NALLA PERUMAL M

RA1911033010012

nm7824@srmist.edu.in

NYX WOLVES DEEP LEARNING TASK

1. **Github Repo of the code**

https://github.com/NallaPerumalVIKKI/NYX-wolves-Deep-Learning-Task

2. **Result video :**

https://drive.google.com/file/d/10   7MUsqSj3xYnjxx-s6hPEFHcYolYWe/view?usp=sharing

3. **README file with proper instructions to recreate the project**

https://github.com/NallaPerumalVIKKI/NYX-wolves-Deep-Learning-Task

4. Explanation Documentation containing

A. Model used: MediaPipe pose model

- The primary model used is the MediaPipe Pose model, which is part of the MediaPipe framework. This model is responsible for estimating human poses in each frame of the input video.

- I have defined a set of action classes based on the presence of specific landmarks. These action classes are used for action recognition.

- This is how the model works within the code

  - **Initialization**: The model is initialized with the following parameters:
    min_detection_confidence: This parameter sets the minimum confidence score (ranging from 0 to 1) required for a pose detection to be considered valid. In the code, it's set to 0.5, meaning that only poses with confidence scores of 0.5 or higher are accepted.

    min_tracking_confidence: This parameter sets the minimum confidence score required for a pose detection to be considered as part of a continuous tracking process. In the code, it's also set to 0.5.

  - **Processing Frames**: The video frames captured from a video file are processed using this Pose model. Each frame is converted from the default BGR color format to RGB format because the MediaPipe model expects RGB input.

  - **Pose Estimation**: The MediaPipe Pose model processes each frame and estimates the locations of various key landmarks on the human body. These landmarks include points on the shoulders, hips, knees, wrists, and other body parts.

  - **Action Recognition**: After obtaining the pose landmarks, the code defines a set of custom action classes. Each action class is associated with specific landmarks.

- **Visualization**: The recognized action (or "Undefined" if no action is detected) is annotated on the video frame using OpenCV's drawing functions.

B. Framework used

1. OpenCV (Open-Source Computer Vision Library):

Explanation:

- OpenCV is a popular open-source computer vision and image processing library.
- It provides a wide range of tools and functions for various computer vision tasks, including video capture, frame manipulation, image processing, and video writing.

Usage in the Code:

- **Video Capture**: OpenCV is used to initialize video capture from a video file.It helps read frames from the input video.
- **Frame Manipulation**: OpenCV is used to convert frames from BGR to RGB color format (required by MediaPipe) and for drawing bounding boxes, key points, and text on the frames.
- **Video Writing:** OpenCV is used to create an output video file and write the processed frames into it.

2. MediaPipe:

Explanation:

- MediaPipe is an open-source framework developed by Google for various multimedia and machine-learning tasks.
- Media Pipe’s Pose model is specifically designed for human pose estimation.

Usage in the Code:

- **Pose Estimation**: The core functionality of the code relies on MediaPipe's Pose model. It is used to estimate human poses by detecting and tracking key landmarks on the human body.
- **Pose Landmarks**: The code utilizes MediaPipe's pose landmarks, which are the output of the Pose model. These landmarks represent key points on the human body (e.g., shoulders, hips, knees, wrists).

- **Action Recognition**: The code uses MediaPipe's Pose Landmark to define action classes based on specific landmarks. This information is used for action recognition.

3. **Numpy**

Explanation:

NumPy is a fundamental package for scientific computing with Python. It provides support for working with arrays and matrices. In this code, NumPy is used for array operations, especially when processing pose key points.

Usage in the code:

NumPy is used for array operations in bounding box and processing pose keypoints.

C. Datasets on which the models are trained

For action recognition datasets, I explored publicly available datasets like:

- **UCF101**: The UCF101 dataset contains 101 action categories, and it is widely used for action recognition research.
- **HMDB51**: The HMDB51 dataset includes 51 action classes, and it is another popular dataset for action recognition.
- **Kinetics:** The Kinetics dataset is a large-scale dataset containing human action videos spanning a wide range of categories.

MediaPipe, as a framework, is designed to be highly effective without requiring users to train models on custom datasets for the following reasons:

1. **Pre-trained Models**: MediaPipe provides a range of pre-trained models for various computer vision and machine learning tasks. These models have already been trained on large and diverse datasets by Google's research teams. They have learned to recognize a wide range of objects, gestures, and poses. Users can directly utilize these pre-trained models without needing to collect or train on their datasets.

2. **Transfer Learning**: Many of MediaPipe's models are based on deep learning architectures, which have been pre-trained on massive datasets for tasks like image

recognition. Transfer learning is a technique that allows these pre-trained models to be fine-tuned for specific tasks or domains using relatively small amounts of task-specific data. This makes it easier to adapt the models to new applications without retraining from scratch.

D. Why you have used the above

1. MediaPipe provides a range of pre-trained models for various tasks, including pose estimation, object detection, hand tracking, and more. These models are ready to use "out of the box," saving you the time and resources required for model training.

2. **Data Augmentation**: MediaPipe models often incorporate data augmentation techniques during training, enhancing their ability to handle diverse scenarios and data variations.

3. **Abstraction of Complexity**: MediaPipe abstracts away the complexities of training and managing deep learning models. Developers can focus on using pre-trained models for inference, saving time and effort.

4. Many MediaPipe models are based on deep learning architectures and can be fine-tuned for specific tasks with relatively small amounts of data. This allows for customization without the need for extensive training.

5. **Large Community and Resources**: MediaPipe has a growing community of developers and users, providing access to tutorials, documentation, and resources for support and collaboration.

6. **Broad Applicability:** MediaPipe's pre-trained models can be used across various domains and applications, including image and video analysis, gesture recognition, facial tracking, and more.

7. **Integration with Google Services:** MediaPipe can be integrated with other Google services and libraries, such as TensorFlow and TensorFlow Lite, expanding its capabilities and integration options.

8. **Scalability:** MediaPipe is suitable for both small-scale and large-scale applications, making it adaptable to a wide range of project sizes and complexities.

E. References to GitHub repos and Research Papers

[1] https://www.ijraset.com/research-paper/body-pose-estimation-using-deep-learning

[2] https://ieeexplore.ieee.org/document/9696513

[3] https://ieeexplore.ieee.org/document/10075445?denied=

[4] https://github.com/dronefreak/human-action-classification

[5] https://github.com/yeemachine/kalidokit

[6] https://github.com/Pradnya1208/Squats-angle-detection-using-OpenCV-and-mediapipe_v1

[7] https://www.mdpi.com/2076-3417/13/4/2700