5-2024

Context dependent training data selection for automatic target detection.

Tylman Michael
University of Louisville

CONTEXT DEPENDENT TRAINING DATA SELECTION FOR AUTOMATIC TARGET DETECTION

By

Tylman Michael
B.S. Computer Science and Physics (Dual), Union University, 2020

A Thesis
Submitted to the Faculty of the
J.B. Speed School of Engineering
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science in Computer Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2024

CONTEXT DEPENDENT TRAINING DATA SELECTION FOR AUTOMATIC TARGET DETECTION

By

Tylman Michael
B.S. Computer Science and Physics (Dual), Union University, 2020

A Thesis Approved On

4/19/2024

By the following Thesis Committee:

Hichem Frigui, Ph.D., Thesis Director

Olfa Nasraoui, Ph.D.

Ibrahim Imam, Ph.D.

Kevin Chou, Ph.D.

ACKNOWLEDGEMENTS

This thesis and my whole gradschool experience would not have been possible without a large number of people. First and foremost, my wife Rachel has supported from the very beginning at much personal sacrifice as I left my job to pursue education full-time. Additionally, I would like to thank her parents who opened their home to us for the first full year of graduate school during the housing crisis.

I would also like to thank my team at St. Jude Children's Research Hospital who gave me the time I needed to finish this semester strong. It's an honor to be able to perform such meaningful work with a group of some of the brightest minds in the world.

Thank you to my committee for agreeing to review my work and spend the time to evaluate my contributions.

Finally, and most certainly not least, I would like to thank my advisor, Dr. Hichem Frigui. This last year has had so many ups and downs for me personally, professionally, and academically, and I could not have done it all without his effective guidance and intellectual expertise.

Thank you all.

ABSTRACT

CONTEXT DEPENDENT TRAINING DATA SELECTION FOR AUTOMATIC TARGET DETECTION

Tylman Michael

4/15/2024

An Automatic Target Detection (ATD) algorithm is capable of identifying the location of targets of interest captured by Infra-Red imagery in vastly different contexts. ATD is often a precursor in a 2-stage methodology in order to ascertain the location and nature of a target in both military and civilian applications. In order to train an ATD algorithm, a large amount of data from varied sources is required. One drawback of this requirement is that some sources of data may harm the performance of the method in different contexts.

This thesis explores utilizing an unsupervised method to identify a subset of training data that optimizes the performance of a given test dataset. The methodology summarizes all available data by a set of visual prototypes. These prototypes are then used to map any collection of data into a single vector. The similarity between two collections can be quantified by comparing their mapped feature vectors using Jenson-Shannon (JS) Divergence.

We validate this methodology using a large Infra-Red object detection dataset consisting of various training and testing collections. For each test collection, we train the ATD using only the most similar training collections that have the lowest JS divergence. We show that exclusing dissimilar collections from training can improve the trained ATD model.

TABLE OF CONTENTS

LIST OF TABLES

TABLE **Page**

LIST OF FIGURES

FIGURE **Page**

CHAPTER 1

INTRODUCTION

Automatic Target Detection (ATD) in Infra-Red images (IR) with Deep Neural Networks (DNN) is an area of major interest in modern civilian and military systems. The goal of ATD is to provide fast, consistent, and accurate assessments of the location of targets of interest such as personnel, vehicles, animals, and more. In order to see consistent adoption, the ATD system must perform reliably across vastly different terrains, locales, and ambient temperatures. To achieve this, the ATD system needs to be trained on data collected from equally varied locations that are each laced with their own latent batch effects and biases. While these effects are often times minimal compared to the benefit gained by having more data, there is a possibility that the negative impact from the unique aspects of certain data could outweigh their benefits, leading to a net decline in the performance of the model for a specific task. A method that optimizes the compilation of training datasets for specific contexts may pave the way for a more tailored model deployment scheme, as opposed to a one-size-fits-all training approach.

One of the central assumptions for modern Deep Learning and Machine Learning methods is that the data used to train and evaluate the model must be drawn from a similar distribution as the data upon which the model will be applied. This is the "identical" in the"Identical and Independently Distributed" assumption referenced in many pieces of classic statistics literature. [1] In the context of ATD, it is important that the appropriate training data for a desired context be selected. Not all data may be equally suited for each specific context in which the model will be deployed. For example, it is entirely possible for data collected in the Arctic to negatively impact the performance of the system in a hot desert environment. In such cases, manual selection of the dataset would be understandable to remove any obvious clashes in setting. In other cases, certain classes of target may not be well represented in a given dataset that could be crucial for the final task. Although it would seem trivial to prune the training set by hand in those cases previously described, doing so may introduce human biases and will be bound by limitations in the operator's domain knowledge. It is also possible that the models may struggle gaining value from data that

humans would intuitively deem helpful due to how models "see" the world.

The cameras on which ATD methods are deployed frequently rely on Infra-Red frequencies of light ($\sim 2\mu m - 4\mu m$ for Mid-Wave, and up to $100\mu m$ for Long-Wave) for gathering their input data possess their own unique strengths and weaknesses which are beyond the intuition of most human operators [2]. IR cameras do a fantastic job of visualizing and capturing heat. This is useful for targeting vehicles as all engines create heat while in use. While this is a fantastic strength to identify active vehicles and personnel, it comes with tricky problems of its own. For example, the gradient of heat on a vehicle running in the cold snow-blinding sub-zero temperatures of the arctic is much different than the gradient of heat on the very same vehicle running in the blazing sun of the desert. In one we can expect to see an isolated hot spot of the engine which is quickly being cooled, and in the other we can expect to see a more uniform distribution of heat as the whole vehicle is heated by the sun. In this case, while a human could clearly see that these vehicles are the same in the visible spectra, IR imaging may struggle to fully capture the similarity of the vehicles due to environmental effects. Likewise, an image which may fool a human in the visible spectra due to camouflage or obfuscation by thin foliage will be easily identifiable in IR.

Modern computer vision models employ a wide range of methods for translating images from a human-understandable pixel space representation to a computationally effective feature space representation. Choosing a training subset by a method driven by the feature representations of the available data collections will provide a much closer "apples-to-apples" comparison of how the datasets will behave in ATD as opposed to what a human can do by hand. Employing an empirical comparison method will mitigate any chance for operator bias to influence the selection process, but caution must be taken to avoid the pitfall of cherry-picking individual training samples to maximize performance on a test set. Selecting the dataset too finely by only keeping the easily described samples will critically lower the model's ability to generalize due to overfitting. Thus, the problem of selecting the best subset of available data to train a model for a given context can be reduced to assessing the similarity between data collections at a coarse level

A common way to compare the similarity of datasets empirically is through statistical methods, such as the statistical moments of their distributions. However, these methods are not well suited to describe how Deep Neural Networks capture the high dimensional, long-tailed nature of computer vision datasets. As described in a recent paper [3], Deep CNNs memorization of the rare representations of classes can counter-intuitively influence the generalization of the model for the better. The samples which exist in the long-tails of the class distributions have been shown to

possess some of the strongest influence on the behavior of the model in testing. Intuitively, it could be understood that samples from the center of the distribution are possibly redundant due to the likelihood that the dataset will have many other samples containing the same features, but a unique representation of a class by a long-tail sample will provide more information that can bridge the gap in generalization. Unfortunately, it is these samples that exist in these long-tails of the datasets that would be least described by broad statistical measurements like the first moments of the distribution. Because of this long-tailed nature of large computer vision datasets, standard statistical descriptions of distributions lack the power to describe the differences and similarities of datasets without likely stripping them of important nuances and semantics which may have a strong influence on the final model.

The method chosen for selecting the training data must be done at a high level with broad data collections in order to prevent reducing the model's capability of generalizing, while still enabling the adaptation to a specific context. The manner in which the whole dataset available for training is split into these high level collections is something that will be application specific, and a 'one-size fits all' rule cannot be easily defined. In our case, our data has been collected over a wide variety of locations and times in a small number of large batches. Splitting along these groups, referred to as collections, we can reasonably identify collections which will better prepare our models for a specific task without running the risk of pruning our datasets too finely. However, even with the high-level collections of data separated intuitively by location and time, it is not enough to qualitatively select data by hand as this can depend on clutter, targets, and other unknown conditions. Instead, we propose utilizing the feature representations of the images themselves to reliably quantify the similarities of the collections without bias in a completely unsupervised way.

An unsupervised approach is the best choice when comparing the similarities of our collections fairly, effectively, and quantitatively. An unsupervised method capable of quantifying dataset-level similarities across high-level collections of data will be capable of increasing task-specific performance while avoiding overfitting and cherry-picking. In this Thesis, we will be investigating the efficacy of utilizing a feature-based unsupervised similarity estimator for selecting collections to train an ATD method. Our proposed approach will be validated using YOLO object detection network.

1.1 Declaration of Contribution

One method [4] for accomplishing dataset-level similarity comparisons extends the common Bag-of-Words representation in Natural Language Processesing to computer vision tasks by using

image prototypes instead of words. Datasets expressed in this representation are then compared using the Jenson-Shannon divergence. We refer to this combination of as BOP+JS

The original work which proposes BOP+JS for dataset-level applications explores two possible uses: comparing training suitability of datasets, and assessing test set difficulty without labels. This thesis is an application and extension of the first use case. Previously, the original work focused on validating that applying BOP+JS between individual training and testing sets produces a similarity score that is highly correlated with classification accuracy of models trained and tested on the same sets. The focus of this thesis is to explore the utility of leveraging this correlation to combine multiple training sets to build optimal subsets of training collections to increase performance for individual test sets by attempting to answer 4 research questions:

- RQ1: Can the coarse BOP+JS representation distinguish between different target types?
- RQ2: Can BOP+JS identify similar collections?
- RQ3: Will training on the most similar datasets outperform training on the least similar datasets?
- RQ4: Will greedily building a training set from the most similar datasets outperform training on all datasets?

CHAPTER 2

RELATED WORK

2.1 Background

Training datasets form the backbone of modern Machine Learning applications. As such, datasets have earned the investment of an increasingly large amount of effort into researching ways to better understand and clean them [5]. Works related to representing, comparing, and ultimately pruning datasets are plentiful and diverse, with each step building on the last. This thesis's ultimate goal of dataset-level selection is built by naturally progressing from representation to comparison to selection.

2.1.1 Data Selection

Speaking broadly, selecting optimal subsets of training data is not a novel concept and has been done in a myriad of ways throughout the history of Machine Learning. A prerequisite of good data-driven methods is good data; as such, methods ranging from the trivial manual scraping of erroneous samples to highly-advanced influence estimation techniques [6] have been employed to select a better subset of the available training data by removing poor samples.

One example of advanced dataset pruning in the context of deep learning utilizes Koh and Liang's Influence function [7] to estimate the influence of individual samples on the parameter change of the network. Then, they construct a subset of minimal size by selecting the smallest amount of samples whose inclusion will result in a desired predicted amount of change. The goal of this method is to identify and remove individual redundant samples, which will result in equivalent performance with a smaller dataset. This is extremely useful in very large experiments, and the work resulted in a 40% decrease in size of the CIFAR-10 dataset with only a 1.3% accuracy decrease. While this method seems to accomplish something similar to our goal of dataset selection, it intends to maintain accuracy while lowering dataset size, rather than task-specific improvement of performance.

Another method [8] adjacent to the task of assessing similarity focuses on quantifying the structural regularity of individual samples. The intuitive idea of structural regularity is to capture

how common a representation of a sample is compared to other members of the same class. Taking the image class of a tennis ball as an example, a high-def closeup of a tennis ball sitting on a tennis court would earn a high regularity score, while an action shot of a large golden retriever catching a small tennis ball would score an extremely low regularity score. This concept of structural regularity is similar to our concept of dataset similarity, but this method is based on quantifying how an individual sample compares to other samples within the same class. This focus on individual samples makes this method difficult to apply for dataset-to-dataset comparison, and the supervised nature of within-class comparison disqualifies this method from being a candidate for our application. A more appropriate approach for our purposes is to re-frame the problem of data selection as a similarity estimation between distributions of subsets of the data.

2.1.2 Data Representation

Before two datasets can be compared, they must first be represented in a computationally effective way. One way to represent text data for deep learning tasks is with a Bag-of-Words (BoW) representation [9]. This method of representation has proven to be effective in Natural Language Processing (NLP), and continues to be applied in many other modern methods [10].

The BoW represents text data by mapping the text into a histogram counting the frequency of each word that appears in the document regardless of order. This representation encodes individual samples into a single vector of counts that is both convenient for computation and deceptively powerful. Multiple improvements to this basic idea have been developed in the field of NLP, such as n-grams and term frequency-inverse document frequency [11]. Due to its success in NLP, the BoW approach has been adopted to Computer Vision tasks [12].

One of the first methods used to extend BoW to Computer Vision tasks was to encode large numbers of descriptors of localized features of individual images into a histogram vector [13]. These features were clustered together and assigned to their nearest cluster center. This process would result in each image having a bag of "visual words" made up of the counts of each local descriptor's nearest cluster. This method effectively extends the BoW representation of NLP to Computer Vision tasks. However, since it focuses on characterizing individual images, it cannot be used to represent image collections.

The BoW naturally lends itself to characterizing individual samples over large datasets. For our goal, we need a method that can characterize groups of samples into a singular vector that can be used for comparison. Instead of using a representative list of "words" to describe a single image, we

treat each sample as an individual word to contribute to a histogram of the whole dataset. Treating an individual image in this way will result in a flexible BoW-style histogram representation capable of describing predefined sets of data compactly, which can be used to quantitatively compare their similarities.

2.1.3 Data Comparison

Given a compact representation of a dataset as a histogram, we have a few options to compare them. One of the most fundamental ways is to think of the histograms as a discrete probability distribution and to compare their divergence. Treating the datasets in this way will normalize the distributions to allow for direct comparison even of datasets with notable differences in size.

One of the most popular methods for comparing probability distributions is the Kullback-Leibler (KL) Divergence. The KL divergence measures the relative entropy between two discrete probability distributions as follows:

$$KL(P||Q) = \sum_{x \in \chi} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{2.1}$$

The KL captures the divergence between the two distributions, P and Q, by taking the expectation of the log of ratios (or logarithmic difference) of the two distributions over all available values χ. One key issue with the KL divergence is that it is a *divergence* and not a distance metric. A divergence has some features that are undesirable for the task of selecting datasets. Most notably, the KL divergence is unbounded and asymmetric. This can lead to some difficulty interpreting the results and making dataset selections as some dataset pairings may result in divergence values orders of magnitude higher than other comparisons, while their true effectiveness for the tasks may likely be similar. Additionally, the lack of symmetry can lead to results that are unnecessarily challenging to understand intuitively.

2.2 Bag of Prototypes Representation for Data Selection

While there are many ways to compare similarity or analyze structures of datasets, it is desirable to use an unsupervised method which will compare dataset-to-dataset similarity at a very high level. The representation methodology which meets all of our criteria that will be the focus of this thesis is known as "Bag Of Prototypes" (BOP) [4]. This method is very similar to the common "Bag Of Words" representations of text data, but it is extended to represent image datasets using

the individual samples in the datasets as words. Instead of counting the amount of times a word appears in a document, this method generates representative "Prototypes" and counts how many samples are assigned to each prototype. The final result of a dataset represented in this way can be seen as a histogram of counts for each prototype, which can then be compared to another dataset with Jenson-Shannon (JS) Divergence [14] to quantify the similarity between two large sets of data quickly and effectively. We refer to the combination of representing data with Bag-Of-Prototypes and comparing them with Jenson-Shannon divergence as BOP+JS.

The original work in which this method was proposed focused on investigating the correlation of an individual training collection's BOP+JS score on a given task with the performance of a ResNet-101 model architecture trained on that set. They showed that BOP+JS is predictive of the suitability of an individual training set on a few specific tasks with high correlation. The original work also investigated the reverse relationship to show that BOP+JS can predict the test set difficulty without labels.

CHAPTER 3

DATASET-LEVEL CHARACTERIZATION WITH BAG OF PROTOTYPES

3.1 Objective and Approach

This thesis addresses the task of training an ATD that can adapt to different contexts. We assume that a large and diverse training data is available. We postulate that different subsets of the training data should be selected to train the ATD and adapt it to different contexts.

Our datasets are made up of collections of videos that contain targets of interest with recorded ground truth. Each collection contains multiple videos of different lengths and different targets, collected at a given site and within a short time frame. Our data is partitioned into 15 training collections and 12 validation collections. These collections vary greatly in content beyond just targets due to temperature, background, camera type, etc. Additionally, these collections vary greatly in size, with some of the larger collections containing over 100x the number of frames as the smallest collections. We train our ATD method by taking each of these videos and sampling them by selecting one frame every R_c frames. R_c in this case is referred to as the "Frame Sample Rate of collection c" and is selected for each collection to reduce the large collection imbalance. The selected image frames are then treated as individual images, referred to as samples, to train our ATD method. Our objective is to select an optimal subset of collections from all 15 available training collections. We will select subsets of varying sizes from these 15 training collections to train an ATD with a goal of optimizing the performance on each validation collection individually.

The first step of our approach consists of representing each collection by a single feature vector in order to compare collections and quantify their degree of similarity. The mapping is data-driven and relies on features extracted using a pre-trained Convolutional Neural Network (CNN). This CNN uses target image patches as input and is trained and optimized to identify different target types. In order to use this CNN, we first detect and extract image patches of high-interest areas from the image frame with a pre-trained YOLO network [15] as shown in figure 3.1. Next, we use a clustering algorithm to summarize the extracted features from all image patches of all training collections by K representative prototypes as shown in 3.2. Finally, the K-prototypes are used to

Figure 3.1: Overview of our process to translate a single full image into multiple YOLO-Detected feature patches.
Picture Credit: Carter Sprouse

map the content of each collection to a K-dimensional feature vector.

3.2 Extraction and Representation of Image Patches

Typically, the raw data captured by IR cameras contain few small targets in a vast background. Image patches of targets can be extracted by utilizing the available ground truth information. Alternatively, they can be extracted in an unsupervised way using a pre-trained ATD network. In this thesis, we report our results using the latter approach. Not relying on ground truth labels allows our approach to be as general as it can be used even when the collected data is not labeled.

We use a version of YOLO [15–21] to identify the location of potential targets as the first step before passing on those patches for further processing. We use a YOLO model that has been

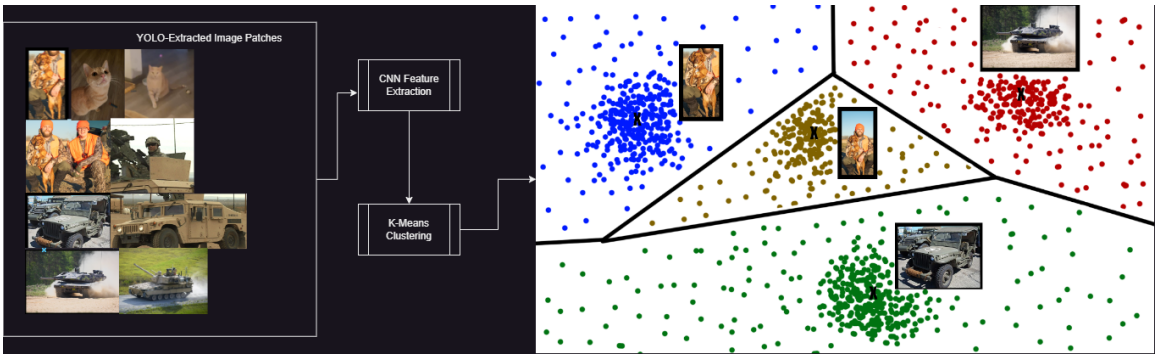

Figure 3.2: Overview of our process to summarize multiple image patches by K representative prototypes

trained on the same training data collections. We initialize the network with pretrained weights on the MS-COCO dataset [22], using the DarkNet code base. Data augmentation is performed on the training image patches including scalar augmentation, intensity inversion, random horizontal flipping, and random cropping.

YOLO is fast, robust, and can potentially meet the requirements of real-time object detection, which makes it well suited for ATD. Using a YOLO model to generate the image patches, that we will consider as individual images, allows us to remove the majority of trivial background information that would bias our unsupervised method and also have a methodology that we can apply to labeled and unlabeled data alike. Furthermore, YOLO is likely to identify multiple patches when the targets are difficult to identify. This can lead to larger and more diverse data, delivering better granularity specifically for small datasets. As such, utilizing image patches around targets will improve our unsupervised comparison method's ability to tease out important differences and similarities without the high volume of background information drowning them out. We can then use these unlabeled YOLO patches as the basis for comparing the collections they hail from. In addition, we can use their connected ground-truth labels, if available, to validate our method is capturing high-level semantic information. After extracting the image patches, we use a deep classifier model to map them to feature vectors. Out model is based on vision transformers [23] and was trained for discriminating between target types which makes it a prime choice for characterizing these patches.

3.3 Building the Mapping Codebook

The backbone of the BOP representation of a dataset utilizes a feature extractor to translate the images from pixel space to feature space, and then uses these features to learn the prototypes.

The method for feature extraction will be application specific, but it is recommended to use a similar feature extractor as the model whose performance we are intending to improve. The set of points in feature space that serve as the prototypes are the clusters' centers learned by clustering all of the training data. This set of points, referred to as the "codebook," will be used to map any split of our data.

After learning the codebook, we take all of our collections and assign each individual sample in them to their nearest prototype. Next, for each pre-defined collection of samples, we count how many times each cluster appears in that group. The data can be split into any combination of collections depending on available metadata. This versatility offers several advantages as it allows us to perform any number of group-to-group comparisons without recomputing the feature representations, codebook, or cluster assignments. Additionally, as long as the feature extractor and clustering model remain unchanged, this can be easily expanded to new datasets without affecting any previously discovered relationships. In our work, we will be focusing on a high-level split based on collection location and time; We will also experiment with another split based on the targets' class to provide an illustration and validation of our approach.

Let $X^{Tr} = \{x^1, x^2, \ldots, x^N\}$ refer to the set of N image patches extracted by YOLO from all training collections in their native pixel space. Each patch in X^{Tr} is fed to the CNN and mapped to a feature vector, x_f. Let $X_f^{Tr} = \{x_f^1, x_f^2, \ldots, x_f^N\}$ refer to the features extracted by the CNN for all images in X^{Tr}. X_f^{Tr} is an $N \times F$ matrix where N is the whole number of patches in X^{Tr}, and F is the dimensionality of the feature space. Next, we use a clustering algorithm to cluster X_f^{Tr} into K clusters. The K clusters provide a good summary of the entire collection, and are referred to as "Prototypes." The set of all K prototypes is referred to as the "Codebook."

The codebook is then used to map a given collection into a K-dimensional vector. Let $c_x = \{x_f^1, x_f^2, \ldots, x_f^{N_c}\}$ represent the features of the N_c image patches extracted from a given collection c. Each x_f^i is assigned to the closest cluster k^i in the codebook using

$$k^i = \arg \min_{j=1\ldots K} dist(x_f^i, k^j)$$

Next, we map the feature vector x_f^i, representing patch x^i, to a K dimensional vector, x_m^c, using

$$x_m^i = \{x_{m1}^i, x_{m2}^i, \ldots, x_{mk}^i\}$$

where

$$x_{mj}^i = \begin{cases} 1, & \text{if } j = k^i \\ 0, & \text{otherwise} \end{cases}$$

Finally, collection c will be represented by a K-dimensional vector, c_m, using

$$c_m = \frac{1}{N_c} \sum_{j=1}^{N_c} x_m^j \tag{3.1}$$

In other words, collection c will be represented by a vector that represents the cluster assignment probability of all its image patches. We will refer to this representation defined in 3.1 as the "Bag of Prototypes" or BOP.

The above methodology for vectorizing a collection is versatile, and augmenting it for different purposes is computationally efficient. At the core, a valid BOP representation of our data is created by assigning feature representations of image patches to their nearest prototype and then grouping these assignments by some criteria. In the previous example definition we applied the method at a collection level, but once the image patches are assigned to their prototype, we can build the final discrete probability vector by aggregating on any criteria. As such, the clustering model, image features, and prototype assignments only need to be calculated once to be used repeatedly in different experiments. In this thesis, we will primarily group our data by collections as described above, and similarly by targets $t_x = \{x_f^1, x_f^2, \ldots, x_f^{N_t}\}$ where N_t is the number of patches belonging to the target type t.

3.3.1 Similarity Comparison with Jenson-Shannon Divergence

Once our data is split into collections and the counts of each of the K clusters are gathered, we divide by the total number of samples to convert each frequency into a percentage. This normalization converts the histograms into discrete probability distributions and allows for better comparison between collections of differing sizes. Next, the normalized histograms can be compared using Jenson-Shannon Divergence. JS Divergence is a smoothed, symmetric, and bounded extension of the Kullback-Leibler Divergence perfectly suited to capturing similarity between two collections. JS Divergence is formulated as follows:

$$JS(P||Q) = \frac{KL(P||M) + KL(Q||M)}{2} \tag{3.2}$$

Where P and Q are the normalized histograms for two collections, and M is the mean of the two probability vectors. $KL(X||Y)$ in this notation is the Kullback-Leibler divergence of X vs. Y as defined in eq.2.1. Another, less obvious, benefit of the JSD in eq.3.2 is that its bounds are based on the base of the log used to compute the underlying KL divergence. If we choose $\log_2$, then we can

bound our scores from 0 to 1, with 1 being maximally divergent and 0 being identical distributions. We refer to the method of using JS as defined in eq. 3.2 on datasets represented by BOP as BOP+JS.

3.4 Evaluation Strategies

The goal of this thesis is to explore the validity of BOP+JS dataset comparisons in IR ATR. Due to the flexibility of this method, we can group our individual image patches in different ways to interrogate separate capabilities of this method. We aim to answer 4 Research Questions (RQ's) to accomplish this:

3.4.1 RQ1: Can the coarse BOP+JS representation distinguish between different target types?

Motivation: Before we use our BOP for selecting relevant training data, we test our hypothesis that the BOP+JS comparison method is capable of capturing high-level semantics in our datasets. We propose achieving this by grouping our images by their true classes into high-level collections of similar items for comparison to one another. Doing so will allow us to qualitatively ensure that classes containing targets that share real-world similarities are also behaving similarity under our methodology.

Approach: Our data has many target types that are too specific for this question. So, to lower the number of spurious comparisons and obfuscate sensitive classifications, we grouped our many classes into 15 superclasses that capture the broad similarities of the classes. For example, instead of labeling the specific model of civilian vehicles, we grouped all civilian vehicles into one superclass. These 15 target classes and their abbreviations are defined in table 3.1:

For all our experiments in this thesis, we separate our data into a reference (ref/r) set and a query (qry/q) set. The reference set is comprised of all the feature patches that were used to build the BOP codebook and the query set is comprised of all other patches. Let $c^{s-g} = \{x_f^1, x_f^2, \ldots, x_f^{N_c}\}$ refer to collection c that is comprised of N_c feature patches quantized to their nearest prototype in the codebook that belong to superclass s and are in group $g \in [\texttt{r}, \texttt{q}]$. We refer to the full set of available superclasses described above as $S = \{\text{A, CW, FV, } \ldots \text{, P, U, UV}\}$. Finally, let c_m^{s-g} represent the k-dimensional normalized histogram representation of the collection. For example, all feature patches which contain a person and were not used in training the codebook are mapped by our codebook to the k-dimensional BOP vector c_m^{P-q}.

Next, we use Jenson-Shannon Divergence to measure the divergence between all pairs of (ref,

TABLE 3.1

Class Names and Descriptions

Name	**Abbv.**	**Description**
ANIMAL	A	Individual animals such as a deer, squirrel, rabbit, etc.
Civilian_Wheeled	CW	All common civilian wheeled vehicles such as vans, cars, and pickup trucks.
Flying_Vehicle	FV	All vehicles which fly, regardless of military or civilian designation, such as airplanes and helicopters.
GROUP_ANIMAL	GA	Clusters of animals too close together to separate, such as a tight herd of cattle or a flock of geese.
GROUP_VEHICLE_MIXED	GVM	Clusters of vehicles with mixed tracked/wheeled types, such as a wreckage yard or military convoy.
GROUP_VEHICLE_UNKNOWN	GVU	Clusters of vehicles that could not be specifically identified due to occlusion or any other reason.
GROUP_VEHICLE_WHEELED	GVW	Clusters of wheeled vehicles, such as civilian rush-hour traffic, parking lots, or unarmored convoys.
Industrial_Vehicle	IV	Large, non-military work vehicles, such as tractors and trains.
Military_Tracked	MT	Military vehicles propelled by tracks instead of wheels, such as tanks.
Military_Wheeled	MW	Military vehicles propelled by wheels regardless of armor, such as humvees.
Non-Target	NT	The label given to erroneous YOLO boxes with less than a .5 IOU overlap with a true target.
OTHER	O	A few sensitive classes that could not combine into the other groups and must remain unspecified.
Person	P	All personnel, regardless of their location, equipment, or proximity to others.
UNKNOWN	U	Locations where something is present, but could not be positively identified as anything in particular.
Unknown_Vehicle	UV	Locations which definitely contain a vehicle, but the type of vehicle could not be positively identified.

qry) collections as follows: Let D represent the $R \times Q$ matrix where R and Q are the number of reference and query collections respectively. Then let $D_{ij} = JSD(c_m^{i-r}, c_m^{j-q}$ where $\{i, j\} \in S$.

This Research Question will be answered by comparing the relative similarities of a few key superclasses in reference to the rest of the target types in three different experiments. First, we compare our two meaningless and high-entropy superclasses, UNKNOWN and OTHER, to investigate their relationships with the more concretely defined classes. Second, we isolate our most discriminable and semantically unique superclass, PERSON, to investigate the relationship between this individual class all other classes. Finally, we select two superclasses sharing moderate intuitive similarities, civilian wheeled vehicles and military wheeled vehicles, to investigate if their high-level intuitive similarities are present in their divergence measurements.

3.4.2 RQ2: Can BOP+JS identify similar collections?

Motivation: Once we have explored the effectiveness of BOP+JS for characterizing similarities between semantically-intuitive target types, we must perform the same experiment at the collection level. This will be much more difficult, as these collections have been made with the intent of providing quality training data with minimal biases and all contain similar target information. As such, we must show that BOP+JS can characterize similarities between more difficult comparisons.

Approach: Due to the flexibility of BOP+JS, we can apply the same approach as defined for RQ1 to address RQ2 by simply grouping our data via collection of origin instead of by target. As before, let $c^{s-g} = \{x_f^1, x_f^2, \ldots, x_f^{N_c}\}$ refer to collection c that is comprised of N_c feature patches quantized to their nearest prototype in the codebook that belong to superclass s and are in group $g \in$ [`r, q`], and the full set of superclasses be referred to as S. The only difference between this experiment and the target experiment is that our superclasses are now defined by the method in which the data was collected. The full list of collections S is defined as $S = \{$v0,v1, ..., v10, v11, v13, v14, v15$\}$. These names are standardized within our research group and are obfuscated to maintain privacy, and v12 was omitted from these experiments. Once this is done, we will perform similar comparisons as in RQ1, except instead of comparing hand-picked target types, we will be comparing collections of data taken at the same location at different times of year.

We will evaluate the effectiveness of BOP+JS for distinguishing collections of data through three experiments. First, we will investigate v13 as that is a unique collection which only contains targets belonging to the Person class to determine if this uniqueness is preserved in our experiment. Second, we must show that there is perceivable self-similarity between the validation sets with matching training sets by investigating the diagonal of the matrix D. Finally, we will investigate a few collections which were collected at the same locations either back-to-back or within a month of one another to determine whether these similarities are preserved.

3.4.3 RQ3: Will Training on the most similar datasets outperform training on the least similar datasets?

Motivation: Before we attempt to build a larger dataset of only the best collections and compare them to training on all of our data, it is important to establish a proof of concept that using BOP+JS as a method for selecting training data will have an effect on the final performance of our model.

Approach: We compare the results of training an ATD model on the best collections vs. an equivalently sized dataset made of the worst collections for a given query set s. Let D_s be the column of D corresponding to query set s. We then build a training set $T_s^{\pm i}$ which contains the i lowest-scoring $(-)$ or the highest-scoring $(+)$ reference sets in D_s. We compare the performance of these two models to ascertain if using BOP+JS as a heuristic for greedily building training sets for ATD models will have an effect on the performance of those models.

3.4.4 RQ4: Will greedily building a training set from the most similar datasets outperform training on all datasets?

Motivation: The end goal for our investigation of BOP+JS is to improve our models performance on specific validation collections by selecting relevant subsets of training data. If BOP+JS proves effective at selecting relevant training datasets, then a door may be opened for deploying models specifically trained with their deployment context in mind, boosting performance in the field.

Approach: We take the same experiments as in RQ3, except we only consider those models built off of the low-scoring collections. Instead of comparing the performance of the models in a head-to-head fashion, in this experiment we track the performance of the models as we add more and more collections: $i = 3, 6, 9, 12, 15$ to see if performance improves or deteriorates as we add collections to the training set.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Comparison Methodology

In the context of BOP, the terms of "Train" and "Test" can possibly be a bit misleading. In the context of training a model, these terms are clearly defined as data which were used to create the model and data used to evaluate the model, but in many BOP contexts, this is not necessarily the case. As described in this section, the core of the BOP codebook is a KMeans model that is used to assign every individual image patch to their nearest cluster. In this way, the KMeans model is being used as a means to represent data regardless of the involvement of the data in training the KMeans model. The key nuanced idea that must be understood is that we use BOP to compare a single query collection of interest with any arbitrary set of other reference collections even if none of them were used to train the model. In this thesis, we treat our test collections as the query collections and the train collections as the reference collections, but it is important to know that this is merely a symptom of the context of our application. BOP is equally valid and interpretable when used to compare any two collections of data regardless of if it was used to train the model or not. For example, one could theoretically use BOP to compare the similarity of a novel collection to the previously test collections in order to predict their performance. As such, we've made the intentional choice to refer to collections as "reference" (or ref) and "query" (or qry) in some figures and sections to acknowledge the flexibility of this method beyond our use case, but for the specifics of our use case reference/train and query/test are interchangeable for each other.

Our dataset consists of 15 separate reference collections which each contain targets belonging to 15 target types. The collections were collected with some combination of different times, locations, hardware, and target content. 12 of these 15 reference collections are accompanied by their respective query collections which were collected close to the same time with the same location, hardware, and target content. Once the data is separated into reference or query, we can leverage the flexibility of BOP+JS to group the data by collection or use the available ground truth to group by target. Our goal for grouping by target is to present a proof-of-concept that BOP+JS can characterize intuitive

high-level similarities on IR target patches. Our goal for grouping by collection is to select the best subset of reference collections to optimize performance on a given query collection. We do so by using BOP+JS as a heuristic in a greedy method to grow the training set by iteratively adding the lowest divergence reference available, and track the performance of an architecture as the training set grows to include all data.

The above experiment to use BOP+JS as a heuristic cannot be validated quantitatively as different collections may have some inherent similarities that we cannot predict. To validate, we perform two intermediary experiments where we first show that the proof-of-concept high-level target similarity experiment is extendable to collections. Next, we perform our greedy aggregation of training collections twice, once as described previously, and once in reverse to build the worst training set by adding the highest divergence references. The performance of the same architecture trained on these two opposite greedy training sets are then compared to show that a model trained on low divergence collections will outperform a model trained on an equivalent high divergence training set. These two intermediary experiments will provide additional context to aid interpreting the results of the final experiment.

4.2 Aggregating Training Sets

To train a task-specific model for each of the query collections considered, we select the training collections in multiples of 3 which score the lowest or highest divergence scores to train our ATD models. These training collections are then sampled at the closest integer rate in which we can achieve a satisfactory number of frames in the training sets while also maintaining an even split in collection representation. Let A be the target dataset size for all experiments. Then, let $S^c_{\pm i}$ refer to the training set aggregated for a given query collection c with i separate reference set members, such as S^{v6}_{-3} referring to the training of the 3 lowest divergence reference collections for the query collection v6. In order to maintain a balanced dataset, let R_r refer to the sample rate of an individual reference collection $r \in S^c_{\pm i}$. We calculate R_r as follows:

$$R_r = \max(\lfloor \frac{iN_r}{A} \rfloor, 1)$$

where N_r is the number of available frames in reference collection r. R_r is rounded down to the nearest integer, unless $R_r < 1$ in which case it is rounded up to 1 which corresponds to selecting every frame in a reference. Selecting the sampling rate in this way allows each collection in $S^c_{\pm i}$ to represent $\frac{1}{i}$ of the target training set size A as closely as possible, regardless of the relative dataset

sizes within $S^c_{\pm i}$. An exception to this is when $N_r < \frac{A}{i}$, in which case 100% of the available frames in r are used and the training dataset cannot reach the size desired. This is a middle ground between forcing a training set to meet the size goal A by oversampling from the larger references at the cost of reference imbalance, and forcing a training set to maintain perfect reference balance at the cost of lowering the training set size to that of the smallest member.

We then train our ATD model on the selected training sets for 16 epochs and select the best epoch of the 16 using an automated selection method based on the PD@0.1FAR, PD@0.5FAR, and the AUC. PD stands for "Probability of Detection," which is the probability that a ground-truth object in the test set will be positively detected by our model. FAR stands for "False Alarm Rate," which is the probability that our model will return erroneous detections that do not contain any targets. Finally, AUC stands for "Area Under the Curve," which is the area under the ROC curve made by calculating the PD while sweeping through a possible FAR range.

The range of interest for possible FAR in our methodology is 0.1 and 0.5 FAR. These two endpoints are key milestones that must be kept in mind with equal importance as the overall AUC between them. Normally, the best epoch for an individual model are selected by hand to evaluate the trade-offs of performance at different FAR for the specific task required, but in our case, we have a large number of experiments and we needed to avoid the possibility of human bias.

The PD at 0.1 FAR, PD at 0.5 FAR, and the AUC all have significantly different distributions, which makes simple aggregation of the 3 metrics not a good option. In order to select the best epoch for a given model, we propose a method which takes these metrics for all 16 epochs and expresses them as a percentile within their own distribution. Then, we take the mean of the 3 percentile-represented metrics for each epoch and select the highest scoring model of all of them. Let p_i^m be the percentile representation of metric $m \in \{\text{AUC, PD@0.1FAR, PD@0.5FAR}\}$ for epoch i. We then select our best model as the one obtained after epoch I, where

$$I = argmax_i\left(\frac{p_i^{\text{AUC}} + p_i^{\text{PD@0.1FAR}} + p_i^{\text{PD@0.5FAR}}}{3}\right)$$

This methodology is applied for all experiments involving model training to ensure a consistent comparison between models.

4.3 Answers to Research Questions

The JS scores are confined to the $[0, 1]$ range. However, their distributions with this range depend on the data and the extracted features. To provide an idea of the relative values for our

TABLE 4.1

Broad Qualitative Descriptions of JSD scores

JSD Range	**Qualitative Description**
$< .05$	Nearly identical distributions
.05 - .15	Strong Similarity, causally linked
.15 - .25	Ambiguously Similar, may contain common contexts
.25 - .35	Dissimilar, but may have a few related concepts
.35 - .45	Very Dissimilar, with no common concepts
.45 - .55	Strong Dissimilarity, may be caused by extremely different environments
$> .55$	Datasets are completely different, may be caused by no overlap in target content or prevelance of obstructed data

data, in Table 4.1, we display a broad qualitative description for different JS ranges.

We display the JS scores for a given query against all available references in a spider plot such as in figure 4.1, which shows the JS scores for all targets. The names of the reference sets are displayed by their corresponding spoke on the plot, and the name of the colored query lines are given in the legend on the right. When we investigate specific queries they will remain on the visible on the plot and in the legend while all other queries will disappear from the plot and have their legend icons greyed out.

The original work in [4] shows that the method is robust to the number of clusters used, so the choice of codebook size is not of great importance in this thesis. However, for completeness, the BOP codebook that is used to generate the representations for this thesis is developed by training a $K = 256$ cluster KMeans model [24] on the $F = 768$ dimensional feature patches extracted from all of the available training data by the vision transformer [23].

4.3.1 RQ1: Can the coarse BOP+JS representation distinguish between different target types?

First, in our effort to validate our embedding, we wish to show that our two meaningless, high-entropy classes of UNKNOWN and OTHER do not show any similarity with any of the other well-defined classes. We expect to see these classes uniformly show extremely high JSD (.5 or above) for 2 different reasons. For UNKNOWN, we expect to see these perform poorly, because they are by definition clearly not background information while also possessing some reason that we cannot identify them as targets. As such, they should be different from all of our definite classes and even our NT group since they have their key identifying features obstructed. For the OTHER group, we

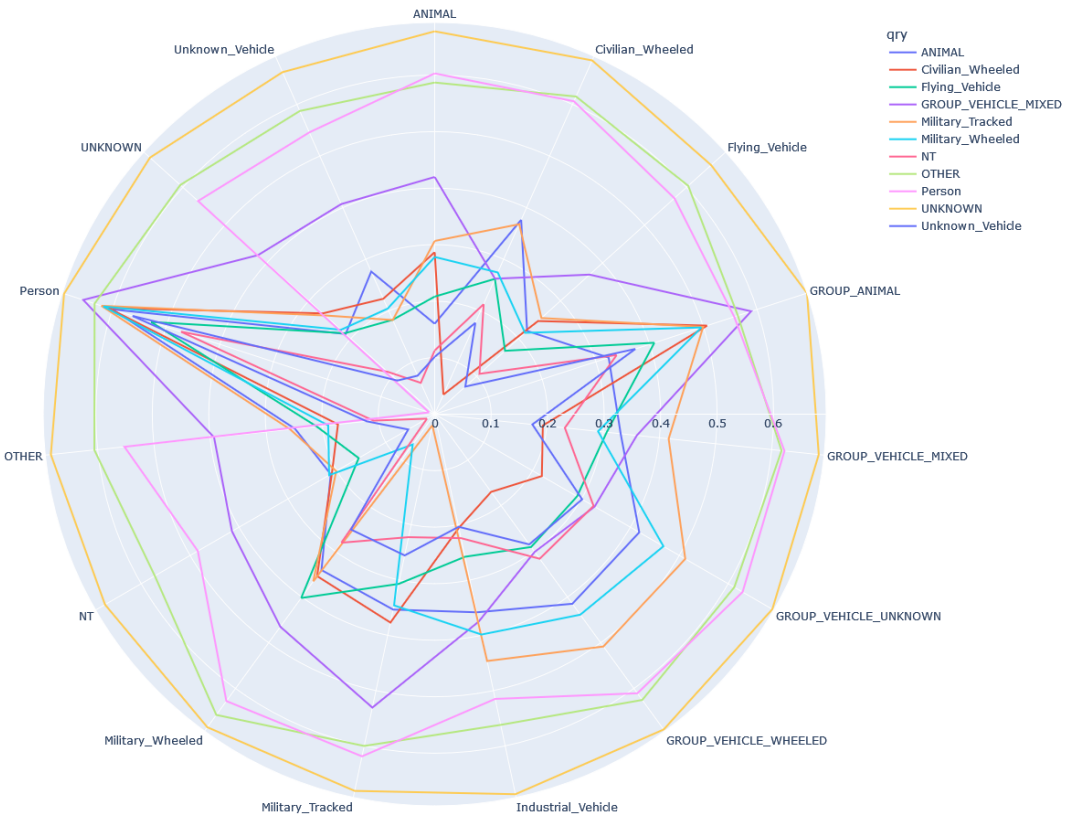

Figure 4.1: Spiderplot showing the JS scores for all ref/qry target combinations. The lines on the plot correspond to the behavior of a given query whose name and color are given in the legend at the right. When specific queries are isolated, all other query lines will disappear from the plot and their relevant legend icons will become greyed out allowing for consistent legend placement between figures.

expect to see them to have a poor similarity to other classes, because this group is made up of a few individually identifiable classes that are more distantly unrelated to each other mixed together. By mixing the distributions of highly different classes, we expect the resulting distribution to have no good relationship to a group of classes which contain similar classes. In short, we are investigating if grouping targets together randomly yields significantly different results than grouping similar targets together.

In figure 4.2, we isolate the JS scores for only the UNKNOWN and OTHER target classes. As it can be seen, our method does in fact show that the OTHER and UNKNOWN classes show a high level of dissimilarity with all classes we have, even their own training sets. It is reasonable that UNKNOWN would not even be similar to its own training set, but OTHER is a little bit surprising at first glance. This behavior is due to the fact that the OTHER class is made up of a few random

and only tangentially related distinctly identifiable classes, the distribution of prototypes are very sensitive to class balance which we know is not maintained between test and train sets. All of these measurements are as expected in the "Datasets are completely different" range.

The second point which we wish to establish is that the most easily discriminable class is extremely similar to itself and no others. For this, we will use the "Person" class as 60% of our reference classes are vehicles of some kind, 2 of them are various kinds of animals combined together, and the last 3 classes contain unpredictable contents. Additionally, People are consistently some of the easiest targets to identify from IR cameras due to their unique shape, uniform temperature distribution, and prevalence in the data so BOP+JS should be capable of identifying the uniqueness of this class. We can see in figure 4.3 that the person class has some of the strongest self-similarity possible at around .01, while also having extremely high dissimilarity for all other classes. This is strong evidence that our methodology is effective at identifying similar groups of data and ignoring classes that we would expect to have no relation.

The final point we wish to show is also the most complex. First, we show that our methodology can distinguish two semantically similar classes from one another while also having them act similar when compared to 3rd-party classes. The behavior we expect that will validate this capability is that our two related classes score poorly when compared to one another, but then their individual scores show similar trends when compared to all other classes. The two classes we chose for this are the two wheeled vehicle classes, CIVILIAN_WHEELED and MILITARY_WHEELED, which we will refer to as Civilian and Military, respectively. We first show that our methodology acknowledges the distinctness of these two classes in table 4.2 where we can see that when we compare Civilian Ref to Civilian Query and Military Ref to Military Query our scores are in the expected range for self-similarity. The Civilian class is mildly easier for our method, as this class has smaller inter-class variation than the Military class.

Now that we have established the distinctness of our two related classes, we can look at how they behave when compared to the other classes we have in figure 4.4. We can see that these two classes maintain very similar scores for the classes of Animal, Flying Vehicle, Group Animal, Military Tracked, NT, OTHER, UNKNOWN, Person, and UNKNOWN Vehicles, as expected. These two related classes score within .05 JSD of each other when compared to 69% of the other classes. The other 4 classes require some discussion individually to be sure the differences are reasonable. 3 out of 4 of these classes are Group Vehicle classes, which are locations where multiple vehicles are close together and overlapped such that the boundaries of individiual vehicles cannot be determined. The

TABLE 4.2

Inter-class and intra-class similarities of the Civilian and Military Wheeled Vehicle classes

	Civilian Query	**Military Query**
Civilian Ref	0.037	0.275
Military Ref	0.354	0.066

specific differences between these groups are that the Group Vehicle Wheeled class predominantly consists of civilian vehicles, Group Vehicle Unknown consists of obscured or unidentifiable vehicles, and Group Vehicle Mixed contains a mixture of Military and Civilian vehicles. With these contexts in mind, the behavior of the two wheeled classes is reasonable. The group class that scored most similarly to our military wheeled class is the only group which definitively contains some military vehicles. The worst scoring group class for both civilian and military wheeled classes is the obscured Group Vehicle Unknown group, and the group class which scores well for civilian wheeled and poorly for military wheeled is the group class made up of purely civilian vehicles. The final class which did not possess a similar JS score between Civilian and Military is the Industrial Vehicle class, which is made up of items like tractors, trailers, and trains. There are no military vehicles of any kind in this class. While this level of disparity initially surprising, it is sensible that many key features of military vehicles like camouflage, armor, armaments, etc. are not present in these targets which would lead to a closer similarity for civilian vehicles than for military vehicles.

In this section, we relied on the available ground truth to show that BOP+JS is capable of characterizing high-level semantic information for groups of data by establishing some baseline capabilities of our methodology. First, it will not ascribe meaning or similarity to relationships which possess none. Second, it is capable of identifying the most unique class in our dataset as strongly distinct from all others. It is important to remember that the image patches used here are captured from the same parent images as other classes, so if a person is standing next to their vehicle, our system would generate two separate image patches and assign them to the groups Person and Civilian Wheeled, respectively. This means that the similarities and differences detected by our methodology are primarily driven by true target similarities and only minorly driven by sub-context background cues. Finally, we showed in detail that high-level semantic similarities of two conceptually related classes are faithfully reconstructed by our methodology. These three qualitative experiments give us confidence that the answer to RQ1: "Can the coarse BOP+JS representation distinguish between different target types?" is a definitive Yes.

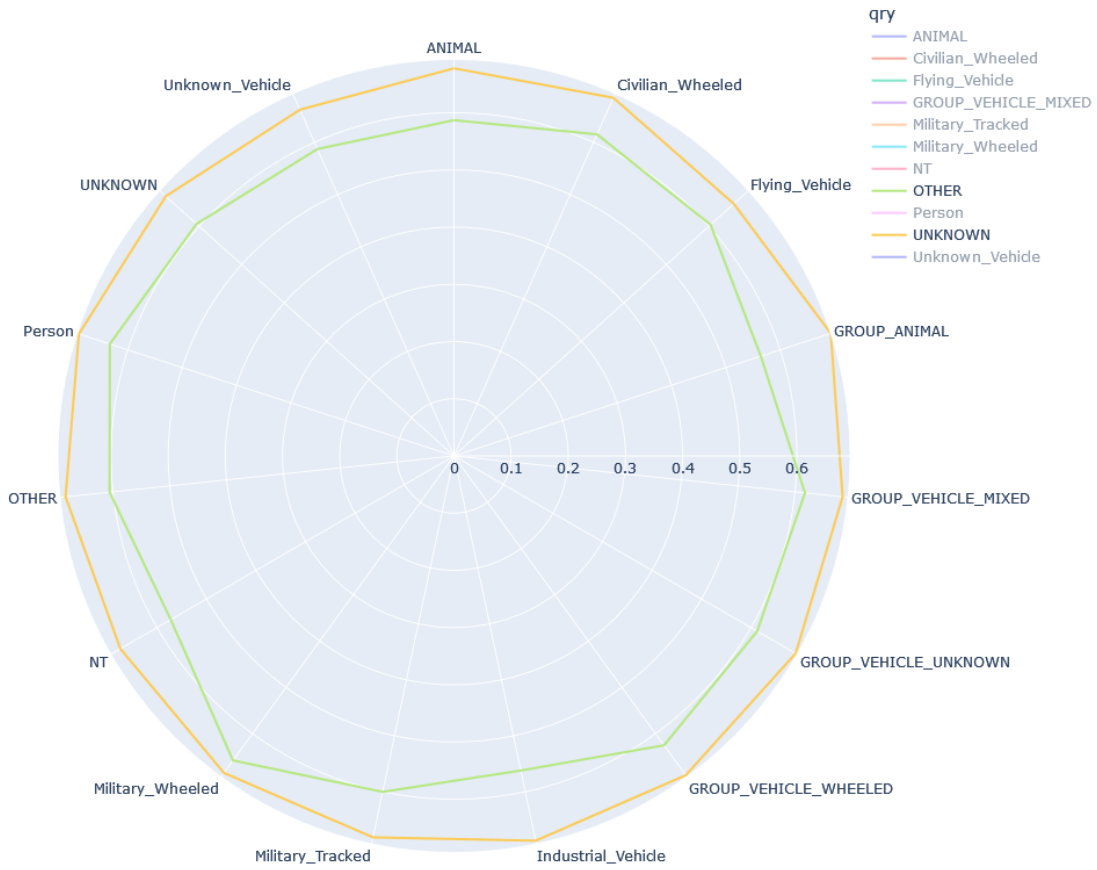

Figure 4.2: Spiderplot of semantically Meaningless Targets: Unknown and OTHER. Both of these classes show very high divergence scores for every target class analyzed as expected.

4.3.2 RQ2: Can this method extend to identifying similar collections as well?

The method for feature extraction we employ is optimized for differentiating between target types, which helps to explain the strong success on RQ1. However, when comparing collections, this can become a hindrance as now a driving factor of collection similarity is going to be the target prevalence between the collections. Nearly all of our data collections are either entirely comprised of vehicles or at least a majority of the focus of them is vehicles. Collection 13 is the only collection which entirely consists of only personnel images, which makes it completely ill-suited to train the vehicle focused target discrimination required to succeed on any other collection. Therefore, we expect to see collection 13 to have the strongest self-similarity of any collection and to have the highest divergence when compared to any other collection. We can see the results of this experiment for every collection in figure 4.5. The major stand out collection is indeed v13, which shows the strongest self-similarity of any ref/query pair. In addition to strong self-similarity, we can see that

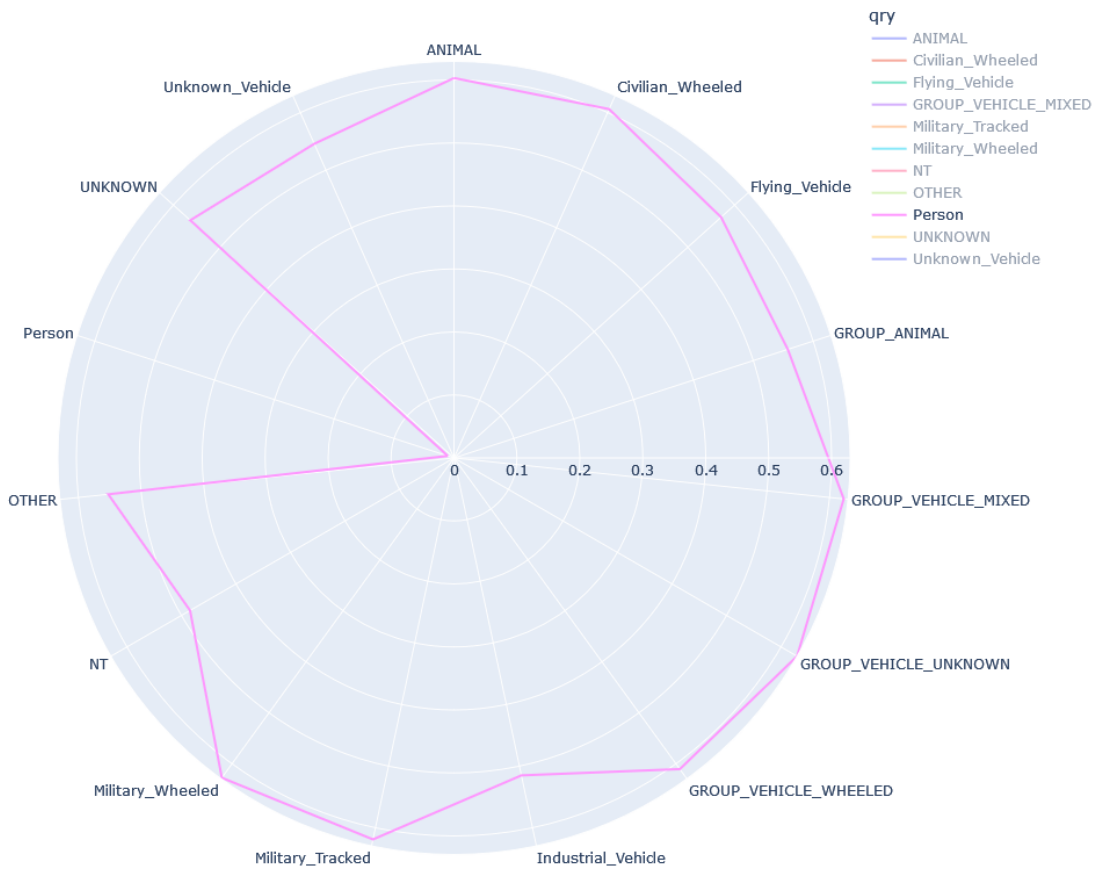

Figure 4.3: Spiderplot isolating only the data containing Person targets. This target shows strong self-similarity and high divergence for all classes other than itself as expected.

it is the least similar collection to any other collection on the plot by a wide margin for both the reference and the query.

Next we wish to show that self-similarity is still prevalent for collections even if less pronounced than the self-similarity was for differentiating targets. We can see this in figure 4.6 along the trace diagonal of the matrix. We can see that the self-similarity diagonal has lower divergences than the majority of other comparisons, but not by a wide margin.

Finally, we look at collections v4 and v5, which are the only collections who were collected at the same time and location as each other. The only difference between these two collections is the hardware used to capture the video, where v4 was captured with a Long-Wave IR (LWIR) camera and v5 was captured with Mid-Wave IR (MWIR) camera. The background, targets, temperature, and even the angle of the images should be nearly identical, and the data augmentation steps we take in training the CNN should overcome any major differences due to MWIR vs. LWIR. Therefore, we

Figure 4.4: Spiderplot isolating Civilian Wheeled Vehicles and Military Wheeled Vehicles. These classes show similar JS behavior for every class other than each other, where we see JS has identified them as distinct classes by their strong self-similarity.

expect their divergence spider plots to be nearly identical, which we can see is the case in figure 4.7.

In this section, we built upon the experiments done at a target level to show that BOP+JS can faithfully characterize similarities at the collection level as well, albeit with less fidelity. First, the unique collection that is the only one with a single target focus does show itself to be entirely different from all of collections. Second, self-similarity is still present as a general rule, but it is not as stark as it was on a target level. This is understandable, as our feature space is optimized for discriminating targets and not collections. Finally, collections which are nearly identical in their content are shown to have a nearly identical divergence score signature, which validates the stability of the method for technical differences. Overall, the answer to our research question "Can this method extend to identifying similar collections as well?" is a qualified yes. We do see that this method can extend to collections, but the results are nowhere near as definitive as it was for the targets.

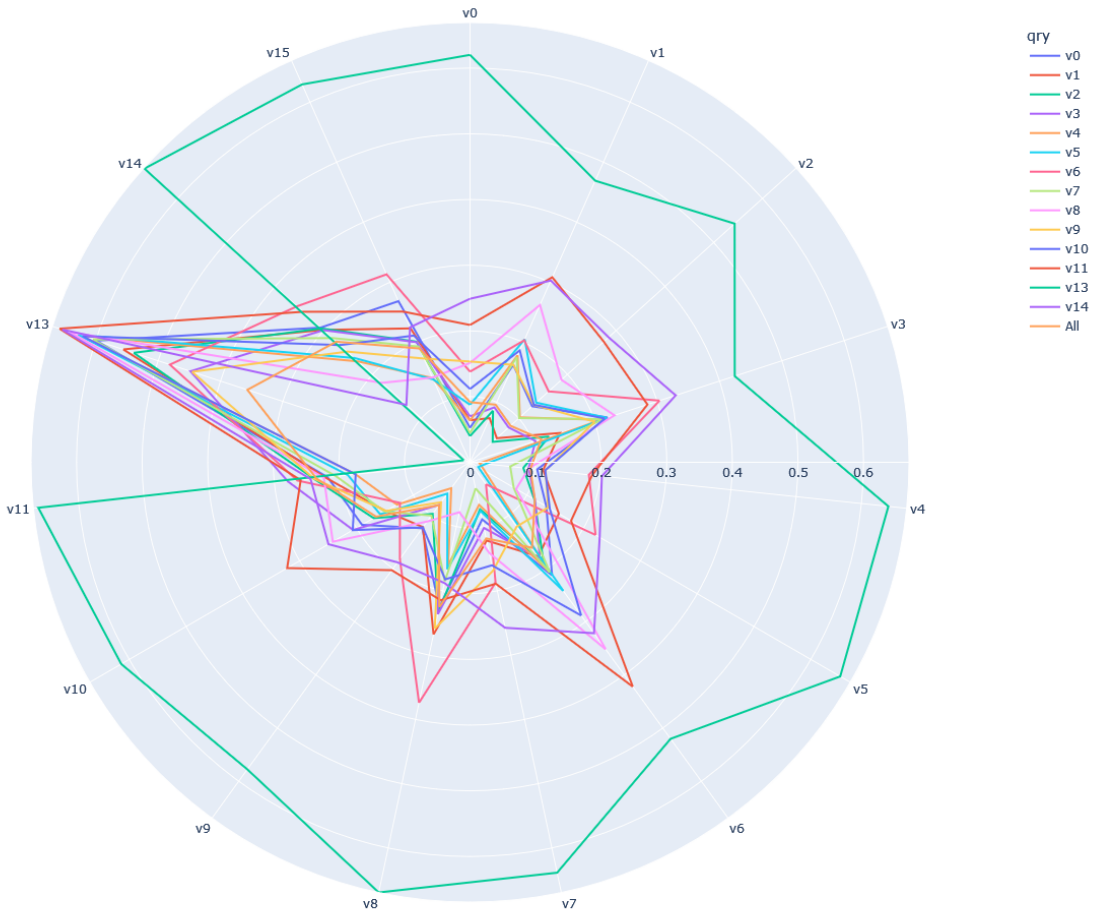

Figure 4.5: Divergence Spider Plot of Query Validation Collections vs. Reference Train Collections for all collections tested. Points close to 0 are more similar and points far away from the center are less similar.

4.3.3 Research Question 3: Will Training on the most similar datasets outperform training on the least similar datasets?

We have established that the similarities and differences at the collection level is not nearly as stark as at the target level. This leads to a difficult observation that the majority of ref/query comparisons we have at a collection level are below a divergence score of .25. For the purpose of making the best ATD system this is good for us, as that means almost all of our collections are suitable for training on one another. However, having the majority of reference collections being acceptably similar our query collections lowers the possible performance increases we can expect to see by selecting only the best collections. This is not a weakness of the BOP+JS methodology, this is an unexpected hurdle brought to us by the unusually high-quality training data we available. Intuitively, if there is not a wide enough spectrum of good and bad collections to select, then selecting

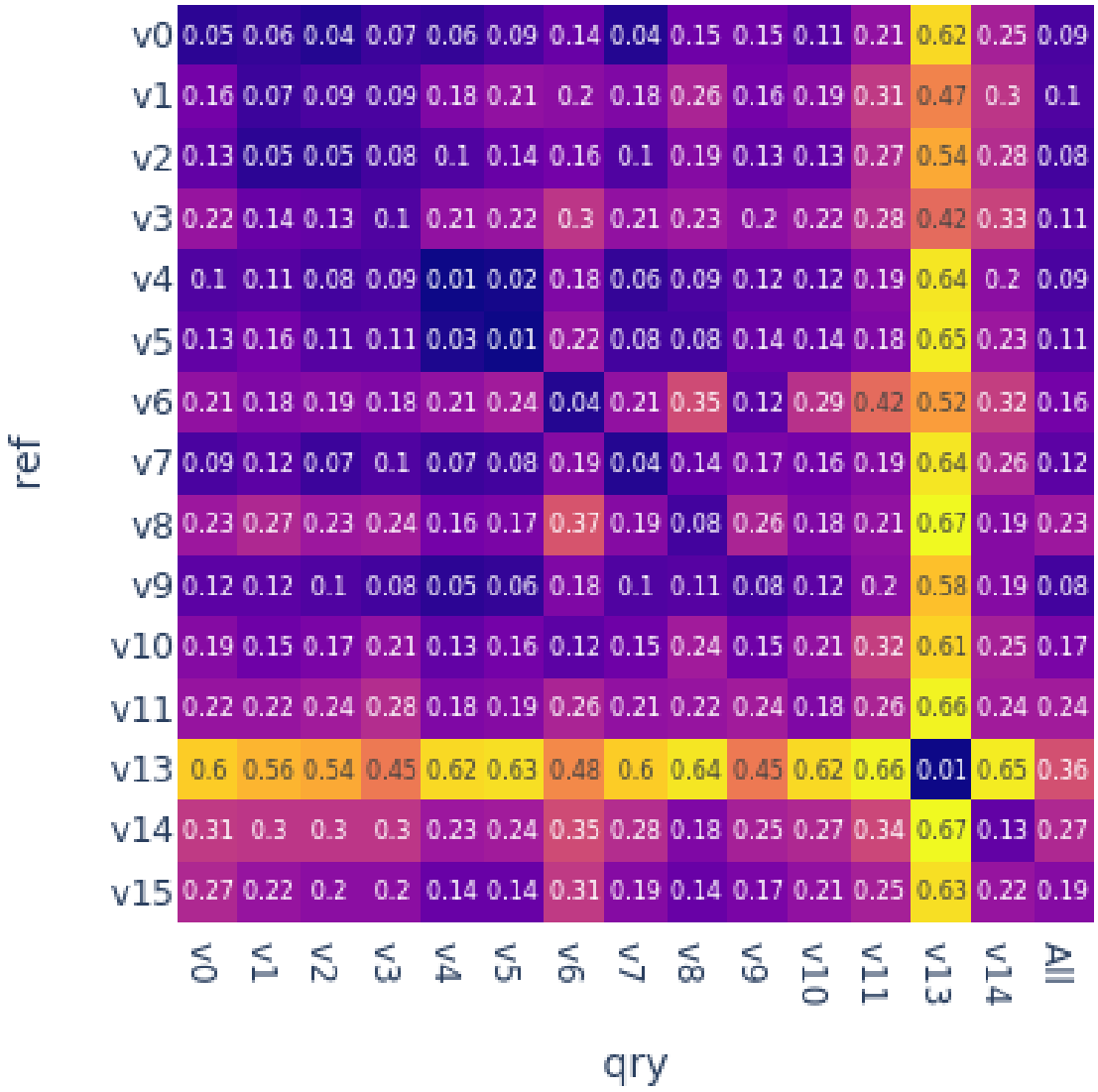

Figure 4.6: Pairwise divergence scores for all collections. Self-Similarity is shown as the trace of this matrix, which we expect to be low. It's good to note in this image that not all reference training datasets have a matching query validation dataset, and that collection 12 is skipped in this thesis.

the best collections will not provide much value. In fact, there is intrinsic value to a varied training dataset, so removing acceptable collections from the training pool will likely only lower performance.

In order to overcome this predicted weakness in our dataset, we are first going to show a proof-of-concept experiment that pits the n best reference collections as decided by BOP+JS vs. the n worst reference collections for a few query collections. By pitting the best and the worst against each other, we should be able to see a stark difference in performance in all 3 of our chosen metrics: AUC, PD@0.1FAR, and PD@0.5FAR. We plot the performance of every model trained on

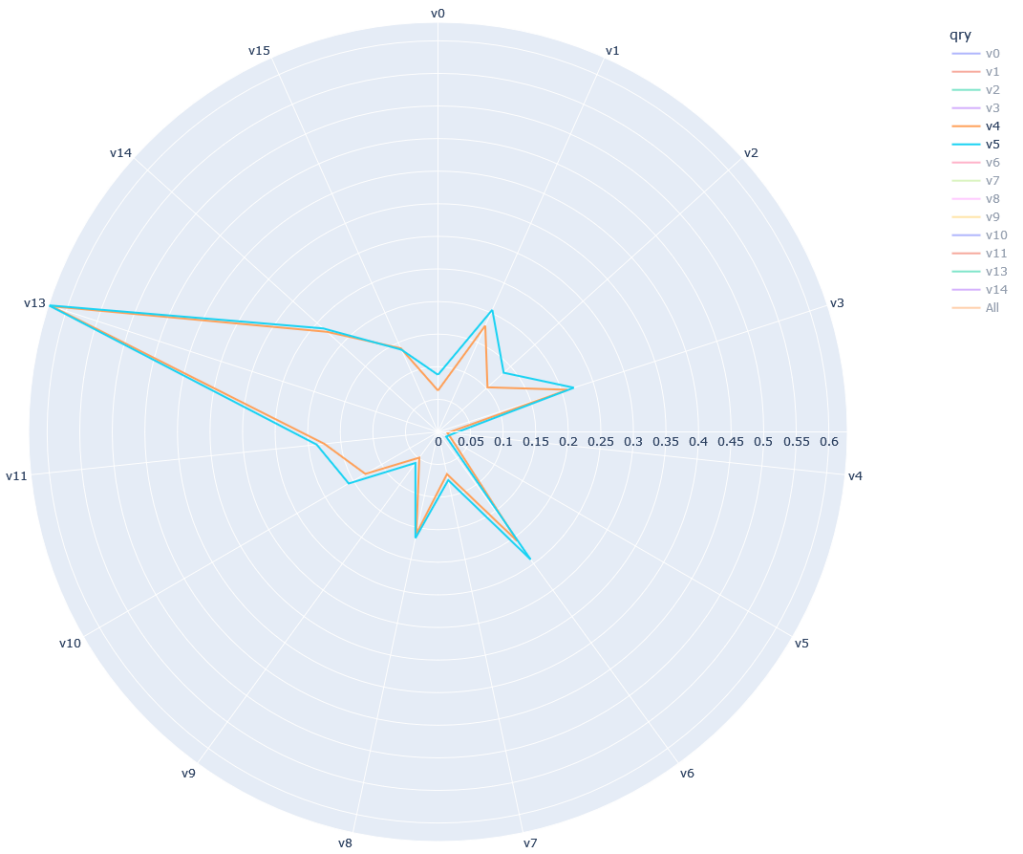

Figure 4.7: Divergence Spider Plot of query collections V4 and V5 which were collected on the same day on the same targets. The only difference between these two collections is the band of infra-red frequencies used to capture the images. We can see these two collections are exceedingly similar to one another in their divergence behavior for all collections.

the most similar data, S^c_{-i}, for each of our metrics on the X axis with the matching performance for the least similar data, S^c_{+i}, on the Y axis. When the performances are plotted in this way, anything below the X=Y line signifies a success where the models trained on the S^c_{-i} outperformed the equivalent dataset S^c_{+i}. Furthermore, we should see that the gap in performance converge to the X=Y line as the number of collections i increases, where both sets of collections begin to gather the same elements. However, this specific convergence behavior is again subject to the problem of our collections lacking in a large enough variety of JSD scores, so it will likely not hold for all collections.

We selected 6 query collections from our full dataset to perform this experiment on pseudo-randomly. Before running this experiment, we had historical results on the performance of our ATD method trained on all reference collections for each of our query collections. We selected 6 query collections by selecting two of the top performing query collections, two collections that had

adequate performance, and two collections which were performing poorly. The only collection of these that was hand-picked for any particular reason is collection v6, which showed promise as the collection with the largest variety in JSD scores. As such, we expected v6 to match our hypotheses the closest. These 6 query collections were tested on by models trained by selecting the training datasets in groups of the top 3, 6, and 9 collections.

This sampling scheme was decided to minimize the computational cost of these experiments. With 15 reference collections and 12 query collections, if we had exhaustively trained a model for each relevant combination of collections, we would have had to train an estimated amount of about 300 models to complete this experiment. A single model takes anywhere from 3-7 days to train and score on an NVIDIA A100 GPU, depending on the size of the training set and number of false-positives that model generates. As such, an exhaustive experiment was unfeasible.

We can see the results of our experiments for each of the performance metrics in figures 4.8, 4.9 and 4.10. These experiments universally reflect the proof of concept that the models trained on the reference collections with lower JSD scores outperform their counterparts trained on the high JSD scores collections. Our experiment yields 54 data points across 3 metrics, and 53 of them are below the X=Y line with the 1 outlier appearing nearly on the X=Y line. This is significant evidence validating our method's correlation with dataset suitability, and that a greedy approach for training set aggregation is reasonable. The answer to Research Question 3: "Will Training on the most similar datasets outperform training on the least similar datasets?" is a confident yes.

4.3.4 Research Question 4: Will greedily building a training set from the most similar datasets outperform training on all datasets?

This research question signifies the primary goal of this thesis, which is to investigate the effectiveness of BOP+JS at selecting training data collections to improve the final performance on specific testing data collections. This performance increase would come from removing training collections which are poorly suited for the specific query at hand. Unfortunately, as discussed previously, our data shows acceptable similarities across almost any comparison we make which greatly limits our expected performance increases. Our hypothesis is that we should see slight performance increases while our training dataset grows in size to the target dataset size A, and then have the performance steadily drop as samples from the higher divergence collections replace the samples from lower divergence collections. Specifically, given two training sets $S^c_{-i_1}$ and $S^c_{-i_2}$ with equivalent sizes and $i_1 < i_2$, we expect to see the model trained on the smaller set of collections $S^c_{-i_1}$

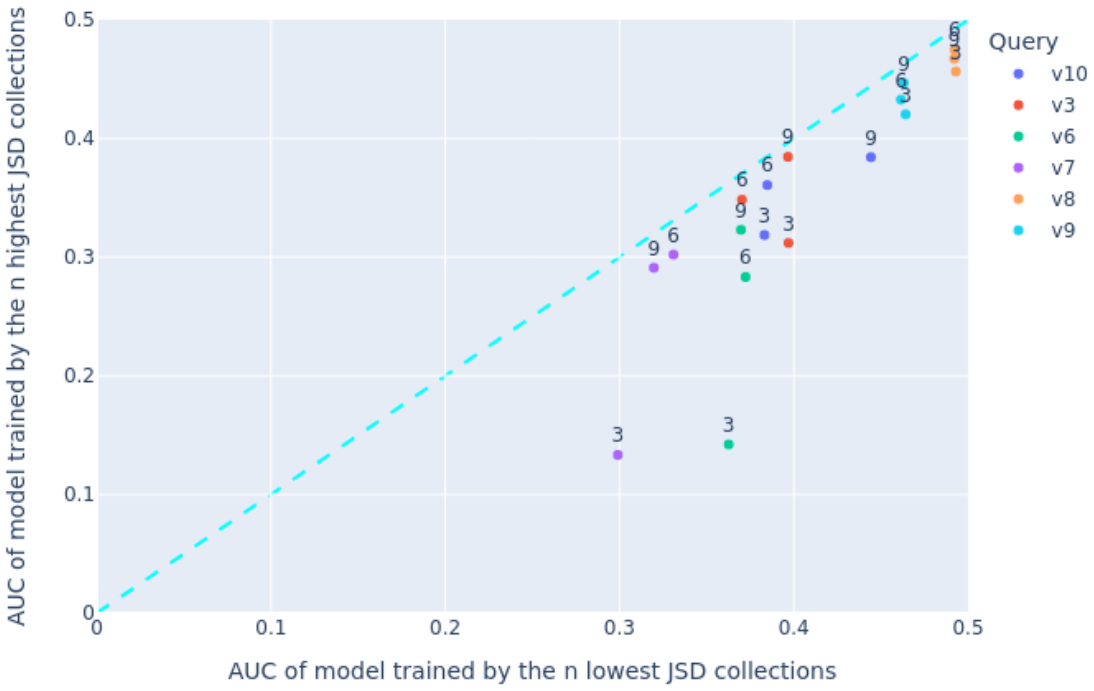

Figure 4.8: AUC of the models trained on the S^c_{-i} low JSD set vs the S^c_{+i} high JSD set for some validations. The number of collections used for each model, i, appears above the point associated with it. The models trained on low JSD collections are expected to perform better.

to outperform the model trained on $S^c_{-i_2}$ However, in actuality with our specific dataset we expect to see the scores to remain rather steady or even increase in some cases when the dataset grows in rich variety with little drawbacks as the data added is of similar divergence to the data removed.

The results of our experiments are in figures 4.11, 4.12, and 4.13 where we can see that our two best-performing validations show a slight downward trend in performance across all 3 metrics. We then see that we have two semi-chaotic collections which show a large drop in performance going from the top 3 collections to the top 6 collections that then rebounds afterwards as the variety of training data improves. Finally, we see two collections show behavior exactly matching our hypothesis as the performance increases going from 3 to 6 collections, doubling the variety and increasing the dataset size, followed by a steady decrease as data from less-applicable collections are added. Notably, the collection we identified of having the highest likelihood of success, v6, is one of these two collections.

Unfortunately, it seems like the answer to Research Question 4: "Will greedily building a

Figure 4.9: PD@0.1FAR of the models trained on the S^c_{-i} low JSD set vs the S^c_{+i} high JSD set for some validations. The number of collections used, i, for each model appears above the point associated with it. The models trained on low JSD collections are expected to perform better.

training set from the most similar datasets outperform training on all datasets?" is dependent on the data on which it is deployed. There is promising evidence that leads us to believe that a greedy method with BOP+JS may improve performance if used on collections with more JS score variety, but many of our higher-performing collections show a good fit with almost all of our references which limits the possible benefit gained by removing collections from the reference.

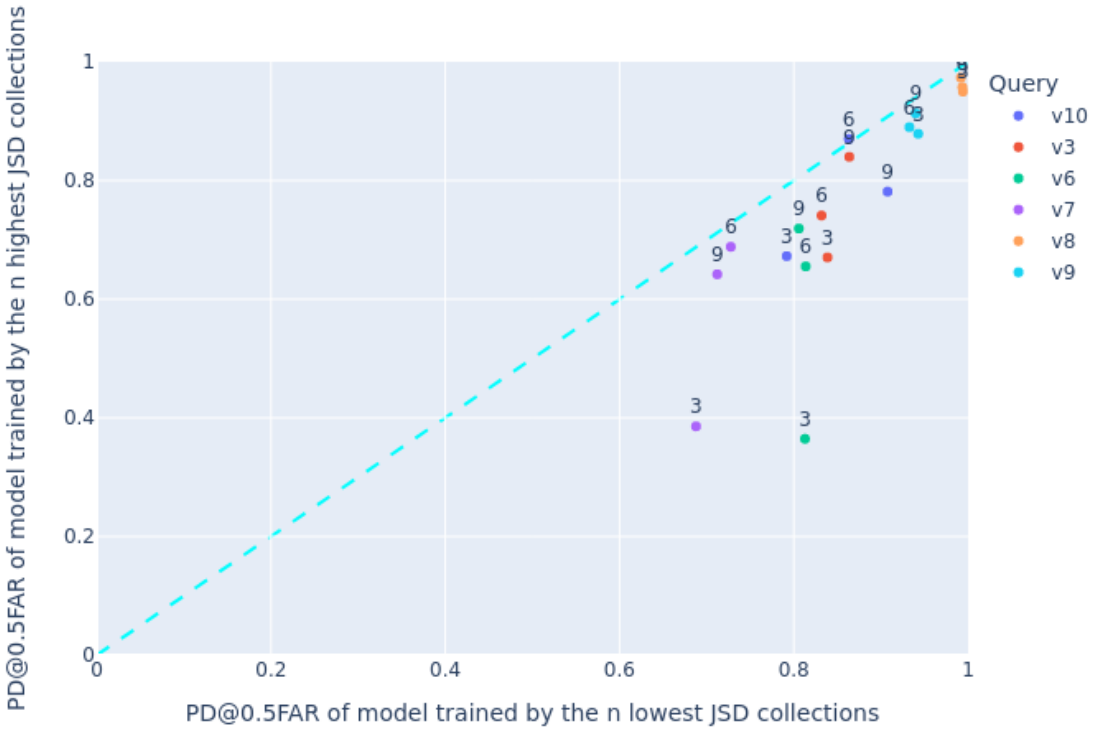

Figure 4.10: PD@0.5FAR of the models trained on the S^c_{-i} low JSD set vs the S^c_{+i} high JSD set for some validations. The number of collections used for each model, i, appears above the point associated with it. The models trained on low JSD collections are expected to perform better.

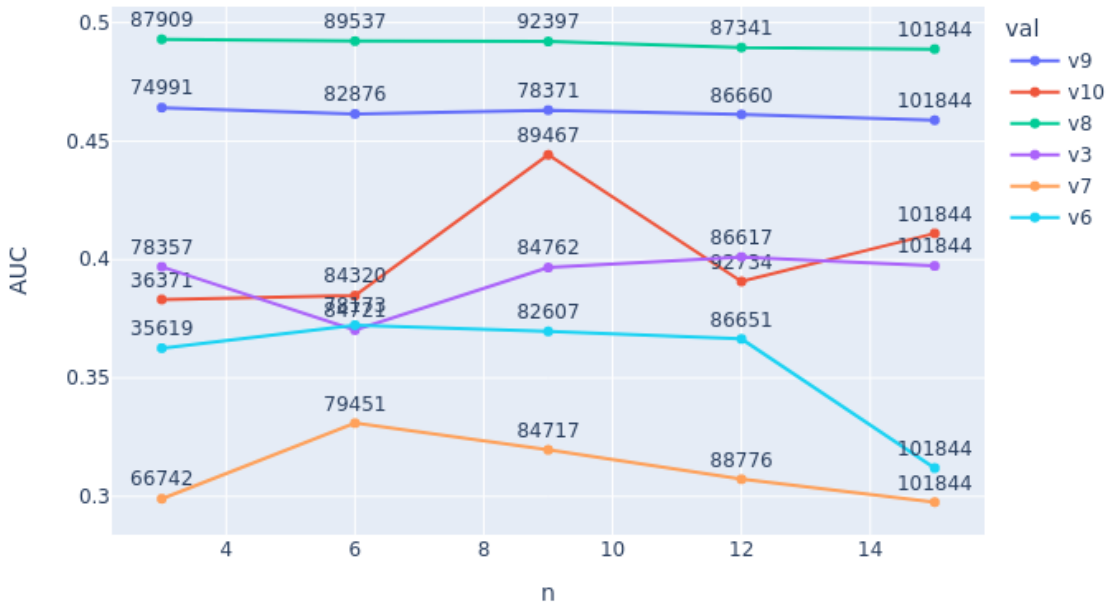

Figure 4.11: AUC of best models trained on the top 'n' best-fit collections. The number of samples in each training set is not constant and is displayed above each point. The training sets were sampled at the uniform sampling rate which would deliver both a balanced dataset and the closest number of samples to about 8500. We can see that the collection with the most room for improvement, v7, shows a significant jump in performance as we double the training set variety from 3 collections to 6 that then decays as less-suitable collections are added.

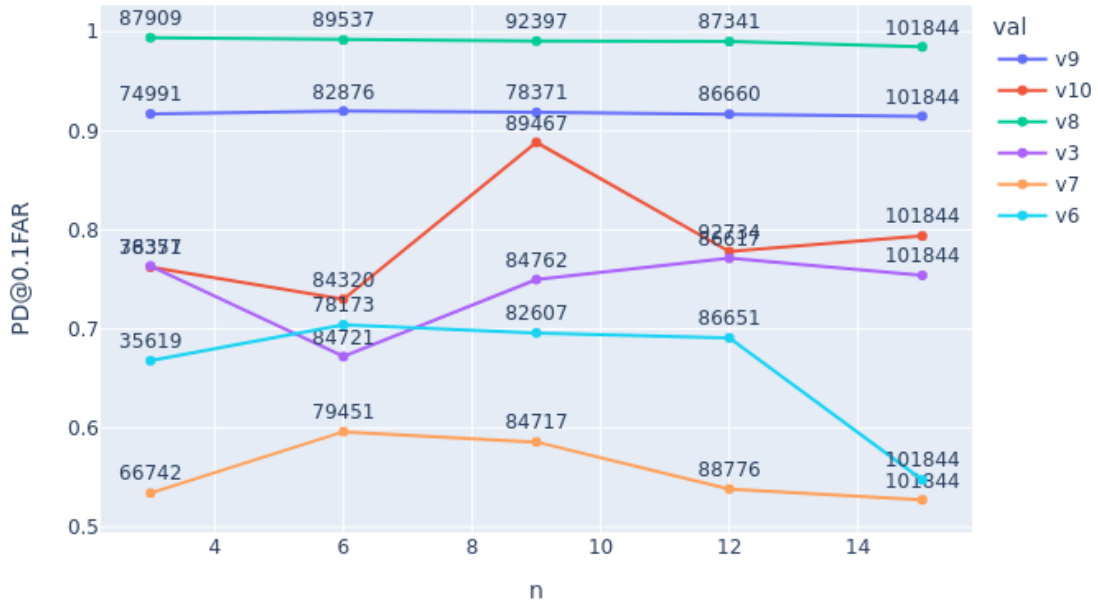

Figure 4.12: PD@0.1FAR of best models trained on the top 'n' best-fit collections. The number of samples in each training set is not constant and is displayed above each point. The training sets were sampled at the uniform sampling rate which would deliver both a balanced dataset and the closest number of samples to about 8500. We again see that collection v7 shows performance gains as the dataset is expanded, which then decays as less-suitable collections are added.

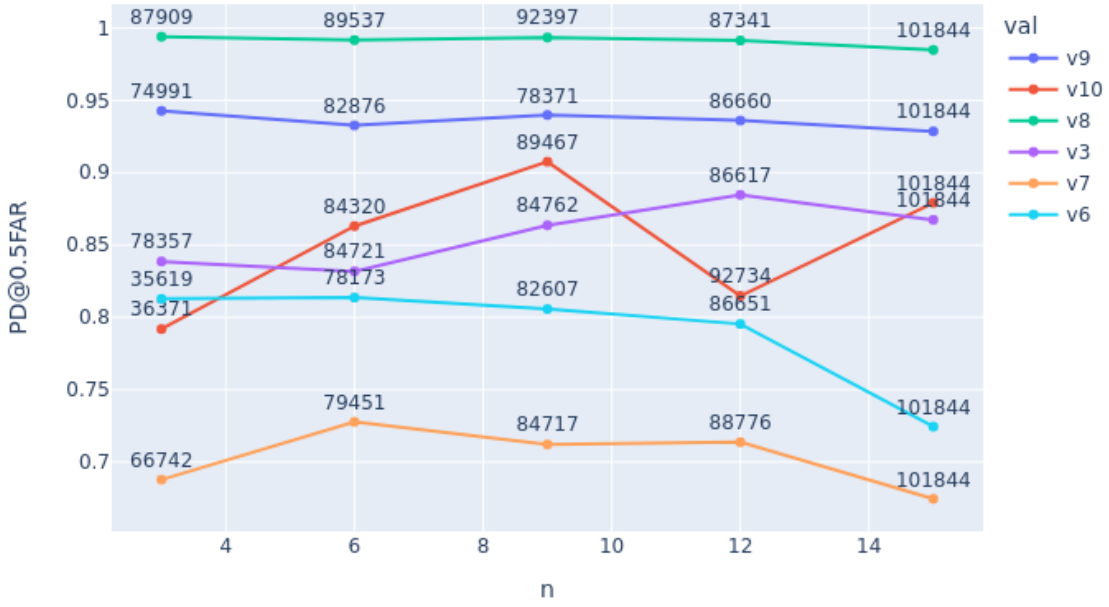

Figure 4.13: PD@0.5FAR of best models trained on the top 'n' best-fit collections. The number of samples in each training set is not constant and is displayed above each point. The training sets were sampled at the uniform sampling rate which would deliver both a balanced dataset and the closest number of samples to about 8500. We again see that collection v7 shows performance gains as the dataset is expanded, which then decays as less-suitable collections are added.

CHAPTER 5

CONCLUSIONS

In this thesis, we investigated the application of BOP+JS for task-specific unsupervised greedy training collection selection for ATD of IR images containing civilian and military targets. We explored four different research questions which can be understood as three increasingly relevant proof of concept questions followed by our primary goal. BOP+JS showed promise in every proof of concept experiment we attempted, but, unfortunately, we were unable to confidently conclude that a model trained using a dataset greedily picked by using BOP+JS as a heuristic will outperform a model that's trained on all available data. We believe this is due to our data lacking enough "bad apples" in the training set to make pruning collections worthwhile, but we do not have enough data to prove that statement.

We leveraged the versatility of BOP+JS to group our data by target type and showed that it is able to distinguish between target types while conserving semantic relationships. The high-level groups of targets were accurately characterized with the quantitative divergence values mirroing the qualtitative semantic similarities in targets while preserving the uniqueness of unrelated target classes. We then performed an equivalent experiment on the same patches grouped by collection to show that BOP+JS is still capable of preserving high-level similarities at the collection level.

We performed a paired model experiment where we selected groups of collections sorted by their expected suitability to compete predicted high-suitability groups against low-suitability groups. We confirmed that BOP+JS can estimate the relative suitability of training collections for specific tasks even with the limitations of our data by showing the models trained on the highly suitable collections universally out-performed models trained on the low-suitability collections across the AUC, PD@.1FAR, and PD@.5FAR. With this final proof of concept in hand, we compared models that were trained on increasingly large subsets of collections that were aggregated in a greedy method using BOP+JS as a heuristic. Out of the six collections we included in the experiment, two of them showed minor improvement in performance, two of them showed unpredictable behavior, and two of them showed a moderate improvement in performance that matches our hypothesis. These results

lead us to believe that the success of BOP+JS depends on the variety of the data on which it is used. If BOP+JS is used as a heuristic on data that does not possess enough high-divergence collections whose removal will be beneficial, then the performance increase gained by their removal will be minimal at best.

In this thesis, we utilized a CNN for feature extraction that was optimized for recognizing target types. We chose this as features optimized for target types will be closely aligned to the final feature space representations of images within our ATD. An alternative to this approach is to use a CNN autoencoder as a feature extractor. This may improve characterization as the autoencoder will more faithfully capture all information in an image patch instead of focusing only on the target. We also intentionally avoided areas of pure background information in our experiments, but utilizing the autoencoder may increase the effectiveness of including patches of pure background information in the future. We may also use a different methodology for developing our codebook to specifically reserve some features for different purposes. For example, we could train a 300 prototype codebook with 100 prototypes trained on YOLO patches, 100 prototypes trained on precise ground-truth patches, and 100 prototypes trained on random patches of background information to develop a codebook that can provide more versatility to its characterization.

The versatility of BOP+JS extends beyond the scope of this thesis. BOP+JS is a data-driven unsupervised method for quantifying dataset-level similarities between collections of images regardless of how the images are represented in feature space, how the images are split into collections, or the content of those images. This flexibility allows for many more applications than what we have explored here including other image modalities like x-ray, lidar, CT scans, etc. For example, one experiment that is applicable in automatic target recognition is to improve the differentiation between two similar classes by selecting collections which show the maximum internal divergence between those two classes. Likewise, another valid experiment to test if BOP+JS can be used to predict difficult classes in a confusion matrix by correlating the pairwise JSD to the final class confusion rates may be worthwhile. However, as this thesis is focused on automatic target detection instead of automatic target recognition, these experiments are left as ideas for future work.

Overall, BOP+JS showed great promise for assessing task-specific dataset suitability, but the performance benefits gained from its application to our data were moderate at best and inconclusive at worst. BOP+JS appears more appropriate when selecting from collections that contain more significant differences between one another than our datasets showed. The variety of applications and versatility of BOP+JS are significant and our results should not dissuade others from experimenting

with applying BOP+JS for their own tasks as we believe it is a valuable and versatile method.

REFERENCES

[1] Konrad Jacobs, *Independent Identically Distributed (IID) Random Variables*, pp. 65–101, Birkhäuser Basel, Basel, 1992.

[2] "Infrared and thermal imaging system benefits and applications," Feb 2018.

[3] Vitaly Feldman and Chiyuan Zhang, "What neural networks memorize and why: Discovering the long tail via influence estimation," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 2881–2891, Curran Associates, Inc.

[4] Weijie Tu, Weijian Deng, Tom Gedeon, and Liang Zheng, "A bag-of-prototypes representation for dataset-level applications," 2023.

[5] Lora Aroyo, Matthew Lease, Praveen Paritosh, and Mike Schaekermann, "Data excellence for ai: Why should you care," 2021.

[6] Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li, "Dataset pruning: Reducing training data by examining generalization influence," 2023.

[7] Pang Wei Koh and Percy Liang, "Understanding black-box predictions via influence functions," 2020.

[8] Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C. Mozer, "Characterizing structural regularities of labeled data in overparameterized models," 2021.

[9] C. Horne and R. Saeger, "An advanced configuration management tool set," in *1988 Conference on Software Maintenance*, Los Alamitos, CA, USA, oct 1988, pp. 229–234, IEEE Computer Society.

[10] Wisam A. Qader, Musa M. Ameen, and Bilal I. Ahmed, "An overview of bag of words;importance, implementation, applications, and challenges," in *2019 International Engineering Conference (IEC)*, 2019, pp. 200–204.

[11] Akiko Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing Management*, vol. 39, no. 1, pp. 45–65, 2003.

[12] Gabriela Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray, "Visual categorization with bags of keypoints," *Work Stat Learn Comput Vision, ECCV*, vol. Vol. 1, 01 2004.

[13] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," 11 2003, vol. 2, pp. 1470 – 1477 vol.2.

[14] M.L. Menéndez, J.A. Pardo, L. Pardo, and M.C. Pardo, "The jensen-shannon divergence," *Journal of the Franklin Institute*, vol. 334, no. 2, pp. 307–318, 1997.

[15] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," 2018.

[16] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022, The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19.

[17] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[18] Joseph Redmon and Ali Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[19] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[20] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei, "Yolov6: A single-stage object detection framework for industrial applications," 2022.

[21] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi, "Real-time flying object detection with yolov8," 2023.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Eds., Cham, 2014, pp. 740–755, Springer International Publishing.

[23] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[24] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, 2023.

CURRICULUM VITAE

NAME: Tylman Michael

ADDRESS: Computer Engineering & Computer Science Department
Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION:

M.S., Computer Science
May 2024
University of Louisville, *Louisville, Kentucky*

B.S. Computer Science and Physics (dual)
May 2020
Union University, *Jackson, Tennessee*

HONORS AND AWARDS:

1. First Place in Major Field Achievement Test, Union University, 2020
2. Outstanding Research Award from the Center for Scientific Studies, Union University, 2019
3. Outstanding Research Award from the Center for Scientific Studies, Union University, 2018
4. First Year Programming Award, Union University, 2017

PROFESSIONAL EXPERIENCE

1. Associate Software Developer, Pathology, St. Jude Children's Research Hospital, 2020-2022
2. Sr. Software Engineer, Computational Biology, St. Jude Children's Research Hospital, 2023-Present