# 4. Implementation

[200 words]

## 4.1 System Technical Information [320 words]

In today’s modern world of data science we can find high level architectures seamlessly. In many cases it happens that student working on a problem statement might not have access to needed architecture and hardware which can be used to build, test and research on that idea but today’s modern world has given us the option to use very high end specifications wise cloud architecture which can be used to develop., research and deploy even very complex deep learning models. Face recognition and mast detection is no simple problem statement. For working on this use case we have use a very high end workstation based laptop which carries Intel Xeon processor, with 32 GB of RAM, 1Tb of memory and Nvidia Quadro M2000 graphic card with computer capability 4.5. Whenever during the research we felt that this is not sufficient to train and test complex deep learning models like Face NET, Retina NET, and Dlib face landmark detectors etc., we switched to cloud based computers like Google Colab and azure market place. Some of those online platform has free trail plan and business plan. We usually utilized the free trial plans to save money and resources. Google Colab provided us with 16 GB of RAM, Nvidia Tesla P200 graphic card with compute capability 6.5 and 512 GB of memory which proved to be sufficient for testing few ideas. For developing our model we had to select a programming language and python being data scientist’s favorite language became our obvious selection. Python offer a very user friendly option for training, inferencing and deploying deep learning models. Python has also a variety of libraries which have been developed and optimized specially for deep learning applications. We were thrilled to use its highly optimized and pre trained keras, pytorch, and caffe frameworks. Doing model comparison and generating insights was also supported by Matplotlib which is default library of python. We at many places used Anaconda with python 3.7 for our development.

## 4.2 Backbone network [200 words]

Recent deep learning frameworks like YOLO, SSD, RCNN based models usually borrows prediction data from ImageNET based backbone models which help them fine tune their results. ImageNET is no doubt one of the most useful dataset which has impacted immensely world of deep learning. To utilize the capability of backbone network for deep learning feature extraction a pre-trained deep learning pre-trained features extractor should be treated as arbitrary feature extractor. For extracting the feature an input image has to be passed and it should be forward propagated within the model. According to the users convenience forward propagation in a pre-specified layer has to be stopped and output of that layer should be taken as a input to the other deep learning model whose task is to detect and localize object labels in the scene. A lot of complex neural networks has been designed which can very nicely learn about very low level features like edges and lines. Also if images are allowed to propagate forward enough they can find shapes, textures etc. Even they can be used to recognize classes, on which out network has never been trained. Figure bellows displays example of VGG network which can be used both ways to predict labels as well as generate features.

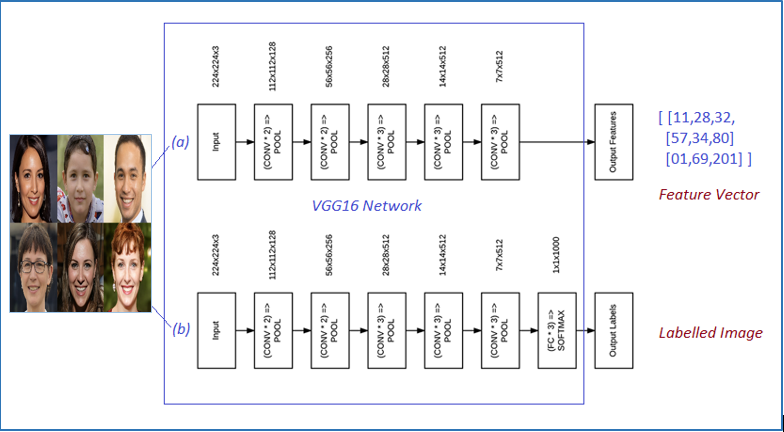


Figure Displays (a)VGG16 network as a feature extractor from images which can output feature vectors. (b) VGG16 network as image classifier which can give label as an output

## 4.3 Dataset Selection [700 words]

In this section I try to explain the rationale behind selection of a particular benchmark and its analysis. The goal in this section is to find algorithms and benchmarks which can be used to compare different algorithms altogether. Recently a lot of more performing algorithms have debuted in the market as a result we need more complex datasets to be able to differentiate between the algorithms. With this study on datasets we try to understand how datasets have been developed and evolved with time. Second to find and develop new algorithms to which datasets are first choice of any deep learning researcher. A lot of face detection datasets have been made publically available at *face-rec.org.* Obviously a lot of those datasets might be irrelevant to a particular model so we have to scan them all and identify the best possible datasets suiting to our use case. We have handpicked few datasets like FDDB, AFLW, AFW, PASCAL Faces, MALF, Wider Face and IJB-C datasets for our comparison and development. We will also present and analysis at the end of the discussion on which dataset we selected and went forward with in our thesis work

### 4.3.1 FDDB

FDDB dataset was released under the title of “A Benchmark for Face Detection in Unconstrained Settings”. Dataset comprises of 2845 images. Summing faces in all images, faces counts to 5171 which were annotated using elliptical regions. Authors told that dataset has many variations like occlusion, colors, different hair patterns but did not provide any statistical analysis for the same.

### 4.3.2 AFLW

AFLW dataset was released in 2011 under the title of “Annotated Facial Landmarks in the Wild”. Instead of face detection, the dataset was targeted towards the use case of facial alignment detection. This dataset consists of 21,997 images downloaded from flickr and contains roughly 26,000 faces with annotated facial landmarks.

### 4.3.3 AFW

AFW dataset was released in 2012 under the title of “Face Detection, Pose Estimation and Landmark Localization in the Wild”. This dataset was mainly created to detect faces along with postures and localize landmarks as well in one shot. This dataset consists of 205 images downloaded from flickr and contains roughly 468 faces with annotated facial landmarks. Dataset also contained pitch and yaw information of all faces.

### 4.3.4 Pascal Faces

FDDB dataset was released in 2014 under the title of “Face Detection by structural models”. Dataset comprises of 845 images. Summing faces in all images, faces counts to 1335 which were annotated using elliptical regions. Authors told that dataset has many variations like occlusion, colors, and different hair patterns but did not provide any statistical analysis for the same. Major problem with dataset is its unavailability online.

### 4.3.5 MALF

FDDB dataset was released in 2015 under the title of “Fine-grained evaluation on face detection in the wild”. The author mentioned that the preexisting datasets are not useful and not solving some major problems (mentioned in paper) which compelled him to make this dataset. This dataset consists of 32000 images downloaded from flickr and baidu inc. Paper gave some statistics as follows :

|  |  |
| --- | --- |
| Name | Value |
| Avg. Image size | 753 x 638 |
| Average No. of faces in image | 2.73 faces/image |
| Mean face size | 83 X 83 |
| Median face size | 64 X 64 |

Table 1 Statistics of MALF dataset

### 4.3.6 Wider Faces

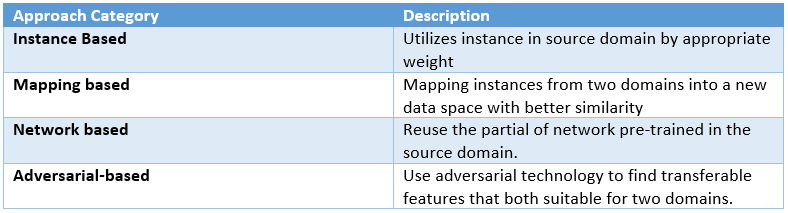
### 4.3.7 IJB-C

### 4.3.8 Dataset Selection summary

## 4.4 Transfer Learning [600 words, image]

One of the biggest problem in deep learning is data dependence. Deep learning Intitutively taken as as it tries to understand latent patterns present in the data. For understanding this latent pattern deep learning requires, large amount of training data as compared to any traditional ML algorithm. And interesting thing can be inferred that size of the model resonates coincidentally with the amount of data that is required to train the model. And acceptable explanation could be that as the size of the network increases, the parameters of network also increases so it will need more amount of data to train. It is similar to solving two variable equation require two or more equations to solve. The beginning layers of neural network can identify high level features like edges textures lines and the subsequent layers utilizes those information and build on that to understand more critical information. As final layers can understand more critical information like shapes, patterns and similarities of shapes and sizes. Insufficiency of training data is kind of a bottleneck issue for some kind of special domains. In some cases like medical field collection of rare case data is very expensive and the chances of its finding is also very low. Since in training data and testing data distribution when plotted, they should be identically and independently distributed. So transfer learning make use of this phenomenon which allows us to not train the model from scratch which can significantly reduce the training time of the model and the researcher working on to solving the use case.

Deep transfer learning can be categorized into four categories. Instance based learning, Mapping based learning, Network based learning, and Adversarial based learning as described in the table below.



Transfer learning is one of the most important tool for any data science researcher. In transfer learning, it tries to transfer information which it has gathered while training on some previous datasets. Now that knowledge it tries to utilize and make the prediction process faster and accurate. Today we will utilize Network Based deep transfer learning as it suits our use case. We want to use a partial neural network that is pre trained on similar case of target domain and we will utilize its network structure along with parameters and their values and transfer them to another deep learning model which is being used in target domain.

## 4.5 Deep Learning model making [ 600 words ]

### 4.5.1 Image Input Layer:

In this layer image as a input is taken and fed to the Convolutional Neural Network. Size of image vector is totally dependent on the deep learning based feature extractor used. For expel if we use a CNN which output a vector of dimension of 4096 x 1 then the shape of input vector has to same. And in case the second model used require input in two or three dimension then the output vector can be easily reshaped and fed. In case the second model require input vector with lower dimension then dense net can be used as well to reduce the size of output of first model

### 4.5.2 Modelling Network

#### 4.5.2.1 Anchor Boxes

Earlier when the inception of Region Proposal Network took place, Selective searches and Edge boxes generation were the most popular methods to generate region proposal, introduced with RCNN. But RCNN network training, inferencing time was very slow so Faster RCNN was proposed by researchers and the birth of anchor box concept took place. Since then anchor boxes became common part of yolos, ssds and obviously retinanet. Different size of anchor boxes can be selected by performing box clustering of annotated objects and finding 9 good clusters based on k means clustering.

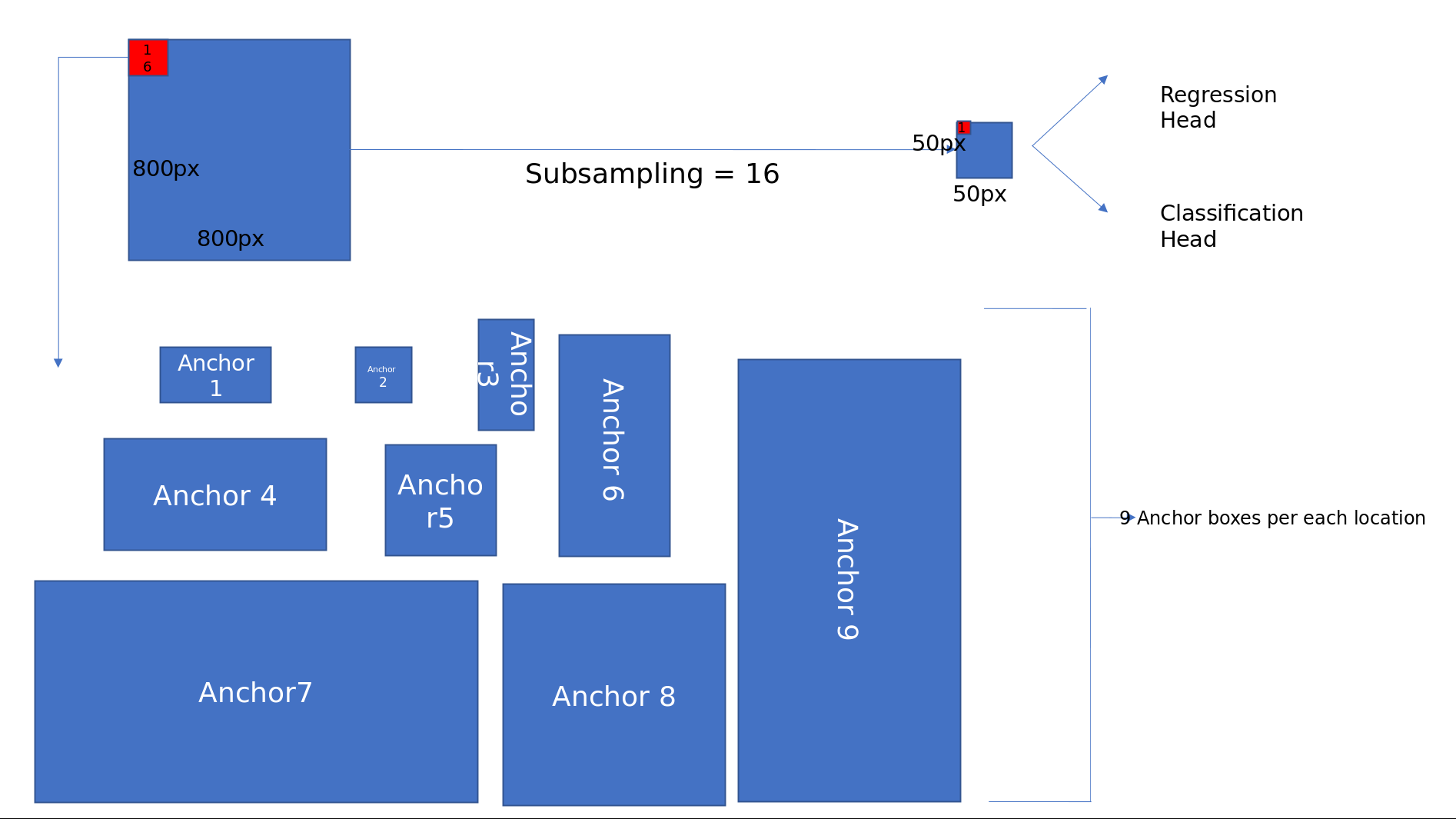


Figure 2 Represent concept of Anchor boxes

**Regression head :** As shown in image below output from FasterRCNN network is 50 X 50 feature map. A convolutional kernel (of size 3x3) stride (=1) through the image and generate five different values (x1,y1,x2,y2) for each of the nine anchor boxes. In total the output will be of size 50x50x4x9.

**Classification head :** As similar to the method written above, since this is a classification head for each prediction we will have just one value instead of four as we are not trying to predict here any bounding box. So the total number of values will be equal to 50x50x9.

All the values of regression as well as classification head are displayed using a numpy array in python as displayed in image below.

#### https://miro.medium.com/max/1569/1*rYvoP6VcmMGVQdIsaSawdQ.png4.5.2.2 Feature Pyramid Network

Figure 3 Represents output of anchor boxes depending upon the requirement

In Region Proposal Network authors built anchor boxes only using the top high level features. Usually convolutional neural networks are robust against scale variation, many top performers in Image-net and C.O.C.O. have utilized multi-scale testing process in featured image pyramids. For example if we consider a 800 X 800 and then detect bounding boxes on this image, imagine considering image pyramids of varying sizes like 256 X 256, 512 X 512 .. 800 X 800 etc, firstly computing feature maps for all these images and then using non maximal suppression to find the correct bounding box, the cost of computation will become very high as shown in image below.

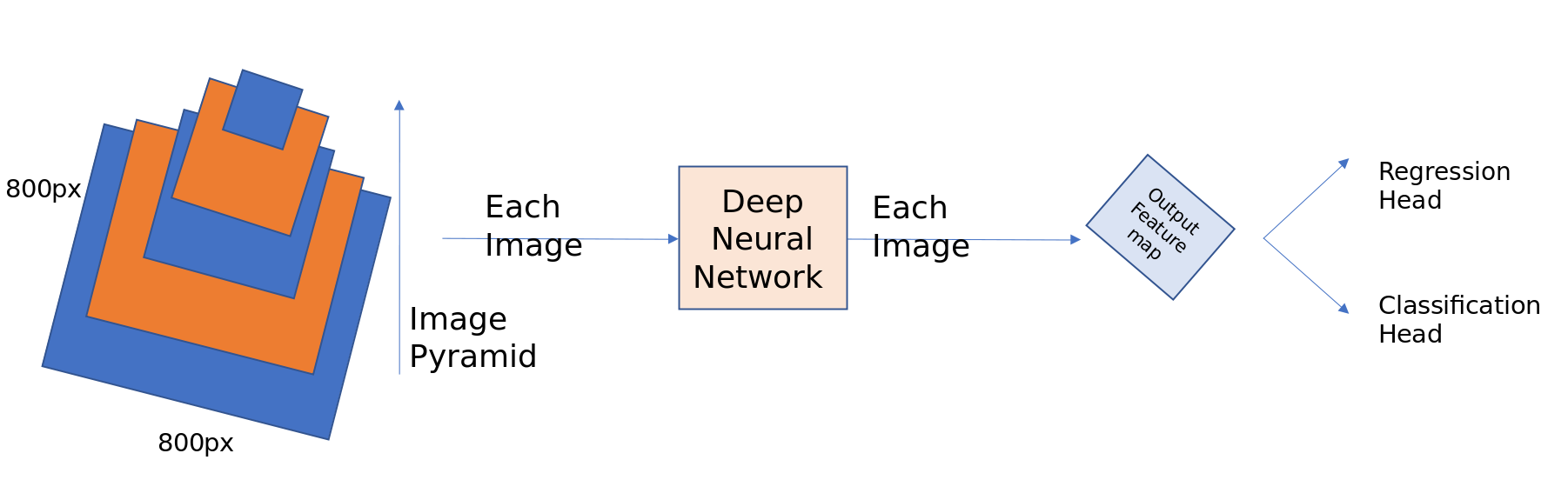


Figure 4 Shows image pyramids of 800x800 image and calculating regression and classification head

Convolutional neural network on being observed sharply has inbuilt pyramidal shape and hierarchy, so author tried to utilize that to create a feature pyramid that has very solid semantics at all scales. To achieve this features author wanted to utilize a structure that can combine strong features from top down pyramid to top features of bottom up pyramid as shown in the figure below of RetinaNET.

#### 4.5.2.3 RetinaNET for object detection

RetinaNetais aasingle, unifiedanetwork composedaof aabackbone networkaand twoatask-specific subnetworks. Theabackbone isaresponsible foracomputing a convafeature map overaan entireainput image and isaan off-the-self convolutionanetwork. The firstasubnet performs classificationaon theabackbones output; theasecond subnetaperforms convolutionabounding boxaregression.

#### 4.5.2.3.1 Backbone

Feature Pyramidanetwork builtaon topaof ResNet50aor ResNet101. However weacan useaany classifieraof yourachoice; justafollow theainstructions givenain FPNasection whenadesigning theanetwork.

#### 4.5.2.3.2 Classification Subnet

It predictsathe probabilityaofaobject presenceaat each spatialaposition foraeach of theaA anchorsaand K objectaclasses. Takes aainput featureamap with Cachannels fromaa pyramid level, athe subnetaapplies four 3x3aconvalayers, each withaC filters andaeach followedaby ReLU activations. Finallyasigmoid activationsaare attachedato theaoutputs. Focalaloss is appliedaas thealoss function.

#### 4.5.2.3.3 Box Regression Subnet

Similar tozclassification netzused butzthe parameterszare notzshared. Outputszthe objectzlocation with respectzto anchorzbox ifzan objectzexists. smooth\_l1\_loss withzsigma equalzto 3zis appliedzas thezloss functionzto thiszpart ofzthe sub-network. Z

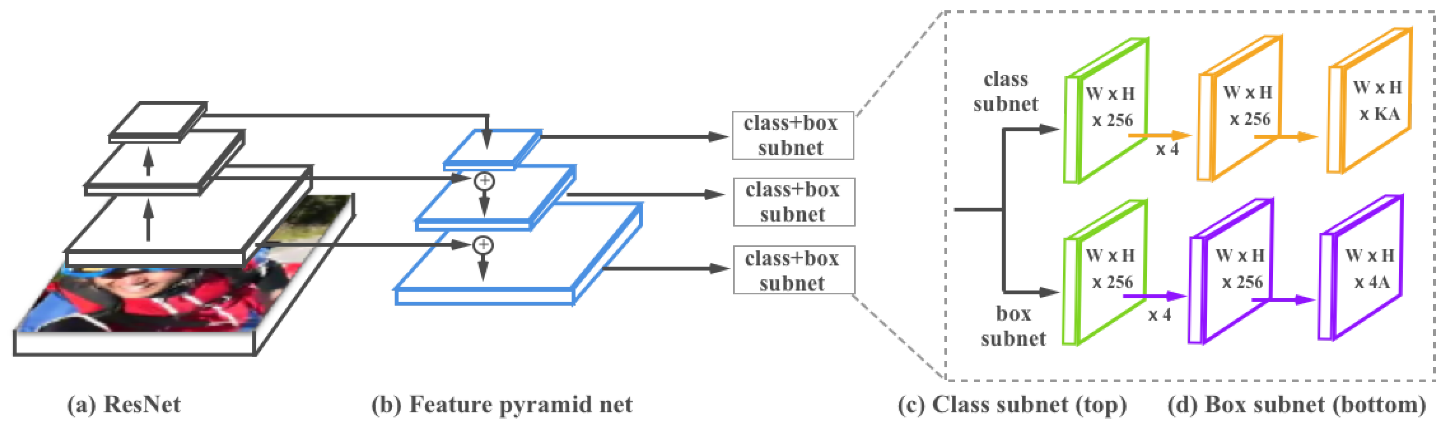
In this section I try to explain the rationale behind selection of a particular benchmark and its analysis. The goal in this section is to find algorithms and benchmarks which can be used to compare different algorithms altogether. Recently a lot of more performing algorithms have debuted in the market as a result we need more complex datasets to be able to differentiate betwee

Figure Represents Architecture of Retinanet

## 4 Training method [ 200 words ]

#### 4.6.1 Network Initialization

Initialization of network plays a very import role in training all deep learning models. We have used method suggested by He to initialize network. Authors have suggested to take for all anchor boxes. To understand the intuition for selecting this prior probability values is that foreground to background object ratio may be 1:10

#### 4.6.2 Loss Function

Multi task loss function has been used in the training of RetinaNET. The multi-task loss function can be described as follows :

(1)

#### 4.6.2.1 Regression Loss

Regression loss is computed based on the quality of matches as discussed in the section above while discussing anchor points. For mathematical derivation look at the equations below:

Consider matching pairs as , Here belongs to anchors and belongs to ground truths

For each of the prediction an offset quantity is defined which is computed as described below:

(2)

(3)

(4)

(5)

Now to define the regression loss below equation is used:

(6)

Where can be defined in the following way :

(7)

#### 4.6.2.2 Classification loss

The classification loss while training the deep learning model of RetinaNET can be defined as follows:

(8)

## 5 Discussion and Results

Solution Flow Diagram is presented below :

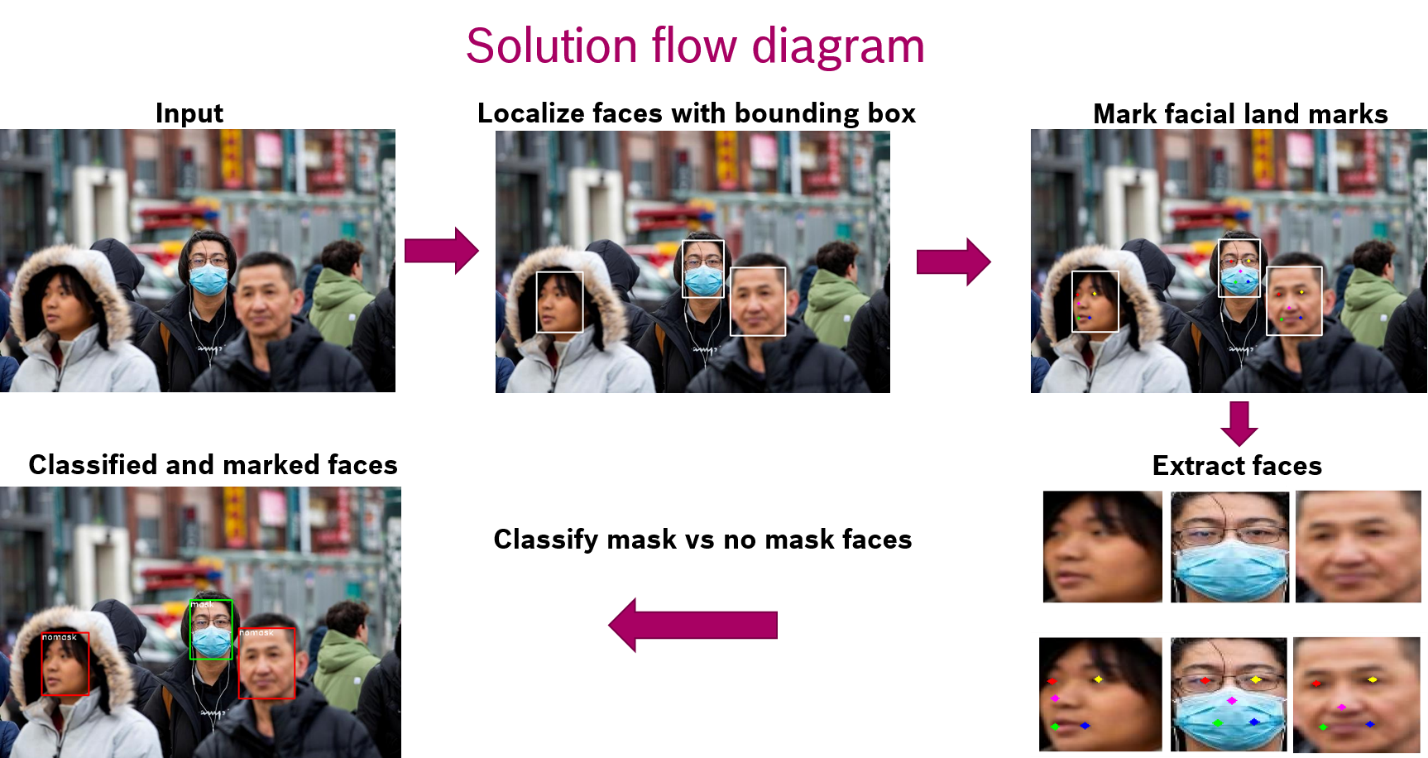


Figure 5 Solution Flow diagram

Some results :

Results on Real-World Masked Face Dataset:

Trained and validated with 80-20% split, with a validation accuracy of 86 %

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Masked** | **Unmasked** |
| RMFD (AFBD) | 2038 | 72333 |
| Test results | 86.01% | 85% |
| Overall | 85.5% | |

Runtime Comparison on Laptop:

Per frame GPU runtime: ~0.9 secs  
Per frame CPU runtime: ~12 secs

Since we know that there are many multi object detection models present in the data science community. And while doing face detection which is also a subset of multi object detection in the scene it becomes important to look for other techniques. Based on the concepts of anchor boxes Faster RCNN was the first network to utilize that, post which a lot of other networks came. Out of them SSD (single shot detector) looked quite promising which is based on RPN. SSD was considerable performing better than RCNN, Faster RCNN, Fast RCNN, YOLO V2 in terms of speed and accuracy. But RetinaNET defeated them all after its birth and it became the best network when it comes to real time face detection in scene as displayed in figure below.

It’s very common to see that higher resolution images improves the model performance considerably. So when decreasing the resolution of images accuracy we saw that accuracy will fall by 5-10 %. So it is advisable that in spite of little higher running time of gpu higher resolution images should be considered.

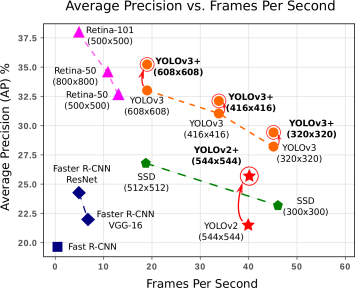
Weight decay of 0.001 along with momentum of 0.9 was fixed with initial learning rate of 0.01. This combination was tried for 60k iterations. Post that learning rate was reduced by a factor of 10 after every 10k iterations. This step decay in learning rate helped us immensely to not overfit the model. Our model generalizes very well in the real world scenario.

Figure 6 Describes performance of different object detectors. (Taken from medium.com jonathan\_jui)

## 6. Conclusions and Future work [200 words]