Andreas L. Teigen

Few-shot open world learning

Master's thesis in Cybernetics and Robotics

Supervisor: Annette Stahl

September 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Andreas L. Teigen

Few-shot open world learning

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
September 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Summary

Computer vision systems are gradually seeing an increased use in real world applications in a variety of domains. However, this transition from controlled lab environments to a real-world setting introduces several new problems. Instead of encountering only the classes used during training, potentially any class can be presented to the model in an open world scenario. As a result, in addition to the normal classification, the model must be able to identify the new classes and efficiently learn to adapt to these new classes, preferably with minimal downtime. This is known as the Open World Learning Problem, and it is comprised of two sub-problems: Incremental learning, which deals with the updating and continuous learning of the model, and open world classification which deals with the classification and discovery of new classes.

This thesis proposes a framework as a solution to the open world learning problem based on few-shot classification strategy in combination with an outlier detection module. The few-shot classifiers employ a similarity-based classification scheme and are highly adept at generalization, requiring no training and only a few labelled examples of a new class before adapting to it, natively presenting a good solution to the incremental learning problem. The discovery of new classes is performed by the outlier detection module that utilizes the similarity space created by the few-shot classifier to identify samples that are sufficiently different from the known classes and removes them from the classification process.

Based on extensive experimentation with different combinations of few-shot classifiers, outlier detectors and open set recognition algorithms, this thesis highlights the ideal variations of the proposed framework for different applications. The results show that the framework is realizable with a moderately low accuracy loss compared to standard few-shot classifiers.

Preface

I would like to first thank my supervisors Associate Professor Annette Stahl and Postdoctoral Fellow Aya Saad who have helped me enormously during the writing of this thesis and encouraged me to write a paper based on this work. The paper was accepted by the OCEANS 2020 Gulf Coast conference and is to be published in IEEE, see appendix. This work is part of the AILARON project which is funded by CN FRINATEK IKTPLUSS program (project number 262701) andsupported by NTNU AMOS.

I also want to thank my parents for letting me occupy their cabin while working on this thesis without asking for anything in return. It has been a huge help to focus and concentrate in a peaceful environment.

The framework presented in this thesis is targeted towards the plankton domain for use in autonomous underwater vehicles, so all experiments and results are based on planktonic datasets, but no dataset dependent specialization is performed on the framework, so the models presented are also generalizable to other datasets and domains.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Glossary

Training data/classes	=	Data/classes used to train the classifier
Known data/classes	=	Data/classes the classifier has obtained previous information about but not used during training
Unknown data/classes	=	Data/classes the classifier has no previous information about.
Training set	=	The data a traditional classifier is trained over
Validation set	=	The data that a traditional classifier is validated over. Contains the same classes as the training set, but different samples
Background set	=	The classes used for the training of one/few-shot classifiers
Evaluation set	=	The classes used for validation of one/few-shot classifiers. All classes are different from the background set.
Closed world	=	All samples are from training/known data
Open world	=	Samples might be from training/known data or unknown data
Open set classification	=	Classification with rejection of unknown data
Open world learning	=	Open set classification with iterative model updates
Open set recognition	=	Open set classification in the field of computer vision
Open world recognition	=	Open world learning in the field of computer vision

Abbreviations

AI	=	Artificial Intelligence
ANN	=	Artificial Neural Network
CNN	=	Convolutional Neural Network
DOC	=	Deep Open Classification (algorithm)
MAV	=	Mean Activation Vector
NNO	=	Nearest Non Outlier (algorithm)
OSR	=	Open Set Recognition
OWR	=	Open World Recognition

Notation

$\mathbf{x}$	Input data
$\mathbf{y}$	Output data
$\hat{y}$	Predicted output labels
D	Training set
D_k	Training data of class k
J	Loss
K	Number of classes in the training set
k	Class enumeration
N	Number of data samples in the training set
N_C	Number of classes per training episode (k-way)
N_S	Number of support examples per class (n-shot)
N_Q	Number of query examples per class
M	Mean class vector
m	Embedding dimensions
O	Open space
R	Rejected set of query data
$\mathbf{r}$	Radius vector
S	Episode support set
S_k	Episode support set of class k
U	High dimensional image space
$\mathbf{v}(\mathbf{x})$	Activation vector from the penultimate layer of the network
ϵ	Threshold variable
μ	Class prototype (mean embedding for a class)

Chapter 1

Introduction

The field of computer vision has seen rapid advances over the past decade. Reaching near human performance in a variety of tasks. One aspect where artificial intelligence still lags behind however is in it's lack of ability to quickly generalize based only on a small amount of new data. Classical approaches requires a large amount of data, and for every new class that is included in the classification the entire model needs to be retrained. This retraining process also have a high chance of reducing the overall accuracy of the model.

As a response to this, One-shot and Few-shot learning algorithms were created, and they are now well established techniques in the field of computer vision. They posses a remarkable ability to adapt, only requiring a minimal amount of data in order to classify new classes. They also comes with the added benefit of not having to retrain the entire model if simply one class is added to the dataset, as is required by classical computer vision algorithms.

The variation of these classifiers utilized in this work introduce another advantage, they are not trained on class features, but on the similarity and differences between classes. This means that they can identify how similar or different an object is compared to what it knows from pre-existing classes, and use this similarity measure to classify the object, either as the classes that are known or potentially entirely unknown classes.

Given both the ability to easily have the model adapt to new classes and the potential use of their similarity based classification method to identify an image as unknown, gives the one-shot and few-shot classifiers a promising starting point of being used in an open world setting, where both classes that are known and unknown can be presented to the classifier. As opposed to traditional closed world classification where only the classes used during training can correctly be classified.

This thesis is an investigation into the viability of using one-shot/few-shot learners in combination with outlier detection algorithms and/or open set algorithms to create an open world learner that can identify new classes and learn to classify them correctly with no additional training and based only on a minimal amount of labeled data.

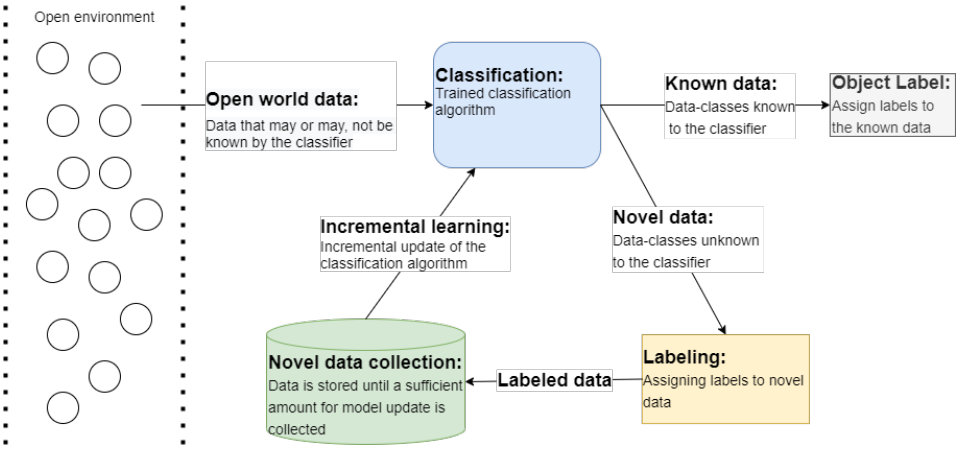


Figure 1.1: Overview of an Open World Learner algorithm

1.1 Motivation / Application

One of the main problems in open world learning is the problem of learning to classify the new object classes it is detecting. This problem is called incremental learning, to update the model in order to recognize new classes incrementally after the initial training of the model. Incremental learning is a challenge that usually requires a significant amount of labeled data and often causes the overall accuracy of the model to decrease due to the changes the model has to perform when adapting to new classes.

One-shot and few-shot classifiers are designed to adapt to novel object categories based on very few data samples, making them highly competent at generalization. The classifiers that are detailed in section 3.1, 3.2 and 3.3 has the added advantage of doing this adaption without the need to retrain the network or change any parameters. This leads to the problem of incremental learning being reduced to simply labeling a few samples from the new class and adding them to a reference dataset. This means that the only remaining problem to overcome in order to convert the one/few-shot learners into complete open world learners is to make them capable of identifying unknown classes. To the best of our knowledge, the idea of utilizing the natural generalizational ability of the one-shot and/or few-shot learners and adapt them as open world learners is a novel idea.

This work is performed for the AILARON project for use in a AUV (Autonomous Underwater Vehicle) that is performing real time monitoring and classification of underwater microbial biology, mostly in the form of the plankton species.

1.2 Aim of study

Plankton comes in a plethora of shapes and sizes and cataloguing new species of plankton is an ongoing process. Training a traditional neural network to classify all planktonic species that it might encounter in an open real world environment is an infeasible task due

to the model complexity required and the limited availability of labeled data.

The aim of the study is to classify planktonic species in-situ in an attempt to discover and catalogue unseen species for future classification. This is known as the open world learning problem. As illustrated in figure 1.1 the open world problem entails the classification of open world data, and incrementally updating the classification algorithm to adapt to new classes that it might encounter.

We also want to reduce the labeling effort as much as possible since this is a time consuming process. In this pursuit, we look into one-shot and few-shot algorithms for use as a base of our solution since these are methods that require a minimal amount of labeled data in order to adapt to new classes.

1.3 Research questions

This thesis studies the viability of using one/few -shot image classification algorithms in combination with outlier detectors/open set algorithms to create an open world learner. First, an overview of existing research is presented. Then, we carry out a thorough comparison between relevant models, and finally propose a complete open world learner framework utilizing the top scoring algorithms. The work answers the following research questions:

How well does one/few-shot algorithms classify the classes used during the training process? State-of-the-art image classifiers already achieve good results on many closed world classification task. We examine one/few -shot algorithms and determine the performance of these models on the traditional closed world classification task.

How well does one/few -shot algorithms classify classes stored in it's reference database, but is not used during training? One/few -shot classifiers are good at generalizing and are designed to be well suited for the task of transfer learning, even without the need of retraining the network each time a class is added to the classification task. This thesis explore the framework's ability to perform this well on the transfer learning task native to the one/few-shot learners.

How well does our proposed framework of one/few -shot learners combined with outlier detection/open set algorithms reject unknown/novel classes? Traditional closed world classifiers always make a erroneous classification if presented with an unknown class. Our proposed open world recognition algorithm needs to be able to perform this rejection reliably with a high accuracy. This thesis explores the proposed framework's ability to perform this rejection by identify these novel classes and marking them as unknown.

What is the most suitable algorithm in regard to the AILARON project? For use in an autonomous underwater science platform, there are several requirements that has to be met in order for a computer vision algorithm to be appropriate. It has to be able to distinguish known and unknown classes to an appropriate level of certainty and in a reliable manner. It has to be able to do this in real-time, therefore the notion of speed and computational

complexity is also an important factor to take into consideration, given that there is limited computational resources available on the AUV. The algorithm's implementation has to contribute more advantages than disadvantages, compared to traditional classifiers. Which means that several of the research questions above have to be performed with a satisfying degree of accuracy.

1.4 Contributions

In this work, we propose a framework for a complete open world learner based on the few-shot classification algorithm Prototypical network in combination with outlier detectors and/or open set recognition algorithms.

We highlight two versions of this framework: Prototypical network in combination with the outlier detector XGBOD (Extreme Gradient Boosting Outlier Detector) for the best performance in most configurations, and Prototypical network in combination with the open set recognition algorithm NNO (Nearest Non Outlier) for use-cases where speed is important and/or there is very limited available data.

To build this framework we performed extensive exploration and experimentation in order to find the best possible combination of algorithms. These algorithms include two few-shot classification algorithms and 19 outlier detector and open set recognition algorithms. Every variation of the framework was benchmarked and measured on several tests and criteria, including speed, different closed world scenarios, new class rejection capability and a final open world classification test. All tests were conducted several times for each framework variation in order to gauge the performance given differing amount of reference data. This was to get a broad view of the capabilities of the framework variations and make an informed decision for the algorithm selection for the proposed framework. In order to combine these algorithms into open world learning algorithms, there was a substantial effort put into the design adjustment and merging process to reach compatibility between each of the algorithms.

In addition to this, we looked into improving the performance of the one-shot learner Siamese network by swapping out the simple base embedding network for a more advanced network. In this process we modified several well known classification networks and adapted them to the embedding task before measuring their performance up against each other as well as the Siamese base embedding network. The Siamese network itself was also adapted into a few-shot classification algorithm in order to compare it's performance against the other few-shot classifiers for the proposed framework.

Part of the early work of this thesis mainly regarding the exploration of the Siamese network was also accepted in the student competition track of the OCEANS 2020 Gulf Coast Conference, and is set to be published in IEEE. This paper is included in the appendix along with the conference poster. The finalized work is to be submitted to the ICMV 2020 conference.

1.5 Structure of the thesis

The framework proposed in this thesis is a combination of two or more preexisting algorithms, these algorithms and their relevant background information are found in the theoretical background 2 and the literature review 3 chapters. As a lot of different algorithms are used in this thesis, not all of them are covered in detail, these are listed with a short explanation in section 2.5, the rest are found in the literature review. The framework itself is presented in chapter 4, and the implementation details to reach compatibility between the algorithms for the different variations of the framework are explained in chapter 5. The experimental setup and the selection criteria used for the evaluation of the framework variations are discussed in chapter 6. In chapter 7 the results are presented and discussed, while chapter 8 primarily focuses on highlighting and reasoning about different variations of the framework for different use-cases and answering the research questions posed in 1.3. The thesis is then concluded in chapter 9.

Chapter 2

Theoretical background

This chapter covers the conceptual fundamentals pertaining to the proposed framework. It starts by introducing some base machine learning concepts vital to modern computer vision (section 2.1) before explaining the concept of one-shot (section 2.2) and few-shot (section 2.3) learning. Then it presents the definition of open world learning (section 2.4) and the problem of open space risk needed to be overcome in order to realize the framework. Finally it covers the concept of outlier detection algorithms (section 2.5) which is one of the methods used for identifying new species, and some distance metrics 2.6 used in several of the relevant algorithms.

2.1 Machine learning

Machine learning is a sub-field field of artificial intelligence(AI). It deals with extracting patterns from raw data and using the data to create advanced statistical prediction models. In the recent years, machine learning has borrowed a lot of inspiration from biological systems, an example of this is the neural networks which have been modeled on a simple interpretation of connections in the human brain. These reinterpretations of biological intelligence has been a leading factor in the tremendous progress seen in the field of machine learning over the past few years.

2.1.1 Methods of learning

The field of machine learning is further split up into sub-fields based the types of problem they are intended to solve and the type of input data that they are designed to work with. Three of these sub-fields are supervised learning, unsupervised learning and transfer learning.

Supervised learning:

Supervised learning methods [1] require training data that is presented with labels or targets $\{(\mathbf{x_t}, \mathbf{y_t})\}$. The training is then performed by establishing a prediction model that

performs label predictions $\{\hat{\mathbf{y}_\mathbf{v}}\}$ based on some data $\mathbf{x_v}$. Then with the help of a cost function, it compares the true labels $\mathbf{y_v}$ to the predicted labels and uses this information to update the prediction model. Supervised methods typically require large amounts of data to produce useful models.

Unsupervised learning:

Unsupervised learning methods [1] only require the raw data $\{\mathbf{x}\}$. Rather than optimizing over a cost function based on labeled feedback, the unsupervised method attempts to find natural patterns and structures like clusters in the data, and use these patterns to produce the models.

Transfer learning:

Transfer learning [2] is a method of reusing knowledge from a problem with abundant data on a related problem where little or no data is available to ease the learning process of the new task. For example, the knowledge attained from identifying a set of plankton classes with a lot of data could be used to more easily learn to identify a new and rare species of plankton.

2.1.2 Artificial neural networks

Artificial neural networks (ANN) are the reason behind most of the recent success of AI, especially in the fields of computer vision, reinforcement learning and natural language processing. It even reaches superhuman performance in many tasks.

ANNs are function approximators. Given data $\{(\mathbf{x}, \mathbf{y})\}$, the networks approximates a function f, so that $y \approx \hat{y} = f(x)$. The power of the ANNs comes from the modularity of the models. Both in terms of size and complexity as well as the property that the input $\{\mathbf{x}\}$ can be a large variety of different representations and of differing intricacies. Examples of the latter includes parameterized sounds and image pixel values. ANNs are conceptually not difficult to understand, but can produce mind boggling pattern approximations. They consists of an alternating sequence of parameterized linear transformations and non-linear static transformations. Often in the form of the linear perceptrons with an non-linear activation function.

In the traditional approach for ANNs, the goal is to train the network by learning specified features and connection to match the input x to a known class y. This is done by splitting all the training data into two categories; the training set and the test set. Both sets contain the same classes y, but with different data x.

2.1.3 Activation functions

Activation functions are the non-linear transformation performed after every linear transformation. These are static and only dependent on the input of the function. They decide the activation value of the layer based on the input, weights and biases. The reason for the activation function is to make every layer more complex, and as a result, it is capable of performing more complex approximations.

ReLU:

The ReLinear Unit (ReLU) is one of the most popular activation functions for use in neural networks.

The ReLU is defined as:

$$f(x) = max(0, x) \tag{2.1}$$

It is commonly used in both regression and classification networks.

Sigmoid:

The Sigmoid activation function is defined as:

$$S(x) = \frac{e^x}{e^x + 1} \tag{2.2}$$

This activation function is popularly used as the activation function of the final layer in binary classification networks. This is because it can produce values from 0 to 1, which can be interpreted as a probability score.

Softmax:

The softmax activation function is defined as:

$$\hat{P}(y = j|\mathbf{x}) = \frac{e^{\hat{x}}}{\sum_{i=0}^{N} e^x} \tag{2.3}$$

This activation function is a popular choice for multi-class classification networks. This is attributed to the fact that it produces a value similar to a probability score for all classes, which sums up to 1. $\sum_{i=0}^{N} \hat{P}(y = i|\mathbf{x}) = 1$

The softmax activation function can also be used for minimization functions by multiplying all input with -1.

2.1.4 Back propagation

Back propagation is the method in which the ANN optimizes the parameters of the linear transformations. This is done by comparing the predicted labels $\hat{\mathbf{y}}$ with the actual labels $\mathbf{y}$ with the help of a cost function. Then an optimization algorithm can be used on the cost function to minimize it by changing the weights and biases of the ANN. It does this by considering the values produced by the cost function and calculates the desired values of the final layer of the network in order to minimize this function. Then the desired values of the second to last layer is calculated based on the desired values for the last layer. This process is propagated backwards for all the layers in the network until the network produces the optimized value for the cost function.

2.1.5 CNN - Convolutional Neural Network

For machine learning problems related to computer vision, like the problem we are looking to solve, the Convolutional Neural Network (CNN) class of ANNs have gotten tremendously popular due to their impressive performance and computational efficiency. This

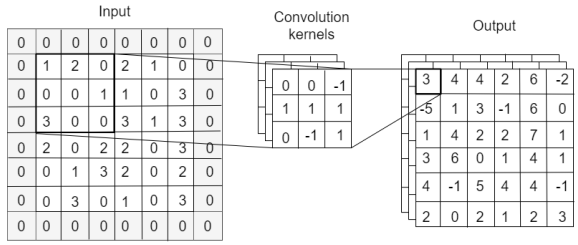


Figure 2.1: Sliding kernel dot product in CNN

efficiency is attributed to their shared weight system that drastically reduce the number weights needed per layer, which is important due to the large number of input values needed to represent a complete image.

A layer of a CNN network function by sliding parametric matrix kernels over an image, and performing the dot products between the kernels and the image pixel values at every iteration of this process. This produces m_i new feature maps which are lower level representations of the image. The sliding kernel dot product process is depicted in figure 2.1. The grey outline in figure 2.1 is the padding to insure no information on the edges of the input is lost during the matrix calculations.

This process is usually followed by an activation function and a pooling layer. The pooling layer performs an operation where it slides a 2d kernel over a feature map and produces a single value based on the values inside the kernel. This is done to gradually decrease the complexity of the data. A popular example of a pooling layer is a max pooling layer, which passes only the larges value of the feature map kernel.

By performing these series of actions several times in a trained CNN, the image features can be abstracted while at the same time reducing the complexity enough to perform image classification and/or other tasks.

2.1.6 Performance metrics

To correctly determine the performance of a model there are several different methods available. Each one has it's strengths and weaknesses, and all of them are situationally dependent in their usefulness. The ones mentioned in this thesis are all based off of the confusion matrix.

Confusion matrix:

The confusion matrix is a binary overview of the results of predictions done by a classifier, and it is the foundation for several different performance metrics. It gives insights into the errors, and what types or errors are being made by the classifier. The confusion matrix is depicted in table 2.1.

- Positive(P): Observation is predicted to be positive. Ex: Observation is predicted to be an inlier.
- Negative(N): Observation in predicted to be negative, Ex: Observation is predicted to be an outlier.

		Predicted	
		Positive	Negative
Actual	True	TP	TN
	False	FP	FN

Table 2.1: Confusion matrix example.

- True(T): Predicted assumption is true. Ex. predicted inlier is in fact an inlier.
- False(F): Predicted assumption is false. Ex. Predicted inlier is in fact an outlier.
- True Positive(TP): Observation is predicted to be positive and prediction is true.
- True Negative(TN): Observation is predicted to be negative and prediction is true.
- False Positive (FP): Observation is predicted to be positive, but prediction is false.
- False Negative (FN): Observation is predicted to be negative, but prediction is false.

Metrics:

The 4 different values of the confusion matrix in table 2.1 can be used to calculate different aspects of the performance of a machine learning model. The 4 most common are as follows:

Accuracy:

Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

This accuracy is easy to understand but only viable if there doesn't exist a significant class imbalance.

Precision:

Precision is defined by:

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

If the prediction is positive, how often is that prediction true.

Recall:

Recall is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

Given that that the actuality is true, how often is it predicted correctly.

F1-score:

F1-score is defined as:

$$F_1 = 2\frac{precision * recall}{precision + recall} \tag{2.7}$$

The F1-score is a weighted average of precision and recall. It is often favoured over accuracy when dealing with imbalanced datasets.

2.2 One-shot learning

In traditional supervised image classification techniques [3][4][5][6][7], the goal is for the model is to identify class specific features in order to make a class prediction. These features are highly trained and specialized for every class in a dataset. Requiring a lot of examples and computational power in order to recognize generalized features and produce accurate predictions. These techniques are therefore ill suited for cases where only a small amount of data is available.

The goal of one-shot learning is to successfully recognize classes previously unseen in the training process with the aid of just a single example of that class. Drastically decreasing the amount of labeled data required to recognize new classes. In order to do this, a one-shot algorithm has to be able to employ some sort of knowledge transference scheme. Using general information previously learned from the training classes to quickly and efficiently adapt to new data.

The one-shot setting is an extremely challenging problem given the huge information variance that can occur even within a single class of any dataset. Because of this, one-shot learning is often used in narrower domains where the variance is smaller and/or more controllable, like facial recognition [8][9], or character recognition [10]. This field of research is still in its infancy but have already seen some promising developments which are discussed in the literature review.

When training and testing a one-shot classifier 2 datasets are used, a background set and an evaluation set. The **background set** contains the classes used for training, while the **evaluation set** contains the classes used for validation. Note that no classes appear in both datasets.

2.3 Few-shot learning

The few-shot learning problem is only a short extension of the one-shot learning problem. The goal of few-shot learning is to successfully recognize classes previously unseen in the training process with the aid of just a few examples of that class. The difference between the two problems being the amount of reference samples given in order to classify a previously unseen class. Consequently few-shot learning algorithms needs more labeled data from new classes than is required for one-shot algorithms.

The additional data given to a few-shot algorithm compared to a one-shot algorithm also increases the models potential to generalize from the previously unseen data. In general, few-shot learners score significantly better at classification tasks in comparison to

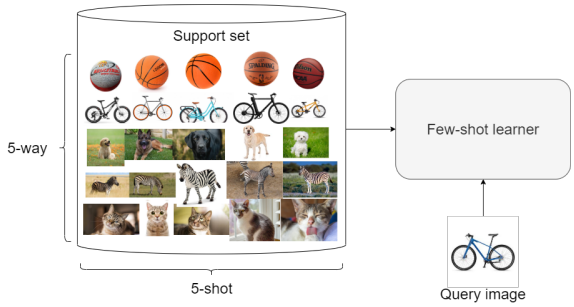


Figure 2.2: 5-way 5-shot example

one-shot learners [11] [12] [13] at the cost of requiring more labeled data and more computational power. Some methods of solving the few-shot problem relevant to this thesis are explained in detail in the literature review, section 3.2 and 3.3.

Few-shot notation:

For few-shot classification tasks only a small support set S consisting of N_N labeled examples $S = (\mathbf{x}_1, y_1), ..., (\mathbf{x}_{N_N}, y_{N_N})$ is given to the classifier prior to the classification task. In S each example $\mathbf{x}$ is represented in a d-dimensional space $\mathbf{x}_i \in \mathbb{R}^d$, and $y_i \in 1, ..., K$ is the corresponding label. S_k is the set of examples belonging to class K and N denotes the length of S_k, aka. number of samples per class.

In a few-shot setting, a task is characterized by the numbers K and N, and is referred to as a $\mathbf{K}$-way, $\mathbf{N}$-shot task. A typical example is to present a classifier with 5 classes (5-way) each with 5 data samples (5-shot), and then predict the class of the query sample out of the 5 classes. Figure 2.2 visualizes this.

2.4 Open-world learning

2.4.1 Definition

Open world learning is defined by [14] as follows:

1. At a particular time point, the learner has built a multi-class classification model F_K based on all past N classes of data $D^P = D_1, D_2, ..., D_K$ with their corresponding class labels $Y^K = l_1, l_2, ..., l_K$. F_K is able to classify each test instance to either one of the known classes $l_i \in Y^K$ or reject it and put it in a rejected set R, which may include instances from one or more new or unseen classes in the test set.

2. The system or a human user identifies the hidden classes C $\in$ R, and collects training data for the unseen classes.

3. Assume that there are k new classes in C that have enough training data. The learner incrementally learns the k classes based on their training data. The existing model F_K is updated to produce the new model F_{K+1}

This means that the algorithm can perform open set classification, which is equivalent to classification with rejection of unknown classes (further explained in section 2.4.2 under the title of open set rejection, as this is the computer vision term for open set classification). Given sufficient data in the rejection category, the data can be sorted into classes, either by the algorithm itself or a human, and these classes can be used to iteratively update the classifier to gain the ability to classify the new classes.

Nomenclature:

Below is a few definitions that is important to know to fully understand the following sections. They are used actively throughout the thesis. A full nomenclature list is presented at the beginning of the thesis.

Open set recognition - Open set classification in the field of computer vision.
Open world recognition - Open world learning in the field of computer vision.
Training data/classes - Data/classes used to train the classifier
Known data/classes - Data/classes the classifier has obtained previous information about, but not used during training, like the support set in few-shot learning.
Unknown data/classes - Data/classes the classifier has no previous information about.
Closed world - All samples are from training/known data
Open world - Samples might be from training/known data or unknown data

2.4.2 Open set recognition

Open set recognition (OSR) is image classification with rejection of unknown classes. The rejection feature of OSR does however introduces a few problems. First is that it implies that not all classes are known, rendering Bayes' law and the law of total probability not directly applicable to the problem. This is due to the normalization of the probability performed by these equations, which seizes to be a constant if new classes, and their respective probabilities are added iteratively, limiting the interpretation of the total probability. Another problem is the open space risk, this is a more comprehensive problem and is discussed separately in section 2.4.3.

2.4.3 Open space risk

The decision boundary is a fundamental tool of classification, they are optimized to be the differentiator between different classes, a typical closed world example of decision boundaries in a multi-class setup is shown in figure 2.3.

If we were to introduce open world data to this scenario it might however look quite different. An examples is shown in figure 2.4. There is a new sample introduced. In a closed world setting, a classifier would mark this new data as the "blue" class, despite being a significant distance away from the cluster of blue objects. An open set recognition algorithms would probably (correctly) reject it as it is a novel class. This is the open space risk. The risk of labeling too much space with no positive samples of a class as that class.

Open space risk was first formalized by Scheirer et al. in their 2013 paper [15]. Let S_O be a ball of radius r_O that contains the positively labeled open space O and all known positive training examples. And let f be the estimated classification function, and $f_y(x) =$

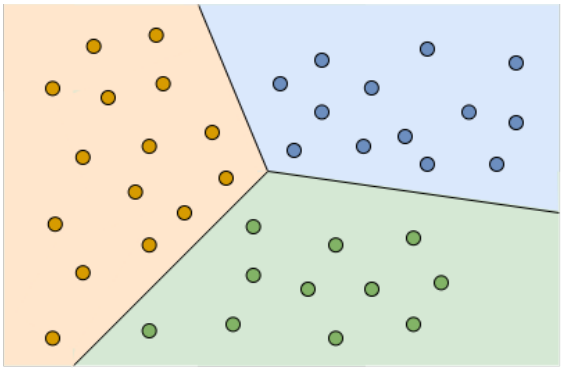


Figure 2.3: Multi-class decision boundaries

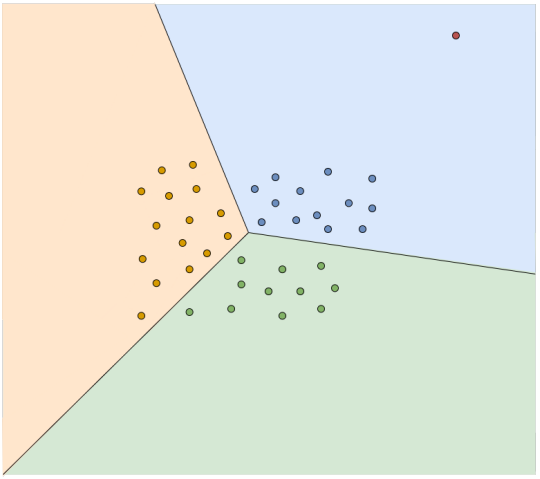


Figure 2.4: Open space risk example

1 where class y is estimated positive and $f_y(x) = 0$ where class y is estimated negative. Then probabilistic open space risk $R_O(f)$ of function f for a class y can be defined as:

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f_y(x)dx}{\int_{S_O} f_y(x)dx} \tag{2.8}$$

This indicates that the more space that is labeled as open, the greater the open space risk.

This definition was iterated on by Fei et al. in their 2015 paper [16], by specifying the positively labeled open space O. They defined it as the positively labeled area that is sufficiently far from the center of the positive training examples. And explained it by letting $B_{r_y}(cen_y)$ be a ball of radius r_y with the same center as a positive class $y(cen_y)$, and ideally encompasses all positive samples from class y. They further specified S_O to be a larger ball $B_{r_O}(cen_y)$ sharing the same center cen_y and with a radius of r_O. Then the classification function $f_y(x) = 1$ when $x \in B_{r_O}(cen_y)$ and $f_y(x) = 0$ when $x \notin B_{r_O}(cen_y)$. Then they defined open space as:

$$O = S_O - B_{r_y}(cen_y) \tag{2.9}$$

And radius r_O is estimated from the training data and used as the decision boundary of f.

2.5 Outlier detection

An outlier is an object that is outside of the norm, and for some reason should not be included in set of objects. The goal of outlier detection is to identify these types of objects. The definition of ”outside the norm” can vary, but in statistical analysis it is usually based on a distance metric from a cluster of objects.

As with machine learning, there are 2 main branches of outlier detection: supervised and unsupervised, and they follow the same definition as in section 2.1. There are many different methods developed to perform outlier detection as they are useful in a plethora of fields and applications. In this thesis they are utilized for reducing the open space risk by rejecting queries that are outside of known class clusters.

16 different outlier detectors are tested during the research of this thesis, because of their varying level of importance to the work, most of them are only briefly explained in the list below, the more complex and significant outlier detectors are explained in grater detail in the literature review.

- **Angle Based Outlier Detection (ABOD) [17]:** Outliers are determined by the variance of it’s cosine distance scores to all it’s neighbours.
- **Average K Nearest Neighbours (AvgKNN) [18]:** Outliers are determined based on the average of the distances to the k nearest neighbours.
- **Connectivity-Based Outlier Factor (COF) [19]:** Outliers are determined by the average chaining distance compared to the average chaining distance of it’s neighbours.

- **Deviation-based Outlier Detection (LMDD) [20]:** Outliers are determined by the smoothing factor: Amount of dissimilarity that can be reduced by removing a subset of samples.
- **Extreme Gradient Boosting Outlier Detector (XGBOD) [21]:** Explained in section 3.7.
- **Feature Bagging [22]:** Outliers are determined by training several estimators over subsets of the data.
- **Isolation Forest (IForest) [23]:** Outliers are determined by randomly selection features and selecting an arbitrary value between the maximum value and the minimum value of the selected feature to try to isolate the sample.
- **K Nearest Neighbours (KNN) [18]:** Outliers are determined based on the distance to the k-th nearest neighbour.
- **Lightweight On-Line Detector of Anomalies (LODA) [24]:** Outliers are determined by using an ensemble of weak outlier detectors.
- **Local Outlier Factor (LOF) [25]:** Outliers are determined by estimating the local density of a sample with respect to it's neighbourhood.
- **Local Correlation Integral (LOCI) [26]:** Similar to LOF, but also handles clusters of outliers.
- **Median K Nearest Neighbours (MedKNN) [18]:** Outliers are determined based on the median of the distances to the k nearest neighbours.
- **One-Class Support Vector Machines (OCSVM) [27]:** Outliers are determined by creating a high dimensional decision boundary around the normal data.
- **Principal Component Analysis (PCA) [28]:** Outliers are determined by projecting the data into a low dimensional hyperplane based on the eigenvectors of the data.
- **Subspace Outlier Detection (SOD) [29]:** Outliers are determined by creating subspaces spanned by it's neighbours.
- **Stochastic Outlier Selection (SOS) [30]:** Outliers are determined based on a concept proportional to similarity, and if all other objects have are sufficiently different from it.

2.6 Distance metrics

In order to determine if a sample is an inlier or an outlier, most outlier detectors use a distance measure of some kind, either directly like the KNN or indirectly like calculating the distances between all objects to recognize clusters. This distance can also be measured in several different ways, the three relevant distance metrics for this thesis are explained below.

Euclidean distance:

The euclidean distance is the length of the straight line from point A to point B and is defined by the formula:

$$d(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{m}(A_i - B_i)^2} \tag{2.10}$$

Where m is the total number of dimensions of the vector space points A and B are located in, and A_i is the i-th dimensional value of A.

Squared Euclidean distance:

The squared Euclidean distance is the formula of the Euclidean distance metric with the exception that it does not take the square root. This allows for increased computational speed. It is defined by the formula:

$$d(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{m}(A_i - B_i)^2 \tag{2.11}$$

Where m is the total number of dimensions of the vector space points A and B are located in. And A_i is the i-th dimensional value of A.

Cosine distance:

The cosine distance, also known as cosine similarity is the angle between two points from the perspective of the origin. This means that it does not take into regard either weight or magnitude. Cosine distance is defined by the formula:

$$d(A, B) = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{m} A_i B_i}{\sqrt{\sum_{i=1}^{m} A_i^2}\sqrt{\sum_{i=1}^{m} B_i^2}} \tag{2.12}$$

Where m is the total number of dimensions and A_i is the i-th dimensional value of A.

Chapter 3

Literature review

This chapter covers the literature review relevant for the exploration and development of the proposed framework. Every section focuses on one theoretical concept. It begins by introducing the one-shot algorithm Siamese-network (section 3.1) which heavily inspired this research. Then the Matching network (section 3.2) and the Prototypical network (section 3.3) which both iterates on the performance of the Siamese network are discussed. Sections 3.4, 3.5, 3.6 and 3.7 then summarize various outlier detectors and open world recognition algorithms used to combine with the few-shot learners to create the proposed framework.

3.1 Siamese Neural Networks for one-shot learning

This research was heavily inspired by the Siamese network as it is a one-shot learner that can reject unseen classes. However, as shown in section 6.5 the performance is incompetent for the task of open world learning.

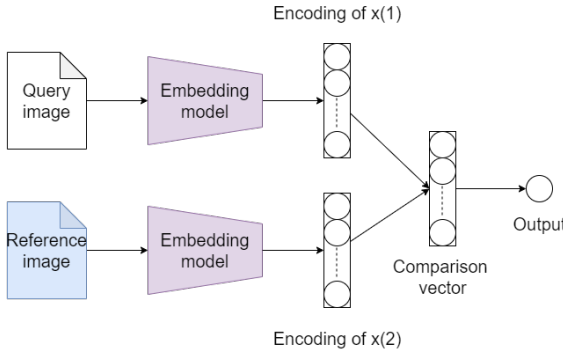


Figure 3.1: Siamese network architecture

Siamese network was first introduced in [8][31] for use in the domain of facial recognition and was in 2015 was adapted as an approach for the one-shot learning problem by Koch et al. [10].

In short, the Siamese network for one-shot learning looks at two images and tries to evaluate the similarity between them. It does this by first passing one image through a CNN based neural network that outputs an M-dimensional embedding of the inputted image in an M-dimensional similarity space $f_\phi : \mathbb{R}^U \rightarrow \mathbb{R}^m$ where $\mathbb{R}^U$ is the high dimensional image space and ϕ are the trainable parameters of the model. A second image is passed through the same network and produces a different M-dimensional vector. By using the distance/difference between these image embeddings in the learned similarity space, the model can identify whether they belong to the same or different classes. The siamese network does this by performing a vector subtraction on the embeddings before passing them through a fully connected layer followed by a softmax activation function which finally produces the binary 'match' or 'not a match' classification. The Siamese network is depicted in figure 3.1.

The siamese network utilizes *Triplet loss*. A training scheme developed by [9], also for use in facial recognition. This method is based on using 3 images per training episode: [A], [P] and [N]. Image [A] is the anchor image. Image [P] is the positive image which is of the same class as [A], and [N] is the negative image which is of a different class than [A]. We denote the embedding function of the Siamese network as f(*).

Given this, a well suited embedding function produces outputs that reduces the distance between f(A) and f(P) while increasing the distance between f(A) and f(N), resulting in the function:

$$\big\|f(A) - f(P)\big\| - \big\|f(A) - f(N)\big\| + \alpha \leqslant 0 \tag{3.1}$$

Where α is a defined margin to avoid the trivial answers of:

$$f(*) = 0 \ \cup \ \big\|f(A) - f(P)\big\| = \big\|f(A) - f(N)\big\| \tag{3.2}$$

In addition to adapting to new classes with just one example image another big advantage the siamese model holds over traditional classifiers is its ability to classify previously unknown classes. In the case of the traditional classifiers, adding a class to the dataset does not only require a large amount of data from the new class, but also incurs the need of retraining the entire model from scratch, possibly incurring a performance loss due to the newly introduced class. A one-shot learner model is per definition quick to adapt to new classes, requiring only one sample of the class. The siamese network only needs to be presented with a single reference image, with the model already trained to extract features from the image, rendering the need for any further training of the network obsolete. Hence, it does not suffer any performance loss of other classes as a consequence of the introduction of the new class.

A bonus of using the *Triplet loss* training scheme, is that training data is generated by matching two different images at a time, either from the same class or different classes. This matching of images for a training episode results in an augmentation of the dataset to $N!$ samples, where N is the number of images originally in the dataset.

In theory, a one-shot model such as the siamese network does not learn general features, but rather how to extrapolate them from the images it is presented. This is probably

not true, given the known practical limitations of the model. It works well in narrow domains such as character recognition with the omniglot dataset[32], and facial recognition with datsets like Labeled Faces in the Wild[33], but is not well suited for more diverse datasets like imageNet[34] as was shown in [11].

A notable weakness with this approach is that the accuracy of the classification is not only dependent on the quality of the query image like in a traditional classifier, but also on the quality of the reference image. Resulting in one additional point of failure compared to the traditional classifiers.

Algorithm 1 shows the evaluation computation for the siamese net, with one reference image and one query image. In the algorithm RANDOMSAMPLE(S,N) denotes a set of N elements chosen uniformly at random from set S, without replacement. Q_x and S_x are the data of the query and support example while Q_y and S_y are the classes belonging to those examples.

Algorithm 1 Evaluation computation for Siamese network.

Require: Trained network $f_\phi(.)$ where ϕ denotes the trainable parameters.
Require: Trained last network layer $L_{\phi'}$
Require: Background set D = $(x_1, y_1), ..., (x_N, y_N)$, where each $y_i \in 1, ..., K$.
1: $S \leftarrow$ RANDOMSAMPLE$(D, 1)$ {Select reference example}
2: $Q \leftarrow$ RANDOMSAMPLE$(D, 1)$ {Select query example}
3: $P(S_y = Q_y|Q_x) = L(d(f_\phi(Q_x), f_\phi(S_x)))$
4: **if** $P(S_y = Q_y|Q_x) > 0.5$ **then**
5: Assign $Q_y = S_y$
6: **else**
7: Reject input
8: **end if**

3.2 Matching Networks for One Shot Learning

The matching network, conceptualized by Vinyals et al. in their paper Matching Networks for One Shot Learning[11] is conceptually similar to the Siamese network, but it significantly improves the performance. The Matching network uses images as input and produces multi-dimensional similarity based embeddings for those images with a trained convolutional network. However there are two important distinctions; firstly, while the Siamese network is optimized for a true/false classification, the Matching network can perform multi-class classification. The other distinction is that while chasing performance improvements, the Matching network sacrifices the ability to reject unseen classes. The Matching network is shown in figure 3.2.

The matching network does this by allowing for n different images from k different classes to be used as input for the network, producing $(k * n)$ different embeddings in the similarity space, and using formula 3.3 to calculate the probability for the query class belonging to either of the k different classes.

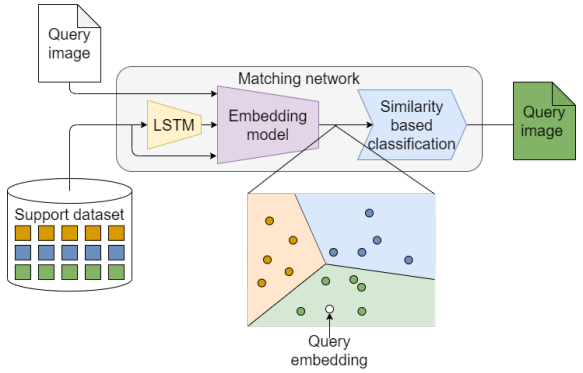


Figure 3.2: Matching network architecture

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{3.3}$$

Where $x_i$and y_i are the samples and labels of the support set S and $\hat{x}$ and $\hat{y}$ is the query image and the corresponding predicted class. The function $a(*)$ is referred to as the attention kernel. $a(*)$ is defined as:

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}} \tag{3.4}$$

Where f(*) is the embedding function for the query image, while g(*) is the embedding function for the support set. The authors Vinyals et al. also specified the option of letting $f(*) = g(*)$ to simplify the model. Function 3.4 calculates the cosine distance between the query embedding and the different embeddings of the support set and uses these distance-values to calculate the softmax score for each class.

These two functions (3.3 and 3.4) produce the equivalent of a kernel density estimator. Not too dissimilar from an advanced nearest neighbours estimator. Hence, the matching net assigns the query image to one of the k classes in the support set by comparing the query embedding with the point distribution belonging to different classes from the support set.

Vinuals et al. also highlights a problem with this approach on its own; depending on the variance of the classes, the Matching network have differing performance due to the risk of overlapping classes in the embedding space when classifying closely related classes. For instance, if the model is trained on imageNet[34], the model is expected to tell the difference between a dog and a bird, as well as different species of birds. The latter being a more nuanced problem, possibly exhibiting the previously mentioned overlapping classes problem. To mitigate this Vinyals et al. also proposed a solution they called *Full context embeddings*: Before applying the embedding function $g(*)$ to the support set, the support set images are run through an LSTM network with determines the context/nuance of the set. This LSTM then changes the encoding strategy of $g(*)$ resulting in a function

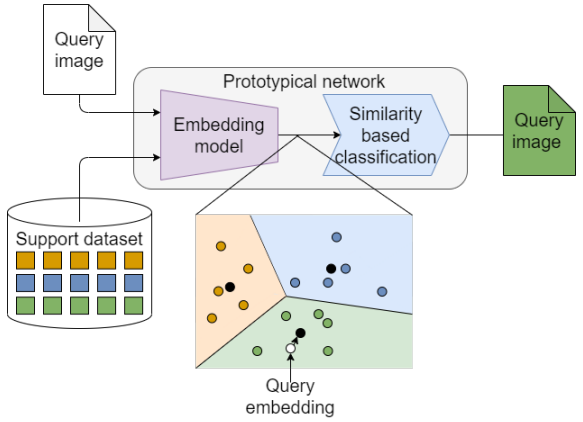


Figure 3.3: Prototypical network architecture

$g(x_i, S)$ where S is the support set, allowing for encoding of images in the context of the entire support set.

3.3 Prototypical Networks for Few-Shot Learning

Matching networks greatly improved on the accuracy of the Siamese net and improved on its general functionality by allowing for testing of a query image against all known classes simultaneously. Despite this, it still suffers from the drawback of the unnecessarily complicated Full context embedding network to compensate for closely related classes overlapping in the embedding space.

Snell et al. iterated on this by introducing the Prototypical network [12], proposing a simplification by getting rid of the Full context embedding network and replacing it with class prototypes; instead of using the class cluster densities in the classification process, Snell et all. simply proposed to compute the mean position of all the known class support embeddings, and classify a query embedding as the same class belonging to the closest prototype. This would nullify ambiguity of class overlap as well as making the decision boundaries more easily interpretable. The Prototypical network is depicted in figure 3.3.

Through empiric experimentation, they also discovered that using Bregman divergences [35] as a distance measure was favourable to other measures such as cosine distance proposed for the matching network. They settled on the simplest Bregman divergence, the Euclidean square distance. This distance is calculated from a query embedding to the class prototypes to produce the class predictions. This prediction is done by the softmax function over the negative distances.

$$p_\phi(y = k|x) = \frac{exp(-d(f_\phi(x), \mu_k))}{\sum_{k'} exp(-d(f_\phi(x), \mu_{k'}))} \tag{3.5}$$

Where μ_k is the prototype for class k, and $f_\phi(x)$ is the embedding of query x given trainable parameters ϕ, and $d(a, b)$ is the distance between a and b.

During learning the optimization is performed on the negative log-probability $J(\phi) = -log\, p_\phi(y = k|x)$ of the true class k.

Algorithm as written in the original article can be seen in algorithm 2. The only change is some variables to adhere to the naming convention used in this thesis. Note that this algorithm trains with N_Q number of query images per class per training episode, RANDOMSAMPLE(S,N) denotes a set of N elements chosen uniformly at random from set S, without replacement and Q_x is the input data of query Q

Algorithm 3 is an algorithm explaining an evaluation episode of the prototypical network after the initial training.

Algorithm 2 Training episode loss computation for Prototypical Networks.

Require: Training set D = $(x_1, y_1), ..., (x_N, y_N)$, where each $y_i \in 1, ..., K$.
D_k denotes the subset of D containing all elements (x_i, y_i) such that $y_i = k$

$V \leftarrow$ RANDOMSAMPLE$(1, ...K, N_c)$ {Select class indices for episode}
for k in $1, ..., N_C$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE(D_{V_k}, N_S) {Select support examples}
  $Q_k \leftarrow$ RANDOMSAMPLE$(D_{V_k}/S_k, N_Q)$ {Select query examples}
  $\mu_k \leftarrow \frac{1}{N_C} \sum_{(x_i,y_i)\in S_k} f_\phi(\mathbf{x}_i)$ {Compute prototype from support examples}
end for
$J \leftarrow 0$ {Initialize loss}
for k in $\{1, ..., N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**

$$J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_\phi(\mathbf{x}), \mu_k) + log \sum_{k'} exp(-d(f_\phi(\mathbf{x}), \mu_{k'})) \right] \tag{3.6}$$

    {Update loss}
  end for
end for

3.4 Towards open world recognition

In the paper [36], Bendale et al. proposes the Nearest non outlier (NNO) open world recognition algorithm. The Nearest non outlier utilizes model vector $M = [\mu_1, ..., \mu_K]$ of class mean embedding vectors and a radius vector $\mathbf{r} = [\tau_1, ..., \tau_K]$ to represent the decision boundary for all classes. The multi-class open set recognition function is then defined by:

$$\varphi = [f_1(x), ..., f_K(x)] \tag{3.7}$$

Where $f_k(x)$ represents the likelihoods of query x being in class k. The predicted class $\hat{y}$ is calculated by:

Algorithm 3 Evaluation episode computation for Prototypical Networks.

Require: Trained network $f_\phi(.)$ where ϕ denotes the trainable parameters.
Require: Training set D = $(x_1, y_1), ..., (x_N, y_N)$, where each $y_i \in 1, ..., K$.
D_k denotes the subset of D containing all elements (x_i, y_i) such that $y_i = k$
$V \leftarrow$ RANDOMSAMPLE$(1, ...K, N_c)$ {Select class indices for episode}
for k in $1, ..., N_C$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE(D_{V_k}, N_S) {Select support examples}
  $\mu_k \leftarrow \frac{1}{N_C} \sum_{(x_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$ {Compute prototype from support examples}
end for
$Q \leftarrow$ RANDOMSAMPLE$(V, 1)$
$p_\phi(y = k|x) = \frac{exp(-d(f_\phi(Q_x), \mu_k))}{\sum_{k'} exp(-d(f_\phi(Q_x), \mu_{k'}))}$ {Calculate class probability}
$\hat{y} = argmax_k P(y = k|\mathbf{x})$ {Assign class $\hat{y}$ to query Q}

$$\hat{y} = \underset{y \in K, f_y(x) \in \varphi(x)}{\operatorname{argmax}} f_y(x) \tag{3.8}$$

$$F(x) = \begin{cases} 0 & \text{if } f_{\hat{y}}(x) \leqslant 0 \\ \hat{y} & \text{otherwise} \end{cases} \tag{3.9}$$

Where $F(X) = 0$ is the rejection case, and the estimate of $f_k(x)$ is defined as:

$$\hat{f}_k(x) = \frac{\Gamma(\frac{m}{2} + 1)}{\pi^{\frac{m}{2}} \tau^m} (1 - \frac{1}{\tau} \left\| W^T x - W^T \mu_i \right\|) \tag{3.10}$$

Where $\hat{f}_k(x)$ is the estimated probability of x belonging to class k. The first fraction is the inverted definition of an m-dimensional sphere with radius τ. Resulting in a tent-like probability function with the center around the mean embedding vector μ_K. τ is optimized over the $\hat{f}_k(x)$ values for all training data.

A big advantage of this model as opposed to a 1-vs-all support vector machine is that this modelling is done for all classes separately so adding a class is as simple as adding one entry to both M and $\mathbf{r}$, rather than retraining the entire model.

Algorithm 4 shows a probability estimation with rejection of unknown or uncertain inputs with the NNO model.

3.5 Towards Open Set Deep Networks

In the paper [37], Bendale et al. introduces the open set classification algorithm Openmax. With it's main purpose of combating fooling data[1] used on deep networks, Bendale et al. details an algorithm to serve as an open set alternative to the softmax activation function. Bendale et al. calls this the openMax activation function.

It works by utilizing a support vector machine similar to a cluster density algorithm to produce a probability score of the query being inliers and outliers of the different classes.

[1] Data specifically designed to act as the red circle in figure 2.4 in order to fool classifiers

Algorithm 4 NNO probability estimation.

Require: Class mean embedding vector: $= [\mu_1, ..., \mu_K]$
Require: Trained class radius vector: $\mathbf{r} = [\tau_1, ..., \tau_K]$
1: $\hat{f}_k(x) = \frac{\Gamma(\frac{m}{2}+1)}{\pi^{\frac{m}{2}}\tau_k^m}(1 - \frac{1}{\tau_k}\left\|W^T x - W^T \mu_i\right\|)$
2: **if** $max_k \hat{f}_k(x) \leqslant 0$ **then**
3: Reject input
4: **else**
5: $\hat{y} = argmax_k \hat{f}_k(x)$
6: **end if**

It uses the inlier probability to modify the class scores produced by the network, and it uses the outlier probability to create a rejection class. Then it takes all modified class scores and the rejection class and passes them all through a normal softmax activation function, resulting in an open set classification algorithm. It does not take part of the base training of the network itself, so a traditional softmax activation function is used during training of the network parameters. Just after training is the softmax function switched out with the openMax, which then has to be trained by itself.

Instead of looking at the output of the neural network, the proposed activation function utilizes values produced in the penultimate layer of the neural network. The penultimate layer is the last layer of the neural network before the final softmax activation function in the original network architecture. The activation in the penultimate layer due to input passed through the network is called an activation vector $\mathbf{v_i}(\mathbf{x})$.

The training of the openMax activation function is performed by first computing the mean activation function (MAV) for every class k. This is defined as the class mean of all the activation vectors that lead to a correct classification in the native softmax version of the network. Then K individual weibul distribution fittings [38] are fitted based on the distance between the class MAV and all the individual correctly classified activation vectors of that class. This produces the fitting model $\rho_j = (\tau_i, \lambda_i, \kappa_i)$. K is the total number of training classes.

The classification is performed by using the weibul fitting model to get information if a query falls inside or outside the current known distribution of class samples. Instead of using the weibul fitting directly, Bendale et al. used the inverted weibul fitting and subtracted 1, as seen in algorithm 5. (This weibul fitting was also used more directly to create an outlier detection svm [39] known as w-svm in [40].) This fitting is interpreted as the probability of a query being an outlier/inlier of the known classes, and this score is used to update the activation vector, while the reminding probability of being an outlier is used in the creation of a custom activation vector for the query belonging to a novel class as seen in line 6 of the openMax algorithm. To reduce the impact of classes with lesser probability of being the correct class, Bendale et al. introduced a weighing term on the weibul fitting $\frac{\alpha - i}{\alpha}$, in order for only the top α classes to contribute towards the probability of the query belonging to a novel class. Finally the revized activation vectors alongside the novel class activation vector is used as input for the classic softmax activation function.

Algorithm as written in the original article:

Algorithm 5 OpenMax probability estimation with rejection of unknown or uncertain inputs

Require: Activation vector for $\mathbf{v}(\mathbf{x}) = v_1(x), ..., v_N(x)$
Require: **means** μ_j and libMR models $\rho_j = (\tau_i, \lambda_i, \kappa_i)$
Require: α, the number of "top" classes to revise
Let $s(i) = argsort(v_j(x))$; Let $\omega_j = 1$
for $i = 1, ..., \alpha$ **do**

$$\omega_{s(i)}(x) = 1 - \frac{\alpha - i}{\alpha} e^{-\left(\frac{x - \tau_{s(i)}}{\lambda_{s(i)}}\right)^{\kappa_{s(i)}}} \tag{3.11}$$

{Perform inverted weibull fitting and subtract 1}
end for
Revise activation vector $\hat{v}(x) = \mathbf{v}(\mathbf{x}) \circ \omega(\mathbf{x})$
Define $\hat{v}_0(x) = \sum_i v_i(x)(1 - \omega_i(x))$ {$\hat{v}_0(x) =$: novel class activation vector}

$$\hat{P}(y = k|x) = \frac{exp(\hat{\mathbf{v}}_\mathbf{k}(x))}{\sum_i^K exp(\hat{v}_i)} \tag{3.12}$$

Let $\hat{y} = argmax_k P(y = k|\mathbf{x})$
Reject input if $\hat{y} == 0$ or $P(y = \hat{y}|\mathbf{x}) < \epsilon$

3.6 DOC: Deep Open Classification of Text Documents

Deep Open classification (DOC) [41] by Shu et al. is a newer and simpler alternative for open set classification. It proposes to switch out the closed set softmax activation function with K number of Sigmoid functions, each representing the individual probability for a class. Where K is the the number of training classes. The classification is then done by selecting the class with the highest individual probability value given by the Sigmoid functions, or rejecting the query as a new class if all classes fall bellow their thresholds.

$$\hat{y} = \begin{cases} \text{reject, if } Sigmoid(\mathbf{V_i}(\mathbf{x})) < \tau_i, \forall l_i \in \mathbb{Y}^\mathbb{K} \\ \underset{l_i \in K}{\text{argmax}}\, Sigmoid(\mathbf{V_i}(\mathbf{x})), \text{ otherwise} \end{cases} \tag{3.13}$$

The entire network is trained by the loss function:

$$Loss = \sum_{i=1}^{K} \sum_{j=1}^{N} -\mathbb{I}(y_j = l_i)\, log\, p(y_j = l_i) - \mathbb{I}(y_j \neq l_i)\, log(1 - p(y_j = l_i) \tag{3.14}$$

Which is simply the summed loss over all the K sigmoid functions over all the training data N.

Normally the threshold set on sigmoid activation functions is 0.5, but Shu et al. noticed that for most classifications, there was a high degree of certainty, usually above the 90 percentile mark. So they introduced the idea of applying a Gaussian fitting for the certainty

of all the correctly predicted training data. This Gaussian fitting could then propose a threshold value for each class individually, further decreasing the open space risk.

This method can be used on many different types of networks like CNN, RNN and LSTM. Shu et al. decided to test with CNN to make a valid comparison the openMax algorithm.

Algorithm 6 shows a probability estimation with rejection of unknown or uncertain inputs with the DOC model.

Algorithm 6 DOC probability estimation with rejection of unknown or uncertain inputs

Require: Activation vector for $\mathbf{v}(\mathbf{x}) = v_1(x), ..., v_K(x)$
Require: $\sigma = \sigma_1, ..., \sigma_K$ Variance from the Gaussian fittings.
Require: Vectorized input data x
$P(y = k|\mathbf{x}) = Sigmoid(\mathbf{v}(\mathbf{x}))$
Let $s(i) = argsort(P(y = k|\mathbf{x}))$
for $i = 1, ..., K$ **do**
  if $P(y = s(i)|\mathbf{x}) > 1 - \sigma_{s(i)}$ **then**
    $\hat{y} = s(i)$
    Break
  end if
end for
if $P(\hat{y}|\mathbf{x}) == None$ **then**
  Reject input
end if

3.7 XGBOD: Extreme Gradient Boosting Outlier Detector

The XGBOD algorithm [21] is an ensemble outlier detection algorithm. Meaning that it combines several other outlier detection algorithms into one. This is often a helpful way to reduce the bias of any one algorithm, while at the same time getting lower variance on the prediction, usually resulting in improved accuracy.

The XGBOD algorithm consists of 3 main parts:

1. **Base model prediction:** First the XGBOD algorithm utilizes c number of unsupervised outlier detection algorithms to create c different scores of "outlyingness" for each training sample.

2. **Classifier selection:** Then p number of models out of the c are chosen ($p \subseteq c$) based on the overall accuracy and the degree of classifier correlation. The goal is to incrementally choose the group of classifiers that has the highest accuracy, and have low correlation in order to increase variance of the prediction over the training data. This is done by checking the classifiers with the function:

$$\Psi(\Phi_i) = \frac{ACC_i}{\sum_{j=1}^{S} |\rho(\Phi_i, \Phi_j)|} \tag{3.15}$$

Where Φ_i is the results of the outlier detection algorithm nr. i, S is the list of already accepted classifiers and ρ is the Pearson correlation coefficient [42]. Then a refined feature space is created by combining the the original outlier detection space X and S to get $[X, S]$.

3. **Prediction with XGBoost:** Finally the supervised XGBoost [43] classifier is applied to the refined feature space to classify outliers vs inliers. This classifier is chosen because it is designed for high dimensional datasets as well as being robust against class imbalanced datasets, which is significant because supervised outlier detection is inherently a class invariant problem.

Chapter 4

Proposed framework

In order to solve the open world problem with minimal labeling effort as introduced in section 1.2, we propose the novel open world learner framework based on the few-shot classifiers detailed in sections 3.2 and 3.3. This framework is able to classify known classes, reject unknown classes and quickly adapt to new classes without any additional training required. The proposed framework is shown in figure 4.1.

This framework consists of 4 main modules: The support dataset, the embedding module, the classification module and the outlier detection module. **The support dataset** is the reference database, containing only a few samples from each known class and is considered the memory of the model. **The embedding module** is a neural network trained to cluster similar objects and separate dissimilar objects in a high dimensional similarity space. The module is trained as the embedding module of a few-shot learner and its output are the embeddings from the support set and open world data. These embeddings are passed on to both the classifier and the outlier detector. **The classification module** performs closed world classification on the open world data based on it's similarity with the inferred embedding clusters of the different reference classes from the support dataset. **The outlier detection module** utilizes these same inferred embedding clusters from the support dataset to determine if the open world data belongs to the known classes or if it is a novel class. By combining the classifier with the outlier prediction, we produce a classification with rejection, aka. an open world prediction.

By utilizing the few-shot classifier's ability to generalize to new classes that it hasn't seen during training, simply by including a few labeled samples of that class in the support dataset, our framework can iteratively add new classes from the rejected data without any additional training. This iterative learning process is normally a big problem that can quickly lead to a loss of accuracy, but our framework bypasses this problem by not reacquiring further training.

To best answer the research questions from section 1.3 we want to identify the best combinations of algorithms that produce the most proficient version of this framework. This includes investigating the performance of different few-shot learners and reviewing their performance in combination with different algorithms that can adapt it to the open

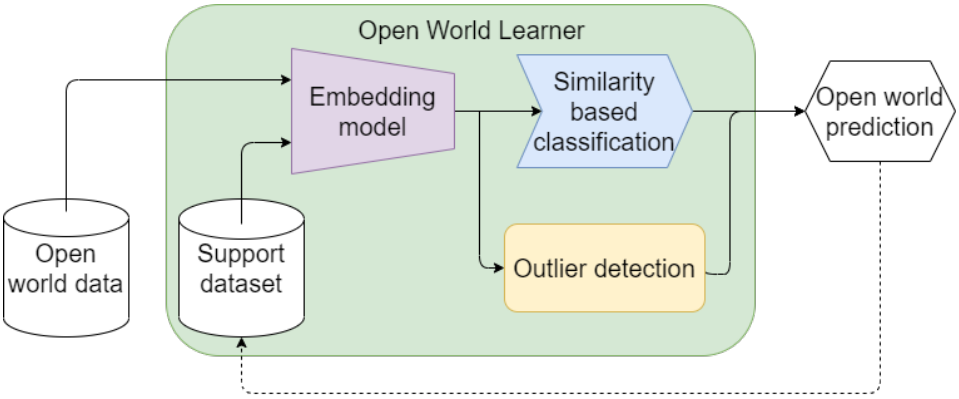


Figure 4.1: Proposed open world learner generic architecture

world problem. There are 2 different algorithm types that are able to do this adaptation: outlier detectors and open set recognition algorithms. We look into the different model architectures these 2 types of algorithm combinations create, as well as testing out several different outlier detectors and OSR algorithms in framework to find the best possible fit.

The few-shot learners tested are the matching network and prototypical network, explained in section 3.2 and 3.3. The different outlier detectors and open set recognition algorithms are listed in sections 4.1 and 4.2 respectively.

The framework presented in this thesis is targeted towards the plankton domain for use in autonomous underwater vehicles, so all experiments and results are based on planktonic datasets, but no dataset dependent specialization is performed on the framework, so the models presented are also generalizable to other datasets and domains.

4.1 Outlier detection architecture

The first possible architecture for realizing the proposed framework is built up around the few-shot learner who contributes both the pre trained embedding model and the similarity based classification module. The outlier detection module is just an outlier detection algorithm that utilizes all the embeddings produced in the few-shot learner to produce an outlier prediction of the open world data in comparison with the support dataset embeddings. This architecture is shown in figure 4.2.

As we want to identify the algorithm that produces the best performance in combination with the few-shot classifiers, several well known outlier detection algorithms are tested. List 4.1 comprises all the tested outlier detection algorithms.

These outlier detectors were all implemented with the PyOD library, and the algorithm choice came down to the algorithms in that library that could be easily implemented with a limited amount of data. The information detailing the implementation of the concatenation of these algorithms and the few-shot classifier is listed in section 5.1.

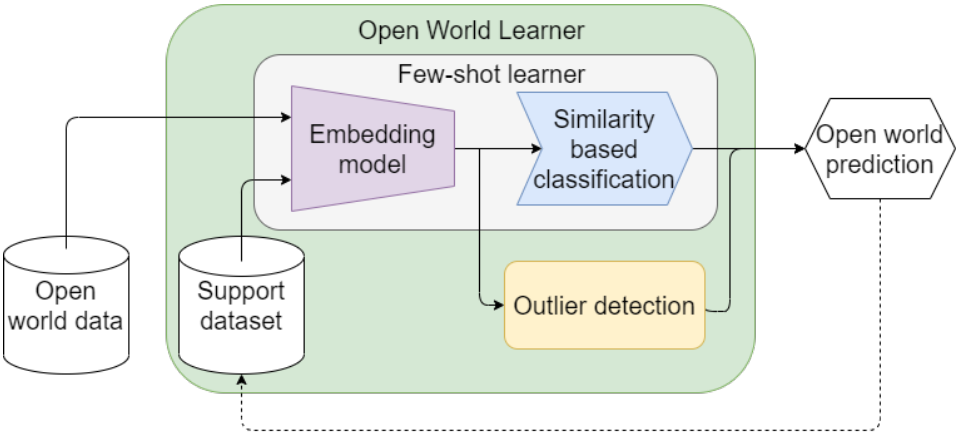


Figure 4.2: Proposed open world learner architecture (few-shot + outlier detector)

4.2 Open set recognition architecture

The second possible architecture for realizing the proposed framework only utilizes the embedding module from the few-shot learner while the open set recognition algorithm functions as both the classification module and the outlier detection module. This architecture is shown in figure 4.3.

The open set recognition algorithms tested with the few-shot classifiers is given in list 4.2.

The information detailing the implementation of the concatenation of these algorithms and the few-shot classifier is listed in sections 5.2, 5.3 and 5.4.

4.3 Summary

There are many potential combinations of algorithms that allow for the realization of the proposed framework, all with different strengths and weaknesses, making them applicable in different situations. As is be shown in the results section (7) and argued for in the discussion section (8), we highlight and advocate for two different variations of the framework: The Prototypical network in combination with the XGBOD outlier detector proved to be the most accurate variation of the framework given a certain number of samples per class are available. It is a complex model so some samples are required to properly build the model. For tasks where speed is important and/or only a very few samples are available from each class, the combination of the Prototypical network and the simple NNO algorithm proved to be one of the most suited variations. Both these framework variations also exhibit a synergy between the outlier detection module and the classification module that results in a very high classification confidence given that the outlier detection module correctly classifies an object as an inlier. The details regarding this and the extra potential use-case in a closed world scenario are discussed in greater details in section 8.4.

Algorithm	Learning method	Type
Angle Based Outlier Detection (ABOD)	Unsupervised	Probabilistic
Average K Nearest Neighbours (AvgKNN)	Unsupervised	Proximity-Based
Connectivity-Based Outlier Factor (COF)	Unsupervised	Proximity-Based
Deviation-based Outlier Detection (LMDD)	Unsupervised	Linear Model
Extreme Boosting Based Outlier Detector (XGBOD)	Supervised	Outlier Ensemble
Feature Bagging	Unsupervised	Outlier Ensembles
Isolation Forest (IForest)	Unsupervised	Outlier Ensembles
K Nearest Neighbours (KNN)	Unsupervised	Proximity-Based
Lightweight On-Line Detector of Anomalies (LODA)	Unsupervised	Outlier Ensembles
Local Outlier Factor (LOF)	Unsupervised	Proximity-Based
Local Correlation Integral (LOCI)	Unsupervised	Proximity-Based
Median K Nearest Neighbours (MedKNN)	Unsupervised	Proximity-Based
One-Class Support Vector Machines (OCSVM)	Unsupervised	Linear Model
Principal Component Analysis (PCA)	Unsupervised	Linear Model
Subspace Outlier Detection (SOD)	Unsupervised	Proximity-Based
Stochastic Outlier Selection (SOS)	Unsupervised	Probabilistic

Table 4.1: List of outlier detection algorithms tested for the framework.

Algorithm
Nearest Non-outlier (NNO)
OpenMax
Deep open classification (DOC)

Table 4.2: List of open set recognition algorithms tested for the framework.

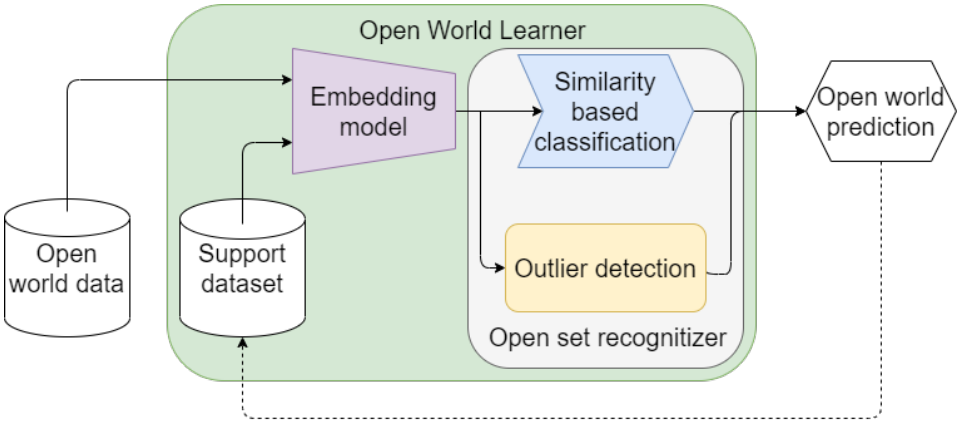


Figure 4.3: Few-shot + open world recognition architecture

Chapter 5

Implementation

This chapter describes how each algorithm combination for the proposed framework was implemented for testing. From the preliminary results covered in section 6.5 it was found that the Prototypical network is the most ideal few-shot algorithm for the framework so this is the base model used for implementation.

This chapter covers the inputs used and the problems encountered, as well as the solutions put in places to reach compatibility between the prototypical network and the open set recognition/outlier detection algorithms. Also mentioned is any other significantly relevant information about the implementation. Section 5.1, 5.2, 5.3 and 5.4 describes the implementation details of the outlier detectors from the pyod library, the nearest non outlier algorithm, the openMax algorithm and the deep open classification algorithm respectively in conjunction with the Prototypical network. Every section begins by listing up the original input and the problems encountered during implementation before listing the adjusted input and solutions to the problems encountered to reach compatibility.

5.1 PyOD - Outlier detection algorithms

The outlier detectors from list 4.1 was all implemented with the PyOD library [44]. This library uses a list of vector embeddings as input and is therefor easily compatible with the embeddings from the prototypical network. Since they all share a common interface they were easily implemented with the prototypical network and tested in parallel.

The unsupervised outlier detectors: Training N_C number of classifiers every episode. Every class k uses only S_k to train the classifier.

The supervised outlier detectors: Training N_C number of classifiers every episode. Where every class k uses S_k as inlier examples and the remaining data in S as outlier examples for training the classifier.

No further modification or adaptations were required to run either the supervised or unsupervised outlier detectors from the PyOD library.

5.2 Nearest non outlier

The Nearest non outlier algorithm utilized the exact same type of embedding produced by the prototypical network embedding module, so only minor modification needed to be done with the algorithm to reach compatibility.

Original input:

- N_S vector embeddings for all N_C classes.

Problems encountered during implementation:

1. The original implementation details of the estimation of τ were poorly written in the paper and simply directed to supplementary material that does not longer exist.
2. The embeddings produced by the prototypical network is a 1600 dimensional vector, so when combined with equation 3.10, the fraction produces near 0 values due to the exponential terms. This slows the optimization process due to shallow gradients.

Adjusted input:

- N_S support set embeddings for all N_C classes.

Solutions to the problems encountered during implementation:

1. Problem 1 from the list above was solved by utilizing the log loss over the function $f_k(x)$ from equation 3.10, and using the Nelder-Mead optimization technique through scipy's optimization library.
2. Problem 2 from the list above was handled by flipping the fraction. This results in extremely large values, but it is easier to deal with for an optimizer than extremely low values due to the steeper gradients.

The original code from the Nearest non outlier algorithm was no longer supported, so the entire algorithm had to be coded from scratch.

Algorithm 7 shows the appropriated version of the nearest non outlier algorithm.

Algorithm 7 Appropriated NNO probability estimation with rejection of unknown or uncertain inputs

Require: Class mean embedding vector: $= [\mu_1, ..., \mu_K]$
Require: Trained class radius vector: $T = [\tau_1, ..., \tau_K]$
Require: Prototypical network query embedding x

$\hat{f}_k(x) = \frac{\pi^{\frac{m}{2}} \tau_k^m}{\Gamma(\frac{m}{2}+1)} (1 - \frac{1}{\tau_k} \left\| W^T x - W^T \mu_i \right\|)$
if $max_k \hat{f}_k(x) \leqslant 0$ **then**
    Reject input
else
    $\hat{y} = argmax_k \hat{f}_k(x)$
end if

5.3 OpenMax activation function

As the openMax function is designed for being appended to the end of a traditional CNN, quite a substantial amount of modifications were needed to reach compatibility with the distance based classification of the prototypical network.

Original input:

- Activation vector: The output values $\mathbf{v}(\mathbf{x})$ of the penultimate layer of convolutional network given the input $\mathbf{X}$. $\mathbf{v}(\mathbf{x})$ is a K dimensional vector with one value for every class.
- Mean activation vectors: Based on the activation vectors from training examples that were correctly classified on the original network with the softmax activation function. Performed on each class individually.
- Weibul fitting training data: Based on distance from the activation vector to the class mean activation vector for training examples that were correctly classified with the original network and the softmax activation function. Performed on each class individually.

Problems encountered during implementation

1. The openMax function natively uses the distance metric from the activation vector to the mean activation vector to produce the weibul score and uses the result of the weibul score to adjust the values in the activation function. This works since the activation function inherently is an estimation that a classification can be based off of. In the prototypical network a query embedding is not inherently such an estimation, so using the embedding alone as the activation vector does not work.
2. The openMax function is designed for maximisation problems only. The openmax algorithm can not be converted to a minimization problem simply by multiplying the input values with (-1) as is possible with the softmax function. The proposed definition of the rejection class $\hat{v}_0(x) = \sum_i v_i(x)(1 - w_i(x))$ would result in a decreased probability of a classifying an object as a new class, rather than an increased probability, when summing over i with negative values, and applying the equation:

$$\hat{P}(y = j|\mathbf{x}) = \frac{e^{\hat{v}_j(x)}}{\sum_{i=0}^{N} e^{\hat{v}_i(x)}}$$

3. The hyper-parameters used to balance the output probabilities of the openmax are specifically tuned to the dataset and network used in the paper, respectively imageNet and alexNet. So when applied on a different architecture and with a different dataset, the predicted probability of the novel classes $\hat{P}(y = 0|x)$ produces hugely different values compared to the original algorithm, changing dramatically based on small changes in some hyper-parameters. Scaling the possible values in order to fit the algorithm to the equation: $y* = argmax_j P(y = j|x)$ for a new task is a unnecessarily complicated process.

4. The α values used in algorithm (5) also poses a problem. The values of the penultimate layer in the alexNet classifier and from the weibul fitting produced by the imageNet dataset are very different from the values of the euclidean squared distance in the prototypical network and weibul fittings produced in a few shot setting. So when $\frac{\alpha-i}{\alpha}$ from line 3 in algorithm (5) is multiplied with the weibul fitting from the proposed framework, it tends to result in the reversal of the probability ranking of the "top α" classes rather than a simple adjustment in the probabilities. This α value is originally used as a relevance weighting of the weibul fitting, since classes that have a low probability of being the correct class should not contribute a lot to the probability of the query being a novel class, but due to the drastic change in use-cases, their purpose is defeated.

Adjusted input:

- Activation vector: The distance from the query embedding $f_\phi(\mathbf{x}_i)$ to the prototypes $\mu_{k'}$. This is a N_C dimensional vector.
- Mean activation vectors: Created from the class support sets, this is equivalent to the prototype of that class. Performed on each class individually.
- Weibul fitting training data: Based on the support set that would be correctly correctly classified if every support sample were tested against the prototypes. This is done for each class individually.

Solutions to the problems encountered during implementation:

1. Problem 1. from the list above was handled by using the distance from the query to the prototype as a substitute for the activation vector of the native openmax algorithm.
2. Problem 2. from the list above was handled by dividing the distance from the query embedding to the prototype by the mean value of the distance from support vectors to their prototypes. Then flipping the problem from a minimization problem to a maximization problem by multiplying all values with a Gaussian function centered around 0 before passing the result of this as the input to the openMax function.
3. Problem 3. from the list above was handled by using a threshold variable on the novel class "probability" $\hat{P}(y = 0|x)$ to determine if the sample belongs to a new class, instead of a direct value comparison with the other class probabilities. This threshold is attained by noting the values of the novel class probability when performing tests on novel cases and comparing it to tests on non-novel classes. This can be optimized with an optimization function, but is omitted as the focus is a proof of concept at this stage.
4. Problem 4. from the list above was handled by simply omitting the use of the α value from the algorithm. Given the low amount of classes ($k - way$) usually considered in few-shot episodes, this is likely to have less impact, compared to the amount of classes in the imageNet dataset.

Algorithm 8 Appropriated OpenMax probability estimation with rejection of unknown or uncertain inputs

Require: Distance from the query to the prototypes output of the prototypical network: $\mathbf{d}(\mathbf{x}) = d_1(x), ..., d_K(x)$

Require: Threshold τ for unknown class, and threshold ϵ for known classes

1: Use a Gaussian filter to invert the activation vector values

$$d(x) = a * exp(-\frac{(x-b)^2}{2c^2}) * d(x) \tag{5.1}$$

2: Calculate $\mathbf{means}$ μ_k and libMR models $\rho_k = (\tau_k, \lambda_k, \kappa_k)$

3:

$$\omega_k(x) = 1 - e^{-\left(\frac{x-\tau_k}{\lambda_k}\right)^{\kappa_k}} \tag{5.2}$$

{Perform inverted weibull fitting and subtract 1}

4: Revise activation vector $\hat{d}(x) = \mathbf{d}(\mathbf{x}) \circ \omega(\mathbf{x})$

5: Define $\hat{d}_0(x) = \sum_k d_k(x)(1 - \omega_k(x))$ {$\hat{d}_0(x) =$: novel class activation vector}

6:

$$\hat{P}(y = k|x) = \frac{exp(\hat{\mathbf{d}}_\mathbf{k}(x))}{\sum_k^K exp(\hat{d}_k)} \tag{5.3}$$

7: Let $y^* = \underset{k,k\neq 0}{\operatorname{argmax}} P(y = k|\mathbf{x})$

8: Reject input if $P(y = 0|\mathbf{x}) > \tau$ or $P(y = y^*|\mathbf{x}) < \epsilon$

The resulting algorithm is shown in algorithm (8).

The original code posted by Bendale et al.[1] was used in this project by copying the necessary code and adjusting it to fit the prototypical network as described in the lists above.

5.4 Deep open classification

Similarly to the openMax algorithm, the DOC algorithm is designed to be used instead of a softmax function at the output layer of a standard convolutional network. Given the non-standard output of the embedding module in the framework, some customization had to be performed to reach compatibility.

Original input:

- The output values $\mathbf{v}(\mathbf{x})$ of the penultimate layer of the network given input $\mathbf{x}$. $\mathbf{v}(\mathbf{x})$ is a K dimensional vector with one value for every class.

Problems encountered during implementation:

1. The distances between a query embedding and the prototypes in the prototypical network is poorly compatible with the sigmoid activation function as the sigmoid produces most nuanced output values for input values between -1 and 1 while the distance between an embedding and the prototype can be in the range of $[0, \inf]$.

2. As with the softmax function in the prototypical network, the sigmoid activation function needs to be applied to the negative distance between the query embedding and the prototypes since greater input value leads to greater output value, and greater probability. But due to the distances being solely in the range of $[0, \inf]$, only the left half plane of the sigmoid function is usable $-\inf < -d < 0$.

Adjusted input:

- The distance from the query embedding $f_\phi(\mathbf{x}_i)$ to the prototypes $\mu_{k'}$. This is a N_C dimensional vector.

Solutions to the problems encountered during implementation:

1. Problem 1 from the list above was handled by retraining the prototypical network with the sigmoid functions and the DOC algorithm loss from equation 3.14, naturally training it to produce values in a logical range for the sigmoid activation function.

2. Problem 2 from the list above was handled by interpreting 0.5 as the max probability produced by the sigmoid function, and performing the Gaussian fitting of the class probabilities around 0.5 instead of 1.

[1] OpenMax source code available online at https://github.com/abhijitbendale/OSDN

The original code posted by Shu et al.[2] was used in this project by copying the necessary code and adjusting it to fit the prototypical network as described in the lists above.

The algorithm 9 is the updated evaluation episode algorithm, given the changes required above.

Algorithm 9 Appropriated DOC probability estimation with rejection of unknown or uncertain inputs

Require: Distance from the query to the prototypes output of the prototypical network: $\mathbf{d}(\mathbf{x}) = d_1(x), ..., d_{N_C}(x)$
Require: $\sigma = \sigma_1, ..., \sigma_{N_C}$ Variance from the Gaussian fittings.
1: $P(y = k|\mathbf{x}) = Sigmoid(\mathbf{d}(\mathbf{x}))$
2: Let $s(i) = argsort(P(y = k|\mathbf{x}))$
3: **for** $i = 1, ..., N_C$ **do**
4: **if** $P(y = s(i)|\mathbf{x}) > 0.5 - \sigma_{s(i)}$ **then**
5: $\hat{y} = s(i)$
6: **Break**
7: **end if**
8: **end for**
9: **if** $P(\hat{y}|\mathbf{x}) == None$ **then**
10: Reject input
11: **end if**

5.4.1 Summary

When adjusting the algorithms to be compatible with the design of the prototypical network some of the adaptations required more modification than others. The outlier detectors and the NNO are combined with the Prototypical network without much effort, the OpenMax and the DOC on the other hand are designed to be placed as the final layer in traditional CNN network, and some significant modifications were needed. Especially for the OpenMax which had to be drastically modified in order to be compatible with the Prototypical network, and as a result end up with rather terrible results as seen in sections 7.4.1 and 7.4.2. The DOC seemed also to be affected somewhat negatively by the changes done, given it's notoriety as state-of-the-art in open set recognition, but still produced good results.

[2]DOC: source code available online at https://github.com/leishu02/EMNLP2017_DOC

Chapter 6

Experimental setup

This chapter describes the experiments conducted, the selection and evaluation criteria that are focused on during model evaluation, as well as the datasets used in this research. The selection criteria is listed and described in section 6.1, the experiments are described in section 6.2. The Siamese network, which is used as a baseline is explained in section 6.3, and the details about the implementation of the few-shot base is covered in section 6.4. As some of the preliminary results had an influence on the desicion making in regards to the framework, these preliminary results are covered in section 6.5. Finally the datasets and hardware used to run the experiments are covered in sections 6.6 and 6.7.

6.1 Selection criteria

The following selection criteria are chosen in order to answer the research question given in section 1.3:

1. Ability to classify training classes
2. Ability to classify known classes
3. Ability to reject unknown classes
4. Combined open world performance
5. Speed (online computational requirements)

The following is an explanation of these criteria in the context of this thesis.

Ability to classify training classes: The standard classifier performance metric used in classic image recognition models. A measure of how well the classifier performs in a closed environment, and a good baseline to compare against other state of the art classifiers.

Ability to classify known classes: The ability to generalize to classes not used during training, by just using a few known references of those classes. This is the few-shot recognition problem, which is essential when testing a few-shot classifier. This test is conducted in a closed world setting.

Ability to reject unknown and novel classes: The proficiency of the outlier detection module to reject unknown classes in the similarity space not included in the support set. This means that the outlier detector has to perform well with the similarity embeddings produced by the embedding module.

Combined open world performance: The combined performance of both the outlier detection module and the classification module. This criteria measures how many samples are correctly identified as inliers, and how many of these are given the correct class label by the classifier module.

All the selection criteria above is measured by the accuracy metric detailed in section 2.1.6.

Speed: As speed is a vital metric for several different applications, the question of computation efficiency is also a relevant for a thorough analysis of the different possible framework variations. In addition to this, few-shot algorithms can increase their accuracy by increasing it's reference database and comparing against more instances of every class. Allowing for the possibility of increasing the accuracy at the cost of computational efficiency. This is measured based on the time it takes for one episode to run. The definition of an episode is shown in section 6.4.

6.2 Experiments

In order to investigate all the selection criteria from the list in section 6.1, three types of experiments are performed:

1. Classify training classes with the closed world assumption.
2. Classify known classes with the closed world assumption.
3. Classify known classes and identify unknown classes with the open world assumption.

Classify training classes with the closed world assumption: This test involves training the base few-shot classifiers in the classical method of machine learning (explained in section 2.1.2), with a training set and a testing set containing different samples from the same classes. The test is evaluated on the accuracy of the prediction over the training set in a closed world setting, without the outlier detection module.

Classify known classes with the closed world assumption: This test involves training the base few-shot classifiers in the one/few-shot method(explained in section 2.2), with a background set and evaluation set containing different classes. The test is evaluated on the accuracy of the prediction over the evaluation set, without the outlier detection module.

Classify known classes and identify unknown classes with the open world assumption: This tests involves testing all the possible framework variations with open world data. So the query image for each test can either belong to one of the classes in the support set, or it can be a class not existing in the support set.

There are three results that are of interesting in this experiment:

- **Pure outlier detection module performance:** The accuracy results of the outlier vs inlier classification for each outlier detection module based on the similarity space embeddings of the prototypical network.
- **Combined architecture performance:** The combined accuracy results of both the outlier detection module and the classification accuracy of the classification module. So for a correct prediction in this test, the image either has to be correctly labeled as an outlier or it has to be correctly classified as an inlier and classified as the correct class by the classification module.
- **Speed:** The computational requirements for every possible framework variations. For reference, all results are performed on the NVIDIA RTX 2080 Ti.

The split of known and unknown classes for the open world test test is set to 50/50.

Although under the closed world assumption, few-shot classifiers are usually tested with n-shot values of 1 and 5, in order to give the outlier detection module more ideal conditions the tests are conducted with n-shot values of 5, 20 and 50 for this test.

6.3 Siamese network baseline

The siamese network is often used as the baseline in few-shot papers [11] [12], as it is easily extendable into a few-shot classifier by performing the ”match”/”no match” classification over n-samples from k-classes of the support set and then assigning the prediction to the highest average probability class.

$$\hat{y} = argmax_k P(y = k|x) \tag{6.1}$$

However, most of papers that use the Siamese network as a baseline operate under the closed world assumption and do not include the rejection property of the Siamese network. As this thesis does not adhere to this assumption we also want to utilize this rejection property in a multi-class classification scheme to employ it as an open set recognition algorithm. This rejection can be included by rejecting the query image if the average probability of all k classes are below the trained rejection threshold of the siamese net.

The Siamese net is still always trained in the original one-shot manner, even when tested on open world recognition tasks and a k-way, n-shot setting.

During early research of this thesis, the Siamese network was the main experimental focus, looking into the possibility of further developing a framework based on this core classifier. Even with the disappointing results in section 6.5, there was a valiant effort made to increase the performance of the Siamese network by utilizing more advanced and/or task specific embedding networks in the Siamese model. It further led into the exploration of few-shot learners and the framework presented in section 4. The siamese

net research was kept as a baseline for the performance of the proposed framework, as is often done in few-shot papers, as well as to log all research done for this thesis.

The tests to find more advanced and/or tasks specific embedding networks for the Siamese model was set up like a normal one-shot classification test, and several different networks were pitted up against the base embedding model of the Siamese net. As these networks usually are traditional classifiers themselves, the last fully connected layer of the different network where switched out with the last embedded layer of the native Siamese network embedding module to produce an embedding instead of a classification and to attain compatibility with the Siamese network.

The networks tested as alternative embedding networks are the VGG net [45] and ResNet [46] with their various permutations, the full list is listed in table 6.1. The specific networks were chosen as they had proved to produce good classification results on plankton classification. The results for this test are presented in section 7.1 along with the results of using the Siamese network in a few-shot open world setting (as explained above).

Networks
VGG11
VGG16
VGG19
ResNet18
ResNet34
ResNet50
ResNet101
ResNet152

Table 6.1: Siamese embedding network variations

6.4 Setup - Few-shot learner

For both the Matching network and the Prototypical network, the same setup as in the original papers were used. Both of these models use the same embedding network structure consisting of a simple four block setup. Each block is made up of:

- Convolutional layer
- Bath normalization layer
- Relu activation layer
- MaxPool activation layer

This architecture produces embeddings in a 1600-dimensional output space. These are the similarity space embeddings used for the classification.

Both models were trained over 80 epochs consisting of 1000 training episodes. Where one training episode is defined as testing N_Q query images for all N_C classes over a support set consisting of $(N_C * N_S)$ images.

		5-way Acc.		10-way Acc.	
Model	Dist	1-shot	5-shot	1-shot	5-shot
Siamese Network	-	55.8%	64.8%	44.2%	45.6%
Matching Network	Cosine	72.4%	80.8%	55.6%	59.6%
Prototypical Network	Euclidean squared	68.4%	95.0%	54.5%	87.7%

Table 6.2: Preliminary classification results on the WHOI-plankton dataset, traditional method.

		5-way Acc.		10-way Acc.	
Model	Dist	1-shot	5-shot	1-shot	5-shot
Siamese Network	-	47.0%	51.6%	29.8%	37.0%
Matching Network	Cosine	71.8%	78.6%	55.9%	64.8%
Prototypical Network	Euclidean squared	70.2%	91.8%	53.6%	82.6%

Table 6.3: Preliminary classification results on the Kaggle-plankton dataset, traditional method.

For the training process, a k-shot, n-way configuration has to be chosen for the classifiers to optimize the training for. This doesn't limit the different configurations available during testing, but it slightly affects the performance of other configurations than the one the classifier was trained for. The authors of [12] points out that training the same n-shot and a higher k-way than is used during testing produce better results. The reason for this being that training with a higher k-way seems to cluster the embeddings better.

6.5 Preliminary results

As experiments 1 and 2 from section 6.2 are performed with the few-shot algorithms alone, the following results were available early in the testing. Tables 6.2 and 6.3 lists the performance for different k-way, n-shot test configurations of the base versions of the Siamese network, the Matching network and the Prototypical network when applied to the WHOI-plankton and Kaggle plankton datasets.

All of these results are achieved with the training parameters: k-way $= 10$ and n-shot $= 5$. The training configuration was fixed as to increase the comparability of the models and eliminate the need for fine-tuning each model to optimize each individual score.

For each entry in the result tables 6.5 and 6.4, a separate model was trained with early cutoff based on the best results of the evaluation set. And all models were evaluated based with the k-way, n-shot values as shown in the tables. For instance, for the 5-way, 5-shot result, the prototypical network was evaluated after every epoch in a 5-way, 5-shot scenario, and the model and score from the epoch that produced the best evaluation score was stored.

As can be seen from these results, the accuracy of the Siamese network was less than satisfactory for the proposed framework. This was what originally implored the search for other types of embedding networks in the siamese net model, and the eventual transition into the more proficient few-shot classifiers.

The results also show that the Prototypical network perform markedly better than the

		5-way Acc.		10-way Acc.	
Model	Dist	1-shot	5-shot	1-shot	5-shot
Siamese Network	-	55.4%	62.0%	34.8%	44.0%
Matching Network	Cosine	61.4%	65.0%	43.9%	47.2%
Prototypical Network	Euclidean squared	58.8%	83.6%	43.5%	69.7%

Table 6.4: Preliminary classification results on the WHOI-plankton dataset, few-shot method.

		5-way Acc.		10-way Acc.	
Model	Dist	1-shot	5-shot	1-shot	5-shot
Siamese Network	-	43.0%	50.2%	29.2%	35.8%
Matching Network	Cosine	67.6%	73.6%	50.5%	56.5%
Prototypical Network	Euclidean squared	70.0%	88.4%	54.5%	76.4%

Table 6.5: Preliminary classification results on the Kaggle-plankton dataset, few-shot method.

Matching network, which correspond well with the results noted in [12]. The combination of this discovery in addition to the Prototypical network being a much simpler model led to the conclusion that further investigation into the Matching network be halted and the Prototypical network was decided as the backbone of our framework. As a result, all tests of open world experiment (3. experiment) is performed with the Prototypical network as the base model.

It is worth noting, as can be seen in these results, that increasing the k-way value increases the complexity of the tasks, and therefore decreases the accuracy, for instance, the Prototypical network scores 88.4% in the 5-shot 5-way setting, while it scores 76.4% in the 5-shot 10-way setting. On the other hand, increasing the n-shot value increases the number of reference images per class and consequently increases the accuracy. For instance, the Prototypical network scores 70.0% in the 1-shot 5-way settings, while it scores 88.4% in the 5-shot 5-way settings

6.6 Datasets

All tests were performed over two well-known planktonic datasets: WHOI-Plankton [47] and the plankton dataset from the Kaggle National science bowl [48]. Both datasets are reputable in-situ planktonic datasets that feature images of singular plankton. They are excellent for research tasks based on pure classification of various planktonic sub-species.

There is no data pre-processing performed on any of the datasets.

6.6.1 WHOI-Plankton

The WHOI-Plankton dataset [47] is authored by Sosik et al. and provided by WHOAS: Woods Hole Open Access Server. It consists of 329 936 black and white images of planktonic subspecies divided up into 103 categories. These categories are of a non-uniform size and the smallest category contains one image while the biggest class contains 266 156 images. The images themselves are also of non uniform size.

All classifiers are tested on both the traditional classification problem as well as the few-shot problem, this means that the datasets are split both into a training set and a test set, as well as being split into a background set and an evaluation set. These splits were performed as explained in sections 2.1.2 and 2.2. The initial splits were:

Initial traditional classification problem data-split:

- Training set: 80% of all 103 classes
- Test set: 20% of all 103 classes

Initial few-shot classification problem data-split:

- Background set: 83 classes
- Evaluation set: 20 classes

As the classifiers are trained with up to a 20-shot configuration and tested with up to a 50-shot configuration, all classes in the training and background set with less then 20 samples were removed from the sets. Equally all classes in the test and evaluation set with less than 50 samples were removed.

The whoas dataset also has a class called 'mix' and 'mix elongated', which contains a mix of species not labeled as any specific class. These mix classes would not work well in comparison to the similarity based classification due to the disparity of the data in these classes, therefore these classes are omitted from the dataset.

After all this shedding of classes, the remaining data splits is as follows:

Final traditional classification problem data-split:

- Training set: 80% of 49 classes, 47 873 images
- Test set: 20% of 19 classes, 11 449 images

Final few-shot classification problem data-split:

- Background set: 41 classes, 56 093 images
- Evaluation set: 10 classes, 7 310 images

In order to have a stronger foundation for comparison, all tests are performed on this reduced dataset, no matter how many samples of each category the test actually require.

6.6.2 Kaggle

The Kaggle Plankton dataset [48] is assembled by Oregon State University's Hatfield Marine Science Center, and provided by Kaggle. It consists of 30 459 black and white images of planktonic subspecies divided up into 121 categories. These categories are non-uniformly represented with 9 images in the smallest category and 1 979 images in the biggest class. The pixel-size dimensions of the images also vary.

As with the WHOI-Plankton dataset, the classes falling bellow the minimum required n-shot for for all four different data splits also needed to removed for the Kaggle Plankton dataset. The remaining data splits is as follow:

Final traditional classification problem data-split:

- Training set: 80% of 107 classes, 24 259 images
- Test set: 20% of 36 classes, 4 641 images

Final few-shot classification problem data-split:

- Background set: 72 classes, 22 143 images
- Evaluation set: 30 classes, 7 921 images

6.7 Hardware

All experiments were performed through SSH on a powerful computer appropriated by the AILARON project. The components of this computer are as follows:

- OS: Ubuntu 18.04.2 LTS
- CPU: Intel LGA1151 i9 - 9900K
- GPU: 2x ASUS RTX2080Ti Turbo
- RAM: 64 GB
- SSD: Crucial MX500 2TB
- HDD: Seagate Skyhawk 6TB

Chapter 7

Results

This chapter presents all the results of the research with exception to the results given in section 6.5. These results were covered earlier as they were attained at an early stage in the research and had an influential effect on the decision making for the development of the framework. Section 6.5 covers the results of experiments 1 and 2 (from section 6.2 for the outlier detection architecture as the classification module is unchanged from the base classification module of the Prototypical network. As the open set recognition algorithms provide both the outlier detection module and the classification module of the framework, the results from experiment 1, 2 and 3 (from section 6.2) for the open set recognition framework are all be presented in this chapter.

This chapter starts by reviewing the results from the Siamese network. This algorithm was the main focus of this study early on in the research, but given the discovery that few-shot models are better candidates for the proposed framework, the results are rather kept as a baseline and to log all the research done for this thesis. Section 7.2 presents the few-shot close world results of the Prototypical network model that is used as the backbone in all the final variations of the framework. Then section 7.3 covers the results of the first two experiments (from section 6.2) for the open set recognition architecture. And section 7.4.2 finally covers the results of the third experiment; the performance of the different architectures in the open world setting. Both the pure rejection results as well as the combined framework accuracies.

7.1 Siamese network baseline results

7.1.1 Embedding model swap

In the test to find a better performing embedding model of the Siamese net, we tested out several more advanced networks in addition to the base model by [10]. This test was conducted purely on the ”match”/”not match” based one-shot task described in 2.2. Table 7.1 shows the results of ‘best’ accuracy produced by the respective Siamese network variations. After performing several repetitions of this test, we observed that there was

Sub Networks	Performance Metrics	
	Accuracy '%'	*Speed 'img/sec'*
Default	74.1%	2200
VGG11	64.5%	1183
VGG16	66.1%	707
VGG19	63.5%	591
ResNet18	75.9%	2653
ResNet34	72.7%	1791
ResNet50	68.3%	930
ResNet101	72.9%	590
ResNet152	73.4%	435

Table 7.1: Performance of Siamese embedding module variations

a considerable variance in the results for the sub-networks, even when trained with the 'exact' same setup. Variances from the experiment were found in the range of 10% for some sub-networks. This number drastically reduces the performance and validity of embedding alternative networks in the architecture. The default baseline Siamese network proved to be the most stable, having a range of variance, for different training rounds, of around 2% while still producing decent accuracy scores. Table 7.1 shows that the default baseline Siamese network achieves an accuracy of 74.1% and a relatively high speed of 2200 images per second. All other Siamese network base tests are therefore performed with the base Siamese model without any architectural modifications. It is worth noting that this test is performed on the WHOI-dataset detailed in section 6.6.1.

7.1.2 Siamese open set recognition

The Siamese network open set recognition test was conducted to produce a few-shot, open world baseline for the proposed framework. It is performed by performing one-shot classification over n-samples from k-classes. For a class to be rejected, the average value from all classes had to be deemed 'no match'. This test is conducted with the same setup as explained in experiment 3. in section 6.2, with 50% of the query images belonging to a known class, and 50% of the query images belong to unknown classes. Table 7.2 presents the accuracies of the Siamese network for the pure task of rejection versus inclusion with regards to the classes in the support set. Here, a 50% score is equivalent to the trivial solution of guessing only positive or only negative given our 50/50 divide of the test data.

5-way Acc.		10-way Acc.	
1-shot	5-shot	1-shot	5-shot
53.9%	53.7%	52.2%	51.3%

Table 7.2: Accuracy of the Siamese network on the pure rejection task on the Kaggle plankton dataset.

Table 7.3 shows the confusion matrix resulting from the 5-way, 5-shot task, where true positive indicates an image class correctly classified as known, while the true negative is a new class that is promoted as a class never seen before. Only 57 images representing 5% of the whole dataset were correctly predicted as new classes out of the 500 examples of new classes that were used during testing. Based on these results, we can observe that the network tends to allocate 90% of the images to the known classes.

		Predicted	
		Positive	Negative
Actual	True	480	57
	False	443	20

Table 7.3: Confusion matrix for the Siamese network on the pure rejection task on the Kaggle plankton dataset.

7.2 Prototypical network baseline results

To generate a comparable baseline for the open world results, a single Prototypical network model was trained as the backbone for all open-world tests. This model was trained with a 10-way 5-shot training configuration, saving the model that performed the best on a 5-way 5-shot evaluation task during that training. Table 7.4 shows the the closed world, few-shot results of this model.

As expected, the results show a positive increase in the performance given more training data per class, but seem to saturate at around the 30-shot configuration in a closed world setting, as the accuracy produced by the 50-shot configuration is approximately equal. The reason for not performing a 10-way, 50-shot test is due to hardware limitations, as the graphical memory required for this test surpasses the available GPU memory on the computer used to run these experiments.

7.3 Closed world results

The closed world results comprises the results from experiment 1 and 2, detailed in section 6.2. Here the performance of all tested variations of the proposed framework are tested with the rejection capability turned off. When testing the outlier detection architecture, the classification module remains unchanged, so the results presented in 6.5 are valid as complete results to experiments 1 and 2 for the outlier detectors. Tables 7.5 and 7.6 shows the results of these two experiments for the open set recognition architecture.

We see that the closed world results of the NNO algorithm are very similar to the ones produced as a baseline for the Prototypical network in table 7.4. This is because both classification schemes are based on the distance from the class center (the prototype). This means that the classification module is practically unchanged for the NNO variation, making it act more like an outlier detection algorithm even though it is defined as an open set recognition algorithm.

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
Prototypical network	84.7%	88.6%	90.7%	90.2%	74.9%	81.4%	81.8%

Table 7.4: Prototypical network closed world model results on the kaggle dataset

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
NNO	87.9%	91.2%	92.4%	93.3%	78.0%	85.4%	84.1%
OpenMax	54.9%	84.8%	89.1%	88.6%	42.0%	75.8%	80.2%
DOC	82.3%	85.8%	85.3%	85.8%	71.5%	75.5%	75.6%

Table 7.5: Classification results of the open set recognition architecture on training classes with closed world assumption over the Kaggle dataset

In the closed world setting, the openMax algorithm still uses the weibul fitting to modify the predictions based on how well they fit with the model created, this seems to incur a small performance loss compared to the base Prototypical classification module. From the results, we also observe that the openMax classification is sensitive to the various n-shot values, performing poorly for lower values, this is again due to the weibul fitting, which requires a certain amount of data to be properly trained.

Due to the unique loss function utilized by the DOC algorithm, it is the only tested variation of the framework that requires it's own custom embedding model. Despite the differing training scheme of the DOC algorithm, the framework variation still performs well with only a small drop in performance compared to the prototypical baseline of section 7.2.

7.4 Open world results

The open world results details the outcome of experiment 3 from section 6.2. This tests is performed with the rejection capabilities of the tested variations of the proposed framework turned on.

7.4.1 Pure rejection results

The results of the pure rejection task for the tested variations of the framework is presented in tables 7.7 and 7.8. The highlighted numbers are the top performers of each configuration.

In table 7.7, LOCI, OCSVM and XGBOD all mark themselves out to be the stronger contenders for higher n-shot values. Which is indicative to their computationally complex design, which makes them more likely to require a larger amount of data to perform well. OCSMV falls off slightly in the 50-shot configuration, the reason for this could be attributed to the algorithm overfitting the class data. Regardless of the reason, it does however signal that it is more sensitive to varying n-values compared to the other two

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
NNO	84.7%	85.8%	87.8%	91.2%	72.7%	80.2%	84.2%
OpenMax	53.4%	80.5%	85.7%	87.2%	41.4%	74.3%	77.0%
DOC	81.5%	83.4%	85.4%	85.0%	69.2%	72.1%	73.8%

Table 7.6: Classification results of the open set recognition architecture on known classes with closed world assumption over the Kaggle dataset

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
ABOD	68.8%	64.5%	61.2%	60.2%	68.2%	63.8%	61.4%
AvgKNN	71.1%	72.3%	69.3%	68.2%	73.8%	71.3%	69.5%
COF	73.7%	65.8%	61.4%	60.4%	71.8%	65.6%	62.0%
LMDD	53.0%	52.5%	52.3%	51.8%	52.0%	53.2%	52.3%
XGBOD	62.0%	77.4%	**77.8%**	**80.4%**	56.7%	69.4%	71.0%
Feature Bagging	**75.5%**	75.0%	69.6%	68.8%	73.0%	74.0%	70.2%
IForest	58.5%	57.5%	57.4%	57.0%	56.6%	57.4%	57.7%
KNN	75.3%	72.2%	69.6%	68.6%	**74.5%**	71.5%	70.2%
LODA	45.6%	49.4%	52.0%	52.2%	45.9%	50.4%	51.4%
LOF	75.3%	72.7%	69.5%	68.3%	72.9%	71.1%	70.2%
LOCI	50.0%	76.3%	76.7%	78.0%	50.0%	74.8%	**77.0%**
MedKNN	71.1%	72.0%	69.3%	68.0%	73.8%	70.8%	69.5%
OCSVM	51.8%	**79.0%**	76.5%	73.0%	52.2%	**76.8%**	76.6%
PCA	50.0%	67.8%	69.2%	71.0%	50.1%	67.9%	69.0%
SOD	72.2%	68.7%	66.2%	64.7%	71.8%	67.8%	65.6%
SOS	71.3%	71.2%	67.5%	67.8%	69.2%	70.7%	68.3%

Table 7.7: Accuracy of the outlier detection architecture on the pure rejection task on the Kaggle plankton dataset.

approaches.

The very simple KNN based algorithms (KNN, AvgKNN, MedKNN) are also all around strong performers producing especially good values at low shot setting, but they do fall off a bit at the higher n-shot values, this can be related to the fine tuning of the classifiers, as these models needs to be tuned in regards to the amount of neighbours included in the prediction. For these result attained in this experiment, the number of neighbours included is 1/3 of the current n-value.

In table 7.8 the results of the open set recognition architecture variations are listed, these algorithms produced some interesting results. The best scoring model among all the open set recognition variations for this task is also the simplest, the NNO algorithm which is comparable in performance with some of the better performing outlier detection architecture variations. What is most impressive about this model, is it's robustness to change in n-shot value, producing nearly uniform results for all values of n.

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
NNO	72.6%	71.6%	71.2%	75.6%	66.2%	66.9%	65.2%
OpenMax	50.3%	52.7%	53.6%	54.9%	49.9%	51.7%	50.2%
DOC	60.5%	72.6%	71.6%	70.1%	59.9%	66.0%	67.3%

Table 7.8: Accuracy of the open set recognition architecture on the pure rejection task on the Kaggle plankton dataset.

The DOC also produces decent results for higher n-shot values, but also showing it's dependency of a minimum requirement of data to estimate good Gaussian fittings that are used for the rejection. However, for every other test than the 5-shot configurations, it performs on par with the NNO algorithm.

Probably the most disappointing results on the pure rejection task is those of the OpenMax algorithm which barely outperforms a random guess for each sample. The reason for this is twofold: First is that it appears to be highly dependent on several hyperparameters, making it difficult to tune for every test configuration. The other reason is the amount of customization performed to reach compatibility with the prototypical network (detailed in section 5.3 is very significant, to the degree that it is almost a new algorithm, probably weakening it's performance compared to the original paper.

Note that when using accuracy as the performance metric, given that there is a 50/50 split in outliers and inliers as used on the test of pure rejection experiment, an accuracy of 50% is considered the trivial score as guessing that all input is either only outliers or only inliers produces this accuracy.

Only the Kaggle dataset is used for this test as the modified WHOI-Plankton dataset does not hold enough classes in the evaluation set to account for a 10-way classification in addition to the open world classes.

7.4.2 Combined architecture results

The combined performance of all the different variations of the two different architectures in tandem the Prototypical network classification is shown in table 7.9 and 7.10. These results show the ultimate performance scores for all the tested variations of the proposed framework.

As shown in 7.9, the top scoring outlier detector architecture, in most of the tested configurations, proved to be the XGBOD algorithm in combination with the Prototypical network. Achieving 79.4% score on the 5-way, 50-shot configuration which is only a 10% loss compared to the closed world setting for the Prototypical network baseline.

Something noteworthy about these results is that there is less than a 1% performance loss on the combined results compared to the pure rejection results for the XGBOD framework variation. If we compare this to the LOCI variation that was the strongest competitor to the XGBOD variation in the pure rejection task, we see a bigger performance loss on combined framework, resulting in the LOCI algorithm loosing out to XGBOD in terms of accuracy. A similar relative performance loss can be seen in almost all of the tested variations of the outlier detection architecture.

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
ABOD	47.3%	60.4%	57.1%	55.1%	59.0%	56.6%	53.1%
AvgKNN	**71.1%**	68.0%	65.3%	63.1%	65.5%	64.4%	61.1%
COF	68.6%	61.5%	57.2%	55.1%	63.4%	58.7%	54.0%
LMDD	47.3%	48.0%	48.0%	46.5%	43.6%	45.9%	43.7%
XGBOD	62.0%	**77.0%**	**77.2%**	**79.4%**	56.4%	68.8%	**70.4%**
Feature Bagging	70.2%	71.0%	65.6%	63.6%	64.3%	67.3%	61.7%
IForest	52.1%	53.1%	53.0%	51.7%	46.8%	63.7%	49.1%
KNN	70.4%	68.2%	65.6%	63.6%	**66.1%**	64.6%	61.8%
LODA	38.8%	44.9%	47.7%	46.9%	35.6%	42.9%	42.7%
LOF	70.4%	68.4%	65.4%	63.1%	64.6%	64.3%	61.7%
LOCI	42.6%	72.6%	73.2%	73.4%	38.6%	68.6%	69.7%
MedKNN	**71.1%**	67.8%	65.4%	63.0%	65.5%	64.0%	61.2%
OCSVM	51.8%	76.0%	73.0%	68.1%	52.0%	**71.4%**	69.2%
PCA	50.0%	66.3%	67.1%	68.1%	50.0%	64.9%	64.6%
SOD	66.5%	64.1%	59.5%	59.8%	61.7%	60.2%	58.4%
SOS	69.5%	67.0%	63.4%	62.6%	65.7%	63.7%	59.8%

Table 7.9: Combined outlier detection architecture accuracy (outlier detector acc + few shot acc)

	5-way Acc.				10-way Acc.		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
NNO	71.0%	70.8%	69.6%	74.8%	64.3%	65.4%	64.5%
OpenMax	36.1%	51.5%	51.8%	52.9%	49.9%	50.5%	49.2%
DOC	58.5%	70.2%	67.7%	67.9%	55.8%	59.8%	61.2%

Table 7.10: Combined open set recognition architecture accuracy (outlier detector acc + classification acc)

For the combined performance with the open set recognition architecture shown in table 7.10, we see that the NNO algorithm actually is one of the top performers across the board for the 5-shot configurations. It also continues to produces good results that are independent of the n-shot values. Similar to the XGBOD results, the NNO performance loss from the pure rejection task to the combined results were $< 1\%$, this is more easily explainable for the NNO algorithm as it effectively learns a radial cutoff point out from the prototype, and then classifies the query as the closest prototype in the same manner as the base Prototypical network does, and by doing this removes most of the samples that would be difficult for the classifier to label. The NNO clearly shows it's advantage by being a simple model that is easily optimized and therefore performs very well on low amounts of data, unfortunately it looses out to the XGBOD on performance for higher n-shot values.

	5-way		10-way	
Algorithm	5-shot	30-shot	5-shot	30-shot
ABOD	9.9	6.5	15.2	6.4
AvgKNN	185.1	24.7	87.7	13.3
COF	267.2	17.7	145.7	9.4
LMDD	5.3	0.4	3.3	0.2
XGBOD	0.3	0.1	0.2	0.1
Feature Bagging	22.9	13.0	12.6	6.8
IForest	2.7	2.2	1.6	1.3
KNN	200.3	24.2	88.6	13.9
LODA	27.0	25.0	22.3	12.3
LOF	250.9	42.9	125.8	22.8
LOCI	4.2	1.0	5.7	0.6
MedKNN	134.8	19.8	65.1	10.4
OCSVM	435.3	81.5	227.1	42.3
PCA	355.3	81.5	215.6	42.4
SOD	2.0	1.6	3.2	1.9
SOS	4.6	8.3	4.3	5.9

Table 7.11: Speed of the outlier detection architecture variations [img/s]

7.4.3 Computational speed

Table 7.11 and 7.12 shows the computational speed of all the possible variations of the framework. This score is given in episodes per second where k number of models based on n number of samples are trained, and five support images are classified per episode. The speeds are measured over the outlier detection modules for all algorithms as this is the most time consuming part of all the classifications, and the rejection strategy is what most significantly sets the different algorithms apart.

From table 7.11 it is clear that the accuracy of the top performing algorithms comes at a cost, and algorithms such as XGBOD and LOCI are some of the slowest of the ones tested, with the XGBOD needing 10 seconds to compute one 10-way 30-shot episode. Worth noting is also models like the KNN algorithms and the OCSVM that despite their accuracy still has good computational efficiency.

The DOC algorithm in table 7.12 proves to be the fastest of all algorithms tested for most testing configurations. This is most likely due to the relatively simple outlier detection module of the DOC framework variation, only requiring to fit a simple Gaussian fitting for each of the class probabilities. An extra point to note is that the possible computational speed of the NNO algorithm is most likely much higher than what is given here since the entire algorithm was coded from scratch without concern for computational efficiency.

	5-way.		10-way.	
Algorithm	5-shot	30-shot	5-shot	30-shot
NNO	111.0	4.2	32.8	1.1
OpenMax	240.8	35.0	30.3	22.8
DOC	631.6	163.8	115.9	91.3

Table 7.12: Speed of the open set recognition architecture variations [img/s]

Chapter 8

Discussion

This chapter discusses the proposed framework based on the results presented in chapter 7 and the preliminary results of section 6.5. The discussion is mainly targeted towards the reasoning behind the choice of algorithms that make up the proposed framework, and to answer the research questions posed in section 1.3. Also considered are some general thoughts about the experimentation as well as the results of the one-shot learner the Siamese network and why it failed for our use-case.

8.1 General discussion

Accuracy vs F1-score as performance metric for open set recognition problems:
In this paper, we use accuracy as the prevalent performance metric, but this can be seen as a controversial choice as several papers [36][37] argue that for open set recognition problems, the F1-score should be used instead of accuracy since it is independent of true negative samples. They argue that the model should not be affected by scenarios where there are a lot of outliers as the model can achieve a good accuracy by just rejecting everything. Aka. increasing the accuracy by inflating the true negative. But by ignoring the true negative predictions when measuring performance, risk averse models are rejected, rather favoring models that are too inclusive in their classification. This problem can however quite simply be solved by ensuring balance between the known samples and the unknown samples in the training data, as we have done in this thesis; splitting it up into 50% known samples and 50% unknown samples. We have still included the F1-score results in the appendix as a reference in comparison to earlier papers.

In fact all the confusion matrix results for all the open world classifications are available online at `https://github.com/AndreasLTeigen/few_shot_open_world/tree/master/Results`.

Problem with comparison to traditional classifiers:
Establishing a reference by comparing performance between few-shot learners and traditional classification algorithms would be ideal, however such a direct comparison is not

feasible. They are different tools for different jobs. While the traditional classifiers excel at classification on large datasets with a lot of classes and a lot of samples per class. few-shot learners excel at classification tasks over small datasets with a limited amount of samples as well as a limited amount of classes and even to generalize this to classes that wasn't used during training. Making a comparison would therefore be more indicative of the test on which they were compared rather than a comparison between the algorithms.

Parameter tuning:
In this work, there as not been a focus on parameter tuning. There is reason to believe that several of the tested models could produce better results given meticulous tuning of all parameters. The problem that arises with this is that the models would need a different tuning for each of the tested k-way, n-shot configurations, which is not desirable as the few-shot algorithms are designed to be flexible in this regard and we want maintain this flexible in a framework that is not highly dependent on tuning.

8.2 Siamese network

The results from section 7.1 show that the test to improve the performance of the Siamese network for the plankton dataset by swapping out the embedding networks proved to be fruitless. Even though the advanced networks could achieve results that outperformed the default embedding module of the Siamese net, there proved to be a huge variance in these results, even as much as 10% for some of the networks. The reasoning for this large variance is thought to be due to variational degrees of overfitting, resulting in a performance degradation that depends on the images presented in the training process.It is worth noting that the base model is the best overall performer in the Siamese net given the relatively high accuracy as well as the reliable performance score it delivers in the one-shot setting.

When testing the Siamese network in a few-shot classification setting it the results in tables 6.2, 6.3, 6.5 and 6.4, it shows a clear drawback of adapting the Siamese network to such a scenario. Resulting in a significant accuracy degradation. One of the reasons for this is that the Siamese network is designed for 'match'/'not match' classification, which is inherently different from a few-shot setting with is a multi-class problem, so the Siamese network has a natural disadvantage compared to the few-shot algorithms that were designed for that task. Also the pure rejection task results in table 7.2 and 7.3 show that it rarely rejects objects as outliers and consequently produces accuracy scores that are barely better than the trivial solution. As mention in section 7.1, the trivial solution is equal to 50% since there is a 50/50 split between inliers and outliers.

Accordingly we came to the conclusion that the Siamese network was not a good base to build our desired framework around, which was the reason it was discarded and replaced with the few-shot detectors.

8.3 Few-shot baseline

The few-shot algorithms' ability to perform this generalization can easily be seen in the small difference in accuracy between tables 6.2, 6.3 and 6.4, 6.5. Among the few-shot learners tested, especially the Prototypical network shines in this regard, loosing only a few percentage points when classifying classes only ever seen in the support set (known classes) compared to when classifying classes used during training.

The selection of the few-shot learner used as a base in proposed framework mostly came down to the accuracy and the models ability to generalize to new classes. This is of vital importance since the goal is to design an open world learning algorithm. It was on this ground that the Prototypical network was chosen as the base algorithm. The simpler design of the Prototypical network compared to the Matching network came as a bonus.

Even though the Matching network was discarded on the basis of accuracy on the closed world data. It should be mentioned that there might still be a chance that it could have worked better in tandem with the outlier detectors and the open set prediction algorithms, despite the base performance is worse than the Prototypical network.

8.4 Framework decision

Given that the base few-shot model was chosen as the Prototypical network in the preliminary results section 6.5 and argued for in section 8.3, what remains is to discover what outlier detector or open set recognition algorithm that produces the best open world learning algorithm.

After the experimentation of all the different framework variations, the XGBOD outlier detection algorithm presents itself as an excellent match with the Prototypical network for adapting it to open world data. The results shows that this combination yielded the best performance out of all the tested framework variations, reaching the highest accuracy on the combined problem of rejection and classification for most k-way, n-shot configurations.

So how well does our proposed framework answer the research questions posed in section 1.3? We answer these questions from the vantage point of the XGBOD variation since this is the model that produces the best accuracy scores.

How well does one/few-shot algorithms classify the classes used during the training process? The tables 6.2, 6.3, 7.5 show that given enough class images in the support set, it achieves a good accuracy on the traditional classification problem over the Kaggle dataset for the tested k-way configurations. Although it was mentioned in section 8.3 that a direct comparison between the framework and traditional classifiers doesn't tell us much, what it can do is give a pointer to the general level of difficulty of the dataset. The top scoring classifier in the Kaggle competition featuring the Kaggle-plankton dataset produced an accuracy of 81.52%, proving that it is a rather demanding dataset as the models presented at Kaggle competitions tends to be highly specialized for the problem, utilizing a lot of data pre-processing and optimization. This is also a good example of the comparison difficulty between traditional classifiers and few-shot based framework. When testing

with the 5-way 5-shot configuration, the achieved accuracy of 91.8% in 6.3 far superseded the top scoring results of the Kaggle competition. However if we were to test for all 121 classes, there would be a heavy degradation of the performance of the few-shot learners, even when testing for only 10 classes the performance is only on par with the top scoring algorithm that tested for all 121 classes.

How well does one/few -shot algorithms classify classes stored in it's reference database, but is not used during training? The few-shot algorithms ability to generalize to unseen data is best shown in this test. Tables 6.4, 6.5 and 7.6 shows that the accuracy retention compared to tables 6.2, 6.3, 7.5 is very impressive. Especially in the case of the Prototypical network based implementations, usually resulting in only a couple of percentage point accuracy loss. This means that the proposed open world learning framework proposed in this thesis, can in fact bypass the incremental learning problem mentioned in sections 1.1 and 4, and instead of retraining the model to adapt to new classes, they can simply be included in the framework support set, utilizing the generalizational ability of the framework.

How well does our proposed framework of one/few -shot learners combined with outlier detection/open set algorithms reject unknown/novel classes? Tables 7.7 and 7.8 show that many variations of the framework achieve a good accuracy on the pure rejection task. What is really interesting is what can be seen by looking at the tables listed above in conjunction with tables 7.4, 7.9 and 7.10. Firstly we see that compared to the closed world baseline in 7.4, there is a noticeable performance hit incurred by the rejection task performed by the outlier rejection modules when used in an open world setting, about 10% or so for the XGBOD outlier detection algorithm. However, when we compare the rejection score to the combined accuracy scores, we see that for some models like the XGBOD and the NNO, the loss in accuracy between these two tests are less than 1% for most configurations and no greater than 2% loss for any configuration. This means that these outlier detection modules can preemptively reject samples that lead to low confidence classifications!

Unfortunately, a small amount of the errors made by the outlier detectors in the open word setting are false positives, and when passed on is confidently incorrectly labeled by the classification module. It can however be leveraged in a closed world setting since there are no false positives to reject all samples that are difficult to predict and increase the confidence of classifications. A few sample tests were performed to explore this use-case, and they confirmed the validity of the claim. For instance the 5-way, 50-shot configuration resulted in 72.6% accuracy on the pure rejection task and 71.1% combined framework accuracy. Showing that after the outlier detector, the framework performs classifications with a confidence of 98%.

What is the most suitable algorithm in regard to the AILARON project? As mentioned, the AILARON project employs an AUV that takes images and analyses them in real time, putting requirements on both speed and accuracy. From tables 7.11 and 7.12. We see that the top scoring XGBOD variation of the framework is on the slower side, requiring a few seconds for each episode calculated. There are two different alternatives to improve the suitability for real time image processing. The first one is to rather use the NNO variation of the framework, this variation sacrifices some accuracy at the trad-off

of being a much faster algorithm, and still retains the low accuracy loss between the pure rejection task and the combined framework accuracy. This configuration also excels at situations with very limited amount of support samples per class as shown in 7.10. The other option is to pre-process all prototypes and outlier models for all classes with all available samples in advance. This both increases the accuracy and reduces the on-line computational time.

Chapter 9

Conclusion

The goal of this research has been to explore the viability of giving one-shot and few-shot learners a rejection ability in order to adapt them to the open world setting, utilizing the generalizability of these algorithms to bypass the problem of incremental learning. Several different one-shot and few-shot algorithms have been tested in combination with outlier detectors and open set recognition algorithms to discover the combination that produced the best performance. Although few-shot learners have received considerable attention in the last few years, nobody has previously explored the use of them in an open-world setting, to the best of our knowledge.

This work presents a novel framework for an open world learner algorithm that in addition to performing normal classification, can also identify unknown classes and learn new classes by only a few samples of labeled data. From the extensive tests performed, the top scoring algorithm combination of the proposed framework is shown to be the few-shot learner; Prototypical network and the outlier detector XGBOD (Extreme Gradient Boosted Outlier Detector). Out of all the tested variations, this was the combination that produced the highest total accuracy for the majority of configurations, reaching an accuracy of 79.4% for the Kaggle-plankton dataset in an open world setting when tested on data that was not part of the training set. We also discovered a second use case for this framework; because of the low performance loss between the rejection accuracy and the classification accuracy, the rejection capability can be utilized to preemptively reject samples that lead to classifications of low confidence, resulting in an accuracy classification confidence of 98% in a closed world setting.

We also highlight a variation of our framework consisting of the Prototypical network and NNO (Nearest Non Outlier) for time sensitive tasks where there is very limited data per class. This framework variation also maintains the closed world, high confidence classification feature of the proposed framework.

There was an investigation into the Siamese network to see if the performance of the model could be improved by changing the embedding module of the network to more advanced and specialized neural network. This investigation led to the discovery that these more advanced models caused a huge variance in the results on different runs. The

native embedding network of the Siamese network proved to produce the most reliable in this regard while still achieving one of the highest accuracy scores.

Chapter 10

Future work

During the work on this thesis there were several related directions of research that presented themselves. This chapter covers some of these improvements and extensions that were not included in the scope of this thesis:

Fully autonomize the proposed open world learner framework. In the proposed framework, new classes can be added to the classifier by a human sorting the rejected data into classes and including them in the support set. This process can be automated by using a clustering algorithm to predict new classes by identifying embedding clusters in the similarity space out of the rejected data. This would drastically reduce the need for human labeling effort, and almost fully autonomize the system.

Inter-class variations. As the classification strategy of both the one-shot and the few-shot algorithms covered in this thesis are based on similarities, it should be possible to identify degrees of inter-class variations depending on the class prototype and the variance of the support set. This can be useful in domains such as plankton classification, where there are often only small variations between species, as well as plenty of inter-species variations.

In-situ configuration testing. As mentioned in section 8.4, both the computational speed and accuracy can be increased by pre-processing all models for all available classes using all available data samples. By doing this, new outlier detection models wouldn't need to be computed every episode, and the computational time of the on-line calculations would be drastically reduced for all models. Since all samples could be utilized without any penalty, the accuracy of the framework would also improve, although most likely only marginally as the performance starts to saturate already for the tested configurations in section 7. Using all available samples does however diverge far away from the concept of few-shot and is therefor not included in this thesis.

Bibliography

Bibliography

[1] P Norvig and S Rassell. Artificial intelligence: A modern approach, edition: 3rd, 2010.

[2] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[3] Quoc V Le, Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *Intnl. Conference on Machine Learning*, pages 265–272, 2011.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

[6] Robert S Scalero and Nazif Tepedelenlioglu. A fast new algorithm for training feedforward neural networks. *IEEE Transactions on Signal Processing*, 40(1):202–210, 1992.

[7] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology press, 1995.

[8] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[9] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[10] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[11] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[12] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[14] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi. *Lifelong Machine Learning: Second Edition*. 2018.

[15] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[16] Geli Fei and Bing Liu. Breaking the closed world assumption in text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 506–514, 2016.

[17] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452, 2008.

[18] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

[19] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 535–548. Springer, 2002.

[20] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. A linear method for deviation detection in large databases. In *KDD*, volume 1141, pages 972–981, 1996.

[21] Yue Zhao and Maciej K Hryniewicki. Xgbod: improving supervised outlier detection with unsupervised representation learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[22] Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 157–166, 2005.

[23] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[24] Tomáš Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

[25] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[26] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*, pages 315–326. IEEE, 2003.

[27] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[28] Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015.

[29] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 831–838. Springer, 2009.

[30] JHM Janssens, Ferenc Huszár, EO Postma, and HJ van den Herik. Stochastic outlier selection. *tech. rep.*, 2012.

[31] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[32] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[33] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[34] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[35] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[36] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[37] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[38] Walter J Scheirer, Anderson Rocha, Ross J Micheals, and Terrance E Boult. Meta-recognition: The theory and practice of recognition score analysis. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1689–1695, 2011.

[39] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[40] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324, 2014.

[41] Lei Shu, Hu Xu, and Bing Liu. Doc: Deep open classification of text documents. *arXiv preprint arXiv:1709.08716*, 2017.

[42] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[43] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.

[44] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.

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

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

[47] Eric C Orenstein, Oscar Beijbom, Emily E Peacock, and Heidi M Sosik. Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification. *arXiv preprint arXiv:1510.00745*, 2015.

[48] Kaggle, 2015. `https://www.kaggle.com/c/datasciencebowl/data`, [Accessed 30-March-2020].

Appendix

Leveraging Similarity Metrics to In-Situ Discover Planktonic Interspecies Variations or Mutations

Andreas Langeland Teigen
Dept. of Engineering Cybernetics
NTNU
Trondheim, Norway
andrealt@stud.ntnu.no

Aya Saad
Dept. of Engineering Cybernetics
NTNU
Trondheim, Norway
aya.saad@ntnu.no

Annette Stahl
Dept. of Engineering Cybernetics
NTNU
Trondheim, Norway
annette.stahl@ntnu.no

***Abstract*—Planktons are vital to our planet's ecosystems. Mapping and monitoring planktonic-distribution in the ocean is of great importance in understanding these ecosystems as well as gaining insights on the general health of our planet. The task of identifying different planktonic species and their concentrations is a critical yet challenging part of this endeavor. In this paper, we explore the utilization of *one-shot* classifiers on in-situ captured planktonic images. The similarity-based classification method used by the *one-shot* classification algorithms gives rise to various research opportunities outside of the typical classification paradigm. Exploring the development of a proficient plankton image classification framework that can determine how different or how similar two plankton specimens are, can inform us if they belong to the same species, if they are different variations of the same species or if we have encountered a completely unseen species of plankton. This similarity is valuable information that assists in further mapping the diversity of the planktonic species. We further extend the Siamese *one-shot* classifier with *few-shot* classifier to improve the model performance. Empirical evaluations of the new extension yield promising results.**

***Index Terms*—one-shot, few-shot, Siamese Network, plankton-taxa classification, in-situ planktonic image analysis**

I. Introduction

The planktonic biomass makes up much of the fundamental marine trophic groups, as well as producing an estimated 50% of the world's oxygen supply. Mapping and monitoring the state and distribution of planktonic organisms with respect to both location and intensity is an essential step in understanding both the oceanic and terrestrial ecological structure due to the microbial plankton's vast influence over bionomical factors and global climate. Traditional approaches in this study utilize ships for point data collections [1] [2], data that later have to be meticulously categorized and classified. The deployment of autonomous underwater vehicles (AUV), rigged with advanced real-time sensory systems and powerful processors, allows for the utilization of in-situ identification and classification of microbial biology imaging samples [3] [4] [5] [6].

Since the success of AlexNet [7] [8] in 2012, the study of deep convolutional neural networks (DNN) has skyrocketed, Deep Learning (DL) image classification technologies even surpassed human ability in many image recognition tasks. Today's state-of-the-art DL classification networks [9] [10] [11] [12] [13] achieve a high accuracy on the task of plankton identification and classification on labeled datasets [14] [15] [16]. An emerging problem with these traditionally structured convolutional neural networks (CNN) is their shortcomings with regards to the vast plethora of planktonic species and the countless interspecies variations and mutations, making in-situ identification and classification a strenuous task for conventional models.

To mitigate this issue, we propose a framework for a novel implementation of the Siamese network [17] in the domain of plankton classification, as seen in Figure. 1. The design and the implementation of this framework lead to a list of contributions stated as follows:

1) Customizing the Siamese network for planktonic domain adaptation
2) Tailoring the Siamese network for multi-class planktonic classification
3) Adapting the framework to few-shot setting
4) Identifying unseen classes in-situ

The Siamese network is a *one-shot* learning algorithm that employs a triplet loss function [18] to estimate and rank similarities naturally between an unlabeled object class and labeled reference images based on a DNN optimized embedding, where the embeddings' Euclidean distance corresponds to a degree of similarity. This method excels in generalizing based on limited available data, for new data as well as for new classes from unknown distributions. In the proposed framework, we leverage this similarity metric to determine species of plankton, significant inter-species variation or mutation, while simultaneously cataloging novel species unseen by the system before. An additional advantage introduced by the Siamese network is the convenience of adding classes to the classifier without the need for retraining the model. The newly explored classes of planktonic species can be added to the framework in-situ and in real-time. The Siamese class of network architectures is proposed initially with a simple CNN of four convolutional layers used as the sub-network for the encoding generation. This modular architecture enables the network to feature a flexible design. Hence, we were motivated to additionally propose alternative sub-networks which are more suited for the task of classifying planktonic

This research is funded by RCN FRINATEK IKTPLUSS program (project number 262701) and supported by NTNU AMOS.

organisms, and further perform a thorough comparative study on these substitute networks, with a focus both on accuracy and computational speed.

As an extension to our work, we explore few-shot class of models introduced in [19] and [20] designed for ImageNet [21] and miniImageNet [19] classification. This extension, based on empirical evaluations, achieves higher accuracy while maintaining the beneficial purpose of our proposed framework. The resulting architecture further enhances real-time image analysis feedback, which in turn will enable the process of continually update probability density maps and eventually provide insights on planktonic abundances and their inter-species variations and mutations.

II. Background

This section details the basic concepts of the *one-shot* and *few-shot* learners, which represent the theoretical foundation of the classifier frameworks sought in this paper.

A. One-shot learning

In traditional supervised image classification techniques [7] [8], the goal for the model is to identify class-specific features in order to make a class prediction. These features are highly trained and specialized for every class in a dataset, requiring a lot of training examples and computational power for training to generalize learned features for accurate classification. These techniques are, therefore, ill-suited for cases where only a small amount of data is available.

The goal of *one-shot* learning is to successfully recognize classes previously unseen in the training process with the aid of just a single example of that class hence reducing the amount of labeled data needed for the recognition. In order to do this, a *one-shot* algorithm employs a so-called knowledge transference scheme. This scheme typically uses general information from previously learned training classes to quickly and efficiently adapt to new data.

The *one-shot* setting is an extremely challenging problem given the considerable information variance that can occur even within a single class of any dataset. Because of this, *one-shot* learning is often used in narrower domains where the variance in the data is smaller, hence more controllable, like in facial recognition and user identification systems [18] [22]. For such systems, to react in real-time and in a speedy manner, it is undesirable to include thousands of pictures in the process of individual recognition, a typical requirement for a traditional image recognition model. Another common task used in benchmarking the *one-shot* learners is character recognition [17], that recognizes handwritten characters of different alphabets.

B. Few-shot learning

The few-shot learning problem is only an extension of the *one-shot* learning problem. The goal of few-shot learning is to successfully recognize classes previously unseen in the training process with the aid of a few examples of that class. The difference between the two problems is the amount of reference samples given that help classifying unseen classes. Few-shot learning algorithms, compared to *one-shot* algorithms, hence need more labeled data from the classes to be predicted.

The additional data provided to few-shot-based algorithms compared to *one-shot*-based algorithms increases the models potential to generalize from the previously unseen data. In general, *few-shot* learners score significantly better at classification tasks in comparison to *one-shot* learners [19] [20] [23] at the cost of requiring more labeled data and more computational power.

Notation: In a *few-shot* setting, problems are usually referred to as k-way n-shot, where k denotes the number of classes used in the classification task, and n is the number of samples provided for each class. The term support set refers to all the samples n for all the classes k utilized by the algorithm.

III. Related Work

This section explores some of the existing algorithmic solutions in the literature for the *one-shot* and *few-shot* problems that we studied to recognize species from images in the planktonic domain.

A. Siamese Network

Koch et al. introduced the Siamese network as an approach for the *one-shot* learning problem [17]. A Siamese network is a model that was first developed by [22] [24] in the domain of face recognition. The algorithm works by passing an image through a Convolutional Neural Network (CNN) that outputs an M-dimensional representation of the inputted image in an M-dimensional space $f_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$ where ϕ are the trainable parameters of the model. In this step, the used CNN aims at reducing the high-dimensional representation of the image to a lower-dimension so-called embedding. A second image is passed through the same network and produces a different embedding with an identical dimension M. The image embeddings output from the model are then compared using a distance measurement which infers the assignment of the images to the same or different classes. This measurement is calculated through performing a vector subtraction of the inferred embeddings. The siamese network passes the distance measurements through a fully connected layer [1] followed by a sigmoid activation function [2]. The activation function consequently produces a binary classification that infers whether the images under consideration 'match'; hence they belong to the same class or 'do not match', and they belong to different classes. The Siamese net architecture is depicted in figure 1.

The training process of the siamese network for *one-shot* learning, targeting to learn image recognition and classification, differs from that of traditional neural networks. The *one-shot* learning process utilizes the so-called Triplet loss for the training. It is a training scheme developed by [18], also for use

[1] A fully connected layer is a layer in a neural network where every output of the previous layer is connected to every neuron in the layer [25].

[2] A sigmoid function is an 'S' shaped function, in machine learning the definition $S(x) = \frac{e^x}{e^x+1}$ [26] is often used.

in facial recognition. This method utilizes one anchor image [A], one image from the same class as the anchor called a positive image [P] and one image from a different class called the negative image [N]. We denote the embedding function as f(*). The triplet loss is visualized in figure 2. Given this, a well suited embedding model produces an output that reduces the distance between f(A) and f(P) while increasing the distance between f(A) and f(N), resulting in the loss function:

$$|f(A) - f(P)| - |f(A) - f(N)| + \alpha \leqslant 0 \tag{1}$$

Where α is the margin to avoid the trivial answers of:

$$f(*) = 0 \ \cup \ |f(A) - f(P)| = |f(A) - f(N)| \tag{2}$$

In this training scheme, the model looks at pairs of images and attempts to evaluate the similarity between them; training data is generated by matching two different images at a time, either from the same class or different classes. Creating a sample by matching two images is a decent approach that augments the dataset by $N!$ samples where N is the number of images original in the dataset.

A significant advantage the siamese model holds over traditional classifiers is its ability to classify unknown classes; in other words, classes the model has never seen before. In the case of conventional classifiers, adding a class to the dataset does not only require a large amount of data from the new class but also incurs the need of retraining the entire model from scratch, possibly suffering a performance loss due to the newly introduced class. A *one-shot* learner model is per definition quick to adapt to new classes, requiring no extra training of the model, only one sample of the class in order to start recognizing objects of the new class. Hence, it does not suffer any performance loss on other classes as a consequence of the introduction of the new class, given that it is from the same domain.

A slight weakness with this approach is that the accuracy of the classification is not only dependent on the quality of the query image like in a traditional classifier, but also the quality of the reference image, resulting in one additional point of failure compared to the traditional classifiers.

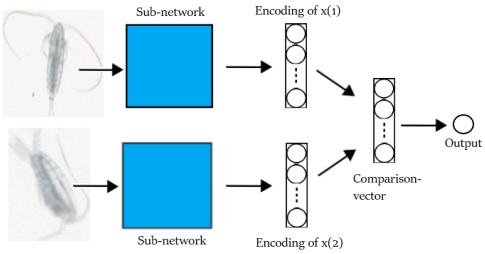


Fig. 1. Siamese net architecture.

Fig. 2. Triplet loss function – Insert reference.

B. Matching Network

The matching network, conceptualized by Vinyals et al. in their paper Matching Networks for One Shot Learning [19] is conceptually similar to the Siamese network: The model uses an image as an input and produces a multi-dimensional embedding for that image by passing it through a trained convolutional network. The siamese net is, however, optimized for a true/false classification, rather than a categorical classification.

The matching network mitigates this problem by allowing n different images from k different classes to be used as input to the network, producing $k*n$ points in the embedding space, and using equation (3) to calculate the probability of the query class belonging to the k different classes

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i, \tag{3}$$

where $x_i$and y_i are the samples and labels of the support set S and $\hat{x}$ and $\hat{y}$ is the query image and its corresponding predicted class respectively. The function a(*) is referred to as the attention kernel. a(*) is defined as:

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}} \tag{4}$$

Where f(*) is the embedding function of the query image, and g(*) is the embedding function of the support set. Vinyals et al. also specified the option of making $f(*) = g(*)$ to simplify the model. Function 4 is the softmax function [27] it calculates the cosine distance [28] between the query embedding and the different embeddings of the support set.

The two functions 3 and 4 produce the equivalent of a kernel density estimator[3]. Not too dissimilar from an advanced k-nearest neighbour estimator[4]. Hence, the Matching network assigns the query image to one of the k classes in the support set by comparing the query embedding to the point distribution belonging to different classes from the support set.

Vinyals et al. also highlight a problem with this approach on its own; depending on the variance of the classes, the matching network has differing performance due to the risk of overlapping classes in the embedding space when classifying closely related classes. For instance, if the model is trained

[3]Kernel density estimation is an estimation of the probability density function of a point cluster.

[4]Nearest neighbour estimator is a method estimating the label of a datapoint by assigning it the same label as the majority of its k-th nearest neighbours.

on imageNet, a specific dataset [21], the model is expected to tell the difference between a dog and a bird, as well as different species of birds. The latter being a more nuanced problem, possibly exhibiting the previously mentioned overlapping classes problem. To mitigate this Vinyals et al. also proposed a solution they called full context embeddings: Before applying the embedding function $g(*)$ to the support set, the support set images are run through an LSTM network [29], encoding the images in the context of the entire support set. Including this LSTM also in the function $g(*)$ results in $g(x_i, S)$ where S is the support set. Altering the focus distance of the problem based on the level of nuance.

C. Prototypical Network

Matching networks significantly improved the accuracy of the Siamese network and improved its general functionality by performing better classifications tasks as they can test a query image against all known classes simultaneously. Despite this fact, it still suffers from the drawback of an unnecessarily complicated full context embedding network to compensate for closely related classes that overlap in the embedding space.

Snell et al. [20] extended the matching network architecture by introducing the Prototypical Network. They proposed a simplification by dropping the LSTM module and attention kernels from the network architecture and replacing them with class prototypes. Instead of using the class cluster densities in the classification process, Snell et al. used the mean position of all the known classes of support embeddings; then, they classified a query embedding as the same class belonging to the closest prototype. This fact nullified the ambiguity in the class overlap as well as made the decision boundaries that identify the differences between the classes more easily interpreted.

Through empirical experimentation, they discovered that using Bregman divergences [30] as a distance measure was favorable to other measures such as cosine distance proposed for the matching network. They used the simplest Bregman divergence, the euclidean square distance, as the measuring distance from a query embedding to the class prototypes to produce the class predictions. The prediction itself is performed by the softmax function over the negative distances from the query embedding to all prototypes

$$p_\phi(y = k|x) = \frac{exp(-d(f_\phi(x), \mu_k))}{\sum_{k'} exp(-d(f_\phi(x), \mu_{k'}))}, \tag{5}$$

where $\mu_{k'}$ is the prototype for class k', and $f_\phi(x)$ is the embedding of query x given trainable parameters ϕ, and $d(a, b)$ is the distance between a and b.

During learning the stochastic gradient descent[5] is performed on the negative log-probability $J(\phi) = -log\, p_\phi(y = k|x)$ of the true class k.

[5]Stochastic gradient descent is a type of iterative algorithm for optimizing an objective function.

IV. Method

Adopting the Siamese net for in-situ classification of plankton is a promising research direction. As detailed in II, Siamese network can distinguish the variations between interspecies as well as identify new classes. To implement the Siamese network classification, we look at their performance on planktonic datasets and investigate possible alterations on their architectures to maximize their accuracy.

In light of this, we discuss the use of altering the subnetwork that produces the embeddings in the similarity space to produce better performance, the difference between a *one-shot* classification and a multi-class classification and the possibility of exchanging sub-schemes. We also detail the possibility of extending *one-shot* to *few-shot* for potential performance benefits. Finally, we showcase how the siamese network can be adopted to identify unknown plankton species.

A. Siamese net customization for domain adaptation

As the *one-shot* sub-network is a simple generic CNN, we explore the possibilities of improving the performance of the model with more advanced, domain-specific sub-networks. Swapping out the original sub-network with other alternative network architectures, keeping the vector subtraction, and the final embedding layer that performs the discriminative classification of the Siamese network unchanged. Table I lists the different networks tested for this process.

TABLE I
Tested Sub-Networks: VGG [9] and ResNet [12] and their variations

Networks
VGG11
VGG16
VGG19
ResNet18
ResNet34
ResNet50
ResNet101
ResNet152

B. Tailoring the Siamese network for multi-class classification

As the *one-shot* learning problem score is based on a 'match' or 'not a match' classification, it is fundamentally different from a multi-class classification problem. The conversion to a multi-class classification problem can be done for the Siamese net by checking the *one-shot* prediction value for several classes k and choosing the class with the highest value as described in equation (6)

$$\hat{y} = argmax_k P(y = k|x). \tag{6}$$

We are interested in both the pure *one-shot* score, to compare it to previous results achieved with the model, as well as the score of the multi-class classification provided by the Siamese network. The latter is the actual in-situ performance of the model.

C. Adapting the framework to the few-shot setting

We extended the Siamese net with a *few-shot* classifier by running the training process over k classes, each with n reference images to further boost the network performance. The k-way n-shot scenario is used to compare the similarity metric over all the reference images of all the given classes with regard to the query image to determine the most similar reference image.

Given more reference images and classes to compare against, it is logical to assume that the framework will reach a higher accuracy score than the *one-shot*'s natively 1-way, 1-shot method of classification since there is more data to compare. In addition to this, it is a more straightforward task to determine the most similar reference image, as opposed to defining if it is 'enough' similar where the term 'enough' first has to be defined. It is worth noting that adding more reference images removes many of the false positives that would result from the typical Siamese net framework. This algorithm was presented in [19] to properly compare the Siamese network to the Matching network. Adapting a 5-way, 5-shot framework in the Siamese Network, in the task of character recognition with the omniglot dataset [31], achieved a 98.4% accuracy. This extension improves the performance accuracy by 7.2% compared to 91.2% achieved by the native 1-way, 1-shot native network in [17].

This train of thought opens up the possibility of extending our work using dedicated few-shot classifiers, such as in [19] [20], which share the same type of similarity metric as the Siamese network, and are known to produce better accuracies. They are trained to perform multi-class classification natively. Because of this, we further look into the viability of *few-shot* learners on the task of identifying planktonic species by applying the models on the task without introducing any further modification to the architectures.

D. Novel class identification

The Siamese network's 'match' or 'not a match' classification method in a multi-class classification setup can leverage a rejection feature to identify new classes that do not exist in the support set. The rejection capability is tested, in both multi-class and few-shot settings to check the applicability of this feature in the Siamese network for the in-situ classification. The network rejects an object in a k-way n-shot setting when no similarity is detected between this query objection in comparison with all classes sample images.

The few shot classifiers [19] [20], however, don't utilize a rejection-based training strategy. The rejection score can only be recorded by the Siamese network when the query object 'do no match' the set of sample images.

V. Datasets

All tests were performed over two well-known planktonic datasets: WHOI-Plankton [15] and the plankton dataset from the Kaggle National science bowl [16]. Both datasets are reputable in-situ planktonic datasets that feature images of singular plankton. They are excellent for research tasks based on pure classification of the various sub-species.

- **WHOI-Plankton:** The WHOI-Plankton dataset is provided by WHOAS: Woods Hole Open Access Server [15]. It consists of 329 936 black and white images of planktonic subspecies divided into 103 categories. The provided categories are non-uniformly represented; the smallest category contains 1 image, while the biggest class contains 266 156 images with varying pixel-size dimensions (height and width).
- **Kaggle-Plankton:** The Kaggle plankton dataset is assembled by Oregon State University's Hatfield Marine Science Center, and provided by Kaggle. It consists of 30 459 black and white images of planktonic subspecies divided up into 121 categories. Similar to the WHOI-plankton dataset, these categories are non-uniformly represented with 9 images in the smallest category and 1 979 images in the biggest class. The pixel-size dimensions of the images also vary.

Both datasets are split into a background set and an evaluation set. The background set is used for training the network and the evaluation set is used for validating the learning process. Both sets contain images from different categories so that no class is represented in both datasets as per the *one/few-shot* problem. The experimentation is performed with two different configurations of n-shot values: $n = 1$ and $n = 5$, placing a minimum requirement of 6 images per species (5 for the support set and 1 for the query image); all classes that contain less than 6 samples are removed from the datasets.

VI. Experiments

We conduct three types of experiments[6] [7]: *one-shot* classification with varying sub-networks of the Siamese network to explore the possibility of optimizing the accuracy for planktonic datasets, a multi-class classification experiment to get a more realistic in-situ accuracy score, performed with both *one-shot* and *few-shot* algorithms, and a novel class identification experiment to investigate the performance of the rejection property of the Siamese network.

A. Sub-networks alteration experiment

In the first experimental scenario, we embed various sub-networks, including the default proposed architecture by [17], within the Siamese network architecture used in *one-shot* setup. Table II shows the results of 'best' accuracy produced by the respective Siamese network variations. After performing several repetitions of this experiment, we observed that there was a considerable variance in the results for the sub-networks, even when trained on the 'exact' same hyperparameters[8]. Variances from the experiment were found in the range of 10% for some sub-networks. This number

[6]Code available online at: https://github.com/AILARON/One-few-shot-classifier.

[7]The original Siamese network code available online at: https://github.com/ac-alpha/One-Shot-Learning-using-Siamese-Networks

[8]Hyperparameters are parameters that direct the learning process

TABLE II
SUB-NETWORK PERFORMANCE

Sub Networks	Performance Metrics	
	Accuracy '%'	*Speed 'img/sec'*
Default	74.1%	2200
VGG11	64.5%	1183
VGG16	66.1%	707
VGG19	63.5%	591
ResNet18	75.9%	2653
ResNet34	72.7%	1791
ResNet50	68.3%	930
ResNet101	72.9%	590
ResNet152	73.4%	435

[a]Speed metric is noted by image comparisons per second, performed on an RTX 2080 ti.

drastically reduces the performance and validity of embedding alternative networks in the architecture. The default baseline Siamese network proved to be the most stable, having a range of variance, for different training rounds, of around 2% while still producing decent accuracy scores. Table II shows that the default baseline Siamese network achieves an accuracy of 74.1% and a relatively high speed of 46403 images per second. It is worth noting that this test is performed on the WHOI-dataset detailed in section V.

B. Multi-class classification experiment

The second tested scenario is the multi-class classification score in a k-way, n-shot setting. Here we have included the performance of the Siamese network as well as the matching network and the prototypical network, described in section III. The results of this test over the Kaggle plankton dataset is presented in table III and the results over the WHOI-Plankton dataset is presented in table IV, where the 5-shot setting provides better accuracy scores for all the networks, and the prototypical networks outperform the rest of the architectures. It is worth noting that increasing the k-way value increases the complexity of the task, and therefore decreases the accuracy, for instance, the Siamese network scores 62.0% in the 5-shot 5-way settings, while it scores 44.0% in the 5-shot 10-way setting for the WHOI dataset.

On the other hand, increasing the n-shot value increases the number of reference images per class and consequently increases the accuracy. For instance, the Siamese network scores 55.4% in the 1-shot 5-way settings, while it scores 62.0% in the 5-shot 5-way settings.

TABLE III
PRELIMINARY CLASSIFIER RESULTS ON THE KAGGLE-PLANKTON DATASET.

	5-way Acc.		10-way Acc.	
Model	1-shot	5-shot	1-shot	5-shot
Siamese Network	43.0%	50.2%	29.2%	35.8%
Matching Network	67.6%	73.6%	50.5%	56.5%
Prototypical Network	70.0%	88.4%	54.5%	76.4%

All the tests are performed without any data augmentation. Koch et al. conclude in their Siamese network paper that this will only result in a small positive performance increase for the model. In [19] and [20], results are achieved with the training parameters: k-way $= 10$ and n-shot $= 5$. These parameters were chosen because Snell et al. state in their paper [20] that the few-shot classifiers work best when trained with a higher k-way value and equal n-shot value compared to the test scenario. Hyperparameters in [17] are not available for the algorithm; thus the network, in this experiment, is trained in the standard *one-shot* manner.

TABLE IV
PRELIMINARY CLASSIFIER RESULTS ON THE WHOI-PLANKTON DATASET.

	5-way Acc.		10-way Acc.	
Model	1-shot	5-shot	1-shot	5-shot
Siamese Network	55.4%	62.0%	34.8%	44.0%
Matching Network	61.4%	65.0%	43.9%	47.2%
Prototypical Network	58.8%	83.6%	43.5%	69.7%

C. Novel class identification experiment

The third tested scenario is the Siamese network's ability to predict novel classes, aka. classes that do not exist in the support set. The experiment is also conducted with the same k-way, n-shot configurations as the previous test, as this is a significant scenario for in-situ classification tasks. For a class to be marked as a novel class, the average value from all classes had to be deemed 'no match'. This test is conducted with 50% of the query images belonging to a plankton class present in the support set, and 50% of the query images are 'new classes' which do not belong to any of the known plankton classes. Table V presents the accuracies of the Siamese network for the pure task of rejection versus inclusion with regards to the classes in the support set. Here, a 50% score is equivalent to the trivial solution of guessing only positive or only negative given our 50/50 divide of the test data.

TABLE V
NOVEL CLASS IDENTIFICATION RESULT OF THE SIAMESE NETWORK ON THE WHOI-PLANKTON DATASET.

5-way Acc.		10-way Acc.	
1-shot	5-shot	1-shot	5-shot
53.9%	53.7%	52.2%	51.3%

Table VI shows the confusion matrix resulting from the 5-way, 5-shot task, where true positive indicates an image class correctly classified as known, while the true negative is a new class that is promoted as a class never seen before. Only 57 images representing 5% of the whole dataset were correctly predicted as new classes out of the 500 examples of new classes that were used during testing. Based on these results, we can observe that the network tends to allocate 90% of the images to the known classes.

VII. DISCUSSION

Results from the standard *one-shot* experiment show that the 'best' sub-network produces decent max accuracy scores. However, advanced sub-networks such as the VGG and

TABLE VI
CONFUSION MATRIX FOR NOVEL SPECIES IDENTIFICATION

		Predicted	
		Positive	Negative
Actual	True	480	57
	False	443	20

ResNet, which proved to be top performers, especially in the traditional classification problem, produce significant anomaly variance. We argue that the reason behind this large variance is due to the variational degree of overfitting [9], resulting in a performance degradation that depends on the type of data presented in the training process. It is worth noting that the base model is the best overall performer in the Siamese net given the relatively high accuracy as well as the reliable performance score it delivers.

The *few-shot* multi-class classification experiment shows clear the drawback of adopting the Siamese network in a *few-shot* setting. Results show an accuracy degradation for the multi-class classification problem in the planktonic domain, as opposed to a significant increase in the measurements as described in [19]. Accordingly, we discourage the use of the Siamese network as a multi-class classification solution for the in-situ plankton classification. We argue that the model, in this case, prioritizes some planktonic species over others during the classification due to the fact that those species appear more frequently in the *few-shot* test. Favoring species amplifies the error rate of the *one-shot* implementation. The *few-shot* learners, designed to use the entire set of clusters that are created by the support set to draw conclusions based on the data presented as a whole, even if it is limited, are more suited for the problem of multi-class classification.

In the experiment of new class detection in a multi-class setting, we observe from the confusion matrix that the Siamese net tends to allocate the query object to one of the classes presented in the support set.

VIII. CONCLUSION AND FUTURE WORK

In this paper we explore different essential tasks for the in-situ classification of planktonic species, such as multi-class classification and novel class identification. We test the adaptation of the *one-shot* algorithm Siamese network to these tasks and explore possibilities to optimize the framework to produce the best possible results. We also look into the viability of using *few-shot* classifiers which are known to produce better results than *one-shot* algorithms at the expense of requiring more data and more computational time.

Conducted experiments show that Siamese network produces low accuracy scores in the multi-class classification setting. Hence they are not suitable in the scenario of inter-species variations as this is a more nuanced and difficult task than the standard classification. The *few-shot* classifiers, and especially the Prototypical networks, perform markedly better at the plankton multi-class classification problem, making them far more suitable for in-situ plankton classification. They are, however, no direct substitution for the Siamese network, as they lack the ability to identify unknown and novel plankton species. This shortcoming marks the outline of our future work, to mitigate it and investigate the possibility of including this feature in the *few-shot* architecture to improve the proposed framework aiming at enhancing the in-situ image analysis processes by providing insights on planktonic inter-species distributions.

[9]Overfitting is an estimation that is too exact for a set of data, possibly leading to poor estimation of future observations [32]

REFERENCES

[1] P. F. Lermusiaux, "Uncertainty estimation and prediction for interdisciplinary ocean dynamics," *Journal of Computational Physics*, vol. 217, pp. 176–199, 2006.

[2] P. C. Reid, J. M. Colebrook, and J. B. L. Matthews, "The continuous plankton recorder: Concepts and history, from plankton indicator to undulating recorders," *Prog. Oceanogr*, vol. 58, p. 117–173, 2003.

[3] E. J. Davies and R. Nepstad, "In situ characterisation of complex suspended particulates surrounding an active submarine tailings placement site in a norwegian fjord," *Regional Studies in Marine Science*, vol. 16, pp. 198–207, 2017.

[4] E. J. Davies, P. J. Brandvik, F. Leirvik, and R. Nepstad, "The use of wide-band transmittance imaging to size and classify suspended particulate matter in seawater," *Marine pollution bulletin*, vol. 115, no. 1-2, pp. 105–114, 2017.

[5] M. D. Ohman, R. E. Davis, J. T. Sherman, K. R. Grindley, B. M. Whitmore, C. F. Nickels, and J. S. Ellen, "Zooglider: An autonomous vehicle for optical and acoustic sensing of zooplankton," *Limnology and Oceanography: Methods*, vol. 17, no. 1, pp. 69–86, 2019.

[6] A. Saad, E. Davies, and A. Stahl, "Recent advances in visual sensing and machine learning techniques for in-situ plankton-taxa classification," presented at Ocean Sciences Meeting 2020, San Diego, CA, 16-21 Feb., 2020, 636384.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

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

[10] J. Dai, R. Wang, H. Zheng, G. Ji, and X. Qiao, "Zooplanktonet: Deep convolutional network for zooplankton classification," in *OCEANS 2016-Shanghai*. IEEE, 2016, pp. 1–6.

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

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[14] H. M. Sosik and R. J. Olson, "New technologies to acquire time series of phytoplankton community structure: Submersible image-in-flow cytometry and automated taxonomic classification," *Proceedings of Ocean Optics XVIII*, vol. 22, no. 1, pp. 1–13, 2008.

[15] E. C. Orenstein, O. Beijbom, E. E. Peacock, and H. M. Sosik, "Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification," *arXiv preprint arXiv:1510.00745*, 2015.

[16] Kaggle, 2015, [Accessed 30-March-2020]. [Online]. Available: https://www.kaggle.com/c/datasciencebowl/data

[17] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[19] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3630–3638. [Online]. Available: http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.pdf

[20] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4077–4087. [Online]. Available: http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning.pdf

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

[22] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[23] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *CoRR*, vol. abs/1703.03400, 2017. [Online]. Available: http://arxiv.org/abs/1703.03400

[24] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[25] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[26] M. N. Gibbs and D. J. MacKay, "Variational gaussian process classifiers," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1458–1464, 2000.

[27] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *International Workshop on Multiple Classifier Systems*. Springer, 2003, pp. 125–134.

[28] A. Singhal *et al.*, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] L. M. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR computational mathematics and mathematical physics*, vol. 7, no. 3, pp. 200–217, 1967.

[31] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[32] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

Leveraging Similarity Metrics to In-Situ Discover Planktonic Interspecis Variations or Mutations

Andreas L. Teigen
Dept. of Engineering Cybernetics
NTNU Trondheim, Norway
andrealt@stud.ntnu.no

IEEE OES Oceanic Engineering Society

marine technology SOCIETY
Opportunity runs deep™

Introduction

- Plankton are vital to both land and sea. Mapping and monitoring planktonic-distribution in the ocean is of great importance in understanding these ecosystems as well as gaining insights on the general health of our planet.
- We explore the utilization of one-shot classifiers on in-situ captured planktonic images.
- The similarity-based classification method used by the one-shot classification algorithms gives rise to various research opportunities outside of the typical classification paradigm.
- Exploring the development of a proficient plankton image classification framework that can determine how different or how similar two plankton specimens are, can inform us if they belong to the same species, if they are different variations of the same species or if we have encountered an unseen class of plankton. This similarity is valuable information that assists in further mapping the diversity of the planktonic species.
- We further extend the Siamese one-shot classifier with few-shot classifier to improve the model performance.

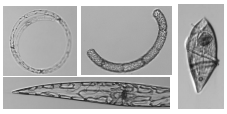
Contribution

- Customizing the Siamese network for planktonic domain adaptation
- Tailoring the Siamese network for multi-class planktonic classification
- Adapting the framework to a few-shot setting
- Identifying unseen classes in-situ

Dataset

Experiments were conducted with 2 well-known in-situ planktonic datasets:

- WHOI-Plankton[1] – 320 000 plankton images, 103 classes
- Kaggle Plankton (National science bowl)[2] - 30 000 plankton images, 121 classes.



Examples images of 3 different classes from the WHOI-Plankton dataset

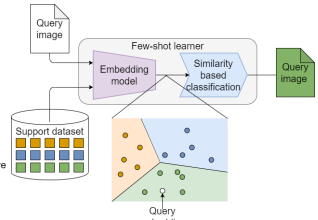
Theory

We were interested in looking into two different types of algorithms for adaptability on in-situ plankton classification:

- **One-shot algorithm:** Artificial intelligence algorithms that are designed to learn new classes from a single example image.
  - Siamese net[3]
- **Few-shot algorithm:** Algorithms that are designed to learn new classes from only a few sample images, but has greater potential for generalizability compared to one-shot learners.
  - Matching network[4]
  - Prototypical network[5]

These algorithms learns class similarities based on the Triplet loss[6] learning scheme:

Triplet loss: Learning scheme for the similarity-based classifiers

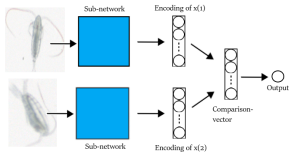


Few-shot learner architecture

Method

Three experiments are conducted:

- **Sub-network optimization:** Finding better suited neural networks for identification of planktonic datasets
- **Multi-class classification:** Adapting the Siamese network for multi-shot classification and comparing results with few-shot classifiers
- **Novel class identification:** Measuring capability of the Siamese network to identify novel classes.



Siamese network architecture

Results and discussion

Results from the three experiments are listed in table 1,2,3:

Table 1 – Siamese network: one-shot classification results on the Kaggle dataset

Sub Networks	Performance Metrics	
	Accuracy '%'	*Speed 'img/sec'*
Default	74.1%	2200
VGG11	64.5%	1183
VGG16	66.1%	707
VGG19	63.5%	591
ResNet18	75.9%	2653
ResNet34	72.7%	1791
ResNet50	68.3%	930
ResNet101	72.9%	590
ResNet152	73.4%	435

Table 2 – All classifier results on the multi-class few-shot task with the WHOI-Plankton dataset

Model	5-way Acc. 1-shot	5-way Acc. 5-shot	10-way Acc. 1-shot	10-way Acc. 5-shot
Siamese Network	55.4%	62.0%	34.8%	44.0%
Matching Network	61.4%	65.0%	43.9%	47.2%
Prototypical Network	58.8%	83.6%	43.5%	69.7%

Table 3 – Siamese network result on the novel class identifcation experiment (50% novel classes, 50% known classes)

5-way Acc. 1-shot	5-way Acc. 5-shot	10-way Acc. 1-shot	10-way Acc. 5-shot
53.9%	53.7%	52.2%	51.3%

Conclusion:

- Conducted experiments show that Siamese net produces low accuracy scores in the multi-class classification setting. Hence, they are not suitable in the scenario of inter-species variations.
- The few-shot classifiers, and especially the Prototypical network perform markedly better at the plankton multi-class classification problem.
- Few-shot classifiers are however no direct substitution for the Siamese net as they lack the ability to identify unknown and novel plankton species.
- This shortcoming marks the outline of our future work, to mitigate it and investigate the possibility of inclusion of this feature in the few-shot architecture to further improve the proposed framework.

References:

[1] E. C. Orenstein, O. Beijbom, E. E. Peacock, and H. M. Sosik, "Whoiplankton-a large scale fine grained visual recognition benchmark dataset for plankton classification," arXiv preprint arXiv:1510.00745, 2015.
[2] Kaggle, 2015, [Accessed 30-March-2020]. [Online]. Available: https://www.kaggle.com/c/datasciencebowl/data
[3] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in ICML deep learning workshop, vol. 2. Lille, 2015.
[4] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra,"Matching networks for one shot learning," in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3630–3638.
[5] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4077–4087.
[6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

Co-authors:
Aya Saad - aya.saad@ntnu.no
Annette Stahl - Annette.Stahl@ntnu.no
Dept. of Engineering Cybernetics
NTNU Trondheim, Norway

Acknowledgments:
Thanks to ONR and NOAA for their grants to help present this work

This research is part of the AILARON project funded by RCN Frinatek IKTPLUSS program (project number 262701) and supported by NTNU AMOS
https://www.ntnu.edu/web/ailaron

The Research Council of Norway
NTNU AMOS Centre for Autonomous Marine Operations and Systems
SINTEF
NTNU Department of Engineering Cybernetics

	5-way				10-way		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
ABOD	73.9%	72.9%	71.3%	71.0%	73.8%	72.6%	71.5%
AvgKNN	78.0%	77.6%	75.8%	75.4%	76.3%	76.9%	76.1%
COF	76.2%	73.6%	71.6%	71.3%	74.9%	73.5%	71.8%
LMDD	62.3%	66.6%	66.7%	67.1%	60.9%	67.0%	66.9%
XGBOD	50.4%	74.4%	74.7%	78.6%	26.0%	57.4%	60.7%
Feature Bagging	**78.3%**	79.0%	76.0%	75.9%	76.4%	**78.5%**	76.6%
IForest	68.3%	69.6%	69.8%	69.8%	67.1%	69.7%	70.0%
KNN	77.3%	77.4%	76.0%	75.8%	**77.0%**	76.9%	76.5%
LODA	61.8%	65.8%	67.0%	67.4%	62.2%	66.4%	66.9%
LOF	77.4%	77.8%	76.1%	75.6%	75.8%	76.7%	76.6%
LOCI	66.6%	79.1%	**79.6%**	**80.9%**	66.6%	78.1%	**80.0%**
MedKNN	78.0%	77.4%	75.8%	75.4%	76.3%	76.6%	76.1%
OCSVM	7.5%	**79.6%**	**79.6%**	78.1%	9.4%	78.3%	79.7%
PCA	0.2%	59.0%	67.4%	70.0%	0.5%	60.8%	66.8%
SOD	74.7%	75.3%	74.2%	73.6%	74.3%	74.7%	73.5%
SOS	63.6%	76.8%	74.8%	75.3%	61.5%	76.7%	75.4%

Table 10.1: F1-score of the outlier detection architecture on the pure rejection task on the kaggle plankton dataset.

	5-way.				10-way		
Algorithm	5-shot	20-shot	30-shot	50-shot	5-shot	20-shot	30-shot
NNO	65.1%	64.9%	65.2%	70.2%	51.9%	52.2%	49.3%
OpenMax	6.9%	26.0%	36.0%	45.2%	0%	17.1%	12.6%
DOC	43.1%	68.3%	68.0%	67.4%	46.6%	63.7%	65.8%

Table 10.2: F1 of the open set recognition architecture on the pure rejection task on the kaggle plankton dataset.

Andreas L. Teigen

Few-shot open world learning

NTNU
Norwegian University of
Science and Technology