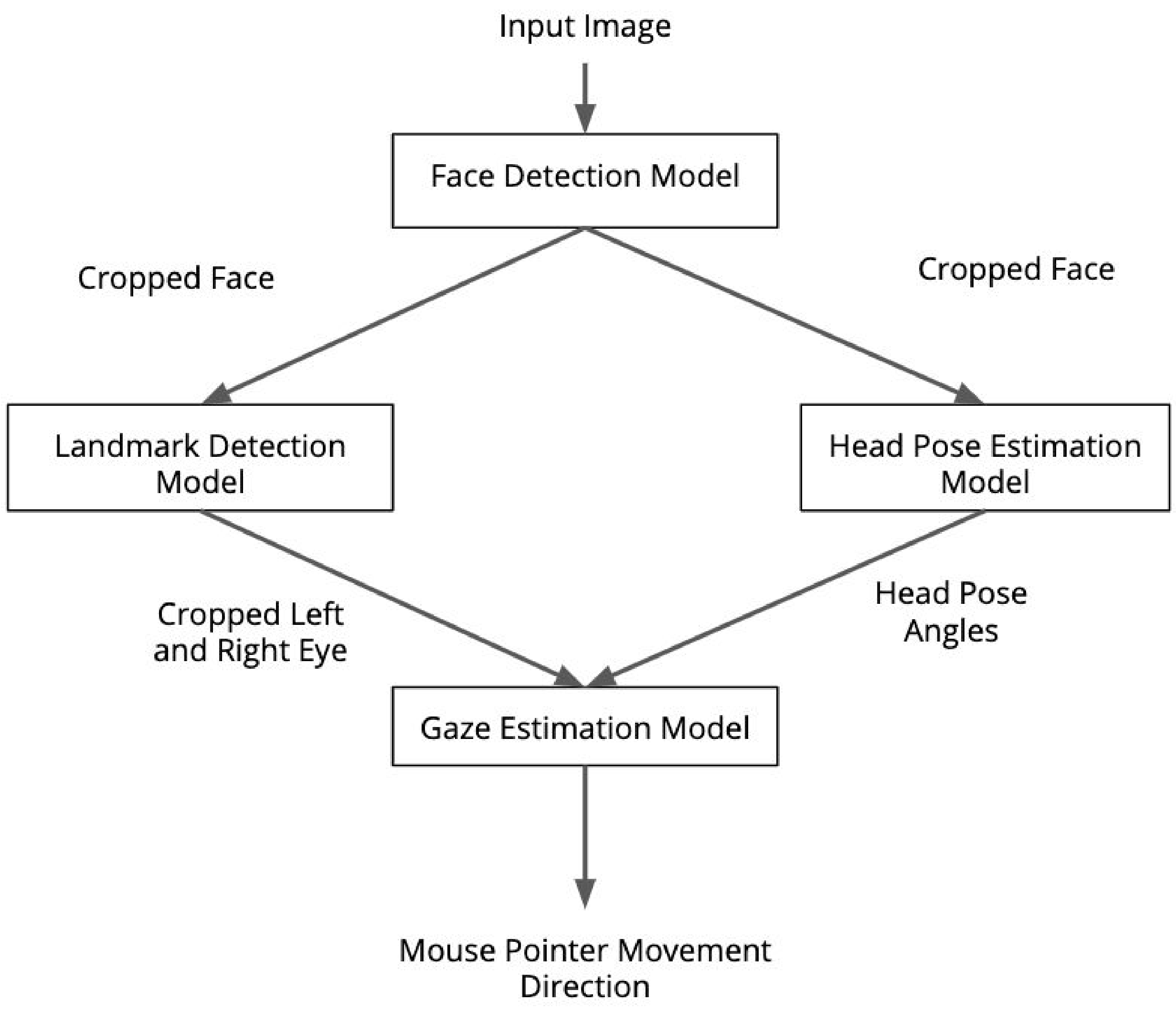
# Computer Pointer Controller

Project is to control the mouse pointer of your computer using a gaze estimation model. It is an application that uses a gaze detection model to control the mouse pointer using an input video or a live stream from your webcam.

## *Pipeline*



## Project Set Up and Installation

- Follow the guidelines to install the openVino
- Clone this repo and just follow the following steps.

### Setup Virtual Environment

Create a virtual environment using the following command in the root directory of the project.

```
python3 -m venv ./venv
```

Then activate the virtual environment with the command:

```
source venv/bin/activate
```

Install the basic project requirements by running the command:

```
pip install -r Requirements.txt
```

### Intsalling pre-trained models

```
cd C:\Program Files (x86)\IntelSWTools\openvino_2020.1.033\bin\
```

```
setupvars.bat
```

Download the following models by using openVINO model downloader:-

1. Face Detection Model(Face Detection Model)

```
python
/opt/intel/openvino_2020.1.033/deployment_tools/tools/model_downloader/download
er.py --name "face-detection-adas-binary-0001"
```

2. Facial Landmarks Detection Model(Facial Landmarks Detection Model)

```
python
/opt/intel/openvino_2020.1.033/deployment_tools/tools/model_downloader/download
er.py --name "landmarks-regression-retail-0009"
```

3. Head Pose Estimation Model(Head Pose Estimation Model)

```
python
/opt/intel/openvino_2020.1.033/deployment_tools/tools/model_downloader/download
er.py --name "head-pose-estimation-adas-0001"
```

4. Gaze Estimation Model(Gaze Estimation Model)

```
python
/opt/intel/openvino_2020.1.033/deployment_tools/tools/model_downloader/download
er.py --name "gaze-estimation-adas-0002"
```

Running Demo

- Go to project directory, then src.
- Run the following Command

Run the main.py file

```
python main.py -fdm <Path of xml file of face detection model> \

-flm <Path of xml file of facial landmarks detection model> \

-hpm <Path of xml file of head pose estimation model> \

-gem <Path of xml file of gaze estimation model> \

-i <Path of input video file or enter cam for taking input video from
webcam>
```

## Command Line Arguments for Running the app

Following are command line arguments that can use for while running the main.py file

`python main.py`:-

- -h : Get the information about all the command line arguments
- -fdm (required) : Specify the path of Face Detection model's xml file
- -flm(required): Specify the path of Face’s Landmarks model’s xml file
- -hpm (required) : Specify the path of Head Pose Estimation model's xml file
- -gem (required) : Specify the path of Gaze Estimation model's xml file
- -i (required) : Specify the path of input video file or enter cam for taking input video from webcam**(for cam use “cam”**).
- -d (optional) : Specify the target device to infer the video file on the model. Suppoerted devices are: CPU, GPU, FPGA (For running on FPGA used HETERO:FPGA,CPU), MYRIAD.
- -l (optional) : Specify the absolute path of cpu extension if some layers of models are not supported on the device.
- -pt (optional) : Specify the probability threshold for face detection model to detect the face accurately from video frame.
- -flag (optional) : Specify the flags from fd, fld, hp, ge if you want to visualize the output of corresponding models of each frame

  Ex - `-flag flm fdm gem`

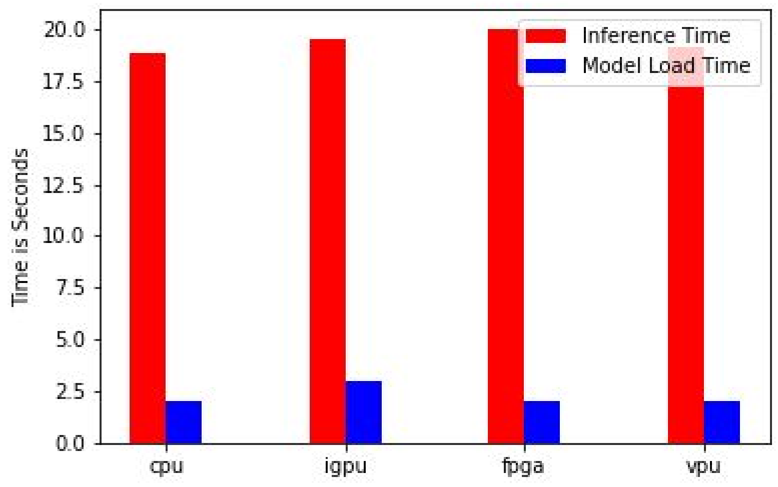
## Directory Structure

- src folder contains all the source files:-
  - face_detection.py
    - Contains preprocession of video frame, perform infernce on it and detect the face, postprocess the outputs.
  - facial_landmarks_detection.py
    - Take the deteted face as input, preprocessed it, perform inference on it and detect the eye landmarks, postprocess the outputs.
  - head_pose_estimation.py

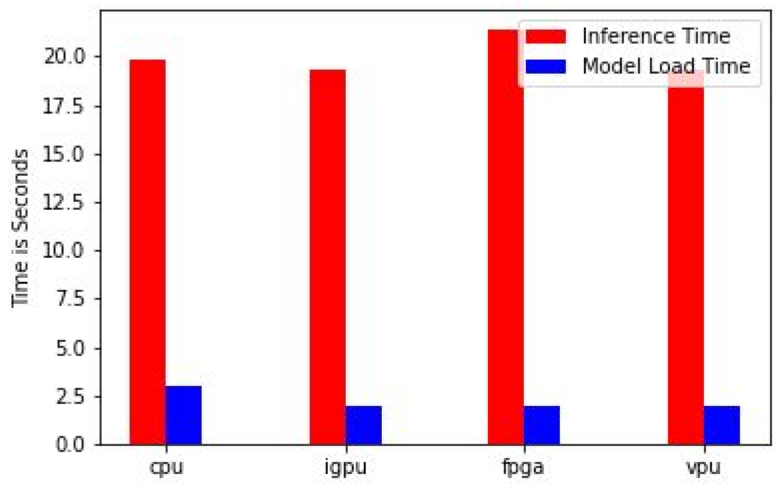
        - Take the detected face as input, preprocessed it, perform inference on it and detect the head postion by predicting yaw - roll - pitch angles, postprocess the outputs.
    - gaze_estimation.py
        - Take the left eye, rigt eye, head pose angles as inputs, preprocessed it, perform inference and predict the gaze vector, postprocess the outputs.
    - input_feeder.py
        - Contains InputFeeder class which initialize VideoCapture as per the user argument and return the frames one by one.
    - mouse_controller.py
        - Contains MouseController class which take x, y coordinates value, speed, precisions and according these values it moves the mouse pointer by using pyautogui library.
    - main.py
        - Users need to run main.py file for running the app.
- bin folder contains demo video which user can use for testing the app.
- README file for instructions
- Mouse file for command to run test
- Requirements file

## Benchmarks

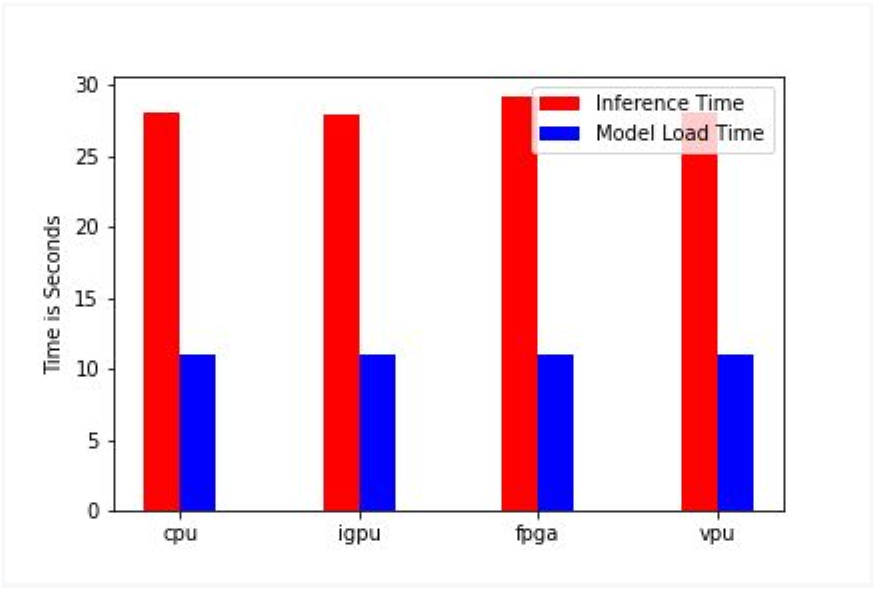
**Results with Model Precision-FP32:**



## Results with Model Precision-FP16:



## Results with Model Precision-FP32-INT8:



## Conclusions

Accuracy of the model decreases with decrease in precision.

As clear from the above results, total inference time and model load time is almost equal for all the devices. While running an application with model precision = FP32, it took a slight more time as it would require more computation.

However, FPGA takes higher inference time because it works on each gate and programmed it to be compatible for this application.