Multi-Model Fusion with Cross-Attention for Wildlife Classification

GitHub repository

Muhammad Talha [1] **Uzair Ahmed** [1]

Abstract

Automated classification of camera-trap imagery is vital for ecological monitoring, yet accurate recognition of morphologically similar mountain species is hampered by the scarcity of labelled datasets and the limited work focusing on this habitat. We propose a concise multi-model fusion pipeline: (i) **MegaDetectorV6** crops animal regions; (ii) frozen **YOLOv11x_cls** and **EfficientNet_B0** independently extract features; and (iii) a lightweight cross-attention block fuses these representations, the only component we train, with class-balanced loss and strategic sampling to offset severe imbalance. On a geographically distinct test set our method attains **84.94%** accuracy—improving single-classifier baselines by **48.80%** and **55.90%**—and markedly reduces errors on traditionally confused species, highlighting the value of detection-guided, cross-modal fusion for fine-grained mountain-wildlife recognition where annotated data are sparse.

1. Introduction

Camera traps generate vast image datasets vital for ecological and wildlife monitoring, yet the manual analysis required presents a significant bottleneck. Deep learning offers powerful tools to automate the extraction of information from this 'big data', overcoming the limitations of manual processing. However, classifying animals accurately in diverse and uncontrolled environments remains challenging due to factors like variable illumination, distance, occlusion, and the inherent visual similarity between species or blending with the background. This work tackles these difficulties by proposing a deep learning approach specifically for classifying four large mammal groups: bears, big cats, big dogs, and deer. Our framework combines an initial object detection stage with a novel cross-attention mechanism to fuse features from multiple distinct classifier models, aiming for enhanced robustness and accuracy on this challenging classification task.

2. Methodology

Our proposed method for classifying wildlife images into four categories (bear, big cat, big dog, deer) plus an 'other' category for all other objects employs a multi-stage pipeline designed to leverage the strengths of object detection and multi-model feature fusion. The overall framework first localizes potential animal subjects using an object detection model and then classifies these localized regions using a novel cross-attention fusion module that integrates features from two distinct, pre-trained convolutional neural networks (CNNs).

3. Related Work

Camouflaged Object Detection with Feature Decomposition and Edge Reconstruction. Chunming He *et al.* (CVPR 2023) introduce FEDER, which decomposes features into multiple frequency bands via learnable wavelets, enabling a frequency attention module to selectively highlight subtle foreground–background distinctions; they further propose an ODE-inspired edge reconstruction branch as an auxiliary task to refine ambiguous object boundaries, achieving state-of-the-art COD performance with reduced computation and memory overhead[PDF](1)

Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Mohammed S. Norouzzadeh *et al.* (PNAS 2018; arXiv:1703.05830) deliver the first large-scale demonstration of deep CNNs for camera-trap analysis by training ensembles of ResNet-based models to jointly predict species, count, and behavior across 3.2 M images spanning 48 species, achieving 93.8 [PubMed], [PDF](2)

Towards Automatic Wild Animal Monitoring: Identification of Animal Species in Camera-trap Images using Very Deep Convolutional Neural Networks. Alexander Gómez *et al.* (Ecological Informatics 2017; arXiv:1603.06169) pioneer the use of very deep residual architectures (ResNet-101) for wildlife species classification, evaluating robustness to occlusion, intra-class variation, and image bursts, and achieving 88.9 [PDF](3)

3.1. Dataset and Preprocessing

The dataset utilized in this study comprises distinct training and testing sets. The training set contained a total of 6,860 images distributed across the four target classes: bear (600 images), big cats (1,800 images, including diverse species like tigers, leopards, and cheetahs), big dog (1,500 images, including foxes and hyenas), and deer (2,960 images, encompassing various deer species). This composition highlights a significant class imbalance within the training data.

The test set included 300 bear images, 500 big cat images, 500 big dog images, 600 deer images, and an additional 1,195 images belonging to an 'others' category, totaling 3,095 test images used for evaluation.

All images were preprocessed using `torchvision.transforms`. For the test set, images were resized to 224×224 pixels and converted to tensors using `Resize` and `ToTensor`. The custom `AnimalDataset` class managed data loading, preprocessing, and the subsequent detection-based cropping step.

3.2. Animal Detection and Cropping

To focus the classification effort on relevant image regions, we employed a pre-trained object detection model, specifically `MegaDetectorV6` (version `MDV6-yolov9-e`), to identify initial 'animal' bounding boxes within each image. For each image, the bounding box corresponding to the first 'animal' detection was extracted. This bounding box was then expanded by a margin of 25% on all sides to incorporate contextual information, ensuring the padding did not exceed image boundaries. The resulting region was cropped from the original image. In cases where `MegaDetectorV6` did not detect any animals, the image was dropped in training and in inference it was classified as 'other'.

Initially, we used the YOLOv12 detection model, which took a conservative approach to detection. As a result, any image where no animal was detected was passed through unchanged, without cropping. However, when we began incorporating an 'other' class into our model, we realized this approach was no longer viable. It became clear that dropping images without any detected animals was more effective than continuing to train on them. After further research, we transitioned to MegaDetectorV6, which demonstrated significantly better performance in detecting animals, as it was specifically trained on camera trap wildlife datasets. MegaDetectorV6 is now used in our final model, while intermediate versions that do not include the 'other' class still rely on the YOLO detector.

3.3. Feature Extraction Backbones

Following the detection and cropping stage, features were extracted using two distinct, pre-trained CNN backbones:

- **YOLO-based classifier**: We used `YOLOv11x-cls.pt`, accessed via a custom `YOLOFeatureExtractor` module. This model consists of 176 layers with 29,637,064 parameters and requires approximately 112.0 GFLOPs per inference.
- **EfficientNet**: We employed `EfficientNet-B0`, pre-trained on ImageNet, containing 5,288,548 parameters. Features were extracted using a custom `EffNetFeatureExtractor` module.

Crucially, the weights of both these backbone models were frozen (`requires_grad_(False)`) throughout training, ensuring they served solely as fixed feature extractors. These extractors were designed to provide intermediate feature representations from layers preceding the original final classification heads.

We selected these models because they are state-of-the-art in image classification and have been trained on large-scale datasets, some of which contain camera trap or wildlife imagery. Their proven performance in general object recognition tasks, combined with their architectural efficiency and robustness, made them well-suited for extracting meaningful features in the context of wildlife classification.

We also trained the model using only the features of one of these models rather than both, there was a small performance drop for both of these. We provide further details about these below.

3.4. Cross-Attention Fusion Module

The core of our classification approach is the `CrossAttentionFusion` module, designed to synergistically combine features from the YOLO and EfficientNet backbones. The detailed architecture of this module is presented in Table 1.

Table 1. Architecture of the Cross-Attention Fusion Module.

Layer (type:depth-idx)	**Parameters**
CrossAttentionFusion	–
Linear: 1-1 (YOLO Projection)	512,512
Linear: 1-2 (EfficientNet Projection)	655,872
MultiheadAttention: 1-3	1,050,624
Sequential: 1-4 (Classifier Head)	263,428
Total Parameters	**2,482,436**
Trainable Parameters	2,482,436
Non-trainable Parameters	0

This module first projects the feature vectors obtained from both extractors into a common embedding space of dimension 512 using separate linear layers (`yolo_proj`, `effnet_proj`). These projected features are then treated as a two-element sequence. A standard `torch.nn.MultiheadAttention` layer, configured with an embedding dimension of 512 and 8 attention heads (`num_heads = 8`), performs scaled dot-product self-attention (with $\sqrt{d}$ normalization) on this sequence. Prior to attention, we align the two feature streams by trimming or padding to the smallest batch size, ensuring consistency. This mechanism allows the YOLO and EfficientNet representations to mutually inform and refine each other by attending to salient aspects across both feature sets.

The output features from the attention mechanism, corresponding to the refined representations for each backbone, are concatenated. This combined feature vector is subsequently passed through a final classifier head, implemented as an `nn.Sequential` module containing linear layers and ReLU activation, which maps the fused representation to logits for the four target animal classes.

3.5. Training Details

The training process focused exclusively on optimizing the parameters of the `CrossAttentionFusion` module; the object detector and the two feature extraction backbones remained frozen. For reproducibility, we fixed all random seeds (Python, NumPy, and PyTorch) and dynamically determined the fusion input dimensions via a dummy forward pass. Lightweight error handling in the training loop skips empty or invalid batches to prevent interruptions, and we checkpoint the fusion module whenever the epoch loss improves.

To address the significant class imbalance observed in the training data, we employed two specific techniques. First, the `nn.CrossEntropyLoss` function was configured with class weights (`weight=class_weights`) computed to be inversely proportional to the number of training samples for each class, giving higher importance to under-represented classes. Second, the training `DataLoader` utilized a `torch.utils.data.WeightedRandomSampler`, ensuring each batch contains a more balanced class representation by oversampling minority classes and undersampling majority classes.

The model was trained using the `Adam` optimizer with a learning rate of 1×10^{-4}. Training was conducted for 10 epochs with a batch size of 8. All experiments were implemented using the PyTorch framework on an NVIDIA RTX 3060 GPU.

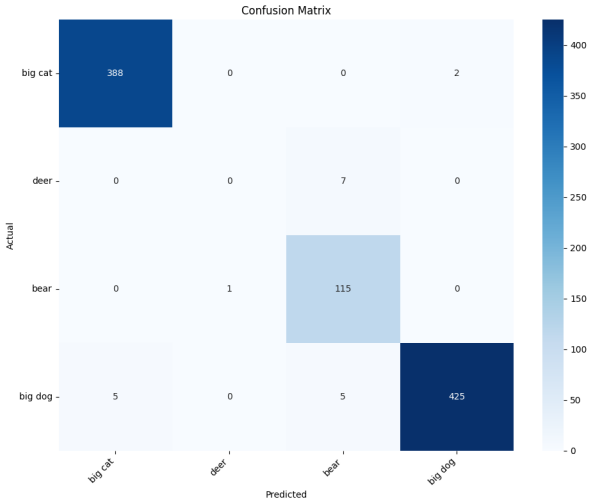

Figure 1. Confusion Matrix for the baseline EfficientNet-B0 model. The matrix reveals complete classification failure for the 'deer' class and moderate performance for 'bear' class.

4. Ablation Study

To thoroughly evaluate our proposed approach and understand the contribution of individual components, we performed a comprehensive ablation study exploring various model configurations. This section describes our systematic exploration from baseline models to our final architecture.

4.1. Baseline Model Performance

We first established baseline performance using the pretrained EfficientNet-B0 and YOLOv11x-cls models independently, applying them directly to the classification task without any fusion or specialized detection pre-processing.

4.1.1. EfficientNet-B0 Baseline

The `EfficientNet-B0 model`, comprising 5,288,548 parameters, achieved an overall accuracy of 48.80% across the four target classes. As shown in Table 2 and Figure 1, this baseline demonstrated high precision for 'big cat' (0.987) and 'big dog' (0.995) categories when it made predictions. However, it notably failed to correctly classify any 'deer' images (Recall = 0.000), indicating a significant limitation in handling this particular class.

Table 2. Detailed Per-Class Results for EfficientNet-B0 Baseline.

Class	Support	Correct	Incorrect	Accuracy	Precision	Recall	F1-Score
Bear	300	115	185	0.383	0.906	0.383	0.539
Big Cat	500	388	112	0.776	0.987	0.776	0.869
Big Dog	500	425	75	0.850	0.995	0.850	0.917
Deer	600	0	600	0.000	0.000	0.000	0.000
Overall Accuracy (4 Classes):				**48.80%**			

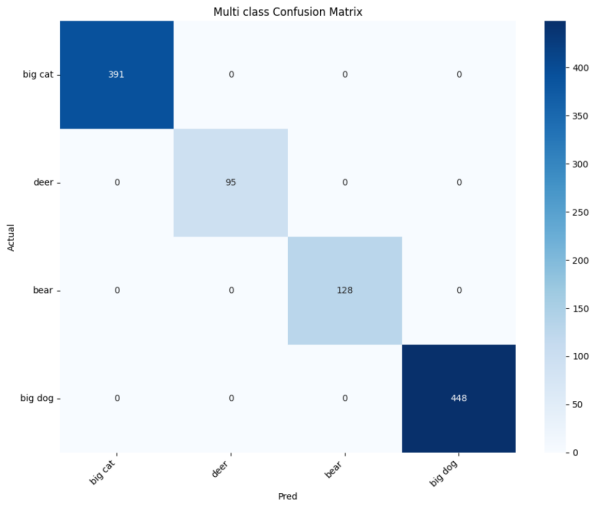

Figure 2. Confusion Matrix for the baseline Yolov11x-cls model.

4.1.2. YOLOv11x-cls Baseline

The `YOLOv11x-cls` model, comprising 176 layers with 29,637,064 parameters, achieved a slightly better overall accuracy of 55.90% across the four target classes. As detailed in Table 4, this model showed strong performance on 'big dog' (Recall = 0.857) and 'big cat' (Recall = 0.820) categories. Unlike EfficientNet-B0, the YOLO-based classifier demonstrated some ability to identify 'deer' images, though with limited recall (0.158). Its confusion matrix is given in this Figure 4

Table 3. YOLOv11x-cls Model Details.

Property	**Value**
Model Type	YOLOv11x-cls
Total Layers	176
Total Parameters	29,637,064
Computational Cost	112.0 GFLOPs
Pre-trained On	ImageNet

Table 4. Detailed Per-Class Results for Yolov11x-CLS Baseline.

Class	**Support**	**Correct**	**Incorrect**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Bear	300	128	172	0.427	0.921	0.427	0.583
Big Cat	477	391	86	0.820	0.869	0.820	0.844
Big Dog	523	448	75	0.857	0.794	0.857	0.824
Deer	600	95	505	0.158	0.979	0.158	0.273
Overall Accuracy (4 Classes):				**55.90%**			

4.2. Development of Fusion Approaches

Following the baseline assessments, we systematically explored various fusion strategies to leverage the complementary strengths of both models.

4.2.1. Initial Cross-Attention Fusion Model (Without 'Other' Class)

Our first significant improvement introduced a detection-crop-classify pipeline using YOLOv11x for localization followed by a cross-attention fusion module that combined features from both EfficientNet-B0 and YOLOv11x-cls. This intermediate model was trained only on the four target animal classes without considering an 'other' category.

As shown in Table 5 and Figure 3, this fusion approach dramatically improved performance, achieving an overall accuracy of 95.37% across the four target classes. The model demonstrated particularly strong improvements for the previously challenging 'deer' class, achieving perfect precision (1.000) and significantly improved recall (0.895).

Table 5. Detailed Per-Class Results for the Intermediate Fusion Model (no 'other' class).

Class	**Support**	**Precision**	**Recall**	**F1-Score**	**Commentary**
Bear	300	0.828	0.997	0.905	Very high recall, moderate precision
Big Cat	500	0.988	0.964	0.976	Excellent precision and recall
Big Dog	500	0.961	0.988	0.974	Strong performance overall
Deer	600	1.000	0.895	0.945	Perfect precision, lower recall
Overall Accuracy (4 Categories):		**95.37%**			
	Macro Average:	0.944	0.961	0.950	
	Weighted Average:	0.959	0.954	0.954	

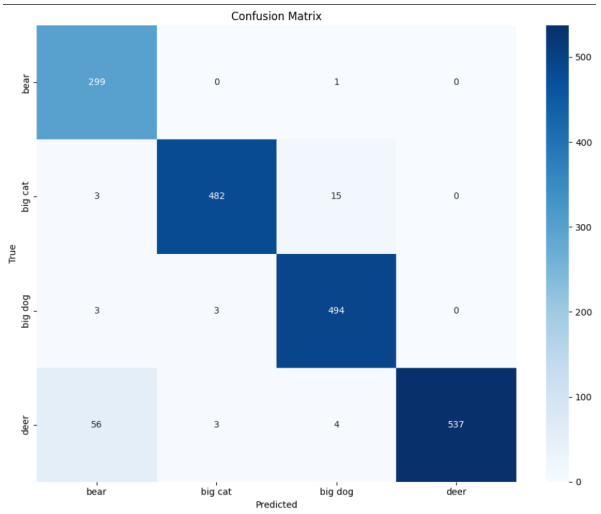

Figure 3. Confusion Matrix for the intermediate model with no 'other'. The matrix shows significantly improved classification performance across all target classes when tested in images of these classes.

4.2.2. Single-Backbone Fusion Variants

To understand the contribution of each backbone to the fusion performance, we developed two additional variants:

1. **YOLOv11x-cls Only Fusion**: In this configuration, we maintained the detection-crop-classify pipeline but used only the YOLOv11x-cls features in the fusion

module, excluding EfficientNet-B0 features. This model showed a small performance drop compared to the full fusion model, suggesting that the EfficientNet features contribute additional discriminative power.

2. **EfficientNet-B0 Only Fusion**: Similarly, we evaluated a variant using only EfficientNet-B0 features in the fusion module. This configuration also demonstrated reduced performance compared to the complete fusion model, confirming that the YOLO features provide complementary information.

While both single-backbone variants outperformed their respective standalone baselines, neither matched the performance of the combined model, validating our hypothesis that feature fusion leverages complementary strengths from each backbone.

4.2.3. Alternative Fusion Mechanisms

We further explored different fusion strategies to determine the optimal method for combining features:

1. **Weighted Average Fusion**: Instead of concatenating attended features, this variant applied learnable weights to create a weighted average of the features from both backbones. The model used softmax normalization to ensure weights summed to 1

2. **Gated Fusion**: This approach implemented a gating mechanism to dynamically control information flow from each backbone, allowing the model to selectively emphasize different feature sources based on input characteristics.

Both alternative fusion mechanisms showed comparable performance to our concatenation-based approach without significant improvements. Given the added complexity without corresponding performance gains, we opted for the simpler concatenation-based fusion in our final model.

4.3. Handling the 'Other' Class Challenge

A critical challenge for practical wildlife classification is distinguishing target classes from all other possible images. We explored several approaches to incorporate an 'other' class:

1. **Direct Classification Training**: Initially, we attempted to include 'other' class samples in the training data. However, this approach proved ineffective due to the immense diversity of potential 'other' images, making it impossible to comprehensively represent this class in the training data.

2. **Maximum Softmax Probability with Temperature Scaling**: We implemented temperature scaling to calibrate model confidence, followed by thresholding on softmax probabilities to identify 'other' class samples. Despite careful temperature optimization on a validation set, this approach incorrectly classified many clear examples of target classes as 'other'.

3. **Detection-Based Filtering with Thresholding**: Our final solution combined two key improvements:

   - Replacing YOLOv11x with MegaDetectorV6, a specialized wildlife detector that demonstrated superior performance in identifying animal regions
   - Implementing a simple yet effective thresholding mechanism on softmax outputs for distinguishing between target and non-target animal classes
   - Classifying images with no animal detections directly as 'other'

This approach balanced the competing requirements of high recall for target classes while effectively filtering out non-target images.

4.4. Summary of Ablation Findings

Our ablation study revealed several key insights:

1. Neither baseline model alone was sufficient for high-quality wildlife classification, with particular challenges in the 'deer' class.

2. The cross-attention fusion mechanism substantially improved performance by leveraging complementary features from both backbones.

3. Both feature sources (YOLOv11x-cls and EfficientNet-B0) contributed to the overall performance, with the combination outperforming either individually.

4. The detection-based preprocessing step was crucial for focusing classification efforts on relevant image regions.

5. Simple concatenation-based fusion performed comparably to more complex fusion mechanisms while maintaining architectural simplicity.

6. The 'other' class was most effectively handled through a combination of specialized detection with MegaDetectorV6 and confidence thresholding.

These findings guided the development of our final proposed model, which is detailed and evaluated in the Results section (Section 5).

5. Results

We present an evaluation of our multi-model fusion framework with cross-attention for wildlife classification, focusing on key performance metrics and comparative analysis.

5.1. Overall Performance

Our model combines frozen `MegaDetectorV6` for detection with `YOLOv11x-cls` and `EfficientNet-B0` features through a trainable `CrossAttentionFusion` head (2.48M parameters). It achieves **84.94%** overall accuracy on the test set containing five categories (four target classes + 'others'), representing a **36.14** percentage point improvement over the best baseline (EfficientNet-B0 at 48.80%).

5.2. Class-wise Performance

Table 6 shows per-class metrics with notable patterns:

Table 6. Per-Class Results for Cross-Attention Fusion Model

Class	Support	Precision	Recall	F1-Score
Bear	300	0.771	0.907	0.833
Big Cat	500	0.977	0.930	0.953
Big Dog	500	0.998	0.926	0.961
Deer	600	0.871	0.632	0.732
Others	1195	0.768	0.879	0.820
Accuracy		**84.94%**		

Key class-specific improvements over baselines include:

- **Deer**: Addressed EfficientNet's complete failure (0% recall) with 63.2% recall
- **Bear**: More than doubled recall from 38.3% (baseline) to 90.7%
- **Big Dog/Cat**: Maintained high performance while adding 'others' handling

5.3. Performance Comparison

Table 7 highlights key advantages over baselines:

Table 7. Model Performance Comparison

Model	Accuracy	Macro F1	Parameters
EfficientNet-B0	48.80%	0.581	5.3M
YOLOv11x-cls	55.90%	0.631	29.6M
Our Model	**84.94%**	**0.860**	**2.5M***

*Trainable parameters (backbones frozen)

The fusion approach demonstrates:

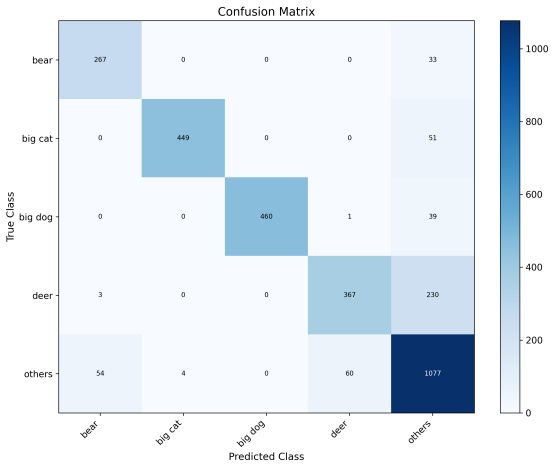

Figure 4. Confusion matrix for the final model including all five classes.

- **36.14%** absolute accuracy gain over best baseline
- Effective 'others' filtering (0.820 F1) absent in baselines
- 12x parameter efficiency vs YOLO baseline (2.5M vs 29.6M)

5.4. Summary

Our cross-attention fusion model demonstrates:

- Effective integration of complementary features through cross-attention
- Robust class-specific improvements, particularly for challenging categories
- Computational efficiency via frozen backbones and focused fusion training

The combination of 84.94% accuracy with efficient parameter usage makes the model practical for real-world ecological monitoring applications where both performance and resource constraints are critical.

6. Discussion

The substantial performance improvements demonstrated by our multi-model fusion framework can be attributed to several key factors:

Complementary Feature Extraction: The two backbone networks, YOLOv11x-cls and EfficientNet-B0, likely extract different but complementary features from the input images due to their distinct architectures and pre-training

regimes. YOLOv11x-cls was originally designed for object detection before being adapted for classification, potentially offering stronger localization capabilities, while EfficientNet-B0 was optimized for image classification with a focus on efficiency. The cross-attention mechanism effectively leverages these complementary strengths.

Cross-Attention Information Flow: Our cross-attention mechanism enables bidirectional information flow between the YOLOv11x-cls and EfficientNet-B0 feature spaces. This allows each model's representations to be refined based on complementary information from the other model. Unlike simple feature concatenation or averaging, cross-attention dynamically emphasizes the most informative aspects of each feature set, particularly helpful for challenging cases where one model may perform better than the other.

Two-Stage Detection Pipeline: The detection-based cropping stage using MegaDetectorV6 serves as an effective attention mechanism at the image level. By expanding detection boxes by 25%, we maintain contextual information while focusing computational resources on regions most likely to contain animals. This approach is particularly effective for wildlife camera trap imagery where animals may occupy varying portions of the frame and appear at different scales.

Handling of Non-Detected Animals and the 'Other' Class: Our approach implements a critical decision boundary by only forwarding crops to the classification stage when MegaDetectorV6 identifies an object as an animal. Images without animal detections or with detections classified as non-animals are automatically categorized as "other," creating an effective filtering mechanism. This design choice helps the fusion model focus exclusively on learning features of target animal classes rather than attempting to learn the vast diversity of non-target objects, significantly improving precision.

Multi-level Class Imbalance Mitigation: Our framework addresses class imbalance at multiple levels - through weighted sampling during training to ensure balanced representation of all classes, and through a weighted loss function that places greater emphasis on underrepresented classes. This dual approach helps prevent the model from being biased toward more frequently observed animals, which is critical for ecological applications where rare species detection is often of particular interest.

7. Conclusion

This paper presented a novel multi-model fusion framework for wildlife classification that combines object detection with cross-attention-based feature fusion. By leveraging the complementary strengths of YOLOv11x-cls and EfficientNet-B0 through a trainable cross-attention mechanism, our approach achieved significant improvements over single-model baselines.

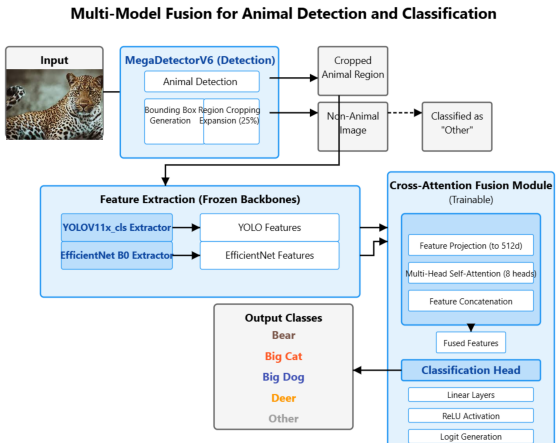

Figure 5. multi-model fusion Architecture.

The proposed model achieved an overall accuracy of 84.94% on a challenging test set containing bears, big cats, big dogs, deer, and non-target species, outperforming the EfficientNet-B0 baseline by 36.14 percentage points. Particularly notable is the successful classification of deer, a class that completely failed under the baseline approach.

These results demonstrate the efficacy of our approach for fine-grained wildlife classification, offering a valuable tool for ecological monitoring and biodiversity conservation efforts. The framework's ability to effectively distinguish between visually similar mammal groups while also filtering out non-target species makes it particularly suitable for processing large-scale camera trap datasets.

Future work could explore extending this approach to a wider range of species, incorporating temporal information from video sequences, and investigating more advanced attention mechanisms to further enhance feature fusion.

References

[1] He, C., Li, K., Zhang, Y., Tang, L., Zhang, Y., Guo, Z., & Li, X. Camouflaged Object Detection with Feature Decomposition and Edge Reconstruction. In *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, pp. 22046–22055, 2023. doi: 10.1109/CVPR52729.2023.02111.

[2] Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., & Clune, J. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National*

Academy of Sciences, 115(25): E5716–E5725, 2018. doi: 10.1073/pnas.1719367115.

[3] Gómez-Villa, A., Salazar, A., & Vargas-Bonilla, J. F. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *Ecological Informatics*, 41: 24–32, 2017. doi: 10.1016/j.ecoinf.2017.07.004.

[4] Beery, S., Morris, D., Yang, S., et al. MegaDetector v6: A general animal detection model for camera trap images. Microsoft AI for Earth, 2023. Available at: https://github.com/microsoft/CameraTraps/blob/main/megadetector.md.

[5] Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. YOLO by Ultralytics: Real-Time Object Detection Models (v8–v12). *Ultralytics*, 2023. Available at: https://github.com/ultralytics/ultralytics.

[6] Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. YOLO by Ultralytics: Version 8 and Classification Models. *Ultralytics*, 2023. Available at: https://github.com/ultralytics/ultralytics.

[7] Tan, M., & Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proc. Int. Conf. Machine Learning (ICML)*, pp. 6105–6114, 2019. doi: 10.48550/arXiv.1905.11946.