

Fig 1 : Flow diagram for solution design

Hardware : Camera, Laptop

Software : Camera SDK (Nvidia Gstreamer), Azure Services (IOT Hub,IOT Edge)

Machine Learning & Data: Trained Detection model (Face Detection + Person Detection), Database (Whitelist & Blacklist)

## Camera

* Selection of camera based on requirement is very important. We should consider following things which selecting and setting up camera:
  + IP vs Wired Camera
  + Lens type (narrow angle vs wide angle,field of view)
  + Rolling vs Global shutter
  + Software settings (shutter speed, iso)
  + Positioning (angle, height and distance from subject)
* Camera SDK is another crucial thing we have to consider if hardware vender is providing charging addition cost and providing required toolkit then we should go with it. Otherwise we should develop our own SDK using Nvidia gstreamer and other open source softwares
* I will suggest using two cameras. Camera mounted on entry gates will capture object coming inside while camera mounted at exit gate will capture going away objects

## Compute

* If we choose on cloud deployment then assuming a consistence internet with minimum speed of 2 mbps it should be able to send frames over cloud. There are prebuilt APIs like Azure Face Api which we can directly use with camera to create quick setup without worrying a lot about ML pipeline or else we can create our own ML pipeline which necessary models deployed VM or clusters. There are many services available for this based on requirement
* If we are choosing on edge inference then I will recommend using central gpu which will be cost effective

## Machine Learning

* The three major steps are involved in ML pipeline:
  + **Face Detection**: This is done by trained custom yolo model which is trained for detecting person and faces within an image. Once image face got detected we can crop face to ingest into upcoming step for feature extraction. Person detection will help to track movement of the person. This can be done by linking cameras mounted at entry with other entries cameras and cameras mounted at exit with other exits cameras. (Id will be assigned to person and based on color,shape and size same id will be reflected when he moves inside or away)
  + **Feature Extraction** : For this we can use CNN based architecture to create feature map for faces. I will strongly recommend using FaceNet as it is still best model since it’s first version which was published in 2015. Under the hood, it uses deep learning architectures ZF-Net and Inception which is trained on triplet loss (optimise parameters based on adjusting distance between positive and negative examples with anchor)

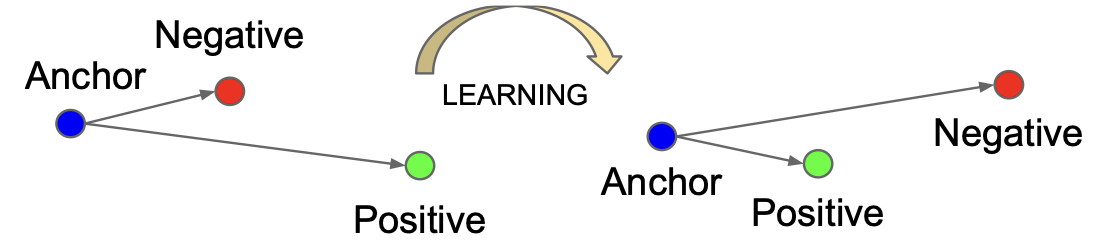


Fig 2 : Embeddings before and after training

* + **Face Classification**: It is final step of classifying faces based on extracted features. Using kNN based approach to calculate distance between known and new faces is not efficient. Instead I suggest,Softmax classifier which memorises boundaries between people which is much more efficient. We can use any other classifier as wel such as SVM, Random Forest, etc. until we have good embeddings

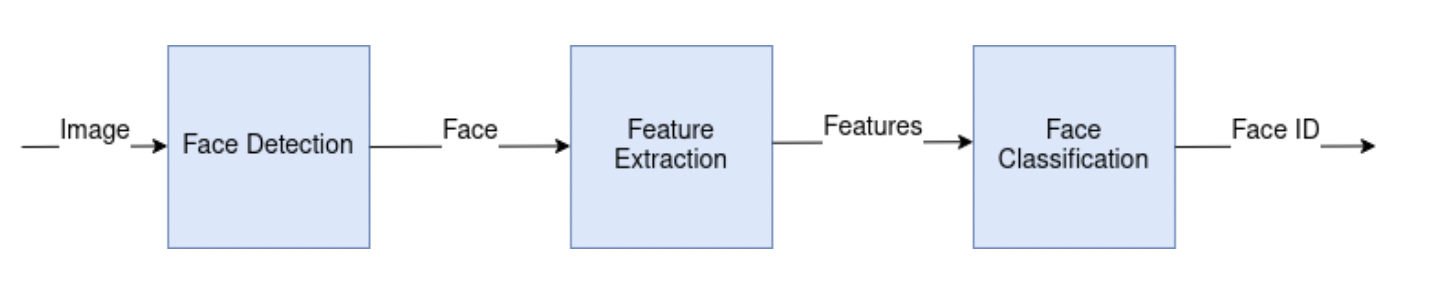


Fig 3: ML Pipeline flow diagram

I am not going into details of ML algorithms, since it will take some time to put everything here for more details refer :

Yolov4 <https://arxiv.org/pdf/2004.10934v1.pdf>

FaceNet <https://arxiv.org/pdf/1503.03832v3.pdf>

* Implementation of Yolov4 can be found here in <https://github.com/pjreddie/darknet> in Pytorch and for Tensorflow use this <https://github.com/tensorflow/models/tree/master/official/vision/beta/projects/yolo>
* Implementation of FaceNet in Tensorflow can be found here <https://github.com/davidsandberg/facenet> and in Pytorch <https://github.com/timesler/facenet-pytorch>