Jukestapose - Playing Music By Detecting Human Poses

Ryan Rosich, Rahul Kulkarni, Catherine Cao
W251 Fall 2019

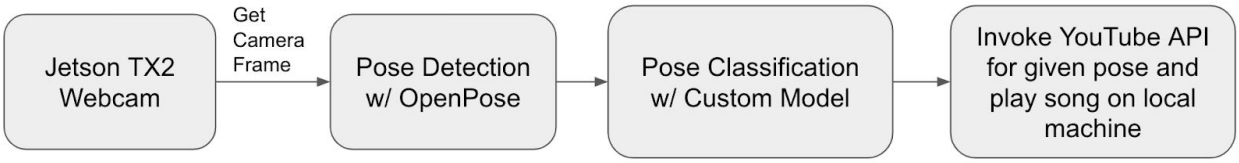
Overview

The objective of this project is to build a robot called Jukestapose, which utilizes neural network that detects human poses in an edge device then plays a corresponding song from Youtube according to the pose.

The neural network model in Jukestapose is built upon an existing deep learning system called OpenPose. The OpenPose model detects human poses in the camera, then a custom classification model is built on top of that to classify those detected poses.

Once a specific human pose is detected, Jukestapose will open a link on YouTube according to the pose and play the audio of that video in local media player.

System Architecture



The edge device used in this project is Jetson TX2 and a webcam. The Jetson TX2 will be constantly capturing through the camera. Each camera frame is obtained and sent through the OpenPose model for pose detection. Once a pose is detected, it will be sent through another custom model for pose classification. The custom model is a classification model with three categories: a dab pose, a t-pose, and all other. Finally, if a dab pose or a t-pose is detected, the sound of a YouTube video that contains the given pose will be played on local media player.

Installation

The first step is to install OpenPose on Jetson TX2. The dependencies required for OpenPose are:

- Python 3.6x
- Tensorflow 1.4.1+
- opencv3, protobuf, python3-tk

- Slidingwindow

We followed the installation as described in here: https://github.com/ildoonet/tf-pose-estimation. Each of us encountered different issues along the way. For example, one of us needed to uninstall and rebuild OpenCV, then go back and make tf-pose-estimation. There was also an issue with pip system wrapper being an older version and incompatible with user pip. Uninstalling and reinstalling pip solved the issue. Other problems include previously installed packages conflicting with the install, and a “sudo apt autoremove” solved the problem. On average, it took us 8-10 hours to have OpenPose models up and running on Jetson TX2.

OpenPose Model

OpenPose is a real-time system that detects “human body, hand, facial, and foot keypoints (in total 135 keypoints) on single images” (OpenPose). For our purpose we primarily focus on human joints for pose detection.

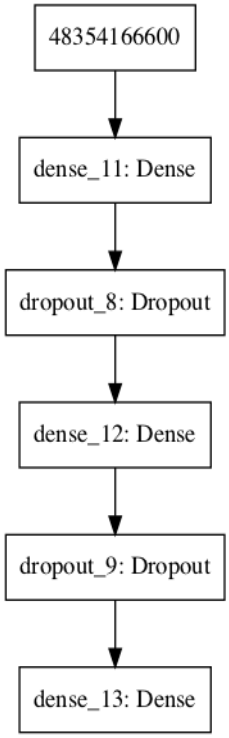
The OpenPose model for pose detection outputs 25 body/foot keypoints in 3 dimensions, X for the position, Y for the position, and confidence for the position. We run OpenPose on Jeston TX2 to capture a series of poses and save the corresponding outputs into numpy arrays. The outputs of the captured poses (with dimensions 25x3) become data for custom classification model that further classifies poses.

Custom Model

The custom model is a classification model built on data captured by OpenPose. The first two positional values are used and the confidence value is ignored. The X and Y values are each divided by 720 and 1280 to convert to a number between 0 and 1. Each input has a dimension of (25, 2) then flattened to (50,).

The model consists of two dense layers of 128 outputs, each followed by a dropout layer, and a softmax layer in the end. Each dense layer uses ReLU activation. The last layer is a softmax layer that identifies the type of the pose.

The model is trained on 171 captured poses using OpenPose. The final model is trained with 2000 epochs with batch size 25.



Pose Detection and YouTube Integration

Once the model is trained, we can use it to detect poses. We use OpenCV to capture each frame of the camera from Jetson, run OpenPose model to obtain the 25x3 positional and confidence values for each captured pose. We convert the 25x3 data to the proper input into the custom model. We then use the custom model to run prediction on those poses.

If a dab pose or a t-pose is predicted, we create a video instance from a pre-defined YouTube url using pafy. We get the audio of the YouTube video, then play it on local VLC player. Currently the link to the YouTube video is statically coded for the dab and the t-pose. In the future, this can be made dynamic from a pool of Youtube videos.

Challenges and Future Improvements

The most difficult part of this project is to get everything set up correctly, including installing OpenPose and various packages. We had to hack our way through resolving dependency and installation failures. In class all requirements are packaged into a container with pre-defined docker image. We learned to appreciate having a clean environment and the effort put into creating that.

The current model classifies two specific poses, the dab and the t-pose. We would like to develop a model that classifies more poses. To do this, we will label more poses in the captured data, and replace the softmax layer with more of the labeled poses. Having more poses detectable would also enable us to do multi-frame pose detection with more interesting movements.

The current model classifies poses on a single frame. For future improvements, we would like to be able to detect a series of movement. This requires keeping a queue of detected poses, and checking against pre-defined movements (list of poses).

We could also make Jukestapose a general purpose JukeBox using a Cloud Music Plugin model that can interface with online music portals. To do this, we need to be able to detect genre of dance moves (80s, hip-hop, jazz, ballet/classical). We need to keep an inventory of dance genre and its corresponding music genre. Finally we would map the dance moves randomly to its corresponding music. The most challenging part of this would be to detect dance moves, as we can no longer rely on single frame poses and we will have to create our own datasets with series of dance moves to build the model.

Finally, we should package all installation and system dependencies into a container with a docker image, so that the app is more easily deployable in the future.

References

https://github.com/CMU-Perceptual-Computing-Lab/openpose

https://github.com/CMU-Perceptual-Computing-Lab

https://github.com/ildoonet/tf-pose-estimation

https://github.com/burningion/dab-and-tpose-controlled-lights