Tiger/Animal Condition Detection Project – Binary Classification

# Table of Contents

1. Introduction

2. Problem Statement

3. Objectives

4. Literature Review

5. Dataset Preparation

6. Preprocessing Steps

7. Model Architecture

8. Training & Evaluation

9. Results & Analysis

10. Website Integration & Deployment

11. Conclusion

# 1. Introduction

Monitoring the health and welfare of tigers is critical for conservation and timely intervention.

This project leverages deep learning to automatically classify animal behavior as Normal or Abnormal from video data,

aiding caretakers in zoos and wildlife settings to detect early signs of health issues or distress.

# 2. Problem Statement

Manual monitoring of animal behavior is time-consuming and prone to error.

There is a lack of automated systems that analyze animal videos in real-time and accurately distinguish healthy behavior

from signs of abnormalities.

# 3. Objectives

* - Develop a deep learning model for binary classification of tiger behavior from video clips.
* - Incorporate 3D pose keypoint information to improve classification accuracy.
* - Deploy the model through a Flask web app for accessible real-time or batch inference.
* - Create a flexible system extensible to other animal species.

# 4. Literature Review

Transformer-based video models like TimeSformer and pose-based transformers have shown strong

spatiotemporal and skeletal dynamics understanding. Combining raw visual features from videos and pose keypoints enables

more accurate behavior classification. 3D pose estimation techniques help capture posture and movement intricacies.

# 5. Dataset Preparation

* - Videos labeled as Tiger\_Normal or Tiger\_Abnormal.
* - Frames extracted uniformly, resized to 224×224 (or 256×256) for input consistency.
* - 3D pose keypoints extracted or estimated for all clips, representing 34 keypoints with (x, y, z) coordinates.
* - Prepared PyTorch datasets loading video frames and corresponding pose tensors.

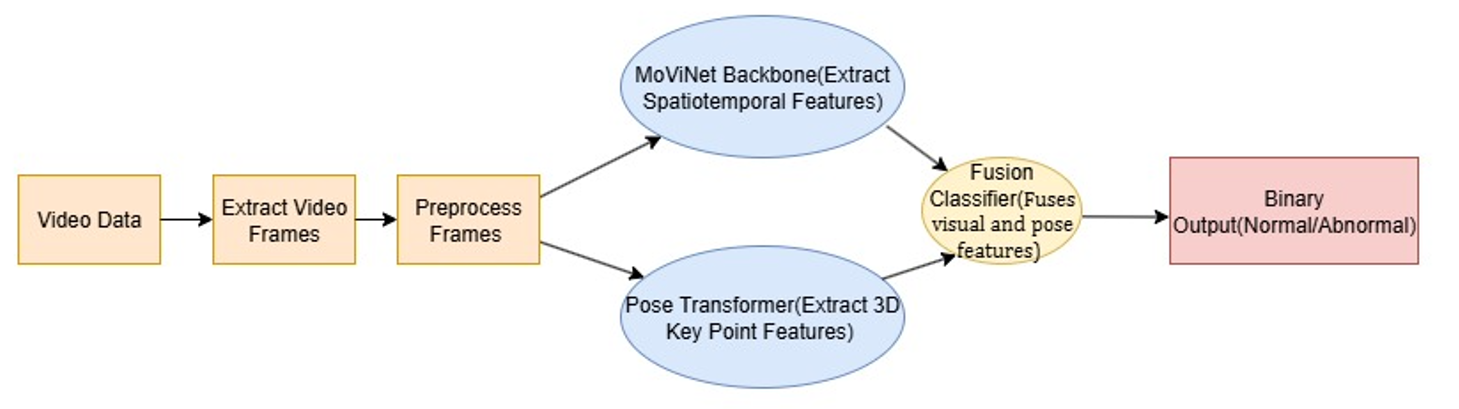
# 6. Preprocessing Steps

* - Uniform frame sampling from videos (e.g., 8-frame clips).
* - Frame resizing and RGB normalization.
* - Pose tensor preparation aligned with video frames.
* - Optional 3D pose dynamic estimation or loading from precomputed .npy files.

# 7. Model Architecture

The architecture consists of:

* - MoViNet backbone: Extracts spatiotemporal features from video frames.
* - Pose Transformer: Learns representation from 3D pose keypoint sequences.
* - Fusion Classifier: Fuses visual and pose features for binary Normal/Abnormal classification.
* - TigerClassifier: Integrates all modules into an end-to-end network.



# 8. Training & Evaluation

* - Optimizer: Adam with a learning rate of 1e-4.
* - Loss: CrossEntropyLoss for binary classification.
* - Metrics: Binary accuracy computed with torchmetrics.
* - Batch size, epochs, and training details adapted for hardware constraints.
* - Validation periodically runs to monitor and save the best performing model checkpoint.

# 9. Results & Analysis

* - The combined video + pose model outperformed models using video or pose alone.
* - Binary classification achieved high accuracy, enabling robust health monitoring.
* - Model demonstrated sensitivity to subtle abnormal behaviors.

# 10. Website Integration & Deployment

* - Flask-based REST API built to upload videos (or images) and receive classification results.
* - Video preprocessing pipeline integrated with frame extraction and dummy (or real) 3D pose estimation.
* - Model inference performed on GPU or CPU depending on deployment.
* - JSON responses include predicted label ("Normal" or "Abnormal") and abnormal subclass if applicable.

# 11. Conclusion

This project successfully built an automated system for detecting abnormal conditions in tigers using video and pose data.

The hybrid transformer-based architecture facilitated accurate behavior classification. The Flask app provides an accessible interface for caretakers.

The framework is extendable to other species and clinical applications in wildlife health monitoring.