International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-025-02602-3

ORIGINAL ARTICLE

ARTEMIS: animal recognition through enhanced multimodal integration system

Edoardo Fazzari[1,2,3] · **Donato Romano**[1,2] · **Fabrizio Falchi**[1,3] · **Cesare Stefanini**[1,2]

Received: 28 October 2024 / Accepted: 3 March 2025

Abstract
This paper introduces Animal Recognition Through Enhanced Multimodal Integration System (ARTEMIS), a transformer-based framework designed for multilabel animal action recognition by fusing video, image, and textual modalities. ARTEMIS utilizes state-of-the-art captioning and language models, such as BLIP2 and Llama 3, to generate textual descriptions from video frames, which are input to the model, significantly enhancing its performance unlikely previous results that do not consider this modality. Through comprehensive ablation studies, we explore the contribution of various model components and propose optimization strategies, including genetic algorithms and reinforcement learning, to dynamically adjust ensemble weights. Our feature alignment techniques-using contrastive and cosine similarity losses-further improve multimodal integration. Evaluations on the Animal Kingdom dataset, which includes 30,100 clips across 140 action classes, demonstrate that ARTEMIS achieves a new state-of-the-art mAP of 79.82, outperforming existing methods. The combination of multimodal fusion and ensemble strategies makes ARTEMIS a robust solution for complex animal action recognition tasks. The code of our fusion method is available at https://github.com/edofazza/ARTEMIS.

Keywords Animal action recognition · Multi-modal deep learning · Ensemble · Genetic algorithm · Reinforcement learning · Contrastive learning

1 Introduction

Animals exhibit a wide range of behaviors, from affectionate and defensive actions to feeding, movement, and aggression. These behaviors are of great interest in ethological studies [1], as they can be indicative of an animal's health and well-being [2]. By associating specific actions with health conditions, researchers can potentially identify pathological behaviors or detect anomalies in wildlife and livestock [3, 4]. This is particularly valuable for environmental monitoring and improving animal welfare [5].

In pursuit of understanding these behaviors, researchers have published datasets [6] and developed methodologies for extracting meaningful information from sensor data, images, and videos [7, 8]. However, much of this research has been limited to individual species and a narrow set of actions [9, 10]. The release of the Animal Kingdom [11] dataset marked a significant shift in this field. Animal Kingdom introduced a dataset encompassing a wide variety of animal species and reorganized actions in a way that allows for cross-species application. The hope is that models trained on this benchmark dataset will generalize across species, paving the way for the development of a foundational model for animal action recognition. Such a model could have widespread applications in agriculture [12], (neuro-)ethology [6], and other research fields, significantly reducing the need for retraining or fine-tuning for specific species.

✉ Edoardo Fazzari
edoardo.fazzari@santannapisa.it

Donato Romano
donato.romano@santannapisa.it

Fabrizio Falchi
fabrizio.falchi@cnr.it

Cesare Stefanini
cesare.stefanini@santannapisa.it

[1] The BioRobotics Institute, Sant'Anna School of Advanced Studies, Viale Rinaldo Piaggio, 56025 Pontedera, Italy

[2] Department of Excellence in Robotics and AI, Sant'Anna School of Advanced Studies, Piazza Martiri della Libertà, 56127 Pisa, Italy

[3] Institute of Information Science and Technologies, National Research Council of Italy, via G. Moruzzi, 56124 Pisa, Italy

Published online: 18 March 2025

Deep learning has emerged as a key tool for solving the problem of animal action recognition. Researchers initially employed Convolutional Neural Networks [13], but more recent efforts have shifted towards multimodal deep learning approaches [14]. However, current models applied to the Animal Kingdom dataset have struggled to achieve satisfactory performance, indicating that further research is required.

In response to these challenges, we propose a novel multimodal approach for recognizing animal actions in videos, images, labels, and video descriptions, which we call ARTEMIS (Animal Recognition Through Enhanced Multimodal Integration Systems), as depicted in Fig. 1. ARTEMIS consists of two branches, where the information is fused in two stages. In the first stage, each branch undergoes an initial fusion between two modalities via simple concatenation: one branch fuses video data with the corresponding video descriptions, while the other combines averaged CLIP embeddings from input frames with label embeddings. The second stage of fusion occurs by integrating the two branches through a Transformer layer, followed by a fully connected layer for final action recognition. We conducted an extensive ablation study to determine which components-such as the sequence processing layers for video descriptions and residual connections-were most effective in improving performance on the Animal Kingdom dataset. To further enhance the results, we employed both average and weighted ensembles. One challenge when using weighted ensembles is determining the appropriate weights. To address this, we utilized genetic algorithms and reinforcement learning. Genetic algorithms are well-suited for optimizing a fitness function and have proven effective through evolutionary mechanisms [15], though they do not guarantee the discovery of the optimal solution [16]. On the other hand, reinforcement learning has a strong track record in improving deep learning results [17], with applications spanning diverse fields such as physics [18] and video game playing [19]. Another critical aspect of our research was generating video descriptions as input to our models. The Animal Kingdom dataset does not provide textual descriptions of the actions occurring in each clip, only the video clips and frames. Manually obtaining this information from experts is impractical, as it is not automated or scalable. Therefore, we used deep learning models, leveraging captioning techniques and large language models (LLMs), to automatically generate these descriptions.

The main contributions of this paper are as follows:

1. We introduce ARTEMIS, a novel multimodal architecture for animal action recognition that achieves state-of-the-art performance. Unlike previous approaches, ARTEMIS integrates textual descriptions of videos into its processing pipeline.
2. We utilize large language models (LLMs) to generate textual descriptions of videos and augment the Animal Kingdom dataset with these descriptions.
3. We introduce a set of optimization strategies, including ensemble methods, genetic algorithms, and reinforcement learning, to further improve model performance.
4. All generated descriptions and source code are made publicly available to support reproducibility and further research.

The remainder of this paper is organized as follows: Sect. 2 reviews related work on animal action recognition (AAR). In Sect. 3, we detail the network architecture, fusion modules, additional loss functions for feature alignment, and optimization strategies. Section 4 provides experimental evaluations and discusses the results, and conclusions are drawn in Sect. 5.

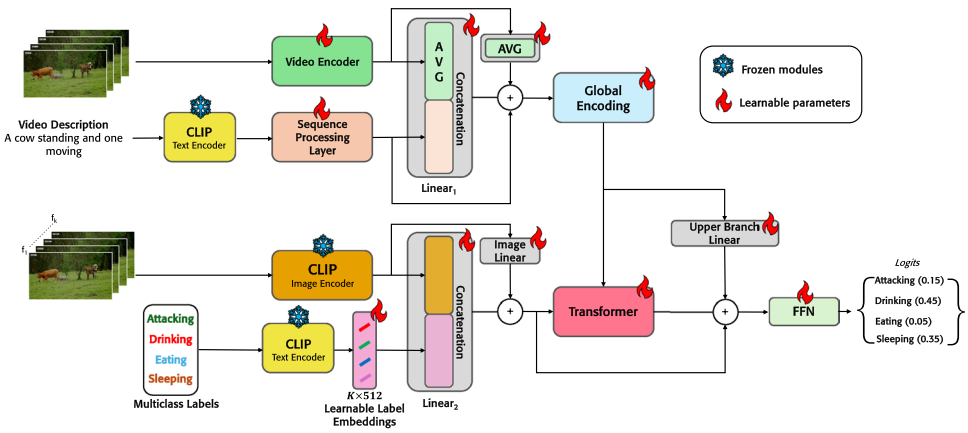

Fig. 1 Overview of ARTEMIS for multimodal multilabel animal action recognition. It has three components: a video-text fusion branch, an image-text fusion branch and a multimodal Transformer decoder

2 Related work

Animal Action Recognition (AAR) initially emerged as an extension of Human Action Recognition (HAR) [20], with deep learning strategies from HAR being directly adapted to this new domain. Two primary approaches dominated early AAR research [21]: pose estimation of animal body parts, followed by the association of these movements to specific actions through classical machine learning or deep learning methods; and the use of sensor data for action recognition.

The first approach significantly advanced animal pose estimation techniques, although these developments were often driven by progress in human pose estimation. Tools such as SLEAP [22], DeepLabCut [7], DeepEthogram [23], and MARS [8] have been instrumental in enabling researchers to study animal behavior for purposes ranging from improving farming welfare-such as detecting feeding and fighting behaviors in pigs [24]-to neuroethological investigations, like studying the effects of diseases [25]. On top of pose estimation, machine learning models such as Support Vector Machines (SVMs), Long Short-Term Memory networks (LSTMs), or convolutional neural networks are commonly applied to identify specific behaviors [26, 27].

While pose estimation-based AAR begins with visual input from videos or images, sensor-based AAR relies on temporal data captured by gyroscopes, GPS, and accelerometers [28]. This sensor data is fused and processed using machine learning algorithms, primarily for recognizing actions in livestock such as cows and sheep [29]. However, the use of sensors comes with significant limitations, particularly for animals in the wild. Sensor readings can be affected by external factors such as temperature, humidity, or vibrations, leading to unreliable data [30]. Moreover, improper placement of sensors can result in poor data quality or misinterpretation of actions. Additionally, wearable sensors may be intrusive or uncomfortable, potentially interfering with the animals' natural behaviors [31].

In multimodal learning, one possible fusion strategy involves combining visual and textual data. This is widely used in tasks such as image captioning [32], visual question answering [33], and generating images from textual descriptions [34]. In AAR, textual information has also been leveraged to enhance recognition accuracy. For instance, MSQNet [14] employs text embeddings of action labels generated by CLIP to improve the model's ability to recognize actions in the Animal Kingdom dataset's multilabel action recognition task [11]. MSQNet outperformed previous models that relied solely on video data. Further advancements were made with Mamba-MSQNet [35], a more efficient version of MSQNet that uses fewer parameters and computational resources while achieving superior mean Average Precision (mAP).

Even if the Animal Kingdom is the biggest and more complete dataset for animal action recognition, spanning 140 distinct actions over 850 different species, making it being considerable as a benchmark dataset for AAR it is not the only one available. Quite recently, other research efforts in providing public dataset for this topic became available, BaboonLand and MammalNet. The BaboonLand dataset [36] is a collection of mini-scenes limited on a single species, baboons, performing 12 distinct actions and an additional actions for marking videos where no animal is visible. The MammalNet dataset [37] focused only on mammals and their most common behaviors, limiting the number of species to 173, way less than Animal Kingdom, and only 12 actions.

While Animal Kingdom is the largest and most comprehensive dataset for animal action recognition-covering 140 distinct actions across 850 species, establishing itself as a benchmark dataset for AAR, it is not the only one available. Recently, other research efforts have provided additional public datasets for this topic, notably BaboonLand and MammalNet. The BaboonLand dataset [36] is a focused collection of mini-scenes featuring a single species, baboons, performing 12 distinct actions, along with an additional category for frames where no animal is visible. On the other hand, the MammalNet dataset [37] concentrates exclusively on mammals and their most common behaviors, reducing its scope to 173 species and only 12 actions, significantly fewer than Animal Kingdom.

3 Proposed framework

In this section, we design the ARTEMIS model. We begin by describing the architecture and specific components of our model, highlighting how the model operates to achieve effective action recognition. Following that, we explain the process used to generate video description for each video clip and what we used to further improve ARTEMIS.

3.1 ARTEMIS details

We present ARTEMIS, the Animal Recognition Through Enhanced Multimodal Integration System, designed for multilabel multimodal action recognition in animals. As illustrated in Fig. 1, the model is composed of three key components: (1) a *video-text fusion branch*, which processes and integrates video information from a video encoder with embeddings of the video description; (2) an *image-text fusion branch*, which combines embeddings from individual frames with action-specific class embeddings; and (3) a multimodal Transformer decoder, which performs

late-stage fusion of the two branches and generates the final predictions.

3.1.1 Video-text fusion branch

In the video-text fusion branch, video data is processed to extract frame-level features, which are subsequently fused with external textual information in the form of summary embeddings. This branch begins with video input consisting of t frames, where each frame $\mathbf{I}$ is represented as a tensor of size $C \times H \times W$, with C channels, and spatial dimensions H and W. These frames are passed through a Timesformer backbone [38], which is a transformer-based architecture for spatiotemporal learning from video data. The Timesformer extracts frame-level feature representations, $\mathbf{X} \in \mathbb{R}^{T \times D}$, where $T \times D$ represents the dimension of the latent feature space.

To align the first dimension with the expected temporal resolution in subsequent operations, we apply *adaptive average pooling*. This operation down-samples T to t through:

$$\mathbf{X}' = \text{AdaptiveAvgPool1D}(\mathbf{X}^\top, t)^\top, \tag{1}$$

where $\mathbf{X}' \in \mathbb{R}^{t \times D}$ represents a compressed feature map that reduces the temporal size while maintaining the final dimensions intact.

Simultaneously, textual data corresponding to the video is processed. The textual video descriptions are embedded into a latent space through CLIP text encoder [39], producing embeddings $\mathbf{S} \in \mathbb{R}^E$, where E is the dimension of the embedding space. Depending on the model configuration, this embedding may be transformed using a recurrent. If a BiLSTM is employed, the text embedding becomes:

$$\mathbf{S}' = \text{BiLSTM}(\mathbf{S}), \tag{2}$$

where the output maintains the same embedding dimension E. Other recurrent options like biGRU or 1D convolution could similarly be used for the summary transformation.

This textual embedding is then expanded across the temporal dimension to match the length t of the video feature sequence:

$$\mathbf{S}' \otimes \mathbf{1}_t \in \mathbb{R}^{t \times E}. \tag{3}$$

The expanded textual embeddings are concatenated with the pooled video features $\mathbf{X}'$, resulting in a concatenated tensor $\mathbf{Z} \in \mathbb{R}^{t \times (D+E)}$. This tensor is then passed through a linear transformation:

$$\mathbf{H}_v = \mathbf{Z}\mathbf{W}_1, \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{(D+E) \times E}$, producing the output $\mathbf{H}_v \in \mathbb{R}^{t \times E}$, which represents the fused video-text features.

A residual connection is used to preserve important information from the original Timesformer backbone and the sequence features. First, the original video features $\mathbf{X}$ are passed through a linear layer and averaged over time:

$$\mathbf{R}_{\text{backbone}} = \text{AvgPool}(\mathbf{X}\mathbf{W}_{\text{res}_1}), \tag{5}$$

where $\mathbf{W}_{\text{res}_1} \in \mathbb{R}^{D \times E}$ transforms the backbone features into the appropriate dimensionality. Similarly, the fused video-text features $\mathbf{H}_v$ are linearly transformed into a residual form:

$$\mathbf{R}_{\text{sequence}} = \mathbf{H}_v\mathbf{W}_{\text{seq}}, \tag{6}$$

where $\mathbf{W}_{\text{seq}} \in \mathbb{R}^{E \times E}$. These two residuals are summed together before applying the positional encoding:

$$\mathbf{R}_{\text{total}} = \mathbf{R}_{\text{backbone}} + \mathbf{R}_{\text{sequence}}, \tag{7}$$

To preserve temporal context, a positional encoding is applied to $\mathbf{H}_v$, which adds temporal positional information to each frame in the sequence. The positional encoding process is mathematically described as:

$$\mathbf{H}v^{\text{pos}} = \mathbf{R}_{\text{total}} + \text{PE}(t), \tag{8}$$

where $\text{PE}(t) \in \mathbb{R}^{t \times E}$ is the positional encoding for the sequence length t. This addition enables the model to incorporate the temporal order of frames, critical for understanding the temporal dynamics of actions.

This process produces a final fused representation for the video-text branch, $\mathbf{H}_v^{\text{final}} = \mathbf{H}_v^{\text{pos}}$, which integrates both the spatiotemporal features from the video and the contextual information from the textual summary. The residual connections ensure that important features from both the backbone and sequence fusion are preserved throughout the process.

3.1.2 Image-text fusion branch

Parallel to the video-text branch, the image-text fusion branch operates on individual video frames. Each frame $\mathbf{I}_t$ is processed through the CLIP image encoder, producing a feature vector $\mathbf{F}_t \in \mathbb{R}^D$, which shares the same embedding dimension E as the textual representation. To summarize the frame-level information, the features from all t frames are aggregated using a temporal average pooling operation:

$$\mathbf{F} = \frac{1}{t}\sum_{i=1}^{t} \mathbf{F}_i, \tag{9}$$

which results in a single vector $\mathbf{F} \in \mathbb{R}^D$ that captures the visual content across the entire video sequence.

At the same time, a separate CLIP text encoder processes the set of action labels associated with the video. These action labels are embedded into the same space as

the image embeddings, yielding a matrix $\mathbf{L} \in \mathbb{R}^{K \times E}$, where K represents the number of action labels, and each row corresponds to the embedding of a specific label.

To perform fusion, the image embedding $\mathbf{F}$ is repeated K times and concatenated with the corresponding action label embeddings:

$$\mathbf{Q} = \text{concat}(\mathbf{L}, \mathbf{F} \otimes \mathbf{1}_K) \in \mathbb{R}^{K \times (D+E)}. \tag{10}$$

A linear transformation then projects this concatenated representation into the shared embedding space E:

$$\mathbf{H}_q = \mathbf{Q}\mathbf{W}_2, \tag{11}$$

where $\mathbf{W}_2 \in \mathbb{R}^{(D+E) \times E}$. The result, $\mathbf{H}_q \in \mathbb{R}^{K \times E}$, is a fused representation that encapsulates both visual and textual information about the actions.

To enhance the expressiveness of the model, we optionally introduce a residual connection for the image features. The image embedding $\mathbf{F}$, repeated across all K action labels, is linearly transformed:

$$\mathbf{F}_{\text{residual}} = \mathbf{F}\mathbf{W}_{\text{res}_2} \in \mathbb{R}^{K \times E}. \tag{12}$$

This residual is then added to the fused image-text representation, resulting in the final output of this branch:

$$\mathbf{H}_q^{\text{final}} = \mathbf{H}_q + \mathbf{F}_{\text{residual}}. \tag{13}$$

This image-text fusion branch thus integrates action-level textual information with visual representations of the video, with residual connections used to enhance the fusion process.

3.1.3 Late fusion and recognition

In the late fusion stage, the outputs from the two fusion branches-$\mathbf{H}_v^{\text{final}} \in \mathbb{R}^{t \times E}$ from the video-text fusion branch and $\mathbf{H}_q^{\text{final}} \in \mathbb{R}^{K \times E}$ from the image-text fusion branch-are combined to form a comprehensive multimodal representation of the video.

First, the concatenated outputs of the two branches are passed into a Transformer layer [40], with $\mathbf{H}_v^{\text{final}}$ acting as one set of tokens and $\mathbf{H}_q^{\text{final}}$ as another. This concatenation ensures that the model can learn complex dependencies between the temporal features of the video, the descriptive text, the global image features, and the action-specific textual information:

$$\mathbf{H}_{\text{concat}} = \text{concat}(\mathbf{H}_v^{\text{final}}, \mathbf{H}_q^{\text{final}}) \in \mathbb{R}^{(t+K) \times E}. \tag{14}$$

The Transformer layer applies multi-head self-attention, where each head learns distinct attention patterns between the concatenated inputs. The self-attention mechanism computes an attention score for each pair of tokens, allowing the model to focus on the most relevant parts of the multimodal inputs:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{E}}\right), \tag{15}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value matrices derived from the concatenated inputs $\mathbf{H}_{\text{concat}}$. The output of the multi-head attention is then passed through a feed-forward network, which further refines the fused representation. The final output of the Transformer layer, $\mathbf{H}_{\text{fused}} \in \mathbb{R}^{K \times E}$, integrates in this way the information from both the video-text and image-text branches.

At this stage, two residual connections are incorporated into the fused representation $\mathbf{H}_{\text{fused}}$ to enhance the flow of information and mitigate potential degradation of the learned features. These residuals correspond to the final outputs of the video-text fusion branch $\mathbf{H}_v^{\text{final}}$ and the image-text fusion branch $\mathbf{H}_q^{\text{final}}$. The addition of these residuals ensures that crucial multimodal features from both branches are preserved in the final fused representation. Formally, the updated fused representation $\mathbf{H'}_{\text{fused}}$ is given by:

$$\mathbf{H'}_{\text{fused}} = \mathbf{H}_{\text{fused}} + \mathbf{H}_q^{\text{final}} + \mathbf{H}_v^{\text{final}} W_j,$$

where $W_j \in \mathbb{R}^{t \times K}$ is a learnable weight matrix that aligns the dimensionality of $\mathbf{H}_v^{\text{final}}$ with that of $\mathbf{H}_{\text{fused}}$, ensuring compatibility during the fusion process.

This representation $\mathbf{H'}_{\text{fused}}$ is then used to predict the occurrence of specific actions in the video. A final fully forward layer is applied to each of the K action classes, producing a score for each action:

$$\hat{y}i = \mathbf{H'}_{\text{fused}}\mathbf{W}_i + b_i, \tag{16}$$

where $\mathbf{W}_i \in \mathbb{R}^E$ is the weight matrix for action i, and b_i is the corresponding bias term. This enables the model to output a probability distribution over all potential actions, completing the recognition task.

3.2 Obtaining video descriptions

One of the modalities used in ARTEMIS is a textual description that summarizes the input video. Since the Animal Kingdom dataset does not provide this information, we generated the descriptions ourselves. Given an input video $\mathbf{I}$ with frames $I_1, \ldots, I_K$, each frame is passed through a captioning model, producing individual captions $c_1, \ldots, c_K$, collectively denoted as $\mathbf{C}$. To generate these captions, we used BLIP2 [41] with a maximum caption length of 50 and a beam search of size 4.

The set $\mathbf{C}$ provides frame-specific descriptions, but does not capture a holistic summary of the video. To create a

concise video description, **C** is reduced to a subset **C'**, containing only unique captions. We then construct a query **Q** by appending the text: "Can you write a very short sentence summarizing the information below, focusing on the actions of the animals?" to $\mathbf{C}'$. This query is fed into LLaMA 3-8B [42], a pretrained Large Language Model, which generates the final summary used as the video description input for ARTEMIS.

3.3 Feature alignment

A common approach in multimodal learning is to encourage the alignment or similarity of features across modalities within a shared embedding space. This helps to ensure that representations from different modalities are semantically aligned [43]. In our model, we aimed to align the features of the video-text fusion branch ($\mathbf{f}_{b1}$) with those of the image-text fusion branch ($\mathbf{f}_{b2}$). To achieve this, we applied the techniques: a contrastive loss, a cosine similarity loss and Canonical Correlation Analysis (CCA).

The contrastive loss [44] is designed to minimize the distance between semantically similar pairs and maximize the distance between dissimilar pairs in the embedding space. It is defined as:

$$\begin{aligned}\mathcal{L}_{\text{contrastive}} &= y \cdot \text{dist}(\mathbf{f}_{b1}, \mathbf{f}_{b2}) \\ &\quad + (1-y) \cdot \max(0, \text{margin} - \text{dist}(\mathbf{f}_{b1}, \mathbf{f}_{b2})),\end{aligned} \tag{17}$$

where $y = 1$ if the pair shares the same labels, and $y = 0$ otherwise. The distance metric 'dist' used is the Euclidean distance, with a margin set to 1.

The cosine similarity loss [45] is employed to maximize the cosine similarity between features from different modalities. This method is particularly effective for aligning features in a shared embedding space and is defined as:

$$\mathcal{L}_{\text{cosine}} = 1 - \cos(\mathbf{f}_{b1}, \mathbf{f}_{b2}). \tag{18}$$

Additionally, we employed Canonical Correlation Analysis (CCA) [46], a statistical technique used to find and maximize the correlation between two sets of variables by identifying linear transformations of the data that are maximally correlated. When integrated into a deep learning model, CCA can be formulated as a loss function. The goal is to maximize the correlation between the learned projections of the two modalities by minimizing the negative correlation. The CCA loss function is formulated as:

$$\mathcal{L}_{\text{CCA}} = -\frac{\text{Tr}(W_{\mathbf{f}_{b1}}^T \Sigma_{\mathbf{f}_{b1}\mathbf{f}_{b2}} W_{\mathbf{f}_{b2}})}{\sqrt{\text{Tr}(W_{\mathbf{f}_{b1}}^T \Sigma_{\mathbf{f}_{b1}\mathbf{f}_{b1}} W_{\mathbf{f}_{b1}})} \cdot \sqrt{\text{Tr}(W_{\mathbf{f}_{b2}}^T \Sigma_{\mathbf{f}_{b2}\mathbf{f}_{b2}} W_{\mathbf{f}_{b2}})}}, \tag{19}$$

where $W_{\mathbf{f}_{b1}}$ and $W_{\mathbf{f}_{b2}}$ are the linear transformation matrices for $\mathbf{f}_{b1}$ and $\mathbf{f}_{b2}$, respectively. $\text{Tr}(\cdot)$ represents the trace of a matrix, while $\Sigma_{\mathbf{f}_{b1}\mathbf{f}_{b2}}$, $\Sigma_{\mathbf{f}_{b1}\mathbf{f}_{b1}}$, and $\Sigma_{\mathbf{f}_{b2}\mathbf{f}_{b2}}$ are the cross-covariance and auto-covariance matrices between and within the modalities. The minus sign is used to maximize the correlation between the two sets of features during the training of the neural networks.

Only one of these alignment losses is applied at a time during training. Whichever is used, it is added to the binary cross-entropy loss, with the feature alignment loss weighted by a factor of 0.1 (0.0001 for CCA) before being combined with the primary loss function.

3.4 Ensemble strategies

To improve results we decided to create ensemble models using the best configuration of ARTEMIS we obtained. Ensemble in deep learning is a popular approach to improve results and consists in using multiple trained architecture and combine to make a single prediction, producing better, more accurate and robust predictions than any single model on its own.

In our experiments, three types of ensemble are defined: (1) an average ensemble giving equal weight to each model used; (2) a weighted ensemble where the weights assigned are obtained by using a genetic algorithm; (3) a weighted ensemble where the weights are dynamically decided based on the input given through the used on reinforcement learning.

The first approach is defined as:

$$F(x) = \frac{1}{M} \sum_{i=1}^{M} f_i(x), \tag{20}$$

where M is the number of models in the ensemble, $f_i(x)$ denotes the prediction of the ith model for a given input x. In case of weighted ensemble like in the other two cases, the formulation chages to:

$$F(x) = \frac{\sum_{i=1}^{M} w_i f_i(x)}{\sum_{i=1}^{M} w_i}, \tag{21}$$

where w_i is the weight assigned to the ith model, which is normalized such that $\sum_{i=1}^{M} w_i = 1$.

3.4.1 Genetic algorithm weights

To optimize the weights for the ensemble, we employed a genetic algorithm (GA) [47], which works by evolving a population of weight vectors over multiple generations. The process is described as follows:

- *Initialization* We randomly generated an initial population of 15 individuals. Each individual is represented as a vector of M entries, corresponding to the number of models, with random values between 0 and 1. The weights are normalized so that $\sum_{i=1}^{M} w_i = 1$.
- *Evaluation* The fitness of each individual is evaluated using the mean Average Precision (mAP) on a validation set, obtained through 5-fold cross-validation on the Animal Kingdom dataset.
- *Selection* Tournament selection [48] is used, where 3 individuals are randomly chosen, and the one with the highest fitness is selected for reproduction. Additionally, elitism is employed to ensure that the top 2 individuals from the current generation are directly carried over to the next generation.
- *Crossover* We used BLX-α (Blend Crossover) [49] to generate new offspring from two parent individuals. For each gene i, the offspring (o_1, o_2) are created as:

$$o_{1,i} = \gamma p_{1,i} + (1-\gamma) p_{2,i}, \tag{22}$$

$$o_{2,i} = \gamma p_{2,i} + (1-\gamma) p_{1,i}, \tag{23}$$

  where $p_{1,i}$ indicates the parent p_1 gene i and γ is defined as:

$$\gamma = (1-2\alpha)\cdot r + \alpha \tag{24}$$

  where α is a parameter controlling the blending range, set in our case to 0.5, and r is a random number sampled uniformly between 0 and 1. After generating the offspring, normalization is applied to ensure that the sum of the weights remains 1.
- *Mutation.* Mutation is applied with a probability of 0.2, where each gene o_i is perturbed by adding a Gaussian-distributed random value:

$$o_i' = o_i + \mathcal{N}(\mu, \sigma) \tag{25}$$

  where $\mathcal{N}(\mu, \sigma)$ is a random sample from a normal distribution with mean μ and standard deviation σ.
- *Termination.* The genetic algorithm terminates after 10 epochs.

3.4.2 Reinforcement learning weights

Unlike the static weights obtained through the genetic algorithm, the reinforcement learning approach dynamically assigns weights based on the input. The goal is to create an agent that learns to predict a set of weights tailored to each input sample. To this end, we used several deep reinforcement learning algorithms, specifically Proximal Policy Optimization (PPO) [50].

The agent receives a single observation, which consists of the input video frames, merged across the frame and channel dimensions. The agent's action is to output a set of weights, and the reward is calculated as the negative of the loss function for that input. Updates are processed after all the data in the validation set is viewed by the agent, and each update conducts a training of 3 epochs. Finally, the agent learns over a number of timesteps equal to the length of the validation set multiplied by 50.

We used the default parameters of the reinforcement learning algorithms from the stable-baselines3 library [51], and a convolutional neural network for the policy.

4 Experimental results and discussion

4.1 Experiment settings

Our models were trained using the Animal Kingdom dataset [11], which consists of 30,100 video clips depicting animals performing various actions, encompassing a total of 140 action classes. Training was divided into two phases: initially, the video encoder was kept frozen for 50 epochs, allowing only the other layers to train. Afterward, the video encoder was unfrozen and trained for an additional 150 epochs, or 200 epochs in the case of ARTEMIS and the top three ablation models, as well as for the contrastive loss analysis. All training was conducted using an NVIDIA A100 GPU.

We used binary cross-entropy with logits as the loss function, with Adam [52] as the optimizer. Additionally, a cosine annealing scheduler with warm restarts was employed [53], resetting every 10 iterations.

For data augmentation, we adopted the approach from Mamba-MSQNet [35], applying multi-scale cropping, horizontal flipping, random color jittering, and random grayscale effects.

4.2 Performance

4.2.1 Ablation study

An ablation study was conducted to assess the impact of various residual connections in ARTEMIS and the choice of processing layers for video descriptions on performance. Table 1 outlines all tests performed, which included switching between no processing layer, biLSTM, GRU, or 1-D convolution for each possible combination of residual connections. Among all tested models, we identified three that outperformed the others: BEST_1 ($\mathbf{R}_{\mathbf{H}_v^{\text{final}}}$, $\mathbf{R}_{\text{backbone}}$, $\mathbf{F}_{\text{residual}}$, conv), BEST_2 ($\mathbf{R}_{\mathbf{H}_v^{\text{final}}}$, $\mathbf{R}_{\text{backbone}}$, conv), and BEST_3

Table 1 Ablation study comparing different sequence processing layers for the text input to our model and various configurations of residual connections

Text processing	$R_{H_v^{\text{final}}}$	R_{sequence}	R_{backbone}	$R_{H_q^{\text{final}}}$	F_{residual}	mAP
–	×	×	×	×	×	75.96
biLSTM	×	×	×	×	×	76.23
GRU	×	×	×	×	×	75.91
Conv	×	×	×	×	×	76.17
–	✓	×	×	×	×	76.04
biLSTM	✓	×	×	×	×	75.87
GRU	✓	×	×	×	×	75.71
Conv	✓	×	×	×	×	76.52
–	×	✓	×	×	×	74.96
biLSTM	×	✓	×	×	×	76.19
GRU	×	✓	×	×	×	75.96
Conv	×	✓	×	×	×	75.21
–	✓	✓	×	×	×	75.49
biLSTM	✓	✓	×	×	×	76.51
GRU	✓	✓	×	×	×	76.09
Conv	✓	✓	×	×	×	75.78
–	×	×	✓	×	×	76.29
biLSTM	×	×	✓	×	×	76.46
GRU	×	×	✓	×	×	76.07
Conv	×	×	✓	×	×	76.30
–	✓	×	✓	×	×	76.50
biLSTM	✓	×	✓	×	×	76.15
GRU	✓	×	✓	×	×	75.78
Conv	✓	×	✓	×	×	76.57 (77.19)
–	×	✓	✓	×	×	75.13
biLSTM	×	✓	✓	×	×	76.39
GRU	×	✓	✓	×	×	75.78
Conv	×	✓	✓	×	×	75.75
–	✓	✓	✓	×	×	76.02
biLSTM	✓	✓	✓	×	×	76.24
GRU	✓	✓	✓	×	×	75.66
Conv	✓	✓	✓	×	×	76.28
–	×	×	×	✓	×	75.26
biLSTM	×	×	×	✓	×	75.75
GRU	×	×	×	✓	×	75.40
Conv	×	×	×	✓	×	75.37
–	✓	×	×	✓	×	75.78
biLSTM	✓	×	×	✓	×	76.12
GRU	✓	×	×	✓	×	75.93
Conv	✓	×	×	✓	×	75.68
–	×	✓	×	✓	×	73.54
biLSTM	×	✓	×	✓	×	75.32
GRU	×	✓	×	✓	×	74.83
Conv	×	✓	×	✓	×	74.74
–	✓	✓	×	✓	×	74.63
biLSTM	✓	✓	×	✓	×	75.75
GRU	✓	✓	×	✓	×	75.39
Conv	✓	✓	×	✓	×	74.92
–	×	×	✓	✓	×	75.85
biLSTM	×	×	✓	✓	×	76.16
GRU	×	×	✓	✓	×	75.63

Table 1 (continued)

Text processing	$R_{H_v^{\text{final}}}$	R_{sequence}	R_{backbone}	$R_{H_q^{\text{final}}}$	F_{residual}	mAP
Conv	×	×	✓	✓	×	75.92
–	✓	×	✓	✓	×	76.50
biLSTM	✓	×	✓	✓	×	76.15
GRU	✓	×	✓	✓	×	75.78
Conv	✓	×	✓	✓	×	76.57 (76.97)
–	×	✓	✓	✓	×	75.13
biLSTM	×	✓	✓	✓	×	76.39
GRU	×	✓	✓	✓	×	75.78
Conv	×	✓	✓	✓	×	75.75
–	✓	✓	✓	✓	×	76.02
biLSTM	✓	✓	✓	✓	×	76.24
GRU	✓	✓	✓	✓	×	75.66
Conv	✓	✓	✓	✓	×	76.28
–	×	×	×	×	✓	76.10
biLSTM	×	×	×	×	✓	76.49
GRU	×	×	×	×	✓	75.99
Conv	×	×	×	×	✓	76.44
–	✓	×	×	×	✓	76.18
biLSTM	✓	×	×	×	✓	75.59
GRU	✓	×	×	×	✓	75.68
Conv	✓	×	×	×	✓	76.21
–	×	✓	×	×	✓	74.73
biLSTM	×	✓	×	×	✓	76.24
GRU	×	✓	×	×	✓	75.39
Conv	×	✓	×	×	✓	75.59
–	✓	✓	×	×	✓	75.51
biLSTM	✓	✓	×	×	✓	75.89
GRU	✓	✓	×	×	✓	75.35
Conv	✓	✓	×	×	✓	76.21
–	×	×	✓	×	✓	76.19
biLSTM	×	×	✓	×	✓	75.92
GRU	×	×	✓	×	✓	76.01
Conv	×	×	✓	×	✓	76.57
–	✓	×	✓	×	✓	76.15
biLSTM	✓	×	✓	×	✓	76.11
GRU	✓	×	✓	×	✓	75.67
Conv	✓	×	✓	×	✓	76.71 (**77.33**)
–	×	✓	✓	×	✓	74.89
biLSTM	×	✓	✓	×	✓	76.03
GRU	×	✓	✓	×	✓	75.73
Conv	×	✓	✓	×	✓	75.90
–	✓	✓	✓	×	✓	75.57
biLSTM	✓	✓	✓	×	✓	76.14
GRU	✓	✓	✓	×	✓	75.73
Conv	✓	✓	✓	×	✓	76.16
–	×	×	×	✓	✓	75.59
biLSTM	×	×	×	✓	✓	76.04
GRU	×	×	×	✓	✓	75.56
Conv	×	×	×	✓	✓	76.03
–	✓	×	×	✓	✓	75.54
biLSTM	✓	×	×	✓	✓	76.23

Table 1 (continued)

Text processing	$R_{H_v^{\text{final}}}$	R_{sequence}	R_{backbone}	$R_{H_q^{\text{final}}}$	F_{residual}	mAP
GRU	✓	×	×	✓	✓	75.76
Conv	✓	×	×	✓	✓	75.72
–	×	×	✓	✓	✓	75.98
biLSTM	×	×	✓	✓	✓	75.81
GRU	×	×	✓	✓	✓	75.84
Conv	×	×	✓	✓	✓	76.22
–	×	✓	×	✓	✓	73.96
biLSTM	×	✓	×	✓	✓	75.90
GRU	×	✓	×	✓	✓	74.85
Conv	×	✓	×	✓	✓	74.71
–	✓	✓	×	✓	✓	74.35
biLSTM	✓	✓	×	✓	✓	76.01
GRU	✓	✓	×	✓	✓	75.22
Conv	✓	✓	×	✓	✓	75.20
–	✓	×	✓	✓	✓	76.04
biLSTM	✓	×	✓	✓	✓	76.08
GRU	✓	×	✓	✓	✓	75.89
Conv	✓	×	✓	✓	✓	76.50
–	✓	×	✓	✓	✓	74.51
biLSTM	✓	×	✓	✓	✓	75.70
GRU	✓	×	✓	✓	✓	75.45
Conv	✓	×	✓	✓	✓	75.20
–	✓	✓	✓	✓	✓	75.29 (75.53)
biLSTM	✓	✓	✓	✓	✓	76.44 (76.97)
GRU	✓	✓	✓	✓	✓	75.79 (76.60)
Conv	✓	✓	✓	✓	✓	76.19 (76.53)

$\mathbf{R}_{\mathbf{H}_v^{\text{final}}}$ represents the residual from the video-text fusion branch added to the Transformer's output. $\mathbf{R}_{\text{sequence}}$ refers to the residual from the sequence processing layer added to the output of Linear_1. $\mathbf{R}_{\text{backbone}}$ is the residual from the video encoder, also summed with the output of Linear_1. $\mathbf{R}_{\mathbf{H}_q^{\text{final}}}$ represents the residual from the image-text fusion branch added to the Transformer's output, while $\mathbf{F}_{\text{residual}}$ indicates the residual from the image decoder added to the output of Linear_2. The best result is in bold and the results when training using 200 epochs are reported in parentheses

($\mathbf{R}_{\mathbf{H}_v^{\text{final}}}$, $\mathbf{R}_{\text{backbone}}$, $\mathbf{R}_{\mathbf{H}_q^{\text{final}}}$, conv). These models were retrained for 200 epochs, similar to ARTEMIS. BEST_1 achieved the highest mAP, scoring 77.33. Additionally, these three models demonstrated performance equal to or greater than ARTEMIS with all residual connections, indicating that while residual connections contribute to improved performance, not all of them are necessary.

We also explored the effect of different processing layers on video descriptions. To analyze this, results for each type of processing layer were aggregated and visualized in a boxplot (Fig. 2). Further statistical analysis was performed using t-tests [54] and the Wilcoxon signed-rank test [55] (Table 2), which revealed that using no text processing or a GRU layer led to lower performance compared to biLSTM and convolution layers. While no statistically significant difference was found between biLSTM and convolution, the highest performance was consistently achieved with convolutional layers.

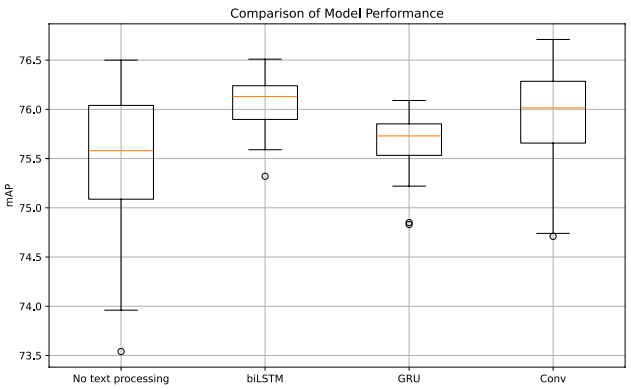

Fig. 2 Boxplot comparison of model performance between different text processing layers during the ablation study

Table 2 Statistical comparison of model performance between different text processing layers during the ablation study

Comparison	*p* value t-test	*p* value Wilcoxon test
No text processing versus biLSTM	0	0.0001
No text processing versus GRU	0.0842	0.1658
No text processing versus Conv	0	0
biLSTM versus GRU	0	0
biLSTM versus Conv	0.0797	0.0739
GRU versus Conv	0.0025	0.0047

The table presents the *p* values obtained from paired t-tests and Wilcoxon signed-rank tests to evaluate the significance of the performance differences between no text processing, biLSTM, GRU, and convolutional (Conv) layers

Table 3 Comparison of mAP scores for ARTEMIS-biLSTM and the best three models from the ablation study using contrastive, cosine, and CCA losses as described in Sect. 3.3

Model	Contrastive mAP	Cosine mAP	CCA mAP
ARTEMIS-biLSTM	69.27	**77.23**	76.59
$BEST_1$	67.07	**77.16**	77.09
$BEST_2$	67.61	**77.20**	76.53
$BEST_3$	67.59	**77.27**	76.25

4.2.2 Feature alignment

Feature alignment was performed on ARTEMIS-biLSTM and the three best models identified in the ablation study. Table 3 presents the mAP scores for these four models using contrastive, cosine, and CCA losses for feature alignment.

Among the alignment strategies, cosine similarity consistently produced the best results across all models. ARTEMIS-biLSTM achieved an mAP of 77.23 with cosine similarity, surpassing its previous score of 76.97. Likewise, $BEST_1$, $BEST_2$, and $BEST_3$ achieved mAP values of 77.16, 77.20, and 77.27, respectively, either maintaining or slightly improving upon their baselines. This consistency demonstrates that cosine similarity effectively aligns multimodal features, enhancing overall model performance without introducing instability.

In contrast, both contrastive and CCA losses resulted in performance degradation across all models. While CCA helped align features, its impact was less beneficial than that of cosine similarity, as models trained for 200 epochs without feature alignment outperformed those using CCA. The contrastive loss showed the most significant drop in performance, likely because the models prioritized feature alignment over task-specific performance, with the contrastive component dominating the overall loss function, thus diminishing the contribution of the primary task loss.

4.2.3 Ensemble

To construct the ensemble, we selected the best configurations achieving the highest mAP: $BEST_1$, $BEST_3$, and ARTEMIS-biLSTM. Both $BEST_3$ and ARTEMIS-biLSTM were trained using cosine similarity loss, which contributed to their strong individual performances.

As shown in Table 4, all ensembling methods applied improved upon the best single-network result. However, the reinforcement learning ensemble underperformed compared to other approaches. This consistent underperformance across different validation folds suggests that the RL agents struggled to find optimal weights for each test sample. This issue is likely due to an insufficient number of samples in the validation set to adequately represent all possible actions, leading to suboptimal results.

On the other hand, the average ensemble and the genetic algorithm-based weighted ensemble demonstrated superior performance. The average ensemble achieved 79.73 mAP without requiring additional training or optimization, while the GA-based weighted ensemble slightly improved this, reaching a best score of 79.82 mAP. The GA-based method's fine-tuning of weights provided a small but meaningful advantage, optimizing the ensemble's performance more effectively than a simple averaging strategy. Interestingly, the GA consistently assigned weights across folds that were similar to the average ensemble, but with slightly more emphasis on $BEST_1$ and $BEST_3$, and less on ARTEMIS-biLSTM.

Overall, these results highlight the potential of ensembling to improve performance, albeit at the cost of increased model complexity. Table 5 compares the best result obtained using the GA on fold 2 with previous research on the Animal Kingdom dataset, demonstrating that our approach surpasses all prior efforts.

Table 6 summarizes the dimensions and inference times of the three best-performing ARTEMIS models. A significant

Table 4 Comparison of model performance results between average ensemble and weighted ensemble strategies using weights derived from genetic algorithms (GA) and reinforcement learning (RL)

Strategy	0	1	2	3	4
Average ensemble	79.73				
Weighted ensemble (GA)	79.81	79.81	79.82	79.79	79.81
Weighted ensemble (RL)	79.17	78.70	78.87	79.24	77.42

The results are based on testing the models on the test set with weights obtained from training on various validation sets, assessed through 5-fold cross-validation. Each column corresponds to a different fold used in the cross-validation process

Table 5 Result comparison between ARTEMIS (single best model and ensemble) and previous researches on recognizing animal action in the Animal Kingdom dataset

Model	mAP
CARe-I3D [11]	16.5
CARe-SlowFast [11]	20.5
CARe-X3D [11]	25.2
MSQNet-VideoMAE [14]	71.2
MSQNet-TimeSformer [14]	73.1
Mamba-MSQNet [35]	74.6
ARTEMIS (single)	77.3
ARTEMIS (ensemble)	**79.8**

Table 6 Number of parameters for ARTEMIS best architectures

Model	Parameters (M)	Inference time (s)
ARTEMIS (best 1)	10,953	0.10
ARTEMIS (best 2)	10,953	0.10
ARTEMIS (best 3)	10,955	0.10
ARTEMIS (ensemble)	11,461	0.30

For BLIP2 and Llama 3 (8B), used in ARTEMIS, we accounted precisely for 2.7 billion and 8 billion parameters, respectively. A single usage of BLIP2, Llama 3 and CLIP-text was considered for obtaining all textual descriptions, which are precomputed prior to model training for efficiency. In the case of the Animal Kingdom dataset this process took around 7 on a NVIDIA A100 GPU. The same GPU was used for computing the inference time, calculating the average inference time over 100 inputs

portion of the model's complexity arises from the video description processing stage. The rest of the architecture comprises 253 million parameters for both $BEST_1$ and $BEST_2$, and 255 million parameters for $BEST_3$, resulting in a total of 761 million parameters for the ensemble model. In terms of inference time, assuming the video descriptions have already been generated, ARTEMIS (single model) requires 0.1 s, while the ensemble model takes 0.3 s. This difference reflects the computational overhead of combining three single models in the ensemble. The primary bottleneck of the approach lies in generating video descriptions. For the Animal Kingdom dataset, which includes 30,100 clips, the process of computing video descriptions was notably time intensive. Captioning all frames required approximately five days, summarizing the extracted information took around three hours, and generating embeddings for all clips took less than a minute using a NVIDIA A100 GPU.

4.3 Video description analysis

The quality of the video descriptions relies heavily on the capabilities of BLIP2 to accurately identify both the animals and their actions in individual frames, and LLaMA's ability to summarize this information into a coherent final description for input into the model. In the majority of cases, the descriptions correctly reflect the events in the video, demonstrating that this approach is effective. However, a few errors do occur, and we categorize these into the following types:

- *Incorrect animal identification* While the captioning model often captures the correct actions, it sometimes struggles with identifying the specific species, leading to multiple or incorrect animal suggestions in the final summary. LLaMA 3-8B occasionally aggregates these uncertainties, resulting in descriptions that mention more animals than are actually present, as in Fig. 3a. In some cases, the model attempts to be overly precise, particularly with bird species, suggesting several species when there is only one, as in Fig. 3b. There are rare instances where the model fails entirely, producing a list of random animals and actions that bear no resemblance to the scene, such as in Fig. 3c, where a mongoose and a black mamba are engaged in a fight, yet the system fails to grasp the scene's dynamics.
- *The "sitting problem"* In many video descriptions (345 occurrences), the verb "sit" is repeated multiple times. Interestingly, this issue arises only in the final video summaries produced by LLaMA 3-8B, not in the initial captions from BLIP2. It seems that LLaMA interprets the repeated mention of this verb across multiple frames as a key feature, leading to overemphasis in the final description.
- *Incorrect action identification* Occasionally, the described actions in the final video summary are incorrect. Examples include Fig. 3c, d, where in the latter case, the action is labeled as "playing", which is inappropriate for a predator–prey situation. However, the remaining actions are usually accurate, making these errors a small noise.
- *Movie or TV references* Rarely (only three instances), the model references TV channels, streaming platforms, or movie characters in the video descriptions. For instance, in Fig. 3e, this likely stems from a visible NETFLIX watermark, while Fig. 3f may have been influenced by the resemblance of the scene to an old Godzilla movie, further confused by a BBC watermark.
- *Focusing on text in the scene* In one case, the captioning model fixated on an inscription visible throughout the video, rather than focusing on the scene itself (Fig. 3g). Although BLIP2 correctly identifies a snake "sitting on top of a rock in the grass", LLaMA's summarization overly emphasizes the text in the scene.
- *Lack of action information* In approximately 60 cases, the video description states that there is no information about

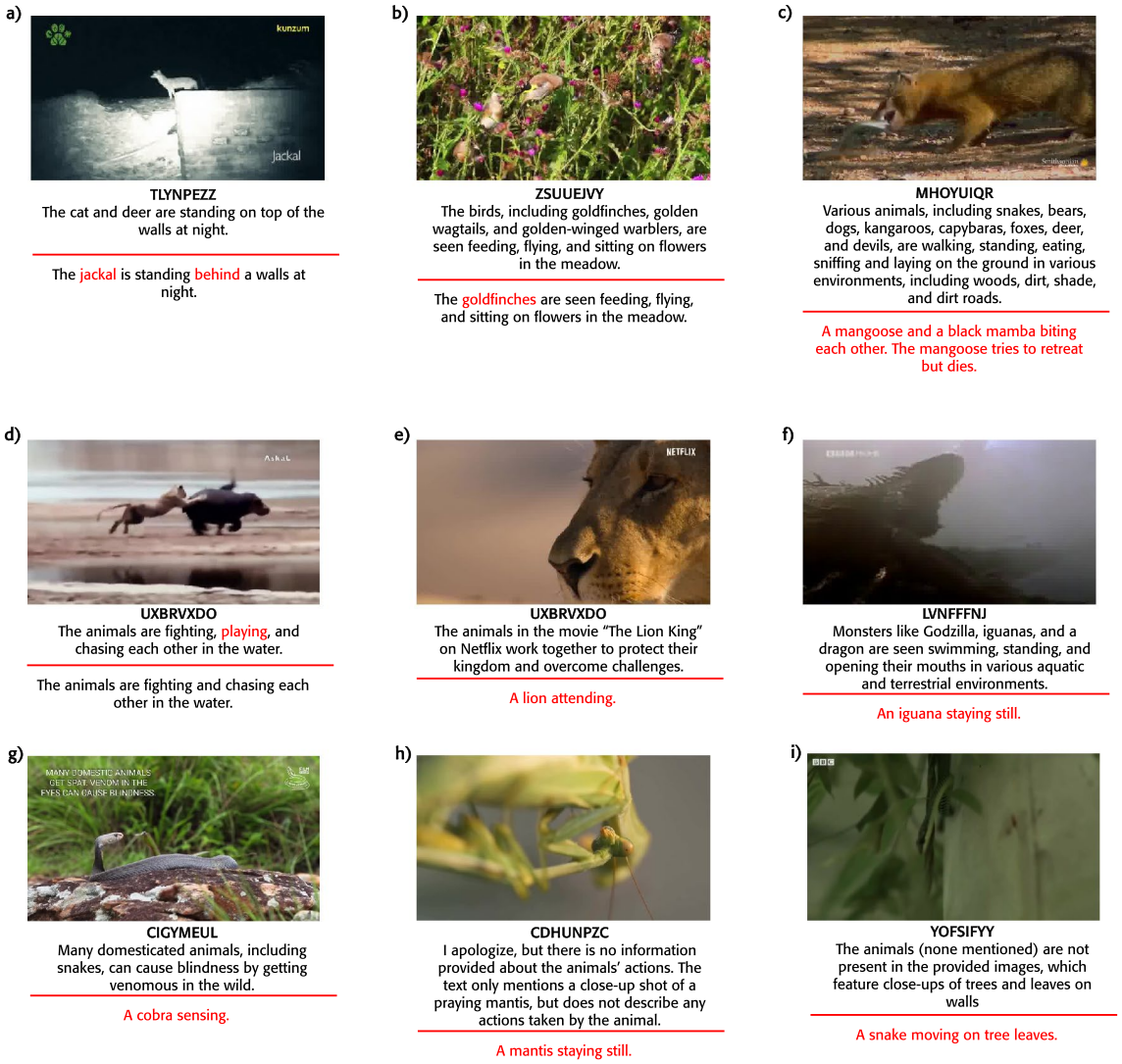

Fig. 3 Examples of incorrect video descriptions. Each example includes the unique identifier of the video and the text generated by Llama 3-8B. Below the red line is a revised version, reflecting the correct information from the Animal Kingdom metadata. Key differences between the two versions are highlighted in red

the animal's actions, despite correctly identifying the species present in the video. This error typically occurs when the captioning model focuses solely on the scene rather than the actions, often producing descriptions like "a close-up of a [animal name]" without any mention of behavior. Figure 3h is an example of this issue, where the description only notes "a close-up of a praying mantis", neglecting to describe any actions.

- *No animal information* In around 30 instances, the video descriptions fail to mention any animals, likely because the captions focus on describing the background instead. This is especially common when the animal blends in with its surroundings due to camouflaging colors, as shown in Fig. 3i.

4.4 Robustness testing

Our research primarily focused on the Animal Kingdom dataset, which, with its extensive range of 140 actions and 850 species, serves as a comprehensive benchmark for animal action recognition tasks. However, to further validate the robustness of our approach, we considered important to evaluate its performance on an additional AAR dataset. For this purpose, we selected the BaboonLand dataset [36], which consists of Unmanned Aerial Vehicles (UAVs) recordings of baboons performing 12 distinct actions (e.g., sitting, walking, drinking, and various forms of grooming) and includes an additional category labeled "Occluded". The "Occluded" category introduces a significant challenge, as it encompasses instances where animals are partially or

Table 7 Mean average precision (mAP) results for the Baboonland dataset

Model	mAP
X3D [36]	63.9
MSQNet [14]	76.5
Mamba-MSQNet [35]	78.9
ARTEMIS (single)	**79.3**

The best result is indicated in bold

entirely obscured, or absent altogether, requiring the model to distinguish between ambiguous and actionable frames effectively. This added complexity highlights the need for models capable of handling diverse and unpredictable scenarios in real-world animal monitoring.

In Table 7, we present the mAP results of previously proposed models compared to ARTEMIS (single). Our model outperformed past architectures, however, its performance is only marginally better than that of Mamba-MSQNet, a less complex model in terms of computational power and parameters. This suggests that the integration of textual information does not result in a substantial performance improvement in this particular context. A potential explanation for this outcome is that ARTEMIS may overfit the training set after only a few epochs (i.e., five), which limits its ability to generalize and learn effectively from the data. We hypothesize that this could be due to the relatively small size of actions in the BaboonLand dataset and the low diversity in action classes, as some actions are variations of grooming behavior. Such a limited scope may reduce the advantages gained from textual descriptions, which are likely to have a more pronounced impact when the dataset contains a larger number of distinct and varied actions, like in Animal Kingdom.

Despite these limitations, we believe that ARTEMIS, due to its more sophisticated architecture, holds greater potential for more complex tasks involving larger and more diverse datasets. The integration of textual descriptions is particularly advantageous in scenarios where multiple, highly varied actions need to be recognized, as it can provide complementary contextual information that enhances the model's learning process.

In line with our approach for the Animal Kingdom dataset, we have made the textual descriptions generated for each video clip in BaboonLand publicly available, along with our code.

5 Conclusions

In this study, we proposed a multimodal fusion model for animal action recognition, named ARTEMIS, which leverages both textual and visual information through two branches: one combining video and text data, and the other fusing image and text embeddings. To evaluate the performance of our model, we used the Animal Kingdom dataset, generating textual descriptions for each clip. These descriptions were obtained by captioning individual frames via BLIP2 and then aggregating the information using Llama 3-8B to create comprehensive narratives focused on animal actions. We have shared these descriptions alongside our code.

ARTEMIS uses the textual descriptions together with video data in the video-text fusion branch, while the image-text fusion branch combines information from images and label embeddings. Additionally, we conducted an ablation study to determine which residual connections were necessary and to identify the best processing layer for handling video descriptions. Furthermore, we explored feature alignment to improve performance and experimented with ensemble models. Feature alignment using cosine similarity loss resulted in a slight performance boost, while contrastive and CCA losses negatively impacted results. For ensembling, we combined the three best models and used averaging, genetic algorithms, and reinforcement learning to find both static and dynamic weights. The genetic algorithm-based approach yielded the best performance, achieving a score of 79.82 mAP on the test set, surpassing previous results on the Animal Kingdom dataset.

While our approach demonstrates significant improvements in performance compared to prior works, there remains considerable room for further improvement. Despite its strengths, the architecture is not without limitations. The integration of BLIP2 and Llama 3 introduces substantial complexity, demanding considerable computational resources. Another potential drawback lies in the generation of video descriptions, which relies on single frames without leveraging spatiotemporal information; an approach that may limit performance for actions requiring contextual understanding over time.

Future work could explore better backbone architectures, alternative fusion mechanisms, or new combinations that could further enhance the model's effectiveness. Additionally, improving the quality of the video descriptions, potentially generating descriptions that more explicitly reference the actions in the labels, could be a crucial factor in achieving higher accuracy.

Acknowledgements This work was partially carried out in the framework of the H2020 FETOPEN Project "Robocoenosis-ROBOts in cooperation with a bioCOENOSIS" [899520]. The founder had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Funding Open access funding provided by Scuola Superiore Sant'Anna within the CRUI-CARE Agreement.

References

1. Heyes C, Dickinson A (1990) The intentionality of animal action. Mind Lang 5(1):87–103
2. Rao Y, Jiang M, Wang W, Zhang W, Wang R (2020) On-farm welfare monitoring system for goats based on internet of things and machine learning. Int J Distrib Sens Netw 16(7):786–985
3. Kostarev S, Sereda T, Tatarnikova N (2020) Building a model for recognition of morphostructure pathologies in animal tissues. J Phys Conf Ser 1515:052005
4. Thenmozhi M, Saravanan M, Kumar KPM, Suseela S, Deepan S (2020) Improving the prediction rate of unusual behaviors of animal in a poultry using deep learning technique. Soft Comput 24:14491–14502
5. Jiang M, Rao Y, Zhang J, Shen Y (2020) Automatic behavior recognition of group-housed goats using deep learning. Comput Electron Agric 177:105706
6. Feng L, Zhao Y, Sun Y, Zhao W, Tang J (2021) Action recognition using a spatial-temporal network for wild felines. Animals 11(2):485
7. Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M (2018) DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat Neurosci 21(9):1281–1289
8. Segalin C, Williams J, Karigo T, Hui M, Zelikowsky M, Sun JJ, Perona P, Anderson DJ, Kennedy A (2021) The mouse action recognition system (MARS) software pipeline for automated analysis of social behaviors in mice. Elife 10:63720
9. Labuguen R, Matsumoto J, Negrete SB, Nishimaru H, Nishijo H, Takada M, Go Y, Inoue K-I, Shibata T (2021) Macaquepose: a novel "in the wild'' macaque monkey pose dataset for markerless motion capture. Front Behav Neurosci 14:581154
10. Yu H, Xu Y, Zhang J, Zhao W, Guan Z, Tao D (2021) Ap-10k. A benchmark for animal pose estimation in the wild. arXiv preprint arXiv:2108.12617
11. Ng XL, Ong KE, Zheng Q, Ni Y, Yeo SY, Liu J (2022) Animal kingdom: a large and diverse dataset for animal behavior understanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 19023–19034
12. Terrasson G, Llaria A, Marra A, Voaden S (2016) Accelerometer based solution for precision livestock farming: geolocation enhancement and animal activity identification. IOP Conf Ser Mater Sci Eng 138:012004
13. O'Shea K (2015) An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458
14. Mondal A, Nag S, Prada JM, Zhu X, Dutta A (2023) Actor-agnostic multi-label action recognition with multi-modal query. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 784–794
15. Soares S, Antunes C, Araújo R (2012) A genetic algorithm for designing neural network ensembles. In: Proceedings of the 14th annual conference on genetic and evolutionary computation, pp 681–688
16. Sivanandam S, Deepa S, Sivanandam S, Deepa S (2008) Genetic algorithm optimization problems. In: Introduction to genetic algorithms, pp 165–209
17. Mou L, Saha S, Hua Y, Bovolo F, Bruzzone L, Zhu XX (2021) Deep reinforcement learning for band selection in hyperspectral image classification. IEEE Trans Geosci Remote Sens 60:1–14
18. Fazzari E, Loughlin HA, Stoughton C (2024) Controlling optical-cavity locking using reinforcement learning. Mach Learn Sci Technol 5(3):035027
19. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
20. Sun Z, Ke Q, Rahmani H, Bennamoun M, Wang G, Liu J (2022) Human action recognition from various data modalities: a review. IEEE Trans Pattern Anal Mach Intell 45(3):3200–3225
21. Fazzari E, Romano D, Falchi F, Stefanini C (2024) Animal behavior analysis methods using deep learning: a survey. arXiv:2405.14002
22. Pereira TD, Tabris N, Matsliah A, Turner DM, Li J, Ravindranath S, Papadoyannis ES, Normand E, Deutsch DS, Wang ZY (2022) SLEAP: a deep learning system for multi-animal pose tracking. Nat Methods 19(4):486–495
23. Bohnslav JP, Wimalasena NK, Clausing KJ, Dai YY, Yarmolinsky DA, Cruz T, Kashlan AD, Chiappe ME, Orefice LL, Woolf CJ (2021) DeepEthogram, a machine learning pipeline for supervised behavior classification from raw pixels. Elife 10:63377
24. Odo A, Muns R, Boyle L, Kyriazakis I (2023) Video analysis using deep learning for automated quantification of ear biting in pigs. IEEE Access 11:59744–59757
25. Blau A, Gebhardt C, Bendesky A, Paninski L, Wu A (2022) Semimultipose: a semi-supervised multi-animal pose estimation framework. arXiv preprint arXiv:2204.07072
26. Fazzari E, Carrara F, Falchi F, Stefanini C, Romano D (2024) Using AI to decode the behavioral responses of an insect to chemical stimuli: towards machine-animal computational technologies. Int J Mach Learn Cybern 15(5):1985–1994
27. Fujimori S, Ishikawa T, Watanabe H (2020) Animal behavior classification using deeplabcut. In: 2020 IEEE 9th global conference on consumer electronics (GCCE). IEEE, pp 254–257
28. Arablouei R, Wang L, Currie L, Yates J, Alvarenga FA, Bishop-Hurley GJ (2023) Animal behavior classification via deep learning on embedded systems. Comput Electron Agric 207:107707
29. Arablouei R, Wang Z, Bishop-Hurley GJ, Liu J (2023) Multimodal sensor data fusion for in-situ classification of animal behavior using accelerometry and GNSS data. Smart Agric Technol 4:100163
30. Zhao L, Stephany RG, Han Y, Ahmmed P, Huang T-P, Bozkurt A, Jia Y (2024) A wireless multimodal physiological monitoring ASIC for animal health monitoring injectable devices. IEEE Trans Biomed Circuits Syst
31. Anderson G, Johnson A, Arguelles-Ramos M, Ali A (2023) Impact of body-worn sensors on broiler chicken behavior and agonistic interactions. J Appl Anim Welf Sci 25:1–10

32. Zhao D, Chang Z, Guo S (2019) A multimodal fusion approach for image captioning. Neurocomputing 329:476–485
33. Gong X, Mohan S, Dhingra N, Bazin J-C, Li Y, Wang Z, Ranjan R (2023) MMG-EGO4D: multimodal generalization in egocentric action recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6481–6491
34. Xia W, Yang Y, Xue J-H, Wu B (2021) Tedigan: text-guided diverse face image generation and manipulation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2256–2265
35. Fazzari E, Romano D, Falchi F, Stefanini C (2024) Selective state models are what you need for animal action recognition. Ecol Inf 25:102955
36. Duporge I, Kholiavchenko M, Harel R, Wolf S, Rubenstein D, Crofoot M, Berger-Wolf T, Lee S, Barreau J, Kline J et al (2024) BaboonLand dataset: tracking primates in the wild and automating behaviour recognition from drone videos. arXiv preprint arXiv:2405.17698
37. Chen J, Hu M, Coker DJ, Berumen ML, Costelloe B, Beery S, Rohrbach A, Elhoseiny M (2023) Mammalnet: a large-scale video benchmark for mammal recognition and behavior understanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13052–13061
38. Bertasius G, Wang H, Torresani L (2021) Is space-time attention all you need for video understanding? In: ICML, vol 2, p 4
39. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J (2021) Learning transferable visual models from natural language supervision. In: International conference on machine learning. PMLR, pp 8748–8763
40. Vaswani A (2017) Attention is all you need. Adv Neural Inf Process Syst
41. Li J, Li D, Savarese S, Hoi S (2023) Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In: International conference on machine learning. PMLR, pp 19730–19742
42. Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, Mathur A, Schelten A, Yang A, Fan A et al (2024) The llama 3 herd of models. arXiv preprint arXiv:2407.21783
43. Akbari H, Yuan L, Qian R, Chuang W-H, Chang S-F, Cui Y, Gong B (2021) VATT: transformers for multimodal self-supervised learning from raw video, audio and text. Adv Neural Inf Process Syst 34:24206–24221
44. Kipf T, Pol E, Welling M (2019) Contrastive learning of structured world models. In: International conference on learning representations
45. Wang H, Wang Y, Zhou Z, Ji X, Gong D, Zhou J, Li Z, Liu W (2018) COSFACE: large margin cosine loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5265–5274
46. Sun Q-S, Zeng S-G, Liu Y, Heng P-A, Xia D-S (2005) A new method of feature fusion and its application in image recognition. Pattern Recognit 38(12):2437–2448
47. Zhiyuan L (2024) Investigating an ensemble classifier based on multi-objective genetic algorithm for machine learning applications. Int J Adv Comput Sci Appl 15(5):756
48. Rudnick EM, Patel JH, Greenstein GS, Niermann TM (1997) A genetic algorithm framework for test generation. IEEE Trans Comput Aided Des Integr Circuits Syst 16(9):1034–1044
49. Takahashi M, Kita H (2001) A crossover operator using independent component analysis for real-coded genetic algorithms. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01th8546). IEEE, vol 1, pp 643–649
50. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
51. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N (2021) Stable-baselines3: reliable reinforcement learning implementations. J Mach Learn Res 22(268):1–8
52. Kingma DP (2014) ADAM: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
53. Loshchilov I, Hutter F (2017) SGDR: stochastic gradient descent with warm restarts. In: International conference on learning representations. https://openreview.net/forum?id=Skq89Scxx
54. Student (1908) The probable error of a mean. Biometrika 1–25
55. Wilcoxon F (1992) Individual comparisons by ranking methods. In: Breakthroughs in statistics: methodology and distribution. Springer, New York, pp 196–202