THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

IMPROVING OBJECT RECOGNITION PERFORMANCE THROUGH SEMANTIC CONTEXT EXTRACTION

BRIGID SMITH
SPRING 2015

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Engineering
with honors in Computer Engineering

Reviewed and approved* by the following:

Vijaykrishnan Narayanan
Professor of Computer Science and Engineering
Thesis Supervisor

Lee Coraor
Associate Professor of Computer Science and Engineering
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

Machine vision is a computationally expensive problem with an exceptionally large number of real-world applications. With the rise of the Internet of Things and the presence of wearables in day to day settings, there is an additional focus on power constraints and the limitations of fixed hardware. In a vision pipeline, the accuracy of the object classification stage will likely affect the usefulness of the pipeline as a whole. However, we find that it is difficult to create a system with the ability to recognize a large number of objects both quickly and accurately because the number of classifiers needed grows with the number of objects. We observe that real world images and the objects in them tend to be sensible and expose relationships between objects and scenes that are used by humans intuitively. This high-level context could potentially be used to inform and improve object classification by allowing us to make reasonable, probabilistic guesses about objects that might occur based on other information that we have about the image. This guesswork will lower the number of classifiers that need to be run, which will also address power and timing concerns. In this paper, we explore the meaning of context, design a framework to store it in a way accessible to a computer, and then evaluate the efficacy of context-based filtering.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

In recent years, computers have become powerful enough to begin tackling the immense amount of processing required to simulate the workings of the human brain. Specifically, advancements in neuroscience that have given us a clearer picture of how the human visual cortex works, alongside this computer power increase, have together created the setting necessary for us to lay the groundwork for a vision framework in silicon approximating that of the human brain. The applications for this are nearly endless; some forms of machine vision are already so standard as to be invisible. Consider a barcode scanner or handwriting-to-text recognition. If a computer could "see" as well as a human can, this could automate and revolutionize any number of fields.

One specific field in which this would be very efficacious is that of wearables. In recent years, there has been a push towards the so called "Internet of Things", in which we see manifold small devices networked together, many of which are sensors and wearables. Already, we see that devices like Google Glass can provide some rudimentary machine vision support. If such a device could more closely mimic the full range of abilities displayed by the human visual cortex, it is possible to imagine such a device functioning as the user's "eyes." This sort of advancement may someday lead to a prosthetic eye for the visually impaired. Within the Microsystems Design Lab, we explore this particular vision as part of the NSF-sponsored Visual Cortex on

Silicon Expedition, and the particular concerns associated with a low-power, fixed-hardware platform inform many of our design decisions.

One such concern is power consumption, especially in the context of small, low-power devices like wearables. If a user can only depend on their "visual prosthesis" for two or three hours at a time, would the vision it affords really be worth the inconvenience? In addition, the specific processing involved in machine vision is inherently computationally intensive, depending largely in part on operations on large matrices. The mammalian visual cortex is organized hierarchically and operates in various stages. Figure 1 is a diagram of our understanding of the computational requirements of this pipeline within the lab. Some portions of this pipeline are distinctly bio-inspired, whereas others represent a higher level interpretation of neural function.

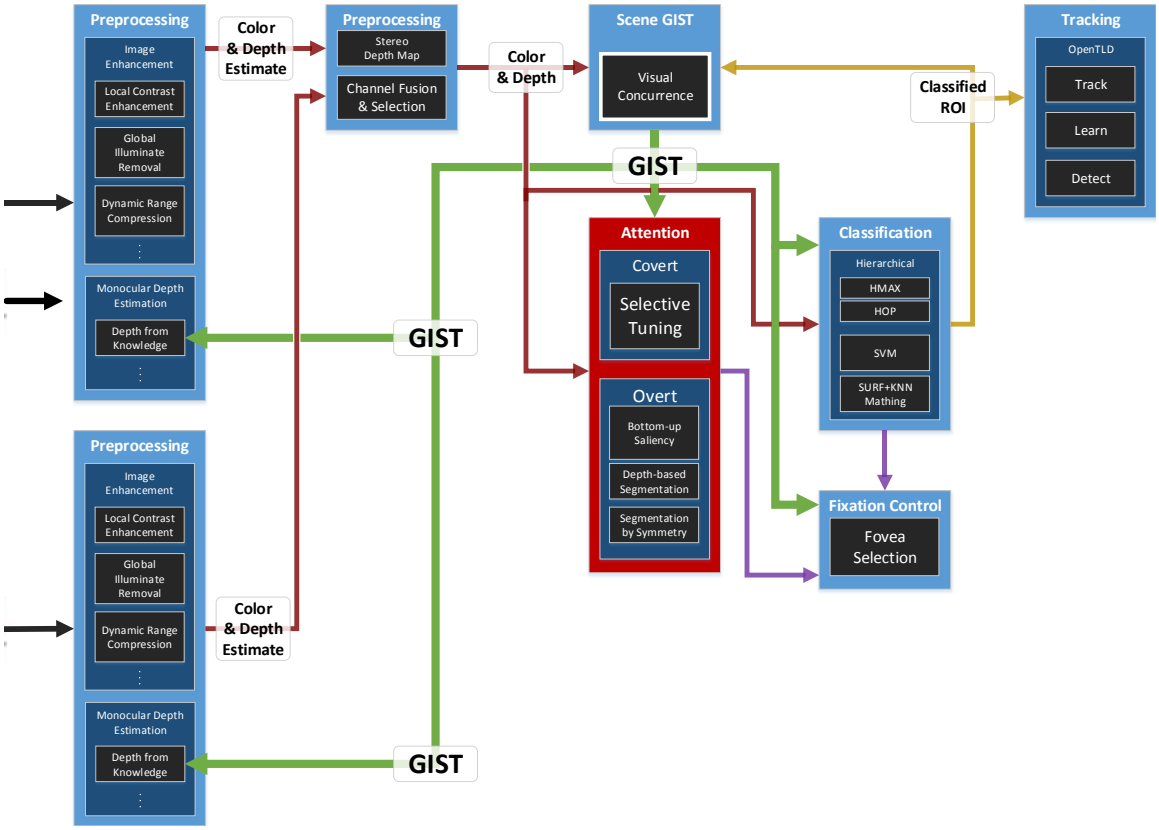


Figure 1: Image Processing Pipeline

In this paper, we focus specifically on the demands of the "Classification" stage. The input to this part is a series of Regions of Interest (ROIs), which are portions of the image being observed that we believe are likely to contain identifiable objects based on other algorithms like saliency that are run earlier in the pipeline. The desired output is a label for each ROI. This stage is understandably extremely important to the success of the pipeline as a whole. We explore a way of looking at high level semantic context information available within an image to make this stage faster and more accurate while additionally decreasing power demands. In Chapter 2, we further discuss what is meant by "context" and explore related works that address this issue in various ways. In Chapter 3, we formulate our own framework for addressing this

problem by considering a variety of design questions. In Chapters 4 and 5, we evaluate our framework on a pair of real world image datasets. Finally, in Chapter 6, we conclude and discuss some next steps that could improve our framework.

Chapter 2

Why Context?

As we begin to explore this problem in greater detail, it is first important to establish what we mean by context. This may be easiest to explore by defining what information from an image we would not consider context, namely the extracted features of an object we wish to identify. A key concept in machine learning as a whole and therefore computer vision as well is the choice of features to extract from an image. In the context of this thesis, the specific representation of the features is open-ended, though in Chapters 4 and 5 we explore a specific implementation that uses HOG vectors. However, in this general discussion, features can be thought of to mean our chosen representation of the object to be identified.

What, then, is context? As previously stated, we consider context to be any information about the scene that we may be able to acquire beyond the specific features of the object to be classified. This can be defined on the same level as the features, such as HOG vectors for the image as a whole, or could refer instead of higher-level information about the scene or other objects in the image or in related images. Context is therefore a very loosely defined idea that can take a variety of forms. Roughly speaking, features will tend to refer to the object to be classified and context will refer to any other information that we can relate to this object, including, but not limited to, other objects in the scene, the type of scene, or information about previous scenes that are somehow related, as in temporal information between adjacent video frames. In this section, we explore some other work related to various forms of content extraction and discuss their usefulness.

GIST

One well known algorithm in this field is GIST content extraction, detailed in two papers with slightly different implementations. [1] [2] The idea of GIST is a form of high-level scene classification based on features of an image as a whole. A key idea here is that we observe objects of interest within an image as part of a larger scene. Both implementations of GIST use filters on different characteristics of the image to extract features from a scene as a whole, similarly to how one might extract features for a region of interest on which identification should be performed. From this, a GIST vector is calculated that represents the scene. In [1], the specific goal is to explore the accuracy of scene recognition between three outdoor locations. [2] seeks instead to identify both broad categories of location, such as "outdoors", as well as more specific ones, such as specific streets or parks. Both show very highly reliable classification results. In addition, [2] performs experiments on unseen locations that might fit into a larger category, which shows an understandable loss of accuracy. Finally, the authors of [2] connect this idea of scene classification to the probability of object presence with the understanding that there is an exposable relationship between scene and the presence and location of objects within. The idea of GIST as a whole is a powerful one: it confirms that real-world datasets have logical relationships that we can exploit. To be able to identify the type of scene will inevitably affect the likelihood of the different objects that we expect to see, as seen in [2]. This is perhaps an obvious observation to a human being, but the ability to quantify this type of knowledge, especially from raw, extracted features of a single image, is crucially important to our understanding of context.

Visual Words

Within images are areas known as key points that can be reliably extracted. These key points are highly salient regions that contain useful identifying information about an image as a whole. Such points are often located around the edges of important objects within the image. The authors of [3] seek to use the presence and frequency of appearance of these key points in a context that mirrors text classification. An image can be described by the key points that it contains, which is known as a "bag of visual words" representation. Using this representation, the goal is to be able to classify image types by the similarity in appearance of specific patterns within the image and predict labels for previously unseen images. This method is already well-established as effective on text documents.

[3] explores a variety of experimental metrics to consider while creating such a system and evaluates the impact of specifics such as the size of the vocabulary, the removal or lack thereof of "stop words", and the granularity of spatial relationship information within images. The results are varied and show that the accuracy is very dependent on the values chosen for said variables, as well as the dataset itself. However, the idea highlights again the fact that context can be extracted from an image in a variety of ways. Specifically, we see that meaningful context can be extracted from the frequency of salient patterns within an image, which could translate to the frequency of objects but more generally simply means regions with similar extracted patterns. This is similar to the GIST approach but with the additional "zooming in" on the important regions of an image that is derived from saliency.

The authors of [4] make use of the same sort of visual words representation of an image but with the intention of retrieving the frames of a video in which a particular object exists. A

"vocabulary" of visual words is built, as in [3], but [4] differs in that it is focused on frames of a video so additional information about visual word saliency can be computed based on the differences between objects in contiguous frames. The overall goal is to be able to identify the same scene from different angles, and to then be able to search for an object that appears in a given frame and find those other frames in which it is present without false negatives from the differences in perspective.

A key detail in [4] is the observation that a temporal relationship that can be exposed between frames of a video, which can be loosely extended as well to images that are known to be related. A great deal of additional information can be extracted in a set of images when they are known to be related in time. This additional dimension establishes much more complex relationships between objects than could be exposed from singular, independent images.

Generating Sentences from Image Features

In [5], the authors explore a leap from simple object recognition to the derivation of meaning from detected objects. In the same way that a human might simply summarize an image by describing the principal action and scene, the authors of [5] seek to algorithmically extract simple sentence representations of images based on detection results. It would, for example, be natural to describe an image as something like "a man sits on a bench in a city" or "a penguin swimming in the ocean", omitting many other details within the image and summarizing the central focus. This same simple description could also be used to identify images in many cases without other details. [5] explores a simple triplet sentence structure that describes an image as a combination of object, action, and scene, and attempt to predict

sentences based on images and vice-versa. The relationships between the components of the triplets are weighted beforehand based on images from Flickr to establish the connection between different scenes, objects, and actions. A key detail here is the understanding that a certain scene, object, or action will influence the likelihood of the value of the other two nodes in the triplet.

[5]'s results are not necessarily stunning in their accuracy, but its authors suggest that a deeper knowledge of object relationships and sentence structure is required and should be explored to deepen the understanding of the principal actions taking place in an image. Overall, the concept that such a concise description of an input image might be extracted with little to no other information is a powerful idea, and the correct annotation of an image in this sense could provide us with additional information about objects that might be in a scene.

Our Approach

In this thesis, as previously mentioned, we focus on high-level semantic context. In a way, this idea encapsulates each of the ideas explored by the other papers just discussed. This semantic context specifically refers to the sort of relationships between objects that a human recognizes intuitively. Consider a simple example: imagine that you have walked into a classroom with the lights off. You can see the outline of an object, but cannot see any of its features clearly. Depending on the size of the object, and with the knowledge that you are in a classroom, there are a list of candidate objects that you might consider, including perhaps a computer, a backpack, a chair, etc. This list of objects would change if you were in, for example, a garage or a movie theater. The probabilistic weighting that a human does innately in a

situation like this stems from a higher level understanding of the relationships between objects and scenes which cannot be extracted from a single image. To extend the previous example, if the scene was a stable and you had never been to a stable before nor did you know anything about horse care, your list of candidate objects would be much less informed and potentially may not contain the mystery object.

This specific sort of object relationship context was used in [5] to weight the relationships of objects, scenes, and actions in triplets. In addition, [2] explored the relationship between an identified scene and the objects that might be present within. However, each paper examines these object relationships in a cursory fashion that, while useful, could be extended. In neither paper do we see the actual derivation or representation of these relationships as the main focus of the work. The relationships between objects are extremely complex, and we believe that the first step in reliably utilizing them is to expose them more thoroughly. Consider the complexity of our understanding of "part of" relationships, categorical grouping of objects, synonyms, and the different relationships two objects might exhibit when seen in different context. Our goal in this thesis is to discover which of these manifold types of relationships are most relevant to object classification and then to find a way to "teach" a computer these relationships that we humans learn and use without thinking every day. This goal requires us to consider several questions. What contextual relationships are important? How can we encapsulate them into data structures? How should a computer interact with the resulting context information? In the next chapter, we look into these questions in more detail.

Chapter 3

Methodology and Design Decisions

In this chapter, we will consider what sort of information informs our decisions when creating a framework for image context. Our goal is to identify three things: what information is important, how this can be translated into data structures, and how we should design access to the framework. This is a wide-open question, as the amount and types of context that might be extracted somehow from an image is quite large, and for a first exploration, flexibility to add more context is important. These design decisions are focused on creating an immediately realizable system that can be tested in a simple situation before delving too deeply into more types of context or relationships, and as such, the discussion in this chapter is informed by practicality more so that extensive brainstorming. In Chapter 6, we revisit these ideas with a focus on brainstorming future work.

What context is important?

This question is perhaps the most interesting and open ended question of this idea as a whole and will have the strongest impact on the rest of the implementation. As such, our goal was to remain open-minded about this and understand that the answer to this may not be available until we experiment. However, the starting point for our work was based on the idea that, for any given object, there are other objects that we might consider to be "related" to it. A towel, for example, might suggest beach-related items like beach balls or coconuts. It could also

suggest bathroom items like a washcloth or shampoo, which immediately gives rise to the importance of context couched in the knowledge of the scene as a whole. For simplicity's sake, we begin our exploration assuming that our scene will be constant and explore the idea of scene-by-scene relationships more thoroughly in Chapter 6. The granularity of what might be considered a "scene" is also variable, but here we will focus on a scene as a location in which we can reliably decide if an object is likely to be present or absent and if two objects are related. For our purposes, then, "house" would be too general, as object relationships vary too greatly between rooms, but "kitchen" or "living room" would represent an acceptable scene.

What, then, are we considering to be the important relationships between objects in a known scene? For our purposes, this is fairly straightforward: we want to know what objects are likely to be present in a given scene, and then of those objects, which are likely to be present near one another. Since we are considering a single setting, the first can simply be considered the whole set of objects that we are able to classify. The second becomes important as we consider our yet-unidentified ROIs within an image. Perhaps we know nothing about our first ROI in an image, but once we have performed a single classification brute-force, we should be able to look up what objects might be seen near that one and look for them first before those that are less likely. This could be seen as a simple true or false mapping in which we consider objects to be related or not to be related, or we could take a more complicated tack in which we weight such relationships with likelihoods or counts so that we could extract not just a list of the related items, but a ranked list of the top *n* related items by probability. In this same way, it might be useful to apply the same sort of probability to items in the scene as a whole, as there are certainly items that are more likely to appear than others in a given scene in general.

How can we represent this in data structures?

To represent discrete objects and their multifarious, potentially hierarchical relationships, the most obvious data structure that comes to mind is a hypergraph in which each node represents an object for which we have a classifier, and each edge is a type of relationship. As previously stated, we are looking for simplicity in this first implementation, so hierarchical relationships may not come into play immediately, but we would like them to be available for representation in the future. However, at this moment, our hypergraph will likely take the form of a regular graph while we are not worrying about more complex object relationships.

A key detail as we think about implementation is to consider the asymmetric nature of existing datasets and allow for objects to have relationships defined at various levels of complexity. For example, one can imagine a dataset in which we are only given an undirected graph of relationships with only a true-false mapping of nodes to one another. In contrast another dataset might provide detail on the relationships, such as spatial relationships, scene information, or weighted relationships. These two granularities of specificity should both be representable in our implementation, so the implementation we choose should have the ability to represent these varied edge characteristics.

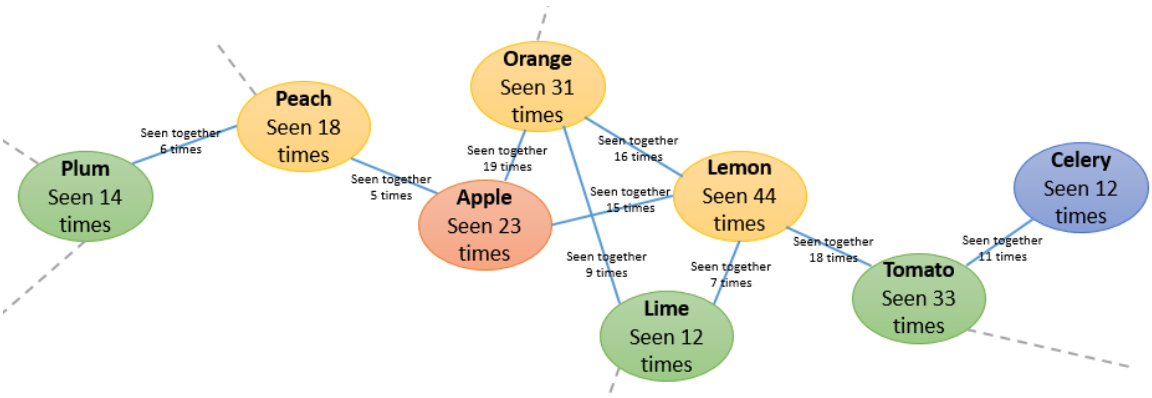


Figure 2: A Simple ViCoNet Example

Ultimately, the data structure that we envision is a graph for a given scene where each node is an object that we have a classifier for and each edge represents some likelihood of the connected nodes being seen close together. Additionally, we can envision storing some simple frequency information in the form of counts on the nodes and the edges. A visualization can be seen in Figure 2. We call this graph a Visual Co-Occurrence Network, or ViCoNet, as it attempts to encapsulate our understanding of what objects are likely to occur near one another in images. The use of counts for probabilistic weighting is not ideal, but it is a simple way to begin to quantify the importance of relationships between objects.

How should our visual pipeline interact with ViCoNet?

In thinking about this question, the most important detail is the relationship of input from the vision pipeline to relevant output that will improve image classification. We can consider some simple scenarios in which the pipeline might desire some input from our network. For example, given an image with no classifications yet performed, the pipeline could ask ViCoNet what objects are seen most frequently in general and begin running classifiers based on

frequency. Once some classifications have taken place, the spatial relationships between the objects in ViCoNet may begin to be useful. For example, if we have identified a plate and a fork located close to another unknown ROI, ViCoNet should be able to suggest likely items like a spoon or knife over, for example, a cabinet or window in a kitchen setting.

In considering interactions with ViCoNet, we are assuming a static knowledge graph that is pre-populated with these relationships and purely provides output to the pipeline. However, it is important that ViCoNet additionally be able to learn and dynamically adjust as the world changes around it. Therefore, it is important to keep in mind the idea that feedback from the pipeline as to the correctness of ViCoNet's guesses might be used to reweight relationships over time. For the time being, we will stick with a static implementation of ViCoNet, but the two-way nature of the relationship between ViCoNet and the pipeline's successes and failures is important to keep in mind, and we will look deeper into a dynamic implementation of ViCoNet in Chapter 5.

From these three questions, we have begun to flesh out an implementable form for ViCoNet. Now, with a framework in place, we can perform evaluation of its usefulness in the context of the rest of the classification stage of our pipeline.

Chapter 4

Context Evaluated in a Retail Environment

Now that we have explored the applications and design decisions that go into storing and accessing this context, we implement the general ideas discussed in Chapter 3 and evaluate the efficacy of context in real image recognition. This experiment was also published as [6].

Retail as an Environment for Evaluation

A critical decision in evaluating the usefulness of context is to choose a setting in which context is likely to provide us with useful information. As explored above, this holds true in most real life settings. However, different environments vary in the complexity of said context. A retail environment is structured in a logical way so that shoppers can easily find the items they seek. This hierarchy works on more than one level. In a grocery store, for example, the store as a whole is divided into aisles. Each aisle is sorted so that items of the same time, at the granularity of "olive oil" or "aluminum foil" tend to be located in the same contiguous area. These areas are further ordered by brand, and the brands are ordered by specific product type. For this reason, a grocery store can be said to have a great deal of information—on the order of thousands of items on average—organized in a complex, intuitive way that the average shopper navigates almost without thought. This rigid hierarchy is ideal for translation into terms a computer can understand, and additionally tends to hold true across grocery stores in general at least at the granularity of aisle. Some of these relationships may even carry over into farmers' market or kitchen settings.

Another advantage of retail in general is that, with the pervasive nature of advertising, it is generally very easy to find high quality, high resolution images of products. This is important for us when training classifiers, as better training images can help improve accuracy. Additionally, items in a store are arranged on shelves so that they face outward along the plane of the aisle. While obviously this is not a perfect system due to objects being knocked over or other effects of human error, it does on average mean that the objects in an image are likely to be facing the same direction and not be obscured or skewed in ways that might make classification more difficult.

Though a grocery store (or, indeed, any other retail establishment) may be designed in this specific, logical way, this is not meant to imply that such hierarchy does not exist in the rest of the world. As seen in the other papers in Chapter 2, the vast majority of real world images will likely contain usable context, and in Chapter 6 we further explore the idea of multiple scenes and how to manage the varied relationships.

The datasets

Within this grocery environment, we gathered and annotated two sets of images. For training, our local Wegmans grocery store allowed us to gather images of the aisles. There are 14 of these panoramic images, each one representing an aisle. We created a list of 102 classes from objects that appear on the shelves in these images, choosing with a focus at representing each aisle and container type (box, packet, bottle, etc.) as evenly as possible. We used Bing's API to download training images in bulk, so we had to additionally modify our list of classes to

contain only those that produced useful results in this image search API. A table containing the counts of class per aisle is shown below.

Aisle	Number of Items
Cereal	8
Chips	5
Cleaning	8
Coffee	7
Condiments	8
Cookies	13
Dental	5
Juice	6
Pasta	2
Refrigerated	7
Sauce	11
Soda	16
Soup	3
Storage	3
Total	**102**

Table 1: Training Data

Within this training dataset, we annotated a number of instances of each class using the LabelMe toolkit [7], which we subsequently used in the training phase described below.

For testing purposes, we used a separate, disjoint dataset from California of images of supermarket shelves. The item classes above were additionally chosen so that as many as possible appeared in this second dataset. Our hope was that the same relationships that are true in a grocery store in Pennsylvania would also hold for one in California, which was part of our motivation for choosing the testing set. We annotated over one thousand instances of 63 of our 102 classes in these test images.

The image processing pipeline

This takes place in the context of a larger pipeline developed by a number of others in the lab, as the goal is to someday link all of the disjoint pieces together into a functional pipeline, which can be seen in Figure 1. Here, we discuss the other components of object classification and establish ViCoNet's location and behavior within this context.

HMAX

HMAX is a hierarchical, bio-inspired object classification algorithm which consists of 5 layers. First, there is an input layer from which specific portions of an image can be extracted. This is followed by four layers of simple and complex cells meant to mimic the behavior of visual cortex neurons. [8] HMAX has several properties that make it ideal for our system. The algorithm itself is quite complicated and computationally intensive, but can be parallelized and accelerated efficiently, making it a good candidate especially in a fixed-hardware setting. Additionally, HMAX performs well on a variety of different datasets, as can be seen in [9]. A downside to HMAX is that, as the number of classes supported grows, the classification accuracy plummets, as shown in [10]. HMAX on its own is therefore not an ideal choice for classification, but its fairly reliable accuracy, combined with the ability to take the top *k* ranked guesses as opposed to simply the top 1, makes HMAX useful as an efficient pre-filter that we can run on an image before anything more computationally expensive.

With our 102 previously defined classes, we find that we can achieve about 35% accuracy on the top 1 HMAX candidate and 70% accuracy when we choose the top 20

candidates. The accuracy curve for HMAX can be seen in Figure 3. HMAX therefore prunes our results space for a given image down to 20% before any classification takes place, which hearkens back to previously discussed power consumption concerns, ideally allowing us to run fewer classifiers overall.

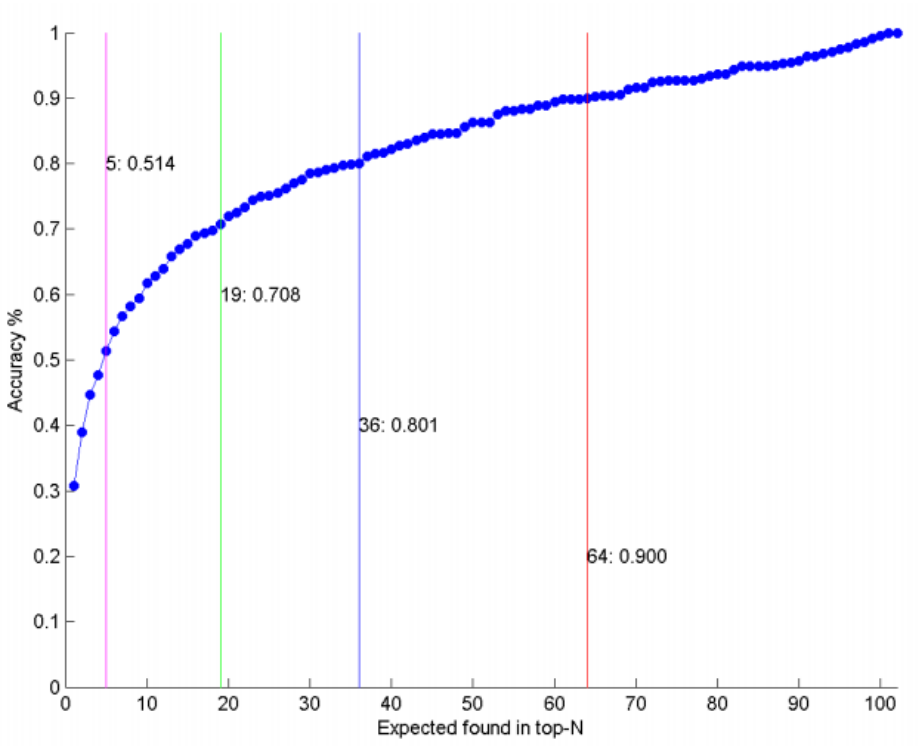


Figure 3: HMAX Accuracy Results

Exemplar SVM

An Exemplar SVM, or ESVM, is an implementation of a traditional Support Vector Machine (SVM) that is trained on a single positive example of a class and many negatives. [11] This means that a given ESVM can identify a specific view of an object with very high accuracy, which is especially well-adapted to a retail environment in which we generally expect to see the front side of an object as it is arranged on a shelf. Each object to be classified requires its own

ESVM to be trained, though, and as the number of classes grows, so too does the number of ESVMs. Without any additional way to make decisions, a purely ESVM detector must run every classifier on every ROI, which is not at all scalable and is prohibitively slow for any realistic use case. Combining the accuracy of ESVM with the pre-filtering of HMAX is explored in [10] and can be shown to improve accuracy while reducing the number of classifiers, and therefore time and energy, needed to use the system. This filtering is agnostic of higher level context, though, and the hope is that the use of ViCoNet can further improve this hierarchical filtering to improve both accuracy and efficiency.

The Visual Co-Occurrence Network

We implemented ViCoNet as a simple graph in C# to attempt to capture the sort of object relationships that we have previously discussed. C#'s object oriented nature allows us to develop a simple graph relationship with the flexibility to later add in additional data items into the edges. Additionally, C# provides an easy-to-use interface to hardware accelerators, which will be important as we move towards a hardware implementation in the future. In Figure 4, ViCoNet's specific relationship to the HMAX-ESVM pipeline can be seen. The high-level goal is for ViCoNet to take the list of $k = 20$ candidates for a given ROI from HMAX, which has already been pruned down from our original 102 classes, and further reduce it based on high level semantic knowledge of our scene. In Figure 4, ViCoNet is given to understand that the candidate image is in a chips aisle and can trim irrelevant items from the list and therefore prevent their ESVMs (greyed out) from being run. However, this does lend itself to the important question of how we understand what aisle we are in, which will be addressed shortly.

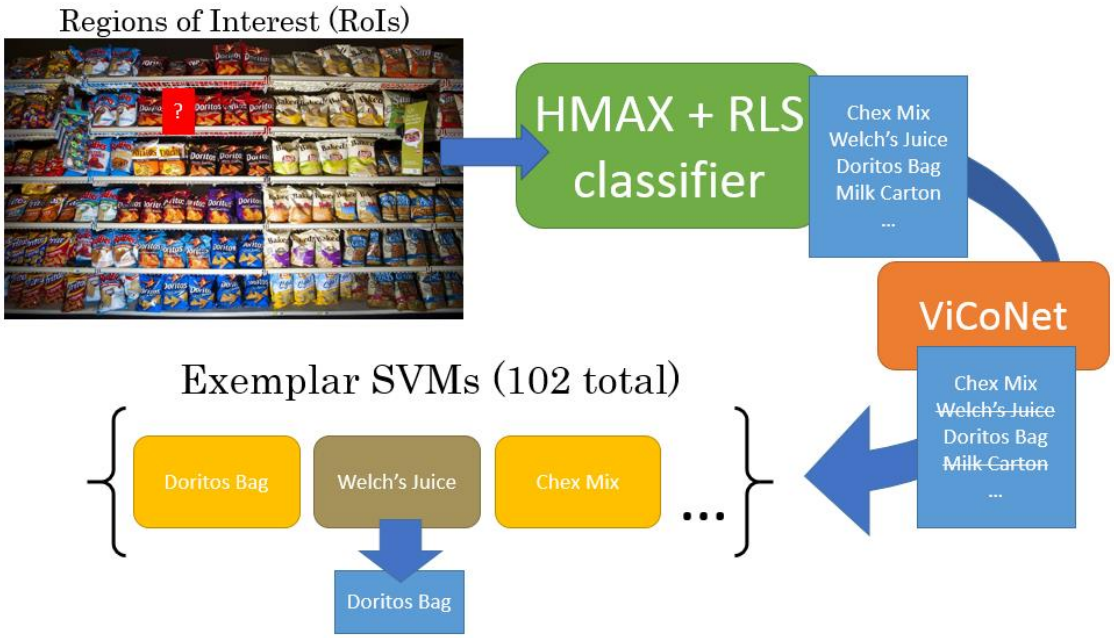


Figure 4: ViCoNet's Location in the Pipeline

Structure

The structure of ViCoNet that we implemented is similar to the visualization in Figure 2. Each ESVM has a node in the graph that associates a count of appearances with the object. We train ViCoNet on the previously detailed Pennsylvania grocery store database of 14 large, panoramic aisle images. Edges are drawn between nodes if they have been seen within 1000 pixels of one another, which corresponds to roughly a few feet with the size of our dataset. This allows us to maintain relationships at a granularity of their location within an aisle, so that objects are neighbors of the objects close to them on the shelf, and related over several hops to all of the other objects further down the aisle. Aisles do not overlap in our dataset, so the overall form of ViCoNet is a collection of disjoint subgraphs.

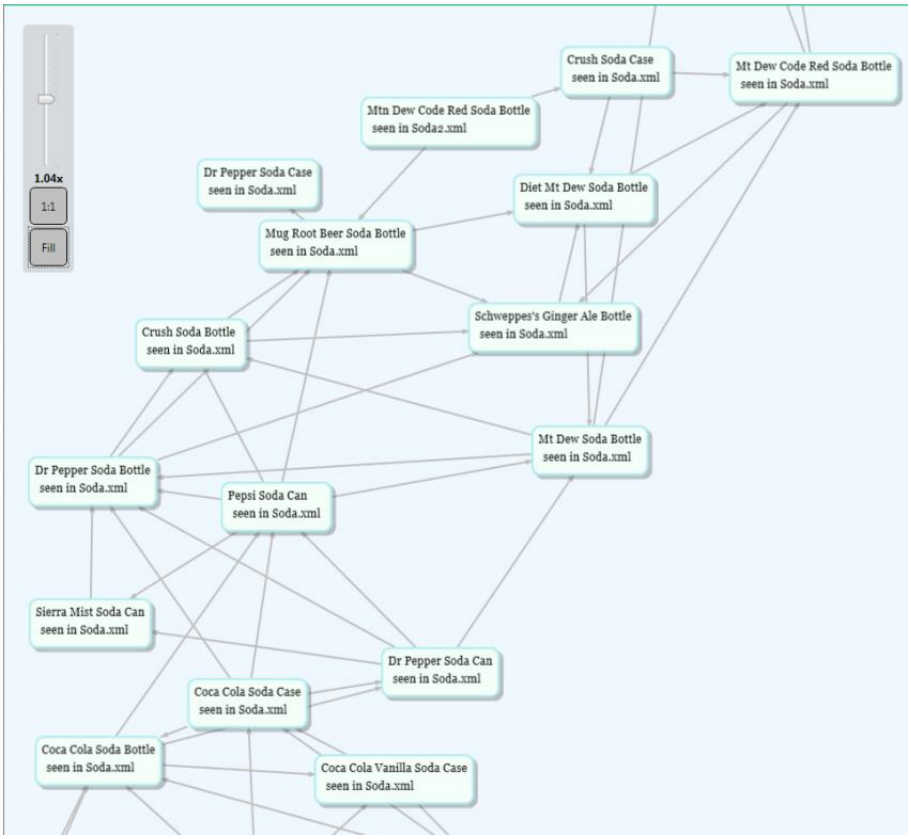


Figure 5: A Subsection of ViCoNet with Aisle Names

Ideally, each aisle would be represented by a single subgraph, but as not every item in our images is annotated, there are situations in which the closest two items in a region of an image are over 1000 pixels apart. We want to maintain an aisle as a contiguous level of categorization even if it is not spatially connected based on our metrics, so nodes are additionally tagged with the name of the annotation file from which they were taken so that it is possible to associate all subgraphs of a given aisle with one another. Figure 5 shows a subsection of our trained ViCoNet corresponding to the soda aisle, generated from the GraphSharp library in C#. A valid question would be how ViCoNet could possibly be trained in the real world and understand that subgraphs are part of the same aisle, and for the time being we assume that this will be possible

using some other method such as GPS or manual user interaction. In the next chapter, we will address this concern more concretely.

Testing

ViCoNet is used within testing to improve the list of $k = 20$ candidates from HMAX by removing items that are unrelated to the current context. However, this requires ViCoNet to discover the aisle in which the detections are taking place. To do this, we perform object classification in two stages: stabilization and pruning. For each aisle, we read in the annotations one after another. Our pipeline is artificially aware of when a new aisle begins. Stabilization takes place on the first 5 ROIs for each aisle. For each of these five objects, the HMAX-ESVM pipeline gives a classification which, as we have seen, represents about 66% accuracy. Our choice of 5 stabilization candidates was with the hope that this accuracy value would give us at least 3 correct values out of the 5. ViCoNet takes these 5 objects and makes an aisle decision based on the most commonly occurring aisle in these classifications. This may also not be the best way to make a decision, but as a first pass at implementation, we tried what seemed to be the most intuitive and simple way to decide. With the aisle decision made, the pruning phase begins for ROI number 6 and onwards. ViCoNet now observes each $k = 20$ candidate list from HMAX and prunes it to contain only objects in the current aisle. Figure 6 shows a visualization of this process.

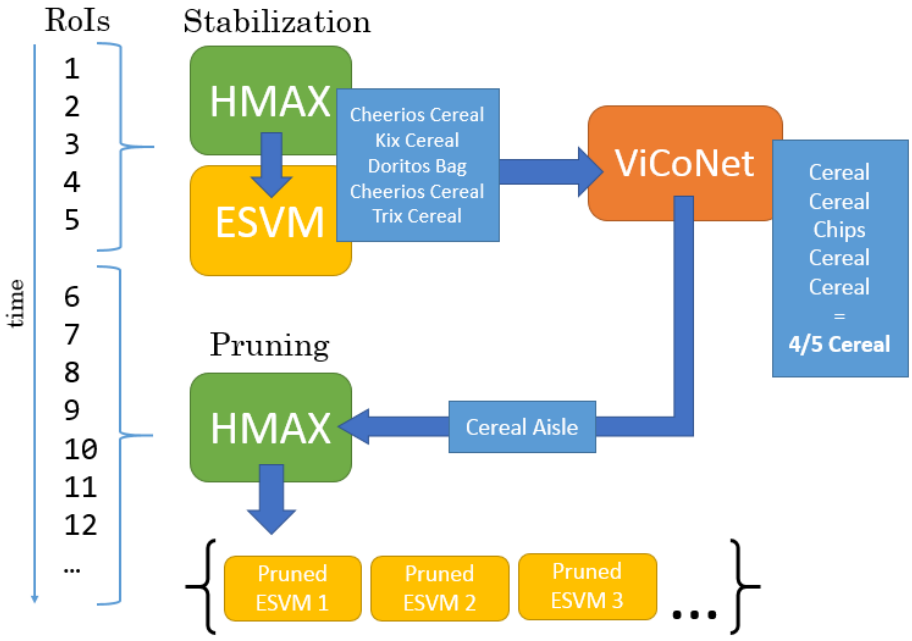


Figure 6: Stabilization and Pruning Stages

Though this two-stage stabilization and pruning system works quite well on average and trims down the number of ESVMs to be run by more than 80%, it also resulted in an accuracy loss of almost 10%. This is due to a phenomenon in which the aisle is mispredicted in the stabilization phase, resulting in 0% accuracy for all items in that aisle. This is because if the correct item is present in HMAX's candidates, it will be pruned out and the corresponding ESVM will not be run regardless. A visualization of this can be seen in Figure 7, where the correct classification for the item (Mug Root Beer) is provided by HMAX but subsequently pruned out with the values in red by ViCoNet during the pruning stage, as it has mispredicted the soda aisle as the cookies aisle. Our first attempt at addressing this was for us to increase the number of ROIs used to predict the aisle from 5 to 7, 9, and finally 11, with the hope that a larger number of potential decision candidates would yield better accuracy for the group overall. We found that, for all of them but 11, our aisle prediction accuracy did not increase. At 11, we were

able to guess only one additional aisle correctly, and it was really only because by chance the next two items happened to balance the guessing in our favor. Additionally, taking more ROIs for stabilization inevitably makes the whole stage slower and would affect the end user who might be waiting for ViCoNet to stabilize and begin providing them with detections. Clearly, there must be a better solution. An important observation to be made here is that the stabilization guesses for these mispredicted aisles are somewhat random in appearance, and that our particular ESVM implementation always provides a classification, even if none of its detectors provide a score of any confidence. In some situations, the classification is the result of a calculation in which the detector is ultimately performing a max operation on an array of 0 values, and whatever the first value is happens to be chosen.

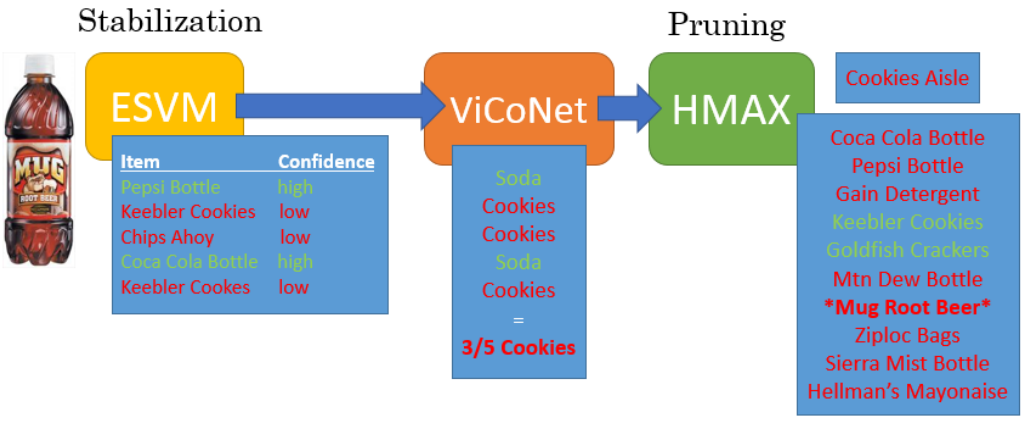


Figure 7: Aisle Misprediction and Faulty Pruning

This misprediction stems from low confidence guesses on the part of the HMAX-ESVM pipeline. To remedy this, we added thresholding to our stabilization phase. Now, instead of taking the first five ROIs, we only take those whose ESVM detection score is above a certain absolute threshold and put any "unknowns" back into the testing queue. We found that this resulted in no aisles being mispredicted. Realistically, we would probably have to just throw out

unknown ROIs, but here we wanted to evaluate the system without much variation between its implementations. One interesting consequence of this is that, for one aisle, there are not enough confident detections to ever stabilize. In that case, we simply run all the detectors corresponding to the *k = 20* from HMAX, so it is essentially returning to the behavior of the HMAX-ESVM pipeline alone.

Thresholding fixed the issues that can occasionally arise from ESVM's imperfect accuracy. However, HMAX can also cause problems with its own lack of accuracy. In some situations, we saw that ViCoNet could correctly guess the aisle, but within the pruning phase the correct item was not present in the k = 20 from HMAX, so even with the correct aisle, we are still not running the correct detector. To counter this, we split the operation of ViCoNet into two different modes: passive and active, which can be seen in Figure 8.

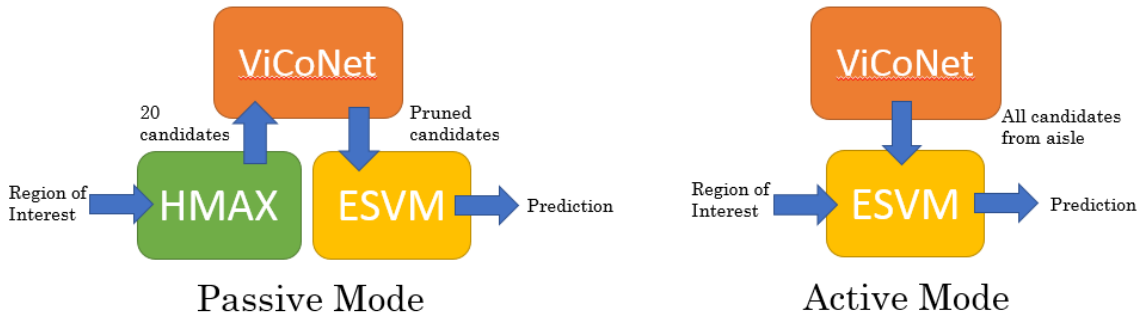


Figure 8: Passive and Active Modes

Passive mode refers to the operation as we have already seen, in which *k = 20* candidates from HMAX are pruned after stabilization decides on an aisle. In active mode, we perform stabilization as we have seen, but instead of then taking the results from HMAX and pruning them, we run all of the detectors corresponding to a given aisle. This results in a higher number

of detectors being run, as would be expected, but it also remedies the problem of HMAX not providing us the correct answer in the candidate list, and in the case of proper aisle prediction, guarantees that we will run the detector corresponding to the object in question.

Results

The results for this simple implementation can be seen in Table 2. We performed evaluations on the active and passive implementations both with and without thresholding (-Th) and compared them to the accuracy of the classifiers alone and the baseline ESVM-HMAX pipeline. Note that the accuracy for HMAX alone refers to the accuracy of the top $k = 1$ guesses.

System	Avg. Accuracy	Avg. No. of Classifiers
HMAX	35.42%	1
ESVM	67.90%	102*Exemplars
ESVM-HMAX	66.05%	20*Exemplars
ViCoNet-Passive	57.01%	3.85*Exemplars
ViCoNet-Active	59.56%	17.18*Exemplars
ViCoNet-Passive-Th	71.93%	**3.91*Exemplars**
ViCoNet-Active-Th	**77.43%**	13.21*Exemplars

Table 2: System Evaluation

We first see that the non-thresholded results are better in terms of the number of classifiers and therefore performance and power consumption, but significantly worse in accuracy. This is as a result of the aisle misprediction problem and was what suggested the thresholding idea. Upon implementation of the threshold, we see a marked improvement in both accuracy and detector reduction in both the active and passive cases. As expected, the active case does run more detectors than the passive but additionally sees an accuracy increase as a result of bypassing the inaccuracy of HMAX.

The tradeoff between accuracy and number of exemplars between active and passive can be considered in terms of the desired application. For example, a low-power device might prefer the more conservative passive approach to avoid the energy associated with running so many detectors. A different device with less power concerns might opt for the accuracy increase from the active case instead. These results demonstrate that even our simple implementation of ViCoNet, which did not even make use of the frequency or aisle location information and pruned instead of using probabilistic weighting, was able to demonstrate very concrete results.

Effect of Object Hierarchy

In this first experiment, we saw that a simple exploitation of context was able to improve both performance and accuracy. However, our experiment revealed some issues with the nature of our pipeline and with the data as a whole. As previously mentioned, the ESVM confidence thresholding that we introduced prevented any aisles from being mispredicted, but introduced the new problem of aisles remaining unknown due to a lack of usable information. Additionally, certain classes tend to have a low detection score through ESVM in general. Soda in particular consistently gets low detection scores, likely because of the similarity between the different objects in a soda aisle. Table 3 shows the average classification rates per aisle in both the passive and active cases (with thresholding) from the previous experiment to demonstrate the non-uniform nature of aisle classification accuracy. Since these aisles are related through very clearly established higher-level semantic object relationships, we now consider the fact that it may be possible to improve detection scores specifically for some of the lower-performing aisles.

Average Aisle Classification Accuracies		
Aisle	*Passive*	*Active*
Cereal	88.5%	95.6%
Cleaning	73.6%	89.8%
Coffee	71.1%	79.3%
Condiments	60.8%	85.4%
Cookies	61.7%	56.8%
Dental	47.5%	84.3%
Juice	15.0%	15.0%
Pasta	79.2%	76.7%
Sauce	74.3%	80.0%
Soda	44.4%	55.9%
Storage	66.3%	73.6%

Table 3: Classification Averages

Looking forward, our goal is to create a network that can represent the manifold complex relationships between objects with which we are familiar day to day. One basic type of relationship is a hierarchy of categories of object. For example, our "Pepsi Soda Bottle" and "Fresca Soda Bottle" could be considered part of a larger group of objects known simply as "Soda Bottle." This type of hierarchical semantic clustering has been explored in depth in Princeton's WordNet database, which also demonstrates the complicated nature of this problem. [12] We do not make use of the WordNet API in this simple extension to our experiment, but it might represent a good opportunity to extract hierarchical mappings in the future. The mapping of objects to groups is not clearly defined or strictly hierarchical, if we consider it in all the forms that humans group objects, but the task becomes slightly simpler if we focus on creating groupings that help us overcome issues in our visual pipeline. Starting out, it is not immediately obvious what groupings will fulfill this criteria, but it does give us an objective way to measure whether a chosen grouping methodology is worthwhile. To begin, we believe that if we group our objects by their similarity, we think that this may improve our detection scores either by reducing noise in test sets through similarity of candidate item shapes. It may also help because

accuracy will likely improve when the number of candidate items for a given ROI decreases. We saw in the passive pruning case from our experiment that lowering the number of detectors that need to be run on a given ROI seems to increase accuracy on average, so our hope is that pruning not just by aisle but additionally by shape may improve the accuracy further.

To evaluate this, we attempted to build on our results from the soda aisle. With an average ESVM confidence of 56% in the active case, it was not quite the worse aisle. Juice had a stunningly bad confidence of 15%. However, the juice aisle predictions were so bad that we did not have enough confident predictions to stabilize in the aisle at all. For our purposes, we wanted an aisle in which we stabilize correctly, but the detection scores are low, allowing us to exploit knowledge of possible object shape groupings within the aisle. However, lack of locality information could potentially also be exploited within this hierarchical system as well, and we discuss some ideas related to the juice aisle issues at the end of this chapter.

Within the soda aisle, we believed that the lower detection scores could potentially be mitigated by lowering the number of classes that needed to be run. There are 13 classes within this aisle, which fall into three distinct shape categories. The breakdown is shown in Table 4 below.

Shape Category	Class Name
Soda Bottle	Pepsi Soda Bottle
	Coca Cola Soda Bottle
	Fresca Soda Bottle
	Mug Root Beer Bottle
	Sprite Soda Bottle
	Diet Pepsi Soda Bottle
Soda Can	Coca Cola Soda Can
	Dr Pepper Soda Can
	Pepsi Soda Can
Soda Pack	Coca Cola Soda Pack
	Dr Pepper Soda Pack
	Sierra Mist Soda Pack
	Crush Soda Pack
	Mountain Dew Soda Pack

Table 4: Soda Aisle Class Groupings

We believed that running the detectors in a given grouping on an ROI as opposed to all 14 in the entire aisle would give better accuracy. This does rely on the assumption that it is possible for us to detect this shape in the first place, which is not a negligible assumption. However, the success of this experiment will motivate our desire to move forward in attempting to develop such a shape detector. We ran each grouping of detectors on each ROI separately and pooled the results, which can be seen in Table 5 ordered by ascending accuracy when grouped by shape.

Class Name	Ungrouped Accuracy	Grouped Accuracy
Fresca Soda Bottle	0.0%	0.0%
Sierra Mist Soda Pack	75.0%	0.0%
Sprite Soda Bottle	0.0%	16.7%
Pepsi Soda Bottle	22.6%	32.3%
Mug Root Beer Bottle	0.0%	33.3%
Coca Cola Soda Can	61.5%	69.2%
Coca Cola Soda Bottle	0.0%	70.0%
Diet Pepsi Soda Bottle	35.0%	85.0%
Dr Pepper Soda Can	85.7%	100.0%
Pepsi Soda Can	100.0%	100.0%
Coca Cola Soda Pack	0.0%	100.0%
Dr Pepper Soda Pack	63.6%	100.0%
Crush Soda Pack	100.0%	100.0%
Mountain Dew Soda Pack	77.8%	100.0%
AVERAGE	**44.4%**	**64.8%**

Table 5: Ungrouped and Grouped Accuracies

The results from this table are excellent! We see that every class except Sierra Mist Soda Pack benefits from this system with accuracies improving greatly for the most part. When exploring the reasons for this further, we find that the individual detection scores for classes do not improve. However, filtering by shape means that we end up removing some classes that act as false positives. Based on a deeper look at the results for Sierra Mist, it appears that the detector for this particular class was not firing most of the time in either the grouped or the ungrouped cases, and that this meant that the pipeline was performing a max operation on a set of 0s from the ESVMs, and in the ungrouped case, the 0 that was first that was selected essentially at random just happened to be the correct one, leading to a falsely high accuracy. In the second iteration, this was not the case simply by chance. This issue of running a max on all 0s suggests the necessity of an additional provision to avoid having such a problem in the future unrelated to the benefits reaped from the grouped approach, which is implemented in [10] as an "unknown" output from the pipeline and might be added here as well to handle that case.

Overall, these results do not necessarily confirm that hierarchical grouping works, but they certainly suggest that the matter is worth additional exploration.

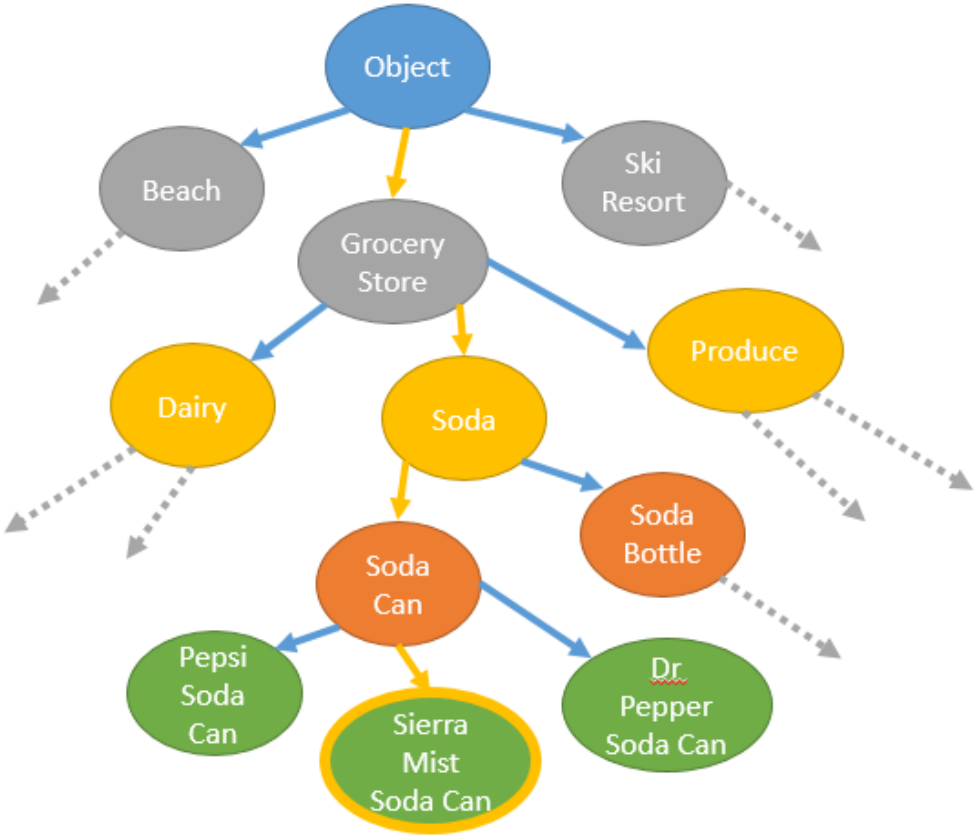


Figure 9: Example of Hierarchical Relationships

When considering all of this in the context of object relationships, and especially when trying to compensate for the relationships between objects in different scenes, these hierarchical results may ultimately lead us to a classifier decision tree like we see in Figure 9. Previously, the number of classifiers that we run on a given ROI was determined by either the pruned $k \leq 20$ we get from HMAX or the total number of classifiers for a given aisle based on whether we are in passive or active mode, which may not necessarily be the most likely subsets of classifiers for an ROI. If we were to arrange objects hierarchically, this would instead become a function of the

height of the tree. Each level of the tree could represent a particular granularity of object at a level that could be reasonably detected with ESVMs (object shape), inferred from ViCoNet (scene information), or extracted from the image in another way, allowing us to trim our results not to just a list of objects but instead to a specific subtree of relationships. This could potentially reduce the number of classifiers to be run if we can obtain meaningful accuracies to reliably prune the tree. In the soda case, for example, we run 14 classifiers on all soda products if we do not have any more specific filtering information. However, with a reliable shape detector, we could run 3 classifiers to determine the subcategory, plus 6 more in the worst case (bottle) leaving with us with at least 5 fewer detectors run overall.

We have previously discussed the open-ended issue of exactly how we will know what aisle we are in. This could potentially address that problem by giving us a way to "brute force" item detect to help us determine context. If we could create shape detectors on a high enough level that we could feasibly detect all of these higher level shapes without any aisle context and at reasonable computational expense, we could potentially narrow down the detection space quickly for objects in unknown aisles, allowing us to stabilize faster and more accurately. This would depend on the assumption that the tradeoffs with regards to accuracy and performance were in favor of the decision tree over the ESVM-HMAX pipeline. However, the results in this small experiment suggest hopeful things about the efficacy of hierarchy that might make such a comparison worth exploring.

Additionally, this hierarchical grouping could help us in the previously mentioned unpredictable aisles problem. For example, we found that the confidences for the juice aisle were so low that, out of all of the ROIs in the aisle, there were only two that passed our

"unknown" threshold. To remedy this, we simply left the test ROIs in that aisle unchanged, which essentially meant that they used the HMAX-ESVM pipeline for their detections with no semantic context. However, if the noise being introduced in our classification really is from the confusion related to the similarity between objects, it might be that a set of higher-level detectors might be the answer. Consider, for example, if we forwent running any "leaf" detectors—i.e. specific products like "Ocean Spray Juice Bottle"—and instead ran all of the higher level product category detectors, such as "Juice Bottle", "Tomato Sauce Jar", and others at that granularity, which would inevitably be a smaller number of detectors as well. Perhaps we can't determine on a first pass if a given item is a specific juice brand due to this noise, but if these more vague shape detectors can demonstrate higher accuracies, we might be able to identify that the previously "unknown" ROIs are juice bottles or juice boxes, which will still allow us to stabilize to the juice aisle and allow ViCoNet to begin running the correct subset of detectors.

Though this single experiment doesn't prove whether hierarchical grouping effectively increases our accuracy, it does open the door to additional experiments. However, as previously discussed, the applications for successful hierarchical detection schemes could possibly address some other issues we have encountered. At the time of writing, we still consider this to be a work in progress.

Chapter 5

A Dynamic Learning Approach to ViCoNet

Until this point, our work on ViCoNet has focused on a static, offline model in which ViCoNet is trained first and on annotations, which is to say ground truth, before it is used. This allowed us to be certain that our object relationships were valid and to verify if such relationships would be useful in the pruning stage. However, this builds on a host of assumptions that will need to be addressed as we move forward with our implementation.

When training our ViCoNet, we assume that the data in our annotation files is correct, since we've manually annotated them ourselves. However, ideally ViCoNet should be learning at all times from the detections that the pipeline performs, and these will not always be accurate, as we have seen. Without the ground truth available, we can't be sure which detections are right and wrong without pestering the user, which we'd like to do infrequently, if at all. This means that we have to assume our pipeline is at least mostly correct as it provides us with detections and allow ViCoNet to learn both the good and the bad, and hope that in the long run that we will generally be right more frequently than wrong and that the incorrect connections can be pruned.

Our previous version of ViCoNet was also entirely static, being trained before any testing took place and not learning any relationships based on the results of our test detections. This means that we are losing the potentially useful information that we gather through our process of classification. The relationships between objects change over time, and ViCoNet should be able to learn these new relationships and re-weight the likelihood of objects. For example, a user might find that a store has been rearranged. Their device should be able to learn the new layout of the store. However, it should also be resilient enough to avoid false positives in restructuring.

A single misplaced item should not reweight all of the expectations learned over a longer time and learned relationships that prove to be extraneous over the long run should eventually be pruned away.

Finally, we perform all training in the previous chapter before any testing. However, a real life implementation of ViCoNet would likely be built incrementally, especially if it was being built from the ground up. This means that we should expect to see ViCoNet's prediction ability increase as it learns more.

In this chapter, we redefine our approach to ViCoNet by looking at in the context of a more interactive vision pipeline. We attempt to interleave the training and testing stages into a single, continuous cycle mimicking a shopper walking through a store. The rough idea of ViCoNet is still the same, as we are still attempting to capture and report object relationships. However, the structure of and interactions with ViCoNet are very different.

ViCoNet Testing Vehicle

The code for this implementation of ViCoNet is entirely separate from that used in Chapter 4. We are still working in C# for the same reasons explained in Chapter 4, but many things have changed. ViCoNet now runs as part of a larger "vehicle" representing something akin to a device containing its data structures and the HMAX-ESVM pipeline as well. The vehicle for ViCoNet stores its information in XML format, the rough structure of which is shown below. The actual relationships stored in ViCoNet are learned from the deserialization of these files, described more fully in the methodology section below.

```
<ViCoNetDataset>
	<Classes>
		<Class ID="0" Name="Class 0" />
		<Class ID="1" Name="Class 1" />
		<Class ID="2" Name="Class 2" />
		<Class ID="3" Name="Class 3" />
		<Class ID="4" Name="Class 4" />
	<Classes>
	<Scenes>
		<Scene>
			<Image>datasets/viconet/images/scene1.jpg</Image>
			<Label>Scene Label</Label>
			<Width>1920</Width>
			<Height>1080</Height>
			<!-- Height may not be strictly needed, but why not! -->
			<Objects>
				<Object>
					<Location>
						<X>0</X>
						<Y>0</Y>
						<Width>50</Width>
						<Height>50</Height>
					</Location>
					<Classification>
						<Ground_Truth>
							<Truth Index="0" />
						</Ground_Truth>
						<RLS_Responses>
							<RLS Class="0" Response="0.5" />
							<RLS Class="1" Response="0.5" />
							<RLS Class="2" Response="0.5" />
							<RLS Class="3" Response="0.5" />
							<RLS Class="4" Response="0.5" />
						</RLS_Responses>
						<ESVM_Responses>
							<ESVM Class="0" Response="0.5" />
							<ESVM Class="1" Response="0.5" />
							<ESVM Class="2" Response="0.5" />
							<ESVM Class="3" Response="0.5" />
							<ESVM Class="4" Response="0.5" />
						</ESVM_Responses>
					</Classification>
				</Object>
					<!-- Multiple objects in each scene -->
			</Objects>
		</Scene>
			<!-- Multiple scenes -->
	</Scenes>
</Dataset>
```

Figure 10: Example of XML format

Within this format, there is a list of classes which corresponds here to the 102 classes used in the experiments in the previous chapter. There is also a list of scenes, each of which represents a single image. Within a scene is some information about the image along with a list

of objects within the scene. In the previous implementation, we drew an edge between nodes in ViCoNet if they were less than one thousand pixels apart, which was a very dataset-specific decision boundary. Here instead we change our edge-drawing criteria to whether or not objects appear in the same scene with one another without worrying about distance. These objects are still based on our annotations from before and the ground truth is available, but it is only used for accuracy calculation. Each object additionally contains information about its location and the confidence results of classification for this ROI through HMAX/RLS and ESVM, which allows ViCoNet to take advantage of the confidence scores, something it previously could not do. The previous version of ViCoNet was disconnected from the details of the pipeline, so our hope is that a closer integration with the pipeline may yield more precise prediction results. Additionally, simulating the entire pipeline in software allows us to experiment with the precise interactions between HMAX and ESVM, which was not nearly as simple before.

Dataset

This new format changes our previous concept of aisles into a more easily generalizable idea of a scene, which in this implementation represents a single image. Part of the goal with this implementation is to make the results of Chapter 4 more generalizable, and a crucial first step is to replace the retail-specific concept of discrete aisles with general scenes. This means that our training grocery dataset, consisting as it does of 14 images corresponding to one panoramic aisle image each, cannot be used. Our test dataset, however, contained various images that are not grouped by their aisle type. Each image is a picture from only a single aisle, but no one image contains all of the important relationships between all objects in an aisle.

Some example images from each dataset can be seen in Figure 11, showing the difference between the much smaller segments in A versus the entire aisle in B. For this reason, we have chosen to work with the test dataset.

A.) Example images from viconet_test

B.) Example image from viconet_train

Figure 11: Differences between training and test datasets

Methodology

In Chapter 4, we split our training and testing into two distinct phases, the latter of which was additionally split into calibration and pruning phases. As previously mentioned, having separate training and testing phases leads us to discard potentially valuable new relationships learned during testing. Additionally, the split between calibration and pruning required us to know exactly when we had entered a new aisle, which is a nontrivial issue. Since our goal is now to address some of these concerns in a practical way and we are abandoning aisle-level granularity, Dynamic ViCoNet treats all of these distinct phases as a continuum.

Our goal is to simulate a person walking through a store, building their ViCoNet incrementally, and occasionally stopping and querying ViCoNet for a classification on a specific item. We therefore establish two modes of functionality: "learning", in which we use the HMAX-ESVM results alone to populate ViCoNet, and "querying", in which we use our knowledge of the current scene and existing object relationships to weight the HMAX-ESVM results and hopefully achieve better accuracy. This should allow us to compare the performance of the pipeline with or without ViCoNet, as before. Additionally, we can now explore the efficacy of ViCoNet with regards to time, with the hypothesis that it should improve in later scenes as we learn more relationships.

Given the XML dataset format shown above, ViCoNet parses one scene at a time. Within a scene, it goes object-by-object with a 5% chance to enter querying mode on any given object. For our purposes, learning happens much more frequently than querying on the assumption that a user walking down an aisle will stop and look at items somewhat infrequently on average. However, this number is not based on any particular empirical measurement.

In the learning stage, we take the top $k = 20$ results from HMAX/RLS and choose whichever has the highest ESVM score, duplicating the basic pipeline from [10] and Chapter 4. This value is added to a list of objects to be added to the scene, which are added as a group once all objects in the scene have been processed. This group adding is important, as ViCoNet needs to know which objects are related to one another. For this reason, Dynamic ViCoNet is also stateful with regards to our current scene. This state both corresponds to and eliminates the need for the results of the stabilization phase in the previous version of ViCoNet. Unfortunately, in losing our aisle knowledge "oracle", we also lose the absolute listing of all products in an aisle,

so depending on how connected certain objects are, ViCoNet's state may not be a comprehensive listing of relationships in an aisle. However the hope is that over time, enough relationships within an aisle will be established to improve this predictive ability.

The querying stage is slightly more complicated. We begin by adding any objects that have been seen already in the same scene, if any, into ViCoNet, both establishing our state and taking advantage of any contextual relationships they may expose. Then, we use the HMAX-ESVM pipeline, augmented with ViCoNet, to perform what we hope will be an improved prediction on the current ROI. A diagram of the high-level differences between the two types of behavior can be seen in Figure 12.

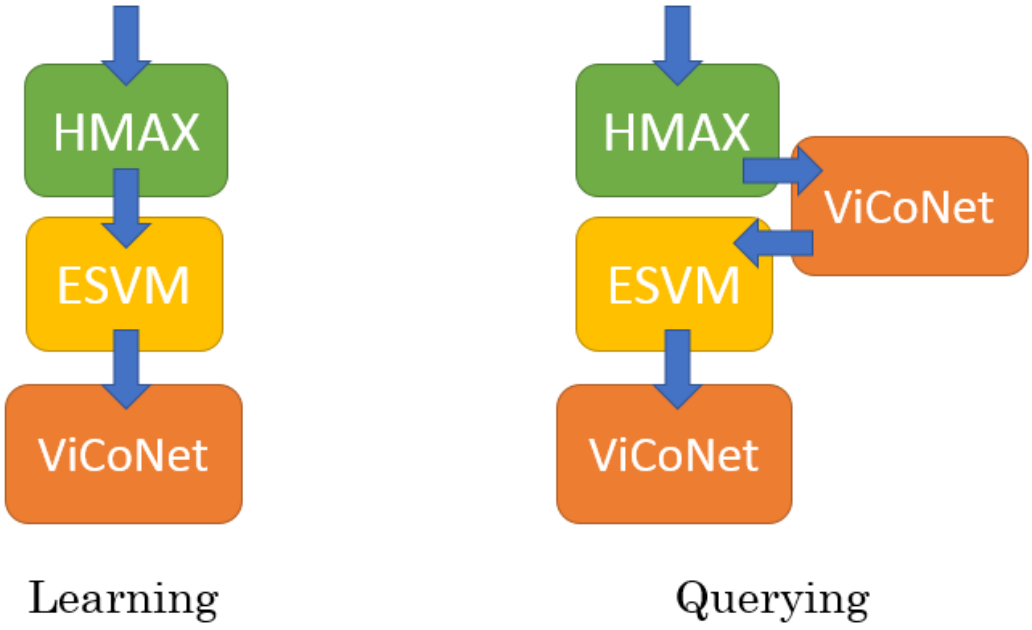


Figure 12: Learning vs. Querying Behavior

First, the vehicle queries ViCoNet for the top *n* candidates a given ROI at a max distance of *h* hops based on its current context. If this particular ROI is the first in the scene, which means there is no context yet for ViCoNet to use, it simply performs detection using the HMAX-

ESVM pipeline without ViCoNet. However, we still count this detection as part of the HMAX-ESVM-ViCoNet detection statistics. If there is context, ViCoNet builds a candidate list as illustrated in Figure 13. The overall scheme for this list-building is to take the top *n* candidates with the highest probability, and the calculation of the values in the table in Figure 13 will be discussed as the process is explained.

C=∅;
while $size(\boldsymbol{C}) < n$ **do**
  if $\nexists$ *node* $x \notin \boldsymbol{C}$ *less than* h *hops from a node in* $\boldsymbol{N}$ **then**
    break;
  else
    find node x where $p(x) = max(p(y)) \forall$ nodes $y \notin \mathbf{C}$;
    add x to **C**;
  end
end

Algorithm 1: ViCoNet Candidate Selection

The overall candidate generation process is implemented as a simple greedy algorithm, shown above. We choose up to *n* classes in order of their probability with the additional constraint that the nodes must be within *h* hops of the current context, a value which can be set to infinity if desired. **N** is the name we use for the set of nodes in the current context, also referred to as the "core" nodes. **C** is the list of candidates being built incrementally. To understand this more fully, we must define the value of the probability of a node. For nodes in the current context **N**, we define the probability with Equation 1 below. The value of the function *w(n)* is the weight of node *n*, which corresponds to the number of times that *n* has been seen. Roughly speaking, the probability of a core node is the ratio of its appearance count to the sum of the appearance counts of all nodes in the context.

$$p(n_i \in \mathbf{N}) = \frac{w(n_i)}{\sum_{\forall n_j \in \mathbf{N}} w(n_j)}$$

Equation 1: Probability of a Core Node

For nodes not in the core, we define their probability to be based on the edge through which we reach them. Each edge contains a count that represents the number of times that the destination node has been seen with the source. The probability of an edge is defined below in Equation 2 as the weight of the particular edge from source node *i* to destination node *j* divided by the sum of all outgoing edges from node *i*.

$$p(n_i \rightarrow n_j) = \frac{w(e_{i \rightarrow j})}{\sum_{\forall e_{i \rightarrow k}} w(e_{i \rightarrow k})}$$

Equation 2: Probability of Node j from Node i

This solidifies what we mean by the probability for a given edge, but a single edge does not fully represent the probability associated with a node, especially since candidate nodes may be multiple hops from the core nodes. We define the path *P* to a node as the set of nodes through which we reach it, as shown in Equation 3 below. The first node in this set must additionally be in the current context. We additionally define the set **P** as the set of all paths to a given node for simplicity, as some nodes may be able to be reached multiple ways from the context.

$$P_i = \{n_0, n_1, ...n_p\}, n_0 \in \mathbf{N}$$

$$\mathbf{P}_{n_i} = \{P_0, P_1 ... P_p\}$$

Equation 3: Definition of Paths

Now we can define what we mean by probability for nodes not in **N**. As previously mentioned, we consider probability to be edge- and therefore path-dependent. The probability for a path is defined below as the probability of each node in the path multiplied together. This means that we use Equation 1 for the first node, which by definition must be in **N**, and then use Equation 2 for the others. The probability for a given node, then, is the maximum probability of all the paths that lead to it.

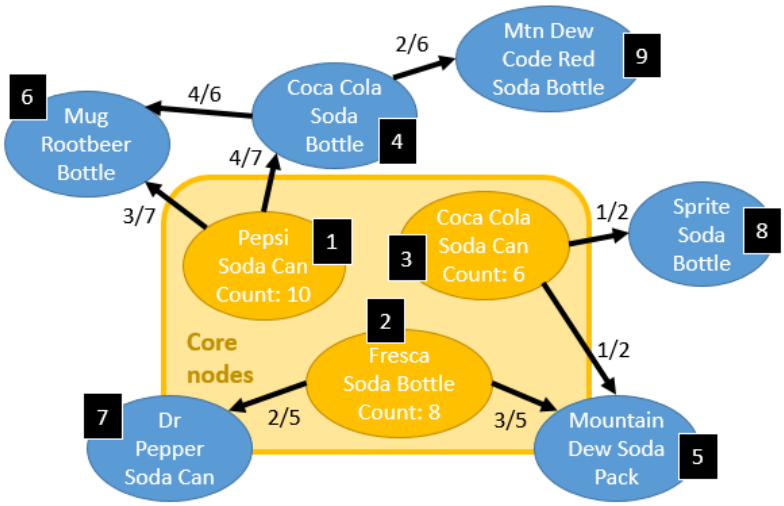
$$p(P) = \prod_{\forall n_i \in P} p(n_i)$$

$$p(n_i \notin \mathbf{N}) = max(p(P_j \in \mathbf{P}_{n_i}))$$

Equation 4: Probability of a Path

The specific definition of the probability as the edge weight divided by the sum of edge weights leads to some convenient properties that make the algorithm more efficient to implement. At first, this may seem like a variation of Dijkstra's algorithm in which we must calculate the value at any given point before we can begin our selection. However, the probability definition means that the sum of outgoing probabilities from a given node is equal to one, and consequently that all probabilities along a spanning tree from a given node will be less

than or equal to the probability of the node itself. This monotonically decreasing feature of the probability means that we can solve this with a simple greedy algorithm.



Order	Node	Probability
1	Pepsi Soda Can	10/24 (0.417)
2	Fresca Soda Bottle	8/24 (0.333)
3	Coca Cola Soda Can	6/24 (0.250)
4	Coca Cola Soda Bottle	5/21 (0.238)
5	Mountain Dew Soda Pack	1/8 (0.125) or 1/5 (0.200)
6	Mug Rootbeer Bottle	5/28 (0.176)
7	Dr Pepper Soda Can	2/15 (0.133)
8	Sprite Soda Bottle	1/8 (0.125)
9	Mtn Dew Code Red Soda Bottle	5/63 (0.079)

Figure 13: How ViCoNet Calculates Probabilities

Figure 13 shows an example intended to demonstrate the algorithm visually and show some important features of the decision process. For example, the probability of Mountain Dew Soda Pack can take on two values depending on which path we use to reach it. With regards to Fresca Soda Bottle, the edge probability is 3/(3+2) = 3/5. With regards to Coca Cola Soda Bottle, it is 1/(1+1) = ½. In addition, we consider these values in the context of the source node probability, leading to two final choices of (8/24)*(3/5) = ½ for Fresca Soda Bottle and (6/24)*(1/2) = 1/8 for Coca Cola Soda Bottle. We take the max of these two possibilities. Another illustrative example is that of Mtn Dew Code Red Soda Bottle, which is an additional hop removed from the other nodes. Its probability is calculated as p(Pepsi Soda Can)*p(Coca Cola Soda Bottle)*p(Mtn Dew Code Red Soda Bottle) = (10/24)*(4/7)*(2/6) = 5/63.

This entire context retrieval algorithm is based on what is at its core an inherently flawed way to reason about object relationship probabilities. On our dataset, we believe it does not cause a major problem because the number of annotated ROIs for each class is relatively even. However, this particular probability model exhibits overfitting on datasets exhibiting asymmetric representation of objects. For example, the initial ranking of context nodes by their appearance probability is based on the assumption that an object's statistical appearance count in previous scenes is proportional to its probability of future appearance. This appears to hold true in our dataset, but is an important consideration for future work.

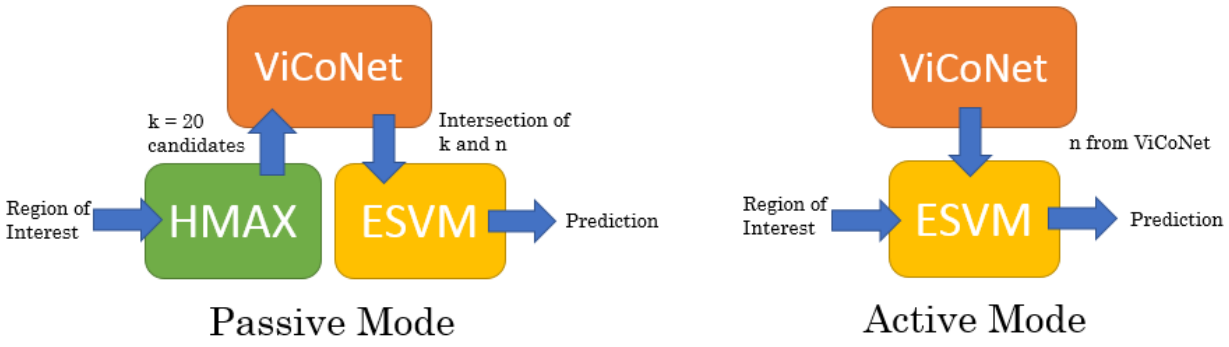


Figure 14: Active vs. Passive Modes

Once the test vehicle has this candidate list, it makes use of it in one of two modes: active or passive, similar but not identical to the corresponding modes in the previous implementation of ViCoNet. In passive mode, we take the intersection of this set of candidate nodes with the top *k* from HMAX, essentially pruning the HMAX guesses to a subset that pertains to our current context. In active mode, we don't look at the HMAX results at all and instead use all *n* from ViCoNet, giving us a larger number of classifiers to run, but potentially better accuracy. A

visualization of this can be seen in Figure 14. Finally, we run the ESVM classifiers associated with this subset and take the maximum response.

Results

We evaluated Dynamic ViCoNet's performance with regard to a variety of variables, some of which have been mentioned earlier in this chapter. As our goal is to compare to the results of chapter 4's ViCoNet implementation, we set $k = 20$ again so we can duplicate the base pipeline performance of HMAX as seen in Figure 3 and Table 3. It is also important to keep in mind that the goal of this new implementation was to do away with what we consider unrealistic behavior, such as the complete population of ViCoNet with ground truth data before any testing begins, so we do not expect to see results improving on the previous implementation. Our goal is rather to preserve an advantage over the base pipeline in the face of these new and more realistic changes to the overall pipeline. Each experiment was run twenty times and averaged. We set the chance to enter querying mode as 5%, as mentioned before. We run both passive and active modes over a range of values of *n* and maximum hop count and report the configuration that gave the best accuracy with ViCoNet. Additionally, we provide the results from a ViCoNet trained on only the ground truth information instead of the results of the detection pipeline to establish a theoretical upper bound and see how much of an effect the imperfect detections have on accuracy. For the configuration with the best accuracy, we also present graphs showing the relationship of accuracy and number of classes to the values of those variables. Finally, we show a histogram of the distribution of correct predictions over time for the best accuracy to explore whether ViCoNet becomes more accurate over time as it learns.

System Configuration	Value of n	Maximum Hops	Average Querying Accuracy	Average number of classes
passive	10	0	66.4%	4.02
active	25	0	71.8%	4.82
passive-gt	30	3	84.0%	1.22
active-gt	15	Infinite	96.9%	2.15

Table 6: Best Accuracy Results for Different Configurations

Table 6 shows our results and demonstrates some interesting trends. The base HMAX-ESVM pipeline provides us with the 65% accuracy established in the previous paper. The average number of classes refers to how many classifiers are run on average per class within the querying stage. In the learning stage, it is always 20 because that is the value of *k* that we take from HMAX. In querying, it is either the size of the intersection of the *k = 20* from HMAX and the *n* from ViCoNet in the passive pruning case, or the size of *n* from ViCoNet in the active case. It is not, however, always *n*, as ViCoNet does not always, or even usually, return exactly *n*. Most of the time there are substantially fewer nodes reachable from the current context. We see here that querying can provide a small increase in accuracy over the baseline 65%.

As might be expected, we see that active mode tends to perform better than passive, but that the number of classes is higher in active. The explanation for this phenomenon is the same as it was in Chapter 4. Additionally, we see that the number of classes is always substantially better than the constant 20 in the learning case. The best accuracy appears when we are in active mode with *n = 25* and max hops set to 0. This is also the querying case in which we use the most classes for classification. However, it is still less than 25% of what we use in the learning case. To further explore the results, let's look at the effect of *n* and max hops on the accuracy and number of classes used.

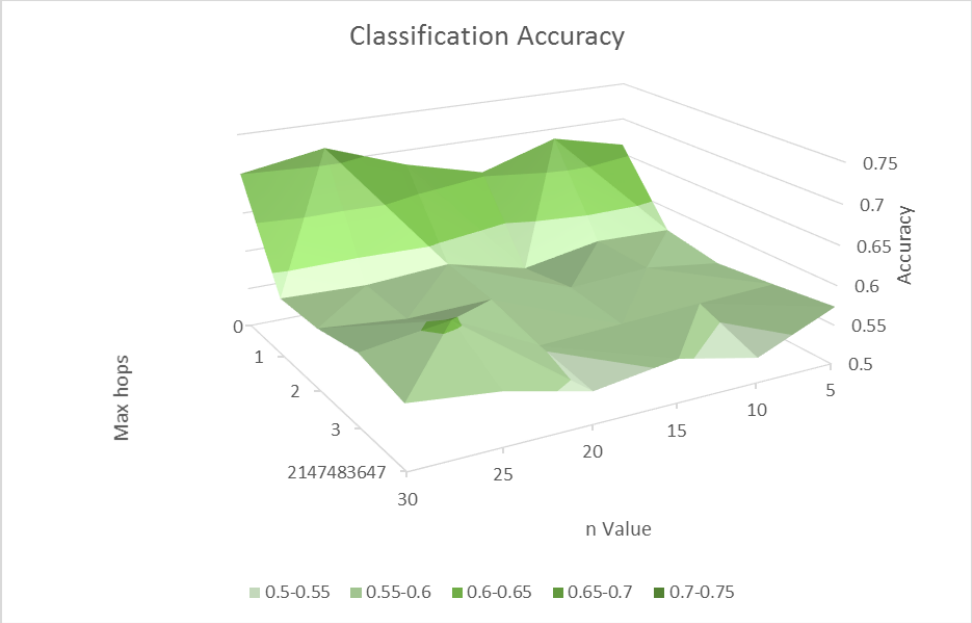


Figure 15: Classification Accuracy for *active*

Here we see what might be a surprising trend: the accuracy drops off precipitously as the value of max hops increases. The largest value on that axis corresponds to max hops being set to infinity. After some careful debugging, we found that the reason for this is because the pipeline can only provide us 65% accuracy when we are learning. In this stage, ViCoNet absorbs a great deal of noisy and inaccurate relationship information, as the HE pipeline is not terribly reliable. We see the highest accuracy when max hops is 0, which corresponds to ViCoNet only choosing nodes in the current context. The nodes in the current context are inevitably affected by this noisiness as well, but as soon as we begin to look at the neighbors of the nodes in the context (max hops 1 or greater), the nodes are now affected both by the correct classification of the context nodes as well as the correct classification of their neighbors in previous settings. Each hop that we go out past the context multiplicatively worsens the probability of correctness by 70% in the best possible scenario, leading to bad accuracy even at one hop. The relationships learned by ViCoNet break down quickly, and for the most part, the neighbors learned don't make

a great deal of sense. We also see here that higher values of *n* seem to slightly improve the accuracy, but the relationship of *n* to the accuracy is much weaker. This also makes sense, as taking more nodes from ViCoNet makes it more likely that the correct node will be selected.

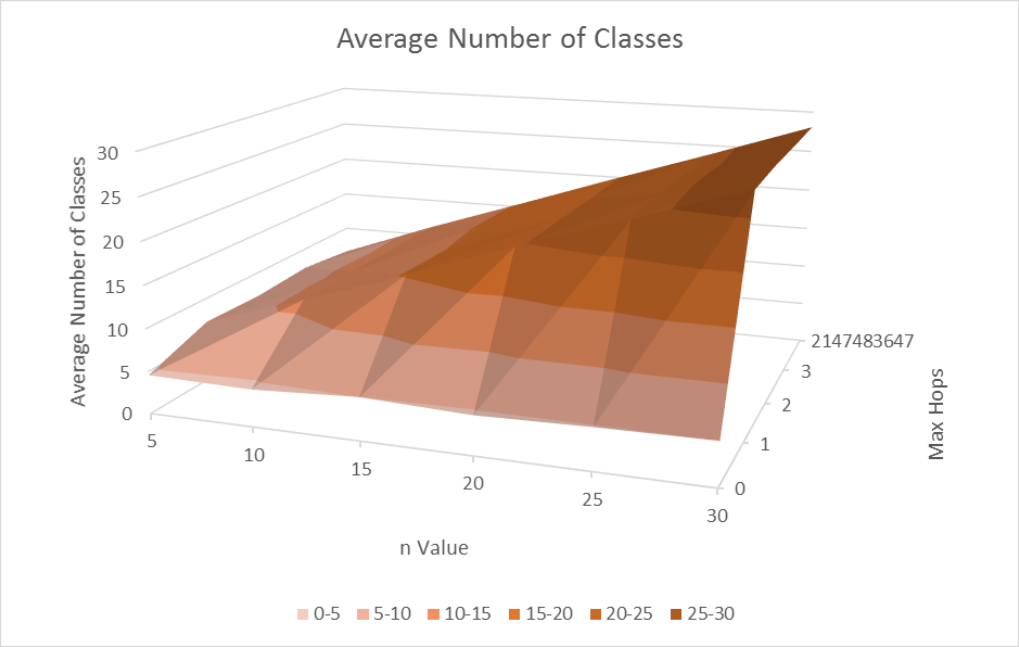


Figure 16: Average Number of Classes for *active*

In Figure 16, we look at the relationship between the average number of classes and the values of *n* and max hops. The relationships here are sensible, if not terribly enlightening. The number of classes increases as *n* increases, as we are asking for more nodes from ViCoNet. Additionally, the value of max hops affects the number of classes, as there may not be *n* nodes located within a certain number of hops. We also see that, after 3 hops, we seem to have reached a plateau, as the infinite hops value isn't much larger. This means that, within our particular data set, most nodes tend to be accessed within two or three hops of the core, which might help provide some insight about the structure of ViCoNet and may allow max hops to be selected both for energy and memory concerns.

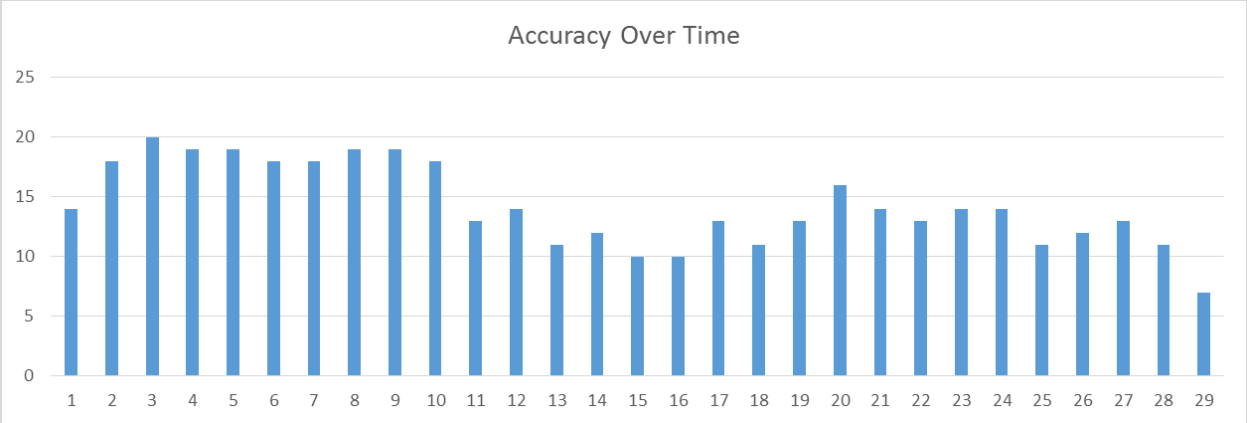


Figure 17: Histogram of Accuracy over Time for *active*

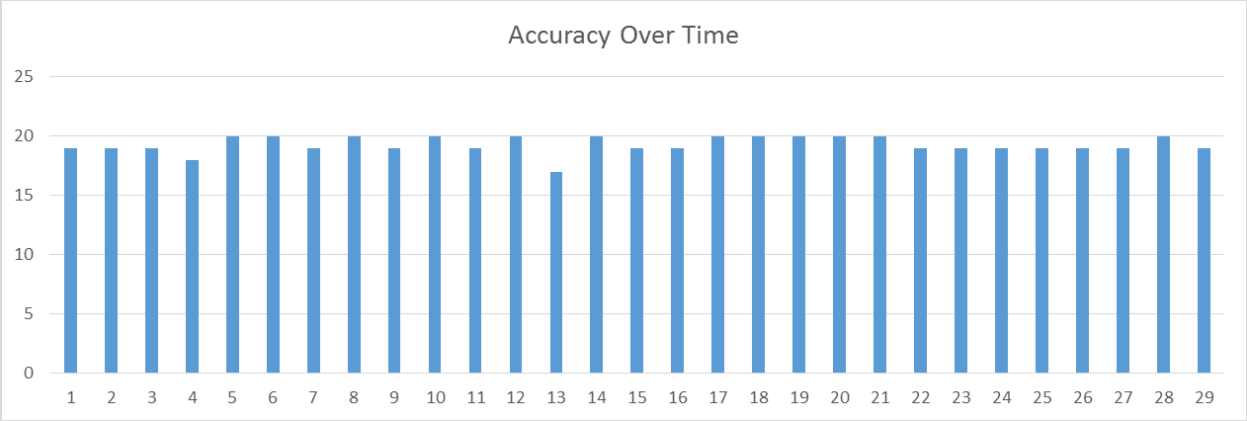


Figure 18: Histogram of Accuracy over Time for *active-gt*

Finally, we look at the temporal effect of classification. We expected that the accuracy would increase over time as ViCoNet is able to learn better relationships and improve its understanding. However, we see the exact opposite happen! The histograms here show the number of times that the pipelines got a particular detection right over 20 runs, so a value of 20 means that it was correct each time and is the best possibility. For the active case, we see that the results begin to obviously drop off fairly quickly, which does not happen in the active-gt case. In the active case, this is the effect of ViCoNet learning bad relationships or context information from its HEV results. We might expect the active-gt case to demonstrate the

increasing accuracy trend that we originally hypothesized, but the accuracy is so close to perfect to begin with that there isn't much room for growth. Additionally, our dataset exhibits the property that objects tend to appear more than once in an image, so taking only the context nodes is often good enough if we have correctly classified them. Though undesirable in the active case, these results show the importance of trying to mitigate noise and bad relationships in ViCoNet, as over time they compound and begin to hurt accuracy.

This exploration of a Dynamic ViCoNet allowed us to take the proof of concept experiment introduced in Chapter 5 and make it more realistic by injecting noise into what was previously ground truth data. In doing this, we purposely made the problem more difficult to solve and saw that ViCoNet still helps to improve the classification accuracy, even when it learns its relationships from the HMAX-ESVM pipeline that only provides 65% accuracy. When we consider training ViCoNet, it may not be unreasonable to consider training it on ground truth data at least in the very beginning so that it can start with the best accuracy which will only help ViCoNet as it continues to learn. Specifically within retail, for example, a store may have a planogram of the various products and their corresponding locations available that could be used to pre-weight ViCoNet before a user begins to shop. An important next step here might be to find a way to improve the basic pipeline accuracy, as ViCoNet can only be as intelligent as the information on which it is trained. Additionally, exploring ways to prune "bad" data out of ViCoNet might be a valuable exercise. Overall, this experiment demonstrates that using high-level semantic context can improve accuracy and energy consumption and is surprisingly robust even with a great deal of noise in its knowledge.

Chapter 6

Conclusions and Future Work

The work we have begun on this idea is only the beginning. Our explorations have been narrow in scope and limited by things such as the lack of large datasets annotated with a fine enough granularity to make use of visual co-occurrence context. Additionally, our actual implementations of ViCoNet were quite simple when considering the flexibility to add many more types of data into the relationships. In Chapters 4 and 5, we sought to demonstrate how this context information can be useful, even at an extremely basic level of filtering. In this chapter, we discuss some of the ideas we have considered going forward, both through brainstorming and also through related works, and consider how they might be represented in ViCoNet and what effect they may have on our decision making.

Color HMAX

Currently, our HMAX implementation is in greyscale for simplicity, but color can be a simple and extremely effective way to filter certain objects. For example, we can be quite sure that an object that is bright blue is not an orange or a strawberry, and we can therefore filter out objects like that, even if they might share other visual characteristics with the object being identified. Additionally, within our retail setting there are certain aisles with particularly low detection results, such as Soda, as explored at the end of Chapter 4. Most soda bottles are a fairly uniform shape and additionally how some labels, like Pepsi and Diet Pepsi, are almost identical except for their color differences, the addition of color within the pipeline could

potentially boost base detection scores in the base ESVM-HMAX pipeline even before any sort of hierarchical support is added, which in turn can only help ViCoNet stabilize more accurately and provide better context support.

One specific example of Color HMAX can be seen in [13]. This system is founded on the assumption that focusing only on the texture and shape pathways of the neural cortex, as is the case in the original HMAX, means neglecting channels within the brain that might yield useful information. When evaluated on a dataset of traffic signs, which is dataset demonstrating a great deal of similarity in terms of shape and texture of different classes, the authors were able to demonstrate improvement over standard HMAX with the additional observation that the detection scores were lighting-invariant. Though this change would not directly affect ViCoNet, the addition of color could improve HMAX's accuracy, leading to higher confidence in initial scene stabilization and allowing ViCoNet to provide more accurate information. Additionally, as we saw in Chapter 5, the base pipeline's low accuracy caused ViCoNet's learned relationships to be fairly unhelpful, so this could be one step towards addressing that problem.

Spatial Relationships between Items

We have established that the relationships between the abstract representations of a pair of objects may vary based on certain variables such as the scene in which they are present. Within a scene, these relationships may additionally be related by the relative spatial positions of the objects to one another. Consider the simple example of what objects you would expect to see above a tree in an image as opposed to those you would expect to see below it or nearby on the ground. In [14], the authors explore this idea of spatial relationships with regards to the structure

of a human face. The paper focuses on the details of the vision algorithm which are not as important to ViCoNet, but demonstrates that both a reduction in problem dimensionality and improvement in accuracy of face detection.

In a retail setting, for example, this sort of information could be used in a task-based scenario where we detect an item that we know tends to be above the item we are seeking, and a cue could be provided to the user to look or reach in the appropriate direction. This would lend itself to another project in our lab in which a camera has been mounted on a glove that is hooked up to a task-based detection program. The user tells the device what object they would like to locate, and the camera seeks to identify it on a shelf and directs a user's hand with vibration on the sides of the glove. These spatial relationships might help the device guess what direction a desired product is most likely to be in based on relationships it has seen before. This would be especially helpful in the context of a specific, well-known store as long as items on the shelves are not rearranged. This spatial information might not be helpful in all cases, but there are many in which it might provide us with valuable information on which we can filter.

GIST

In Chapter 2, we discussed several alternative context extraction methods, including GIST. Now, we can come back to this idea with the realization that the idea of GIST is in many ways similar to what we attempt to achieve with ViCoNet when we are talking about scene classification based on the objects in a scene. Going forward, we can think of GIST as not an alternative option, but instead a complimentary one. For example, if ViCoNet cannot say with enough confidence what our scene is or if it will be significantly more computationally

expensive than GIST, GIST might be called upon to do so instead. The two could even potentially combine their guesses to determine the location with better confidence. Another application for GIST might be to use it not as a location deciding algorithm but rather as a tag on relationships between objects. For example, a table in a kitchen setting will have very different relationships to the items around it from a table in other settings, as seen in Figure 19. We could tag the edges of the "table" node with the GIST vector or resultant classification from the scene in which each relationship was established, which could be used in concert with a scene prediction for an image being classified to probabilistically weight which objects are likely to be together in the same type of scene. In a sense, GIST really just provides an attempt at a feature-extraction-based way to achieve the same level of semantic knowledge that we are aiming to discover with ViCoNet.

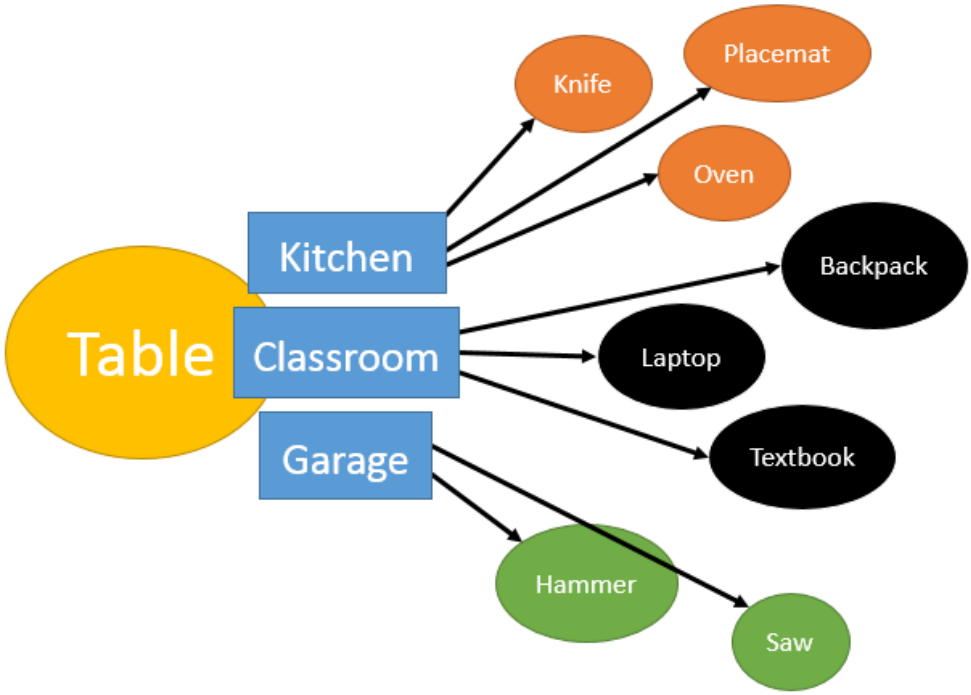


Figure 19: Example of GIST-weighted Relationships

We spent a great deal of time early on discussing how GIST could be incorporated into ViCoNet before we had solidified any decisions about the datasets or setting that we would be using. Within our grocery setting, GIST would not really have helped us, as the GIST for an image in a given aisle would not be very different from that in another aisle in the same store, which is what led to its abandonment. It could be useful to distinguish one store from another with different lighting and décor choices, but we believe that's not the best use of GIST as a filtering feature as it is overly specific and does not help us expand our horizons beyond the retail setting with which we began. However, imagining a ViCoNet that operates in more settings, we had planned originally to group objects by GIST using a centroid and radius method. Objects would be tagged with a GIST vector corresponding to the locations in which they were seen, and then a clustering algorithm such as k-means could be used to group our objects based on their distance from a centroid GIST vector. The idea here would be an unsupervised approach to grouping that might expose more useful relationships based on the similarities of GISTs as opposed to manual grouping.

Classifier Confidence

We could also incorporate confidence information from the classifiers into our ViCoNet weighting. We have already made use of the confidence information available to us in creating an "unknown" threshold. This information might additionally be used to help decide object weights as we train ViCoNet. We have previously explored the issues that can arise from training ViCoNet on detector output in Chapter 5 instead of the ground truth annotations in Chapter 4, but for simplicity, we assumed that the detector was always right and that any

mistakes would eventually be overshadowed by more confident, correct detections. When training ViCoNet, we might consider the strength of an established relationship between objects to be a function of the confidence of the detection from which it is trained. For example, in Chapter 5, we increment the count between objects by one each time the detector reports that they have been seen in the same scene. We might instead consider weighting it by a fraction of one corresponding to the confidence of the detection. The previous system requires the frequency of detections to wash out incorrect candidates—that is, a single incorrect detection weighs just as much as a single, highly confident, correct detection. Coupled with a reasonably reliable detection pipeline, this modification could potentially allow us to wash out low confidence relationships faster.

Additionally, when we consider performing detections, it might be useful to weight the likelihood of items based on the average confidence of a detector in previous detections of said object. In a high power system, this might not be as important, but in a system where we want to run as few detectors as possible, we might be served well by running those with which we can be most confident of a classification.

Group of Objects Detection

Within many scenes and especially within retail, we can often expect to see multiple of one object in a single area. When each of these is extracted as a salient ROI, this can lead to a great deal of redundancy in detection. Theoretically, if we could know that a group of ROIs were all the same object, it would only be necessary to run classification on a single instance to determine the label for all of the objects. Within our lab, other students are working on an

algorithm based on a generalized form of the Hough Algorithm [15] [16] for symmetry detection and the Attention by Information Maximization (AIM) algorithm [17] for saliency. The goal is to be able to schedule detections based on repeating pattern detection to avoid the redundancy discussed previously. The detection of repeated objects like this represents another form of information that could be captured in ViCoNet as a characteristic of the object node. Different objects may be more likely to be seen in repeated groups than others, such as windows on a building versus doors. Additionally, the scene may affect whether objects are likely to be seen in groups or alone. One might expect to see one box of cereal in their kitchen and a tiling of that same box in a grocery store. When extracted and stored in ViCoNet, these relationships may help us to decide whether we should run an algorithm to seek similar objects nearby. The detection of a grouped set of objects might additionally allow us to exploit the reverse relationship and intuit something about our scene. Overall, the results from a group detection algorithm could be another tidbit of information that can be stored within ViCoNet.

Better Probabilistic Edge Weighting

Our simple implementations of ViCoNet keep a count of each occurrence of an object and its associated edges. We do not make use of this in Chapter 4, and in Chapter 5 we determine the relative probability of relationships between objects by dividing the weight of a single outgoing edge by the sum of all edges. However, this method does not really represent a probabilistic guessing of what might be the objects most likely to be seen with any given object. In fact, in the case of reweighting, it will be problematic, as a certain edge may have a large count already, and new counts may take a long time to "catch up" and become statistically

significant in comparison. This is especially problematic when considering a version of ViCoNet strongly trained on a given dataset and then attempting to dynamically learn a change in object relationship, such as a rearrangement of a store. Consider Figure 20, which attempts to illustrate this problem more clearly. Given the recent exposure of gluten intolerance in the public eye, a grocery store might choose to make a separate section of the store for all of the gluten-free items. The central class "Gluten-Free Bread" in this example used to be located in the bread aisle and had established heavy edge weights with other bread nodes, represented in blue. Upon store restructuring, ViCoNet will take some time to learn the new relationships between this item and its new, gluten-free neighbors. However, ViCoNet will not begin to predict new neighbors, such as the "Gluten-Free Crackers" class, until its edge weight becomes comparable to the others. Depending on the other edge weights, this could take a very long time. If the previous bread relationship edges represent years of shopping in a store with the previous layout, it could take the same amount of time to bolster new edge weights enough to make them significant in comparison. Object relationships in the world are too dynamic for us to require such a drastic amount of information to reweight them.

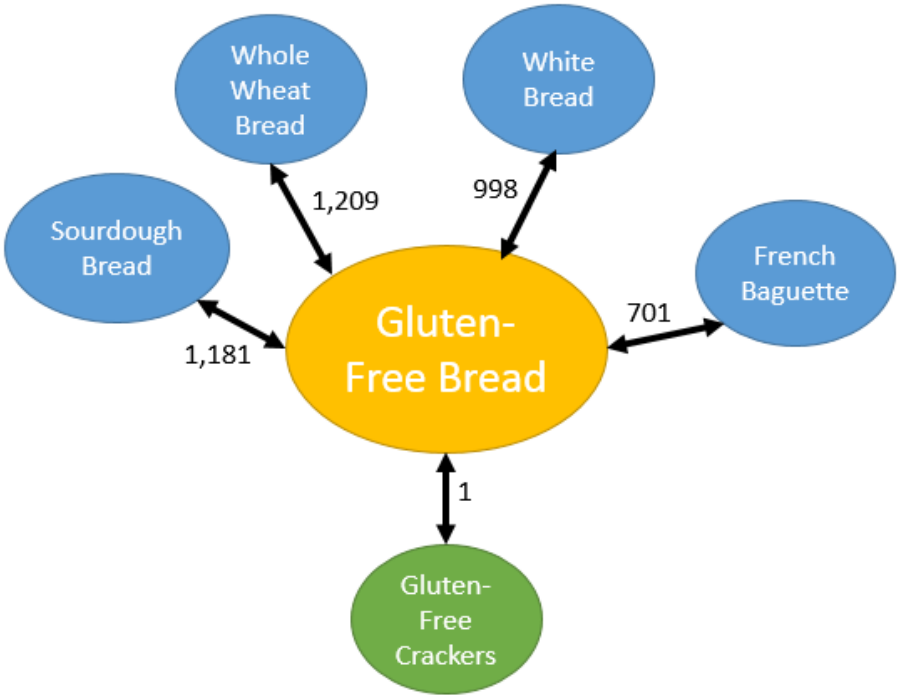


Figure 20: Count-Based Probability Problem

Clearly, some sort of probabilistic framework is needed to manage these edge relationships that will allow us to express the probabilities of edges independent enough of one another to avoid this issue. In [18], we see an example of a system that could potentially address this issue. In this paper, the authors propose and design a software framework for the declaration of variable of type T as Uncertain<T>, which allows their values to be conditioned on a probabilistic distribution instead of being given a concrete value, and provides additional comparative functions to compare the probability of a random variable taking on a certain value or range of values. One can imagine using this framework to assign edges a probability of occurrence instead of a count. We can refine our probability over time by increasing it on co-appearances. Additionally, to address the temporal nature of the problem, we could decrease it if objects are not seen together for a long time. The ability to decrease the probability is perhaps

the more interesting of the two improvements, as it is not as obviously represented in a count. However, as shown in the example above, the ability to both increase relevant relationship probabilities while also decreasing those that become less relevant is crucial when considering how such relationships will evolve over time. The framework for this code was not available at the time of writing, but it will be eventually and might be worth future consideration.

Additional Types of Sensor Data

An additional application we might consider comes from the rise of other sensors in our world today. In this paper, our focus has been on cameras, but even now there are other sensors readily available on consumer electronics. For example, some phones now have the ability to detect a heart rate or scan a finger print. A future where biological feedback is available for a user at any given moment is not difficult to extrapolate from our present circumstance, and with this comes a variety of new types of information that might inform our decision making. For example, humans may have palpable reactions to certain objects: a family member or pet might lead to feelings of excitement or relaxation; a spider could lead to anxiety or fear. Sensors might also be able to give us an intuition about the type of scene or object being observed. The use of a hypergraph allows us to continue to expand our ViCoNet to collect a variety of different types of relationship, and this is just one example of new types of data that may become available in the coming years in conjunction with the data from a camera.

C# Performance

Many of these ideas involve adding more information onto the nodes and edges of our hypergraph. However, a very practical concern related to this is exactly how this will affect the efficiency of our graph, especially as we begin to encapsulate more complex, hierarchical relationships. We have not been considering the importance of this as we work as our simulations are still far from real-time analysis and C# provides a powerful OOP framework for developing and testing the different ideas for our ViCoNet implementation. In the future, though, it may be important to migrate from the flexibility of C# to a more performance-oriented language. Though we have not explored this, the key observation to be made is that ViCoNet itself exists more as an idea than a specific implementation in a given language. C# allows us to evaluate the usefulness of various object relationships as we attempt to determine which will help us the most in terms of the vision pipeline. Once we have established which relationships are most useful, ViCoNet as a framework can be implemented more efficiently—though perhaps less easily—in a language meant for performance.

Energy versus Accuracy

Related to these programming language specific performance concerns, it may be important to identify places within ViCoNet where we can opt for accuracy versus performance depending on the specific setting. We have already seen how the passive and active modes exposed this tradeoff by allowing us the flexibility to run fewer classifiers and therefore conserve energy at the cost of performance. There are other areas within this framework that could be

exploited to provide a similar tradeoff. For example, the dynamic ViCoNet implementation in Chapter 5 demonstrated flexibility in this regard by allowing the user to set, among other things, the number of hops to search within ViCoNet for candidates. This could be set to a low value on an energy- or memory-constrained system, or could be set instead to infinity on a system with more available power. The value *n* that determines how many candidates to extract from ViCoNet could also be adjusted in the same way.

Integration in the Cloud

When we consider this sort of high-capacity learning with regards to wearables, it makes sense to additionally consider how something like ViCoNet might perform in a cloud environment. Up until this point, we have been discussing various forms of ViCoNet that are local to a particular device or user. However, there is some information that will become redundant across the individual ViCoNets of different users quickly, and if there is no centralization, users will need their device to learn basic relationships from the ground up regardless of if others have already done so.

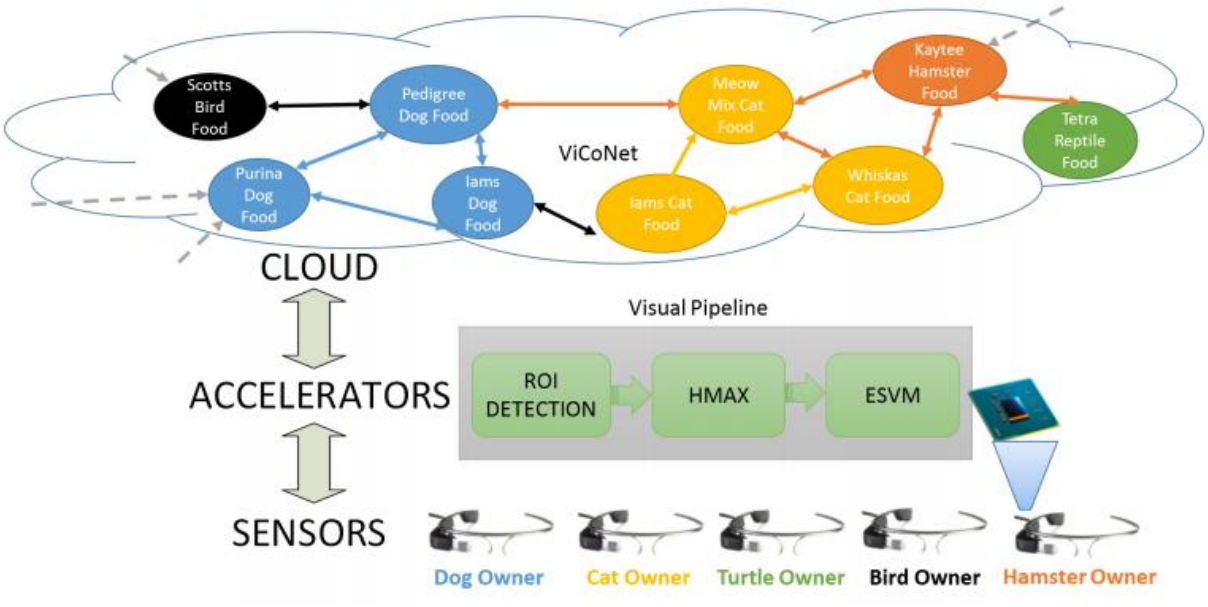


Figure 21: Cloud Framework

This knowledge could theoretically be pooled in the cloud by individual users willing to upload their personal networks. Figure 21 shows a simple diagram of five different users with different pet store shopping habits combining their disjoint knowledge of the store's layout into a more comprehensive network. There are obviously privacy concerns with this sort of sharing of personal data which would need to be considered thoroughly, but the idea at its base is a simple one that we believe greatly extends the learning power of any one device building a ViCoNet. As we discussed before, an individual's ViCoNet may have very personalized details relating even to biometric sensor data which will not really translate from person to person, but the basic details of objects' spatial relationships could be parsed out and shared to create a sort of "blank ViCoNet" available in the cloud for users to initially populate their devices. This would allow users to begin using the predictive powers of ViCoNet immediately, and over time a user's local version of ViCoNet would adapt and reweight the relationships to the user's individual preferences.

Settings beyond Retail

The work within this paper has focused entirely on a single setting with two different grocery stores as representatives of this setting. This is largely due to the difficulty involved in gathering and annotating datasets at the level of granularity required to establish the types of object relationships that we want to expose. There are a number of annotated object datasets available for use, but most do not have enough annotations per image to be useful, and the annotations they contain tend to be too coarsely grained to exploit the finer temporal and spatial relationship capabilities that ViCoNet can expose. Due to time limitations, we thought it best to focus our exploration within the datasets we created, especially since the aisles of a grocery store are so strictly organized. However, these same relationships that exist in retail can be found in the real world as well, if not so rigorously organized. We believe that this system already makes a powerful case for itself as a shopping aid, but much of our focus has been on how the choices we make are applicable outside of retail, and one of the most important next steps would be to take the time to put together a variety of datasets from different scenes and see if ViCoNet truly can hold up under a more complicated set of relationships. This ties into a variety of previously discussed matters, such as the object relationship weighting explored in [5] which was not restricted to retail and the use of GIST to help learn and determine scenes, potentially even in an unsupervised fashion.

While developing ViCoNet, we have tried to avoid making decisions that do not generalize well beyond a retail setting. For example, an important change between our original experiment in chapter 4 and the dynamic version in chapter 5 is that we eliminated the retail-specific concept of aisles and instead moved to a more general representation of scenes. While

any given scene still did represent images from a given aisle, the treatment of the problem changed to a form applicable to any sort of annotated scene instead of those arranged in discrete aisles. Additionally, though we considered using GIST as a tool to differentiate between different stores that might have different object relationships, we believe it might be better used at a much higher level, such as outdoor versus indoor, which would allow for a variety of different settings to make use of its discriminative power.

While these ideas represent a number of possibilities for ViCoNet, they certainly do not represent them all. As we move into a future where more and varied types of sensor information may be available at all times, we will find more opportunities for learning and improving our knowledge of object relationships, and hopefully this will lead to more accurate and efficient machine vision implementations.

BIBLIOGRAPHY

[1] K. P. M. W. T. F. a. M. A. R. Antonio Torralba, "Context-Based Vision System for Place and Object Recognition," *MIT AI Memo,* 2003.

[2] L. I. Christian Siagian, "Rapid Biologically-Inspired Scene Classification Using Features Shared with Visual Attention," *IEEE Transactions on Pattern Recognition and Machine Intelligence,* vol. 29, no. 02, pp. 300-312, 2007.

[3] Y.-G. J. A. H. C.-W. N. Jun Yang, "Evaluating Bag-of-Visual-Words Representations in Scene Classification," in *Proceedings of the international Workshop on Workshop on Multimedia information Retrieval*, Pittsburgh, 2007.

[4] J. S. a. A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, Nice, France, 2003.

[5] M. H. M. A. S. P. Y. C. R. J. H. D. F. Ali Farhadi, "Every Picture Tells a Story: Generating Sentences from Images," in *Proceedings of the 11th European conference on Computer vision: Part IV*, Crete, Greece, 2010.

[6] S. A. M. C. K. I. J. S. a. V. N. Brigid Smith, "Using a Visual Co-Occurrence Network (ViCoNet) for Large-scale Object Classification," in *Sensors to Cloud Architectures Workshop (SCAW) Proceedings*, San Francisco, CA, February 2015.

[7] A. T. K. M. a. W. F. B. Russell, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision,* May 2008.

[8] J. M. a. D. G. Lowe, "Object class recognition and localization using sparse features with limited receptive fields," *International Journal of Computer Vision,* October 2008.

[9] A. Maashri, "Accelerating neuromorphic vision algorithms for recognition," in *Design Automation Conference (DAC),* 2012.

[10] S. A. J. S. K. I. a. V. N. M. Cotter, "A hardware accelerated multilevel visual classifier for embedded visual-assist systems," in *International Conference on Computer-Aided Design (ICCAD),* November 2014.

[11] A. G. a. A. E. T. Malisiewicz, "Ensemble of Exemplar-SVMs for Object Detection and Beyond," in *IEEE International Conference on Computer Vision (ICCV),* November 2011.

[12] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM,* vol. 38, no. 11, pp. 39-41, 1995.

[13] L. Z. a. Z. D. Bo Yang, "C-HMAX: Artificial Cognitive Model Inspired by the Color Vision Mechanism of the Human Brain," *Tsinghua Science and Technology,* vol. 18, no. 1, pp. 51-56, 2013.

[14] H. S. a. T. Kanade, "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition," in *Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, 1998.

[15] D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol. 13, no. 2, pp. 111-122, 1981.

[16] K. M. H. W. a. E. W. Helmut AIt, "Congruence, Similarity, and Symmetries of Geometric Objects," *Discrete and Computational Geometry*, vol. 3, pp. 237-256, 1988.

[17] X. S. a. J. K. T. Neil D. B. Bruce, "Recurrent Refinement for Visual Saliency Estimation in Surveillance Scenarios," in *Ninth Conference on Computer and Robot Vision*, Toronto, 2012.

[18] N. M. T. M. K. S. M. James Bornholt, "Programming the Internet of Uncertain <T>hings," in *Sensors To Cloud Architectures Workshop (SCAW) Proceedings*, San Francisco, CA, February 2015.

ACADEMIC VITA

Brigid Smith
440 Glenside Road, Mountaintop, PA 18707 – bls5359@psu.edu

Education

Master of Science in Computer Science and Engineering
Bachelor of Science in Computer Engineering
Schreyer Honors College
The Pennsylvania State University, University Park, PA

Honors and Awards

Barnak Outstanding Junior Award (2014)
Girl Scouts of America Gold Award (2009)

Association Memberships

Member, Penn State Women in Engineering Program (2010 – present)
Participant, Penn State Engineering MIX program (2012)
Member, Eta Kappa Nu Honors Society (2012 – present)
President, Eta Kappa Nu Honors Society (2013 - 2014)

Professional Experience

Software Engineering Intern
Google, Mountain View CA (May 2014 – August 2014)
Software Engineering Intern
Amazon.com, Seattle WA (May 2013 – August 2013)
Software Engineering Intern
Xerox Corporation (May 2012 – August 2012

Publications

Brigid Smith, Siddharth Advani, Matthew Cotter, Kevin Irick, Jack Sampson and Vijaykrishnan Narayanan, "Using a Visual Co-Occurrence Network (ViCoNet) for Large-scale Object Classification," in *Sensors to Cloud Architectures Workshop (SCAW) Proceedings*, San Francisco, CA, February 2015.