Robust Animal Re-identification: A Framework based on Fine-tuning for Open-Set Recognition

Hou Haoran, Liang Tingshuo, Zhou Xinrong

Department of Statistics, The Chinese University of Hong Kong

May 1, 2025

Contents

1 Introduction

The major difficulty in wildlife research is identifying animals' unique characteristics – such as spots, fringe and so on. Traditional methods mainly rely on manual observation, which is not only time-consuming and labor-intensive, but also prone to some human error. Recent development in image identification and pattern recognition has provided some more systematic identification techniques. However, current image identification models are sensitive to factors such as noise artifacts, illumination conditions, and viewpoint variations, while frequently failing to generalize to novel environments. Moreover, most models are designed for closed set recognition.

Our project aims to tackle these issues by constructing robust models for recognizing individual sea turtles, salamanders, and lynxes. The primary objective is to develop models' capability of both identifying seen individuals and detecting unseen ones - the open set recognition capability most deep learning approaches typically underperform. We mainly implemented the fine-tuning of a powerful pretrained model (MegaDescriptor), combined with some techniques like species-specific adaptive thresholds and appropriate image enhancements.

This works are expected to save wildlife researchers tons of time and give more accurate data comparing conservation efforts—whether it's tracking endangered turtles or monitoring lynx populations.

2 Literature Review

In recent years, numerous innovative and impactful methods have emerged to address the challenges outlined above. One notable approach is OpenMax [3], which enhances a model's ability to recognize unknown categories by replacing the traditional SoftMax layer with OpenMax. This modification effectively enables more reliable rejection of unseen classes. Another advancement is the ALIKED network [17], which incorporates a Sparse Deformable Descriptor Head (SDDH) and relaxes the neural reprojection error (NRE) loss from dense to sparse, thereby improving the training of extracted sparse descriptors. As a result, ALIKED demonstrates both efficiency and strong performance across a range of visual measurement tasks, including image matching, 3D reconstruction, and visual relocalization. These developments can be comprehensively evaluated using state-of-the-art open world object detection systems.[10].

In addition to open-set solutions, image augmentation [4] offers a classical and reliable technique for expanding the dataset, thereby enhancing the model's recognition accuracy. Another effective strategy for achieving higher accuracy with limited data is the adoption of metric learning methods[12], which can significantly improve the performance of supervised learning approaches by optimizing the embedding function by using category labels and enhancing the discriminative ability and generalization performance of the model.

3 Dataset Overview

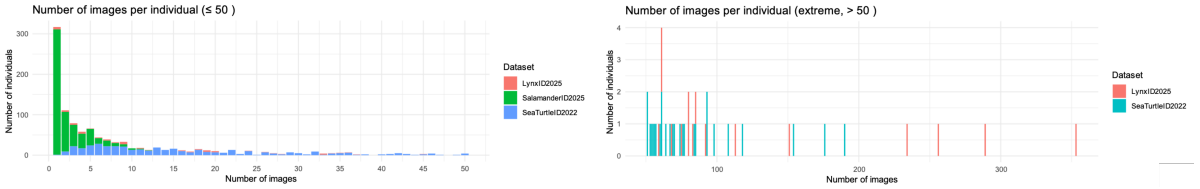
3.1 Data Source

The AnimalCLEF25 dataset[7] from Kaggle was employed for our open-set re-identification experiments. It consists of 1,102 distinct identities and over 13,000 images from three species: loggerhead sea turtles, salamanders, and Eurasian lynxes. Notebly, none of those images are included in the WildlifeReID-10k dataset[1]. From Figure A.1a in Appendix,

we can observe that the images vary a lot in lighting conditions and environments. For instance, there are also some grayscale or underwater pictures with low contrast and resolution. Additionally, some salamander images were taken when holding in hands, resulting in visible human hands and some occlusions. These factors introduce significant noise, thereby increasing the difficulty for the model of learning robust visual features.

3.2 Data Statistical Characteristics

Following figures illustrate the distribution of the number of images per individual. It is clear that there exists a severe data imbalance: most individuals have only a few images, while some have up to hundreds of images. Such imbalance may limit the model's feature extraction ability, not only potentially resulting in poor feature quality and reduced discriminative ability, but also increasing the risk of overfitting and poor generalization performance. Additionally, imbalance among species is also clear; most salamander individuals own few images, while some sea turtle individuals have numerous images. This may bias the model's confidence toward classes with greater representation.[8]



3.3 Data Pre-processing and Augmentation

The python library `imgaug`[11] was employed for image augmentation. We designed a randomized augmentation sequence including various effects, such as rotation, Contrast Limited Adaptive Histogram Equalization, Gaussian blur and so on.[15] Each effect was applied randomly with designed random parameters. As shown in Figure A.1 , this sequence significantly enhances the visual clarity of textures in many images.

As demonstrated in Figure A.2 in appendix, repeated application of the sequence to the same image yields diverse outputs thanks to the stochasticity of the augmentation process. To mitigate data imbalance, an oversampling strategy mentioned by Mohammed et al.[13] was implemented for individuals with less than 5 images. Specifically, existing images were augmented to generate new images until reaching a total of 5 images.

3.4 Dataset Splitting Strategy

Two splitting strategies were included, the open-set and closed-set strategy, both ensure no repeated images in two subsets. The open-set strategy ensures that the second subset contains individuals unseen in the first subset, while the closed-set strategy guarantees that the first subset includes all individuals presented in the second subset, with different images. In the project, we consistently allocated 90% of the data to the first subset and 10% to the second subset. However, due to the special splitting strategy, the percentage is not exactly guaranteed.

Initially, the open-set strategy was employed, resulting in the `availableSet` and `testSet`. The latter was reserved exclusively for evaluating overall performance, containing individuals not appearing in the former. Subsequently, the `availableSet` was further divided with open-set strategy into `trainSet` and `thresholdSet`. The `thresholdSet` would be used for selecting appropriate thresholds facing open-set recognition tasks.

Finally, the `trainSet` was partitioned using a closed-set approach into `training` and `validating`, utilized for training and validation purposes respectively, allowing the monitoring during the training progress.

4 Models

4.1 MegaDescriptor Fine-Tuning

The first approach we adopted is to fine-tune a pre-trained image feature model on our dataset specific to 3 species. And this model will be used as a feature extractor.

MegaDescriptor-L-384[6] is a sophisticated Swin-L vision transformer, specifically designed and pretrained on WildlifeReID-10k[1] for robust species-invariant feature extraction. At its core, the model employs a hierarchical design with shifted window-based self-attention mechanisms, which efficiently processes visual information across multiple scales. This unique attention mechanism, combined with patch merging operations, enables the model to simultaneously capture fine-grained texture patterns and broader contextual information such as body morphology and postural variations. The architecture's substantial capacity, comprising 228.8M parameters across 24 transformer blocks, facilitates the learning of complex feature hierarchies through its four-stage progressive structure (192→384→768→1536 channels). As mentioned above, the model's pretraining datasets don't overlap with current dataset, ensuring the learned features are truly species-agnostic. This comprehensive pretraining strategy, coupled with the model's inherent architectural advantages, makes it particularly suitable for fine-tuning tasks that require robust individual identification under open-set challenge. Specifically, the model's architecture for fine tuning can be decomposed into following blocks, see Figure 4.1.

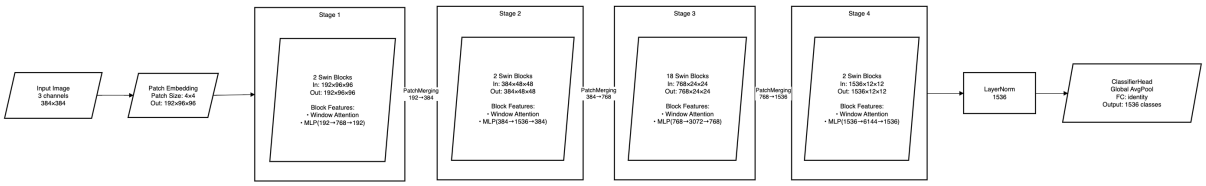


Figure 4.1: MegaDescriptor-L-384 Architecture

4.1.1 Fine-tuning Architecture

The fine-tuning approach employs a selective layer unfreezing strategy to optimize the model for the target task while preserving pretrained knowledge. Initially, all layers are frozen to maintain the robust species-invariant features learned during pretraining. Then, specific components may be strategically chosen to unfreeze:

- (optional) The penultimate stage `SwinTransformerStage(2)` of the backbone, processing lower-level feature representations, with 234 parameters
- The final stage `SwinTransformerStage(3)` of the backbone, which processes high-level semantic features, with 29 parameters
- The final `Norm` layer for adaptive feature normalization, with 2 parameters

- The newly initialized fully-connected layer in `head`, specially designed for corresponding fine-tuning strategy

The strategy balances between leveraging pretrained knowledge and adapting to the specific requirements of our target dataset, while being computationally efficient by only updating a subset of the model's 228.8M parameters.

4.1.2 Optimizer Configuration

We chose AdamW optimizer and employed a differential learning rate strategy. The rationale for using AdamW is its proven effectiveness in transformer fine-tuning tasks, combining adaptive learning rates with proper weight decay regularization. The differentiated learning rates allow more aggressive updates for new layers while maintaining stability in pre-trained components.

Below are two fine-tuning strategies. The first strategy fine-tunes the model by adding a classification head and optimizing with cross-entropy loss, while the second employs a metric learning setup using triplet loss to improve the embedding space for similarity-based tasks.

4.1.3 Classification Fine-tuning

4.1.3.1 Fully-connected Layer

The newly initialized classification head, which is modified from the original architecture to output 912 classes (number of known individuals in `training` dataset). The classification head here is a simplified version in consideration of reducing overfitting. (More complex classification head has also been experimented but yielded inferior results. see Section 6.1.1).

4.1.3.2 Batch Size

In our experiments, we evaluated the impact of different batch sizes on model performance by training with batch sizes of 16, 32, and 64, which are typical choices in this field. As shown in Appendix Figure A.5, the training loss curves for all three batch sizes exhibit similar trends, with no significant differences in convergence speed or final loss values. Furthermore, the validation loss (evaluated on `validating`) in Figure A.6 also remained consistent across the different batch sizes, indicating that the choice of batch size within this range does not substantially affect the model's ability to generalize.

4.1.3.3 Learning Rate

- The final stage of the backbone: 5e-5
- Normalization layer: 1e-5
- Classification head: 1e-5, 2e-5, 5e-5
- Weight decay: 0.05

We adopted a differential learning rate approach to apply different learning rates to the pretrained components and newly added classification head in our model. Specifically, the final stage layers and the normalization layers of the original MegaDescriptor model, which were unfrozen during training, were assigned relatively small learning rates (5e-5 and 1e-5, respectively) to preserve the valuable pretrained features while allowing for gradual adaptation to our dataset. Notably, normalization layers received an even lower rate (1e-5) to stabilize their mean or variance estimates during training.

For the classification head, we scaled the learning rate linearly with batch size (1e-5 for 16, 2e-5 for 32, 5e-5 for 64), a common practice to maintain convergence stability when batch sizes vary, following the standard linear scaling rule [9] that larger batch

sizes can typically accommodate higher learning rates without destabilizing training. As an additional experiment, we applied a OneCycle learning rate schedule, which gradually increases the learning rate to a peak before annealing it towards the end of training as shown in Figure A.7. This approach is widely used and considered reasonable, as it can help the optimizer converge faster and potentially improve generalization.

A weight decay of 0.05 was also applied to regularize the model and prevent overfitting.

4.1.3.4 Loss Calculation

For our classification task, we employed the cross-entropy loss as the objective function. It measures the discrepancy between the model's predicted class logits and the true class labels. For a batch of N samples, it computes:

$$\mathscr{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\left(z_{i,y_i} - \log\left(\sum_{j=1}^{C} e^{z_{i,j}}\right)\right)$$

where $z_{i,j}$ is the logit for sample i and class j, C is the number of classes, and y_i is the true label. This loss internally applies a softmax to the logits, followed by a negative log-likelihood term, optimized for both stability and efficiency.

4.1.4 Embedding Fine-tuning

As the model is used for feature extracting and the final prediction relies on feature vectors, we also consider fine-tuning an embedding space directly. This process is called metric learning, and the goal is to ensure feature vectors of the same individuals are pulled closer together while vectors of different individuals are pushed farther apart.

4.1.4.1 Loss Calculation and Training Sample Selection

Triplet loss with cosine similarity is employed during the metric learning process. The margin is initially set to 0.2 and seemed to work well. However, three input samples: anchor, positive negative are required to guide the embedding space optimization, introducing the challenge of generating informative triplets. For batching, the M-per-class sampler would ensure multiple images of each identity are present within a batch, providing enough options for the miner. Additionally, the miner is configured for semi-hard triplet selection, balancing between difficulty and stability, as experiments indicated that hard triplets incur excessive fluctuations in the loss values.

4.1.4.2 Embedding Head Design and Dimensionality

A linear transformation was used as a fully connected layer to build a 1024-dimensional embedding space. Notably, our experiments revealed that a 512-dimensional embedding space yielded to much inferior performance. This may be attributable to the higher dimensionality enabling the representation of richer and more comprehensive information.

4.1.4.3 Layer Unfreezing Strategy and Learning Rate

Comparing with the Classification strategy, an additional stage `SwinTransformerStage(2)` was unfrozen, allowing the model to learn richer feature representations. Although proved effective in improving performance, this stage contains 234 parameters, thus significantly increasing GPU memory usage and forcing a decrease in batch size.

4.2 Apply Another Framework: EfficientNet

Apart from the MegaDescriptor Fine-Tuning strategies, we also adopted another framework - the EfficientNet, which has been a popular model for its low parameter

count requirement and high accuracy rate. Among all variants of this network, we chose EfficientNet B4 as our training model because it reaches a sweet point among speed, accuracy and GPU demand.[16] First, we tried to directly import our dataset with preimage augmentation into the model and used it to recognize the three animals. Secondly, we chose to freeze all layers and then unfreeze only the classifier head and replaced it with the provided WildFusion classifier head in order to boost performance in the specific animal recognition mission. However, this model fell short of expectations in both attempts comparing with the MegaDiscriptor , refer to Table 6.2 for detail.

5 Prediction

5.1 Similarity Calculation

The prediction pipeline processes the input database and query images through a series of well-defined stages. In the first stage, the WildFusion[5] implements the priority pipeline to extract features from the two datasets with the fine-tuned MegaDescriptor in Section 4. These features then undergo L2 normalization, followed by a cosine similarity calculation. For the top 20 pairs, ALIKED descriptors with LightGlue matching are applied for new scores. Finally, similarity scores from all pipelines are calibrated through isotonic regression with a calibration set to ensure comparability and effective fusion, during which the raw similarity scores are mapped to a probabilistic scale.

5.2 Species-Specific Threshold Selection

Our threshold determination process utilize the `thresholdSet` splitted in Section 3.4 (instead of the `validating` and the `testSet` since the selection of threshold still belongs to hyperparameter tuning) as the query dataset and the `training` used in training as the database dataset. Upon computing similarity scores with all individuals in the `training` for each `thresholdSet` image, if the highest similarity score falls below the species-specific threshold, the individual is predicted as "new_individual"; otherwise, it is predicted as the identity corresponding to the highest similarity score. This binary decision mechanism requires careful threshold calibration to balance between two competing objectives: accurately identifying known individuals (BAKS, to be introduced in Section 6.0) and correctly recognizing new ones (BAUS, to be introduced in Section 6.0).

However, fixed similarity thresholds often prove inadequate for different species due to their varying characteristics. To address this, we introduce a species-specific threshold calibration method that optimizes the trade-off between seen and unseen individual identification accuracy. For each target species, we:

1. Evaluate thresholds across a range (0.05 to 0.95), computing BAKS and BAUS.
2. Select the optimal threshold by maximizing the geometric mean (GEO_MEAN) of BAKS and BAUS:

$$\tau_s^* = \underset{\tau}{\text{argmax}} \sqrt{\text{BAKS}_s(\tau) \times \text{BAUS}_s(\tau)}, \quad s \in \{\text{lynx}, \text{turtle}, \text{salamander}\}$$

6 Result and Evaluation

6.0 Metrics

In open-set recognition, traditional evaluation metrics fail to account for the balance between recognizing seens and rejecting unseens. We employed three specialized

metrics[1]:

- Balanced Accuracy for Known Samples (BAKS): computes the mean per-class recall for known individuals, excluding samples from novel test-set identities ($\mathscr{C}_{\text{test-only}}$):

$$\text{BAKS} = \frac{1}{|\mathscr{C}|} \sum_{c \in \mathscr{C}} \frac{|\{\boldsymbol{x} \mid \hat{y}(\boldsymbol{x}) = y(\boldsymbol{x}) = c, y(\boldsymbol{x}) \notin \mathscr{C}_{\text{test-only}}\}|}{|\{\boldsymbol{x} \mid y(\boldsymbol{x}) = c\}|}$$

- Balanced Accuracy for Unknown Samples (BAUS): measures the mean per-class true negative rate for novel identities, where correct predictions must be labeled as 'new_individual':

$$\text{BAUS} = \frac{1}{|\mathscr{C}_{\text{test-only}}|} \sum_{c \in \mathscr{C}_{\text{test-only}}} \frac{|\{\boldsymbol{x} \mid \hat{y}(\boldsymbol{x}) = \text{'new\_individual'}, y(\boldsymbol{x}) = c\}|}{|\{\boldsymbol{x} \mid y(\boldsymbol{x}) = c\}|}$$

- The Geometric Mean (GEO_MEAN): combines both metrics:

$$\text{GEO\_MEAN} = \sqrt{\text{BAKS} \times \text{BAUS}}$$

6.1 MegaDescriptor Classification Fine-Tuning

6.1.1 Training Process

The training and validation loss curves for the different models are shown in Figure 6.1 and Figure 6.2. With the exception of the Model with complex classification head (Batch64-ComplexHead), all models demonstrate converging training and validation losses, indicating stable training and no significant overfitting. The performance across these models is generally comparable.

Notably, the model trained with the learning rate warmup strategy (Batch64-WarmupLR) exhibits the fastest convergence and achieves the lowest final validation loss, suggesting that the warmup schedule is effective in promoting both rapid and robust learning.

In contrast, the Model Batch64-ComplexHead, which corresponds to a model with a more complex classification head, shows signs of overfitting: the training loss decreases rapidly while the validation loss begins to increase after a few epochs. This behavior is likely due to the increased capacity of the classification head, which allows the model to fit the training data too closely at the expense of generalization. To mitigate this, early stopping was applied during training.

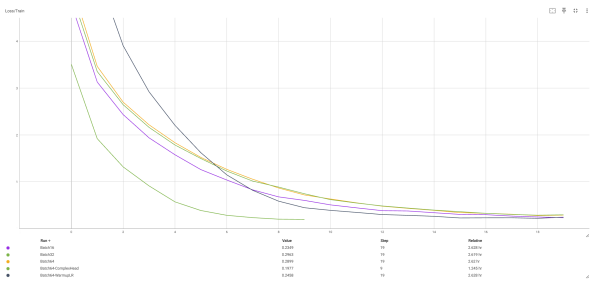


Figure 6.1: Training loss of models

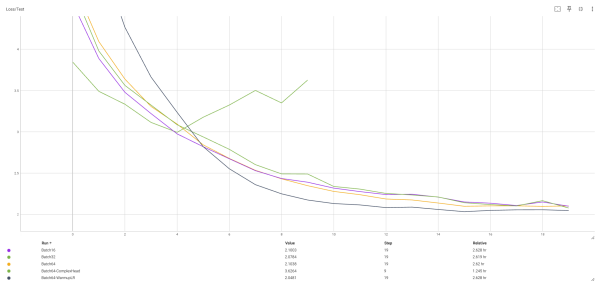


Figure 6.2: Validation loss of models

6.1.2 Prediction Performance

After training, we removed the classification head and used the WildFusion with ALIKED to compute pairwise similarity scores across the `thresholdSet` and `training` datasets. For each species, we plot: (1) BAKS, (2) BAUS, and (3) GEO_MEAN across

decision thresholds. The species-specific optimal threshold is selected at the GEO_MEAN maximum. The visualization of the calibration of Model Batch64-WarmupLR (Figure 6.3) demonstrates how tailored thresholds are selected.

Table 6.1 summarizes the maximum GEO_MEAN values and their corresponding BAKS and BAUS for each species across the evaluated models. Notably, Model Batch64-WarmupLR generally achieves high GEO_MEAN for all species, demonstrating slightly better performance in balancing the identification of known individuals BAKS and rejection of the unknown BAUS. In contrast, Model Batch64-ComplexHead yields the lowest GEO_MEAN, indicating comparatively weaker trade-off optimization for all species.

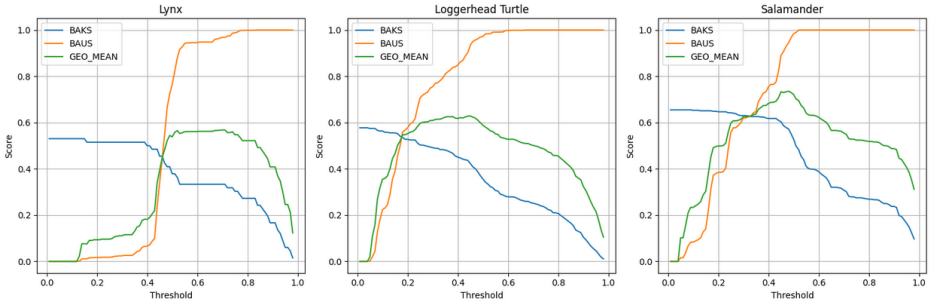


Figure 6.3: Species-Specific Threshold Selection of Model Batch64-WarmupLR

Table 6.1: Performance comparison of different models across species and metrics

Model	Lynx			Loggerhead Turtle			Salamander		
	BAKS	BAUS	GEO_MEAN	BAKS	BAUS	GEO_MEAN	BAKS	BAUS	GEO_MEAN
Batch16	0.318	0.922	0.542	0.442	0.901	0.631	0.603	0.931	0.750
Batch32	0.364	0.949	0.588	0.439	0.859	0.614	0.587	0.871	0.715
Batch64	0.303	0.976	0.544	0.428	0.845	0.601	0.550	0.907	0.706
Batch64-WarmupLR	0.333	0.968	0.568	0.423	0.935	0.629	0.572	0.945	0.735
Batch64-ComplexHead	0.303	0.844	0.506	0.434	0.896	0.623	0.378	0.971	0.606

6.2 MegaDescriptor Embedding Fine-Tuning

6.2.1 Training Process

The loss plot is Appendix Figure A.3, as it exhibits a typical pattern. While after each training epoch, a special validation phase was carried out. During which a percentage accuracy was calculated, facilitating the monitor of model performance. Additionally, to detect potential overfitting, the accuracy was also calculated on a randomly selected subset of the training data, as illustrated in the figure A.4.

6.2.2 Prediction Performance

Achieve an overall 0.6722 geometric mean on 0.17 threshold, with 0.5219 BAKS and 0.8657 BAUS. The performance on different species is illustrated below.

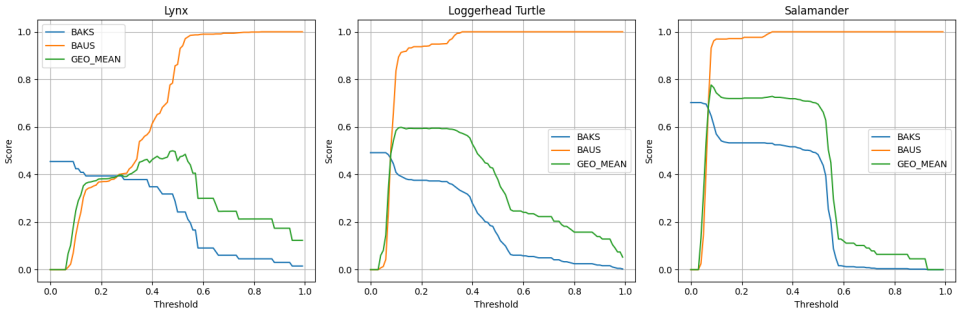


Figure 6.4: Model Performance on Different Species

6.3 Model Comparison

The results in Table 6.2 summarize the performance of different models on the `testSet`, as described in Section 3.4. Among all evaluated models, the model using metric learning (Metric) achieves the highest GEO_MEAN score (0.723), indicating the best overall balance between identifying known individuals (BAKS) and detecting new identities (BAUS). This superior performance is attributed to its ability to maintain a high BAKS while also achieving a strong BAUS, demonstrating robust open-set recognition capability. In contrast, while other models may excel in either BAKS or BAUS, they do not achieve the same level of balanced performance as the Metric model.

Table 6.2: Overall performance comparison of different models

Model	BAKS	BAUS	GEO_MEAN
Batch16	0.518	0.905	0.684
Batch32	0.500	0.843	0.649
Batch64	0.497	0.862	0.654
Batch64-WarmupLR	0.500	0.932	0.683
Batch64-ComplexHead	0.431	0.914	0.628
Metric	0.579	0.903	0.723
EfficientNet B4	0.229	0.706	0.402

7 Discussion

This section includes some potential approaches we believe are promising for future research, along with some preliminary attempts and ideas.

7.1 Data Pre-processing and Augmentation

As mentioned in Section 3.2, some individuals have numerous images. Mohammed et al.[13] explored undersampling as a potential remedy. However, they also point out the risk of removing some valuable data essential for model training, necessitating careful design of undersampling algorithm. Although undersampling has not shown effectiveness in their experiment, it could be a worthwhile endeavor for our future work. We preliminarily propose dynamically selecting images for majority classes during each training epoch.

Regarding image augmentation, our preliminary experiment indicated that online augmentation worked better than offline augmentation, as the model could learn from slightly varied images in each epoch. This continuous variation seemed to be especially beneficial for mitigating overfitting. However, as augmentations are performed each time a batch is fetched, this method introduces considerable computational overhead, doubling

epoch duration from approximately 15 to 30 minutes. Given limited computing resources, we adopted offline augmentation in this project. We may try to fine-tune the finalized model with online augmentation, once optimal hyperparameters and training strategies have been thoroughly established.

7.2 Model Fine-tuning

The images are also annotated with some other metadata such as the camera orientation and date. Appropriate use of the additional information may enhance the model's identification ability. For instance, Adam et al.[2] proposed a time-aware splitting strategy that utilizes the capture date. Regarding the orientation data, we initially propose two potential strategies: directly input along with the images or embedding it at an optimal stage. Further architectural improvements could involve multi-head designs for better processing of heterogeneous inputs.

7.3 Prediction

Currently, the `WildFusion` pipeline with ALIKED and LightGlue involved is employed for similarity score calibration. However, the two extractor and matcher are designed for general-purpose; fine-tuning them may improve the calibration performance. However, such process may require additional data annotation and strategy, which may be too hard to implement. In addition, we try to develop some other models to extend the pipeline.

When it comes to the similarity score itself, current method simply returns the identity of the most similar image from the database. However, each individual is represented by multiple images, which presents an opportunity to aggregate information across these images. Naive approaches, including averaging the similarity scores across all images of the same individual and voting with the top 10 scores, may help produce more robust similarity scores by leveraging more database images.

Finally, we explored the OpenMax[3] for the open-set challenge, a new layer that computes the probability of an input belonging to an unknown class by calibrating softmax outputs based on the Mean Activation Vector (MAV) of known classes. However, this requires at least one correctly identified sample per individual for corresponding MAV establishment, a condition not consistently met currently. As a remedy, we propose assigning extremely low logits to those with no true samples, excluding them from OpenMax calibration and shifting their probability mass toward the unknown.

8 Conclusion

This project primarily leverages strategic fine-tuning techniques to enhance the application of MegaDescriptor in wildlife recognition. Through effective fine-tuning of the MegaDescriptor model and appropriate image processing, we achieved a remarkable 140% improvement on geometric accuracy compared to the baseline performance on Kaggle[14], highlighting the value and application potential of our work. It can, to a certain extent, promote the protection of wild animals. In addition, thanks to our research and workflow on open-set challenge, our approach may have a potential for application in some other practical scenarios beyond wildlife recognition.

References

[1] Lukáš Adam, Vojtěch Čermák, Kostas Papafitsoros, and Lukas Picek. Wildlifereid-10k: Wildlife re-identification dataset with 10k individual animals. *arXiv preprint arXiv:2406.09211*, 2024.

[2] Lukáš Adam, Vojtěch Čermák, Kostas Papafitsoros, and Lukas Picek. Seaturtleid2022: A long-span dataset for reliable sea turtle re-identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7146–7156, January 2024.

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

[4] Marcus D Bloice, Christof Stocker, and Andreas Holzinger. Augmentor: an image augmentation library for machine learning. *arXiv preprint arXiv:1708.04680*, 2017.

[5] Vojtěch Cermak, Lukas Picek, Lukáš Adam, Lukáš Neumann, and Jiří Matas. Wildfusion: Individual animal identification with calibrated similarity fusion. *arXiv preprint arXiv:2408.12934*, 2024.

[6] Vojtěch Čermák, Lukas Picek, Lukáš Adam, and Kostas Papafitsoros. Wildlifedatasets: An open-source toolkit for animal re-identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5953–5963, 2024.

[7] Fine-Grained Visual Categorization. AnimalCLEF 2025. https://www.kaggle.com/competitions/animal-clef-2025/data.

[8] Kushankur Ghosh, Colin Bellinger, Roberto Corizzo, Paula Branco, Bartosz Krawczyk, and Nathalie Japkowicz. The class imbalance problem in deep learning. *Machine Learning*, 113(7):4845–4901, 2024.

[9] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[10] KJ Joseph, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Towards open world object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5830–5840, 2021.

[11] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

[12] Xiaoxu Li, Xiaochen Yang, Zhanyu Ma, and Jing-Hao Xue. Deep metric learning for few-shot image classification: A review of recent developments. *Pattern Recognition*, 138:109381, 2023.

[13] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)*, pages 243–248. IEEE, 2020.

[14] picekl. AnimalCLEF 2025 Baseline. https://www.kaggle.com/code/picekl/animalclef2025-starter-notebook.

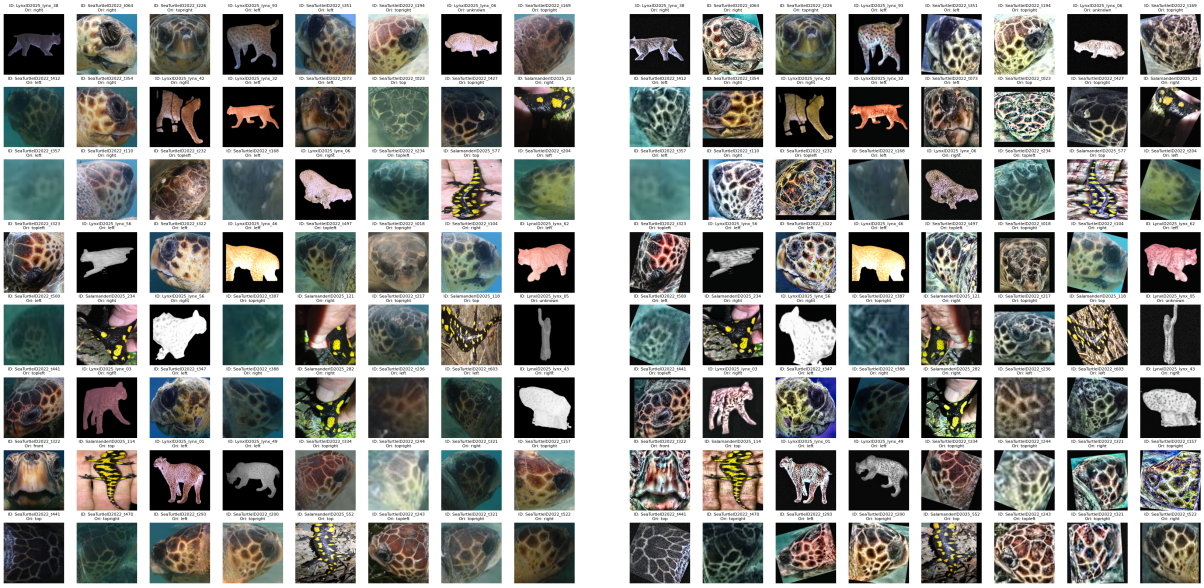
[15] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[16] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[17] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter CY Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE Transactions on Instrumentation and Measurement*, 72:1–16, 2023.

Appendix

A Images



(a) Original pictures from a batch

(b) Augmented pictures from a batch

Figure A.1: Image Augmentation Display

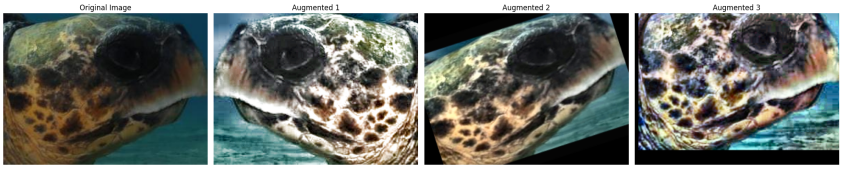
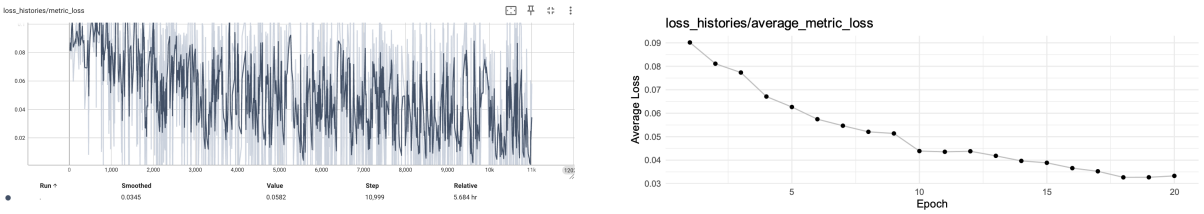


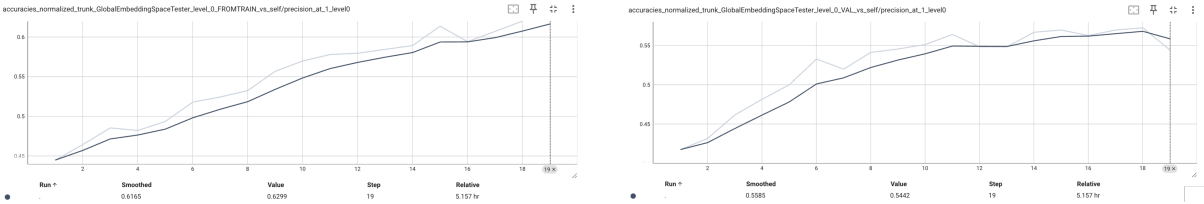
Figure A.2: Random Augmentation Display



(a) Triplet Loss History

(b) Epoch Average Loss

Figure A.3: Loss Plot of Embedding Fine-tuning



(a) Accuracy on Subset of Training Set

(b) Accuracy on Validating Set

Figure A.4: Accuracy of Validation Phase

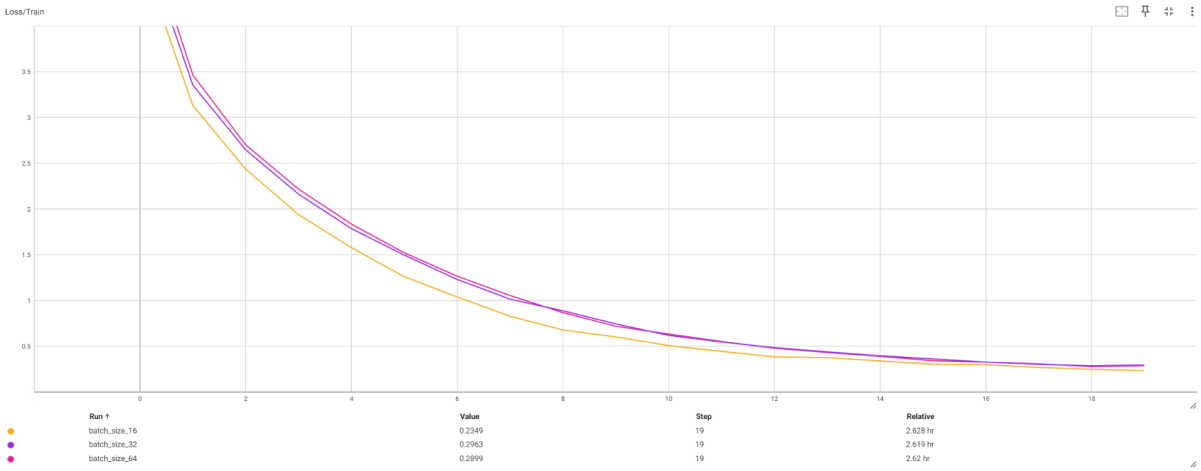


Figure A.5: Training loss with batch size 16, 32, 64

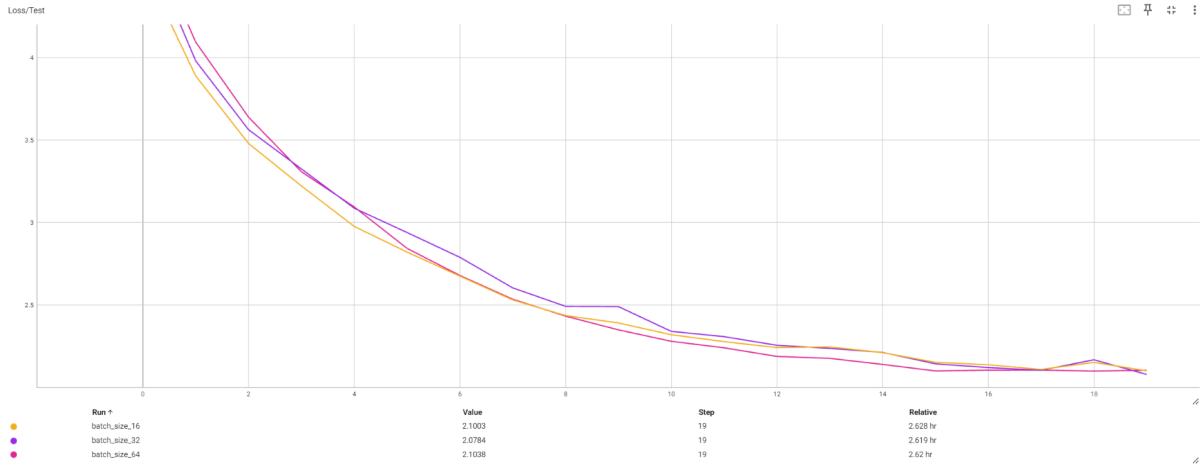


Figure A.6: Validation loss with batch size 16, 32, 64

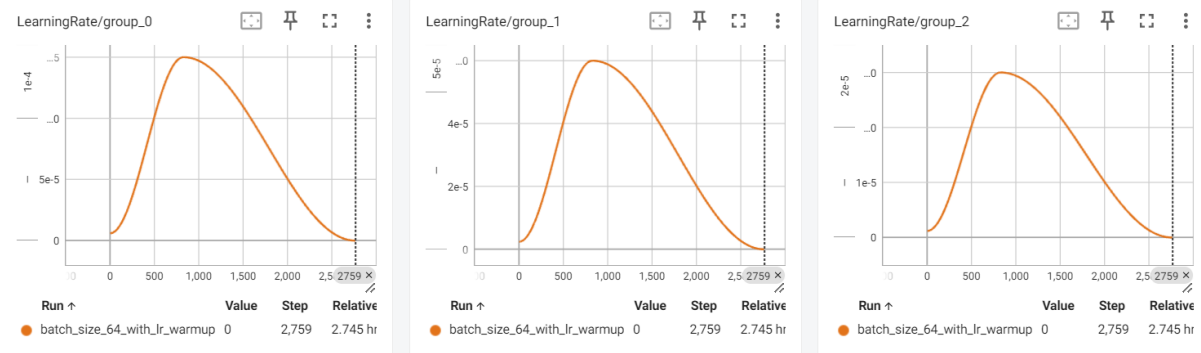


Figure A.7: Learning rate warm-up for three species