NHL STENDEN
computer vision & data science

Optimizing Outdoors Aphid and Insect Detection: A Few-Shot Learning Approach

NHL Stenden Lectoraat in Computer Vision & Data Science

Arsalan Kakaei

Supervisors: Lucas Ramos

Abstract—Aphids pose a significant threat to crop yield and quality, leading to economic concerns in agricultural production. Traditional pest monitoring methods are time-consuming, prompting the need for efficient visual image-based techniques. In outdoor data collection, obtaining large amounts of data for deep learning can be challenging. Therefore, we opted for a Few-shot learning approach. Our study focuses on Few-shot learning to quickly and adaptively identify aphid species with minimal samples. Utilizing prototypical networks, this study investigates two dataset compositions: a 3-class set (aphid, insects, and background) and a 9-class set (2 aphid classes, 6 insect classes, and background). We apply 3 different CNNs as the backbone of Prototypical Networks to both datasets creating 6 scenarios. For each scenario, we evaluate and compare the performance of CNNs. This involves assessing 6 different distance methods, two of which are newly introduced compared to prior research. The goal is to examine the effectiveness of backbones and distance methods in diverse experimental settings and find a reliable way to detect aphids using a few samples. The experiment with Resnet50, 3-class dataset, and cosine similarity method, as one of our scientific contributions, showed impressive recall values of 98% for aphid classification and 99% for overall insect classification. Notably, Euclidean distance and Mahalanobis divergence proved reliable, with Euclidean being more computationally efficient. Larger CNNs yielded superior results, but their resource demands should be considered. The 9-class dataset outperformed the 3-class dataset, highlighting the importance of data richness. Our findings affirm the reliability of prototypical networks as a dependable few-shot learning method, achieving satisfactory levels of aphid detection.

Index Terms—Few-shot learning, Prototypical networks, Aphid, SqueezNetn, Resnet18, Resnet50

1 Introduction

In agriculture, identifying and classifying crop pests is a significant challenge because many pest species look alike [1]. Insects are a major threat to crop yield and quality worldwide [2]. For example, in 2013, the sorghum aphid, Melanaphis sorghi (Theobald), caused economic concerns during sorghum production in the US [3]. Earlier studies have highlighted the detrimental impact of various aphid species on crops and agricultural yields. Examples include the black bean aphid, pea aphid, and green peach, which have been identified as particularly harmful to agricultural production[4]. Monitoring insect pests quickly and reliably is crucial for predicting their populations and taking control measures [2]. This helps in choosing effective pesticides or biological methods to stop pests from spreading further [1].

Manual pest classification is difficult, time-consuming, and requires expertise [1]. To address this, various techniques exist for detecting and classifying pests. They fall into several categories such as acoustic sensors for noisy insects, photoelectric and radar sensors, and visual image-based methods. Visual image methods hold immense potential as they can identify pests by analyzing their distinctive features, offering the advantages of cost-effectiveness, efficiency, and precision [5] [6].
Currently, sticky plates are used in potato farms around Friesland to gather aphid samples and then to manually count the aphids. This method is both time-consuming and requires expertise. This process by capturing data from the sticky plates and creating a dataset. We employ computer vision to detect aphids on these plates and also count the number of insects for each species of interest. More recently, we have transitioned to using a camera in an outdoor setting instead of sticky plates to collect data efficiently.[7].

Some insect pests have random characteristics, and the distribution of plant pests is transnational. This makes it hard to collect samples. Deep learning models need large datasets for training. Lack of image data can lead to overfitting, making it challenging to detect sudden, unknown insect pests [2]. Many studies on pest detection rely on deep learning (DL)[2]. Although DL performs acceptable results when abundant labeled data is available, there is an increasing interest in scenarios with limited images, particularly in fields facing data scarcity, such as identifying insect pest species [2]. Few-shot learning has been proposed to address this issue. It is an emerging method that uses minimal training data, making data collection easier in various fields[1].

This paper discusses how few-shot learning can be used to detect insect species using a minimal number of samples. In outdoor settings, new species may appear unexpectedly. Traditional models might struggle to recognize them, highlighting the need for adaptability. Few-shot learning proves valuable by swiftly accommodating new information, making it a fitting solution for handling unforeseen species in outdoor environments. Our paper explores the effectiveness of few-shot learning in addressing detecting insect species using a minimal number of samples. Our primary objective was to assess whether using a more complex or simpler backbone would enhance performance. Additionally, we investigated the impact of employing distance metrics better suited for comparing features extracted from diverse backbone networks. By combining these models, we aimed to attain the highest accuracy in detecting aphids. Our approach has the potential to help farmers make well-informed decisions regarding the timely deployment of suitable insecticides or other pest management strategies, ultimately bolstering agricultural productivity and food security.

- *Arsalan Kakaei is a Computing Science student at the NHL Stenden University of Applied Sciences, E-mail: arsalan.kakaei@student.nhlstenden.com.*
- *Lucas Ramos is a teacher and researcher at the NHL Stenden Lectoraat in Computer Vision & Data Science, E-mail: lucas.ramos@nhlstenden.com.*

2 State of the art

Deep learning has recently been applied to pest recognition to help farmers take prompt and proper actions to prevent reductions in crop quantity and quality. Deep learning typically demands an extensive corpus of annotated training data, a task frequently delegated to domain specialists for meticulous annotation. However, because collecting insect images in natural environments is difficult and obtaining proper annotations from specialists is costly, deep learning is not an optimized way for pest recognition tasks. [8].

Recently, few-shot learning has been used as a method for object detection and classification based on just a few training samples. It aims to learn new concepts from a few labeled examples, reducing the difficulty of creating large datasets in some cases. Therefore, few-shot learning could help identify unknown pests by relying on the expertise of insect specialists once a few samples are recognized[3] and [9].

For example, the work of [8] supports the use of few-shot learning methods for spotting plant diseases in their survey paper. They highlight the model's impressive results, especially when dealing with limited samples. Additionally, they discuss various types of augmentation to create diverse images and enhance small datasets, noting that physical transformation is a more straightforward implementation.

In a similar vein, in [3], they surveyed more than 27 studies focusing on classifying and detecting insects in field images using deep learning. More than 50% of these studies incorporated data augmentation to enhance their results. The majority of researchers employed different deep learning models such as Alexnet, ResNet, Bridgenet-19, and YOLO. Remarkably, among the studies referenced [1], only one embraced few-shot learning, underscoring its effectiveness in identifying previously unknown insects. This study not only diverged from conventional approaches by utilizing less data but also yielded remarkably high accuracy. This outstanding performance was particularly noteworthy when compared to other research cited in [3]. Additionally, the research highlighted various methods of data collection from fields, including sticky traps, on plants, and in traps.

In [1], researchers designed an insect detection system using the NBAIR and Li datasets. They effectively classified 50 different insect species by employing prototypical networks. Euclidean distance served as the distance method for their prototypical networks, and they utilized 7 diverse models as backbones to extract input data features. Their designed CNN outperformed the other 6 bigger models (GoogLeNet, AlexNet, VGG-16, VGG-19, ResNet-50, ResNet-101), achieving accuracy rates above 95% and 96% for each dataset, respectively.

Notably,[10] expanded the use of Prototypical Network-based few-shot learning by testing various distance metrics, such as Euclidean Distance, Mahalanobis, Kullback–Leibler, and Itakura–Saito. These metrics determined the proximity between prototypes and query insects, with a single backbone serving as the embedding part of their models. Their study employed a novel dataset, IP102, comprising 102 categories of adult insect images and early-stage insect images. The most successful results were achieved using the Kullback–Leibler divergence measure, reaching an accuracy of 86.33% for adults and 87.91% for early stages. Both Matching Networks and Prototypical Networks were explored, revealing that Prototypical Networks produced superior results in classifying their dataset.

The research mentioned above inspired me to explore various distance methods, experiment with different backbones, and investigate two distinct dataset compositions.

Fig. 1. The new approach captures flying insects using a camera directed at a yellow sticky plate. This setup includes LED bars, a trigger, and a line-scan camera recording at 1936x2 resolution. When an insect disrupts the trigger's light, the main camera captures a focused image illuminated by flashing LEDs for 133 μs, saving it at 4512x4512 pixels resolution. The camera was strategically placed on several Frisian farms to capture the necessary data.

3 Materials and methods

3.1 Dataset

In this study, a comprehensive dataset was created using outdoor flying insect images captured with a camera setup in the Frisian farms. When an insect crossed the lens of the setup, the camera captured its image and transmitted it to a server for further processing, Fig 1 [7].

These images served as our primary data source, which we meticulously cleaned and categorized into eight distinct insect types and background classes, shown in Fig 2. We added water drops to our classes since our camera often captures these elements. Additionally, during data preprocessing, reflections were treated as separate objects, requiring separate detection.
Once our data was prepared, we divided it into 3 sets for our few-shot learning experiments, shown in Table 1. In our first dataset, we categorize insects into 8 main classes, and there is an additional background class featuring water droplets and reflections. Our primary research focus is on the two classes of aphids: aphid and wingless aphid. Additionally, we include classes like moth and wasp, which are of particular interest to farmers for counting purposes Furthermore, we generated a 3-class dataset as our second dataset. This involved using the same data but organizing it into three categories: one for aphids, another for a combined group of the remaining 5 insects labeled as the "insects" class, and the third for the backgrounds class Table 2.

The cleaned dataset was then partitioned, with 80 percent of the data allocated for the training set (Support set) and validation set, while the remaining 20 percent was specifically designated for the test set, known as the Query set in few-shot learning.

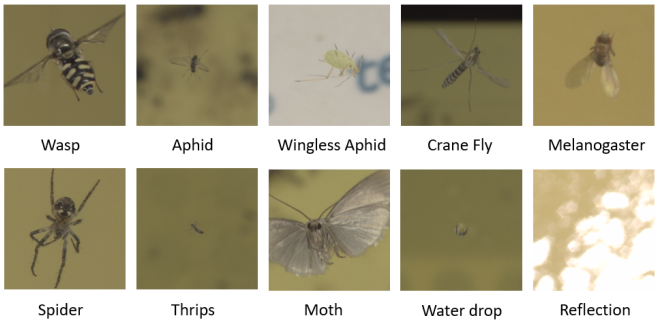

Fig. 2. This figure shows examples of the insect classes in our study, along with two backgrounds featuring water drops and reflections.

Name	Train	Validation	Test
Aphid	14	14	53
Aphid Wingless	14	14	44
Crane Fly	85	14	25
Melanogaster	85	14	25
Moth	14	16	5
Spider	18	14	8
Thrips	41	18	15
Wasp	17	15	8
Backgrounds	145	34	45

Table 1. This table shows information about the first dataset consisting of 8 main insect classes, with a background class including water droplets and reflections.

3.2 Few-Shot learning

Few-shot learning involves learning from a small amount of data and can be implemented in various ways [1]. Metric-based few-shot classification methods predict based on similarities between query images and support examples. These methods include Siamese Neural Networks, Matching Networks, Prototypical Networks, and Graph Neural Networks. Prototypical Networks represent each class and classify query points based on their distance from these prototypes [2]. Data augmentation is essential to expand training data, especially when sampling is challenging, and it includes geometric transformations, color space adjustments, image mixing, generative adversarial networks, and neural style transfer [2][3][8].

3.3 Prototypical network

Prototypical Networks operate on the concept that there is an embedding where points cluster around a single prototype representation for each class [11]. These prototypes are created by calculating the mean of the data belonging to each respective class, providing a central representation that captures essential features for classification. To do this, we learn a non-linear mapping of the input into an embedding space using a CNN and take a class's prototype to be the mean of its support set in the embedding space[3]. Classification is then performed for an embedded query point by simply finding the nearest class prototype. This is done by creating a linear classifier based on the distance method[11].

As an example shown in Fig 3, for each class, a prototype indicator is calculated using a function, such as the mean. When a new data point, represented by X, is introduced to the Prototypical Network, its distance to the prototypes C1, C2, and C3 is computed. The data point is then assigned to the class with the smallest distance, which in this case is C1. In Prototypical Networks, we utilize specific terms to elucidate the settings employed in training the model with data. First, we refer to the classification of the number of classes as N-Ways. Secondly, the images fed to Prototypical Networks for creating prototypes, such as C1, C2, and C3, are designated as the Support Set. The number of images belonging to the Query Set for each class is represented as N-Shots. Moving on, we use "N-Query" or simply "Q" to denote the number of images in the Query Set for each class. In this example, represented by X, these images are fed to Prototypical Networks to compare their distance with prototypes and train our model.

Name	Train	Validation	Test
Aphids	14	14	97
Insects	260	91	86
Backgrounds	145	34	45

Table 2. 3-Ways dataset information table. It grouped two distinct aphid classes, as detailed in Table 1, under the categories of aphid, water drop, and reflection, treating them collectively as background elements. The remaining insect classes are categorized simply as "insects."

In this study, we have organized our data into distinct classes of insects, each represented by five reference images, known as "5-Shots[1]." To tackle the task of classifying insects, our model computes the class prototypes by determining the mean feature vectors for each class. We then leverage the 6 different distance methods consisting of Euclidean Distance (EU), Mahalanobis distance (MA), Kullback–Leibler divergence (KL), Itakura–Saito distance (IS), Wasserstein Distance (WA), and Cosine Similarity (CO) to measure the dissimilarity between these class prototypes and the query image. This distance computation forms the basis for our loss measurement during each training epoch.

We selected the first four distance methods based on a similar study [10] for result comparison. Additionally, we included the Wasserstein Distance (WA) due to its effectiveness in minimizing the negative impact of irrelevant regions, as suggested in related few-shot learning research [12]. Furthermore, we applied Cosine Similarity (CO) in our tests, inspired by the work of [13] where CO demonstrated the ability to quickly learn novel categories without forgetting the base categories and achieved favorable outcomes.

3.4 Training Process

Our approach follows a straightforward pipeline (Fig 4). Initially, we start by loading data from a dataset that contains color images. As a preprocessing step, we resize these images[2] to a standardized 256x256 resolution. To enhance the diversity of our dataset concerning evaluation tasks, we employ a Task Sampler. The Task Sampler is responsible for creating the support set and query set to feed our model. For each training episode, the support set includes N-Shots of images and their corresponding labels for each class, while the query set consists of N-Queries of images and labels per class. Additionally, the Task Sampler helps enhance data diversity by applying simple horizontal image flips as augmentations[3].

Then, we use 3 pre-trained models, ResNet18, SqueezeNet, and Resnet50 as our Convolutional Neural Network (CNN) architecture for separate experiments. Initially, we feed the support set and the query set data into our CNN to extract essential features from our

[1]We chose to use five shots in our experiments because it is a common practice in the existing literature. This consistent usage across various studies allows for better comparability with the results obtained in our research. By aligning with established standards in the field, we ensure that our findings can be effectively compared to and contextualized alongside those of previously mentioned research, contributing to a more comprehensive understanding of the overall landscape in the domain.

[2]Most of our images are standardized to 400x400 pixels, but a few have smaller dimensions ranging from 150 to 300x300 pixels.

[3]In our experiment the number of classes (N-Ways) = (9 and 3), N-Shots=5 and N-Queries=9

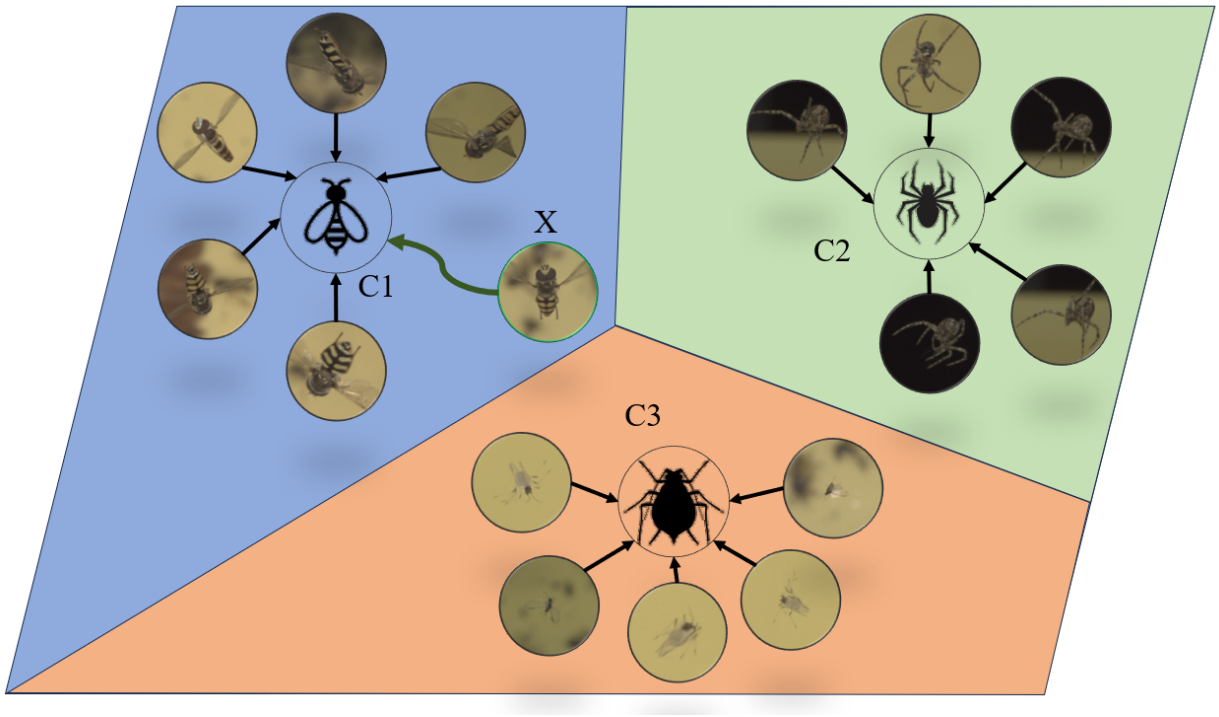

Fig. 3. Showcases an example of the prototypical network applied to 3 classes (3-Ways), each consisting of 5 members (5-Shots).

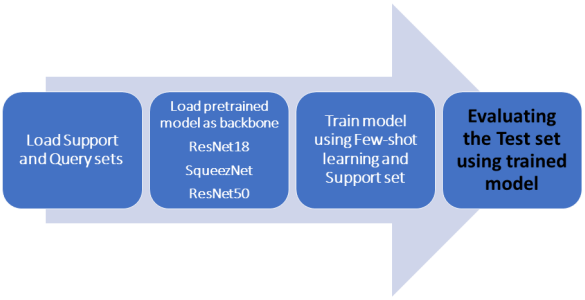

Fig. 4. This figure illustrates the key stages of our pipeline.

images. From CNN's output, we compute the mean of the extracted features from the support set for each class, creating prototypes (Fig 5 - Steps 1 and 2). Subsequently, we gauge the distance between these prototypes and the features extracted from the query set (Fig 5 - Step 3). This process helps in understanding the dissimilarity between classes, a crucial step in our few-shot learning approach. We engage in the training phase, leveraging the power of a Prototypical network. This is where our model learns from the data, adapting to the unique characteristics of the insect classification task.

Finally, we conduct a comprehensive evaluation of the trained model's performance using the test set (Fig 5 - Step 4). We employ evaluation metrics such as precision, recall, F1 score, and accuracy to assess the outcomes of our experiments. Among these metrics, special emphasis is placed on the recall value. This is because our main goal is to accurately identify all true positive instances of aphids, and achieving this objective relies heavily on the recall metric. This concluding step provides valuable insights into the model's accuracy and its suitability for the insect classification task. Our methodological approach in this pipeline allows for a thorough investigation of the effectiveness of few-shot learning in insect classification.

3.5 Backbones

3.5.1 Resnet18

For one of our experiments, we opted for the ResNet18 CNN due to its smaller size, making it suitable for our relatively modest dataset. We enhanced the architecture by adding a Flatten layer to ResNet18 (Fig 6) and utilized a pre-trained version. This choice is influenced by a reference paper that used ResNet18 as the main model for prototypical networks [11].

3.5.2 SqueezNet

In another experiment, we employed SqueezeNet to compare a smaller CNN with Resnet18. We utilized a pretrained SqueezeNet, where we made modifications by removing the classifier layer. Instead, we introduced an Average Pooling layer and a Flatten layer to extract features from our inputs for use in our model (Fig 6).

3.5.3 Resnet50

In a different test, we employed Resnet50 to assess the outcomes of training our model with a larger backbone, comparing it to the results obtained with Resnet18. To modify the architecture, we added a Flatten layer to ResNet50 and used a pretrained version (Fig 6). We made this decision based on a reference paper refer to in the State of the Art section, which highlighted ResNet50 as one of the widely used CNNs for classifying insects [1].

3.6 Distances

In the following formulas, X represents a prototype of support data, Y denotes new data (Query) that needs to be compared with our networks, and n stands for the number of classes (N-Way).

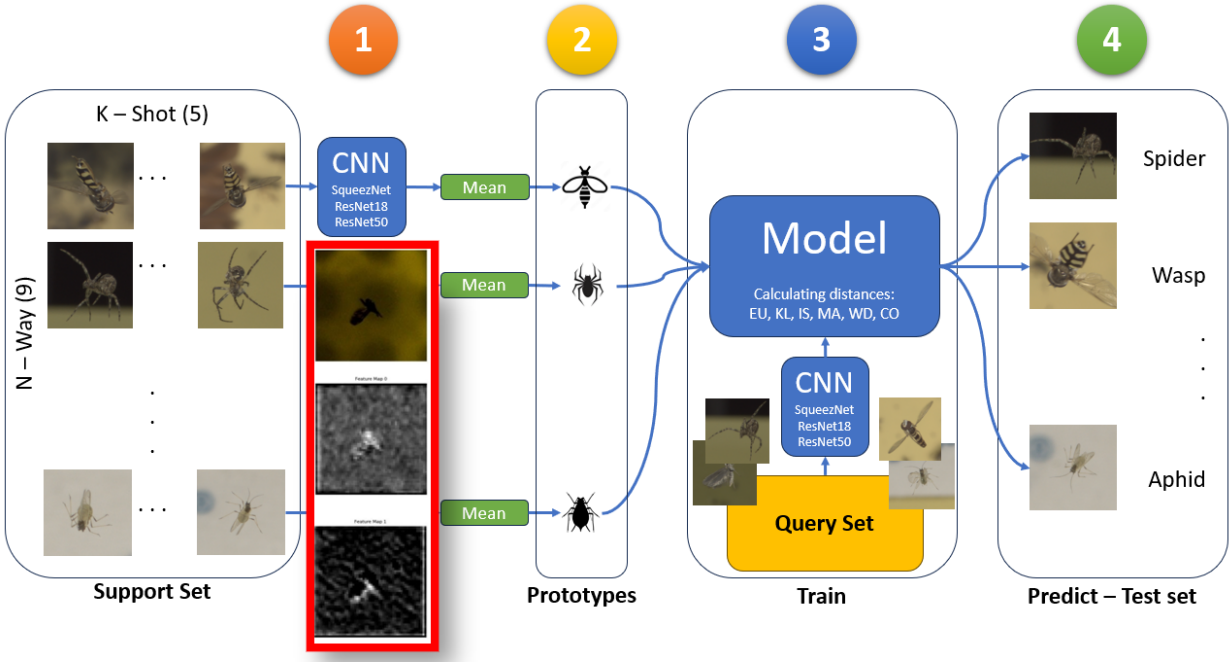

Fig. 5. This figure breaks down the essential components of our model. We utilize three different CNN backbones, Resnet18, SqueezeNet, and Resnet50, for various experiments, employing them separately.

3.6.1 Euclidean Distance (EU)

The Euclidean distance between two vectors X and Y is calculated using the formula:

$$dist(X,Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{1}$$

The EU measures the straight-line distance between two points in the n-dimensional space. It is widely used for comparing the overall spatial difference between vectors.

3.6.2 Mahalanobis Distance (MA)

The Mahalanobis distance between vectors X and Y is computed as follows:

$$dist(X,Y) = \sqrt{(X-Y)^T S^{-1}(X-Y)} \tag{2}$$

Where S is the covariance matrix. MA considers the correlation between dimensions, providing a more accurate measure of distance in the presence of correlated features. This equation attempts to solve the Euclidean distance problem when the data have a linear correlation[4].

3.6.3 Kullback–Leibler Divergence (KL)

The Kullback–Leibler divergence between probability distributions X and Y is given by:

$$dist(X,Y) = \sum_{i=1}^{n} X_i \log\left(\frac{X_i}{Y_i}\right) \tag{3}$$

The Kullback–Leibler divergence measures the difference between two probability distributions. It is commonly used in information theory to quantify how one distribution diverges from another.

3.6.4 Itakura–Saito Distance (IS)

The Itakura–Saito distance between spectra X and Y is defined as:

$$dist(X,Y) = \sum_{i=1}^{n} \frac{(X_i/Y_i - \log(X_i/Y_i) - 1)}{2} \tag{4}$$

IS considers the logarithmic difference between spectral components and is commonly used in speech and audio processing.

3.6.5 Wasserstein Distance (WA)

The Wasserstein distance between distributions X and Y is given by:

$$dist(X,Y) = \inf_{\gamma \in \Pi(X,Y)} \int_{R^n \times R^n} \|x-y\| \gamma(dx,dy) \tag{5}$$

The Wasserstein Distance measures the minimum cost to transport mass from one distribution to another. Here, X and Y are probability distributions, γ represents the set of all joint distributions with marginals X and Y, and $\|x-y\|$ is the distance between elements x and y in the distribution. In simpler terms, γ represents the various ways we can combine elements from X and Y as we calculate the overall transportation cost.

3.6.6 Cosine Similarity (CO)

The Cosine Similarity between vectors X and Y is defined as:

$$dist(X,Y) = \frac{X \cdot Y}{\|X\| \cdot \|Y\|} \tag{6}$$

CO measures the cosine of the angle between two vectors. It is commonly used for comparing the orientation of vectors, providing a measure of similarity rather than dissimilarity.

4 Experiments

In this study, we employed prototypical networks, structuring the research into two scenarios to assess the effects of different dataset compositions, as mentioned in 3.1. Our focus was on testing various distance methods (EU, KL, IS, MA, WA, and CO) and selecting the best one among them in terms of detecting positive instances of aphids. Additionally, we compared the performance of 3 distinct Convolutional Neural Networks (CNNs) – Resnet18, SqueezeNet, and Resnet50 – serving as the backbone for our prototypical networks.

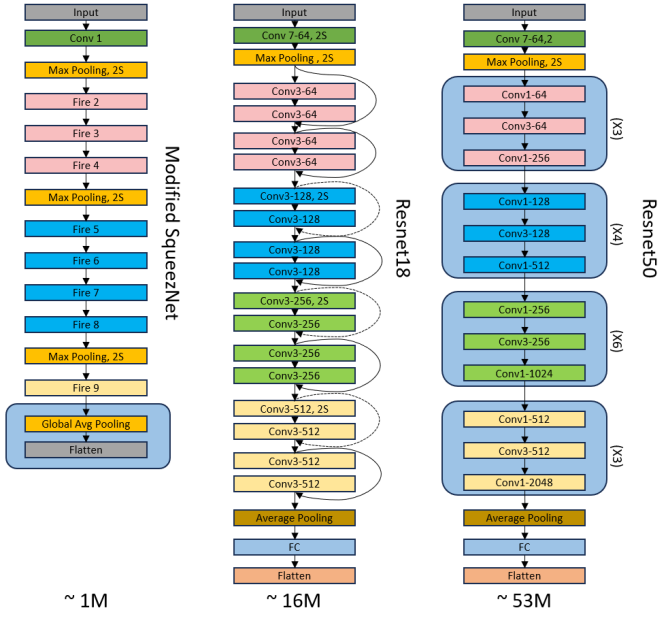

Fig. 6. On the left, modifications were made to the SqueezeNet by removing its classifier layer. Instead, an average pooling layer and a Flatten layer were added in place of the removed classifier. In the middle, a Flatten layer was added to the ResNet18 model. On the right side, a Flatten layer was added to the ResNet50 model. Furthermore, the number of parameters for each CNN is indicated below its diagram, where 'M' represents million.

4.1 Experiment I

In this experiment, our dataset comprises 9 classes, as outlined in Table 1. To train our model, we utilize 6 distinct distance calculation methods: EU, KL, IS, MA, WA, and CO. The experiment is carried out with the following parameter settings: N-Ways set to 9, N-Shots set to 5, and Q set to 9. We conduct this experiment independently for each of our 3 different CNNs (SqueezNet, ResNet18, and ResNet50).

Furthermore, we employ 14 images for each aphid class, aligning with the few-shot learning concept of utilizing a limited amount of input data. For the remaining classes, a larger amount of data is used. Each experiment episode is set to 50, and within each episode, a batch of data is randomly selected, comprising 5 support images and 9 query images. The model is then trained based on this batch.

The training process spans 50 epochs, in each epoch, the model is trained across 20 episodes. Each episode involves using a single batch of data, which comprises the support set and the query set images for each class. Once the data is input, as depicted in Fig 5, the model undergoes training through calculations performed on this dataset. Throughout each epoch, the model undergoes evaluation on our validation dataset to assess its performance and save the best model. After completing the training, we compute evaluation matrices to determine the effectiveness of each distance method and identify which one performs better.

4.2 Experiment II

In our second experiment, we conducted a separate experiment involving 3 categories: aphids, other insects, and background Table 2. The objective of this comparative test is to assess whether grouping other insects and aphids, along with the background in 3 classes, yields better results than classifying each insect separately.

Moreover, similar to Experiment I, we utilize 14 images for the aphid class. We follow the same data utilization pattern for the other two classes as in Experiment I (Table 2). This ensures consistency with the approach of using fewer images for aphid classes, aligning with the principles of few-shot learning, while allowing for a broader dataset for the remaining classes.

During the test phase, we set N-Ways to 3, N-Shots to 5, and Q to 9. Similar to Experiment I, this test uses all 6 distance methods and all 3 backbones. The remaining parameters remain consistent with Experiment I.

4.3 Experiments Setups

The model processes color images in RGB format, standardizing their size to 256x256 pixels. To enhance data diversity, we apply a straightforward random horizontal flip augmentation. Throughout each test, we maintain consistency by using the same random seed, ensuring a fair comparison of results. Our experiments are conducted on a single GPU (NVIDIA A40-12C vGPU (12GB)) for computational efficiency.

5 Results

In the analysis of the results, our focus is on aphids, particularly the recall value. To provide a more accurate comparison, we used the weighted average for calculating aphids' recall, precision, and F1-score collectively. We intentionally refrain from mentioning other evaluation metrics like precision and F1-score because our primary emphasis is on achieving a high rate of detecting positive instances of aphids. The models in our study learn to distinguish between classes by extracting image features using various distance methods. Alongside the Euclidean distance, we explored the outcomes of employing the Mahalanobis distance, Kullback–Leibler divergence, Itakura–Saito distance, Wasserstein Distance, and Cosine Similarity. These methods were incorporated as dissimilarity measures for classifying insects in our datasets and applied across both Experiments I and II. Moreover, we examined the effects of all 3 backbones separately in our experiments. The bold numbers in our results highlight the superior distance method and its corresponding recall, providing a clear indication of the optimal performance in each experiment.

5.1 Experiment I

In our first experiment, we trained and tested prototypical networks with a 9-class dataset using our modified backbones. We conducted a series of different experiments to assess our approach. For each distance method considered, we assessed tasks involving 5 shots, and the results are summarized in Table3 and Table4. Notably, bold numbers in the table highlight the best-performing distance method along with its corresponding recall for this specific experiment.

5.2 Experiment II

In our second experiment, we conducted training and testing of prototypical networks using a 3-class dataset using our modified backbones with the same situation as experiment I. We performed a sequence of experiments to evaluate our approach. For each distance method examined, we evaluated tasks involving 5 shots, and the outcomes are summarized in Table3 and Table4.

5.3 Comparing the experiments' best results

Here, we present the optimal outcomes for all experiments, with the detailed results displayed in Table 5. The results indicate that for our experiments, the CO distance method with Resnet50 performed the best when dealing with 3 classes. CO is a technique that gauges the similarity between images by considering their angle similarities.

Distance	Backbone	*3-Class dataset*			*9-Class dataset*		
		Recall	**Precision**	**F1-Score**	**Recall**	**Precision**	**F1-Score**
EU	SqueezNet	95%	100%	98%	87%	91%	88%
	ResNet18	94%	100%	97%	95%	96%	95%
	ResNet50	97%	100%	98%	**98**%	100%	98%
KL	SqueezNet	94%	100%	97%	92%	96%	94%
	ResNet18	95%	99%	97%	94%	97%	96%
	ResNet50	89%	100%	94%	96%	97%	96%
IS	SqueezNet	94%	100%	97%	91%	98%	94%
	ResNet18	88%	100%	93%	95%	98%	96%
	ResNet50	97%	99%	98%	94%	100%	96%
MA	SqueezNet	94%	100%	97%	95%	98%	96%
	ResNet18	94%	99%	96%	96%	98%	97%
	ResNet50	96%	99%	97%	96%	98%	97%
WA	SqueezNet	94%	99%	97%	94%	98%	96%
	ResNet18	91%	100%	95%	90%	94%	92%
	ResNet50	95%	100%	98%	95%	96%	95%
CO	SqueezNet	79%	100%	88%	91%	93%	92%
	ResNet18	92%	100%	96%	94%	98%	96%
	ResNet50	**98**%	100%	99%	95%	100%	97%

Table 3. The table displays evaluation metrics for Aphid. On the left side, it highlights the 3-class dataset with 3 Ways, 5 Shots, and 9 Query Samples. We employed ResNet18, SqueezeNet, and ResNet50 as the backbone of our prototypical networks, assessing various distance methods. On the right side, the table showcases evaluation metrics for the 9-class dataset with 9 Ways, 5 Shots, and 9 Query Samples, also utilizing ResNet18, SqueezNet, and ResNet50 as the backbone of our prototypical networks. For the 9-class dataset, "Recall" represents the Weighted Average Recall Value for Aphid Classes. The datasets used are detailed in Table 1 and Table 2.

Distance	Backbone	*3-Class dataset*			*9-Class dataset*		
		Recall	**Precision**	**F1-Score**	**Recall**	**Precision**	**F1-Score**
EU	SqueezNet	97%	98%	97%	91%	91%	91%
	ResNet18	97%	98%	97%	95%	96%	95%
	ResNet50	99%	99%	99%	97%	97%	97%
KL	SqueezNet	97%	97%	97%	93%	93%	93%
	ResNet18	97%	97%	97%	96%	96%	96%
	ResNet50	95%	96%	95%	96%	96%	96%
IS	SqueezNet	97%	97%	97%	95%	96%	95%
	ResNet18	95%	95%	95%	95%	95%	95%
	ResNet50	97%	97%	97%	97%	97%	97%
MA	SqueezNet	94%	94%	94%	94%	95%	95%
	ResNet18	97%	97%	97%	97%	97%	97%
	ResNet50	97%	97%	97%	**98**%	97%	97%
WA	SqueezNet	97%	97%	97%	95%	96%	95%
	ResNet18	96%	96%	96%	93%	93%	93%
	ResNet50	97%	97%	97%	96%	96%	96%
CO	SqueezNet	91%	93%	91%	92%	93%	92%
	ResNet18	96%	96%	96%	96%	97%	96%
	ResNet50	**99**%	99%	99%	**98**%	98%	98%

Table 4. The table displays evaluation metrics for all our insect classes. On the left side, it highlights the 3-class dataset with 3 Ways, 5 Shots, and 9 Query Samples. We employed ResNet18, SqueezeNet, and ResNet50 as the backbone of our prototypical networks, assessing various distance methods. On the right side, the table showcases evaluation metrics for the 9-class dataset with 9 Ways, 5 Shots, and 9 Query Samples, also utilizing ResNet18, SqueezNet, and ResNet50 as the backbone of our prototypical networks. For the 9-class dataset, "Recall" represents the Weighted Average Recall Value for Aphid Classes. The datasets used are detailed in Table 1 and Table 2.

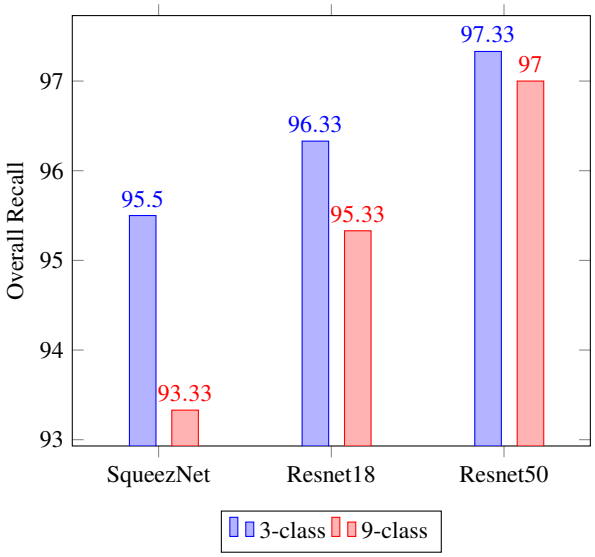

Fig. 7. This bar chart shows how well each CNN performs in terms of average overall recall across six different distance methods. The analysis covers both the 3-class and 9-class datasets.

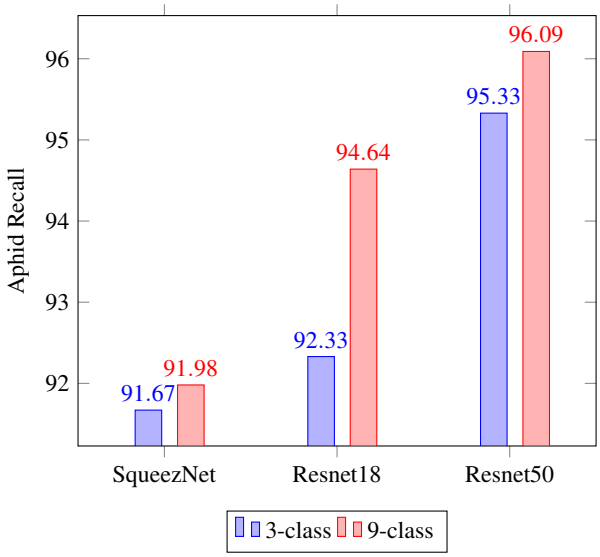

Fig. 8. This bar chart shows how well each CNN performs in terms of average aphid recall across six different distance methods. The analysis covers both the 3-class and 9-class datasets.

5.4 Best methods for each experiment

As our results exhibit minimal differences between the top two performers in each experiment, we have chosen to present the two best distance methods for each scenario in Table 5.

As demonstrated in the comparison, the EU method appears 5 times, while the MA method appears 3 times. This suggests that these two methods consistently deliver reliable results across all our experiments. Considering the additional computational resources required by the MA method, it implies that, for aphid classification, the EU method is likely more dependable compared to other methods.

5.5 Comparing models

In this section, we aim to compare the average recall values for the "aphid" class in each experiment, as well as the overall average recall for each experiment. This comparison allows us to assess the performance of different CNNs and determine which experiments yield superior results. Additionally, we can discern whether the 3-class or 9-class experiments demonstrate better outcomes through this analysis.

We employ two charts to present and compare the data from Table 3 more effectively. As depicted in Fig 7, we draw two key conclusions regarding overall recall. Firstly, larger models exhibit superior recall, benefiting from their ability to extract a greater number of features. Secondly, experiments involving 3 classes consistently outperform those using a 9-class dataset in terms of overall performance. It shows that in our experiments, the dataset with fewer classes can be classified better by few-shot learning.

In Fig 8, which is the second chart, we present the recall values for aphids in each CNN individually, considering both the 3-class and 9-class datasets. In contrast to the earlier chart, we observe that the recall values are higher for the 9-class dataset compared to the 3-class dataset. Notably, in our experiments, the larger model demonstrates superior results, which could be due to its capacity to extract more features and contribute to improved performance.

6 Discussion

In our study, we tackled the significant challenge of recognizing aphid images with limited samples, employing a learning approach called prototypical networks. With a primary emphasis on achieving a high rate of positive instances of aphid detection, our main goal was to attain the highest recall values in our research. To assess performance, we employed 6 distinct distance methods in combination with 3 different CNN architectures. We achieved notable success in our study, attaining recall values of 98% for aphid classification and an impressive 99% for overall recall in insect classification. This achievement is attributed to the incorporation of Cosine similarity, which serves as our scientific contribution to the 3-class dataset. Our model, leveraging a modified ResNet50 as its backbone, played a crucial role in obtaining these commendable results.

In agriculture, identifying insect pests, particularly various types of aphids, is a crucial challenge that significantly impacts the economy[4]. Swift and precise visual recognition is essential for practical applications to manage infestations in crops effectively. Through various experiments in this research, while the best result is related to Cosine similarity, we observed that both Euclidean distance and Mahalanobis divergence prove to be reliable methods. However, when considering computational resources, the Euclidean distance is more efficient. When compared to one of our primary references [10], where the KL method was identified as the optimal approach, we observed contrasting results. This divergence could be attributed to disparities in both the quality and quantity of input data. In the insect dataset IP102 [10], images showcase backgrounds of leaves or grass. It is worth noting that our dataset features a background of sticky plates. Additionally, the IP102 dataset employs a larger number of images compared to the datasets used in our experiments.

Additionally, larger CNNs extract more features, so they have the potential to result in better outcomes. On the flip side, larger CNNs demand more resources, especially in terms of GPU usage.

This research explored the impact of employing two different datasets: a 9-class dataset (Table 1) that included two aphid classes, 6 different insect classes, and a background class. Additionally, we utilized a 3-class dataset (Table 2) where aphids constituted one class, other insects formed the second class, and the third class represented

Backbone	*3-Class dataset*				*9-Class dataset*			
	Distances	**Recall**	**Precision**	**F1-Score**	**Distances**	**Recall**	**Precision**	**F1-Score**
SqueezNet	EU	95%	100%	98%	MA	95%	98%	96%
	KL	94%	100%	97%	WA	94%	98%	96%
ResNet18	KL	95%	99%	97%	MA	96%	98%	97%
	EU	94%	100%	97%	EU	95%	96%	95%
ResNet50	**CO**	**98%**	**100%**	**99%**	**EU**	**98%**	**100%**	**98%**
	EU	97%	100%	98%	MA	96%	98%	97%

Table 5. This table displays the two best distance methods for each experiment. Recall value for aphids is presented since it is our primary research focus. For the 9-class scenario, we have included the weighted average of both aphid classes in this table. The datasets used are detailed in Table 1 and Table 2.

the background. The introduction of the background class was necessary due to our outdoor camera setup, which could be triggered by water drops.

In the course of our experiments, a noteworthy observation emerged: the detection of aphids using the 9-class dataset yielded superior results (Fig 8). This improvement could be attributed to the utilization of 14 images for aphids in the 3-class dataset, aligning with the central concept of few-shot learning that advocates using a limited number of samples. In contrast, the 9-class dataset, featuring two aphid classes, allowed us to employ 28 images. This increase in the number of images provided a richer set of features for comparison when utilizing our chosen CNN architectures.

7 Conclusions

In our study, we delved into the realm of insect pest image recognition, grappling with limited samples. Our experiments encompassed two distinct dataset compositions, revealing the effectiveness of the cosine similarity distance method, as one of our scientific contributions, boasting an impressive 98% recall for aphid detection in the 3-class dataset paired with ResNet50. Across all 36 training iterations, we observed consistently reliable results, hovering around the 95% recall mark.

Among the tested distance methods, while the best result is related to Cosine similarity, Euclidean distance, and Mahalanobis divergence emerged as trustworthy options. However, the former stood out as the preferred choice due to its efficiency in computational resources. In essence, we can rely on prototypical networks as a dependable few-shot learning method. This method achieves 98% of recall in aphid detection, requiring less data and fewer resources for training.

8 Future Works

For future directions, we can delve into studies focusing on the recognition of various types of aphids[4]. This exploration is crucial due to the adverse impacts of different aphid species on agricultural economies. Technically, using various mean functions instead of the ordinal mean can impact and potentially change results. We used the mean function to create prototypes, but exploring alternatives like Weighted Mean or Geometric Mean could yield improved outcomes. Testing different mean functions allows us to assess their impact and enhance overall results. Furthermore, investigating the possibility of altering the CNN architecture and optimizing it presents a promising direction for future research. Specifically, exploring smaller backbones is advantageous as it requires fewer computational resources. Identifying a small yet effective CNN as the backbone for prototypical networks could prove to be valuable.

Acknowledgements

- This project is financially supported by ELFPO and performed within the POP3+ Fryslân project Innovatie luizendetectie.

References

[1] Jiachen Yang Yang Li. Few-shot cotton pest recognition and terminal realization. *Computers and Electronics in Agriculture 169 105240*, 2020.

[2] Zhankui Yang Ming Li Chuanheng Sun Xinting Yang Wenyong Li, Tengfei Zheng. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. *Ecological Informatics 66 101460*, 2021.

[3] Brian McCornack Ivan Grijalva, Brian J. Spiesman. Computer vision model for sorghum aphid detection using deep learning. *Journal of Agriculture and Food Research 13 100652*, 2023.

[4] Sebastian Thiel Tim Mark Ziesche Philipp Batz, Torsten Will and Christoph Joachim. From identification to forecasting: the potential of image recognition and artificial intelligence for aphid pest monitoring. *Frontiers in Plant Science*, 2021.

[5] Tianhai Wang Hongkun Tian. Computer vision technology in agricultural automation —a review. *INFORMATION PROCESSING IN AGRICULTURE. 2020; 7, 1– 19*, 2020.

[6] J. Alex Thomasson Wei Chen Raghupathy Karthikeyan Guangzhao Tian Yeyin Shi Changying Ji Qiong Su Weiyue Xu, Tao Xu. A lightweight ssv2-yolo based model for detection of sugarcane aphids in unstructured natural environments. *Computers and Electronics in Agriculture 211 107961*, 2023.

[7] Maya Aghaei Gavari Fredrik-Otto Lautenbag, Henry Maathuis. Aphid recognition using multi-scale feature representations with vision transformers. *NHL Stenden computer vision data cience*, 2022.

[8] Xi Fu Sunao Ochi Takehiko Yamanaka Jianqiang Sun, Wei Cao. Few-shot learning for plant disease recognition: A review. *Agronomy Journal. 2023; 1–13*, 2022.

[9] Vanessa Aparecida Karla Rejane Hemerson Pistori João Vitor, Arlinda Cantero. Usage of few-shot learning and meta-learning in agriculture: A literature review. *Smart Agricultural Technology. 2023; 5,100307*, 2023.

[10] Díbio L. Borges Jacó C. Gomes. Insect pest image recognition: A few-shot machine learning approach including maturity stages classification. *Agronomy. 2022; 12,1733*, 2022.

[11] Richard Zemel Jake Snell, Kevin Swersky. Prototypical networks for few-shot learning. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*, 2017.

[12] Guosheng Lin Chunhua Shen Chi Zhang, Yujun Cai. Deep emd few-shot image classification with differentiable earth mover's distance and structured classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence. 2023; 5, 45*, 2023.

[13] Nikos Komodakis Spyros Gidaris. Dynamic few-shot visual learning without forgetting. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2018; 4367-4375*, 2018.