Surface Defect Detection and Classification from Images

Prepared by: Muhammad Abdul Basit and Ali Rehman

April 30, 2023

Abstract

In this report, we compare the performance of four popular machine learning classification algorithms, Logistic Regression, Neural Network, Naive Bayes and SVM on our data-set; a collection of three different types of papers: papers with hole, paper with pen mark and papers with no defect. To make the data-set more comprehensible for our classifiers we pass it through a phase of some pre-processing and the mathematical formulation of each technique used in the pre-processing is described. We then present the results of our classifiers,including accuracy and performance metrics, and analyze the strengths and weaknesses of each algorithm. Our findings show that logistic regression and SVM are the two classifiers that standout among the four classifiers in terms of accuracy and other metrics such as F1 score, precision and recall.

1 Introduction

The paper reel classification problem involves categorizing images of paper reels into one of three classes: defect-free paper, paper with a hole, and paper with a pen mark. This problem has several practical applications, such as quality control in paper manufacturing or document scanning. In this report, we aim to evaluate the performance of four different classifiers: Naive Bayes, Neural Network, Logistic Regression, and Support Vector Machines (SVM) on the paper reel data-set.

To prepare the data-set for classification, we have applied several preprocessing steps, such as resizing the images, converting them to the HSV color

space, and defining color ranges for each of the three classes. Additionally, we have extracted spatial binning features, color histogram features, and HOG features for each image.

We trained each classifier on the extracted features and evaluated their performance using various metrics such as accuracy, precision, recall, and F1 score. Our goal is to compare the performance of each classifier on the paper reel data-set and provide insights into the strengths and weaknesses of each method. Furthermore, we aim to identify opportunities to improve the classification performance.

The remainder of the report is organized as follows. Section 2 provides mathematical formulation of the related concepts and techniques used in image classification and paper reel classification. Section 3 describes the four classifiers used in the study and their respective training and testing procedures. Section 4 presents the experimental results and discusses the performance of each classifier. Finally, Section 5 concludes the report and provides recommendations for future work.

2 Mathematical Formulation

The mathematical formulation of the techniques used in the preprocessing and feature extraction goes as follows:
Resizing:
Resizing is the process of changing the size of an image. In this case, the images are resized to 350x350.
Let X be the original image and Y be the resized image. The mathematical formulation for resizing is:

$Y(i, j) = X(\lfloor \frac{i}{s} \rfloor, \lfloor \frac{j}{s} \rfloor)$,

where s is the scaling factor, and i, j are the coordinates of the pixel.

Color Space Conversion:

Color space conversion is the process of converting an image from one color space to another. In this case, the images are converted from RGB color space to HSV color space. Let X be the original image and Y be the converted image. The mathematical formulation for color space conversion is:

$Y(i, j) = T(X(i, j))$,

where T is the color space transformation function:

$$T(R,G,B) = \begin{cases} \text{Undefined} & \text{if } R = G = B \ \max(R,G,B) \\ \text{if } B \leq G < R \ 2R - G - B & \text{if } R < G \ 2G - R - B \\ \text{if } G \leq R < B \ 2B - R - G & \text{if } B \leq R < G \end{cases}$$

where R, G, and B are the red, green, and blue color values of a pixel in the RGB color space. The output of this function is a vector of values in the HSV color space, which consists of hue, saturation, and value components.

Color Masking:

Color masking is the process of selecting specific colors in an image. In this case, three different color masks are applied to select paper with a hole, paper with pen marks, and defect-free paper. Let X be the original image, and M_i and M_o be the masks for the inside and outside of the desired color range, respectively. The mathematical formulation for color masking is:

$$M_i(i,j) = \begin{cases} 1, & \text{if } L_i \leq X(i,j) \leq U_i \\ 0, & \text{otherwise} \end{cases}$$

$$M_o(i,j) = \begin{cases} 0, & \text{if } L_i \leq X(i,j) \leq U_i \\ 1, & \text{otherwise} \end{cases}$$

where L_i and U_i are the lower and upper bounds of the desired color range for mask M_i.

Feature Extraction:

Feature extraction is the process of extracting relevant features from an image. In this case, three different types of features are extracted: spatial binning features, color histogram features, and HOG features. Let X be the input image, and F_s, F_h, and F_{hog} be the spatial binning features, color histogram features, and HOG features, respectively. The mathematical formulation for each type of feature extraction is:

Spatial Binning Features:

$F_s = Ravel(Resize(X, s))$

where $Resize(X, s)$ is the image resized to $s \times s$ pixels, and $Ravel()$ is a function that flattens the image into a 1D vector.

$Ravel(X) = [x_{1,1}, x_{1,2}, ..., x_{1,n}, x_{2,1}, x_{2,2}, ..., x_{m,n}]$

where X is an $m \times n$ matrix.

Color Histogram Features:

$F_h = Concat(Hist(X, M_i))$

where $Hist(X, M_i)$ is the color histogram of the image X using the mask M_i, and $Concat()$ is a function that concatenates the histograms of the three color channels.

HOG Features:

$F_{hog} = HOG(Gray(X), orient, ppc, cpb)$

where $Gray()$ is a function that converts the image to grayscale, $HOG()$ is a function

$Gray(X) = [g_{1,1}, g_{1,2}, ..., g_{1,n}, g_{2,1}, g_{2,2}, ..., g_{m,n}]$

where X is an RGB image, and $g_{i,j}$ is the grayscale value of the pixel at (i, j). The grayscale value can be computed as:

$g_{i,j} = 0.299 \times R_{i,j} + 0.587 \times G_{i,j} + 0.114 \times B_{i,j}$

where $R_{i,j}$, $G_{i,j}$, and $B_{i,j}$ are the red, green, and blue color values of the pixel at (i, j), respectively.

3 Use of Different Classification Algorithms

Here are brief explanations and mathematical formulations for Naive Bayes, Neural Network, Logistic Regression, and Support Vector Machine (SVM)

Logistic Regression: Logistic Regression is a statistical model that uses logistic function to model the probability of a binary outcome. The logistic function is a sigmoid function that maps any input value to a value between 0 and 1. The mathematical formulation for logistic regression can be expressed as follows:

Given an input feature vector x, the output of logistic regression can be calculated as:

$P(y = 1|x) = \frac{1}{1+e^{-(w^T x+b)}}$

where y is the binary output, w is the weight vector, b is the bias term, and e is the base of the natural logarithm. This sigmoid function is used in the classifier to compute the probability of the feature vector x belonging to that class. Then the label of the classifier, which returns the highest probability, is assigned to x. The reason for using the sigmoid function is that it provides us with a probability between 0 and 1, which along with our threshold probability helps us determine the real class of the feature vector x. It is also used because it is differentiable. This predicted probability is then compared to the actual outcome through a loss function that is also differentiable and convex.

The loss function calculates a numerical value that represents how far off the predicted probability is from the actual outcome. The goal is to mini-

mize this numerical value, which means that the predicted probability is as close as possible to the actual outcome. So, the sigmoid and loss functions work together in logistic regression to produce accurate predictions. The sigmoid function maps the inputs to a predicted probability, and the loss function measures how well that predicted probability matches the actual outcome, allowing us to adjust the model's parameters to improve the predictions' accuracy. The model parameters(weights and biases) are adjusted by a technique called gradient descent. By updating the model parameters in the direction of the steepest descent of the loss function, we can find the set of parameters that produce the smallest possible value of the loss function and make the class predictions as accurate as possible. The loss function is called 'cross-entropy loss.'

$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i) + \frac{\lambda}{2}||\mathbf{w}||_2^2$

The term with the lambda is an L2 regularization term that is incorporated in the loss function to prevent over-fitting and promote better generalization of the model to test data.

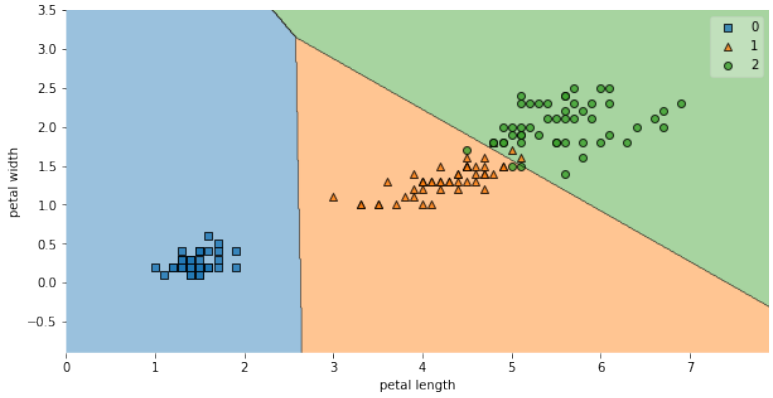

Figure 1: Logistic Regression.

Neural Network: Neural Networks are a type of machine-learning model inspired by the structure of the human brain. It consists of multiple layers of interconnected neurons(nodes are referred as neurons in nueral network), where each neuron performs a weighted sum of its inputs and passes the result through an activation function.

The output of each neuron in one layer becomes the input to the neurons in the next layer, and this process continues until the final output of the neural network is produced. The final output is compared to the true output (i.e., the correct answer) using a loss function, which measures how far off the neural network's prediction is from the true output.

The goal of training a neural network is to minimize the loss function by adjusting the weights and biases of the neurons. This is typically done using an optimization algorithm such as gradient descent, which iteratively adjusts the weights and biases to find the values that minimize the loss function.

The mathematical expression used in the classifier Neural Network is given below:

The calculation performed by each neuron:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

where $x_1, x_2, ..., x_n$ are the input values, $w_1, w_2, ..., w_n$ are the weights associated with those inputs, and b is the bias term.

The activation function:

$$a = f(z)$$

where $f(\cdot)$ is a non-linear function such as the sigmoid function:

$$f(z) = \frac{1}{1 + e^{-z}}$$

or the ReLU function:

$$f(z) = \max(0, z)$$

The output of a layer:

$$\mathbf{a}^{[l]} = f(\mathbf{z}^{[l]})$$

where $\mathbf{a}^{[l]}$ is the output of the l^{th} layer, $\mathbf{z}^{[l]}$ is the input to the l^{th} layer, and $f(\cdot)$ is the activation function.

Note: In our project, we have used the sigmoid function as our activation function

The loss function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

where m is the number of training examples, θ is the set of all weights and biases in the neural network, $\hat{y}^{(i)}$ is the predicted output for the i^{th} training example, $y^{(i)}$ is the true output for the i^{th} training example, and $L(\cdot, \cdot)$ is a loss function such as the mean squared error:

$$L(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

The optimization algorithm:

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$$

where θ_t is the set of weights and biases at time step t, α is the learning rate, and $\nabla J(\theta_t)$ is the gradient of the loss function with respect to θ_t. The optimization algorithm that iteratively adjusts the weights and biases to find the values that minimize the loss function. We use a method 'Back propagation' to implement chain rule for the computation of gradient.

If all of this seems too confusing or perplexing, refer to the following image to absorb the concepts of Neural Network in your 'neurons'.

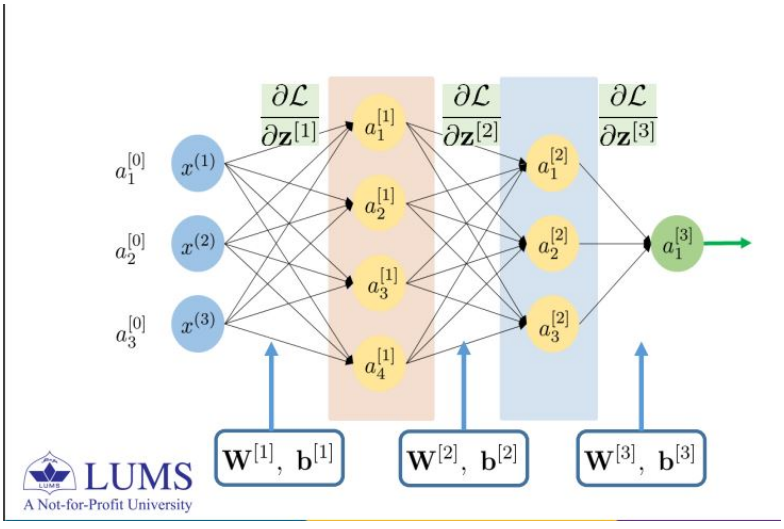

Figure 2: nueral network.

Naive Bayes:Naive Bayes is a probabilistic algorithm used for classification tasks. It is based on Bayes' theorem and assumes that the features of an input are conditionally independent given the class. This simplifying assumption allows for a computationally efficient model with relatively high accuracy, especially in cases where the amount of training data is limited.

The algorithm works by calculating the probability of each class given the input features, and then choosing the class with the highest probability as the predicted output. This calculation involves estimating the conditional probabilities of each feature given each class, and the prior probability of each class.

The mathematical formulation for Naive Bayes can be expressed as follows:

Bayes' Theorem:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \tag{1}$$

where:

$P(y|x)$ is the probability of the output class y given the input features x, $P(x|y)$ is the probability of observing the input features x given the output class y, $P(y)$ is the prior probability of the output class y, and $P(x)$ is the marginal probability of observing the input features x.

Naive Bayes assumption:

$$P(x|y) = \prod_{i=1}^{n} P(x_i|y) \tag{2}$$

where:

$P(x_i|y)$ is the probability of observing the i-th feature given the output class y, and n is the number of features. Maximum a posteriori (MAP) estimation:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\ P(y|x) = \underset{y}{\operatorname{argmax}}\ P(x|y)P(y) \tag{3}$$

where:

$\hat{y}$ is the predicted output class.

Maximum a posteriori (MAP) estimation's equation allows us to predict

the output class that is most likely given the input features. It does this by choosing the output class that maximizes the posterior probability (i.e., the probability of the output class given the input features). This is equivalent to choosing the output class that maximizes the product of the prior probability and the conditional probabilities of the features given the output class.

Laplace smoothing:

$$P(x_i|y) = \frac{N_{y,x_i} + \alpha}{N_y + \alpha n} \tag{4}$$

where:

N_{y,x_i} is the number of times the i-th feature appears in the training data with the output class y, N_y is the total number of training examples with the output class y, n is the number of features, and α is a smoothing parameter (usually set to 1) that prevents zero probabilities when a feature does not appear in the training data with a particular output class.

Support Vector Machines (SVM): Support Vector Machines (SVM) is a machine learning algorithm that can be used for classification or regression tasks. The main idea of SVM is to find the best decision boundary that separates the different classes of data points in the feature space.

There are several types of SVM that can be used depending on the nature of the data and the problem being addressed. The most commonly used types of SVM are:

Linear SVM: This type of SVM is used when the data can be linearly separated. It works by finding a hyperplane that best separates the classes in the feature space.

Nonlinear SVM: In cases where the data is not linearly separable, a nonlinear SVM can be used. This involves mapping the data to a higher-dimensional space where it may become linearly separable. The kernel trick is often used to accomplish this. Some common kernel functions include the polynomial kernel, radial basis function (RBF) kernel, and sigmoid kernel.

Multi-class SVM: When there are more than two classes to be classified, multi-class SVM can be used. One common approach is to train multiple binary classifiers and use a voting scheme to determine the final class.

The mathematical formulation for SVM is:

Given a set of training data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where x_i is the input feature vector and $y_i \in -1, 1$ is the output label, the objective of SVM is to find the hyperplane $w^T x + b = 0$ that maximizes the margin, where w is the weight vector and b is the bias term.

The margin is defined as the distance between the hyperplane and the closest data points from both classes. Let x_+ and x_- be the closest points to the hyperplane from the positive and negative classes, respectively. The margin is given by:

margin $= \frac{2}{|w|}$

The optimization problem for SVM is to find w and b that satisfy the following constraints:

$y_i(w^T x_i + b) \geq 1, \text{ for all } i = 1, \ldots, n$

The objective function of SVM is to maximize the margin subject to these constraints, which can be formulated as the following optimization problem:

minimize $\frac{1}{2}|w|^2$

subject to $y_i(w^T x_i + b) \geq 1, \text{ for all } i = 1, \ldots, n$

The Lagrange dual problem of this optimization problem can be solved using the quadratic programming algorithm, which gives us the values of the Lagrange multipliers α_i. The weight vector w and the bias term b can be computed as:

$w = \sum_{i=1}^{n} \alpha_i y_i x_i$

$b = \frac{1}{n_s} \sum_{i=1}^{n} (y_i - w^T x_i)$

where n_s is the number of support vectors.

Once the weight vector w and the bias term b are computed, the decision function of SVM for a new input vector x is:

$f(x) = \text{sign}(w^T x + b)$

where $\text{sign}(\cdot)$ is the sign function.

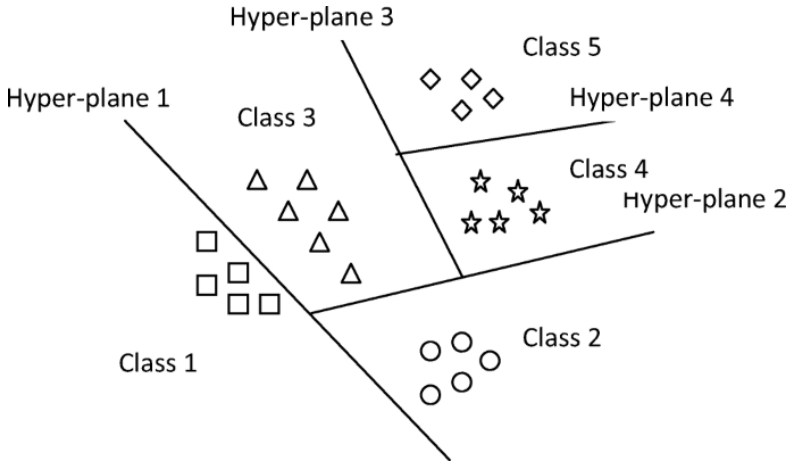

Figure 3: SVM.

4 Experimental results and analysis

We started off by running our dataset on the four classifiers mentioned above. At first, with a train test split of 70-30 respectively, we set the following parameters of each classifier and obtained the following results

Previous results:

Logistic Regression:
Max iterations = 50
Reported accuracy:34.88

Neural Network: layer size = 350
max iterations = 1000
reported accuracy: 66.66

SVM: kernel = 'linear'
regularization parameter(C) = 0.03
reported accuracy: 72.44

Naive bayes: reported accuracy: 68.44

However, we further tweaked the parameters of the classifiers and found out that a small change in the parameters of our data-set such as train test split had a large impact on not only the accuracies of each respective classifier but also on the time they took to run. Such as logistic and neural network regression whose time increased 3x to 4x by only a slight change of 5 percent in train test split ratio. Likely the for Neural network tweaking the hidden layer sizes and the max iterations had a great impact on the accuracies and classifier running time as well with a sharp decrease in the accuracy by an increase in both.

As far as SVM is concerned, there were two choices of running SVM: linear model and non linear polynomial model. On test and trial basis The linear model proved to more successful and accurate for us. Upon reflection on the results of both models, we reached the reason for such results which was our data being linearly separable and simple in which linear model far outperforms polynomial model which is designed for complex and non linearly separable data sets. Furthermore, the choice of C parameter also greatly impacted our results and a lot of test and trial made us reach the optimum value of C that gave us the best accuracy for the SVM linear model.Our final results in terms of all the analysis metrics : accuracies, f1s score, precision and recall for the four classifiers after adjusting parameters are:

Figure 4: classifiers' metrics.

After getting our finals results, we did the second part of the project which was to detect the defects in the defected images, find the location of defected area and draw boundary wall around it. This was accomplished via the rectangular box technique. An example of a successful boundary drawn on a test data sample is shown below.

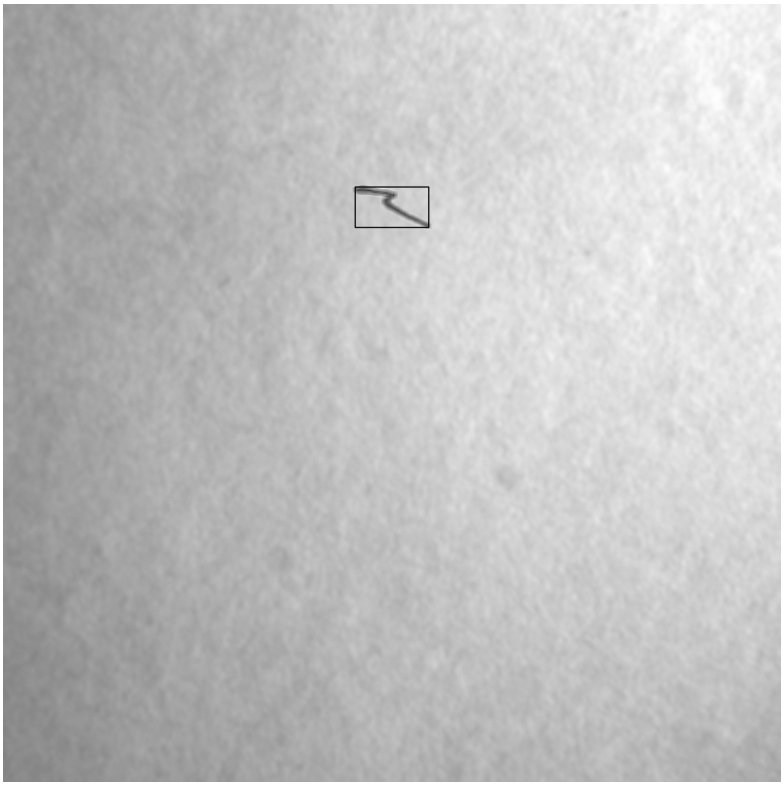

Figure 5: boundary drawn.

5 Conclusion

According to the accuracies reported above in our final results, it is clearly shown that the two classifiers that proved to be best in terms of their accuracies are Logistic Regression and SVM. Though both took a lot of time to run with our adjusted parameters, it is noteworthy, that the accuracy of logistic regression can still be improved as even with our current parameters the model's loss did not fully converge which meant it had more room to minimize itself and further get closer to the true predictions. Hence a better and improved accuracy. Moreover, since the parameters of the classifiers are set up on a test and trial basis, finding a more suitable approach to adjust the parameters can further improve the results.

Apart from this, As far as the second task of image detection and localization is concerned, due to time constraints we were unsuccessful in employing a very vital technique of 'rectangular window' to draw the boundary wall around the defected areas of the defected papers. A small steps involved in rectangular window are described below: Convert the paper image to a grayscale image.

1-Apply a thresholding algorithm to the gray-scale image to convert it into a binary image, where the pixels in the defected area have a value of 1 (white) and the pixels in the background have a value of 0 (black).

2-Use the rectangular window function to create a box around the defected area by setting the pixel values within the box to a specific value (e.g., 255 for white).

3-Apply a dilation algorithm to expand the box created by the rectangular window function, which will create a boundary wall around the defected area.

4- Finally, overlay the boundary wall image onto the original paper image to visually highlight the defected area.