Assignment
AnimalCLEF25 @ CVPR-FGVC & LifeCLEF

Vișan Ionuț

1. Introduction

The AnimalCLEF 2025 challenge [1], hosted on Kaggle, is a leading competition in the domain of computer vision and biodiversity informatics. It focuses on the fine-grained task of individual animal identification across multiple species, under real-world, unconstrained conditions. Participants are required to develop robust machine learning models capable of accurately recognizing individual animals from photographic data, despite variations in image quality, angle, lighting, and environmental context.

The competition provides a large-scale, annotated dataset to support the training and evaluation of algorithms, encouraging the use of advanced techniques in deep learning, metric learning, and hierarchical classification.

2. Data Analysis

The main dataset is split into two main subsets:

- Database – Contains labeled data about the individuals.
- Query – Consists of images for identification, where the task is to determine whether the individual is known (present in the database) or new (not present in the database).

The dataset is organized into structured directories to allow easy access and processing. It consists of labeled images for known individuals (*Database*) and unlabeled images (*Query*) requiring identification, metadata, and a sample submission file.

This dataset is all about individual animal identification of the following three species: loggerhead sea turtles (Zakynthos, Greece), salamanders (Czech Republic), Eurasian lynxes (Czech Republic).

The dataset contains 15209 images (database – 13074, query – 2135) and a metadata.csv file which contains multiple columns (image_id, identity, path, date, orientation, species, split, dataset).

2.1. Data Split

We will use the "database" dataset to train the model, as the "query" dataset does not contain labels and is intended for the evaluation of the results.

In this code, the dataset is split as follows:

1. **Identify rare identities:** All images whose identity appears only once (single-occurrence identities) are isolated into a separate subset (317).
2. **Separate normal identities:** Images belonging to identities that appear at least twice are separated for stratified sampling.
3. **Stratified split:** The normal identities are split into 80% training and 20% testing sets using train_test_split, with stratification based on the "identity" column to preserve the class distribution. A random seed (random_state=42) is set for reproducibility.
4. **Training set:** The training set consists of 80% of the normal identities plus all single-occurrence images.
5. **Testing set:** The testing set contains only the remaining 20% of the normal identities; single-occurrence identities are **not** included in the testing set.

Now we have (number of images):

Train set: 10522

Test set: 2552

Split ratio: 0.2425

2.2. Dataloaders

Custom AnimalDataset objects are created from CSV metadata, applying image transformations and encoding identities, then wrapped into PyTorch DataLoaders with batching, shuffling (for training), and a custom collate_fn to handle image-label-path-identity grouping.

2.3. WildlifeReID-10k

The WildlifeReID-10k [2] dataset is a comprehensive benchmark for wildlife re-identification tasks, comprising over 214,000 images of 10,000 individual animals across approximately 33 species, including marine turtles, primates, birds, African herbivores, marine mammals, and domestic animals.

We will use also this dataset, which shares a similar structure with the previous one, and apply the same preprocessing strategy outlined in Section 2.1.

3. Related Work

As the competition is still in progress, there are currently few available notebooks or discussions showcasing strong-performing implementations. Most existing approaches, such as clustering methods or simple convolutional networks, have so far yielded suboptimal results.

I decided to directly explore an approach known for achieving remarkable results: visual transformers.

We will use architectures such as ConvNeXt [3], Swin Transformer [4] for building our models, alongside classical Vision Transformers.

- ConvNeXt is a modernized convolutional network that incorporates architectural ideas from transformers (like large kernel sizes, LayerNorm, and GELU activations) while retaining the inductive biases and efficiency of traditional CNNs.

- Swin Transformer introduces a hierarchical architecture with shifted local attention windows. Instead of computing attention globally, it restricts it to non-overlapping windows and then shifts them between layers to allow cross-window connections. This enables it to model both local and global patterns efficiently and makes it highly scalable for high-resolution tasks.

We also investigate more advanced transformer-based models like MaxViT [5] and MEGA [6]:

- MaxViT (Maximally Local and Global Vision Transformer) combines convolution, local window attention, and global grid attention in a modular block design. This allows it to extract both fine-grained and high-level features in a computationally efficient manner, benefiting from both CNN-like locality and transformer-style context awareness.

- MEGA (Moving Average Equipped Gated Attention) introduces an attention mechanism that incorporates a moving average feature extractor and gating mechanisms to enhance stability, generalization, and long-range dependency modeling. Initially designed for sequential data, MEGA has shown promise in vision tasks by offering a more stable and expressive alternative to standard attention.

These diverse architectures will allow us to benchmark both convolutional and transformer-based methods, understanding their trade-offs and leveraging their strengths depending on task requirements and data characteristics.

4. Implementation

As noted in the previous milestone, the "convnext_xlarge.fb_in22k_ft_in1k_384" model, trained over 30 epochs with a stable configuration, delivered the strongest performance and secured us a position within the top 40 (rank 37). Building on this success, we now aim to further investigate alternative models and methods to enhance our results.

4.1. Swin Large

We will fine-tune swin_large_patch4_window12_384 on AnimalCLEF dataset with the following configuration:

resize dimension = 384x384

augmentation = RandomResizedCrop, RandomHorizontalFlip, RandomRotation, ColorJitter, AutoAugment (ImageNet policy), RandomErasing

batch size = 16

model = “swin_large_patch4_window12_384”

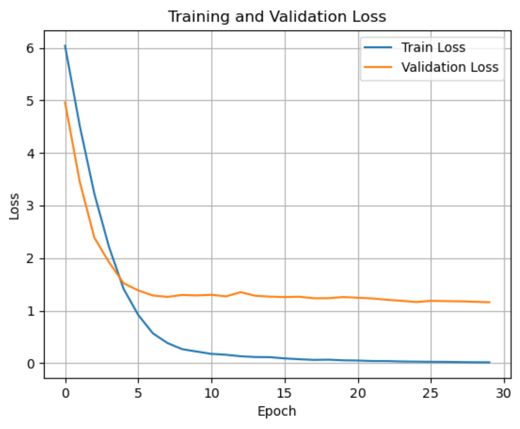
epochs = 30

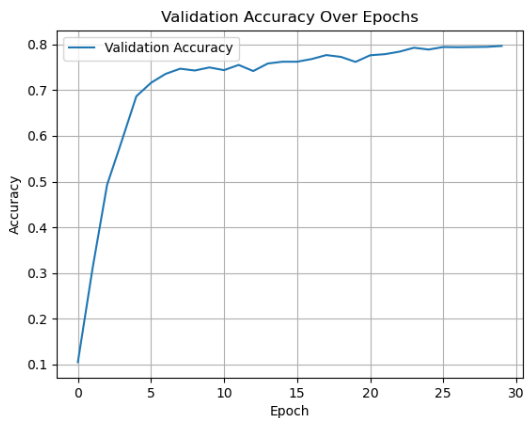
learning rate = 8e-5

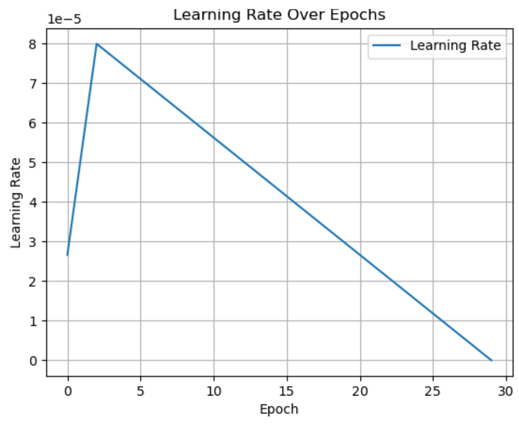
weight decay = 0.01

scheduler = linear (with warmup – 0.1)

Results:







Best model saved at epoch 30 with Val Loss 1.1599 and Val Acc 0.7966.

4.2. MaxViT Large

We will fine-tune maxvit_xlarge_tf_384.in21k_ft_in1k on AnimalCLEF dataset with the following configuration:

resize dimension = 384x384

augmentation = RandomResizedCrop, RandomHorizontalFlip, RandomRotation, ColorJitter, AutoAugment (ImageNet policy), RandomErasing

batch size = 2

model = “maxvit_xlarge_tf_384.in21k_ft_in1k”

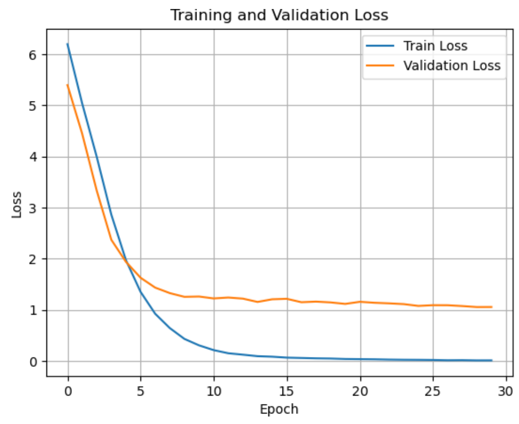
epochs = 30

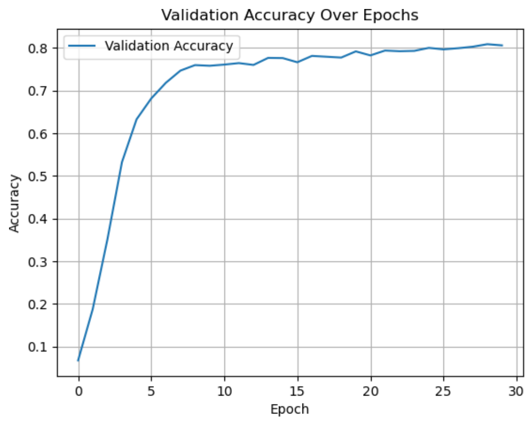
learning rate = 4e-5

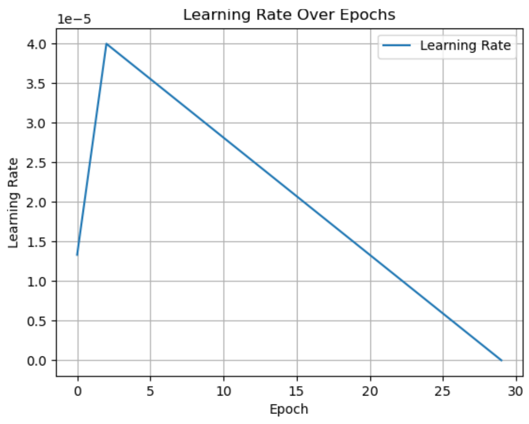
weight decay = 0.01

scheduler = linear (with warmup – 0.1)

Results:







Best model saved at epoch 28 with Val Loss 1.0749 and Val Acc 0.8029.

4.3. Fine-Tuning on WildlifeReID-10k

We will use a multi-stage fine-tuning approach to improve the performance of the convnext_xlarge.fb_in22k_ft_in1k_384 model on the AnimalCLEF dataset. First, we will fine-tune convnext_xlarge.fb_in22k_ft_in1k_384 on the WildlifeReID-10k dataset with the following configuration:

resize dimension = 384x384

augmentation = RandomResizedCrop, RandomHorizontalFlip, RandomRotation, ColorJitter, AutoAugment (ImageNet policy), RandomErasing

batch size = 8

model = “convnext_xlarge.fb_in22k_ft_in1k_384”

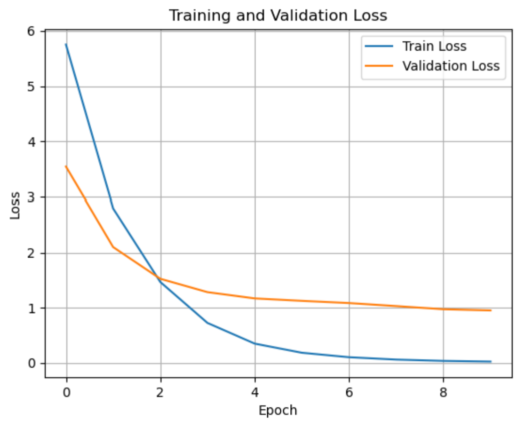
epochs = 10

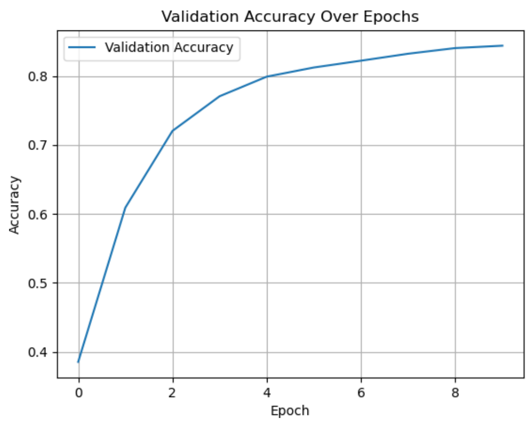
learning rate = 8e-5

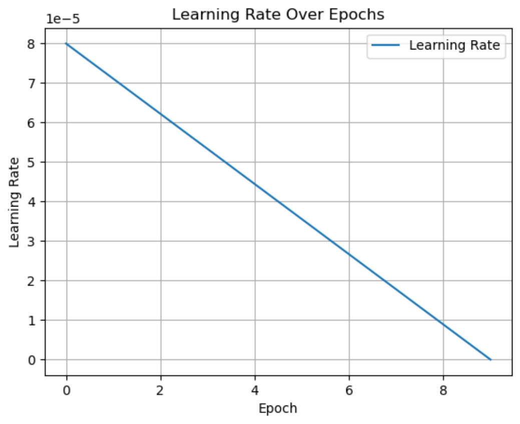
weight decay = 0.01

scheduler = linear (with warmup – 0.1)

Results:







Best model saved at epoch 10 with Val Loss 0.9513 and Val Acc 0.8440.

4.3.1. Fine-Tune 4.3. on AnimalCLEF

We now build upon the previously trained model, already tailored to animal imagery and better aligned with our specific task, and proceed with further fine-tuning on the AnimalCLEF dataset.

resize dimension = 384x384

augmentation = RandomResizedCrop, RandomHorizontalFlip, RandomRotation, ColorJitter, AutoAugment (ImageNet policy), RandomErasing

batch size = 8

model = “convnext_xlarge.fb_in22k_ft_in1k_384”

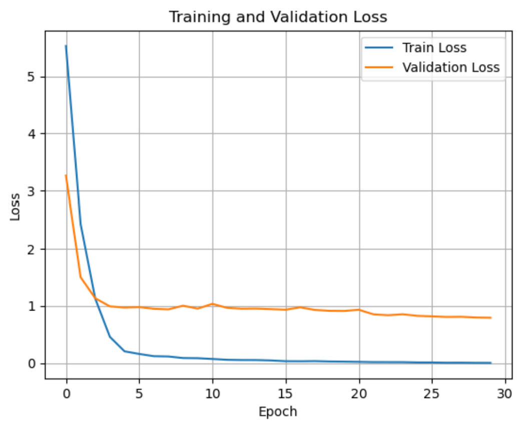
epochs = 30

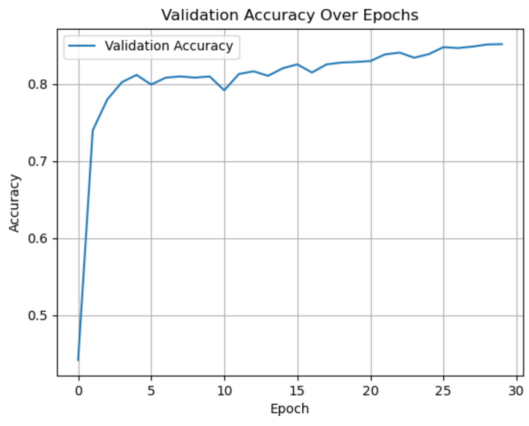
learning rate = 8e-5

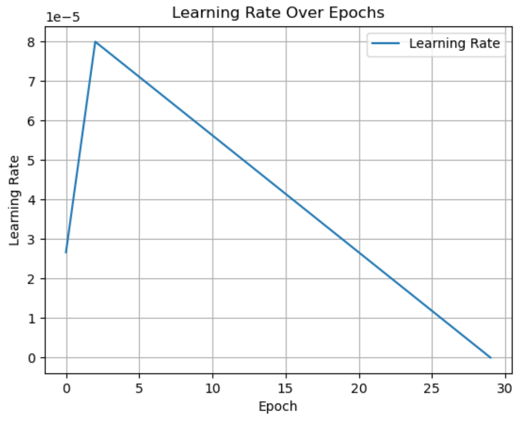
weight decay = 0.01

scheduler = linear (with warmup – 0.1)

Results:







Best model saved at epoch 30 with Val Loss 0.7943 and Val Acc 0.8519.

4.3.2. Explore more into 4.3.1.

To gain deeper insights into the model space outlined in Section 4.3.1. and to investigate potential performance improvements, we propose the following configuration:

resize dimension = 384x384

augmentation = RandomResizedCrop, RandomHorizontalFlip, RandomRotation, ColorJitter, AutoAugment (ImageNet policy), RandomErasing

batch size = 8

model = “convnext_xlarge.fb_in22k_ft_in1k_384”

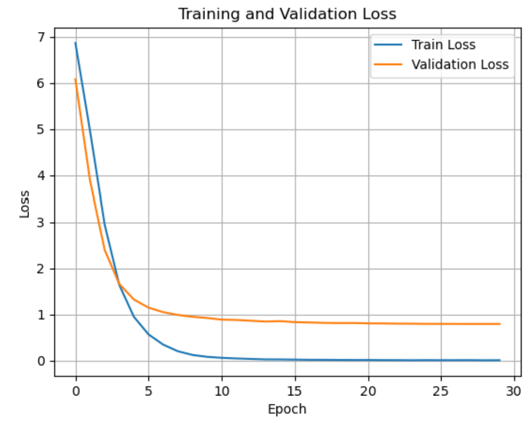
epochs = 30

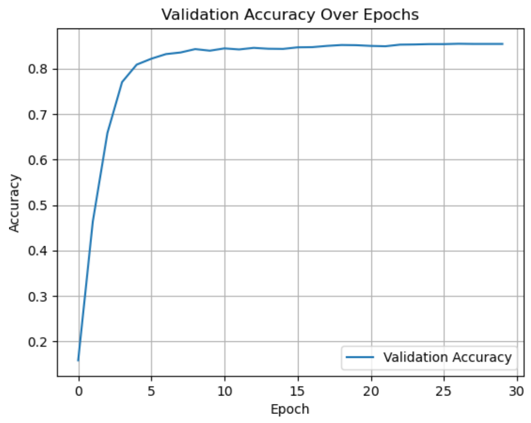
learning rate = 1e-5

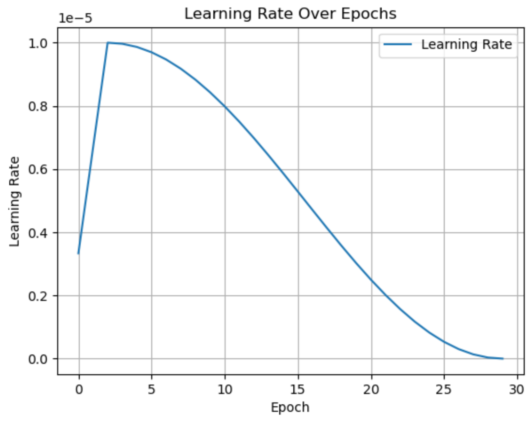
weight decay = 0.01

scheduler = cosine (with warmup – 0.1)

Results:







Best model saved at epoch 27 with Val Loss 0.7979 and Val Acc 0.8546.

4.4. MegaDescriptor + MatchAliked

We construct a robust animal re-identification pipeline by combining global and local feature matching techniques within the WildFusion framework. Specifically, we use the MegaDescriptor ("hf-hub:BVRA/MegaDescriptor-L-384") model for extracting deep global features and ALIKED features matched via LightGlue [7] for fine-grained local correspondences.

Each similarity pipeline is calibrated using isotonic regression to convert raw similarity scores into reliable confidence estimates. The process involves three main datasets: a query set, a database set, and a calibration set (used to fine-tune the confidence scoring).

After calibration, WildFusion fuses the outputs from both global and local pipelines, prioritizing the global (MegaDescriptor) matcher, to compute similarity scores between query and database images. The highest-scoring matches are selected as predictions.

For the calibration process, we utilize two sources: 1,000 representative samples from the database dataset and the test set generated in Section 2.1. To ensure a robust and well-generalized calibration, it is crucial that the selected images are as diverse as possible.

For similarity computation, we will consider two settings: one using a subset of 700 images and another leveraging the full set of available images. This dual approach enables the model to perform under both constrained conditions, where efficiency is prioritized, and unrestricted scenarios, where accuracy can be maximized.

WildFusion was trained to identify similarities between the database and the query images, enabling class predictions for the submission file. Rather than following the traditional training approach focused on extracting evaluation metrics, we leveraged the model in a different way. Section 5 will detail how this strategy contributed to enhancing both performance and overall score.

4.5. Embeddings + Classifier

To enhance the performance of a general model, we will combine the strengths of all previously trained models to build a more robust and accurate classifier.

Embeddings are extracted by feeding images from the train and test datasets into pretrained models:

- convnext_xlarge.fb_in22k_ft_in1k_384 (from previous milestone)
- convnext_xlarge.fb_in22k_ft_in1k_384 (4.3.2.)
- swin_large_patch4_window12_384 (from previous milestone)
- swin_large_patch4_window12_384 (4.1.)
- maxvit_xlarge_tf_384.in21k_ft_in1k (4.2.)
- hf-hub:BVRA/MegaDescriptor-L-384 (4.4.)

For each image batch, the script uses the forward_features method to obtain feature maps from the backbone of the model. If the output is multi-dimensional (e.g., feature maps), global average pooling is applied to reduce it to a fixed-size feature vector. These embeddings, along with their corresponding labels, are collected and saved as .npz files.

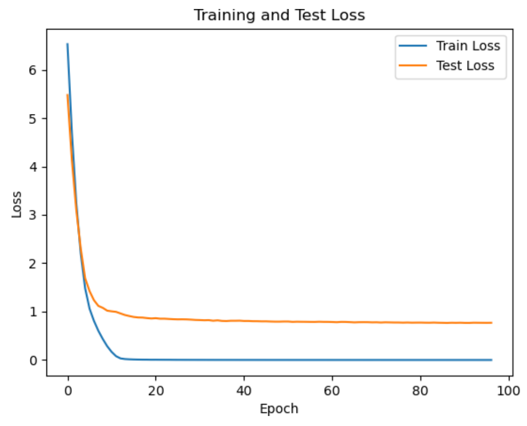
We then aggregate the embeddings into a rich, unified feature representation, followed by standardization and dimensionality reduction using StandardScaler and PCA, preserving 99% of the variance.

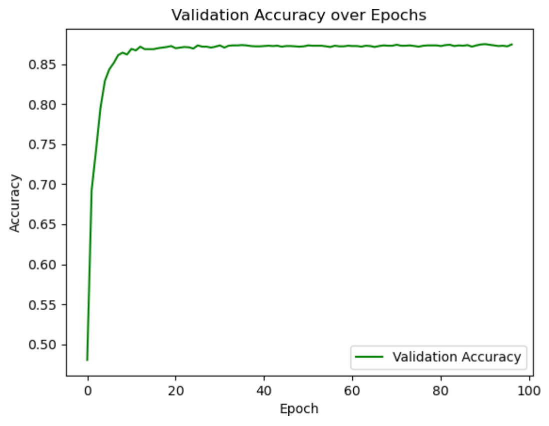
The reduced feature vectors are used to train a deep MLP classifier, which is then evaluated on test data and used to generate confident predictions for unseen query images.

The MLP classifier is a deep fully connected neural network with five hidden layers that progressively reduce dimensionality, using SiLU activations, batch normalization, and dropout for regularization, designed to effectively map high-dimensional input features to class predictions.

The training setup uses the AdamW optimizer with a learning rate of 5e-4, weight decay of 1e-2, a linear learning rate scheduler with warmup, cross-entropy loss, early stopping after 10 epochs of no improvement, and a maximum of 100 epochs.

Results:





The test loss reaches 0.7650 and the accuracy reaches 0.8746.

4.6. Ideas and models to mention

Several additional experiments were carried out using advanced architectures, including ConvNeXtV2-Huge (384 & 512) [8], ConvNeXtV2-Large (384), SwinV2-Large [9], and BEiT-Large [10], along with various training configurations. However, these models underperformed compared to the best baselines, with test losses remaining above 1.1 and accuracies not exceeding 78%.

To further explore performance gains, I attempted to create a custom ensemble model by combining Swin-Large, ConvNeXt-XL, and BEiT-Large backbones. Their features were fused and passed through a fully connected classification head, but the model failed to deliver satisfactory results.

Another approach involved fine-tuning a multi-backbone architecture that fused normalized embeddings from ConvNeXt-XL, Swin-Large, and MaxViT, using a GLU-based classification head. Despite the architectural sophistication, this setup did not yield a meaningful performance boost over simpler models.

I also explored model ensembling techniques:

- A Model Soup approach averaged the weights of multiple ConvNeXt-XL models using a weighted strategy. While the fused model generalized reasonably well, it still did not outperform the best individual models.
- A soft ensemble was implemented by blending the output probabilities of ConvNeXt-XL and MaxViT using a weighted average. Although promising in theory, this method also failed to surpass the standalone models' accuracy.

Additionally, I tested another approach by training an XGBoost classifier on the concatenated embeddings from ConvNeXt and MaxViT. After applying standard scaling and PCA for dimensionality reduction, the model trained and inferred rapidly. However, its performance remained slightly below that of the top-performing neural networks.

Lastly, I experimented with integrating auxiliary metadata, such as date and orientation, into the classification pipeline. The date field was transformed into features like year, month, and day, using normalized values or one-hot encoding. Orientation was treated as a categorical attribute and embedded similarly. These metadata features were concatenated with the visual embeddings prior to classification. Unfortunately, these additions did not lead to any measurable improvement in model accuracy.

5. Submissions

The evaluation uses two metrics: BAKS (balanced accuracy on known samples) and BAUS (balanced accuracy on unknown samples). The final score is the geometric mean of BAKS and BAUS to prevent biased results from trivial predictions.

To determine whether an image belongs to a known class or should be classified as a "new individual," we will use a threshold.

After experimenting with all previously trained models across various threshold settings, I found that only a few consistently delivered strong performance. The most notable among them were:

- ConvNeXt-XL (convnext_xlarge.fb_in22k_ft_in1k_384) from previous milestone
- WildFusion (MegaDescriptor-L-384 + Aliked)

Therefore, we devised a fusion strategy that combines the predictions and confidence scores from the following models:

1. ConvNeXt-XL (convnext_xlarge.fb_in22k_ft_in1k_384) from previous milestone
2. WildFusion 1 (MegaDescriptor-L-384 + Aliked) calibrated using the first 1,000 samples from the database set and computed similarities between all query and database images with a batch size B=700
3. WildFusion 2 (MegaDescriptor-L-384 + Aliked) calibrated using the test database dataset and computed similarities between all query images and the database images with a batch size B=None

The predictions from the two WildFusion models were combined using a weighted strategy: confidences were scaled by 0.45 and 0.55 respectively, giving more weight to the second model (WildFusion 2). If both models predicted the same identity, their confidences were averaged and boosted by a factor of 1.2; if they disagreed, the second model’s prediction was selected and its confidence was amplified by 1.82. The final results, including the selected identity and corresponding confidence for each image, were saved to Combined WildFusion.

The predictions from the Combined WildFusion and ConvNeXt-XL models were integrated by comparing their predicted identities and confidence scores. The ConvNeXt-XL confidence was first scaled by 0.25. If both models predicted the same identity, their confidences were averaged and amplified by a factor of 1.7. If the predictions differed, the identity with the higher adjusted confidence was selected. If ConvNeXt-XL was selected, its confidence was further boosted by a factor of 1.2. The final predictions and confidence scores were saved to Submisson.

A confidence threshold of 0.45 is then applied to the Submission. Predictions with confidence scores below this threshold are reassigned the identity "new_individual". The final output, containing only the "image_id" and "identity" columns, is saved as the Final Submission.

After 85 submissions, Final Submission places us in **6th** position out of 234 competitors, with a score of **0.60483** on private submission set.

6. References

[1] - https://www.kaggle.com/competitions/animal-clef-2025/overview

[2] - https://www.kaggle.com/datasets/wildlifedatasets/wildlifereid-10k

[3] – “A ConvNet for the 2020s” by Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, Saining Xie - https://arxiv.org/pdf/2201.03545

[4] – “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows” by Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo - https://arxiv.org/pdf/2103.14030

[5] – “MaxViT: Multi-Axis Vision Transformer” by Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, Yinxiao Li - https://arxiv.org/pdf/2204.01697

[6] – “Mega: Moving Average Equipped Gated Attention” by Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, Luke Zettlemoyer - https://arxiv.org/pdf/2209.10655

[7] – “LightGlue: Local Feature Matching at Light Speed” by Philipp Lindenberger, Paul-Edouard Sarlin, Marc Pollefeys - https://ar5iv.labs.arxiv.org/html/2306.13643?utm_source=chatgpt.com

[8] – “ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders” by Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, Saining Xie - https://arxiv.org/pdf/2301.00808

[9] – “Swin Transformer V2: Scaling Up Capacity and Resolution” by Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, Baining Guo - https://arxiv.org/pdf/2111.09883

[10] – “BEiT: BERT Pre-Training of Image Transformers” by Hangbo Bao, Li Dong, Songhao Piao, Furu Wei - https://arxiv.org/pdf/2106.08254