Rothenstein, and E. Simine, "The different stages of visual recognition need different attentional binding strategies," *Brain research*, vol. 1225, pp. 119–132, 2008.

[189] D. Varga, A. Csiszárik, and Z. Zombori, "Gradient regularization improves accuracy of discriminative models," *arXiv preprint arXiv:1712.09936*, 2017.

[190] X. Sun, X. Ren, S. Ma, and H. Wang, "meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting," *arXiv preprint arXiv:1706.06197*, 2017.

[191] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, pp. 5975–5985, 2018.

Appendix A

Supplementary Materials

A.1 Implementation Details of STNet

In this section, we provide the implementation details of STNet for the three Convolutional Neural Network (ConvNet) architectures: AlexNet, VGGNet, and GoogleNet. We discuss the realization of the TD selective process for different types of layers. We discuss the experimental results in Sec. A.1.3.

A.1.1 STNet Implementation for Different Types of Layers

We provide details on the implementation of STNet for various types of layers encountered in the three ConvNet architectures.

Max Pooling Layer: The Max Pooling layer could be regarded as a BU Winner-Take-All (WTA) computation where the maximum node activity is selected. Since the gating flow of the BU information is defined in a hard manner, it would be against the inherent nature of the learned representation to select other nodes but the maximum one in the TD processing stream. Therefore, we decide to stick to the the maximum node selection regime and propagate the top gating node activity to gating node correspondence of the maximum node within it's receptive field (RF).

Average Pooling Layer: This type is only encountered in GoogleNet where the convolutional lower part of the network meets the fully-connected (FC) upper part. In other words, the last spatially-ordered hidden layer of the network is squeezed into the hidden vector of the first FC layer using an average pooling layer. We experimentally evaluated what would be the best way of treating the average pooling acting as the link between the lower body and upper body of the network. We decided to choose WTA as the mechanism to select the gating node at the layer below which the top gating node activity will be propagated. It should be noted that in both AlexNet and VGGNet, we defined the concept of the bridge layer at which the lower convolutional body is connected to the upper FC body of the network. However, in GoogleNet, the average pooling layer instead of a FC layer is utilized to connect the lower to the upper. Therefore, GoogleNet does not benefit from the additional level of selection specifically defined for the bridge layer. We experimentally observed that the average pooling layer in GoogleNet is very sensitive to changes in the selection process of the TD processing stream.

ReLU Layer: Since ReLU layers only cut off all the negative activities of the BU processing stream, the TD processing stream simply bypasses the layer and copies the gating node activities of the top layer to the layer below.

Convolutional/Fully-connected Layer: These two types are very much detailed in the main paper. Three stages of the attentive selection process are defined to deal with these two layers. TD processing is implicitly applied to the parts of the visual representation where feature transformation is parametrized such as convolutional layers. It is noteworthy to indicate that in GoogleNet, 1x1 convolutional layers are very dominant throughout the visual hierarchy. Based on the results obtained in the cross-validation stage, we decided to treat such layers the same as we do the FC layers. The sole discrepancy is that at the 2nd stage of the attention selection, all the winner nodes marked by the 1st stage are selected instead of utilizing the SI selection mode in the FC layers. This implies that despite 1x1 convolutional layers do not strive for spatial correlation encoding among their receptive fields, maximal selection of of the nodes in their flat receptive fields provide a significantly better localization result.

Local Response Normalization (LRN) Layer: This layer simply normalize the information flow of the BU processing from the layer below to the top layer over some RF. Therefore, it is straightforward to skip LRN layers in the TD processing by transferring the top gating node activities to the layer below.

A.1.2 Generation of Class Hypothesis Maps

We provide details on the procedure proposed to create Class Hypothesis (CH) maps. Following a similar experimental setup to the localization task, the attention map is extracted from a gating layer. Pixel values in the attention map happen to be very sparsely non-zero. We create the CH map with an equal number of pixels to the attention map filled with zero values. Then, we propose an updating procedure that is iteratively applied to all the non-zero pixels of the attention map as follows. On the CH map, we increment the values of all the pixels falling within the square window centered at the pixel corresponding to a non-zero pixel on the attention map. The size of the window is set to the accumulated RF size of the particular layer the attention map is extracted from. Once all the non-zero pixels on the attention map are visited, the CH map is filtered using a smoothing Gaussian kernel with a standard deviation $\sigma = 6$. Finally, the CH map is visualized as a heat map with the red color representing the maximum value and blue the minimum. In what follows, we provide further details on the modified configurations of STNet for two CH visualization experiments: 1- Context Interference, 2- Correlated Accompanying Objects.

Context Interference: In this experiment, we attempt to highlight the role of the context inference in the localization performance of the TD processing according to the learned representation. The second stage of the selection process in STNet is proposed to tackle this level of contextual noise. Therefore, to show it's efficiency to address the problem, we deactivated the second stage throughout the TD structure. Furthermore, the first stage on the FC layers are modified to implement the WTA mechanism. Consequently, at each FC layer in this regime, there is only one gating node active and the rest remain inactive. This is seen to emphasize the role of the second stage in dealing with the context inference problem.

Correlated Accompanying Objects: We keep the modified version of the first stage for FC layers in this experimental setup, while the second stage on the convolutional layers are taken back into place. The goal is to show that the most confident high-level node at each FC layer will end up localizing a correlated object very frequently accompanying the ground truth category.

A.1.3 Experimental Results

In this section, we provide the high resolution qualitative results of successful bounding box predictions, unsuccessful bounding box predictions, CH visualization using the original STNet, CH visualization for the Context Inference experiment, and CH visualization for the Correlated Accompanying Objects experiment in Fig. A.1, A.2, A.3, A.4, and A.5 respectively. Following a naming convention, ST-VGGNet, for instance, is referred to as STNet with the utilization of VGGNet in the BU processing stream.

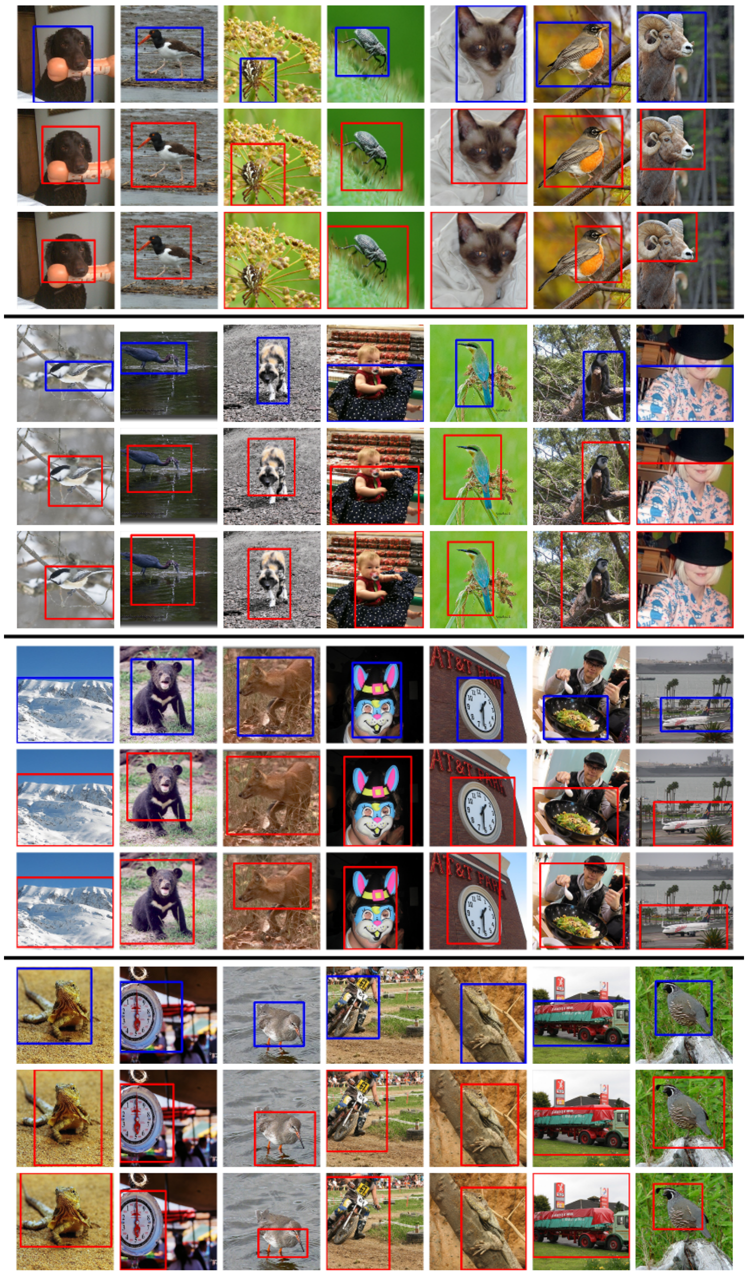


Figure A.1: Illustration of STNet localization performance for both VGGNet and GoogleNet. The top, middle, and bottom row of each section contains images demonstrating the ground truth bounding boxes, bounding box predictions of ST-VGGNet, and ST-GoogleNet respectively.

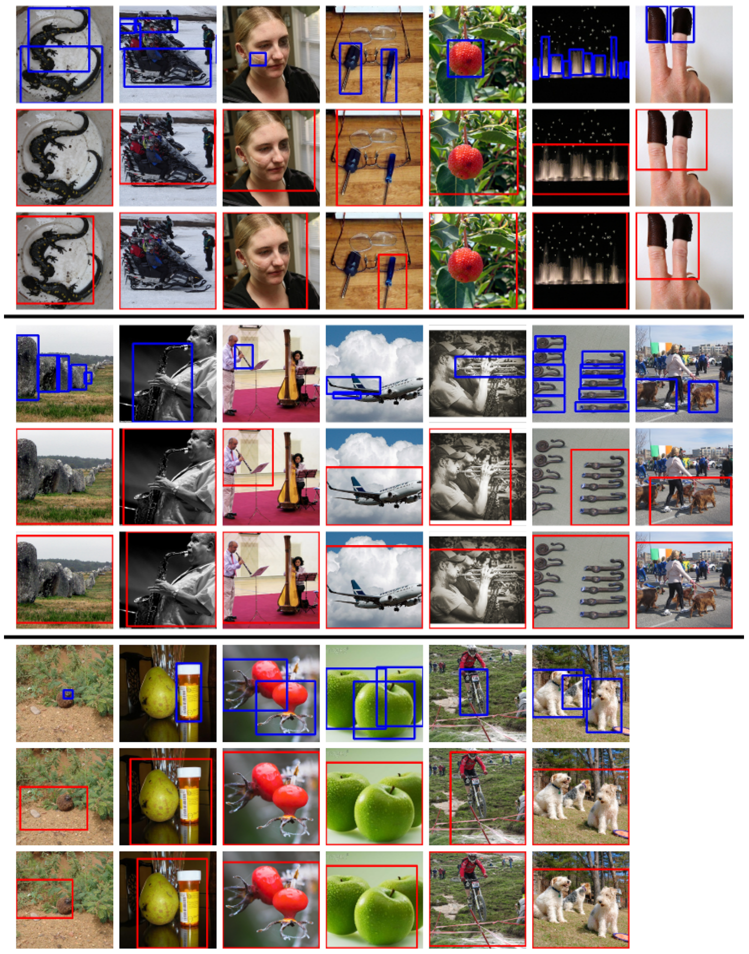


Figure A.2: Unsuccessful localization cases based on STNet bounding box predictions are demonstrated. Multi-Instance and Correlated Accompanying Object scenarios are the two main sources of STNet unsuccessful localization. Each section contains image rows for ground truth, ST-VGGNet and ST-GoogleNet bounding boxes from top to bottom.

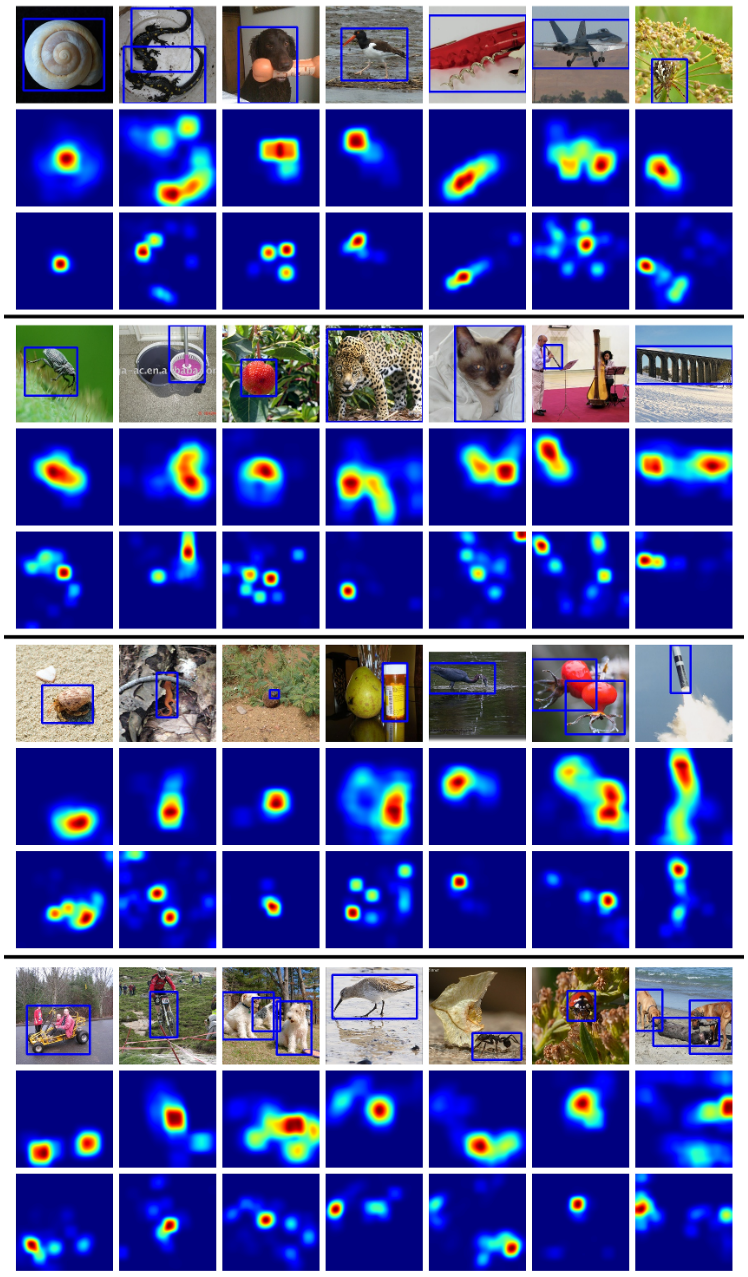


Figure A.3: Class Hypothesis Visualization using STNet. In each section, the top row contains RGB images depicting ground truth bounding boxes, and the middle and bottom row contains the CH maps from ST-VGGNet and ST-GoogleNet respectively.

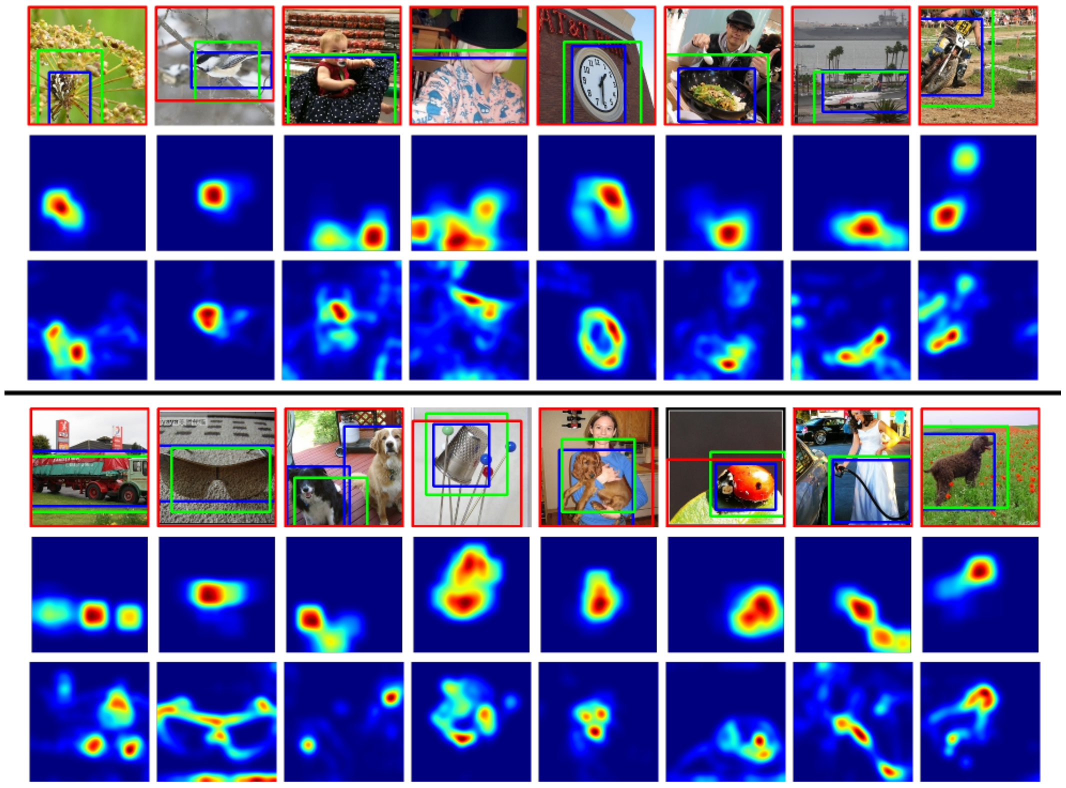


Figure A.4: The effect of the context inference imposed by the learned representation is illustrated in the CH maps given in the bottom rows of each section. The middle row contains the CH maps from the original proposal of ST-VGGNet. The top row provides RGB images with the color-coded bounding boxes. Blue boxes are taken from the ground truth. Green and red boxes represent original and partially-deactivated ST-VGGNet predictions.

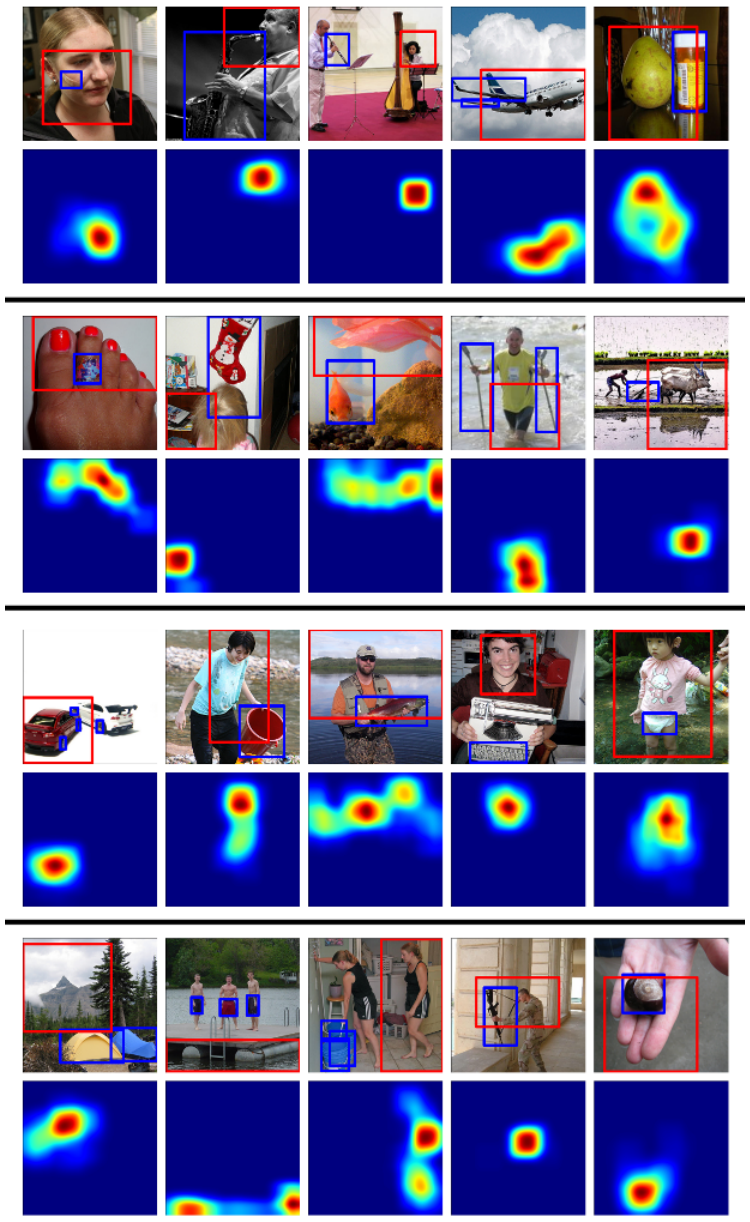


Figure A.5: Correlated accompanying objects prioritize localization of regions outside the ground truth according to the learned representation. In each section, the top row contains RGB images with the ground truth boxes (blue). The red boxes are proposed by the modified ST-VGGNet.

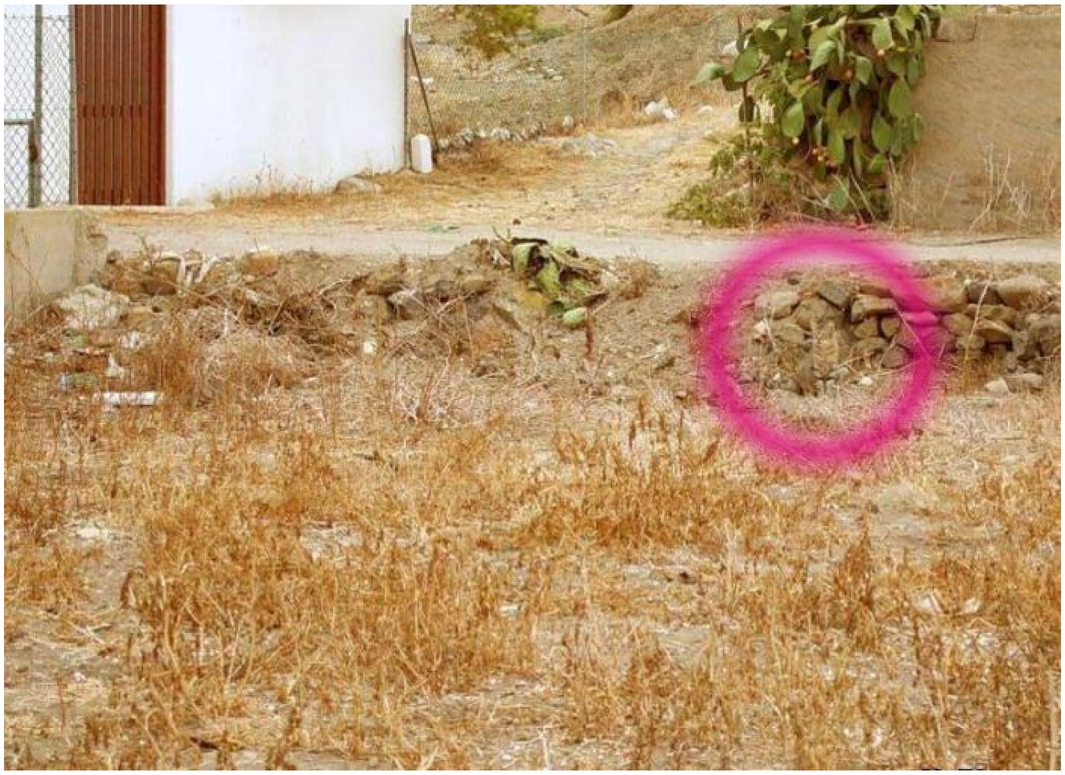


Figure A.6: The location of the cat from Fig. 4.1 in Chapter 4

A.2 Additional Priming Examples

Figure A.6 reveals the location of the cat in the image originally shown in Fig. 4.1.

We further provide some additional results of the proposed method.

Figure A.7 contains some additional results with a primed vs unprimed object detection network (in this case, [5]) in images of varying levels of noise. In many cases, the primed network (blue boxes) remains robust to high levels of noise (lower images in each block) and lowering the confidence threshold does not introduce false detections which are introduced by doing so for the unprimed network.

Figure A.8 contains some additional images of priming the DeepLab [4] segmentation framework. The figure shows groups of four images, with the input image, ground-truth, result of unprimed network and result of primed network.

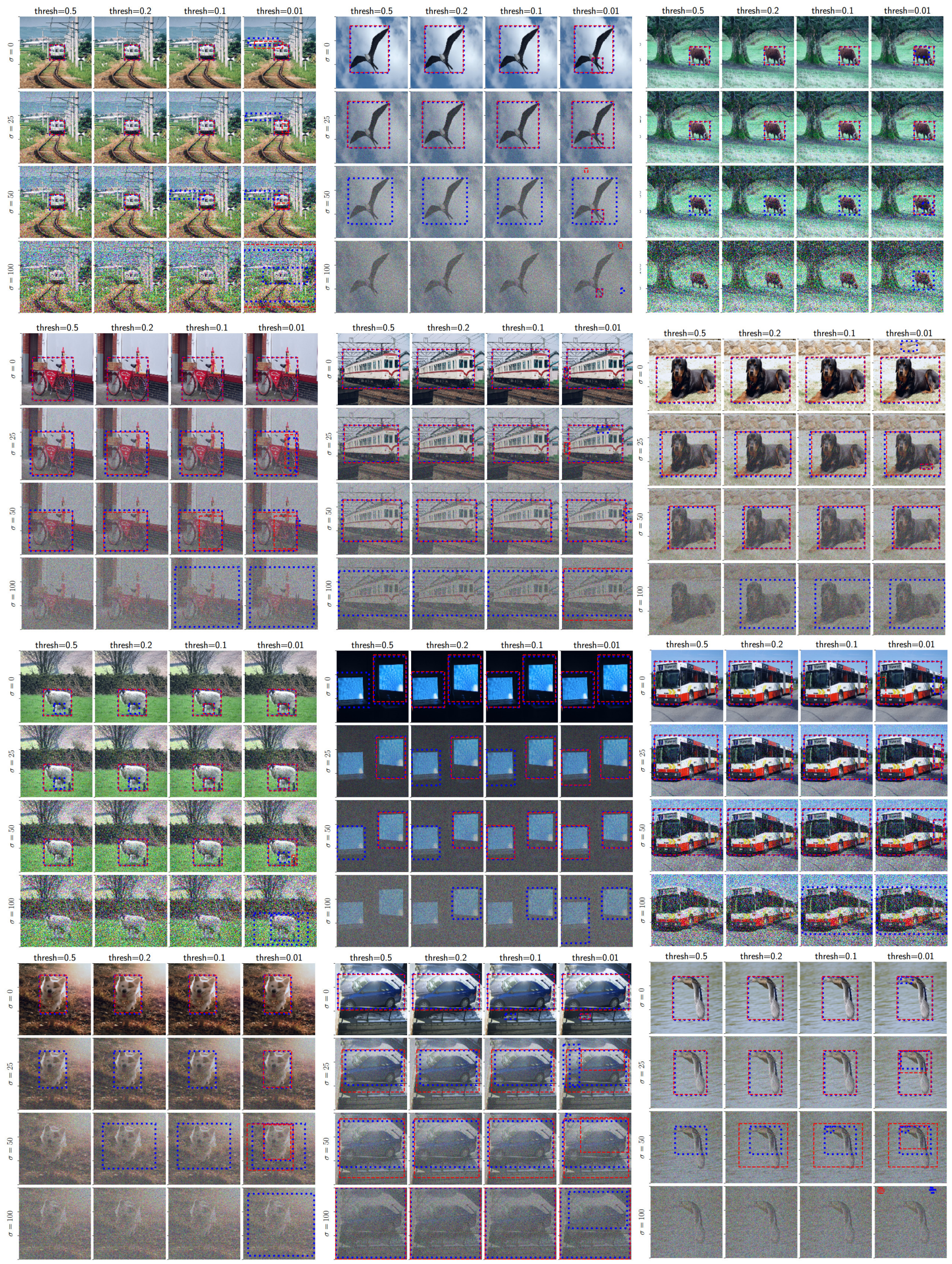


Figure A.7: Additional effects of priming with the SSD [5] object detection network. Each 4x4 block of images shows the detections of the unprimed-network in red and of the primed network in blue. From left to right, the detection threshold is decreased, allowing less confident score to appear, while also surfacing false alarms. From top to bottom, the level of noise increases. A primed network detects objects in noisy image more robustly than an unprimed one.

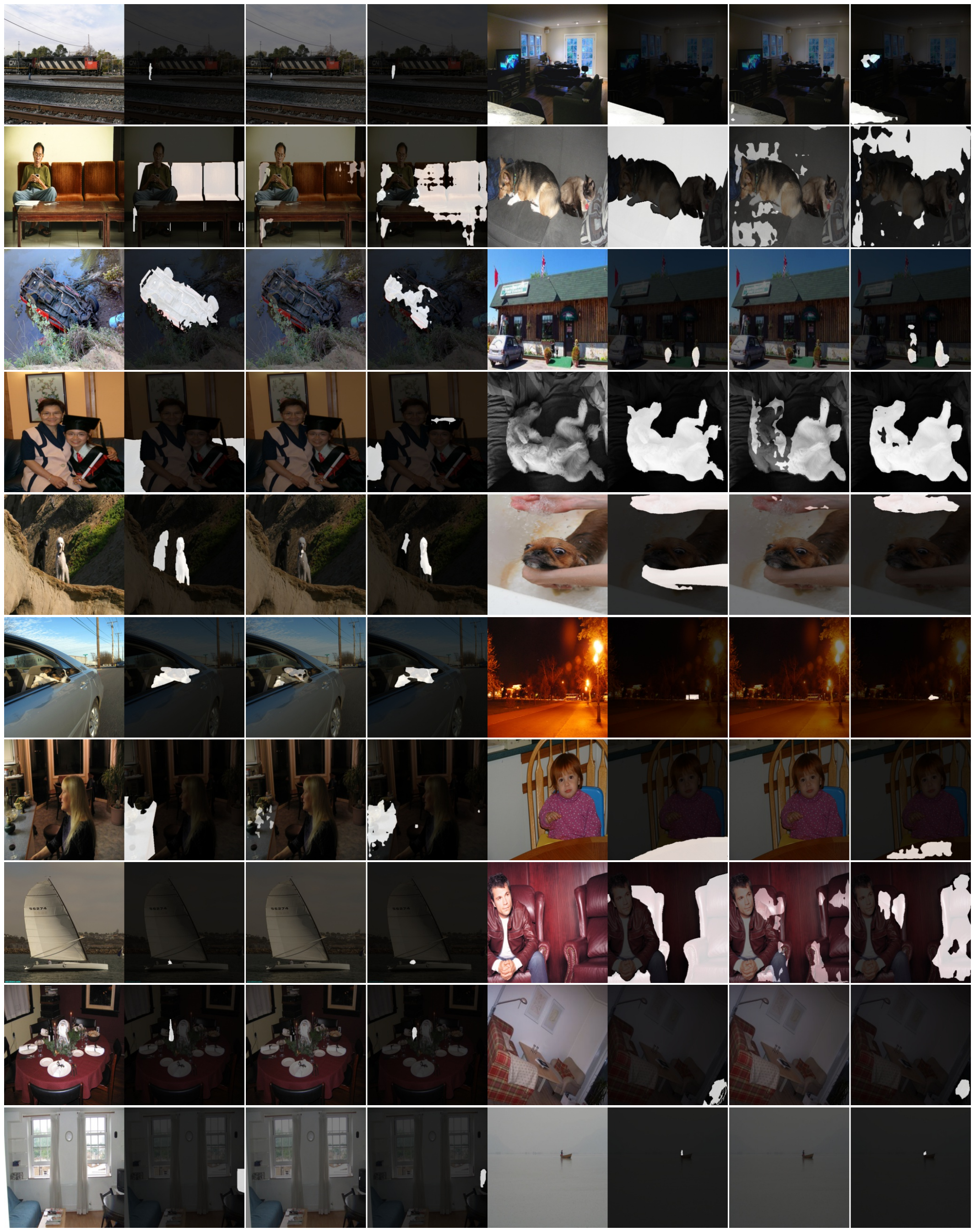


Figure A.8: Additional effects of Priming with the deeplab[4] segmentation network. Each four columns shows from left to right: input image, ground truth segmentation of a specific class, result of unprimed network, result of primed network using proposed method.