Radboud University Nijmegen



and



DIOPSIS

Faculty of Science

Semi-Supervised Hierarchical Classification for Insect Monitoring

an explorative study

Msc Thesis Computing Science Data Science

External Supervisor:
dr. Jacob Kamminga

Internal Supervisor:
dr. ir. Tom Claassen

Second reader:
dr. Inge Wortel

Author:
Heleen Visserman Msc
s1067406

September 24, 2024

Abstract

Insect populations play a crucial role in our ecosystem, yet these animals are facing a 76% decline in biomass over the past 30 years [15], making effective monitoring essential to mitigate this loss. Traditional monitoring methods are labour-intensive and rely heavily on expert knowledge, prompting the need for automated solutions that leverage computer vision and deep learning. These technologies, when combined with the hierarchical nature of insect taxonomy, offer a promising alternative for insect detection and classification. However, challenges such as limited datasets and classification uncertainty, especially for fine-grained species and under suboptimal image quality, remain. In this study, we propose an EfficientNet-based model with customised classification layers to address these issues. We compare a multi-target loss function with a hierarchical loss function to investigate the impact of a hierarchical loss function on the severity of errors made by the classification models. Furthermore, we assess the performance of semi-supervised learning (SSL) versus traditional training methods to see the impact on model performance, when the available unlabelled data is leveraged. Contrary to expectations, our results show that models trained with the traditional method achieve higher F1 scores than models trained with the SSL method, particularly at higher taxonomic levels. For example, at Class level, SSL methods obtain F1 scores of 0.36 and 0.43, while the base models obtain F1 scores of 0.71 and 0.76. While both traditionally trained models perform well for labels with abundant data, with F1 scores of 0.93 and 0.95, the F1 scores drop to 0.21 and 0.27 for taxa with less than 100 data points in the dataset. Recall seems to be the the limiting factor for these F1 scores. We discuss the results of our experiments by exploring key aspects of insect classification and how they impact the obtained results. These key factors include data imbalance, image quality, and hierarchical structure. We conclude that, while semi-supervised learning shows theoretical potential for hierarchical insect classification, especially in dealing with limited labelled data, it is less effective in this study. To further progress research in the field of hierarchical insect classification, future work should take research into one of the following directions: increase the data quantity, tackle class imbalance, simplify model structure, create hierarchical models, tackle fine-grained classification or explore other SSL methods.

Keywords— Insect Classification, Hierarchical Classification, Semi-Supervised Learning, Computer Vision

Contents

1 Introduction **3**

2 Background and Related Work **4**
- 2.1 Insect Classification . . . 4
- 2.2 Hierarchical Classification . . . 5
- 2.3 Out of Distribution Detection . . . 6
- 2.4 Semi Supervised Learning . . . 6

3 Data **7**
- 3.1 Labelled Data Collection . . . 7
- 3.2 Unlabelled Data Collection . . . 8
- 3.3 Data Labels . . . 8
- 3.4 Data Preprocessing . . . 10

4 Methods **10**
- 4.1 Base Model . . . 12
- 4.2 Base Training . . . 14
- 4.3 SSL Training . . . 14
- 4.4 Evaluation . . . 15
- 4.5 Materials and Equipment . . . 16

5 Results **16**
- 5.1 Quantitative Results . . . 16
- 5.2 Qualitative Results . . . 17

6 Discussion **18**
- 6.1 Experiment Results . . . 19
- 6.2 Future Work . . . 23

7 Conclusion **24**

8 Acknowledgements **25**

References **25**

A Ontology **29**

B Out of Distribution Measure **32**

C Annotation Environment Non-Experts **32**

1 Introduction

Insects are essential for the thriving of our ecosystems. They pollinate plants and serve a critical role in the food chain [20]. However, recent studies have shown a concerning decline in insect diversity and abundance [51, 38, 15]. Specifically, a decline of 76% in flying insect biomass has been estimated in an insect monitoring study from 1989 - 2016 in Germany [15]. To understand the scope of this shift, effective methods to monitor insect populations over time are needed [20, 54, 28].

Manual insect population monitoring is a highly costly process. For example, the process itself is time intensive [41] , as each trap needs to be regularly cleared to get insight in the populations over time. Moreover, the identification of insect species largely needs to be performed by expert entomologists, as the species can be remarkably challenging to differentiate, especially species that fall under the same genus or family [57]. Unfortunately, entomologists are scarce and only have limited available time [56]. Hence, automated devices that make use of cameras and deep learning that capture the knowledge of these experts to detect and classify insects, are revolutionising insect monitoring [50, 43, 4]. Especially computer vision methods are shown to be effective for both images taken on traps [5, 20] and images taken in insects' natural environment (*in situ*) [4, 6, 31].

However, even with automated devices, insect monitoring, and specifically classification, is not a trivial problem. Most monitoring devices in the field have limited resources to make them more energy efficient, which causes image quality to be suboptimal. Moreover, insects are small and occupy only a small portion of these full images, making it even more difficult to confidently differentiate between these fine-grained species. Fortunately, the taxonomy of insects, and animals in general, is structured in a hierarchical manner and predictions at higher level in the taxonomy are generally easier to make. For example, as can be seen in Figure 1, it is easier to distinguish between two insects of different Orders (e.g. fly or butterfly) than between two insect Families that fall under the same Order (e.g. large crane flies and limoniid crane flies). Therefore, leveraging the taxonomic hierarchy in automated insect classification can be used as a tool to address classification uncertainty caused by the finegrainednes.

Another advantage of leveraging the taxonomic hierarchy in insect classification is that it allows models to make partially correct predictions instead of always being completely wrong. For example, in Figure 1, if the true label of an image is large crane flies ('Tipulidae'), it is less problematic to misclassify the insect as belonging to the broader Order of flies ('Diptera') than to confuse it with an entirely different Family, such as the plume moths ('Pterophoridae').

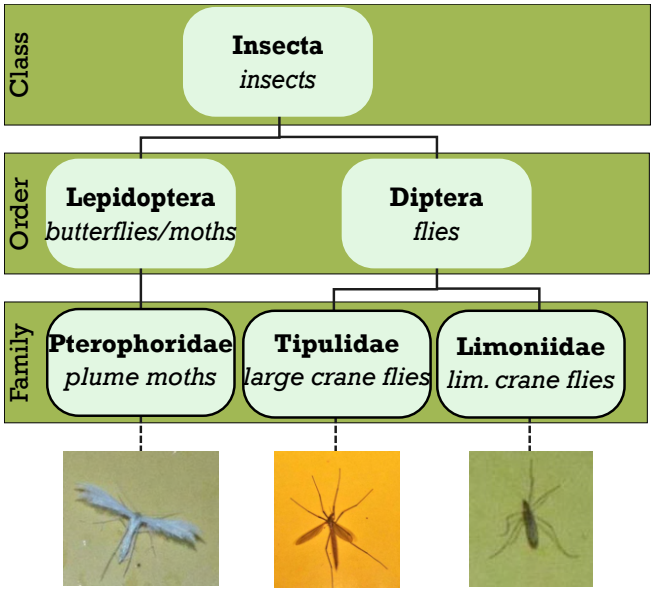


Figure 1: An example of taxonomic classification of three different families of insects 'Pterophoridae', 'Tipulidae' and 'Limoniidae'. The latter two fall under the same order 'Diptera' and are morphologically similar and hard to distinguish, whereas the first two are from different orders and thus easier to distinguish. The figure also shows the order and class under which the insect families fall.

The severity of these errors can be measured using the Taxon Distance [19]. In insect monitoring, the goal is to classify insects at the lowest possible taxonomic level, with more specific identifications providing deeper insights into the local insect community. However, even identifying insects at the family level, rather than the species level, can still offer valuable ecological insights. In an ideal scenario, a well-performing model would achieve high precision and recall at lower taxonomic levels, ensuring accurate species identification. However, sometimes only higher level predictions are possible due to classification uncertainty.

Another cause for the classification uncertainty is the fact that more than 1.02 million insect species have been described [59], making it is unfeasible to collect enough data to train models that can classify all insect species. Hence, if an insect classification model is applied in the wild, it is likely to come across insects that are not present in its training dataset. Moreover, many insects need to be dissected or genetically analysed in order to classify them at lower taxonomic levels [4], making classification at lower taxonomic levels impossible for some species. Using the hierarchical taxonomy, these unseen insect species can still have a correct higher level label. Out-of-distribution (OOD) detection technologies can ensure that previously unseen species or species for which image based species identification is impossible,

are flagged as uncertain at lower taxonomic levels. Thus, OOD detection technologies can prevent incorrect predictions at the species level and allow correct predictions at the appropriate higher taxonomic level.

In order to optimise performance of the deep learning models used for computer vision, generally a large amount of labelled data is required. Unfortunately, there is a shortage of public insect classification datasets that are diverse, labelled and of sufficient size [43]. This is caused by the fact that, as previously mentioned, annotation of insects at lower taxonomic levels is often too expensive [57]. Furthermore, due to the diversity and varying abundance of insect species, insect datasets are severely imbalanced, causing some insects to only have a few data points in these datasets and others to have an excessive amount. This shortage of complete datasets negatively impacts the generalisability of deep learning models, especially in datasets with many classes [43], which underlines the challenging nature of insect classification.

A common method to overcome the lack of annotated training data is the branch of machine learning called semi-supervised learning [25, 13]. This method leverages the availability of unlabelled data, on top of labelled data. Which is advantageous, as acquiring unlabelled data generally results in little to no additional cost [13]. This technique has previously been applied to the separate problems of insect pest monitoring [37] and hierarchical classification applied to different domains [13], but, to our knowledge, never specifically for hierarchical insect classification.

To address the challenges of incomplete annotations in insect monitoring datasets, our research applies hierarchical classification combined with semi-supervised learning. We investigate how semi-supervised learning can enhance the performance of insect classification when using a hierarchical model and an incomplete dataset. Our approach involves first developing a hierarchical classification model and then training it within a semi-supervised learning framework, integrating both labelled and unlabelled data.

This study not only sheds light on the specific challenges of hierarchical insect classification posed by incomplete datasets and their effects on model performance but also addresses the difficulties associated with semi-supervised learning. We explore how to effectively use the hierarchical structure of taxonomic data to make predictions at higher taxonomic levels. Our findings offer recommendations for future research, focusing on improving data quantity and model structure.

The rest of the thesis is structured as follows: Section 2 highlights background information and related previous work. The dataset and methods used are described in, respectively, Section 3 and 4. Section 5 describes the results of these experiments, which are discussed in Section 6. Finally, Section 7 concludes our research. Additional information can be found in the Appendices.

2 Background and Related Work

This section outlines different approaches that use deep learning for the classification of insects. The concept of *image classification* is introduced, and approaches applied to insect classification. Furthermore, *hierarchical classification* approaches are introduced, including approaches applied to different domains, to model the hierarchical nature insect taxonomy. *Out of distribution (OOD) detection* is introduced to identify images that are of too low quality to be classified at a lower taxonomic level, or to classify previously unseen insects at a higher taxonomic level. Finally, *semi supervised learning* is introduced as an approach to leverage the unlabelled data in the training process of insect classification models.

2.1 Insect Classification

Image classification is a key task in computer vision and machine learning that involves sorting images into predefined categories or classes [14]. Traditionally, image classification is performed as a flat, single target and supervised classification task [6, 22, 57]. Which means that, respectively, the deep learning model does not explicitly incorporate the hierarchical structure of the taxonomy, each example in the dataset only has a single target class and the model is trained on a fully annotated dataset.

Convolutional neural networks (CNNs) are a commonly used deep learning technique for image classification [57, 43]. Different forms of CNNs are used for insect classification. Some papers make use of self-designed CNNs [46, 22, 55, 36], however, most insect classification techniques make use of pre-trained image classification models, which are fine-tuned to fit the purpose of insect classification [43, 4, 33, 45]. The most commonly used pre-trained models include EfficientNet [42, 4] and ResNet [16, 7, 44, 1]. These models have been trained on large image classification datasets like ImageNet [9] and MS COCO [27]. The use of these pre-trained networks allows a model to converge more quickly, as it has already learned to extract basic features.

Image based insect identification can be performed performed in situ (in field) [4, 6, 31] or on traps [5, 20, 36]. When working with in situ images, the images are generally taken of insects on plants. Camera trap images are more standardised, as all images look very similar and have the same background, which reduces the interference of the varying background with the performance of the model [43].

Most insect classification papers focus on the task of insect pest detection [37, 32, 26], which focuses on the detection and identification of a few specific insect species that are harmful for crop yield, however, for insect population monitoring the more labels one can distinguish the better insight one gets in the insect population. Unfortunately, the more labels a model has to distinguish, the more complex the classification task and the lower the generalisation performance [43]. Hence, insect population monitoring can be seen as a challenging problem, however since both pest detection and insect monitoring focus on insect classification we can still draw inspiration from the type of models and preprocessing used by pest detection papers.

In this paper we draw inspiration from previous work by using a pre-trained CNN and fine-tuning it for the problem of insect classification on traps. We adjust this model to tackle insect classification as a hierarchical, multi target problem, and approach it as a semi-supervised classification task.

2.2 Hierarchical Classification

The taxonomy of insect species is shaped like a tree going from the broad Kingdom level at the root to the specific Species level at the bottom, passing by several taxonomic levels. Each level down contains a more specific annotation of the animal that is being identified. An example of the taxonomic levels of a specific species of moths can be found in Figure 2. The hierarchical nature of the taxonomic labels inherently makes insect classification a multi-target problem, as a single animal has a label at each taxonomic level. However, even though each animal has a label at each taxonomic level, sometimes assigning this label at lower levels is too difficult or even impossible due to the classification uncertainty. Hence, the labels for insect classification are assigned at the deepest taxonomic level for which the prediction is still confident. This deepest label can be seen as a path through the taxonomy from the root node 'Animalia' to the deepest assigned label, for example 'Lomaspillis Marginata' at Species level. Where previously mentioned works treat insect classification as a form of flat classification [6, 36], there has also been previous work that explicitly incorporates the hierarchical nature of its data in the classification process [4, 30]. The latter is referred to as hierarchical classification.

Hierarchical classification is applied in several different domains [39], among which insects [4, 30, 48], birds [23], other organisms [3], e-commerce [12] and general image classification [34]. Hierarchical classification approaches broadly fall under one of two main approaches: local and global classifiers [39]. Local classifiers employ a set of local classifiers that make predictions per individual node, per parent node [48] or per taxonomic level [4, 30, 13]. Whereas global classifiers consider the entire class hierarchy at once [34]. Actually incorporating the label hierarchy in the classification process can be done in several different ways: in the model structure, in the loss function or by creating hierarchical embeddings of the images [3].

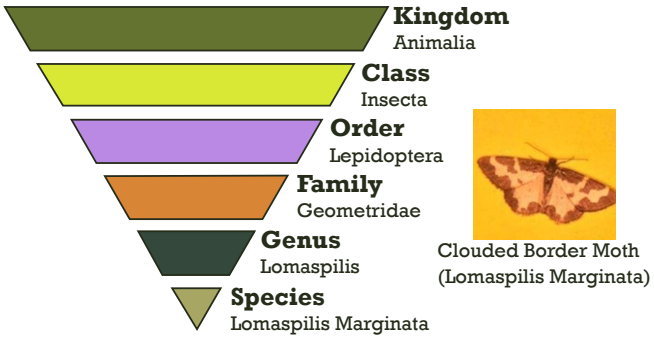


Figure 2: Example of a single insect having a label at six taxonomic levels, showing how insect classification is a hierarchical multi-target classification problem.

Local hierarchical classifiers can tackle the classification problem at multiple parts of the taxonomy. Including classifiers for each individual node, each parent node, or each taxonomic level. For instance, Ong and Hamid[30] use hierarchical classification to classify images of museum insect specimen into the correct taxonomic labels, by training a different CNN for each taxonomic level. They find that a single deep learning architecture is not robust enough to classify different taxonomic levels of specimens. They do not explicitly check for prediction consistency between levels, in contrast to Bjerge et al. [4], who make use of an explicit dependency loss. Bjerge et al. use a fixed CNN feature extractor and for each taxonomic level create two fully connected layers as classification layers, as a form of multitask learning. For the CNN feature extractor they use pre-trained CNNs (ResNet50 [16] and EfficentNetB3 [42]). To take label consistency into account they make use of level wise cross entropy loss in combination with a dependency loss, which checks whether prediction at level l is consistent with the prediction at the level above.

Another paper that incorporates the hierarchical structure in the loss function is the paper by Gao et al. [12]. They propose a hierarchical loss function that performs cross entropy loss on each hierarchical level and then they use a dependency loss function to incorporate the class consistency constraint, for category prediction in e-commerce systems. They also explicitly incorporate the hierarchy in the model structure by concatenating the output of the model at level l to the output of the model at the level $l + 1$ below. Bertinetto et al. [3] also propose a hierarchical loss function that performs cross entropy on each taxonomic level and then they scale the loss for each layer with a variable that punishes higher level errors

worse than lower level errors.

Tresson et al. use a local classifier per parent node for the task of arthropod identification [48]. They include the hierarchy explicitly in the model structure, by using a two-step approach consisting of a super-class detection model based on YOLOv3 [35] and a detailed classification model that classifies the crops into species classes based on the found super-class. Garg et al. [13] also incorporate the hierarchy explicitly in the model structure and similar to Bjerge et al. use a feature extractor that feeds features into several different classifiers for each hierarchical level.

Redmon and Farhadi make use of a global classifier for the task of image detection and classification [34]. They create a hierarchical label structure using WordNet, similar to Garg et al [13]. Which allows them to combine two image datasets, MS COCO [27] and ImageNet [9] for model training, by creating a hierarchical label structure that covers the labels in both datasets. In the end they make leaf node predictions which are propagated back up the hierarchical tree based on conditional probabilities.

In this paper we tackle insect classification as a hierarchical classification task. In this paper we will build upon previous work by using a level wise loss function for our classification model to directly incorporate the hierarchy in our classification approach. We compare this hierarchical approach over a classifier per node approach to see the impact of the different loss functions.

2.3 Out of Distribution Detection

To cover this hierarchical uncertainty and the possibility of coming across previously unseen species, it is crucial to effectively manage model uncertainty. Specifically, hierarchical insect classification gets incrementally more challenging with each lower taxonomic level, hence it is necessary for the model to be able to refrain from predictions at lower levels if the species are too morphologically similar. Similarly, if an unknown insect species is classified, a hierarchical model could have high uncertainty at species level, but could make a more confident prediction at family level. Out-of-distribution (OOD) detection is the field of machine learning that can help to identify when uncertainty is too high, by identifying patterns in data that deviate from expected norms based on past observations [4].

In recent studies, various approaches have been explored to address uncertainty and OOD detection in hierarchical classification tasks. Bjerge et al. introduce threshold-based anomaly tagging, where they model the output of neural network logits for each class in the training set as a normal distribution [4]. By defining a threshold based on the mean and standard deviation of correctly classified samples, their model can identify and reject outlier samples that are likely to be misclassified, thereby improving robustness in uncertain scenarios. However, these types of approaches first need to model the normal behaviour and then identify the outliers later [53]. Van Osta et al., on the other hand, implement a classification-with-rejection approach to reject predictions for bird audio frames that are too uncertain [52]. They utilise the output from the Sigmoid layer of their CNN, setting a threshold value to determine outliers. Given the fact that a Sigmoid output of 0.5 indicates maximum uncertainty, the authors classify a prediction as an outlier, when the model's output is below the sum of 0.5 and a determined threshold value.

Maximum Softmax Probability (MSP) is a similar technique to van Osta's technique, however it works for multi-class classification, as opposed to binary classification. MSP is an anomaly detection technique as proposed by hendrycks et al.[17] and used for insect classification by Rustia et al. [36]. MSP is a threshold based classification with rejection OOD detection approach. It allows a model to reject a prediction when the Softmax probability is below a certain threshold, indicating high uncertainty. Moreover, it is straightforward to implement and computationally efficient [11], making it a robust starting point for uncertainty management, balancing simplicity with effectiveness.

In this research we use Maximum Softmax Probability because it efficiently detects out-of-distribution samples by rejecting low-confidence predictions, ensuring accurate and robust classifications in hierarchical insect classification.

2.4 Semi Supervised Learning

Given the complexity of insect classification, which requires expert annotation at lower taxonomic levels, leveraging the abundance of unlabelled data through semi-supervised learning (SSL) offers a cost-effective solution. SSL is a branch of machine learning that combines unsupervised learning and supervised learning, enhancing the performance of one by utilising the information provided by the other [49]. Specifically, it utilises unlabelled data to improve the generalisation performance of a model [24]. It does this based on the assumption that the underlying data distribution of the labelled and unlabelled data is virtually identical.

Specifically, semi-supervised learning is based on the following three main assumptions [49]:

1. **Smoothness assumption**: If two input points are close in input space, their corresponding labels should be the same. Thus if x_1 is close to x_2 and x_2 is close to x_3 then x_1 and x_3 are expected to have the same label.

2. **Low-Density assumption**: The decision boundary of a classifier should pass through

low-density regions of the input space, reducing the likelihood of crossing densely populated areas, where data points are more likely to share the same label.

3. **Manifold assumption**: High-dimensional data often lie on low-dimensional manifolds within the input space, and points on the same manifold are assumed to share the same label.

A straightforward branch of SSL is self-training, which is based on the low-density assumption. It involves assigning labels to unlabelled data based on the predictions of an initial base model trained on labelled data [49]. These pseudo-labels are then used to iteratively completely retrain the base model, incorporating both the original labelled data and the most confident pseudo-labelled data. This approach assumes that the model's predictions are sufficiently confident to be used as true labels in subsequent training rounds. Yarowsky et al. successfully apply self-training to sense disambiguation in natural language processing, demonstrating that semi-supervised models can achieve performance levels comparable to supervised models that require extensive hand annotations [58].

However, the fact that the base learners in self-training have to be retrained completely in each self-training iteration can be computationally intensive. Pseudo-labelling is a technique similar to self-training, however the model is now fine-tuned instead of completely retrained, making it computationally less intense. Pseudo-labelling is effectively used in image classification [24] and insect classification [37]. In their paper, Rustia et al. fine-tune an initial model using pseudo-labelled data, resulting in an adaptive model that improves classification performance over time [37]. The resulting model can be used for the automatic collection of pseudo-labelled insect pest classification training data. Lee, on the other hand, proposes a simple and efficient semi-supervised learning method called pseudo-label, where the base model is trained on both labelled and unlabelled data simultaneously [24]. Since the pseudo-labels predicted in earlier training stages are generally less reliable, the weights of the pseudo-labelled data is increased over time. Furthermore, they make use of an unsupervised pre-training phase where a denoising autoencoder is trained to create image embeddings.

Regularisation techniques are another branch of SSL that is based on the smoothness assumption. These techniques are founded on the idea that a model should produce similar outputs when its input is perturbed in minor ways. Examples of these techniques are FixMatch [40] and MixMatch [2]. Garg et al. apply both FixMatch and MixMatch in a hierarchical setting, by using samples labelled at coarser level as unlabelled data in a technique called HierMatch [13]. The approach involves having labelled and unlabelled data for each hierarchical level, with the feature extractor feeding into classifiers specific to each level. HierMatch demonstrates the potential of leveraging hierarchical information, reducing the need for fine-grained labels by 50% on datasets like CIFAR-100, with only a 0.59% drop in accuracy. They found that leveraging coarse labels from higher levels of the hierarchy can reduce the need for fine-grained labels.

In addressing the challenges of insect classification, we utilise a hierarchical base model similar to Bjerge et al. [4] and implement a straightforward yet effective pseudo-labelling strategy inspired by Lee[24] to investigate the model performance when utilising the available unlabelled data.

3 Data

To train our models we create two datasets: a dataset with labelled examples and a dataset with unlabelled examples. This section firstly describes how the images in both datasets are collected using the DIOPSIS cameras. Then it discusses how the hierarchical labels are assigned to each image in the labelled dataset and the distribution of those labels over the dataset. Lastly, it describes how the images and labels are then pre-processed to be used in the model training and evaluation process.

3.1 Labelled Data Collection

tThe labelled ground truth dataset used in this research is the DIOPSIS classification dataset [18]. It consists of 3.237 images shot with DIOPSIS cameras at different locations in the Netherlands[1] [21]. The DIOPSIS camera is an insect monitoring device that has a yellow non-sticky plate on which animals can land. Every few seconds a picture is taken of the screen. An example of such an image can be found in Figure 3. During the night the trap is lit to enhance image quality. Each DIOPSIS camera image can contain zero to hundreds of animals at a time. A large variety animals land on the DIOPSIS cameras, of which the majority falls under the taxonomic Class of insects, however, also animals that fall under the Classes of spiders, non winged arthropods and slugs/snails are part of the dataset and thus have been used as training data for the experiments in this thesis. In the rest of this paper we talk about insect classification as this is the main focus of the thesis, however we considered all the previously mentioned Classes of animals.

Each DIOPSIS image in the labelled dataset is annotated by experts by drawing bounding boxes around each individual animal and assigning labels to them. These bounding boxes are converted into crops of the DIOPSIS images, resulting into a total of 39.445 images of individually labelled insects and other

[1] https://diopsis.eu/

animals. A visual example of going from camera to crops can be found in Figure 3.

To evaluate the quality of the manually created corps, we inspect a random sample of example images of the labelled dataset which are displayed in Figure 4, and evaluate the average size of the crops. We find that the crops have a mean height and width of respectively 90.24 ± 74.97 pixels and 93.73 ± 76.35 pixels. The smallest image is 14×14 pixels and the biggest image 1521×1327 pixels. The median height and width are 75 and 78 pixels. This indicates that the crops are quite small and especially for small insects we see blurry images, as can be seen in the example images. Whereas with larger insects the image is quite clear. Another observation is that even though the images have the same background plate colour, lighting can impact the background colour of the images.

3.2 Unlabelled Data Collection

The unlabelled dataset images are collected from DIOPSIS cameras on five different locations in the Netherlands. Similarly to the ground truth dataset, bounding boxes are drawn on these images, however, instead of by human annotators, the bounding boxes are now drawn by an existing insect detection algorithm of Arise Biodiversity [2]. The images are then cropped according to these bounding boxes, like for the labelled images. This leads to a total of 31.706 unlabelled crops of individual animals. An overview of the amount of images per location can be found in Table 1.

Table 1: Overview of the number of unlabelled animal image crops collected per location using DIOPSIS cameras.

Location	Number of Crops
Dokkum	5508
Bildtpollen	12594
Lelystad	2410
Rottigemeente	2893
Vrouwenpolder	8301
Total	31706

To evaluate the quality of the automatically created corps, similarly to the labelled data, we inspect a random sample of example images of the unlabelled set, shown in Figure 5, and evaluate the average size of the crops. We find that the crops have a mean width and height of respectively 110.72 ± 46.95 pixels and 110.48 ± 47.26 pixels. The smallest image is 53×45 pixels and the biggest image 469×462 pixels. The median width and height are 86 and 86 pixels. Thus, the unlabelled images are on average a little bigger than the labelled images and the crops have less extreme size values, i.e. the smallest image size is bigger and the biggest image size is smaller. We also observe that the crops overall look quite similar to the labelled crops with respect to size area around the insect and that the automatic detection algorithm thus produces accurate crops.

[2] https://www.arise-biodiversity.nl/

3.3 Data Labels

The labels in the ground truth dataset are collected through expert annotations of the deepest confident taxon of the animal in the image. The images are annotated according to a predefined ontology. In total this ontology consists of 120 labels, each assigned to a specific level in the taxonomic hierarchy. The ontology has a tree shape where the root node is 'Animalia' at Kingdom level and the children of this level are at the taxonomic Class level, whose children are at Order level, etc.. Each taxonomic level has a number of labels associated with it. An overview of the number of labels per taxonomic level can be found in Table 2. Teixeira et al. found that many labels can harm classification performance [43]. Since we have a total of 120 labels, of which 92 have associated data with them, we have a large number of labels which makes the insect classification task for this dataset particularly challenging.

Table 2: Total number of taxa per taxonomic level in the ontology of the dataset.

Taxonomic Layer	Number of Labels
Kingdom	1
Class	4
Order	15
Family	59
Genus	25
Species	16
Total	120

Table 2 shows that there are more labels at the family level of the taxonomy, than there are labels at the genus and species level of the taxonomy. This makes it inherently impossible for a classification model to correctly classify the species for at least half of the family labels, since there are no species available in the ontology. This underlines the necessity for an uncertainty measure that allows a model to refrain from making a prediction at lower taxonomic levels.

The expert annotators assign labels to the images with the label at the deepest taxonomic level where they are still confident in their identification. This deepest label is then converted into a path through the ontology from the root node, the kingdom 'Animalia' to the assigned label. Due to the images of the insects being very fine grained and image quality not being optimal, it often happens that labels cannot be assigned deeper than family level. Figure 6 visualises

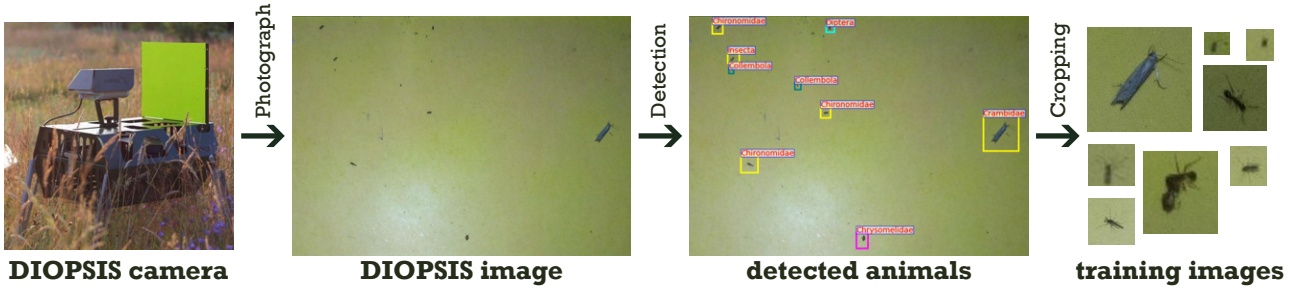


Figure 3: A visual overview of going from camera to training data. All images in the datasets are collected using DIOPSIS cameras. The cameras take pictures every few seconds. To convert these images to training images for our classification models, experts have annotated bounding boxes on the data for all insects. These bounding boxes are used to crop the detected insects out of the original image. These crops are then used as training images. For the unlabelled images the bounding boxes are acquired through an existing insect detection model.



Figure 4: Example crops of the labelled dataset.



Figure 5: Example crops of the unlabelled dataset.

the taxonomic levels and the amount of data points in the dataset with their deepest label at that level. This graph shows that 80% of the data is annotated at family level, and only about 1% of samples have the genus or species as deepest annotation level. This in combination with the lack of available labels at genus and species level in the taxonomy is a good example of the the incompleteness of insect classification datasets, which is a major challenge for insect classification research [43].

To get an overview of the amount of samples per label and the hierarchical structure of the labels we provide a visual overview of the ontology. A subset of this ontology, with number of data points per label, can be found in Figure 7. The full ontology can be found in Appendix A. Figure 7 gives us some interesting insights in the imbalance present in the dataset. First of all, there are two labels at family level that both make up a quarter of the total dataset's deepest labels: 'chironomidae' (non-biting mosquitos) and 'canidae' (mayflies) . Together they make up half of the dataset, highlighting the imbalance present in the dataset. Another interesting observation is that 98.1% of labels at the taxonomic Class level, falls under the 'Insecta' (Insects) label whereas 'Arachnida' (Spiders), 'Gastropoda' (Snails/slugs) and 'Entognatha' (wingless arthropods) only make up, respectively, 0.8%,0.6% and 0.5%of the data. Hence, the model may become biased towards the 'Insecta' label at the Class level.

For the task of insect monitoring, the distribution of the labels present in the dataset is expected to be representative of the distribution of insects in the wild. This way the model is likely to perform well in the wild. Therefore, we decide to keep the data imbalance as is, as insects that occur more often in the dataset are also expected to occur more often in the wild. In Section 5 we analyse the classification model's performance across head, mid, and tail labels to assess its impact on various taxa based on the quantity of training data available. To make the head, mid and tail split, labels with over 1,000 samples are categorised as head labels, those with 100 to 1,000 samples as mid labels, and those with fewer than 100 samples as tail labels. This gives the distribution of labels as can be seen in Table 3.

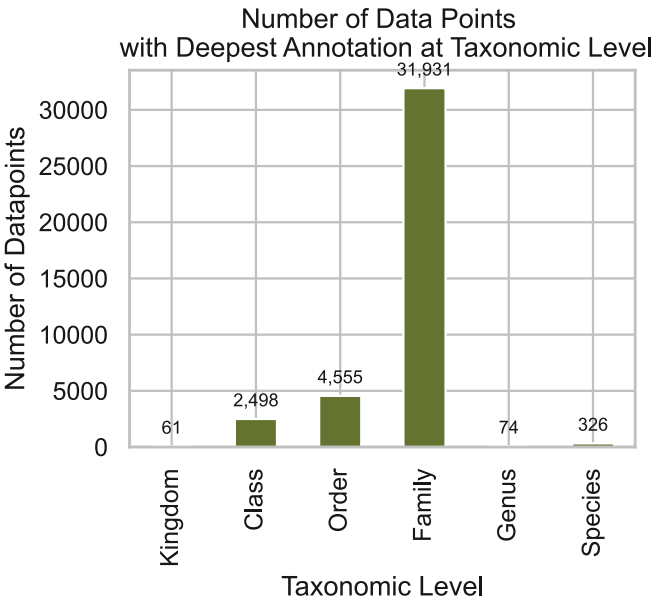


Figure 6: For each taxonomic level this figure shows the number of data points in the ground truth dataset that has its annotated deepest label taxon at this level.

Table 3: Number of labels per head, mid, tail split. Showing that only a few labels have over 1000 data points and that most labels in the ontology only have 0-100 data points in the dataset.

Split [#Data Points]	Number of Labels
Head [1000+]	10
Mid [100-1000]	28
Tail [1-100]	54
Empty [0]	28
Total	120

3.4 Data Preprocessing

Before the crops are fed into the classification model, both the labelled and the unlabelled images are preprocessed by resizing them to 256x256 pixels,the standard input size for the EfficientNet model, using Pytorch's `torchvision.transforms.v2.Resize()` class using the `BILINEAR` interpolation mode.

For the labelled dataset the deepest label is encoded into a path through the ontology from the root node 'Animalia' at the Kingdom level to the deepest assigned label. This makes sure that the parent nodes of the deepest label (and that parent's parent etc.) are also considered correct labels for the image. Specifically, we assign each individual label an id and a list of its ancestors. The list of ancestors consists of the list of label-ids of taxa encountered in the path back to the root node in the ontology, including the deepest label's id. If the label does not have an ancestor at a certain level, e.g. a deepest label at family level will not have ancestors at genus or species level, the ancestor label-id is set to -1 for that level. The list of ancestors is then converted into a onehot encoding for each taxonomic level. If the level of the deepest label is only at family level, the onehot encodings of species and genus levels will be all zeros. An example of a label encoding can be found in Figure 8.

Commonly in machine learning research a train-test split is made of the labelled dataset in order to be able to test the generalisation performance of a classification model on unseen data. However, since our dataset has labels with very few data points, we decide to use 5-folds cross validation for the training process. This way we can ensure that each sample gets used at least once in the test set. By taking the average performance over the different fold's test sets we get a good approximation of the generalisation performance of our model, while not having to give up valuable data points to a separate test set.

We split the data over five fold splits for cross validation. To do this we make make use of the `sklearn.model_selection.KFold` object, with the shuffle variable enabled. This shuffles the data indices and randomly splits all of the indices over the train and test splits. There are labels that have less than 5 samples in the dataset. For these labels we still divide them evenly over the folds. This means that some fold's test split will not contain this label and for that fold the model will not test performance on that label. However, since we are working with 5 folds, the data will always be used for testing at least once and thus we will still get an average generalisation performance for that label over the folds.

For the semi-supervised training method the labelled and unlabelled data are combined in the cross validation folds. Each fold retains the same train and test data from the labelled dataset, but now also has access to the full unlabelled dataset.

4 Methods

To investigate to what extent semi-supervised learning can enhance the performance of hierarchical insect classification, we develop a base model for which we compare two different loss functions, a multi-target loss and a hierarchical loss. Furthermore, we implement a threshold-based out of distribution measure to check for high uncertainty in model predictions. Then we develop two training methods, a basic cross validation training method with only the labelled training data and a semi-supervised training method to leverage the unlabelled and labelled data together. Lastly, we choose evaluation metrics that focus on the standard image classification metrics: F1 score, precision and recall and metrics that focus on the qualitative evaluation of hierarchical performance: the Taxon Distance.

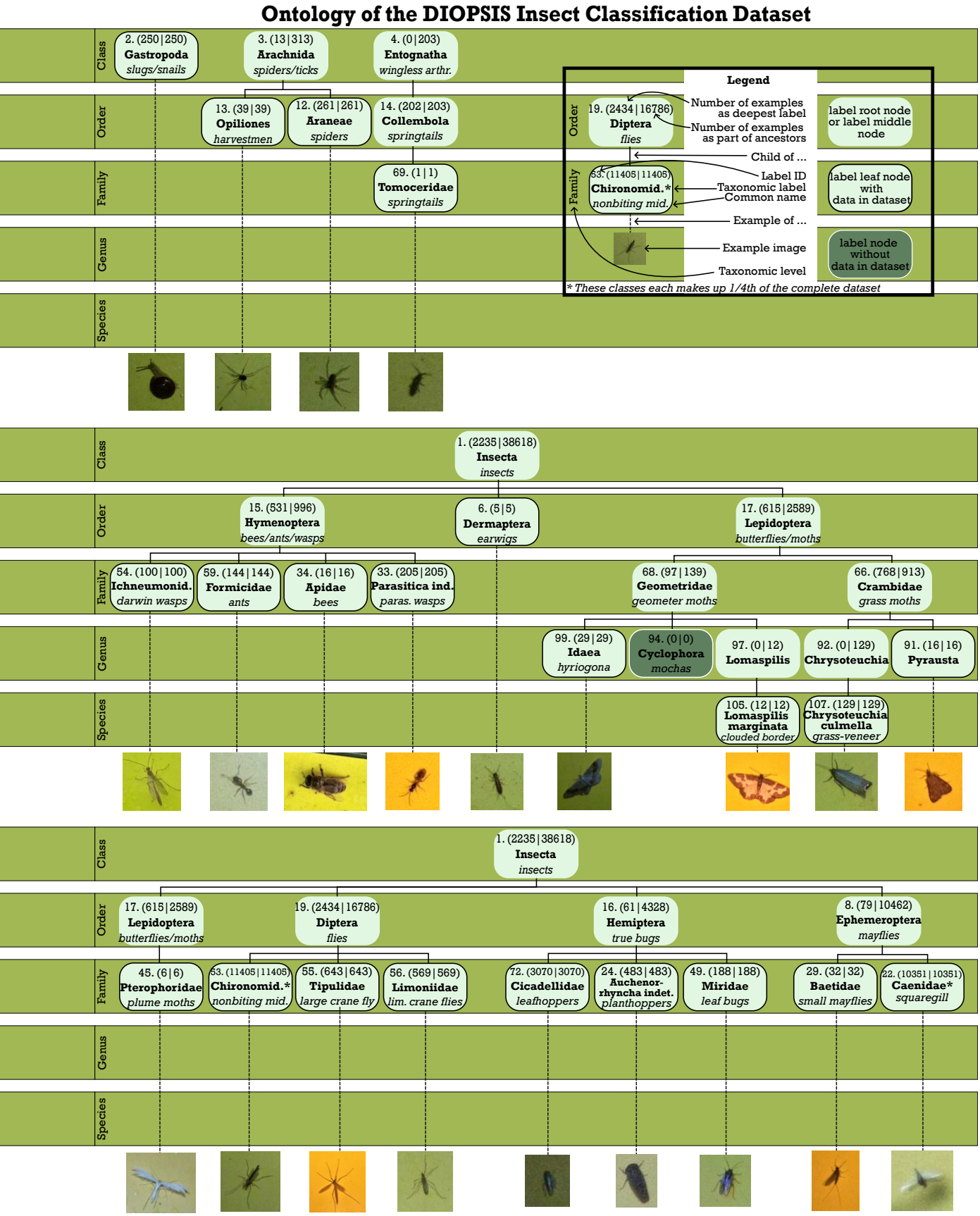


Figure 7: A small version of the ontology for the DIOPSIS insect classification dataset. Including label id, taxonomic label, common name and number of samples present in the ground truth dataset as deepest label and as part of the ancestors. The figure displays the relation between labels and gives an example image for the deepest labels.

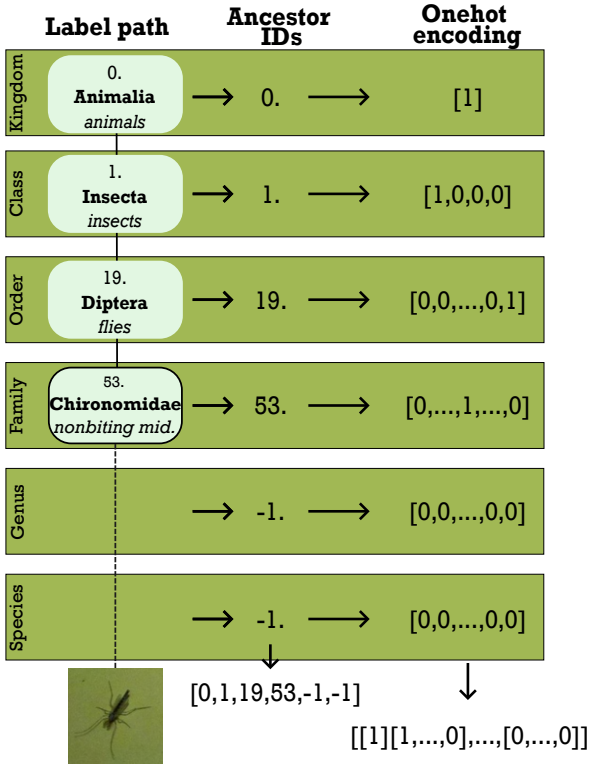


Figure 8: Visual example of how the deepest label of an image labeled as 'Chironomidae' is converted into a list of ancestor IDs with the IDs of the deepest assigned label and all its ancestors. This list of ancestor IDs is then converted into a onehot encoding for training the model. If a label at a taxonomic level is unknown, for example at genus and species level, the ID for that level is set to -1 and the onehot encoding consists of only zeros.

4.1 Base Model

For our base model we develop a CNN based classification model that takes into account the hierarchical nature of the data. The model consists of EfficientNet [42] with the B0 weights that have been pre-trained on ImageNet [9] images. Similarly to Bjerge et al. [4], we replace the classification layer by two fully connected layers with a ReLU and dropout layer in between. Contrary to Bjerge et al., we use only a single set of fully connected layers instead of a set for each taxonomic level, to simplify the model structure. Thus, the first fully connected layer has 512 neurons and the second fully connected layer, or classification layer, has 120 neurons, one neuron for each possible class. The loss functions directly apply Sigmoid (binary cross entropy loss) or Softmax (cross entropy loss), hence the model logit's output are directly fed into the loss function and the output layer of the model is not followed by an activation function. A visual overview of the model can be found in Figure 9. We do make predictions for each individual taxonomic level, like Bjerge et al., however we do this by taking subsets of the neurons, of the final classification layer, associated with the labels of a specific taxonomic level.

We compare two loss functions for the base model: Binary Cross Entropy Loss (BCELoss) and a hierarchical loss (HierLoss), as described by Bjerge et al. [4]. We choose to compare these loss functions to see the impact of explicitly incorporating the hierarchical structure of the labels in the loss function. The BCELoss treats the classification problem as a multi-target classification problem and computes the loss per classification node and treats all labels as independent. HierLoss on the other hand computes a loss per taxonomic level, using levelwise cross entropy in combination with a dependence loss. A comparison of the two loss functions can be found in Figure 10.

Binary Cross Entropy

For the BCELoss we make use of the `torch.nn.BCEWithLogitsLoss()` function by PyTorch. This loss function treats the hierarchical classification problem as a multi-target classification problem, where multiple labels can be assigned to a single image. Each output neuron is treated as a binary classification problem, marking the label as either relevant or irrelevant. This loss does not take hierarchical label consistency into account, meaning that a prediction at level l, might not be the child of the prediction at level $l-1$. The BCELoss for a data sample is computed as follows:

$$\text{BCE} = -\sum_{j=1}^{M} [y_j \log(\tilde{y}_j) + (1-y_j)\log(1-\tilde{y}_j)] \quad (1)$$

Here M is equal the total number of labels, 120 in our case, y_j is the expected model output of the j^{th} label and $\tilde{y}_j$ is the actual model output of the neuron associated with the j^{th} label.

Hierarchical Loss

The HierLoss is implemented as defined by Bjerge et al. [4]. This loss computes the cross entropy loss for the nodes associated with each taxonomic level and computes a dependency loss to take into account the hierarchical label consistency. The level loss (*lloss*) for a taxonomic level l is the cross entropy loss of that level and is defined as follows:

$$\text{lloss}_l = -\sum_{j=0}^{M} y_{lj} \log(\tilde{y}_{lj}) \quad (2)$$

Here M is the total number of labels at the l^{th} taxonomic level, y_{lj} is the expected model output of the j^{th} label at the l^{th} taxonomic level and $\tilde{y}_{lj}$ is the actual model output of the neuron associated with the j^{th} label at the l^{th} taxonomic level.

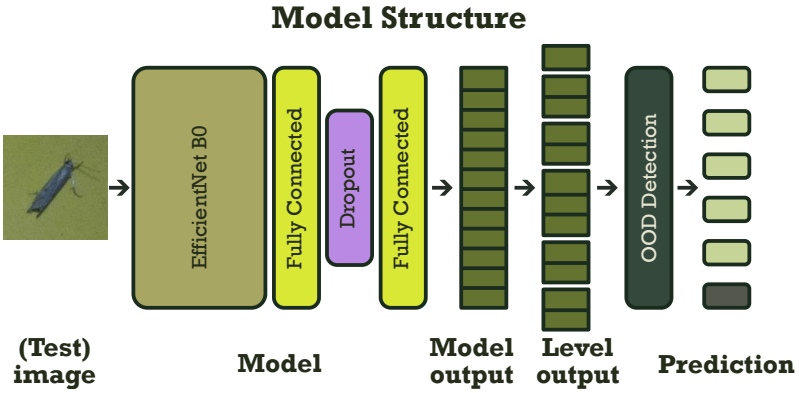


Figure 9: The structure of the base model and how data is fed through the model at inference. An image is fed through the EfficientnetB0 model, followed by a fully connected layer with 512 neurons, a dropout layer and a classification layer of 120 neurons. This leads to a model output for each individual label, which is split over the taxonomic levels. This level output is fed through an out of distribution detection method which creates the final predictions for this image for each taxonomic level.

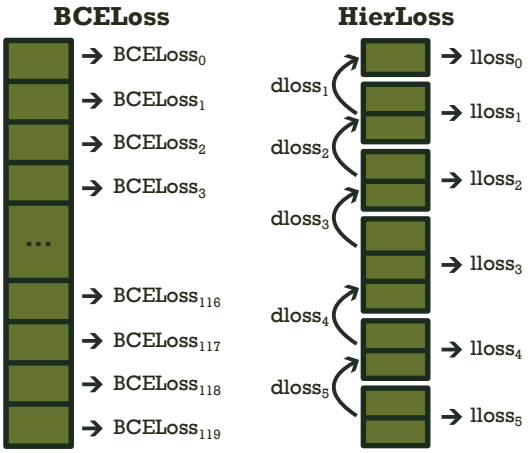


Figure 10: A visual comparison between the BCELoss and the HierLoss. HierLoss takes into account the hierarchical relations between the predictions per taxonomic level, whereas BCELoss treats each label individually.

The candidate label id $\tilde{y}_l$ for a data point at taxonomic level l is determined by computing the Softmax for that level, and taking the label with the maximum value as defined in Equation 3.

$$\tilde{y}_l = \arg\max_{j \in M}(\tilde{y}_j) \tag{3}$$

Here M indicates all the labels that belong the l^{th} level and $\tilde{y}_j$ is the Softmax value of the j^{th} label for a data sample. This candidate prediction is used to compute the dependency loss. The dependency loss penalises the learning model if the predicted labels of two successive levels are not in the relation of parent to child. The dependency loss is defined as follows:

$$dloss_l = \exp^{D_l} - 1, D_l = \begin{cases} 1 & \text{if } \tilde{y}_l \nRightarrow \tilde{y}_{l-1} \\ 0 & \text{else} \end{cases} \tag{4}$$

Here D_l denotes whether the predicted label in the l^{th} level is a child of the predicted label at the in the $l-1^{\text{th}}$ level. Finally the total hierarchical loss $J(\theta)$ of a data sample is defined as the weighted sum of the level an dependence losses, i.e.:

$$J(\theta) = \alpha \sum_{l=1}^{L} \text{lloss}_l + (1-\alpha) \sum_{l=2}^{L} \text{dloss}_l, \quad 0 < \alpha \leq 1 \tag{5}$$

Here α is a hyperparameter that defines the priority between cross entropy level loss ($\alpha = 1$) and the dependence loss ($\alpha = 0$). Bjerge et al. set α to 0.5 as they did not find marginal changes in accuracy when changing the parameter [4], hence we also use this value.

Out of Distribution Detection

As a final step in our model, we apply out of distribution detection to determine whether the prediction made by the model is confident enough to use. This allows the model to refrain from making predictions at lower taxonomic levels when the image quality is too low or the insect's species does not occur in the train set. We choose to follow previous work and make use of classification with a reject option approach [36, 52, 11]. This means that if the output of the model falls out of the allowed uncertainty for all labels at a certain taxonomic level, the prediction is set to 'unsure' in post-processing. Specifically, we make use of the Maximum Softmax Probability (MSP) as described by Hendrycks et al. [17].

To determine whether the output of the model for a specific level falls out of the allowed uncertainty, we take the neurons in the output layer associated with that level and apply Softmax to this subset of labels. Then we determine the label id with the maximum Softmax score to get the candidate prediction for that layer. If the Softmax value of that prediction is higher than γ, the

MSP threshold, we say that the prediction for that level falls within the allowed uncertainty and the candidate prediction becomes the final prediction. If the Softmax value is lower, that level gets assigned the 'unsure' label as the final prediction.

To ensure the final prediction at each level is consistent with the level above, similar to Bjerge et al. [4], we make use of a dependency factor in the OOD detector. This measure checks whether a final prediction at level l is consistent with prediction at level $l-1$. If the prediction at level l is inconsistent with the prediction at level $l-1$, the predicted label's path is cut off at the prediction at level $l-1$ and all deeper levels get the 'unsure' label assigned. This guarantees dependency consistency and reduces the severeness of the errors made by the model.

Thus, the id of the final predicted l^{th} level label of a data sample is determined as follows:

$$\tilde{y}_l = \begin{cases} \tilde{y}_l & \text{if } \max_{j \in M}(\tilde{y}_j) > \gamma \text{ and } \tilde{y}_l \Rightarrow \tilde{y}_{l-1} \\ unsure & \text{otherwise} \end{cases} \quad (6)$$

Here M indicates all the labels that belong to the l^{th} level, $\max_{j \in M}(\tilde{y}_{ij})$ indicates the highest Softmax value for the data sample. We empirically determine that we set γ to 0.9. The selection of the OOD measure is described in appendix B.

4.2 Base Training

To train the base version of the model we define an overarching BatchLoss which computes the loss of the model per batch. It is defined as follows:

$$\text{BatchLoss} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \tilde{y}_i) \quad (7)$$

Here N denotes the number of samples in the batch of labelled data, L denotes the average base model loss over all labels, so either the BCELoss or the HierLoss, y_i denotes the ground truth of the labelled data and $\tilde{y}_i$ denotes the model's output.

We make use of 5-folds cross validation to optimally make use of the limited available labelled data, but still get a representative generalisation performance measure. This means that the model is trained five times, with each time a different fold split as test set. During training the weights of the EfficientNet model are not frozen, but used as a starting point for the fine-tuning of the weights for this specific classification problem.

Each model is trained for 8 epochs, as during experiments we find that at this point validation error goes up after going down for both the BCELoss and the HierLoss. The batchsize used is 16 and the Adams optimiser is used for optimisation. All randomisation seeds for the Python libraries (`Pytorch`, `Numpy` and `random`) are set to 1307.

4.3 SSL Training

For the SSL training method we again make use of cross validation as defined above. Furthermore, we make use of the SSL dataset as defined in Section 3. For the SSL training method we make use of Pseudo-label as defined by Lee [24]. This method assigns pseudo-labels to the unlabelled data in that fold and uses these labels as ground truth labels for the unlabelled data to compute a loss for the unlabelled data, which, together with the loss on the labelled data, is used to fine-tune the classification model.

The pseudo-labels are acquired by feeding the unlabelled data through the base model and converting the model's most confident label for each taxonomic level into the pseudo-label for that level. If the pseudo-label is too uncertain, i.e. when it does not have a high enough Softmax score, similar to the out of distribution measure, the label for that taxonomic level is set to 'unsure'. The final pseudo-labels are then used as ground truth labels to compute the loss between the model's outputs and the pseudo-labels. A visualisation of this process can be found in Figure 11. The pseudo-labels are recalculated every weights update.

In order to train the Base model using the Pseudo-labelling method, a new overarching SSL loss is defined that creates a weighted average of the base model loss for the labelled and unlabelled data. The SSLLoss based on Lee [24]'s overall loss function is defined as follows:

$$SSLLoss = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \tilde{y}_i) + \beta(t)\frac{1}{N'}\sum_{i'=1}^{N'} L(y'_{i'}, \tilde{y}'_{i'}) \quad (8)$$

Here N denotes the number of samples in the batch of labelled data and N' the number of samples in the batch of unlabelled data. L denotes the average base model loss over all labels, so either the BCE loss or the hierarchical loss. y_i is the ground truth of the labelled data and $\tilde{y}_i$ the model's output. y'_i is the pseudo-label of the unlabelled data and $\tilde{y}'_i$ the model's output. $\beta(t)$ is the coefficient balancing the loss for the labelled and unlabelled data.

The $\beta(t)$ coefficient is determined by a weighting schedule over time, this way the model starts with only updating the weights based on the labelled loss and incrementally increases the influence of the unlabelled data. $\beta(t)$ is defined as follows:

$$\beta(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\beta_f & T_1 \leq t < T_2 \\ \beta_f & T_2 \leq t \end{cases} \quad (9)$$

Here β_f is the maximum influence value of $\beta(t)$, T_1 is the iteration at which $\beta(t)$ starts slowly increasing and T_2 is the iteration from which $\beta(t)$ remains the same value. Lee assigns the variables the following values, which we copy in our research: $\beta_f = 3, T_1 = 100, T_2 = 600$ [24].

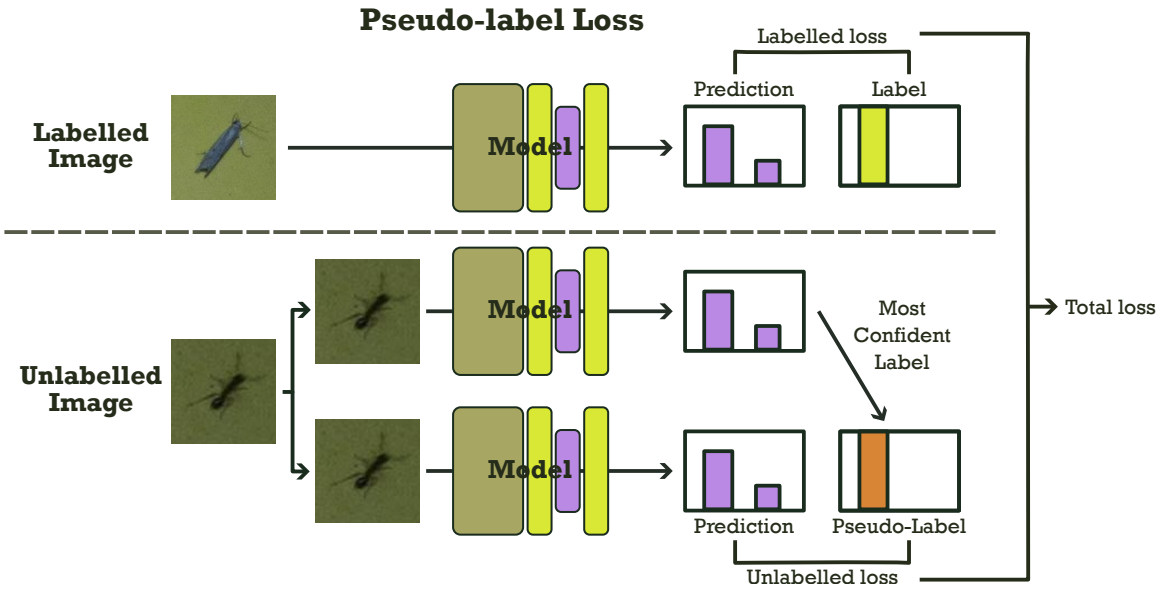


Figure 11: Visual overview of the optimization process of the Pseudo-Label algorithm. Labelled images are fed through the model and the output of the model is compared with the ground truth label and the labelled loss is computed (BCELoss or HierLoss). For unlabelled images the image is fed through the model after which the most confident prediction is converted into a pseudo-label, which is compared against the original prediction to compute the unlabelled loss. The weighted sum of the labelled and unlabelled loss results in the total loss.

The batchsize for the unlabelled data is set at 16, equal to the labelled dataset and the rest the training settings are kept the same as for base model training.

4.4 Evaluation

The goal of this research is to obtain a model that is good at classifying common animals that land on the DIOPSIS cameras. We evaluate the performance of the insect classification model both on generalisation performance and specifically on hierarchical classification performance. The metric scores are computed as a mean and standard deviation over the five training folds for each run.

For the generalisation performance we make use of common multi-class classification evaluation metrics, namely, F1 score, precision and recall, which are also used in previous work [43, 4, 10]. To get more insight in the general classification performance we compute the average scores over all taxa, but also the average scores for each of the taxonomic levels and the average scores for the head, mid and tail split as described in Section 3. These scores, respectively, give us insight in the overall performance, in the performance per taxonomic level and in the impact of the data imbalance on the model performance. The individual metrics are described below.

The **precision** gives the fraction of relevant instances among the retrieved instances, informing how valid the results are on a scale of 0.0 (invalid) to 1.0 (valid). It is computed as follows:

$$\text{precision} = \frac{TP}{TP + FP} \tag{10}$$

The **recall** gives the fraction of the retrieved relevant instances among all relevant instances, informing how complete the results are on a scale of 0.0 (incomplete) to 1.0 (complete). It is computed as follows:

$$\text{recall} = \frac{TP}{TP + FN} \tag{11}$$

The **F1 score** gives the harmonic mean of the precision and the recall, which, in comparison to a basic accuracy score, can better deal with label imbalance. It returns a score between 0.0, low precision or recall, and 1.0, perfect precision and recall. The F1 score is computed as follows:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{12}$$

For the hierarchical performance evaluation we make use of a metric that does not just look at the overall mistakes, but also takes into account the severity of the mistakes. For example, we can say that an error in the same path (e.g. predicting as deepest label the Order of bees instead of the Family of bumble bees) is less bad than an error in another branch (e.g. predicting as deepest label the Family of mosquitoes instead of the Family of bumble bees). Furthermore, we can say that too specific entries could actually be correct. Computing the severity of hierarchical classification errors can be done using a node distance based metric to check the distance between the predicted and ground truth taxa [39]. An example of this is the Taxon Distance (TD) as defined by Hogeweg et al. [19].

The TD is an evaluation measure to indicate how conceptually different two taxa, e.g. prediction and ground truth labels, in an ontology are by computing the distance between them [19]. The TD is the number of edges that needs to be traversed to go from the true taxon to the predicted deepest taxon in the ontology. The TD takes note of two different types of errors: partially wrong - too general or too specific, or completely wrong - in another branch of the ontology. Over all samples TD computes the percentage of each of the types of errors - too general, too specific and other branch, which gives insight in the overall hierarchical classification performance of the model. An example of the computation of the Taxon distance can be found in Figure 12.

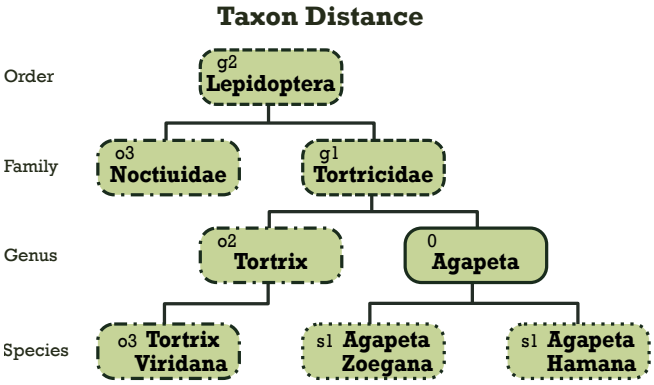


Figure 12: Example of a taxonomic tree with taxon distances (TD) indicated. The TD is the number of edges between the true taxon (Agapeta) and other taxa. The true taxon is indicated by a solid line, the taxa 'Tortricidae' and 'Lepidoptera' are part of the complete taxonomy of 'Agapeta' and not strictly incorrect, but are too general. s1 means a taxon distance of 1 on a too specific path. g1 and g2 indicate predictions that are too general at a distance of respectively 1 and 2 nodes from the true taxon, o2, and o3 refer to taxon distances of respectively 2 and 3 edges in another branch.

4.5 Materials and Equipment

All experiments are ran on university clusters using SLURM scheduler.

- CPU: Intel Xeon Silver 4214 CPU @ 2.20GHz
- GPU: RTX 2080 Ti (11GB)
- Python version: 3.10.12
- PyTorch version: 2.3.1
- CUDA version: 12.0

5 Results

To investigate to what extend semi-supervised learning can improve the insect classification performance using a hierarchical model and an incomplete dataset we compare the insect classification performance of four different models, based on the evaluation metrics as described in Section 4. Firstly, we train two base models with the base cross validation training setup, each with their own loss function. The first model uses the BCELoss and the second uses the HierLoss. In the rest of the thesis these models will, respectively, be referenced as *Base BCELoss* and *Base HierLoss*. Secondly, we train two base models with the semi-supervised learning training setup, each with their own loss function, BCELoss and HierLoss. In the rest of the thesis these models are, respectively, referenced as *SSL BCELoss* and *SSL HierLoss*.

We first investigate the overall quantitative model performance based on common classification evaluation metrics, then we zoom in on subsets of the labels in the ontology based on taxonomic level and number of data points in the dataset. Then we take a closer look at the qualitative results with respect to the classification performance in the label hierarchy by zooming in on the taxon distance and confusion matrices. In Section 6 these results are discussed in the context of the research.

5.1 Quantitative Results

To get the overall performance of each of the models we look at the F1, precision and recall scores averaged over all the labels in the ontology. The results can be found in Figure 13. When inspecting the results, we see that, unexpectedly, the base models overall have higher F1 scores than the SSL models, at 0.42 and 0.48 for, respectively, the *Base HierLoss* and *Base BCELoss*. However, we do see that the Base and SSL Models have a comparable precision. Thus, the difference in F1 score between the models can only be explained by the difference in recall scores. Between the different loss functions we see that the BCELoss has a higher performance during base training, however for SSL training it has a lower performance with respect to F1 score and recall. The precision scores of the loss functions are almost identical.

To get more insight in the performance of the models with respect to the specific parts of the data, we zoom in on the performance metrics based on averages over subsets of the labels. Specifically, we take subsets with respect to the taxonomic level of the label, as can be seen in Figure 14 and we take subsets based on the number of data points of the label in the dataset, which is presented in Figure 15.

When looking at the performance metrics per taxonomic level in Figure 14 we see that the performance for the base models decreases with each lower taxonomic level, where genus and species level have comparable performance. For these models we see a slightly higher recall for the BCELoss in comparison to the HierLoss. Furthermore, we see that the performance for Class and Order level for the base

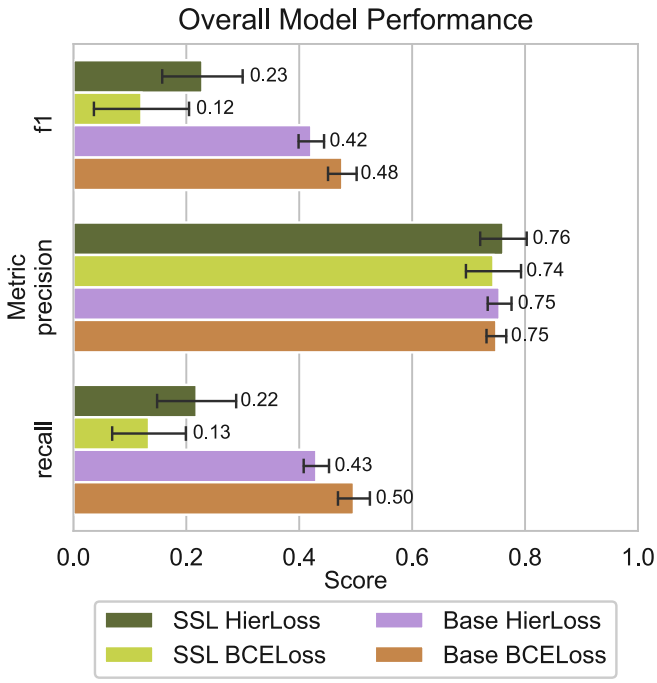


Figure 13: Average model test performance metrics over all of the labels for all four models. F1 score, precision and recall are evaluated.

models is reasonable with F1 scores above 0.6 and precision around 0.9. At family level the precision is still reasonable at around 0.76, but the recall becomes very low at around 0.4.

With respect to the SSL models we see a similar trend in decreasing performance in each of the metrics with each taxonomic level, however, we see a slightly higher performance in the HierLoss in comparison to the BCELoss. Overall the recall is very low, always remaining below 0.37. The Kingdom level only has one label, namely 'Animalia', so each model gets perfect score at that taxonomic level.

When looking at the performance metrics per head, mid, tail split, we see a similar trend as in the taxonomic level. When the amount of data points decreases, so does the performance for all models. However, in this case we see that the performance for the base models is good for the head split with 1000+ data points per label at an F1 score around 0.94 and reasonable for the mid section with 100-1000 data points per label at around 0.62. Both loss functions are comparable for the base model, however the BCELoss shows a slightly higher recall and thus F1 score in all cases.

For the SSL models we also see good performance for the head labels, albeit a little lower than the base models, however for the recall we see a large decrease in recall for the mid and tail labels. Specifically, for the tail labels we see that the average recall for SSL models decreases to 0.06 and 0.00, meaning that the model almost never recalls labels at that level. However, if it recalls labels it is correct more often than the base When comparing the two loss functions for the SSL models we see that the Hierloss model in general performs better than the BCELoss and that the SSL BCELoss model as a higher standard deviation, especially in the tail labels.

To get an even better idea of which aspect of the data, the taxonomic level, or the amount of data points, has the paramount impact on the results, we zoom in even further and look at a scatter plot of the F1 score of the individual labels. On the `x` axis we plot the number of data points of a label and on the `y` axis the taxonomic levels. The result of this F1 score scatter plot for each of the models can be found in Figure 16.

In Figure 16 we immediately see that the base models have higher F1 scores for almost all labels in the head split, as well as the mid split and the top half of the tail split. Furthermore we see that the performance of the base models becomes higher than 0.75 for labels with more than $10^{2.5} \approx 300$ data samples, which is good performance. Additionally, we see for labels with less than $10^{1.5} \approx 30$ data samples that both these models have a near zero F1 score which is smaller than 0.25.

For the SSL models we see an extreme difference in performance between labels, as two labels have very high performance, one of which is the 'Animalia' label at Kingdom level and the other one is the 'Insecta' label at Class level, whereas all other labels have less than 0.75 F1 scores.

The Figure also clearly shows that for the genus and species levels there are never more than 300 data samples, whereas most labels have around 10 data samples or less. When looking at the ontology in Figures 20, 21 and 22, we see that the largest number of samples for a label at genus level and species level are, respectively, 142 and 140, explaining the low performance at species and genus level. Figure 16 also shows the severe amount of labels that have less than 100 data samples.

5.2 Qualitative Results

Besides the quantitative metrics we also look at the severeness of the errors made by the models. Especially with hierarchical labels there are bad and worse errors to make. For example, predicting the parent node of the actual label as deepest label is less bad than predicting a deepest label that does not even fall under the true deepest label's parent.

To inspect the quality of the errors we look at the overall test taxon distance. When inspecting the TD values for the individual models we find that *SSL HierLoss* has an overall taxon distance of 2.54, *SSL BCELoss* has an overall taxon distance of 3.48, *Base HierLoss* has an overall taxon distance of 1.69 and *Base BCELoss* has an overall taxon distance of 1.71. These taxon distances indicate that the base models

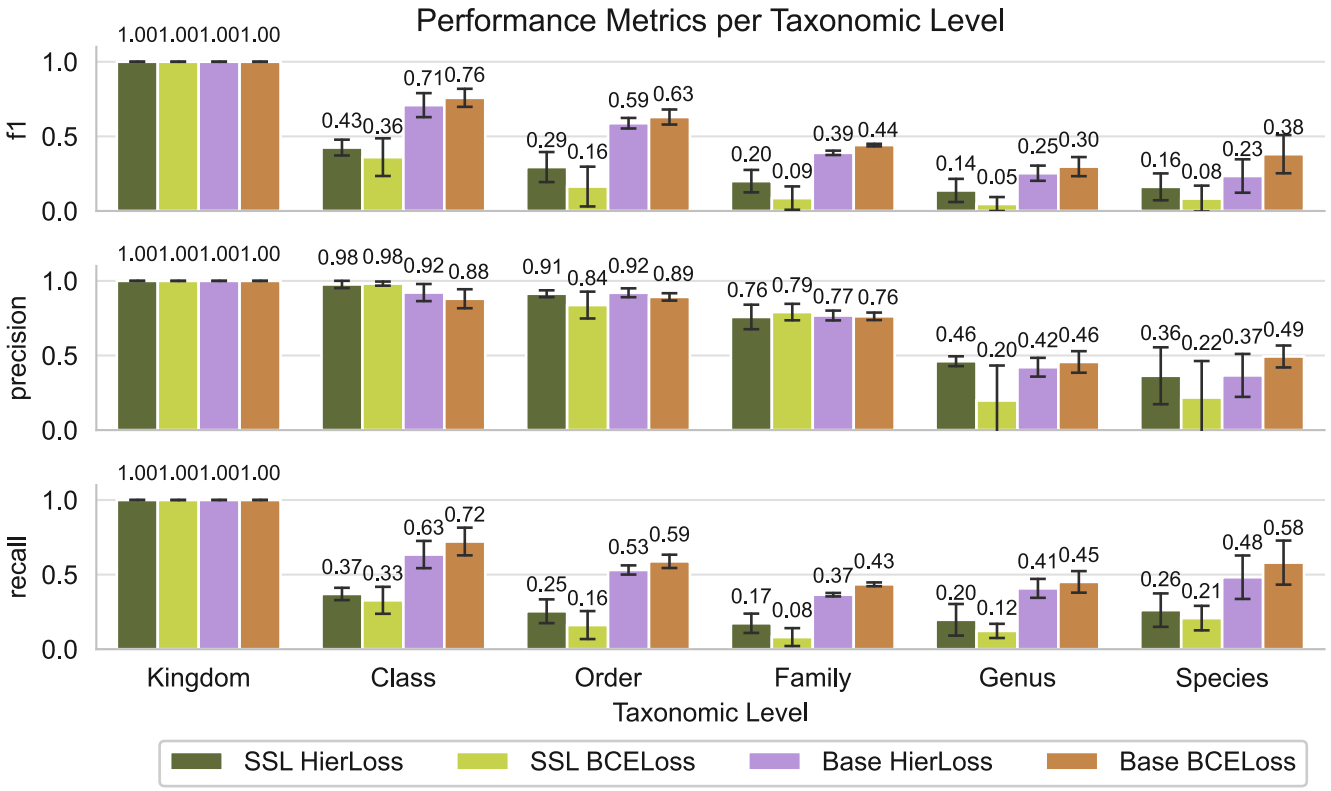


Figure 14: The test performance per model averaged over the labels at the taxonomic levels.

makes predictions closer to the originally assigned deepest label than the SSL models. Furthermore, the SSL HierLoss model makes less worse mistakes than the SSL BCELoss model.

To further inspect the severity of the errors made we look at the types of errors made by the models. In Figure 17 we see, for each of the models, the percentage of predictions that fall under one of the TD categories. In this figure we see that actually both base models have 74% and 77% of the deepest label predictions correct and only make 3% and 4% other branch mistakes.

For the SSL models we see that the main difference with the Base models lies with the percentage of other branch errors, which is much higher in the SSL models. The too general and the too specific errors are comparable in percentages for all of the models. What also stands out is the difference in other branch error percentage between the HierLoss and the BCELoss for the SSL models.

To see which predictions are actually made by the models, we have a look at the confusion matrices (CMs) at the Order and Genus levels of the predictions made by the *Base BCELoss* and the *SSL BCELoss* models, which, respectively, have the overall highest and lowest performance. The CMs can be found in Figure 18. Observations we make from these figures are that originally unsure labels are often assigned to any of the other classes. However, this happens more for the SSL model than for the base model. We see that the SSL model is overfitting on one particular label, which is not one of the largest majority classes in the dataset. We see that the BCELoss model is slightly overfitting to the 'Diptera' Order, but it still keeps quite some true positive predictions. The near zero recall at Genus level for the SSL BCELoss model can be explained by the fact that only the one label is predicted, causing the other labels to have zero recall.

6 Discussion

Insect classification is an important step in the process of insect monitoring, unfortunately insect classification suffers from incomplete datasets where labels are not assigned at all taxonomic levels and where acquisition of these labels can be very challenging due to the finegrainedness of the data and low image quality. We perform experiments to test methods to tackle these challenges. The results are described in Section 5. This section discusses these results by placing them in the contexts of hierarchical insect classification. We discuss the results by comparing the performance of the two loss functions with base model training and SSL training. Furthermore, we zoom in on the aspects of the data and models that affect these results. Based on this discussion we provide recommendations for future work to further advance research into hierarchical insect classification.

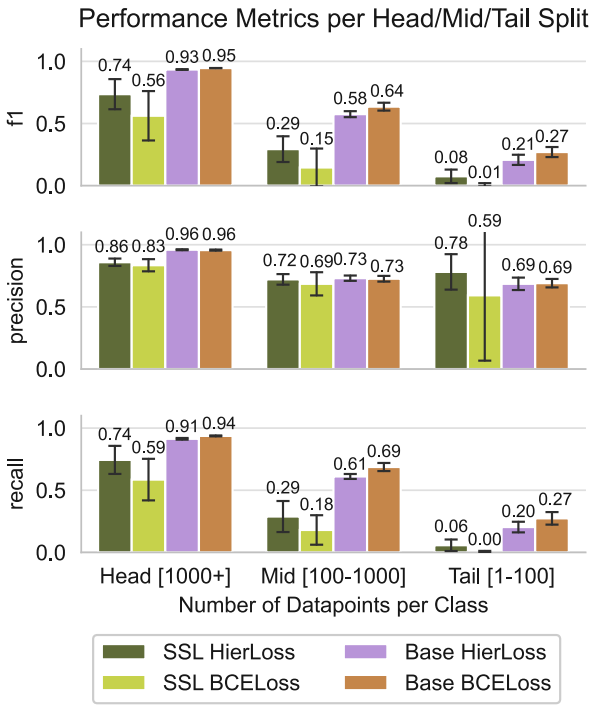


Figure 15: The model test performance averaged over the head, mid, tail split of the labels based on the amount of datapoints per label.

6.1 Experiment Results

In an ideal scenario, the F1 score for each of the labels in the ontology would be above 0.9. This would mean that the model is great at recalling which insect is visible in the image and almost always is correct in its predictions. However, as can be seen in Section 5 this is not the case for all labels. Overall the base models have high F1 scores on the labels with the largest amount of data, which are generally also labels that are located higher in the taxonomy. However, all models suffer from a low recall. Overfitting causes the model to predict label A for images with labels B-Z, meaning that labels B-Z have a low recall score, while class A has a high recall score. However, the recall score is computed as an average over all taxa without weights, hence the overfitting diminishes the recall scores of all of the models.

The models do have a reasonable precision score above 0.75 at Family, Order, Class and Kingdom levels. This means that when the model makes a prediction it is often correct. Thus for labels B-Z the precision might be quite high, while for label A, which is often predicted for other ground truth labels, the precision is quite low. Since the precision is averaged over all the labels without weights, the overfitting minimally impacts the the precision scores for all of the models.

We do not see a big difference between the performance of the two loss functions, this differed from what was anticipated. Due to the Hierarchical nature of the data we expected that the HierLoss would create less bad errors and thus have lower percentage of other branch errors than the BCELoss, due to its dependency factor. We do see a clear difference between the performance of the base models and the SSL models. In comparison to the SSL models, the base models overall have higher F1 and recall scores and the difference in performance of the two loss functions is smaller for the base models. From this we can see that the SSL training method used in this research harms classification performance more than that it helps. In the rest of this Section we will analyse different aspects of the labelled data, base model, SSL training method and unlabelled data to explore how they might impact results and what we can learn from that.

Labelled Data

There are several aspects of the labelled dataset that can impact the results of our experiments, including the number of taxonomic levels, the number of labels, the incomplete annotations, the skewed distribution of data points for these labels and the image quality.

When comparing the number of labels and taxonomic levels in our research to the number of taxa in related work, we see that there is a trade-off between number of taxonomic levels and number of labels. Other papers have higher performance, but use less taxonomic levels [48, 4, 30] and/or less labels per taxonomic level [4, 30]. Meaning that the problems tackled by these papers are less complex than identification of all species in the DIOPSIS dataset. Future work could try to simplify the classification problem by focusing, for example, on predictions on family level and higher, reducing the number of taxonomic levels or focus on only using taxa with enough data.

Another aspect of the data is the incomplete annotations. Based on Figure 6 we can see that only 1% of the annotations go as deep as Genus level and only 0.8% as deep as Species level. Hence, the low performance at these low taxonomic levels is not odd. Especially if you compare it against previous work where all datasets have annotations for all taxonomic levels [4, 30, 48], whose performance is higher. We find it mainly rather interesting that the F1 scores of the models at these two low levels do not differ a lot from the F1 scores at Family level. This can be seen as a good example of the balance between task complexity and amount of data and their impact on model performance. More available data and easier tasks generally have higher performance. The Genus and Species level, respectively, only have 25 and 16 labels, whereas at Family level there are 59 labels, making the task of classification at Family level more complex. However, there is significantly more data that has annotations at Family level, than data that has labels at Genus and Species level, which should work in

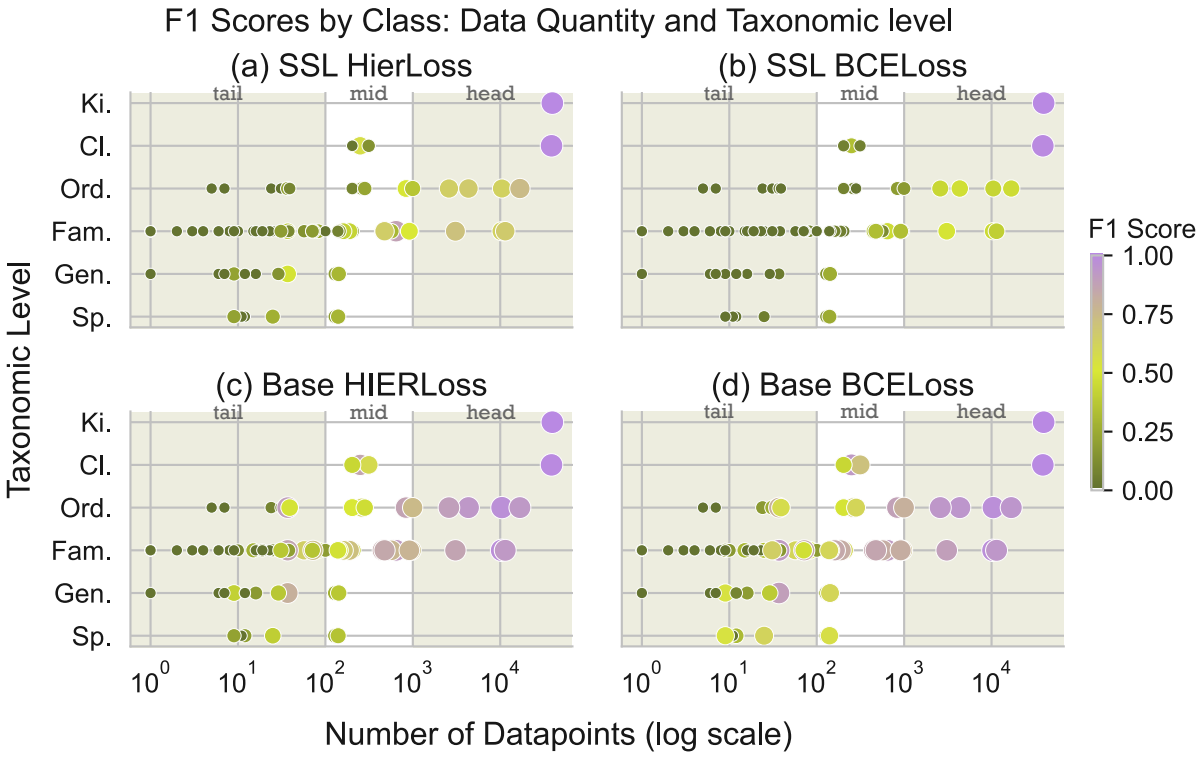


Figure 16: Each scatter plot represents the test F1 scores for each individual label per model. The individual labels have been scattered according to the number of data points in the total dataset, including the data points where the label is part of an ancestor path, and the taxonomic level at which the label is located. The background indicates which labels belong to the head, mid or tail split of the data based on the amount of data points.

favour of the classification task at Family level.

The imbalance in the dataset also makes the classification task more complex for our model, as it causes the model to overfit on majority labels, resulting in low recall for the other labels, diminishing the overall performance. The number of labels with less than 100 data points in our dataset is significantly higher than what can be found in related work. For example, Bjerge et al. use labels that all have over 200 data points [4], and Tresson et al. only have 1 out of 22 labels with less than 100 samples (93 to be precise). Hence the imbalance present in this dataset is bigger than in previous work.

Moreover, the image quality used in our research is lower in comparison to related work. For example, Ong and Hamid [30] use high quality images of insects in museums. Whereas, the size of the insects used in research by Bjerge et al. [4] is bigger, as their research mainly focuses on bee-like insects. The DIOPSIS dataset consists largely of small flies. Hence the quality of the images can have harmed the performance of our model. Furthermore, lighting and appearance of shadows can be limiting factors in the image quality, making the task even more complex.

All in all the labelled dataset contains many factors that can limit the performance of image classification models in comparison to related work. Future work could tackle these problems by improving image quality, through image upscaling or camera quality improvement, or by focusing on classification techniques dedicated to small object identification or tackling the long tail problem, or by simplifying the problem to improve model performance.

Base Model

There are several aspects of the base model that can impact the results of our experiments, including the model complexity, the loss functions and the out of distribution detection method used.

For our base model we choose a quite complex model structure, consisting of a pre-trained image classification model, based on previous work [43, 4, 33, 45]. However, larger models have a bigger tendency to overfit on data than smaller models. A smaller model, like the model used by Rustia et al. [37], who just used a basic CNN for low-resolution image recognition, would potentially experience less overfitting and perform better.

Another path we take with respect to model structure is using a flat classification layer, instead of building the taxonomic hierarchy into the classification model structure itself, as done in previous work [13]. This forces our model to take a multi-task approach where it is has to learn to predict multiple levels at the same time instead of using the prediction of the higher level for the lower levels. Using the flat structure does

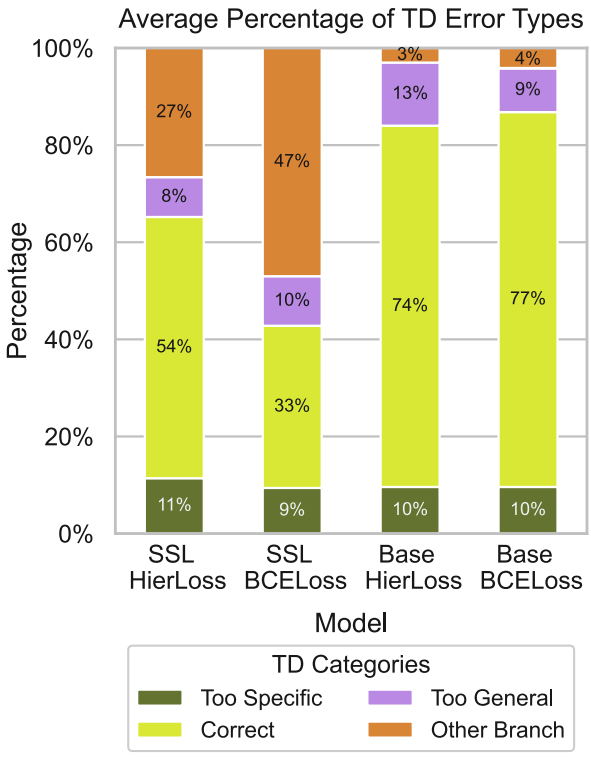


Figure 17: An overview of the percentage of predictions that fall under each of the taxon distance categories per model. The four different models are compared. It can be seen that the SSL models suffer from more other branch errors, whereas the Base models have a high percentage of correct deepest label classifications.

make our model less complex which could reduce overfitting.

With respect to the loss function we do not find a major difference between the BCELoss and HierLoss. This might be caused by the dependency factor in the OOD detector, which forces prediction at a level to be consistent with the prediction at the level above. This hierarchical label inconsistency would be the major downside of using BCELoss over HierLoss, since BCELoss does not assign a penalty to inconsistent predictions between levels, but this disadvantage is nullified by the OOD detection method. When comparing our HierLoss with the losses by Bjerge et al. [4] and Gao et al. [12], we see that their models do get better performance, however Gao et al. also make use of embedding the hierarchy in the model structure and Bjerge et al. have other factors that simplify the classification problem.

The out of distribution detection causes the predictions to be consistent with each other, which is great. However, it also has a negative impact on the recall of the model, as only when the model is very certain with a Softmax score higher than 0.9 are predictions accepted. This causes the model to have high precision, however this might also impact the recall too negatively. Future applications should have an uncertainty measure to allow the model to only make predictions at part of the levels where it is certain enough.

SSL Training Method

The low SSL performance is one of the most unexpected results of this research. The main reason for this seems to be that the SSL models are overfitting on specific labels, as can be clearly seen in the confusion matrices in Figure 18. This is caused by the fact that the pseudo-labeling model cannot correct its own errors. If the model is confident but incorrect in its predictions on unlabelled data, the incorrect data still gets included in training, leading to the amplification of its mistakes as a form of confirmation bias. This issue might become worse when the unlabelled data differs in distribution from the labelled data, because then the model's confidence may not accurately reflect its true performance.

Another aspect that can have influenced the performance of the pseudo-label algorithm is the schedule of the β parameter. To see the impact of the β schedule used in this thesis, we perform the same experiments, but with different settings for the T values in the schedule, among which starting at the 8000th iteration (around epoch 4) and ending at the 30.000th iteration. As can be seen in Figure 19, we see for both schedules that the model's training loss becomes much higher once the unlabelled data loss is used. This means that the β parameter might give too much power to the unlabelled data too quickly. Lee states that proper scheduling of $\beta(t)$ is very important for the network performance. If $\beta(t)$ is too high, it disturbs training even for labelled data. Whereas if $\beta(t)$ is too small, we cannot use the benefit from the unlabelled data [24]. Hence, further tuning of this parameter is desired or as an alternative, exploring other types of SSL methods could be the focus of future work.

Unlabelled Data

One possible reason for the limited impact of unlabelled data in our study could be the relatively small size of the unlabelled dataset, which consists of around 31,000 samples, which is roughly equivalent to the size of the labelled training set. In comparison to other SSL studies, like HierMatch [13], where, in comparison to their labelled dataset, significantly larger unlabelled datasets are used, our dataset lacks the scale typically needed to observe substantial gains from SSL. To truly assess the benefits of unlabelled data, a much larger unlabelled dataset is required, as demonstrated by Garg et al. [13]. Additionally, Oliver et al. [29] show that performance improves as more unlabelled data is added, with performance drops only

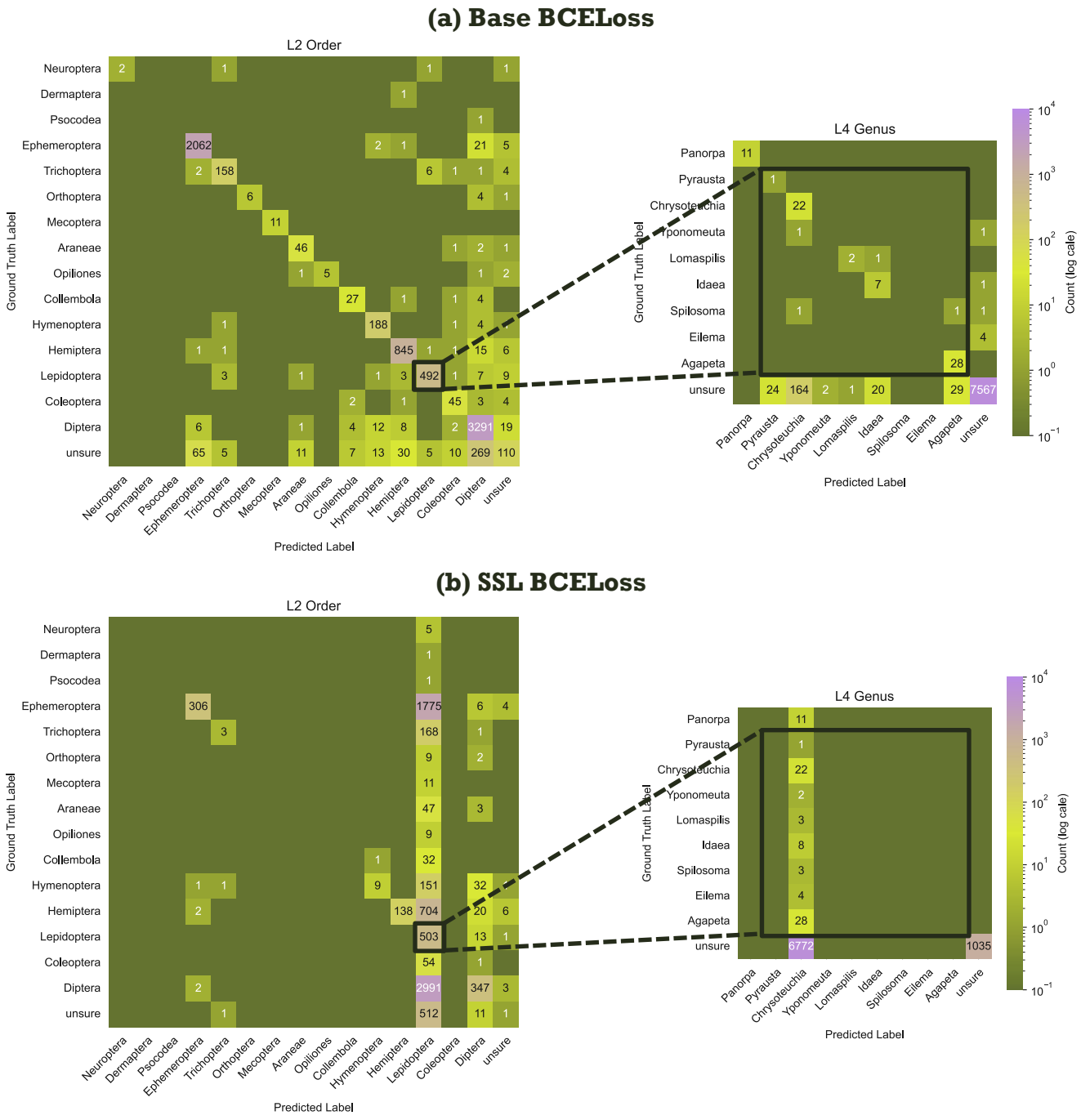


Figure 18: Confusion matrices with the absolute count of the predictions made by the Base BCELoss model (a) and the SSL BCELoss model (b) on the unseen testset of fold 1. The CMs of Order and Genus level are provided, to see the relation between higher and lower level predictions. We see that the SSL BCELoss model is overfitting to a single class, whereas the Base BCELoss model provides reasonable results.

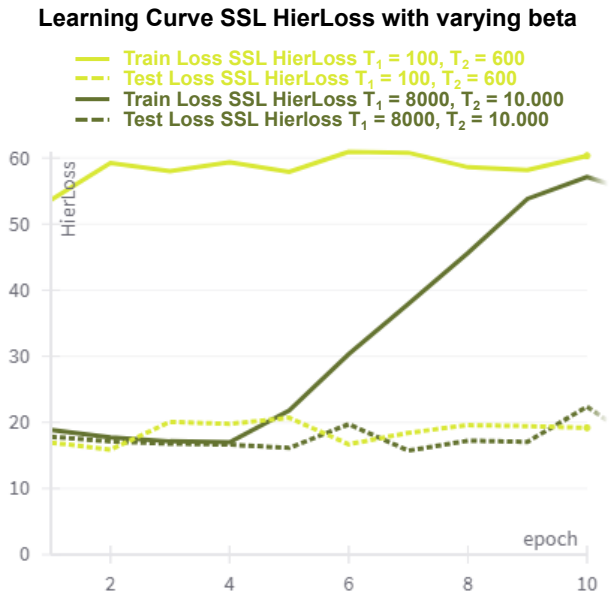


Figure 19: Learning curves of the SSL HierLoss model with two different schedules for the β parameter in the SSL Loss function. The used schedules are: $T_1 = 100$, $T_2 = 600$, the one used in this thesis and $T_1 = 8000$, $T_2 = 10.000$. For both schedules the max β value is set to 3, like done by Lee et al..

occurring when the unlabelled data contains different labels than the annotated data.

In our case, the distribution mismatch between the labelled and unlabelled datasets may contribute to the performance drop, as the labelled data is collected from various locations across the Netherlands, while the unlabelled data came from just five specific locations. Moreover, the types of insects that land on the DIOPSIS camera used for data collection can vary greatly depending on the local ecosystem, such as the nearby presence of colourful flowers or other distractions. Thus, ensuring that both datasets are collected from similar locations is crucial, especially given the vast diversity of insect species. Another aspect that influences the distribution of insects in the unlabelled data is the detection algorithm used to create the crops. It might be that the detection model is biased towards certain insects such that insects that fall under other labels are never detected by this model, which is different from the manual annotations in the labelled dataset, where all insects are detected.

6.2 Future Work

Future work could go in several directions depending on the specific goal of the research. As described the insect classification problem is a very challenging problem, tackling all of the challenges at once makes it even more demanding. We recommend future work to take future research in one of the following directions.

Improve Data Quantity

The collection of labelled insect classification data is difficult, due to the challenging nature of the task, the low image resolution and the lack of available experts. However, to improve performance on minority labels, the collection of more data for those labels would help massively. To efficiently collect more annotated data for those labels, one could make use of active learning [52], where pseudo-labels are generated, however, predictions with high uncertainty are flagged for expert annotation. By training the model to perform well on majority labels with built-in anomaly detection and the ability to make higher-level predictions on unseen insects, we can reduce the workload for experts, who would only need to annotate images with high uncertainty. Incorporating non-expert annotations into this process is another promising avenue, potentially allowing the model to learn from weakly labelled data, as part of a semi-supervised learning structure. An example of an annotation guide for the DIOPSIS Dataset for non-experts can be found in Appendix C, which was developed as potential part of this thesis.

Deal with Class Imbalance

Future research should focus on tackling the longtail problem. However, different goals should take different approaches. For the problem of insect monitoring, which is tackled in this thesis, the goal is to get good performance on as many insects as possible, however the common insects can already give good insight in the insect populations. Hence, if the goal of future work is to get good performance on majority labels, for example when monitoring of common species is desired, we would recommend simplifying the classification problem. For example, by dropping underrepresented labels with less than 100 samples. These images could then still be used for higher taxonomic levels, but will not create noise at the lower taxonomic levels anymore. Additionally, we recommend using only data that has annotations at every taxonomic level on which you want to test performance. In case of the DIOPSIS dataset we would recommend testing on Kingdom-Family level, as those levels have significantly more annotations than the Genus and Species levels. This makes sure the model does not have to learn when it should be too uncertain to make a prediction at a certain level and can just focus on correct classification.

However, if the goal is to increase the performance of the model specifically for the minority labels. For example for research into rare species, there are several approaches that could be explored. Firstly, data augmentation in combination with oversampling could allow the addition of more examples of the minority

labels while creating augmented images of those labels. This could create a more balanced dataset. Secondly, few shot learning techniques could be used to train models on the minority labels or previously unseen labels. Especially if the model learns the hierarchical label structure under the data it should be able to quickly learn that a specific class falls under another. Thirdly, using a weighted loss function, which assigns higher weights to underrepresented labels can also have a positive influence on the performance for minority labels.

Simplify Model Structure

Future work could focus on simplifying the classification model to reduce overfitting, for example with more simple CNN structures, like the model by Rustia et al. [37]. This is interesting, because it would reduce resources needed to train the model and potentially holds similar performance.

Create Hierarchical Models

Future research could also focus on using other methods of explicitly including the hierarchical nature underlying the data. As described by [3] future research could explicitly include the hierarchy in the model, like Garg et al. [13], for example by having classification layers for each taxonomic level, which feeds into the lower level classification layer. Another option would be to train an autoencoder to create hierarchical encodings, similar to Bertinetto et al. [3]. These approaches might capture the hierarchical nature of the problem even more, especially in combination with hierarchical loss functions.

Tackle Fine-Grained Classification

To improve the classification performance for small insects, future work could focus on improving model performance with respect to small object detection. For example, through techniques proposed by previous work [47, 56]. For example, Wang et al. applied domain adaptation approaches to fine-grained image datasets by discovering similarities between these datasets and large amounts of unlabelled data in the wild [56]. The main idea is to use these similarities to enhance the model's ability to generalise across different domains, thereby improving its performance on tasks that require distinguishing subtle differences between closely related categories.

Explore other SSL Methods

Pseudo-label did not turn out to be the best SSL method for hierarchical classification, as it was too severely impacted by the label imbalance. However, previous work has shown successful combinations of hierarchical classification with SSL [13]. Hence, future work should focus on exploring other SSL approaches that will get a performance gain. A simple first step would be to use self-training, a form of semi-supervised learning where a model is completely trained on labelled data. Another option would be to use perturbation based methods like FixMatch [40] and MixMatch [2]. More precisely we recommend future work to look into SSL methods that focus on dealing with label imbalances. For example, as described by de Oliveira and Berton [8]. Our work tries to focus on the hierarchical nature of the data, however, the label imbalance is so severe that it has major impact on the performance and should potentially be tackled first, before one can leverage the hierarchical nature of the data.

Future work should also expand the unlabelled dataset by collecting more images of insects to get a better ratio of labelled versus unlabelled data. In general more unlabelled data will provide better performance [29]. Another approach to use more unlabelled data can be found by taking inspiration from Garg et al. and for each taxonomic level use annotations that have more coarse labels as part of the unlabelled dataset [13], in case of level wise classification.

7 Conclusion

This thesis gives insight in to what extent semi-supervised learning can improve hierarchical insect classification when faced with incomplete datasets, a major challenge for the task of insect classification. To show this we designed two loss functions, BCELoss and HierLoss, and compared a base training method with an pseudo-label SSL approach.

We found that overall SSL demonstrated limited benefits in improving hierarchical insect classification. While the SSL models were expected to leverage unlabelled data and improve overall performance, the results revealed that they often under performed compared to base models trained solely on labelled data. The SSL models were particularly prone to overfitting on specific classes, resulting in poor recall and an imbalance in prediction quality across different taxonomic levels with higher levels having higher performance. As a result, base models trained solely on labelled data outperformed SSL models in terms of F1 scores, precision, and recall, indicating that semi-supervised learning, at least in its current form, hurt rather than helped the model performance.

Several factors contributed to the limited success of SSL in this context. First, the relatively small size of the unlabelled dataset and its distribution mismatch with the labelled data weakened the model's ability to generalise effectively. Moreover, the pseudo-labelling method used was susceptible to confirmation bias,

where incorrect pseudo-labels reinforced existing errors during training. Lastly, the imbalance in the dataset, combined with the hierarchical complexity of insect taxonomy, posed significant challenges that were not fully mitigated by SSL.

When comparing the two loss functions, BCELoss and HierLoss, no big differences were observed in overall performance when used in combination with the base models. Contrary to expectations, HierLoss did not result in a noticeable reduction of errors across taxonomic levels, likely due to the strict out of distribution detection measure used, which forced both the BCELoss and HierLoss model's predictions to be completely consistent between taxonomic levels.

In summary, while semi-supervised learning holds theoretical promise for hierarchical insect classification, particularly in addressing the scarcity of labelled data, it was not as effective in this study. Future research should focus on improving the base model structure, expanding the (un)labelled dataset, and exploring alternative SSL methods that better handle class imbalance and hierarchical data structures.

8 Acknowledgements

As I reach the end of this chapter of my academic journey, I look back with a sense of pride and fulfilment. This thesis is not only a culmination of years of hard work but also the beginning of a new adventure—one where the world beyond student life awaits.

First and foremost, I would like to extend my deepest gratitude to my supervisors, Tom Claassen and Jacob Kamminga. Your supervision and insightful feedback were invaluable throughout this process. More than that, I will look back with a smile on our fun conversations and your thoughtfulness along the way. I also want to thank Inge Wortel, for your flexibility and for taking the time and effort to act as my second reader.

Special thanks to the wonderful people at Arise and Naturalis, especially Chantal, Laurens, and Eelke, for the time and insightful discussions on the insect-side of insect classification. And to Chirag for the fun discussions about the classification-side of insect classification.

To my partner, Maarten—your unwavering support and the many times you've been my confidant kept me grounded and focused. I cannot thank you enough. To my parents, who have been my biggest cheerleaders, your belief in me has meant the world.

A special thanks goes out to everyone in the Thesis Factory. Working alongside like-minded and hard working individuals made the process not only productive but also enjoyable. Finally, to all my friends who made my time as a student unforgettable, thank you for the lovely memories.

Now, with immense gratitude for all these experiences, I step forward into a new chapter—the one of burger!

References

[1] Alves, A. N., Souza, W. S. R., and Borges, D. L. Cotton pests classification in field-based images using deep residual networks. *Computers and Electronics in Agriculture 174* (July 2020), 105488.

[2] Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *Advances in Neural Information Processing Systems* (2019), vol. 32, Curran Associates, Inc.

[3] Bertinetto, L., Mueller, R., Tertikas, K., Samangooei, S., and Lord, N. A. Making Better Mistakes: Leveraging Class Hierarchies With Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 12506–12515.

[4] Bjerge, K., Geissmann, Q., Alison, J., Mann, H. M. R., Høye, T. T., Dyrmann, M., and Karstoft, H. Hierarchical classification of insects with multitask learning and anomaly detection. *Ecological Informatics 77* (Nov. 2023), 102278.

[5] Bjerge, K., Nielsen, J. B., Sepstrup, M. V., Helsing-Nielsen, F., and Høye, T. T. An Automated Light Trap to Monitor Moths (Lepidoptera) Using Computer Vision-Based Tracking and Deep Learning. *Sensors 21*, 2 (Jan. 2021), 343.

[6] Chen, C.-J., Huang, Y.-Y., Li, Y.-S., Chen, Y.-C., Chang, C.-Y., and Huang, Y.-M. Identification of Fruit Tree Pests With Deep Learning on Embedded Drone to Achieve Accurate Pesticide Spraying. *IEEE Access 9* (2021), 21986–21997.

[7] Cheng, X., Zhang, Y., Chen, Y., Wu, Y., and Yue, Y. Pest identification via deep residual learning in complex background. *Computers and Electronics in Agriculture 141* (Sept. 2017), 351–356.

[8] de Oliveira, W. D. G., and Berton, L. A systematic review for class-imbalance in semi-supervised learning. *Artificial Intelligence Review 56*, 2 (Nov. 2023), 2349–2382.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (June 2009), pp. 248–255.

[10] Ganda, D., and Buch, R. A Survey on Multi Label Classification. *Recent Trends in Programming Languages 5*, 1 (Aug. 2018), 19–23.

[11] Gangireddy, R. Knowing the Unknown: Open-World Recognition for Biodiversity Datasets. Master's thesis, University of Twente, Enschede, Apr. 2024.

[12] Gao, D. Deep Hierarchical Classification for Category Prediction in E-commerce System. In *Proceedings of The 3rd Workshop on E-Commerce and NLP* (Seattle, WA, USA, 2020), Association for Computational Linguistics, pp. 64–68.

[13] Garg, A., Bagga, S., Singh, Y., and Anand, S. HierMatch: Leveraging Label Hierarchies for Improving Semi-Supervised Learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2022), pp. 1015–1024.

[14] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Nov. 2016.

[15] Hallmann, C. A., Sorg, M., Jongejans, E., Siepel, H., Hofland, N., Schwan, H., Stenmans, W., Müller, A., Sumser, H., Hörren, T., Goulson, D., and de Kroon, H. More than 75 percent decline over 27 years in total flying insect biomass in protected areas. *PLOS ONE 12*, 10 (Oct. 2017), e0185809.

[16] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition* (June 2016), pp. 770–778.

[17] Hendrycks, D., and Gimpel, K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks, Oct. 2018.

[18] Hogeweg, L., Huijbers, C., Zeegers, T., van Leeuwen, J., and Buesink, R. DIOPSIS camera annotated detection and classification dataset 2021, Mar. 2024.

[19] Hogeweg, L. E., Gangireddy, R., Brunink, D., Kalkman, V. J., Cornelissen, L., and Kamminga, J. W. COOD: Combined Out-of-distribution Detection Using Multiple Measures for Anomaly & Novel Class Detection in Large-scale Hierarchical Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 3971–3980.

[20] Høye, T. T., Ärje, J., Bjerge, K., Hansen, O. L. P., Iosifidis, A., Leese, F., Mann, H. M. R., Meissner, K., Melvad, C., and Raitoharju, J. Deep learning and computer vision will transform entomology. *Proceedings of the National Academy of Sciences 118*, 2 (Jan. 2021), e2002545117.

[21] Huijbers, C., Hogeweg, L., Zeegers, T., Peterse, S., Schoonman, M., and Jongejans, E. DIOPSIS: Automatic cameras and image recognition for standardised, continuous monitoring of insect populations. *IET Computer Vision* (under review).

[22] Kasinathan, T., Singaraju, D., and Uyyala, S. R. Insect classification and detection in field crops using modern machine learning techniques. *Information Processing in Agriculture 8*, 3 (Sept. 2021), 446–457.

[23] La Grassa, R., Gallo, I., and Landro, N. Learn class hierarchy using convolutional neural networks. *Applied Intelligence 51*, 10 (Oct. 2021), 6622–6632.

[24] Lee, D.-H. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (July 2013).

[25] Li, J., Socher, R., and Hoi, S. C. H. DivideMix: Learning with Noisy Labels as Semi-supervised Learning, Feb. 2020.

[26] Li, W., Zheng, T., Yang, Z., Li, M., Sun, C., and Yang, X. Classification and detection of insects from field images using deep learning for smart pest management: A systematic review. *Ecological Informatics 66* (Dec. 2021), 101460.

[27] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014* (Cham, 2014), D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 740–755.

[28] Montgomery, G. A., Dunn, R. R., Fox, R., Jongejans, E., Leather, S. R., Saunders, M. E., Shortall, C. R., Tingley, M. W., and Wagner, D. L. Is the insect apocalypse upon us? How to find out. *Biological Conservation 241* (Jan. 2020), 108327.

[29] Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. In *Advances in Neural Information Processing Systems* (2018), vol. 31, Curran Associates, Inc.

[30] Ong, S.-Q., and Hamid, S. A. Next generation insect taxonomic classification by comparing different deep learning algorithms. *PLOS ONE 17*, 12 (Dec. 2022), e0279094.

[31] Pattnaik, G., Shrivastava, V. K., and Parvathi, K. Transfer Learning-Based Framework

for Classification of Pest in Tomato Plants. *Applied Artificial Intelligence 34*, 13 (Nov. 2020), 981–993.

[32] Preti, M., Verheggen, F., and Angeli, S. Insect pest monitoring with camera-equipped traps: Strengths and limitations. *Journal of Pest Science 94*, 2 (Mar. 2021), 203–217.

[33] Rahman, C. R., Arko, P. S., Ali, M. E., Iqbal Khan, M. A., Apon, S. H., Nowrin, F., and Wasif, A. Identification and recognition of rice diseases and pests using convolutional neural networks. *Biosystems Engineering 194* (June 2020), 112–120.

[34] Redmon, J., and Farhadi, A. YOLO9000: Better, Faster, Stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 7263–7271.

[35] Redmon, J., and Farhadi, A. YOLOv3: An Incremental Improvement, Apr. 2018.

[36] Rustia, D. J. A., Chao, J.-J., Chiu, L.-Y., Wu, Y.-F., Chung, J.-Y., Hsu, J.-C., and Lin, T.-T. Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method. *Journal of Applied Entomology 145*, 3 (2021), 206–222.

[37] Rustia, D. J. A., Lu, C.-Y., Chao, J.-J., Wu, Y.-F., Chung, J.-Y., Hsu, J.-C., and Lin, T.-T. Online semi-supervised learning applied to an automated insect pest monitoring system. *Biosystems Engineering 208* (Aug. 2021), 28–44.

[38] Seibold, S., Gossner, M. M., Simons, N. K., Blüthgen, N., Müller, J., Ambarli, D., Ammer, C., Bauhus, J., Fischer, M., Habel, J. C., Linsenmair, K. E., Nauss, T., Penone, C., Prati, D., Schall, P., Schulze, E.-D., Vogt, J., Wöllauer, S., and Weisser, W. W. Arthropod decline in grasslands and forests is associated with landscape-level drivers. *Nature 574*, 7780 (Oct. 2019), 671–674.

[39] Silla, C. N., and Freitas, A. A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery 22*, 1-2 (Jan. 2011), 31–72.

[40] Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *Advances in Neural Information Processing Systems* (2020), vol. 33, Curran Associates, Inc., pp. 596–608.

[41] Southwood, T., and Henderson, P. *Ecological Methods*, 3rd edition ed. Blackwell Science Ltd, May 2000.

[42] Tan, M., and Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning* (May 2019), PMLR, pp. 6105–6114.

[43] Teixeira, A. C., Ribeiro, J., Morais, R., Sousa, J. J., and Cunha, A. A Systematic Review on Automatic Insect Detection Using Deep Learning. *Agriculture 13*, 3 (Mar. 2023), 713.

[44] Tetila, E. C., Brandoli Machado, B., Menezes, G. V., de Souza Belete, N. A., Astolfi, G., and Pistori, H. A Deep-Learning Approach for Automatic Counting of Soybean Insect Pests. *IEEE Geoscience and Remote Sensing Letters 17*, 10 (Oct. 2020), 1837–1841.

[45] Tetila, E. C., Machado, B. B., Astolfi, G., Belete, N. A. d. S., Amorim, W. P., Roel, A. R., and Pistori, H. Detection and classification of soybean pests using deep learning with UAV images. *Computers and Electronics in Agriculture 179* (Dec. 2020), 105836.

[46] Thenmozhi, K., and Srinivasulu Reddy, U. Crop pest classification based on deep convolutional neural network and transfer learning. *Computers and Electronics in Agriculture 164* (Sept. 2019), 104906.

[47] Tong, K., Wu, Y., and Zhou, F. Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing 97* (May 2020), 103910.

[48] Tresson, P., Carval, D., Tixier, P., and Puech, W. Hierarchical Classification of Very Small Objects: Application to the Detection of Arthropod Species. *IEEE Access 9* (2021), 63925–63932.

[49] van Engelen, J. E., and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning 109*, 2 (Feb. 2020), 373–440.

[50] van Klink, R., August, T., Bas, Y., Bodesheim, P., Bonn, A., Fossøy, F., Høye, T. T., Jongejans, E., Menz, M. H. M., Miraldo, A., Roslin, T., Roy, H. E., Ruczyński, I., Schigel, D., Schäffler, L., Sheard, J. K., Svenningsen, C., Tschan, G. F., Wäldchen, J., Zizka, V. M. A., Åström, J., and Bowler, D. E. Emerging technologies revolutionise insect ecology and monitoring. *Trends in Ecology & Evolution 37*, 10 (Oct. 2022), 872–885.

[51] van Klink, R., Bowler, D. E., Gongalsky, K. B., Swengel, A. B., Gentile, A., and Chase, J. M. Meta-analysis reveals declines in terrestrial but increases in freshwater insect abundances. *Science 368*, 6489 (Apr. 2020), 417–420.

[52] van Osta, J. M., Dreis, B., Meyer, E., Grogan, L. F., and Castley, J. G. An active learning framework and assessment of inter-annotator agreement facilitate automated recogniser development for vocalisations of a rare species, the southern black-throated finch (Poephila cincta cincta). *Ecological Informatics 77* (Nov. 2023), 102233.

[53] Villa-Pérez, M. E., Álvarez-Carmona, M. Á., Loyola-González, O., Medina-Pérez, M. A., Velazco-Rossell, J. C., and Choo, K.-K. R. Semi-supervised anomaly detection algorithms: A comparative summary and future research directions. *Knowledge-Based Systems 218* (2021), 106878.

[54] Wagner, D. L. Insect Declines in the Anthropocene. *Annual Review of Entomology 65*, Volume 65, 2020 (Jan. 2020), 457–480.

[55] Wang, J., Li, Y., Feng, H., Ren, L., Du, X., and Wu, J. Common pests image recognition based on deep convolutional neural network. *Computers and Electronics in Agriculture 179* (Dec. 2020), 105834.

[56] Wang, S., Chen, X., Wang, Y., Long, M., and Wang, J. Progressive Adversarial Networks for Fine-Grained Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 9213–9222.

[57] Xia, D., Chen, P., Wang, B., Zhang, J., and Xie, C. Insect Detection and Classification Based on an Improved Convolutional Neural Network. *Sensors 18*, 12 (Dec. 2018), 4169.

[58] Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics* (USA, June 1995), ACL '95, Association for Computational Linguistics, pp. 189–196.

[59] Zhang, Z.-Q. Animal biodiversity: An introduction to higher-level classification and taxonomic richness. *Zootaxa 3148*, 1 (Dec. 2011), 7–12.

A Ontology

Here you find the full ontology of the DIOPSIS insect classification dataset.

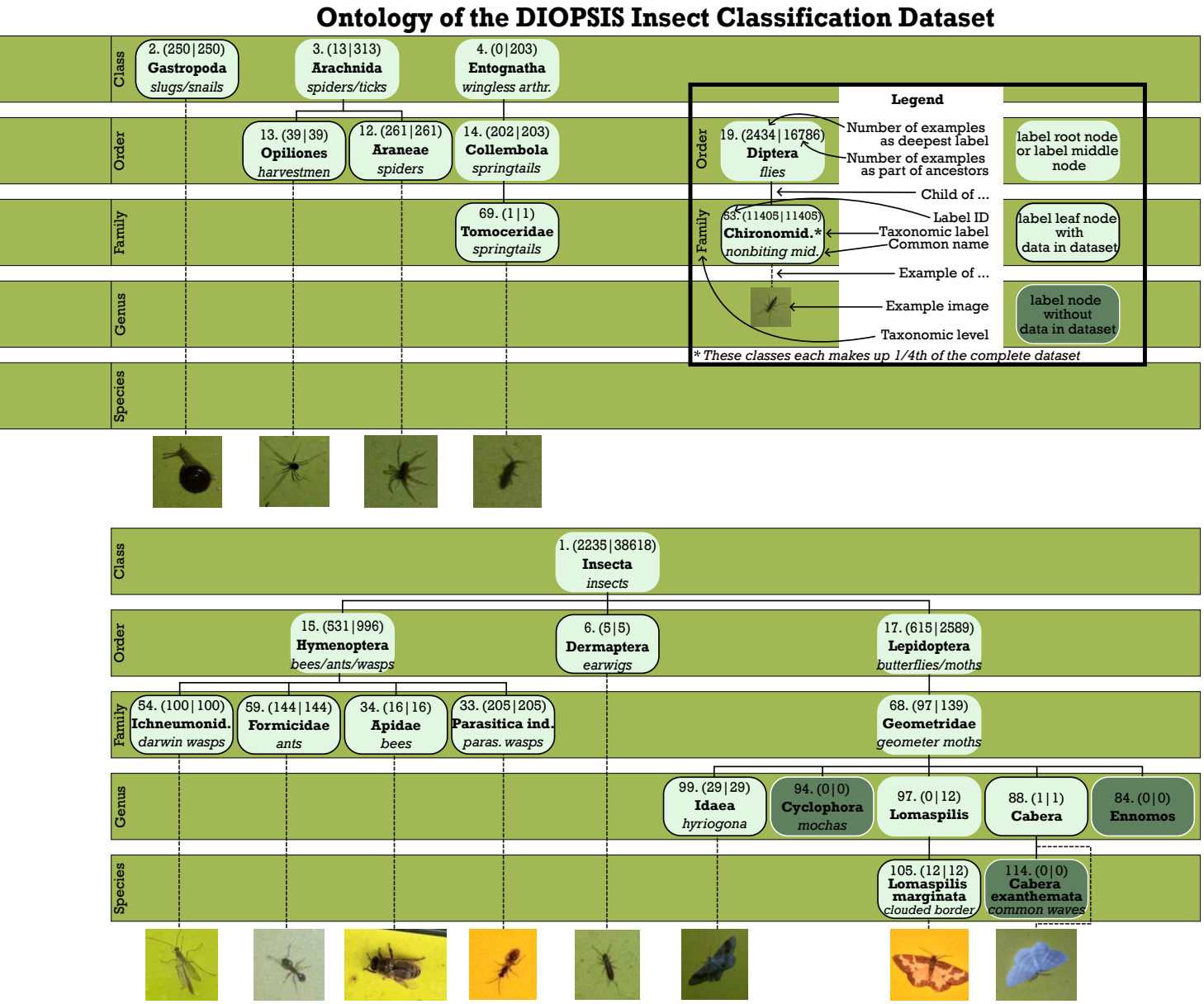


Figure 20: The ontology for the DIOPSIS insect classification dataset. Including number of samples present in the ground truth dataset and an example image from the dataset for the deepest label in that path. (1/3)

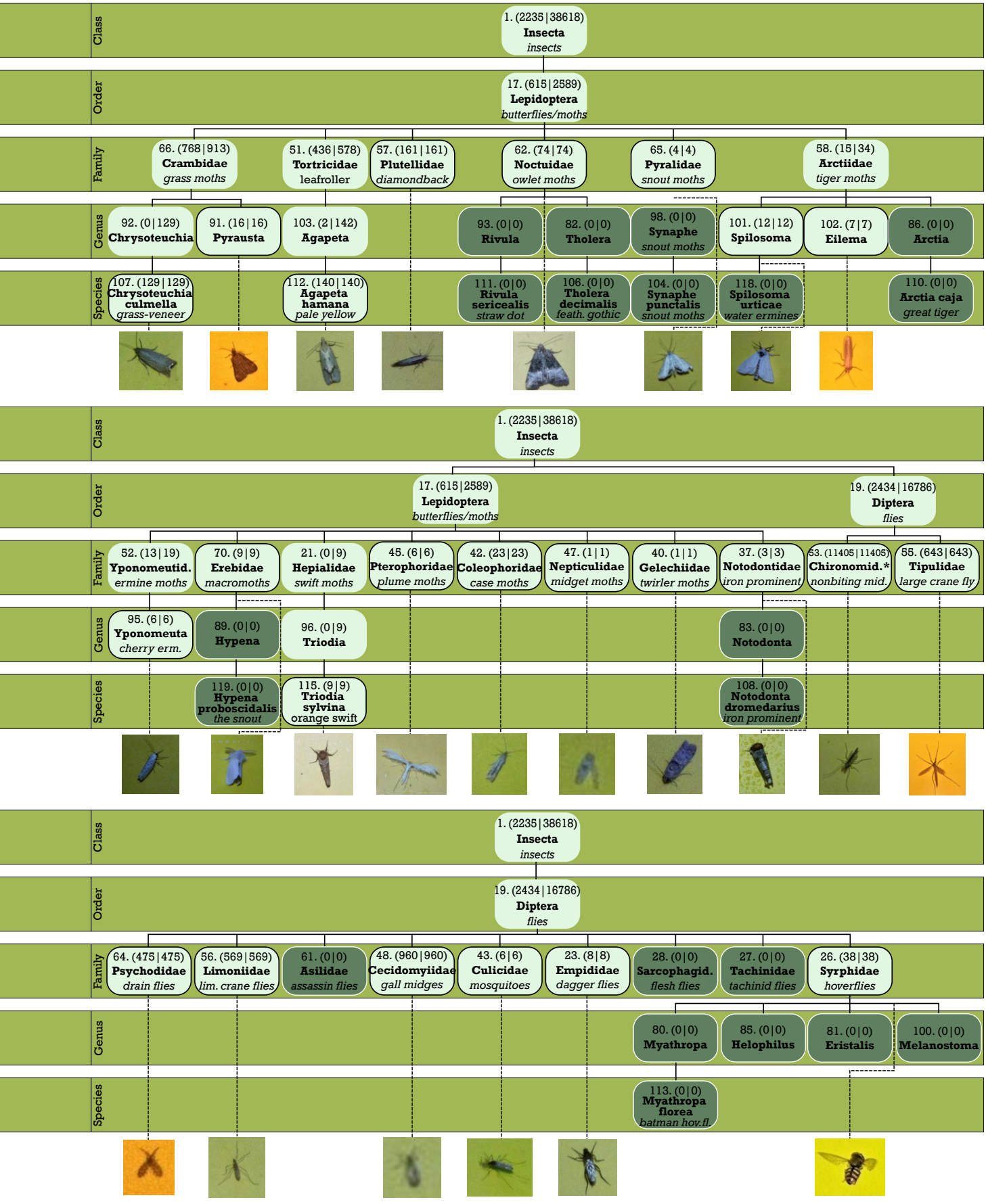


Figure 21: The ontology for the DIOPSIS insect classification dataset. Including number of samples present in the ground truth dataset and an example image from the dataset for the deepest label in that path. (2/3)

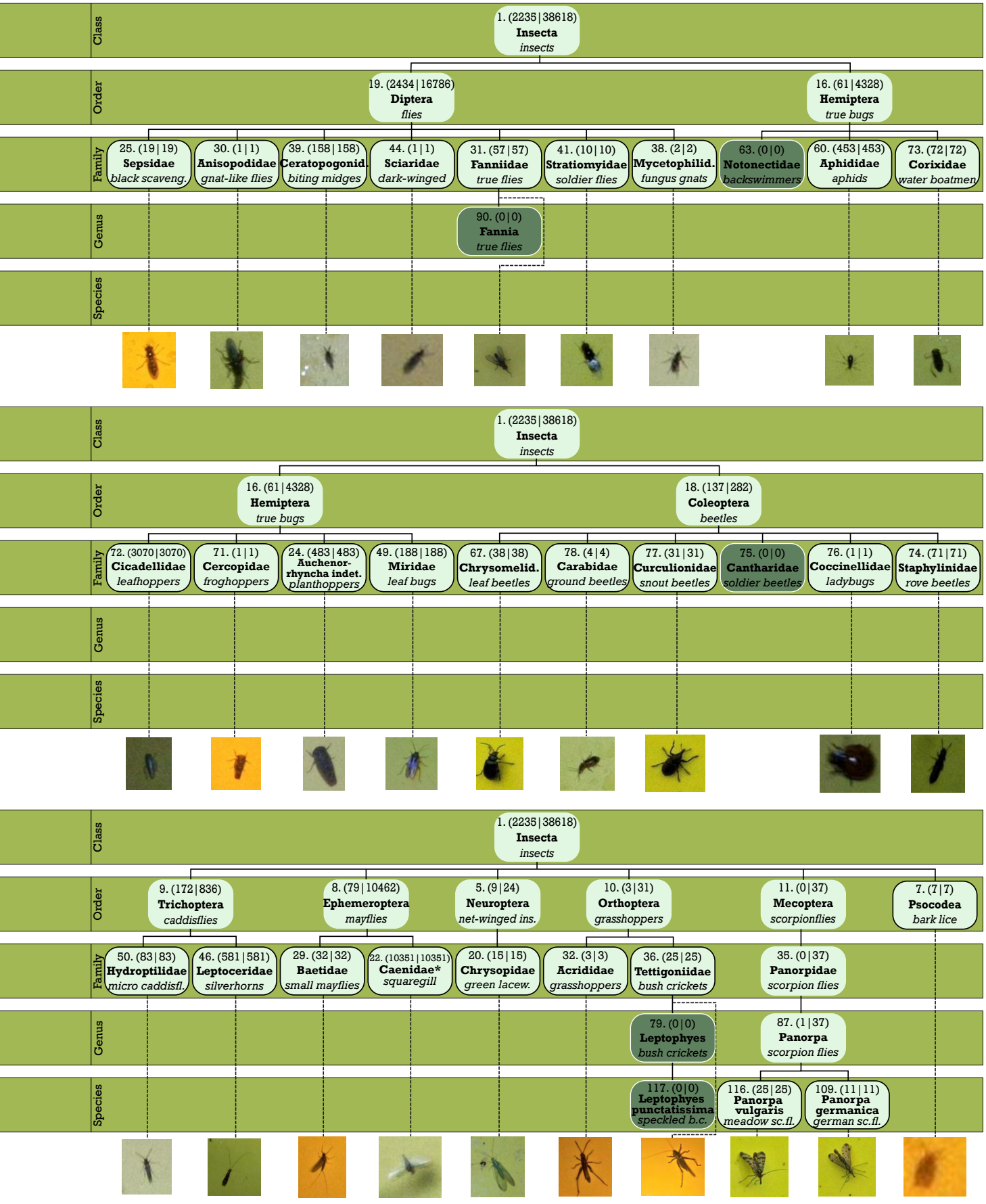


Figure 22: The ontology for the DIOPSIS insect classification dataset. Including number of samples present in the ground truth dataset and an example image from the dataset for the deepest label in that path. (3/3)

B Out of Distribution Measure

To determine the exact definition of when a data sample is marked as 'unsure' and to determine the value of the Maximum Softmax Probability threshold, we perform some exploratory experiments. Firstly, we compare the MSP thresholds of 0.5 and 0.9 and, as can be seen in Figure 23, find that the 0.9 results in a lower percentage of other branch errors. Hence, we decide to use the 0.9 threshold for the MSP threshold.

We also investigate an extension to only the MSP threshold, by including an uncertainty measure similar to bjrege et al. [4] that checks whether a prediction at level l is consistent with prediction at level $l-1$. If the prediction at level l is inconsistent with the prediction at level $l-1$ the predicted label's path ends at the prediction at level $l-1$ and all deeper levels get the 'unsure' label assigned. This guarantees dependency consistency and reduces the severeness of the errors made by the model. For example, the model is less likely to make "other branch' errors if it tries to make a prediciton at species level when there is no label at that level that falls under the assigned family level. Predictions can still end up at other branch, for example if you predict a mosquito species instead of a bee species, however this measure ensures that the predictions made by a model will be consistent with itself. The labels are propagated from the top down, which is intuitively correct, as predictions at higher levels in the ontology tree are easier to make than predictions lower in the tree and thus should be more likely correct.

When using the dependency term for uncertainty we see that precision gets higher already from the start, as the retrieved positive instances are more likely correct. However, we do see that recall gets affected negatively a little bit, as lower level predictions that would be correct might actually get ignored. The biggest impact we see is the percentage of taxon distance errors. As can be seen in Figure 23, the model significantly more often stops at the right taxonomic level for predictions and thus the amount of other branch errors get reduced a lot and we see an increase of 'too general' errors, which are qualitatively less bad. This decrease in error severity and the guarantee for dependency consistent labels is why we decide to use the dependency term in the uncertainty measure.

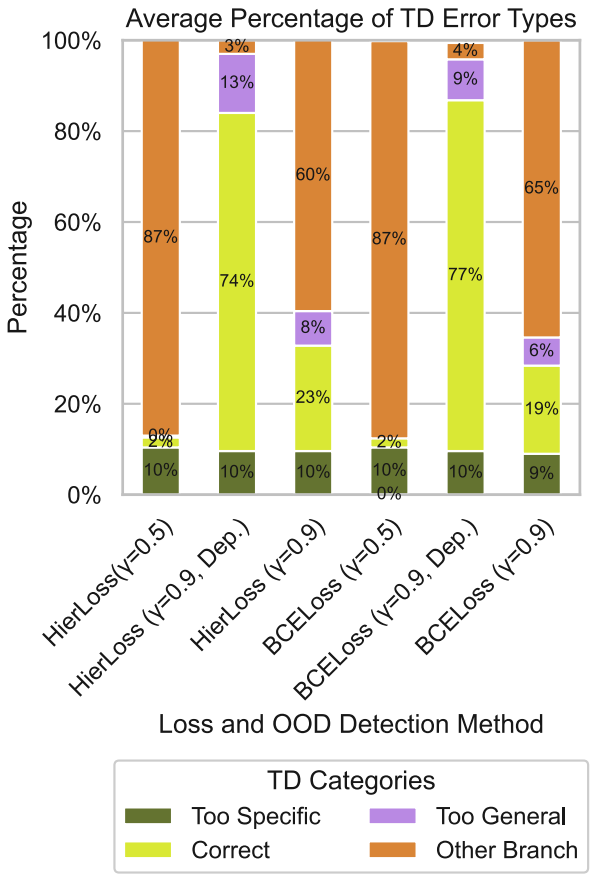


Figure 23: An overview of the percentage of predictions that fall under each of the taxon distance error categories. The two MSP thresholds (γ) and the OOD detection method with and without dependency factor are compared. We can see that the higher threshold, as well as the dependency factor both reduce the percentage other branch errors made and increase the amount of correct deepest label predictions.

C Annotation Environment Non-Experts

On the next pages the annotation guide is given developped for the collection of non-expert annotations for unlabelled DIOPSIS images.

DIOPSIS annotation guide

This document is a work in progress, so if anything is unclear, please add a comment.

Objective

Through this annotation project, we aim to collect more data for the DIOPSIS challenge, by annotating the presence and taxonomic labels of insects and other animals present in DIOPSIS camera images (For an example see Figure 1). As specified on the challenge website, the goal of the DIOPSIS challenge is to develop improved algorithms for the detection and classification of insects in the DIOPSIS image processing pipeline. One of the best methods for improving the prediction capabilities of a machine learning model is by presenting it with more labelled data. However, expert availability is scarce, hence, with this project we aim to crowdsource the annotations and expand upon the already available labelled DIOPSIS datasets for detection and classification.

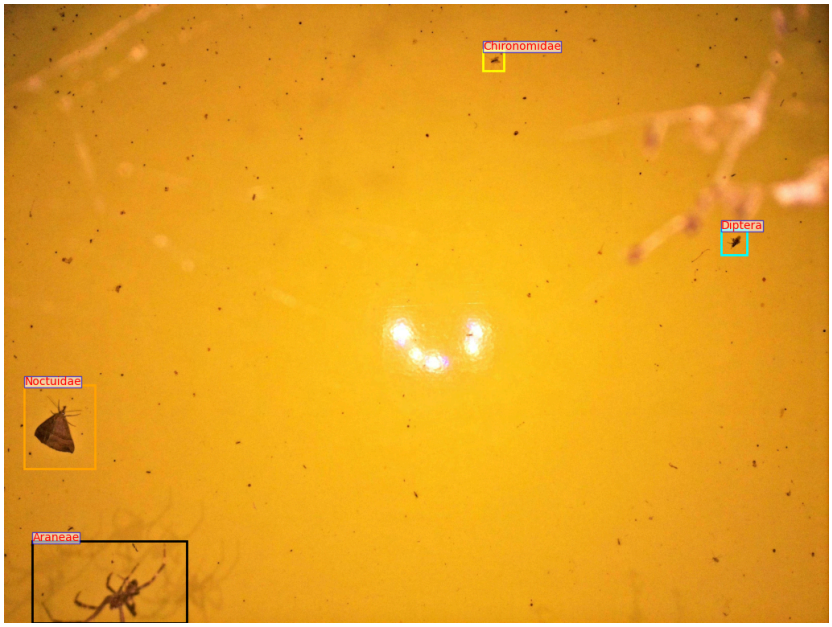


Figure 1: Example of annotated DIOPSIS image

Becoming an annotator

To become an annotator, we will have to manually add you to the project by sending you an invite email. Hence please send an email to heleen.visserman@ru.nl and you will be added to the project as soon as possible.

The annotation platform - Labelbox

We have uploaded unlabelled images collected using the DIOPSIS cameras to Labelbox. Labelbox is an online platform designed to help with data-centric AI and makes the annotation workflow easier.

Using labelbox

There is a lot of documentation available on how to use Labelbox. It is so much you may get lost. To save you some time, we think you can focus on reading the following pages:

- https://docs.labelbox.com/docs/image-annotations#bounding-box
- https://docs.labelbox.com/docs/labeling-queue

Labelling environment

An example of the labelbox environment in which images are annotated can be found in Figure 2.

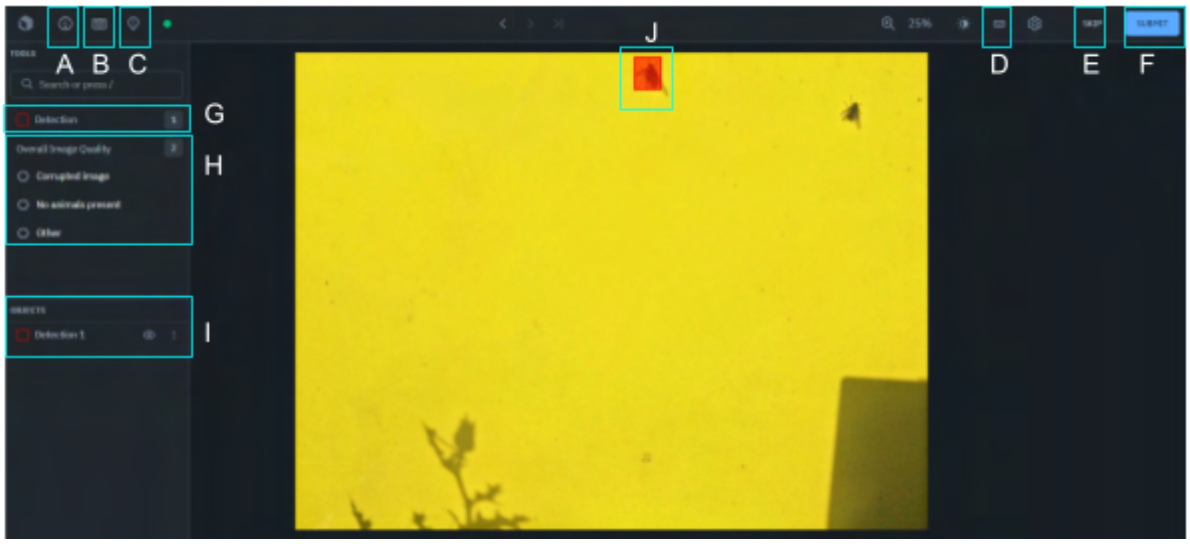


Figure 2: Example of labelling environment

A. **Metadata** - get more information about the specific DIOPSIS image.
B. **Instructions** - Get the labelling instructions (the file you are currently reading).
C. **Issues** - Indicate whether there are specific issues with the image.
D. **Keyboard shortcuts** - There are some nice keyboard shortcuts, here you find an overview.
E. **Skip** - Skips the full image. WARNING: please don't skip an image! (See Best Practices for alternative)
F. **Submit** - Submit your annotations for this image once you have drawn bounding boxes around all animals.
G. **New detection bounding box** - Click (or press 1) to start a new annotation in the image.

H. **Overall Image Quality** - *Optional!* Click (or press 2) to indicate there is something wrong with the *overall image,* for example, there are no animals or the image is corrupted.
I. **Detections** - Overview of all made annotations in this image.
J. **Detection bounding box** - Example of a bounding box visualised in the DIOPSIS image.

A **video** with an example of creating a new annotation can be found here.

Labelling Step by step

You will be presented with one DIOPSIS image at the time. Each DIOPSIS image can contain 0 - many animals. For each image go over the following steps:

1. If the image does not contain animals, please select "*overall image quality*" → "*No animals present*" and submit the image.
2. For each animal draw a bounding box around it and assign the most specific label you are able to give.
    a. Draw a tight **bounding box** around the animal, make sure to include all its parts, including legs, wings and antennae.
    b. Select the most specific taxonomic **label** you can provide for the animal. For a guide to the animals in the ontology, have a look at the Ontology section.
        i. **IMPORTANT**: *Try to be as specific as possible in assigning the label, however do not start guessing. Even labels that just say insecta or gastropoda can contain useful information.*
        ii. **NOTE**: in the currently live version of the annotation environment, the taxonomic tree only goes to order level, meaning that the annotations are less specific.

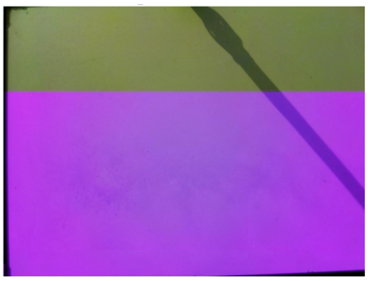
Best practices

- Do not close the tab in which you are performing the annotations until you are done annotating, as else your progress might get lost.
- If the submit button is greyed out, this might mean that you did not assign a label to one or more of your indicated detections.
- If you encounter something wrong with an image that is not covered by the overall image quality tool, you can add an issue using the issue tool (Figure 2C).
- Label all the animals in a picture. Do not only annotate a few of the insects as this will provide the detection model with incorrect information of true negatives.
- Please don't skip an image. Skipping refers to skipping the annotation as a project, not as an individual. These will end up empty in the review queue and will have to be rejected. Rather, use the "*image quality*" -> "*other*" to indicate what is wrong with the image and then submit it.

Data cleaning

Datacleaning is done within Labelbox. Invalid data comprises abnormal situations and is unsuitable for annotating training data. For example, some images get corrupted during transfer, which means the background colour of the image will change. If you encounter such an image, indicate the faultiness of the image with the ‘overall image quality’ tool and submit the annotation.

Exceptions

- **Not a snail, insect or spider** - If you encounter an animal in the image that does not fit the classes Gastropoda, Insecta or Arachnida, you can select the Animalia label and go to the next animal in the image.
    - For example, a frog: 
- **Not an animal** - If you encounter an object that is not an animal, e.g. dirt or a plant, do not annotate it. Only annotate animals.
- **Overlapping animals** - If you encounter two animals overlapping, annotate both. Overlapping bounding boxes could be filtered out later.
- **Too small for identification** - If you encounter a very small animal, and you can still see that it is an animal, give it the appropriate label. If you are not sure whether it is an animal or not, you can ignore it.
    - For example, a very small animal given the label of animal: 
- **Partially visible animal** - If you encounter a partially visible object draw a bounding box around the visible part of the animal.
- **Image is corrupted** - If you encounter an image that looks corrupted, select “*overall image quality*” → “*Corrupted image*”
    - For example, a corrupted image: 
- **Too many insects** - If there are hundreds of insects in an image and it is too challenging to annotate everything, please use the label *“overall image quality”* ->

"other" and mention that there are too many insects. *Please only do this if absolutely necessary, as this means the image will not get annotated anymore.*

Ontology

The ontology used for this annotation project consists of a big hierarchical tree with many possible labels. A PDF version of the taxonomic tree with small descriptions per label can be found here and in the below screenshot.



We also made a labelling aid that provides examples of each of the labels in the tree based on the already annotated ground truth labels. Figure 3 provides a visual example. The project can be cloned here: https://github.com/jacob-kamminga/Diopsis_labeling_aid. **NOTE**: in the currently live version of the annotation environment, the taxonomic tree only goes to order level, meaning that the annotations are less specific than all possible labels available in the tool. It cán however be useful to click through the different suboptions of a certain label to find an insect that looks most like the one you are trying to label.

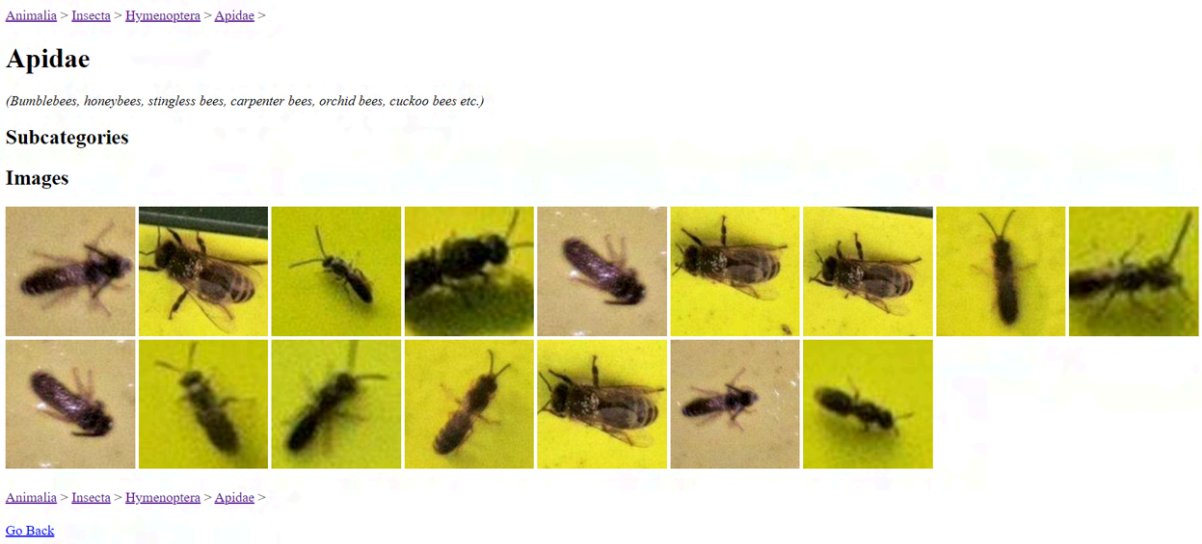


Figure 3: example of labelling aid

Exporting data

An export of the data will be created after several weeks of data collection. The exact week is TBD.