MOSAICO: Open-set hierarchical classification of mosquito species

Mattia Liguoro

Abstract

Mosquitoes are among the primary vectors of several infectious diseases, and monitoring their species composition across different regions is a key step to support early prevention and control measures. However, the manual taxonomic identification of mosquito species is time-consuming and requires highly specialized entomological expertise. In this work, we introduce *MOSAICO*, a deep learning-based support system designed to assist entomologists in mosquito classification. The system automatically assigns specimens to the correct genus, while attempting species-level classification only when sufficient confidence is achieved. To handle uncertainty and improve the model's ability to deal with both known and unseen species, we adopt a modified Evidential Deep Learning (m-EDL) architecture, which provides an explicit estimation of predictive uncertainty. This approach allows for reliable hierarchical classification, supports expert decision-making, and contributes to the development of scalable mosquito monitoring tools.

1 Introduction

In recent years, monitoring mosquito populations has become an increasingly important task for understanding and managing the spread of vector-borne diseases. The composition and relative abundance of mosquito species in a given area directly influence the local risk of transmission, as different species may act as vectors for different pathogens.

Traditionally, mosquito classification is performed by trained entomologists through the morphological analysis of physical specimens. This procedure requires careful examination of anatomical structures such as wing venation, leg markings, and other fine-grained morphological traits. However, many mosquito species share very similar shapes, sizes, and color patterns, making the distinction between species particularly challenging even for experienced taxonomists. Moreover, specimens may sometimes be damaged or incomplete, further complicating the identification process.

The application of machine learning methods offers a promising solution to assist entomologists by automating parts of the classification task. Nevertheless, several challenges emerge when building effective models for this problem. First, the amount of available data is limited, especially at the species level. While genera can usually be assigned with reasonable confidence, reliable species-level annotations require a much higher degree of certainty, and are not always available or consistent across large datasets. Furthermore, the dataset used for training is highly imbalanced: some species are represented by thousands of samples, while others include only a few dozen instances. This imbalance creates a significant risk of introducing bias into the model, causing over-representation of majority classes and poor generalization on underrepresented species.

To address these issues, we propose a hierarchical classification approach in which the model always provides a genus-level prediction, while attempting species-level classification only when sufficient confidence is reached. This selective strategy allows us to mitigate the effects of limited data availability and class imbalance, while still providing fine-grained information when possible. To effectively quantify uncertainty and enable robust decision making, we adopt a modified

- Evidential Deep Learning (m-EDL) architecture, which explicitly models the uncertainty associated with each prediction. This framework is designed to serve as a practical decision support tool for entomologists engaged in large-scale mosquito monitoring and identification tasks.

2 Project's Goals

As previously introduced, the main goal of this project is to develop a support device that can assist entomologists in the classification of mosquito specimens. By automating the most time-consuming parts of the identification process, the system aims to significantly speed up the classification task, reducing both the human and financial resources typically required for large-scale mosquito monitoring.
Beyond the direct reduction of workload for entomological experts, the ultimate and most important goal of the project is to enable the rapid collection of reliable data on the distribution and abundance of mosquito species across different geographical areas. This information is crucial for public health authorities, as it allows for the early detection of potentially dangerous vector populations and the timely implementation of preventive measures.
By deploying the proposed device directly in the field, equipped with the developed software, it becomes possible to carry out mosquito surveys with minimal effort. This approach facilitates the creation of a global surveillance network capable of providing continuous and up-to-date information on mosquito populations, ultimately contributing to the prevention and control of vector-borne diseases.

3 Road Map

The starting point of this project was based on an m-EDL architecture with an EfficientNet backbone. EfficientNet is a family of convolutional neural networks that introduced a compound scaling method, jointly optimizing network depth, width, and resolution, resulting in a highly efficient and scalable architecture compared to traditional CNNs. This allowed us to start with a lightweight ($\sim$ 30 mln parameters) yet performant model suitable for our initial experiments. The training of this first model was carried out in two phases. Initially, only the classification head was fine-tuned on our dataset, while keeping the backbone frozen. Subsequently, we unlocked and fine-tuned the last few layers of the backbone, allowing the model to adapt its high-level feature extraction to our specific domain. Simple data augmentations such as random rotations, random crops, sharpening, and minor color variations were applied to increase data variability and reduce overfitting. The loss function used in this phase was a standard cross-entropy (CE) loss, with the objective of predicting only the species labels, without yet leveraging the available genus information.
Following this initial experimentation, we explored a more advanced model inspired by recent transformer-based architectures: ConvNextV2, developed by Meta.
Just after the backbone, we added a genus classifier with 4 output classes corresponding to the genera available in our dataset. This allowed us to assess the achievable accuracy at the genus level with a more powerful model. Subsequently, we moved towards a multitask learning approach, training a model with two heads: one classifier for genus and one for species. To address the class imbalance and improve model calibration, we employed the *inverse focal loss* as proposed by Wang et al. [9]. This loss enhances the model's sensitivity to well-classified (high-confidence) samples, promoting better calibration in subsequent stages.
In addition to the inverse focal loss, a penalty term was introduced to enforce taxonomic hierarchy consistency between genus and species predictions. Specifically, a penalty was applied when the predicted species did not belong to the predicted genus. For example, if the model predicted the genus *Aedes* but the species *Pipiens*, a penalty was added since *Pipiens* does not belong to the genus *Aedes*.

After training, the model's outputs for both genus and species were further calibrated using temperature scaling. This post-hoc calibration technique allowed for a more accurate interpretation of the model's confidence scores, especially important in critical applications such as disease vector monitoring.
The results obtained through this approach were highly encouraging. At the genus level, we achieved inference accuracies up to $98 - 99\%$. At the species level, peak accuracies reached approximately 95%. However, the strong dataset imbalance remained a limiting factor: for species with few images or highly similar morphology to others, the accuracy often dropped to around 50%. This observation highlighted the importance of avoiding overconfident but incorrect species predictions.
Driven by these findings, we concluded that in ambiguous cases it would be preferable to output an *uncertain* species prediction, leaving the final identification to the entomologist. To support this strategy, we analyzed the distribution of logits produced by the classifiers, observing that, for correctly classified samples, their distributions often approximated a Gaussian. This enabled the definition of a rejection threshold based on the relation $th = \mu - 2\sigma$. Although conceptually promising and well-calibrated in certain cases, this method did not yet provide sufficiently robust results for deployment. However, we foresee that as the device is deployed in the field and new entomologist-labeled samples are collected, this framework could be extended and refined for future versions of the system.
We therefore returned to an m-EDL architecture focused on species-level classification, while extending and improving the previous model. In this updated version, we also incorporated genus prediction and enforced hierarchical consistency during both training and inference phases.

4 Dataset

The specimens used for this study were either collected directly from the field or reared in controlled laboratory environments within universities or research centers.
During data collection, it was observed that field-captured mosquitoes generally exhibit more irregular and damaged features compared to those reared in laboratory conditions. This includes variations in wing integrity, scale patterns, and overall preservation quality. Additionally, even when considering specimens belonging to the same species and sex, noticeable morphological differences were found among individuals captured in different geographical areas. Such variability is likely attributable to environmental factors, local adaptation, and genetic diversity within species.
To address this intrinsic heterogeneity and avoid overfitting to local populations, a portion of the specimens from certain locations was deliberately excluded from the training set. These excluded samples were instead used to assess the model's generalization capability, ensuring that its performance is not solely driven by overrepresented local morphologies.
In the following, the characteristics of the dataset are summarized, reporting the number of specimens, their taxonomic assignment, and the corresponding collection sites.

To obtain an intuitive overview of the spatial distribution of collection sites for each species, we generated an interactive map using the Python library `folium`[1], which builds on the Leaflet JavaScript framework. The script automatically places markers for every (species, locality) pair in the dataset and colors them by species.
In Figure 1, we show a static screenshot of the map. An interactive version of the map is available online at: `mosquito_map.html`.

[1] `https://python-visualization.github.io/folium/`

Species	Locations	Specimens
Aedes aegypti	Bangkok, Brazil, Ecuador, ISS, South Africa, University of Milan, Vietnam	1390
Aedes albopictus	Bigarello (Mantua), Grottaglie (Brindisi), Guardavalle (Catanzaro), ISS, IZS Lombardia, IZS Sardinia, IZS Umbria and Molise, Mezzolombardo (Trento), Montecchio (Terni), Rome, San Michele all'Adige, University of Pavia	1224
Culex pipiens	Casalgrande, Cortefranca, Forlanini, ISS, IZS Sardinia, Mezzolombardo (Trento), Narni (Terni), San Michele all'Adige, Trino (Vercelli), University of Milan, Valcannuta, Valenza (Alessandria), Zafferana Etnea (Catania)	1143
Anopheles maculipennis	Berni (Trento), Novara, Sant'Angelo Lomellina (Pavia), Trino (Vercelli), Vespolate	1109
Aedes caspius	Alluvioni Piovera, Ferrara, Gattinara, Grosseto, Oristano, Palau, Voghera, Grosseto Horse Center	1054
Aedes koreicus	Canal San Bovo (Trento), ISS, Mezzolombardo (Trento), Sospirolo (Belluno), University of Milan	926
Culex theileri	Berni	657
Aedes vexans	Argenta, Bigarello, Capriva del Friuli (Gorizia), Concordia Sagittaria (Venice), Concordia sulla Secchia, Fagagna (Udine), Guarda Veneta (Rovigo), Lignano (Udine), Piacenza, Castel Focognano (Arezzo), Verona	554
Aedes geniculatus	Belluno, Cadeo, Campagna Lupia (Venice), Campli (Teramo), Fagagna (Udine), Fondi, Gibellina (Trapani), Martellago, Mestre, Piacenza, San Canzian d'Isonzo (Gorizia), Teodone (Bolzano), Teramo, Buonconvento (Siena)	377
Culex mariae	Algeria, Crete, Custonaci, Gagliano del Capo, Marina di Lambrone, Marina di Zambrone, Nuoro, Pantelleria, Platja d'Aro, Ponza, Ustica	265
Culex hortensis	Amelia, Arezzo, Asti, Bassano Romano, Dosoledo (Belluno), Feltre, Friuli, Leonessa (Rieti), Matelica, Pietrapiana, Verbano-Cusio-Ossola, Revine Lago (Treviso), Segni, Sicily, Sila, Teramo, Tivoli	255
Anopheles algeriensis	Puglia	195
Culiseta longiareolata	Avezzano, Cassano Spinola, Cerveteri, Rome Verano Cemetery, Foresta Umbra (Foggia), Frosinone/Sora, ISS, Grosseto, Isola del Giglio, Latina, Magliano (Tuscany), Olmedo/Oristano, Matelica, Pisticci (Matera), Spoleto, La Quercia	176
Culex rusticus	Circeo National Park, Insugherata Nature Reserve	172
Culex detritus	Cabras, Caorle (Venice), Grosseto, Lecce, Oristano, San Cataldo, Terralba, Torre dell'Orso (Salento)	135

Table 1: Species, sampling locations (excluding unknown locations), and total specimens.

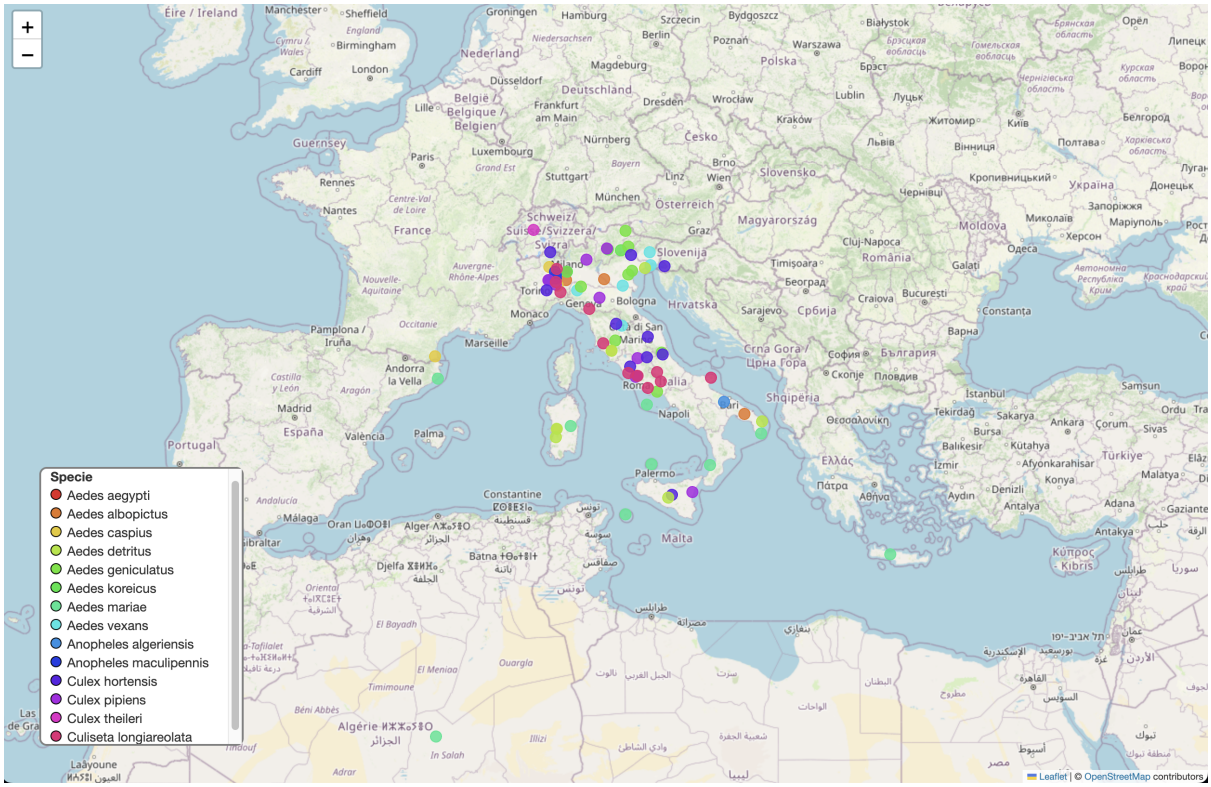
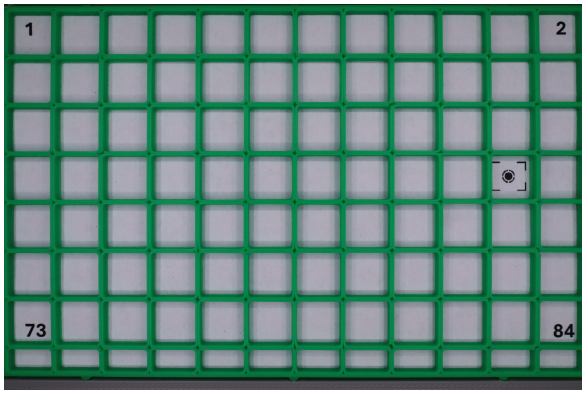


Figure 1: Screenshot of the interactive map generated with `folium`. (zoom-in Italy)

The mosquito specimens were positioned by entomologists into standardized grid frames, as shown in Figure 2. Each grid consists of multiple cells, and every cell contains only one specimen. Both males and females are present within the same frame and from the same taxonomic class.

In order to automate the detection and extraction of individual mosquitoes from the grid images, an object detection model based on the YOLO (You Only Look Once) [7] architecture was applied. YOLO is able to identify and segment each mosquito located in the grid cells, assigning bounding boxes to each valid detection. An example of the full grid before and after detection/segmentation is shown in Figure 2.



Empty grid structure used for positioning.

Grid filled with specimens and segmented by YOLO.

Figure 2: Example of the grid system used for specimen positioning and segmentation.

In order to improve the generalization ability of the model and make it robust to variability

in image acquisition conditions, a strong data augmentation pipeline was applied during training. This choice was motivated by early observations on preliminary models: initial versions of the network tended to rely heavily on color-based features, leading to severe misclassifications when the acquisition conditions (lighting, contrast, saturation) varied between training and test images. In some cases, the model was hallucinating species solely based on differences in background tone or image brightness, which are irrelevant features from a biological standpoint.
To address this issue and force the model to focus on more stable morphological features, a wide set of aggressive augmentations was applied to the training data. These transformations included both geometric distortions (such as random crops, rotations, affine transformations, perspective changes, grid and elastic deformations) and photometric variations (such as drastic changes in hue, saturation, brightness, contrast, channel shuffling, and random gamma correction). Additional noise was injected through Gaussian noise, motion blur, Gaussian blur, and JPEG compression artifacts. Finally, partial occlusions and local defects were simulated via coarse dropout and random sharpening.
The purpose of such heavy augmentations was to exploit the large capacity of the chosen model, which includes approximately 660 million parameters, and encourage the network to extract deeper and more abstract features, rather than relying on superficial and unstable cues present in the training set. This strategy aimed to mimic real-world conditions, where specimens may be photographed under diverse lighting setups, different cameras, or with varying levels of image degradation, ensuring that the model remains robust across domains.

Figure 3: Data augmentation and transformations applied on train dataset

5 Architecture and model

5.1 Hierarchical multitask architecture

The classification problem addressed in this work is inherently hierarchical, reflecting the biological taxonomy of mosquitoes: each specimen belongs to a genus, and within each genus, to a specific species. To effectively leverage this structure, we adopt a hierarchical multitask architecture that performs both genus- and species-level classification in parallel, using a shared visual backbone and separate task-specific heads.
The overall model consists of three main components: a high-capacity feature extractor (ConvNeXtV2-Huge, see next section), a genus classification head trained with standard weighted cross-entropy loss, and a species classification head trained with a modified evidential loss (m-EDL) that explicitly models uncertainty and open-set scenarios. Both heads receive input from the same backbone, enabling the model to benefit from shared representations while allowing each task

to specialize.
This architecture enables the classifier to produce genus-level predictions for all inputs, while attempting species-level predictions only when confident enough. In addition, it naturally supports the detection of inconsistent or biologically implausible predictions—such as species not belonging to the predicted genus—which are penalized through a dedicated hierarchy-consistency loss. This multitask formulation reflects the real-world usage scenario of the system, where partial predictions (e.g., genus only) are still informative and preferable to incorrect species assignments.

5.2 Backbone architecture: ConvNeXtV2-Huge

The backbone used in our model is ConvNeXtV2-Huge, as already mentioned in the previous paragraphs, a state-of-the-art convolutional architecture that combines the strengths of traditional CNNs with some of the structural innovations introduced by vision transformers. Unlike pure transformer models, ConvNeXtV2 maintains a strong inductive bias, meaning that its architectural design incorporates prior knowledge about image regularities, such as the local organization of pixels and the recurrence of patterns across spatial locations. These built-in assumptions guide the learning process and significantly improve the model's ability to generalize in scenarios with limited training data or domain shifts. This inductive bias is particularly advantageous in scenarios where training data is limited or highly imbalanced, as it encourages the network to focus on consistent low-level features such as wing texture, leg segmentation, or body proportions, rather than relying on global color distributions or contextual cues that may not generalize across acquisition conditions.

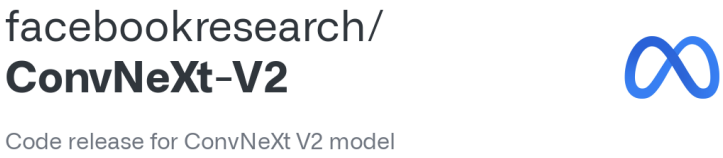
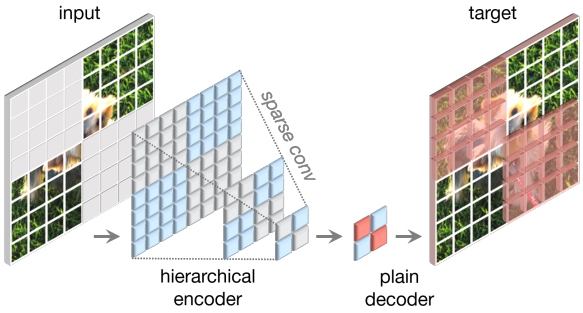


Figure 4: ConvNeXtV2 by Meta corporation

ConvNeXtV2 extends the original ConvNeXt design by integrating ideas from masked autoencoders and self-supervised learning. Specifically, it incorporates a hierarchical encoder-style structure inspired by MAE (Masked Autoencoders), where early layers capture fine-grained features and later stages progressively abstract semantic content. During pretraining, the model is exposed to masked image patches and learns to reconstruct missing content, encouraging the formation of rich latent representations even in the absence of full visual context. This approach improves robustness to partial occlusion and damaged specimens—a common issue in field-collected mosquito samples.
We selected the "Huge" variant, comprising approximately 660 million parameters, which was pretrained on ImageNet-22K and then progressively fine-tuned on our domain using a staged learning rate strategy.



Fully Convolutional Masked Autoencoder Framework

Type	Backbone	size	#param	FLOPS	Val acc.
Conv	Efficient V2-XL	480^2	208M	94.0G	87.3
	ConvNeXt V1-XL	384^2	350M	179.0G	87.8
Hybrid	CoAtNet-4	512^2	275M	360.9G	88.1
	MaxViT-XL	384^2	475M	293.7G	88.5
	MaxViT-XL	512^2	475M	535.2G	88.7
Trans	MViTV2-H	384^2	667M	388.5G	88.6
	MViTV2-H	512^2	667M	763.5G	88.8
Conv	ConvNeXt V2-H	384^2	659M	337.9G	88.7
	ConvNeXt V2-H	512^2	659M	600.7G	**88.9**

Comparison of state-of-the-art visual backbones on ImageNet-1K test set

5.3 Theoretical foundations: Evidential Deep Learning (EDL)

Traditional neural classifiers produce point estimates of class probabilities via softmax activation. However, softmax scores are often poorly calibrated and prone to overconfidence, particularly for out-of-distribution or damaged samples [2]. Evidential Deep Learning (EDL) [8] addresses this by introducing a Bayesian formulation that models uncertainty directly on the output space.

5.3.1 Dirichlet-based output

In EDL, the neural network does not predict class probabilities directly, but instead outputs non-negative *evidence values* $e_k \geq 0$ for each class k (using any non-negative activation function , e.g. Softplus()). These evidences parameterize a Dirichlet distribution over the categorical simplex:

$$\alpha_k = e_k + 1.$$

The expected class probabilities are then given by the mean of the Dirichlet distribution:

$$\mathbb{E}[p_k] = \frac{\alpha_k}{S}, \quad S = \sum_{j=1}^{K} \alpha_j.$$

Thus, instead of producing fixed probability scores, the network generates an entire probability distribution over class probabilities, allowing uncertainty quantification.

5.3.2 Bayesian risk minimization

In Evidential Deep Learning (EDL), training is formulated under the framework of Bayesian decision theory. Rather than minimizing the empirical loss directly on the predicted point estimates, the network minimizes the expected loss (Bayes risk) under the uncertainty modeled by the Dirichlet distribution. Given the true one-hot label vector $\mathbf{y} \in \{0,1\}^K$ and the predicted Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ with parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, the squared error risk is defined as:

$$\mathcal{R}(\boldsymbol{\alpha}, \mathbf{y}) = \mathbb{E}_{\mathbf{p}\sim\mathrm{Dir}(\boldsymbol{\alpha})} \left[\|\mathbf{y} - \mathbf{p}\|^2 \right].$$

Explicitly, this expectation can be written as:

$$\mathcal{R}(\boldsymbol{\alpha}, \mathbf{y}) = \int_{\mathbf{p}\in\Delta^{K-1}} \|\mathbf{y} - \mathbf{p}\|^2 \cdot \mathrm{Dir}(\mathbf{p} \mid \boldsymbol{\alpha})\, d\mathbf{p}, \tag{1}$$

$$\mathrm{Dir}(\mathbf{p} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} p_k^{\alpha_k - 1}, \quad B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}. \tag{2}$$

where Δ^{K-1} denotes the $(K-1)$-dimensional probability simplex and $B(\boldsymbol{\alpha})$ is a normalization constant and is called multivariate beta function.
The $(K-1)$-dimensional probability simplex Δ^{K-1} is defined as the set of all valid categorical probability distributions over K classes:

$$\Delta^{K-1} = \left\{ \mathbf{p} \in \mathbb{R}^K \,\middle|\, p_k \geq 0, \sum_{k=1}^{K} p_k = 1 \right\}.$$

Geometrically, it represents a convex polytope whose vertices correspond to deterministic assignments (i.e., one class has probability 1 while all others have 0), while points inside the simplex represent probabilistic mixtures over the classes. For instance, in the case of $K = 3$ classes, the simplex reduces to a triangle in 2D space. Within the Evidential Deep Learning framework, the Dirichlet distribution models a probability density over this simplex, capturing both the predicted class probabilities and their associated epistemic uncertainty.

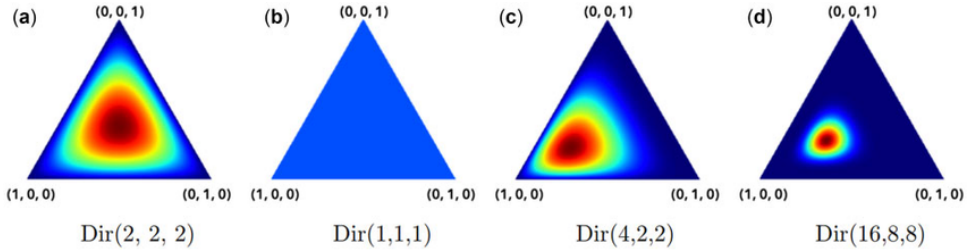


Figure 5: The 2-simplex ($K = 3$ classes). Each point inside the triangle encodes a probability vector (p_1, p_2, p_3) with $p_k \geq 0$ and $\sum_{k=1}^{3} p_k = 1$. Vertices correspond to full certainty on a single class, edges to mixtures of two classes, and interior points to three-way mixtures. In Evidential Deep Learning, the Dirichlet distribution places a density over this simplex, capturing both class probabilities and epistemic uncertainty.

Getting back to the explicit form of the equation (1), with some algebraic manipulation, we reach the following form:

$$\mathcal{R}(\boldsymbol{\alpha}, \mathbf{y}) = \sum_{k=1}^{K} (y_k - \mathbb{E}[p_k])^2 + \sum_{k=1}^{K} \mathrm{Var}[p_k].$$

Using the moments of the Dirichlet distribution and moments' identity:

$$\mathbb{E}[p_k] = \frac{\alpha_k}{S}, \quad \mathrm{Var}[p_k] = \frac{\alpha_k(S - \alpha_k)}{S^2(S+1)}, \quad S = \sum_{j=1}^{K} \alpha_j,$$

we can make it even more explicit:

$$\mathcal{R}(\boldsymbol{\alpha}, \mathbf{y}) = \sum_{i=1}^{K} \left(y_i - \frac{\alpha_i}{S}\right)^2 + \frac{\alpha_i(S - \alpha_i)}{S^2(S+1)} = \sum_{i=1}^{K} (y_i - \hat{p}_i)^2 + \frac{\hat{p}_i(1 - \hat{p}_i)}{S+1}, \quad \hat{p}_i = \frac{\alpha_i}{S}$$

Thus, the risk naturally decomposes into:

- A data-fitting term that measures the squared deviation between predicted class expectations and true one-hot labels.

- A variance term that encourages the model to express higher uncertainty when not enough evidence is reached, thereby avoiding overconfident misclassifications.

In practice, EDL directly minimizes this risk $\mathcal{R}(\boldsymbol{\alpha}, \mathbf{y})$, which can be seen as a generalized mean squared error with an implicit uncertainty regularization arising from the Dirichlet variance.

5.3.3 Regularization

To avoid pathological solutions where the network produces arbitrarily high evidence, a Kullback-Leibler (KL) divergence regularization term is added, penalizing deviation from a flat Dirichlet prior:

$$\mathcal{L}_{\mathrm{KL}} = D_{\mathrm{KL}}\left(\mathrm{Dir}(\boldsymbol{\alpha}) \| \mathrm{Dir}(\mathbf{1})\right).$$

The final loss function used to train the network is:

$$\mathcal{L}_{\mathrm{EDL}} = \mathcal{R}(\boldsymbol{\alpha}, \mathbf{y}) + \lambda_t \cdot \mathcal{L}_{\mathrm{KL}},$$

where λ_t is a scaling factor that may be annealed during training.

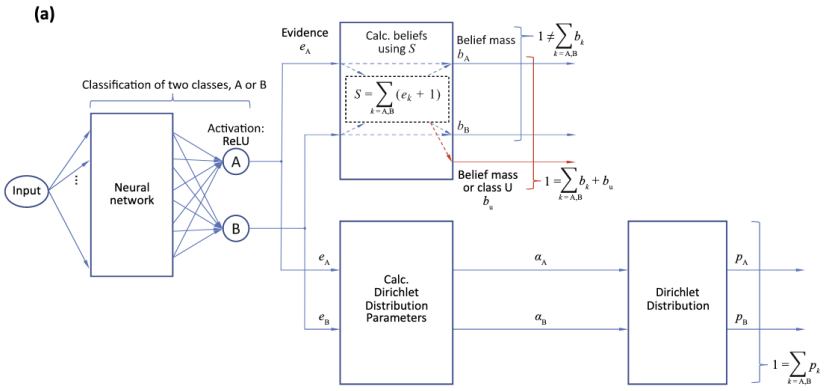


Figure 6: Schematic representation of the EDL architecture for a two-class classifier.

5.4 Extension to Modified Evidential Deep Learning (m-EDL)

While standard Evidential Deep Learning (EDL) captures predictive uncertainty by modeling a Dirichlet distribution over K known classes, it assumes that every input belongs to one of them. This is unrealistic in real-world settings like mosquito classification, where unknown species may be encountered. Modified Evidential Deep Learning (m-EDL) [6] extends the Dirichlet output to include an additional *unknown* class u, leading to open-set awareness.

5.4.1 Dirichlet Extension with Unknown Class

Let the model output evidence values for $K+1$ classes:

$$\mathbf{e}_i^+ = (e_{i1}, \ldots, e_{iK}, e_{iu}), \quad e_{ij^+} \geq 0,$$

with corresponding Dirichlet parameters $\alpha_{ij^+} = e_{ij^+} + 1$, for $j^+ \in \{1, \ldots, K, u\}$.
The expected probability vector becomes:

$$\mathbb{E}[p_{ij^+}] = \frac{\alpha_{ij^+}}{S_i^+}, \qquad S_i^+ = \sum_{j^+} \alpha_{ij^+}.$$

5.4.2 m-EDL Loss Function

The loss function $\mathcal{L}_{\text{species}}^{\text{m-EDL}}(\Theta)$ is defined as the expected squared error between the ground-truth label $\mathbf{y}_i^+$ (one-hot encoded) and the predicted Dirichlet mean, regularized with the Dirichlet variance:

$$\mathcal{L}_{\text{species}}^{\text{m-EDL}}(\Theta) = \sum_{j^+} \left\{ \left(y_{ij^+} - \mathbb{E}[p_{ij^+}]\right)^2 + \text{Var}[p_{ij^+}] \right\}.$$

the loss encourages the model to express uncertainty (low evidence) when unsure, and concentrates belief (high evidence) only for confident predictions. No fixed evidence is imposed on the unknown class: the model learns to allocate it dynamically.

5.5 Adoption in our hierarchical multitask architecture

Building on these foundations, we implemented a hierarchical multitask architecture for mosquito classification.
The system consists of:

- A shared ConvNeXtV2-Huge backbone pretrained on ImageNet-22K and then fine-tuned on our dataset.
- A genus classifier trained via standard weighted cross-entropy over 4 genera.
- A species classifier trained with m-EDL, predicting K known species plus one explicit unknown class.

The final loss combines multiple terms:

$$\mathcal{L}_{\text{total}} = \alpha \cdot \left(\mathcal{L}_{\text{genus}} + \mathcal{L}_{\text{species}}^{\text{m-EDL}} \right) + (1 - \alpha) \cdot \mathcal{L}_{\text{hierarchy}},$$

$$\mathcal{L}_{\text{hierarchy}} = \exp^{D} - 1, \quad D = \begin{cases} 0 & \text{if hierarchically consistent} \\ 1 & \text{otherwise} \end{cases}$$

where $\mathcal{L}_{\text{genus}}$ is the usual cross-entropy, and the last term penalizes biologically inconsistent genus-species combinations [1] , all weighted by a hyperparameter α.

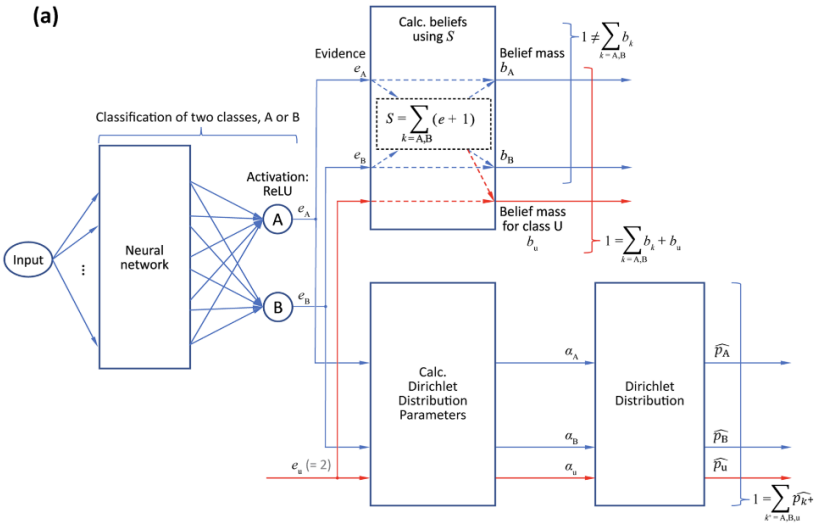


Figure 7: Schematic representation of the m-EDL architecture for a two-class classifier.

5.6 Training Optimization Strategies

In this section, we describe the optimization techniques adopted during model training to ensure stable convergence, efficient fine-tuning, and robust generalization, particularly when dealing with very large-scale pretrained models (ConvNeXtV2-Huge, $\sim$ 660 million parameters). Without the adoption of these strategies, performing multi-phase fine-tuning would have been essentially infeasible due to optimization instability, gradient explosion, or overfitting.

5.6.1 Progressive Block-wise Learning Rate Scheduling

Fine-tuning large pretrained models with uniform learning rates across all layers often leads to unstable optimization, especially when the number of trainable parameters is very high. To address this, we adopted a simple yet effective *block-wise learning rate partitioning* strategy.
The model parameters were divided into three groups:

- **Task-specific heads:** both the genus and species evidential classifiers.
- **High-level backbone layers:** corresponding to the last ConvNeXt stage (*stage 4*, i.e. `backbone.stages.3` in code).
- **Earlier backbone layers:** lower-level pretrained feature extractors.

For each group, different learning rates and weight decays were assigned:

- Task-specific heads used the base learning rate.
- Stage 4 used a reduced learning rate (typically 0.1× or 0.2×).
- Earlier layers used strongly dampened rates (down to 0.04×) and lower weight decay.

This allowed high-level features to adapt progressively while preserving the stability of earlier representations.
To avoid sudden jumps in loss and facilitate smoother convergence, all learning rate adjustments (including warmup and decay) were applied at *per-batch granularity* rather than at epoch steps. This fine-grained schedule significantly improved training stability, particularly during phase transitions where new parameter groups were progressively unlocked.
The full fine-tuning proceeded in three phases:

- **Head-only:** training only the classification heads.
- **Last-stage:** unlocking also the final backbone stage.
- **Full fine-tuning:** updating all layers with discriminative learning rates.

This progressive strategy was crucial to achieve stable optimization across hundreds of millions of parameters while avoiding catastrophic forgetting.

5.6.2 Optimizer: AdamW with Parameter Groups

The optimizer adopted throughout all training phases was *AdamW* [4], applied directly to the parameter groups defined by the block-wise learning rate partitioning. Each group (heads, last-stage, earlier layers) was assigned independent learning rates and weight decays.
AdamW was chosen after empirical comparison with alternative optimizers (SGD and classical Adam), and proved consistently more stable and efficient in this fine-tuning scenario for several reasons:

- **Large-scale backbone fine-tuning:** adaptive optimizers like AdamW handle very large pretrained models better than plain SGD, especially when parameter magnitudes vary significantly across layers.
- **Highly imbalanced dataset:** the severe class imbalance in species-level labels benefits from AdamW's ability to dynamically adjust learning rates for rare and frequent classes without manually rebalancing gradient magnitudes.
- **Decoupled weight decay:** In contrast to classical Adam, AdamW applies weight decay independently from the adaptive updates, preventing implicit biases that would otherwise destabilize training on the pretrained backbones.
- **Small batch sizes:** due to memory constraints with 660M parameters, batch sizes remained small. Adaptive optimizers like AdamW are known to converge more reliably than SGD in such low-batch regimes, avoiding sharp gradient noise accumulation.

5.6.3 Learning Rate Scheduling: Linear Warmup and Cosine Annealing

To further stabilize training, we employed a hybrid learning rate schedule composed of:

- **Linear warmup:** during the first epochs, learning rates were gradually increased from zero to target values, preventing sudden large updates at initialization.
- **Cosine annealing:** after warmup, learning rates decayed following a cosine schedule, promoting smooth convergence to wide minima known to generalize better [3].

5.6.4 Mixed Precision Training: AMP with Gradient Scaling

Given the computational cost of training models with hundreds of millions of parameters, *Automatic Mixed Precision (AMP)* [5] was applied to leverage Tensor Core acceleration and significantly reduce both GPU memory usage and training time.
However, mixed precision introduces risks of underflows or overflows in gradients. To address this, we integrated dynamic *Gradient Scaling* (via `GradScaler`) to automatically adjust scaling factors during training, ensuring numerical stability while preserving the benefits of mixed precision.

6 Training and validation analysis

In this section, we report the main training and validation trends observed throughout the learning process. We analyse both the overall loss behaviour and the task-specific accuracy curves, with particular attention to model stability, generalisation, and potential overfitting.

6.1 Loss and accuracy trends

Figure 8 shows a smooth and consistent decrease in training and validation loss, with no signs of divergence. This indicates stable convergence and the absence of overfitting, even in the later stages of training. Genus-level and species-level accuracy curves, shown in Figures **??** and **??**, both exhibit monotonic improvement and high final accuracy. The genus classifier saturates earlier, as expected from the reduced granularity of the task, while the species head requires more epochs to converge, reflecting the increased complexity of fine-grained classification.

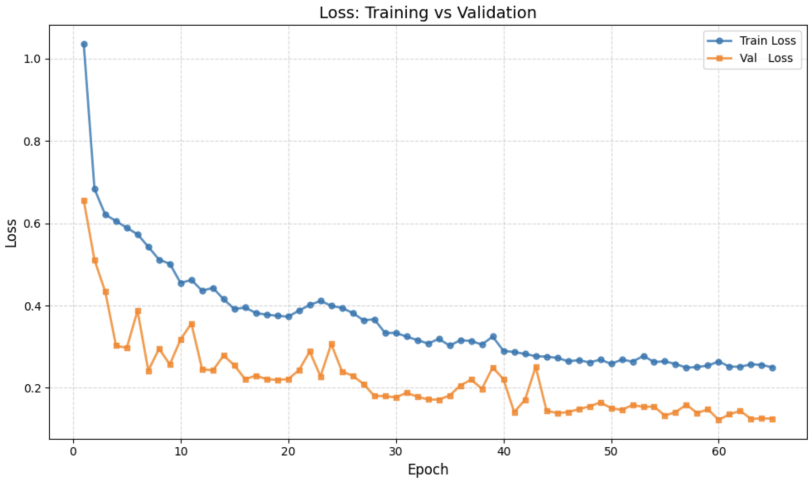


Figure 8: Training and validation loss.

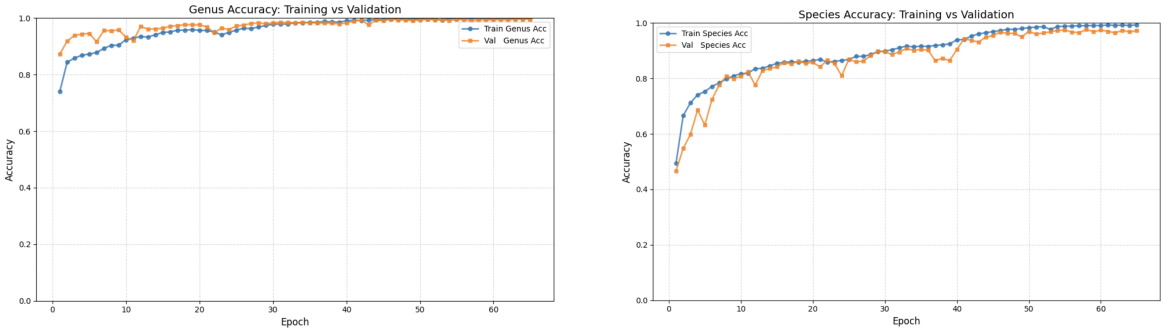


Figure 9: Accuracy trends on training and validation sets for genus (left) and species (right).

6.2 Consistency, generalization, and calibration effects

The close match between training and validation accuracy at both genus and species levels confirms good generalization and suggests that the regularization strategies (data augmentation, dropout, and multitask formulation) effectively prevent overfitting. The slight advantage of validation loss over training loss is typical in models trained with aggressive regularization, and reflects the stochastic nature of data augmentation in the training pipeline. Importantly, the joint optimization of genus and species heads leads to consistent hierarchical behavior during training: genus-level predictions remain stable even when species-level accuracy is still improving.

7 Results and considerations

This section presents a comprehensive evaluation of the proposed hierarchical classifier in terms of classification performance, calibration reliability, and open-set detection.

7.1 Genus and species classification accuracy

The model achieves excellent predictive performance on both classification heads. At the genus level, accuracy reaches **98%** on the test set, indicating that the backbone effectively captures coarse-grained morphological patterns. At the species level, the model reaches a peak validation accuracy of **90%**, confirming the feasibility of fine-grained classification in an open-world context. Nonetheless, some drop in accuracy is observed for rare species, for damaged specimens, or

for individuals captured under acquisition conditions significantly different from those prevailing in the training distribution. This performance degradation reflects both intra-class variability and potential domain shift, and it underscores the importance of uncertainty modeling.

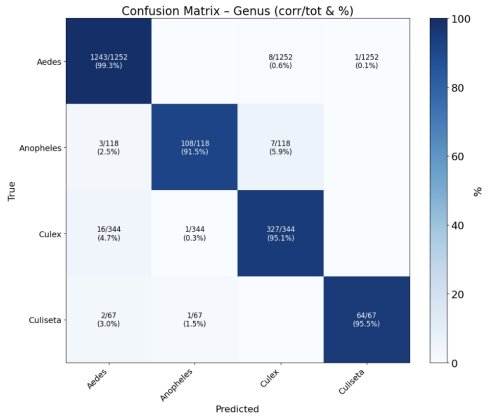


Figure 10: Confusion matrix – Genus

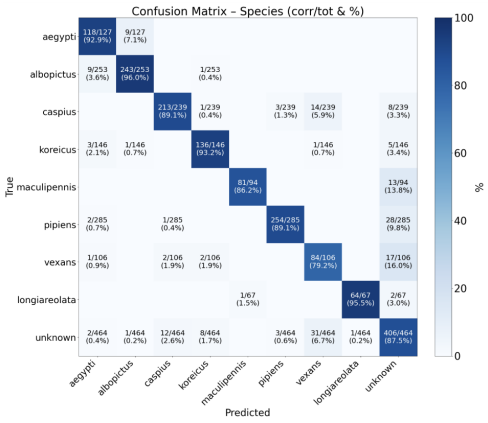


Figure 11: Confusion matrix – Species

The hierarchical structure of the model plays a crucial role in this result. Genus predictions remain robust even when species-level confidence is low, and species misclassifications typically occur within the same genus—confirming the biological plausibility induced by the hierarchy-consistency loss.

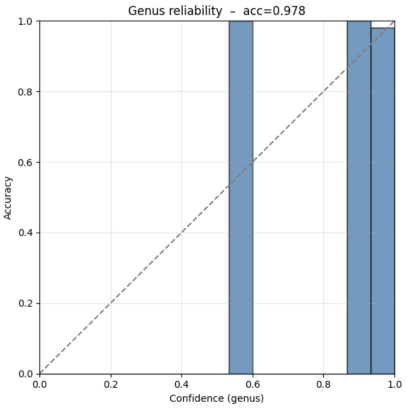
Genus		
Genus	**Correct/Total**	**Acc. (%)**
Aedes	1 243 / 1 252	99.3
Anopheles	108 / 118	91.5
Culex	327 / 344	95.1
Culiseta	64 / 67	95.5
Accuracy	1 742 / 1 781 = 97.8%	

Species		
Species	**Correct/Total**	**Acc. (%)**
Aedes aegypti	118 / 127	92.9
Aedes albopictus	243 / 253	96.0
Aedes caspius	213 / 239	89.1
Aedes koreicus	136 / 146	93.2
Aedes vexans	84 / 106	79.2
Anopheles maculipennis	81 / 94	86.2
Culex pipiens	254 / 285	89.1
Culiseta longiareolata	64 / 67	95.5
`Unknown`	406 / 464	87.5
Accuracy	1 599 / 1 781 = 89.8%	

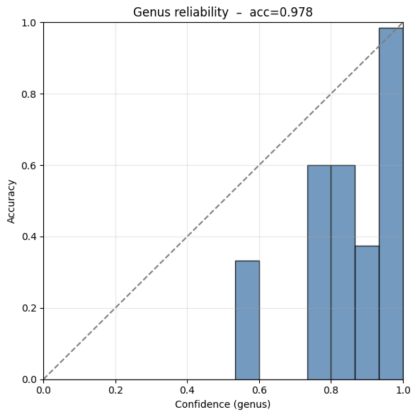
Table 2: Per-class accuracy on the test set for genus (left) and species (right), with consistent color shading by genus. The summary row is left uncolored for clarity.

7.2 Model calibration and reliability

To assess the quality of the predicted confidence scores, we computed the **Expected Calibration Error (ECE)** and plotted reliability diagrams. Before post-hoc calibration, the genus classifier exhibited mild overconfidence, as shown in Figure 12. After applying temperature scaling, the predicted confidence distributions align more closely with empirical accuracy.



Before scaling: T = 1.00, ECE = 0.013



After scaling: T = 1.598, ECE = 0.021

Figure 12: Reliability diagrams for genus classification before and after temperature scaling.

While temperature scaling slightly increased the ECE in this case, it reduced the gap between confidence and accuracy for lower-confidence bins, making predictions more interpretable and consistent.

7.3 Open-set performance and unknown detection

The model's ability to detect samples belonging to unknown species—those not seen during training—was evaluated by monitoring its usage of the dedicated `unknown` class introduced in the species classifier. Overall, it achieves an accuracy of **87.5%** on unknown samples. However, a non-negligible number of misclassifications still occur:

- **73 known specimens** were incorrectly rejected and assigned to the unknown class (*false positives*). These samples typically belong to rare species or show low-confidence predictions with inconsistent genus-species assignments.
- **58 truly unknown specimens** were wrongly classified as belonging to one of the known classes (*false negatives*). These often occurred in cases where the model produced high but misplaced confidence.

8 Conclusion and possible improvements

In this work, we developed a hierarchical multi-task classifier for mosquito species identification, with the ability to handle open-set scenarios via the m-EDL uncertainty modeling approach. While the current results show promising genus-level accuracy and a reasonable trade-off at the species level, we expect significant improvements as the system is deployed and new field data become available.

Indeed, one of the key advantages of m-EDL is its ability to defer ambiguous specimens to expert entomologists whenever the model is uncertain. This property allows the collection of high-quality manual labels precisely in the regions of the data distribution where the model is less confident. Over time, this feedback loop will naturally expand the dataset of known species, gradually reducing the unknown region.

Once a sufficiently large and diverse dataset has been accumulated, we foresee a possible transition to a fully discriminative classifier without explicit uncertainty modeling, as described in the earlier *Roadmap* section. In this future phase, the species head can be trained with advanced calibration-aware loss functions, such as the recently proposed *inverse focal loss* [10], which helps prevent overconfidence by controlling the contribution of well-classified samples to

the loss.
Moreover, while uncertainty estimation may no longer be explicitly modeled via m-EDL, we plan to retain a certain level of safety by studying the distributions of the output logits during training. By analyzing the distribution of logits associated with correctly classified samples, it is possible to derive robust thresholds that allow detection of both out-of-distribution samples and potential outliers at inference time, preserving open-set awareness even in the fully discriminative setting.

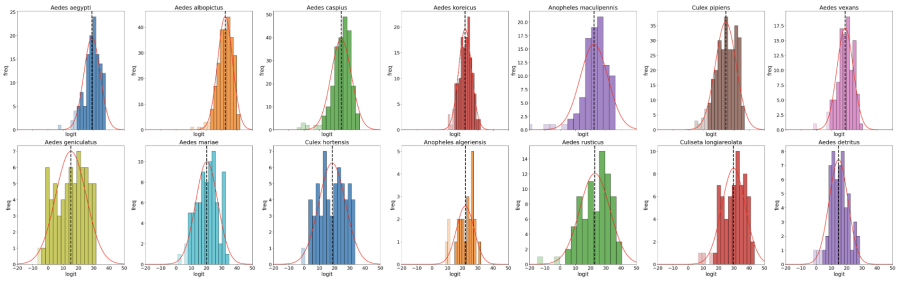


Figure 13: Example distribution of species-level logits obtained from the current dataset. The separation between confident and uncertain predictions enables threshold-based rejection rules.

Computational Resources

The training of the models presented in this work was conducted on the Leonardo supercomputer, one of the most powerful high-performance computing (HPC) systems in Europe. Hosted by CINECA in Bologna, Italy, Leonardo is part of the EuroHPC Joint Undertaking and is ranked among the top HPC facilities worldwide. It is based on a modular architecture that combines general-purpose computing nodes with accelerated modules, offering a peak performance in the exascale range. Using Leonardo enabled the training of very large-scale deep neural networks, such as ConvNeXtV2-Huge (with over 660 million parameters), in reasonable timeframes.
We gratefully acknowledge the computational support provided by CINECA under the computing grant for this project.

Figure 14: The Leonardo HPC system used for model training (CINECA, Bologna)

References

[1] Kim Bjerge et al. “Hierarchical classification of insects with multitask learning and anomaly detection”. In: *Ecological Informatics* 69 (2022), p. 101612. URL: https://proceedings.neurips.cc/paper/2021/hash/61f3a6dbc9120ea78ef75544826c814e-Abstract.html.

[2] Chuan Guo et al. “On calibration of modern neural networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 1321–1330. URL: https://arxiv.org/abs/1706.04599.

[3] Tong He et al. “Bag of Tricks for Image Classification with Convolutional Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 558–567. DOI: 10.1109/CVPR.2019.00065.

[4] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations (ICLR)*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[5] Paulius Micikevicius et al. “Mixed Precision Training”. In: *International Conference on Learning Representations (ICLR)*. 2018. URL: https://openreview.net/forum?id=r1gs9JgRZ.

[6] Akihito Nagahama. “Learning and predicting the unknown class using evidential deep learning”. In: *Scientific Reports* 13.1 (2023), p. 14708. DOI: 10.1038/s41598-023-40649-w. URL: https://www.nature.com/articles/s41598-023-40649-w.

[7] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. URL: https://arxiv.org/abs/1506.02640.

[8] Murat Sensoy, Lance Kaplan, and Melih Kandemir. “Evidential deep learning to quantify classification uncertainty”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018), pp. 3179–3189. URL: https://arxiv.org/abs/1806.01768.

[9] Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. “Rethinking Calibration of Deep Neural Networks: Do Not Be Afraid of Overconfidence”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[10] Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. “Rethinking Calibration of Deep Neural Networks: Do Not Be Afraid of Overconfidence”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021. URL: https://arxiv.org/abs/2106.09292.