DIABETIC RETINOPATHY DETECTION

Submitted in partial fulfillment of the requirements
of the degree of

B. E. Computer Engineering

By

Niral Neres Almeida 04
Rudolph Ignatius Almeida 05
Melissa Allwin D'Cunha 20

Under the Guidance of
Ms. Vincy Joseph
Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2017-2018

CERTIFICATE

This is to certify that the project entitled **"Diabetic Retinopathy Detection"** is a bonafide work of **"Melissa Allwin D'Cunha (Roll no. 20), Rudolph Ignatius Almeida (Roll no. 05), Niral Neres Almeida (Roll no. 04)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering.

Miss. Vincy Joseph
Supervisor/Guide

Dr. Kavita Sonawane
Head of Department

Dr. Sincy George
Principal

Project Report Approval for B.E.

This project report entitled **Diabetic Retinopathy Detection** by Rudolph Almeida, Niral Almeida, Melissa D'Cunha is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1.----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Rudolph Almeida - 05

Niral Almeida - 04

Melissa D’Cunha - 20

Date:

Abstract

The project aims to build a comprehensive and automated system capable of identifying the extent to which a person in suffering from diabetic retinopathy which is one of the leading causes of blindness in the world. Current DR detection methods are manual based i.e. a physician manually checks images for features inherent to DR. The system given a pair of color fundus photographs, will provide a rating between 0-4 which tells the extent to which DR occurs in the eye. The system will be trained using supervised machine learning algorithms with a dataset, which has been labelled and rated by a practicing physician. The goal is to push the automated system to the limit of what is possible – ideally resulting a model which has realistic clinical potential. The system will be a desktop based system with a user interface which allows easy uploading of images to the system, which will then classify the image.

Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1	DR	Diabetic Retinopathy
2	CNN	Convolutional Neural Network
3	ANN	Artificial Neural Network
4	ML	Machine Learning
5	SVM	Support Vector Machine
6	RGB	Red Green Blue
7	DFD	Data Flow Diagram

Chapter 1

Introduction

Diabetic retinopathy is the leading cause of blindness in the working-age population of the developed world. It is estimated to affect over 93 million people. The US Center for Disease Control and Prevention estimates that 29.1 million people in the US have diabetes and the World Health Organization estimates that 347 million people have the disease worldwide. Diabetic Retinopathy (DR) is an eye disease associated with long-standing diabetes. Around 40% to 45% of Americans with diabetes have some stage of the disease. Progression to vision impairment can be slowed or averted if DR is detected in time, however this can be difficult as the disease often shows few symptoms until it is too late to provide effective treatment. Currently, detecting DR is a time-consuming and manual process that requires a trained clinician to examine and evaluate digital color fundus photographs of the retina. By the time human readers submit their reviews, often a day or two later, the delayed results lead to lost follow up, miscommunication, and delayed treatment.

Clinicians can identify DR by the presence of lesions associated with the vascular abnormalities caused by the disease. While this approach is effective, its resource demands are high. The expertise and equipment required are often lacking in areas where the rate of diabetes in local populations is high and DR detection is most needed. As the number of individuals with diabetes continues to grow, the infrastructure needed to prevent blindness due to DR will become even more insufficient.

With a new wave of research in the architecture and applications of CNNs, and various other machine learning algorithms, here is our take on the same.

1.1 Description

Our project aims to build a comprehensive and automated system which expects a color fundus photograph as input and returns a rating based on how severe the spread of DR is in that particular eye. The system generates a value between 0-4 for each image, where 0 represents NO DR and 4 represents proliferative DR. The system is planned to be a desktop based system with a very simple user interface.

1.2 Problem Formulation

DR is a serious and proliferative disease. Most of the blindness cases are due to DR itself.

DR is caused due to long standing cases of diabetes. With the amount of cases of diabetes being detected, the number of patients with DR is likely to increase a lot more.

However, the effects of DR can be avoided and cured if it is detected in the early stages. However, this doesn't usually happen as doctors take a couple of days to analyze a fundus photograph, which is a magnified image of the inner eye taken using a special microscope. This often leads to lost follow-up or loss of data and may result in the patient not knowing about their condition. A comprehensive and automated system is needed which, given a color fundus photograph, can immediately predict the rating and thereby the status of a patient immediately after taking a photograph.

1.3 Motivation

Artificial Intelligence has been touted as the "Next Big Thing" for the longest time now. With all services being automated along with self-driving cars, we are closer than ever to this next big thing. Machine Learning is an integral part of AI in which the machine learns and recognizes patterns without being coded explicitly, i.e. Structured Programming

Many earlier attempts at making an automated DR screening system have not provided satisfactory results. As such doctors still use the manual method of analyzing the image by themselves and providing a rating. Confidence in such systems is low due to their low accuracy and high unreliability. These methods employed techniques based on machine learning, traditional neural nets and image processing.

An important aim of our project is to design a model which will improve upon the results of these previous attempts. We need to achieve high reliability and make a system which is ideally ready for clinical usage.

1.4 Proposed Solution

Initially each input or training image will go through a preprocessing step. Image processing techniques such as changing the channel of the input image to green from RGB. According to our references green channel holds the best information in an image. This will allow us to drastically reduce the size of an input image. Other techniques such as noise removal, histogram equalization and min-max normalization will be employed to process the image, reduce its size,

retain only the information i.e. needed. This allows faster training and processing of the images by the system. The model will be trained on the training dataset which will use features inherent to a particular class and learn to predict it by matching the image features with its learned features for a particular class. The clinical operator will be presented with a simple to use user-interface. This user-interface will allow easy uploading of images to the system and will present the predicted output to the user.

1.5 Scope of the Project

The project is intended to be used by clinical experts and researchers who might want it to use to accurately predict the class of a color fundus photograph. Clinics can employ the system to provide reports to patients at the earliest. This will hopefully lead to DR being detected at its early stages and thereby reducing the cases of serious DR among the population. Providing accurate results is necessary and incase the medical expert notices an error made by the system, they will be able to note the error which will then be corrected by the system.

The system will be trained on an DR dataset. But machine learning models can be adapted for multiple uses with little to no changes. The system can then be easily modified to work with other datasets and problems which deal with noting features from images and the making a decision based on it features. An example of such a system would be brain tumor detection from MRI or X-Ray brain scans.

There is no doubt that in the manual medical practices performed by experts today are going to be replaced or at least augmented by machine learning applications/models. However, this is only going to be possible if the general public who is going to be dependent on these systems have a good amount of trust in the system. This is only going to be possible if the system is reliable and produces accurate results most of the time. We hope the system can be such a system, which paves the way and interest for better DR systems in the near future.

Chapter 2

Review of Literature

2.1 Image Preprocessing

Many different instruments or microscopes are used to obtain or picture the retina of a diabetic person. These results in images of different formats and characteristics. Images in the dataset and in practical use come in various different dimensions, have noise and other artifacts, don't have good contrast and are not uniform in nature. However, the system expects all images to be of uniform nature. Also, we don't want the system to learn artifacts as features of the images, thereby influencing the prediction. Hence it is necessary for the images to be preprocessed before they are fed to the classifier. Image processing is the processing of digital images using mathematical operations by using any form of signal processing. The various techniques to be applied are:

a. Channel Extraction: Digital images captured by the microscopes and instruments consist of 3 channels i.e. RGB. These three channels combined together form the full 24-bit color image; each channel consisting of 8 bits. However, an input of the full 24-bit image will result in very large memory requirements and reduce the speed at which the system can learn and predict. Also, it has been observed that all three channels are not necessary to hold complete information and enough information can be extracted from the green channel itself. Hence all images will be converted to just a single green channel which will hold enough information so as to not reduce the accuracy of the system while reducing the memory and time requirements.

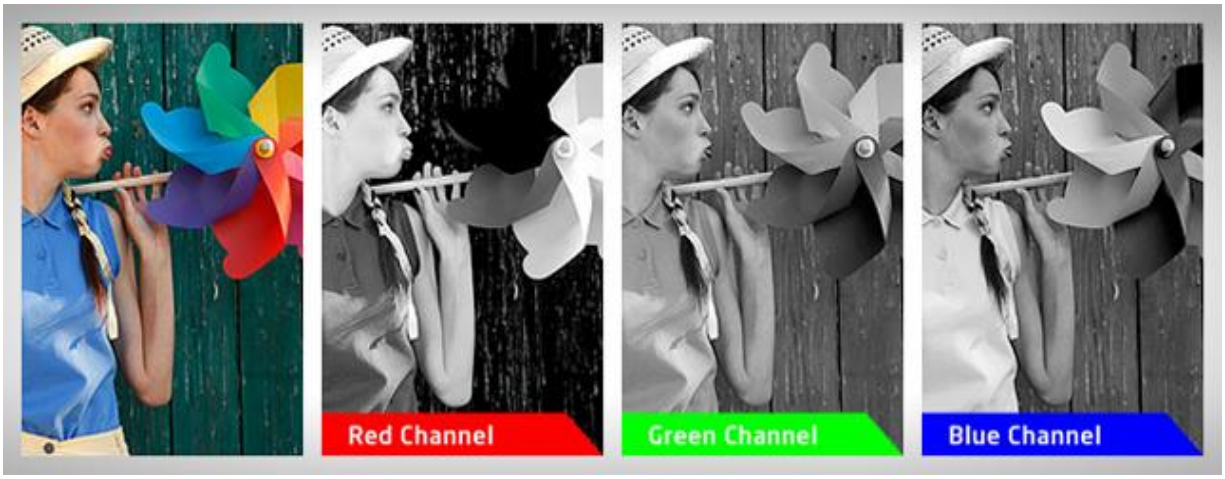

Figure 2.1 - RGB Color Channels of an image

b. Image Reflection about X-axis: Some images in the dataset and through instruments appear as they would appear to an eye anatomically or through a microscope condensing lens. They are characterized by a notch at the top right of the eye, rather than the expected bottom right. Such images will be flipped about the X-axis and brought to the proper alignment.

c. Noise Removal: Both images and labels in our dataset contain noise. Instruments that are used to capture color fundus photographs are bound to have noise in them. Because we don't want the models to learn the inherent noise in the images as features, it is necessary to remove noise from the images. All capturing devices have traits, that make them susceptible to noise. Noise can be random or white noise with no coherence, or coherent noise introduced by the device's mechanism or processing algorithms. In salt and pepper noise (spare light and dark disturbances), pixels in the image are very different in color or intensity from their surrounding pixels; the defining characteristic is that the value of a noisy pixel bears no relation to the color of surrounding pixels. In Gaussian noise, each pixel in the image will be changed from its original value by a (usually) small amount. A histogram, a plot of the amount of distortion of a pixel value against the frequency with which it occurs, shows a normal distribution of noise. While other distributions are possible, the Gaussian (normal) distribution is usually a good model, due to the central limit theorem that says that the sum of different noises tends to approach a Gaussian distribution. In either case, the noise at different pixels can be either correlated or uncorrelated; in many cases, noise values at different pixels are modeled as being independent and identically distributed, and hence uncorrelated. Some methods to remove noise from images are:

i. **Linear Smoothing Filters:** One method to remove noise is by convolving the original image with a mask that represents a low-pass filter or smoothing operation. For example, the Gaussian mask comprises elements determined by a Gaussian function. This convolution brings the value of each pixel into closer harmony with the values of its neighbors. In general, a smoothing filter sets each pixel to the average value, or a weighted average, of itself and its nearby neighbors; the Gaussian filter is just one possible set of weights. Smoothing filters tend to blur an image, because pixel intensity values that are significantly higher or lower than the surrounding neighborhood would "smear" across the area. Because of this blurring, linear filters are seldom used in practice for noise reduction; they are, however, often used as the basis for nonlinear noise reduction filters.

ii. **Wavelet transform:** The main aim of an image denoising algorithm is to achieve both noise reduction and feature preservation. In this context, wavelet-based methods are of particular interest. In the wavelet domain, the noise is uniformly spread throughout coefficients while most of the image information is concentrated in a few large ones. Therefore, the first wavelet-based denoising methods were based on thresholding of detail sub-bands coefficients. However, most of the wavelet thresholding methods suffer from the drawback that the chosen threshold may not match the specific distribution of signal and noise components at different scales and orientations.

d. Histogram Equalization: Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance. The horizontal axis of the graph represents the tonal variations, while the vertical axis represents the number of pixels in that particular tone. The left side of the horizontal axis represents the black and dark areas, the middle represents medium grey and the right-hand side represents light and pure white areas. The vertical axis represents the size of the area that is captured in each one of these zones. Thus, the histogram for a very dark image will have the majority of its data points on the left side and center of the graph. Conversely, the histogram for a very bright image with few dark areas and/or shadows will have most of its data points on the right side and center of the graph. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better

distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

Implementation: Consider a discrete grayscale image $\{x\}$ and let n_i be the number of occurrences of gray level i. The probability of an occurrence of a pixel of level i in the image is

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 \leq i < L$$

L being the total number of gray levels in the image (typically 256), n being the total number of pixels in the image, and $p_x(i)$ being in fact the image's histogram for pixel value i, normalized to $[0, 1]$.

The cumulative distribution function corresponding to p_x as,

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

Which is also the image's accumulated normalized histogram.

We would like to create a transformation of the form $y = T(x)$ to produce a new image $\{y\}$, with a flat histogram. Such an image would have a linearized cumulative distribution function (CDF) across the value range, i.e.

$$cdf_y(i) = iK$$

For some constant K. The properties of CDF allow us to perform such a transform; it is defined as

$$cdf_y(y') = cdf_y\big(T(k)\big) = cdf_x(k)$$

Where k is in the range [0, L]. Notice that T maps the levels into the range [0, 1], since we used a normalized histogram of $\{x\}$. In order to map the values back into their original range, the following simple transformation needs to be applied to the result:

$$y' = y \cdot (\max\{x\} - \min\{x\}) + \min\{x\}$$

Suppose that a 3-bit image (L = 8) of size 64×64 pixels (MN = 4096) has the intensity distribution given below, where the intensity levels are integers in the range $[0, L-1] = [0, 7]$

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19

$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

The histogram of our hypothetical image is given as,

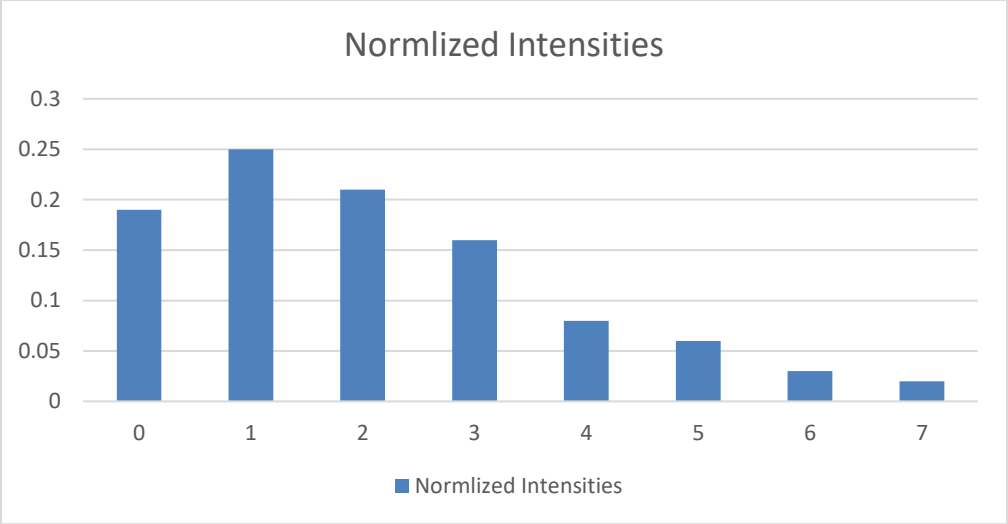

Figure 2.2 - Normalized Histogram of an image

Values of the histogram equalization transformation are obtained as,

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33 \rightarrow 1$$

Similarly,

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08 \rightarrow 3$$

And $s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.86, s_7 = 7.00$. The s values still have fractions because they were generated by summing probability values, so we round them to the nearest integer:

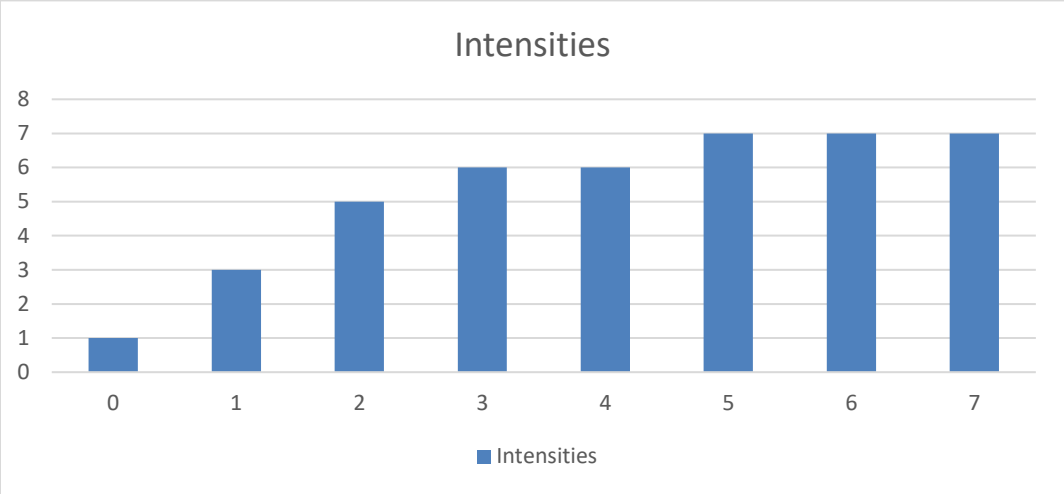

Figure 2.3 - Histogram after Equalization

These are the values of the normalized histogram.

2.2 Convolutional Neural Networks

An Artificial Neural Network replicates the workings that of a nervous system of a body. An Artificial Neural Network consists of an input layer, a set of hidden layers and an output layer. Each layer consists of a given number of neurons that are triggered by an activation function which triggers the network of neurons. Each neuron is then connected to each other neuron by a connection known as a synapse. Each synapse has a weight associated to it that aims to excite other neurons for the same.

For training the model, we will be using a Convolutional Neural Network. A CNN is a class of deep, feed-forward neural networks that have been successfully applied to analyzing visual imagery. CNNs use a variation of multilayer perceptron's designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks(SIANN), based on their shared-weights architecture and translation invariance

characteristics. CNNs were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

Typical Architecture of a Convolutional Neural Network

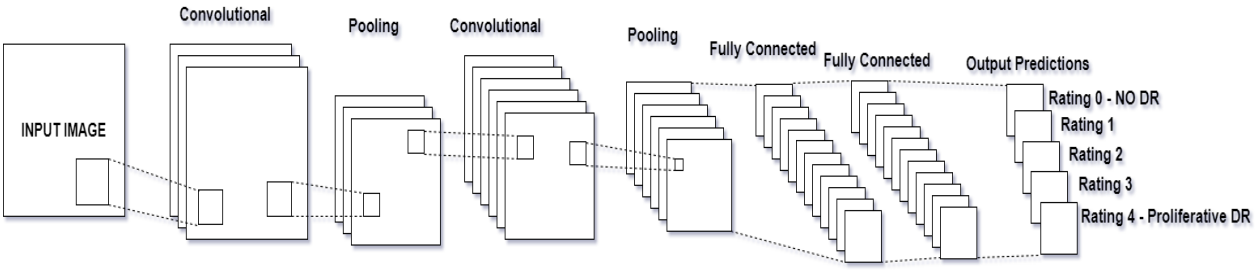

Figure 2.4 - Architecture of a CNN

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are either convolutional, pooling or fully connected.

Convolutional

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. Tiling allows CNNS to tolerate translation of the input image.

Although fully connected feed forward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary even in a shallow architecture. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. In other words, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation

Pooling

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each cluster of neurons at the prior layer.

Fully Connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network(MLP).

Weights

CNNS share weights in convolutional layers, which means that the same filter is used for each receptive field in the layer; this reduces memory footprint and improves performance.

What can CNNS do?

CNNs have applications in image and video recognition, recommender systems and natural language processing.

Hyperparameters in CNNs

- **Number of Filters:** Since feature map size decreases with depth, layers near the input layer will tend to have fewer filters while higher layers can have more. To equalize computation at each layer, the feature x pixel position product is kept roughly constant across layers. Preserving more information about the input will require keeping the total number of activations nondecreasing from one layer to the next
- **Filter Shape:** Common field shapes found in the literature vary greatly, and are usually chosen based on the dataset. The challenge is thus to find the right level of granularity so as to create abstractions at the proper scale, given a particular dataset.
- **Max Pooling Shape:** Typical values are 2x2. Very large input values may warrant 4x4 pooling in the lower layers. However, choosing larger shapes will dramatically reduce the dimension of the signal, and may result in excess information loss. Often, non-overlapping pooling windows perform best.

2.3 Naïve Bayes Classifier

In machine learning, naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong(naïve) independence assumptions between the features.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Naïve Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principal: all naïve Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round and about 10cm in diameter. A naïve Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Probabilistic model

Abstractly, naïve Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\boldsymbol{x = (x_1, x_2, \ldots, x_n)}$ representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k \mid x_1, \ldots, x_n)$$

for each of k possible outcomes or classes $\boldsymbol{C_k}$.

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k \mid x) = \frac{p(C_k)\, p(x \mid C_k)}{p(x)}$$

In plain English, using Bayesian probability terminology, the above equation can be written as,

$$posterior = \frac{prior \times likelihood}{evidence}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features x_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k \mid x_1, \ldots, x_n)$$

Which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$p(C_k \mid x_1, \ldots, x_n) = p(x_1, \ldots, x_n, C_k) = p(x_1 \mid x_2, \ldots, x_n, C_k)$$

$$= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) \ldots p(x_{n-1} \mid x_n, C_k) p(x_n, C_k) p(C_k)$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C. This means that

$$p(C_k \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$$

Thus, the joint model can be expressed as

$$p(C_k \mid x_1, \ldots, x_n) \propto p(C_k, x_1, \ldots, x_n) \propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Where the evidence $Z = p(x)$ is a scaling factor dependent only on $x_1, \ldots, x_n$, that is, a constant if the value of the feature variables is known.

Constructing a classifier from the probability model

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some *k* as follows:

$$\hat{y} = \underset{k \in \{1,\ldots,k\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Discussion

Despite the fact that the far-reaching independence assumptions are often inaccurate, the naive Bayes classifier has several properties that make it surprisingly useful in practice. In particular, the decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This helps alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features. While naive Bayes often fails to produce a good estimate for the correct class probabilities, this may not be a requirement for many applications. For example, the naive Bayes classifier will make the correct MAP decision rule classification so long as the correct class is more probable than any other class. This is true regardless of whether the probability estimate is slightly, or even grossly inaccurate. In this manner, the overall classifier can be robust enough to ignore serious deficiencies in its underlying naive probability model.

2.4 Random Forest Classifiers

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Algorithm: Decision Tree Learning

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie et al., because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspect able models. However, they are seldom accurate.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Tree Bagging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, \dots, x_n$ with responses $Y = y_1, \dots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, …, B:

1. Sample with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

Or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

From bagging to random forests

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by Ho.

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend p/3 (rounded down) with a minimum node size of 5 as the default.

Extra Trees

Adding one further step of randomization yields extremely randomized trees, or ExtraTrees. These are trained using bagging and the random subspace method, like in an ordinary random forest, but additionally the top-down splitting in the tree learner is randomized. Instead of computing the locally optimal feature/split combination (based on, e.g., information gain or the Gini impurity), for each feature under consideration, a random value is selected for the split. This value is selected from the feature's empirical range (in the tree's training set, i.e., the bootstrap sample)

2.5 Support Vector Machines

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering and is often used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass.

Motivation

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So, we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier; or equivalently, the perceptron of optimal stability.

Definition

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors x_i that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation:

$\sum_i \alpha_i k(x_i, x) = constant$. Note that if $k(x, y)$ becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space.

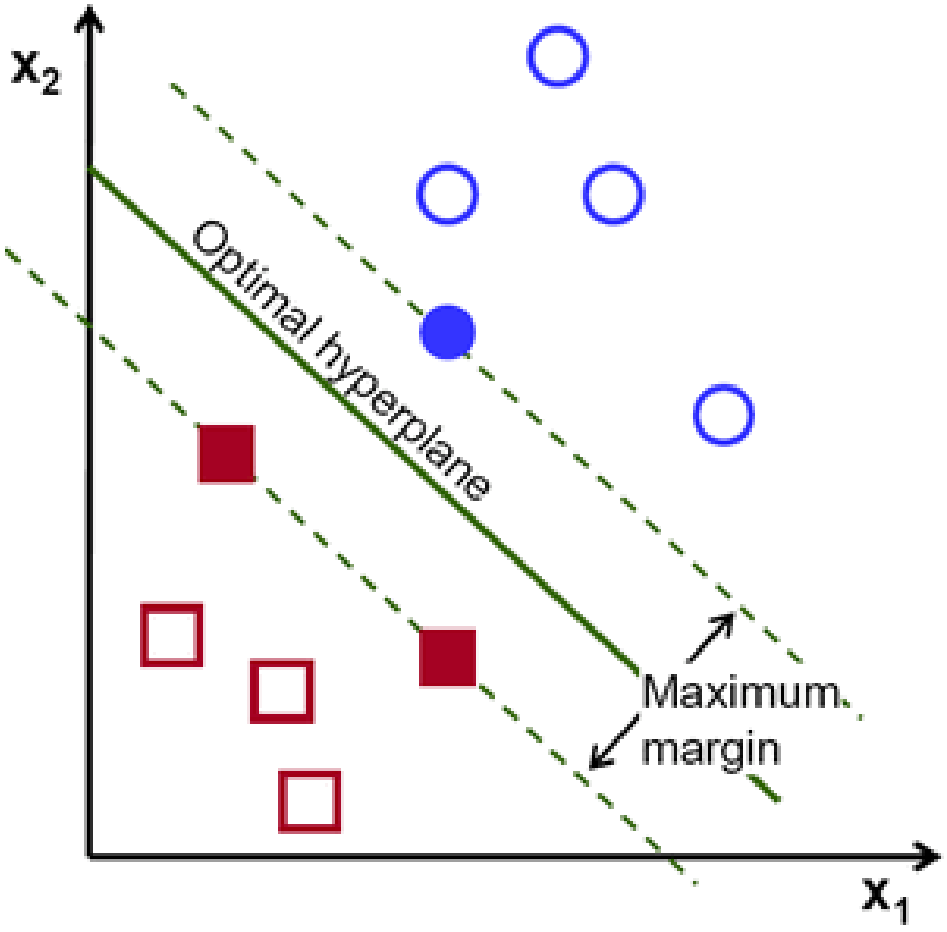

Figure 2.5 - Hyperplanes in SVM

Chapter 3

System Analysis

3.1 Functional Requirements

a. Upload Image: Users such as clinical/medical experts can upload an image from their computer onto our system which will be used for prediction

b. Submit Error: If the medical expert notes a mistake made by the system, they can notify the system that it has made a mistake for that particular input image, which the system will then rectify.

c. Generate result: The system will predict a result based on an input image which will be a value between 0-4 where 0 represents NO DR and 4 represents Proliferative DR.

3.2 Non-Functional Requirements

a. Accuracy: The necessary condition for any medical system that it must be highly accurate. Lives depend on such systems and a single error can cause huge damage. The system needs to be highly sophisticated and accurate and provide a comprehensive result, equal or almost equal to what a practicing physician would have rated.

b. User Friendliness: The system needs to be as simple to use as possible. Users should be easily able to upload images and view their result without doing many actions. The user should be able to specify errors if any.

c. Supportability: The system should work on as many operating systems and platforms as possible, to not limit the usage of the system by the experts based on which OS they use.

d. Speed: The whole point of this system is to reduce the time required for the results to reach the patients. The system should be quick and provide near instantaneous results.

e. Resource Utilization: The system should be use as less resources as possible thereby making it available to be adapted by as many clinics as possible.

3.3 Specific Requirements

a. Hardware Requirements

i. Developer

1. Minimum 4GB RAM. Higher RAM results in faster training

2. Minimum 100 GB hard disk space.
3. Nvidia GPU with minimum CUDA level of 3.0

ii. User

1. Minimum 512 MB RAM
2. Processor: Minimum Core 2 Duo 2GHz or equivalent

b. Software Requirements

i. Developer

1. Linux OS such as Ubuntu/Mint
2. Python 3.5 and above
3. A rich text editor such as Visual Studio Code. The developer may use any text editor he is comfortable with.

ii. User

1. Basic Operating system such as Windows, Linux or Mac OS

3.4 Use-Case Diagrams and Descriptions

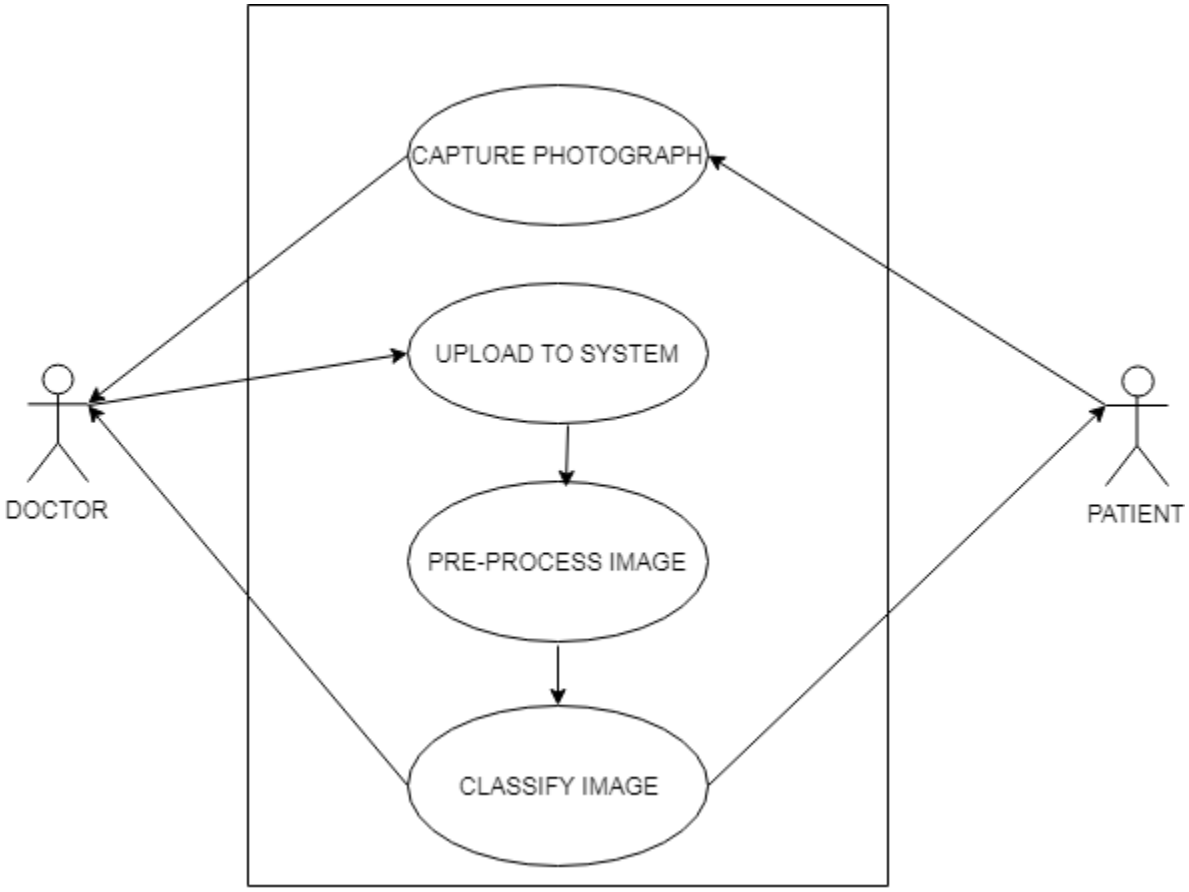

Figure 3.1 - Use Case Diagram

- Doctor is any user who operates and administers the system
- Doctor can capture a color fundus photograph and upload it to the system
- Patient is any user who needs a result from the system
- The patient is given the result of the system

Use Case	Capture Photograph
Use Case ID	UC01
Actor	Doctor and Patient
Description	The doctor captures a retinal image of the patient using a special microscope

Use Case	Upload to System

Use Case ID	UC02
Actor	Doctor
Description	The image is uploaded by the doctor to the system for predicting its result

Use Case	Preprocess Image
Use Case ID	UC03
Actor	-
Description	The image is preprocessed by the image to remove noise, reduce its size and remove unnecessary information from it

Use Case	Classify Image
Use Case ID	UC04
Actor	Patient
Description	The system classifies the image in one of 4 categories and displays the result to the users of the system

Chapter 4

Analysis Modeling

4.1 Activity Diagram

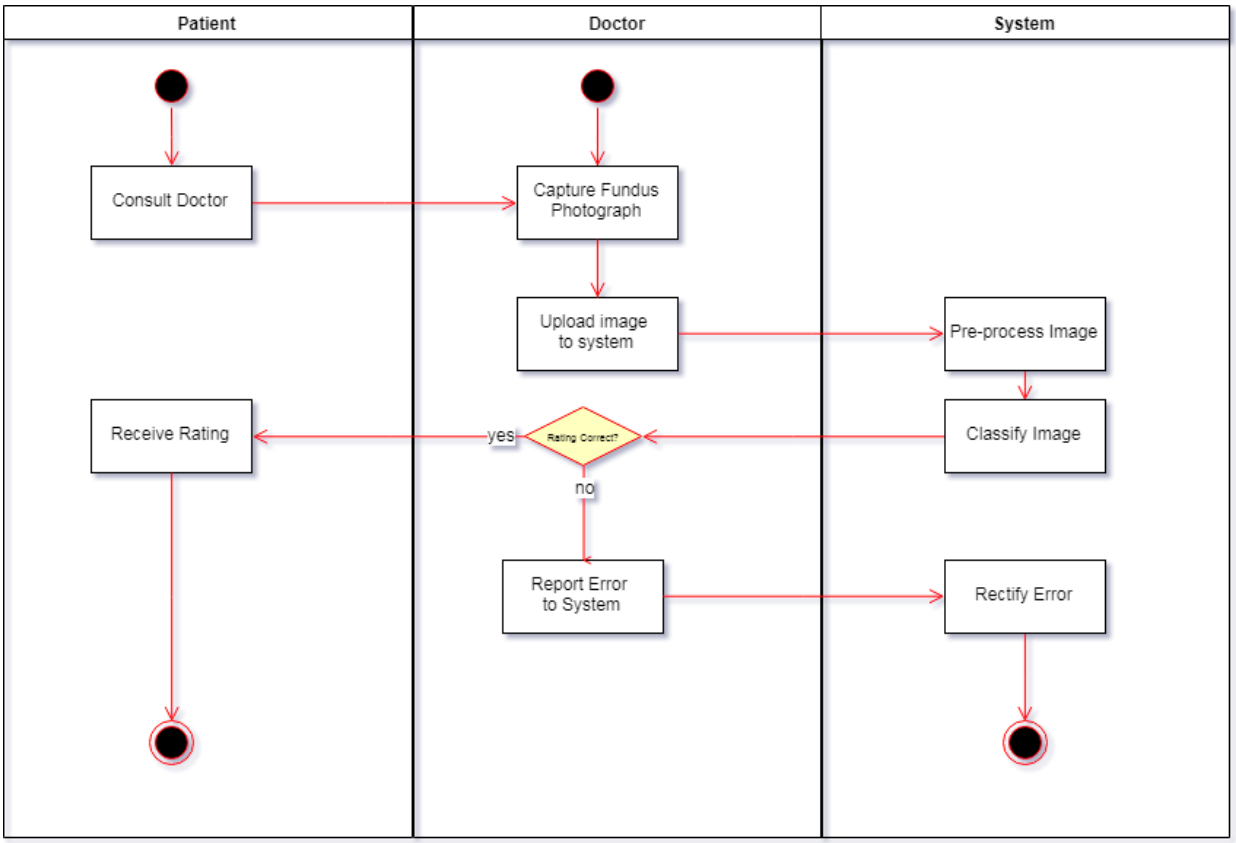

Figure 4.1 - Activity Diagram

4.2 Functional Modeling

LEVEL 0 DFD

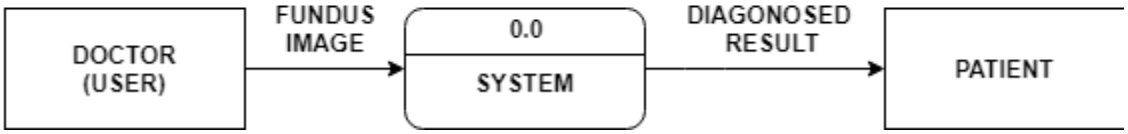

Figure 4.2 - Level 0 DFD

LEVEL 1 DFD

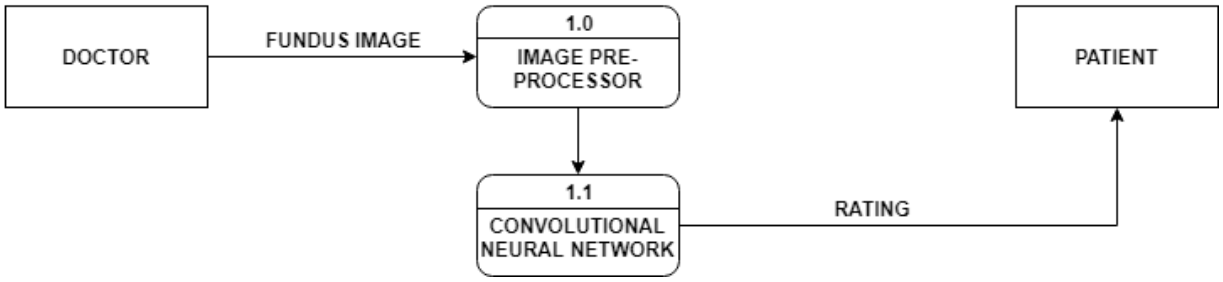

Figure 4.3 - Level 1 DFD

4.3 Timeline Chart

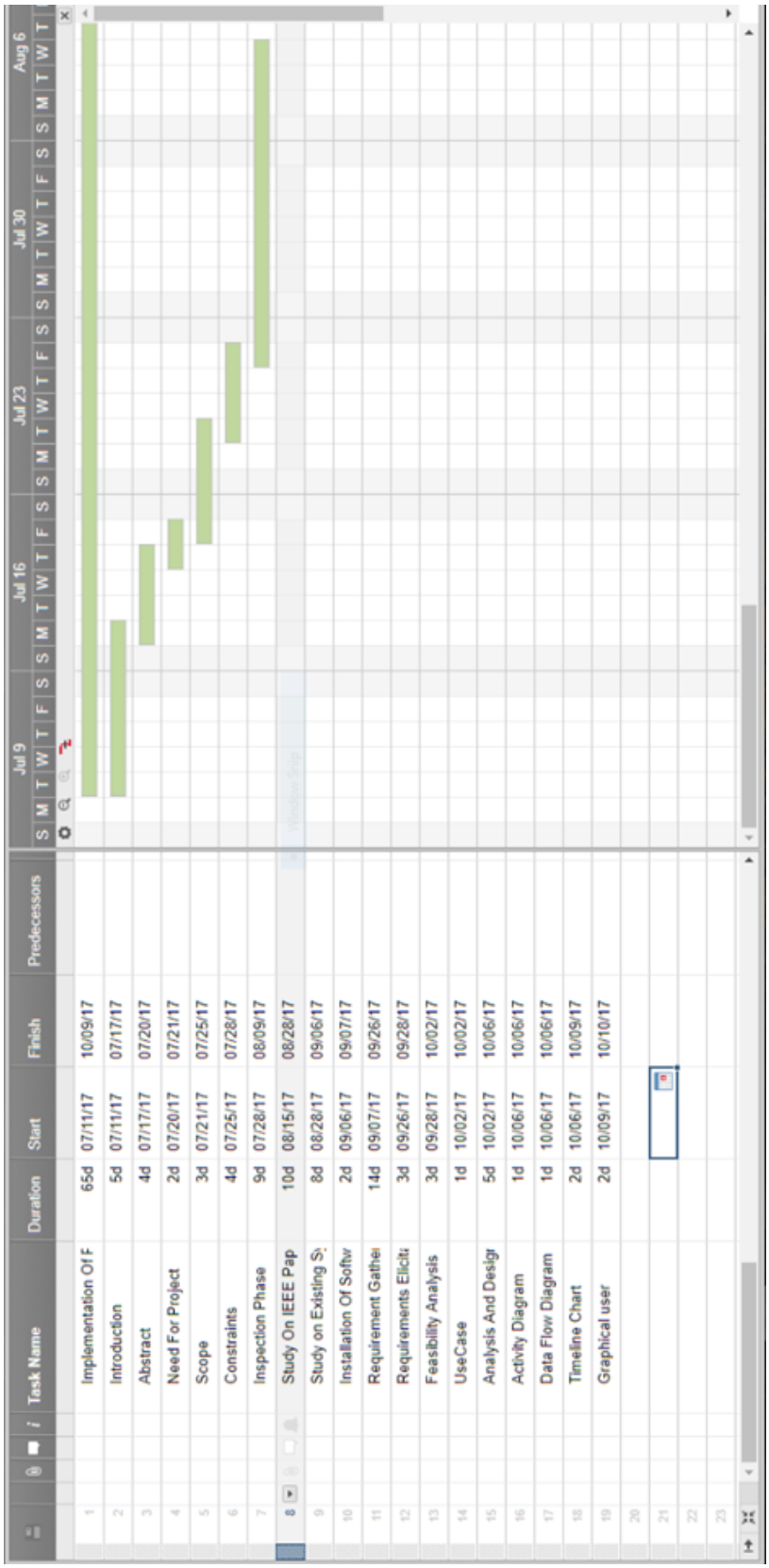

Figure 4.4 - Timeline Chart 1

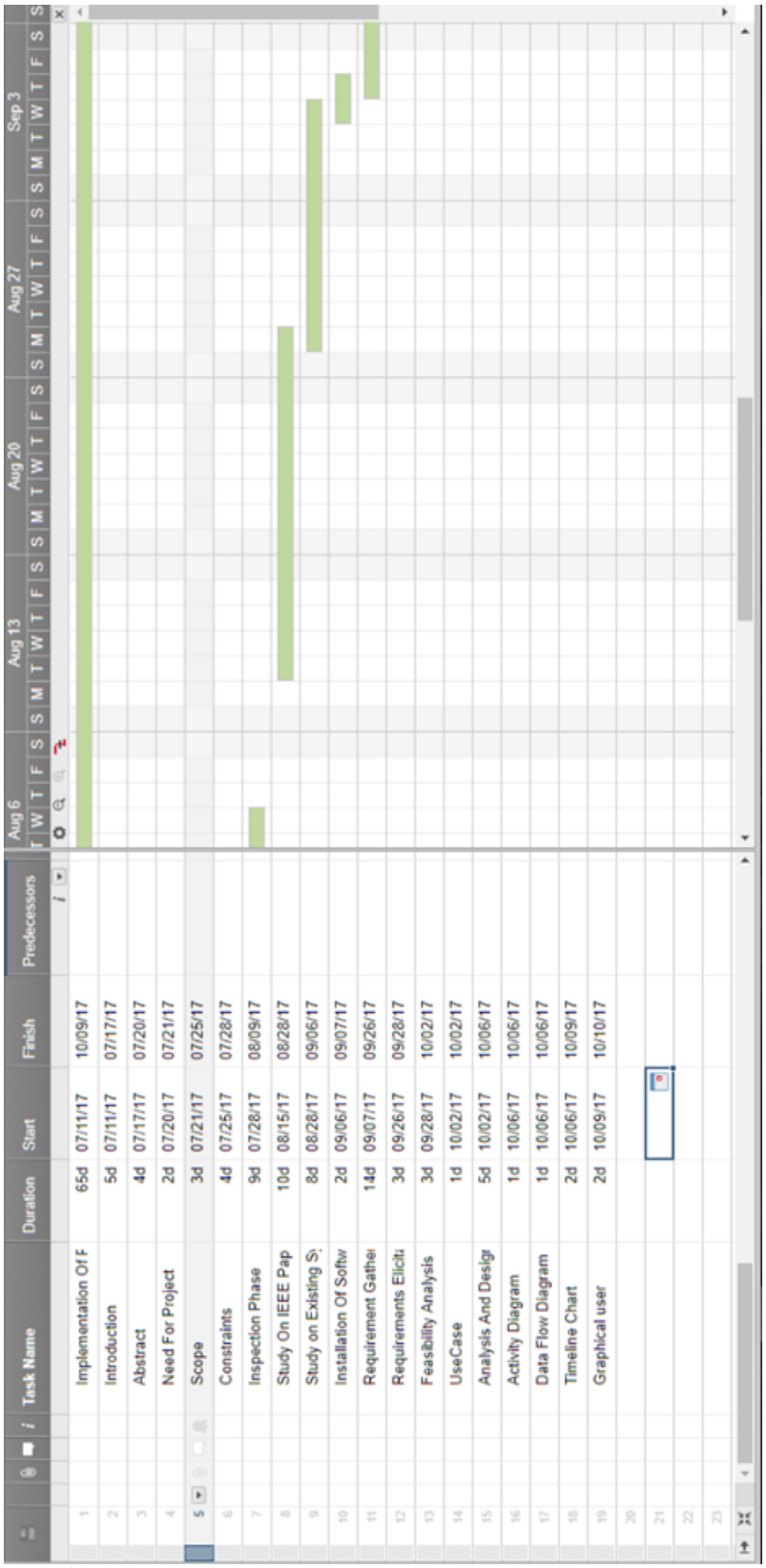

Figure 4.5 - Timeline Chart 2

Figure 4.6 - Timeline Chart 3

Chapter 5

Design

5.1 Block Diagram

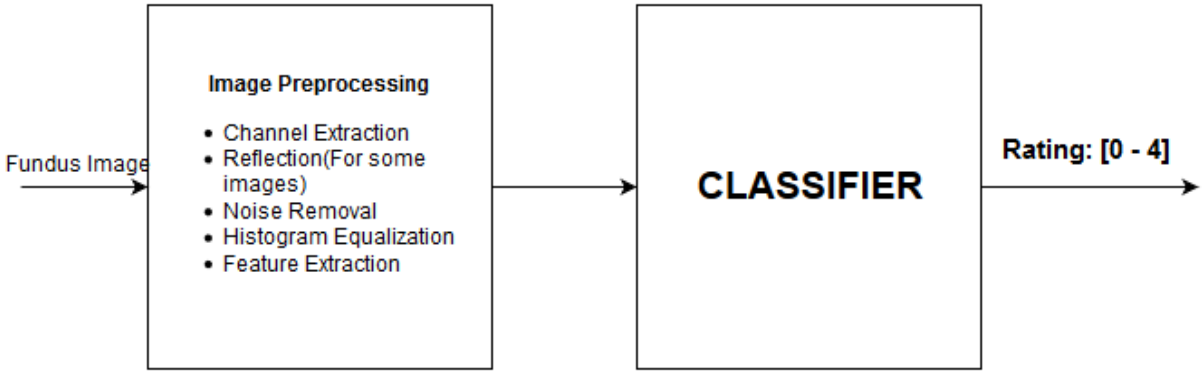

Figure 5.1 - Block Diagram

As the block diagram shows, the system consists of two major blocks, takes a fundus photograph as input and returns a rating between 0 – 4 as output. The input image is first preprocessed in the "Image Preprocessing" block. Some of the algorithms applied to the image in preprocessing step are:

- Channel Extraction
- Reflection
- Noise Removal
- Histogram Equalization
- Feature Extraction

The output of the preprocessing step will either be a processed and transformed image in case of a CNN or a set of features of the image in case of other ML techniques such as Naïve Bayes or SVM. The classifier will then either learn if it is training, or produce an output if it is running in the application.

5.2 User Interface Design

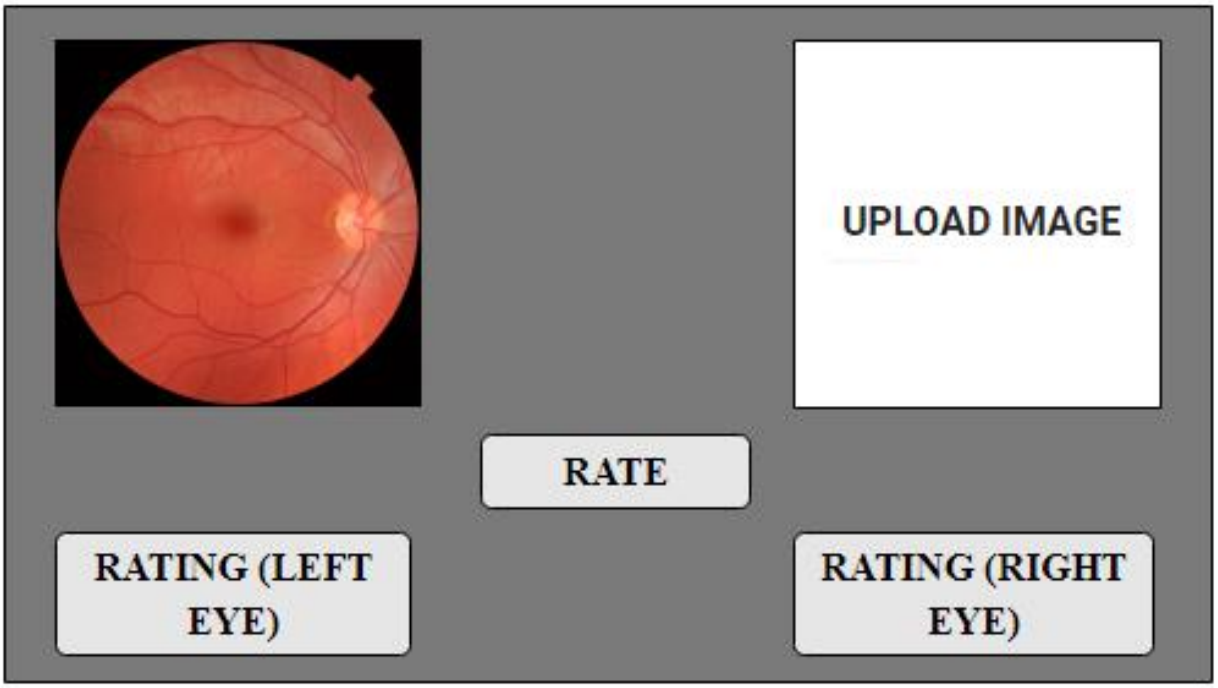

Figure 5.2 - User Interface Design

The user interface is going to be pretty simple. It'll contain all the actions which can be used in one screen.

The components of the screen will be

- Two input fields to input the left and right eye images
- A button to rate each image
- Text Fields to display the rating results.

Chapter 6

Conclusion and Future Scope

This project aims to build a comprehensive and automated system capable of rating color fundus photographs. The rating will be a value predicting how serious the spread of diabetic retinopathy is for the patient. Ratings range between 0-4 where 0 represents NO DR and 4 represents proliferative DR. The ratings will be needed to be accurate as possible to really make the system useful. The system will be trained on a dataset of ~35000 images which are all rated by a trained physician. The system also needs to be quick to warrant the use of such an automated system.

The project is meant to be used by in clinical where it can predict the rating of a patient quickly. In the future we hope to make the system more accurate which is a serious requirement of medical diagnosis systems. The model is also expected to be versatile with little or no changes required to make it suitable for similar medical applications. For example, the same system can be employed for brain tumor detection from MRI or X-Ray images. The system is currently independent from the microscope which is used to click retinal images. In the future the model might be able to be loaded on a microscope itself. The microscope itself will show the rating for the patient, thereby eliminating the need for a doctor to upload to a separate computer system.

Chapter 7

Literature Cited

- [1] Casanova, Ramon, Santiago Saldana, Emily Y. Chew, Ronald P. Danis, Craig M. Greven, and Walter T. Ambrosius. "Application of Random Forests Methods to Diabetic Retinopathy Classification Analyses." PLOS ONE 9, no. 6 (2014): 1-7. Accessed December 26, 2014. www.plosone.org.
- [2] Sinthanayothin, C., J.F. Boyce, T.H. Williamson, H.L. Cook, E. Mensah, S. Lal, and D. Usher. "Automated Detection of Diabetic Retinopathy on Digital Fundus Images." Diabetic Medicine 19 (2002): 105-12.
- [3] Usher, D., M. Dumskyjs, M. Himaga, T.H. Williamson, S. Nussey, and J. Boyce. "Automated Detection of Diabetic Retinopathy in Digital Retinal Images: A Tool for Diabetic Retinopathy Screening." Diabetic Medicine 21 (2003): 84-90.
- [4] Jaafar, Hussain F., Asoke K. Nandi, and Waleed Al-Nuaimy. "Automated Detection And Grading Of Hard Exudates From Retinal Fundus Images." 19th European Signal Processing Conference (EUSIPCO 2011), 2011, 66-70.
- [5] "National Diabetes Statistics Report, 2014." Centers for Disease Control and Prevention. January 1, 2014. Accessed December 26, 2014.
- [6] "Diabetes." World Health Organization. November 1, 2014. Accessed December 26, 2014. http://www.who.int/mediacentre/factsheets/fs312/en/.
- [7] “Convolutional Neural Networks for Diabetic Retinopathy” Harry Pratt, Procedia Computer Science.

Chapter 8

Acknowledgement

It has been a sincere desire of every individual to get an opportunity to express his views and skills, attitude and talent in which he is proficient so as to give him satisfaction and confidence to his ability to do or produce something useful for mankind.

The satisfaction that accompanies the successful completion of any task would be incomplete without acknowledging those who have made it possible and those who's constant encouragement and guidelines has been a source of inspiration throughout the course of this project.

We take this opportunity to express our gratitude towards our project guide Ms. Vincy Joseph, St. Francis Institute of Technology, Mumbai-103., for her constant encouragement, support, constructive criticism and guidance in our endeavor, without which we would have found it difficult to maintain our tempo and enthusiasm. Working with her has been a wonderful experience.

We also thank college authorities, who have through their meticulous planning created a comfortable co-existence of the project and college schedules.