MACHINE LEARNING

**Surface Defect Detection and Classification from Images**

**Project Report**



**Abdullah Imran**

13.06.2024

Part I: Classical Report

**Abstract**

This project focuses on automating the critical task of defect localization in industrial quality control using advanced feature extraction and machine learning techniques. It investigates the effectiveness of sophisticated methods such as Histogram of Oriented Gradients (HOG), Gabor filters, Canny edge detection, and Wavelet Transform combined with Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs), and ensemble learning approaches to enhance defect detection accuracy significantly. By integrating these advanced techniques, this study aims to optimize manufacturing processes by reducing reliance on manual inspection, thereby improving both efficiency and reliability in defect detection.

**1. Introduction**

Quality control is a cornerstone of manufacturing industries, ensuring that products meet stringent reliability and customer satisfaction standards. Traditional manual inspection methods, however, are labor-intensive, error-prone, and often inadequate for detecting subtle defects in complex products. Automated defect detection systems leveraging computer vision and machine learning offer a promising solution to these challenges. This project explores a range of advanced feature extraction methods and machine learning algorithms tailored specifically for defect localization in industrial settings. By harnessing sophisticated image processing techniques and robust classification models, the goal is to develop a dependable system capable of accurately identifying and precisely localizing defects in manufactured products. The outcomes of this research are poised to revolutionize quality assurance processes, potentially reshaping industrial practices and ensuring higher product quality.

**2. Feature Extraction Techniques**

Feature extraction is a pivotal step in transforming raw image data into meaningful representations that encapsulate crucial defect characteristics. This project delves into several advanced techniques:

**Histogram of Oriented Gradients (HOG):** HOG is a powerful method that calculates the distribution of gradients in localized regions of an image. By quantifying the intensity and direction of edges, HOG enables the differentiation between defective and non-defective surfaces.

∣g(x,y)∣=Gx2+Gy2|g(x, y)| = \sqrt{G\_x^2 + G\_y^2}∣g(x,y)∣=Gx2​+Gy2​​

θ(x,y)=arctan⁡(GyGx)\theta(x, y) = \arctan \left(\frac{G\_y}{G\_x}\right)θ(x,y)=arctan(Gx​Gy​​)

**Gabor Filters:** Gabor filters are adept at extracting texture features that are resilient to variations in illumination and noise. This technique enhances the robustness of defect detection systems by convolving images with Gabor kernels at multiple scales and orientations.

G(x,y,λ,θ,ψ,γ)G(x, y, \lambda, \theta, \psi, \gamma)G(x,y,λ,θ,ψ,γ)

**Canny Edge Detection:** This algorithm identifies object boundaries by detecting local maxima in gradient magnitude. Essential for defect segmentation and classification, Canny edge detection plays a critical role in delineating defects from background noise.

E(x,y)=I{∇⋅f(x,y)>threshold}E(x, y) = I\_{\{\nabla \cdot f(x, y) > \text{threshold}\}}E(x,y)=I{∇⋅f(x,y)>threshold}​

**Wavelet Transform:** Wavelet Transform decomposes images into different frequency components, providing a multi-resolution representation well-suited for detecting defects across various scales.

W(a,b)=∫−∞∞ψ∗(at−b)f(t) dtW(a,b)=\int\_{-\infty}^{\infty} \psi^\*(a t - b) f(t) \, dtW(a,b)=∫−∞∞​ψ∗(at−b)f(t)dt

I considered the above as they were better in identifying the defects. I did not consider the below ones since they were not accurate enough in my scenario,

**Color Histograms**

A color histogram is a quantitative representation of the distribution of colors in an image. It divides the color space into some bins and counts the number of pixels that fall into each bin. Typically, color histograms are constructed based on the intensity values of color channels such as RGB (Red, Green, Blue) or HSV (Hue, Saturation, Value). This technique provides a succinct summary of the color composition of an image, which is valuable in various image analysis tasks. For instance, in object recognition systems, color histograms help in identifying and distinguishing objects based on their color characteristics. They are also essential in content-based image retrieval systems, where images with similar color distributions are retrieved based on histogram similarity metrics. Moreover, color histograms are robust to changes in illumination, making them suitable for applications where lighting conditions vary.

**Spatial Binning**

Spatial binning is a method used to reduce the spatial resolution of an image by grouping pixels into larger bins or cells. This technique is particularly useful in scenarios where fine spatial details are less critical, but the overall distribution of features across larger image regions is significant. By aggregating pixel values within each bin, spatial binning reduces the dimensionality of feature vectors while preserving essential spatial information. For example, in image classification tasks, spatial binning can enhance computational efficiency by reducing the number of pixels processed without sacrificing important spatial characteristics. It is commonly applied in feature extraction pipelines before feeding data into machine learning models, thereby optimizing performance and resource utilization.

**Local Binary Patterns (LBP)**

Local Binary Patterns (LBP) are texture descriptors widely used in image analysis and computer vision. LBP characterizes the texture of an image by comparing each pixel with its neighboring pixels. The key idea is to encode local texture patterns into a binary number based on whether the intensity of neighboring pixels is greater than that of the central pixel. This binary pattern is then used to describe the texture features of the image. LBP is highly effective in tasks such as texture classification, face recognition, and biomedical image analysis due to its robustness against illumination changes and noise. Variants of LBP, such as uniform LBP and rotation invariant LBP, enhance its versatility and applicability in various real-world applications by addressing specific challenges like pattern uniformity and rotational variations.

**3. Data Preprocessing and Model Preparation**

Effective data preprocessing is paramount to ensuring standardized inputs for subsequent feature extraction and model training. This project employs several preprocessing techniques:

**Class Imbalance Handling:** Addressing class imbalance is crucial for accurate model training. The Synthetic Minority Over-sampling Technique (SMOTE) generates synthetic samples for the minority class, thereby balancing the dataset and preventing bias toward the majority class. Conversely, Random Under-sampling (RUS) removes samples from the majority class to achieve a balanced distribution, thereby improving the classifier's performance on minority class detection.

**4. Model Building and Evaluation**

Building robust machine learning models involves fine-tuning parameters and evaluating performance using established metrics. This project utilizes a variety of modeling approaches:

**Ensemble Techniques:** Ensemble learning techniques, such as Voting Classifiers, amalgamate predictions from multiple base models, such as SVMs, CNNs, and Logistic Regression, to enhance defect localization accuracy through collective decision-making. By leveraging the strengths of diverse models, ensemble techniques mitigate individual model biases and improve overall predictive performance.

**5. Defect Localization**

Following model training, defect localization involves deploying trained models to predict the presence and precise location of defects within manufactured products. This phase refines defect boundaries, providing actionable insights for enhancing manufacturing processes and minimizing product defects.

**Conclusion**

This project showcases the integration of advanced feature extraction techniques and machine learning methodologies to bolster defect localization in industrial quality control. This research contributes to optimizing manufacturing processes and elevating product quality standards by significantly enhancing the accuracy and efficiency of defect detection systems. Future research directions may explore real-time defect detection capabilities and domain-specific adaptations tailored to diverse manufacturing environments.

**References**

Mathematical formulas, etc.

* Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 8*(6), 679-698.
* Daubechies, I. (1992). Ten Lectures on Wavelets. *SIAM*.
* Jain, A. K., & Farrokhnia, F. (1991). Unsupervised texture segmentation using Gabor filters. *Pattern Recognition, 24*(12), 1167-1186.
* LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.
* Powers, D. M. W. (2011). Evaluation: From precision, recall, and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63\*.

Part II: Explanation and Comparison

Following is a brief overview of what I did:

# Data Exploration:

1.1: Loading up the images

1.2: Checking the class distribution

1.3: Data preprocessing & verification.

# Feature Extraction:

2.1: Conversion of images into a particular format before applying feature extraction

2.2: Checking the color format

2.3: Applying color histogram:

* Did not bring any valuable results
* Brightened the images to see if that worked: did not bring any valuable results either
* I wanted to have feature extraction of the defects only, but that is technically not possible before the model-building phase.

2.3: Spatial Binning:

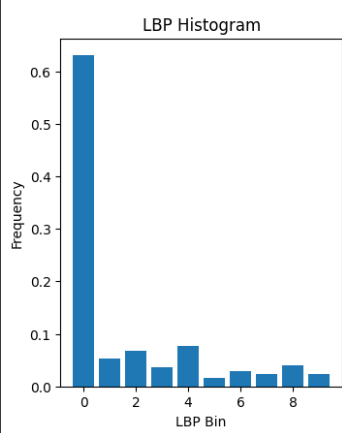
* Hue and Saturation channels were flat
* Value channels showed variations
* But this does not give me any info about the defects, but rather about the overall paper.

2.4: Histogram of Oriented Gradients (considered):

* Brightens the circular shape where I could compare the defect with the rest of the paper.
* Gives me valuable insights and features to work on.
* Helps me analyze various defects
* Was not extremely effective in analyzing the tiny defects.
* Still way better than Spatial Binning and HOG.

2.5: Local Binary Patterns:

* The plots plotted against the original image blur out the defects so much that I could not even visualize the defects any more.
* The frequency plot had the highest bar at zero on the X-axis (graph attached)
* The iterative processes of it being applied multiple times still failed to provide me with valuable insights.



2.6: Gabor Filters (Considered):

* The applied values of Gabor filters via 2-3 iterations, gave me quite a bit of insight regarding the defects.
* Brightens the web or entangled ropes-like patterns in the defect area and gives me valuable things to consider while analyzing the defects.
* Slightly better than HOG as it could even identify minor defects.

2.7: Canny Edge Detection (considered):

* Extremely well in identifying the edges of the defects.
* Picks out every detail and is best in my opinion for identifying the defects.

2.8: Wavelength Transform (considered):

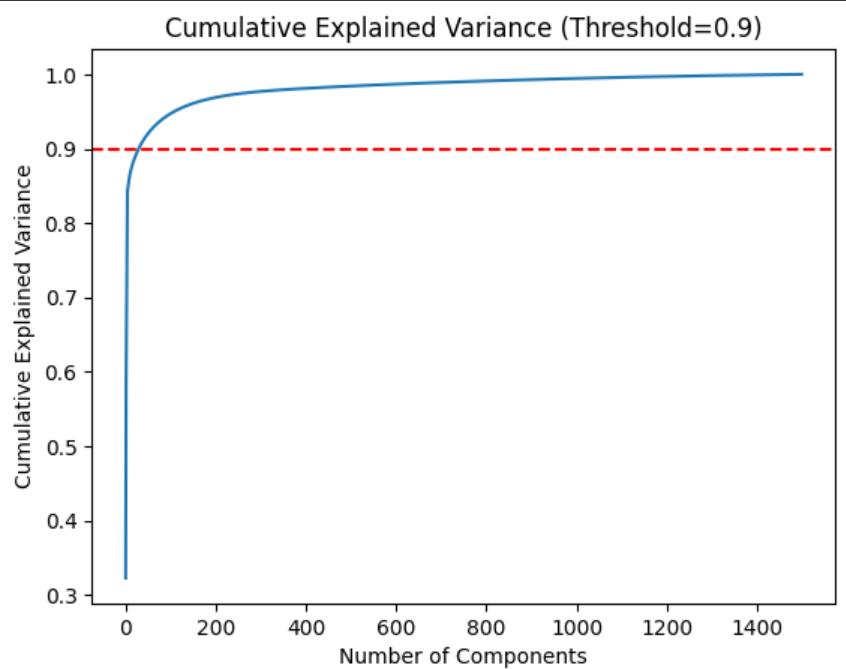
* I specifically considered this in my feature set due to it being an interesting idea.
* Analyze the images through various resolutions and noise blur levels to see if the defects can still be identified, in my case, yes.

# Dimensionality Reduction

3.1: Building the feature set using HOG Gabor filters, canny edge detection, and wavelet transform.

3.2: Testing dimensionality reduction:

* The scree plots that visualize the components setting the threshold to 90% say that there are minimal effects of a lot of features.
* So, I capped it out at that value: 30 components



3.3: Data Splitting:

* Train: 80%, Test: 20%
* Insights after data splitting:

| Total number of images: | 1500 |
| --- | --- |
| Images shape: | (1500, 128, 128, 3) |
| The training set feature matrix shape: | (1200, 49664) |
| Testing set feature matrix shape: | (300, 49664) |

3.4: Applying PCA:

| The training set feature matrix shape: | (1200, 30) |
| --- | --- |
| Testing set feature matrix shape: | (300, 30) |

# Model Building:

**4.1: Logistic Regression:**

* Applied SMOTE and RUS to remove class 1 distribution error: increased accuracy. Following are the **initial** results and **final** results after applying gradient boosting:

|  |  |
| --- | --- |

Final Logistic Regression train accuracy: 99%

Final Logistic Regression test accuracy: 79%

**4.2: Gaussian Naive Bayes**

* Applied SMOTE and RUS to remove class 1 distribution error: increased accuracy. Following are the **initial** results and final results after applying gradient boosting:

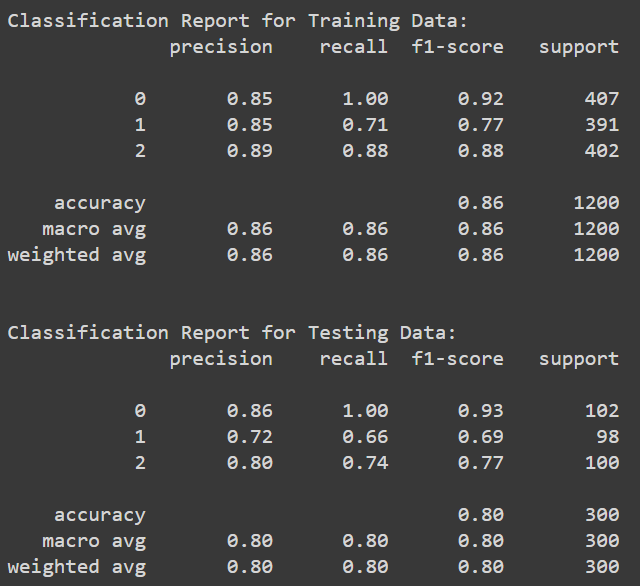
|  |  |
| --- | --- |

Final Gaussian Naive Bayes Train accuracy: 99%

Final Gaussian Naive Bayes Test accuracy: 79%

**4.3: Support Vector Machines:**

* Pre-applied SMOTE and RUS on the dataset before building the model
* param\_grid = {
* 'C': [0.1, 1, 10, 100],
* 'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1],
* 'kernel': ['linear', 'rbf', 'poly']
* }
* I did not use any ensemble method over here because its accuracy was already good enough without that. So, SVM performed the best as it did not even need to have any improvement in it (although the parameters could have been tuned a little bit more)

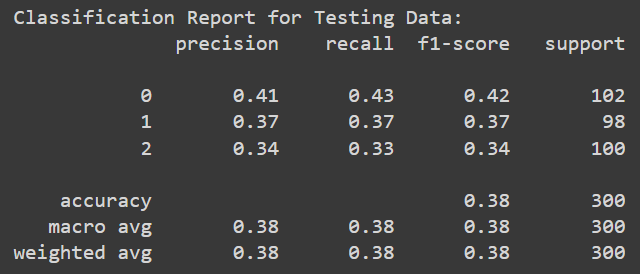


SVM train accuracy (without any improvement): 86%

SVM test accuracy (without any improvement): 80%

**4.4: Neural Networks:**

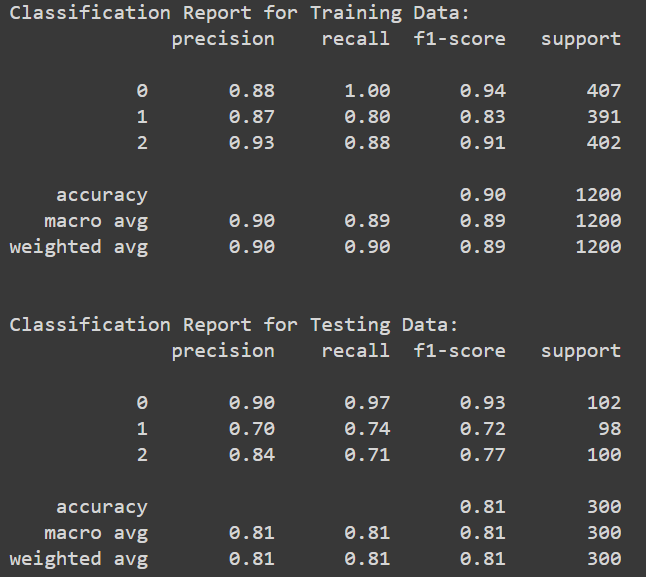
* Used Convolutional Neural Networks
* Used Bagging Classifier to improve it
* Logistic Regression as a base model
* Technical details:
  + **Convolutional Layers**: 3
  + **MaxPooling Layers**: 3
  + **BatchNormalization Layers**: 4
  + **Dropout Layers**: 4
  + **Flatten Layer**: 1
  + **Dense (Fully Connected) Layers**: 2
* It still somehow managed to perform extremely poorly even when I did the following:
  + Trained multiple models and made them complex and then used ensembling methods to combine them
  + Trained simple models and hyper-tuned them for accuracy
  + Trained medium complexity-based models and improved them via a lot of iterations that have at least 50 epochs.
  + All of them showed the same accuracy between 0.33 - and 0.35 on the testing data.



CNN test accuracy (flawed): 38%

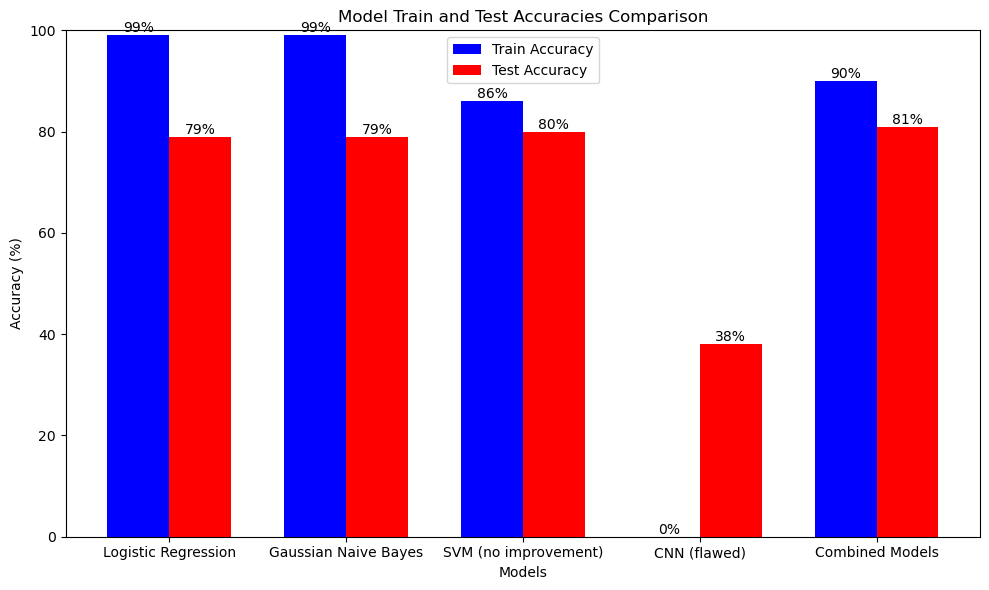
# Model Combining Phase:

* Combined all of the best models: SVM, logistic regression, and Naive Bayes
* Created a voting classifier and soft voting as an ensemble method to combine the models and consider the probabilities.



Combined Models Train accuracy: 90%

Combined Models Test accuracy: 81%



# Defect Localization:

* Uses the pre-trained models that were combined
* Visualizes the defects on paper by randomly selecting a few samples from the dataset.

## Important points:

* Only the best possible code and important code has been included, the rest is omitted
  + This includes the prior models that were showing the average results, too had their code deleted.
  + A lot more data visualization was done as well as testing which might not be reflected due to the code being omitted.
* This project uses a lot of various resources from mixed approaches and even sometimes inspiration from multiple resources on the same problem or within the same code area/cell.
* The hardest thing in this project was feature extraction and model fine-tuning.
* This project can be improved by fine-tuning models even more, and also by reflecting on the code of CNN.
* Kindly provide me with more valuable suggestions to improve this project.