

Strength of Rice Type Sorting Based Texture, Length, and Image Size

Jonah Epstein, Landis Humphrey, Vilcina Laplante, Menghan Wang

¹ Champlain College, Burlington VT 05401, USA

Abstract. This report explores how tree depth affects classification algorithms in machine learning. We look at factors like training time, memory use, complexity, and accuracy. We analyze the datasets of Rice. For methods, we use HOG, Edge Detection, and Gabor Texture Analysis. Classification methods include SVM, Decision Trees, Random Forest, and Neural Networks. SVM with HOG performed best, followed by Decision Trees with HOG. Random Forest showed better accuracy than Decision Trees. Neural Networks with Gabor, HOG, and Sobel filters had high accuracy. Overall, feature extraction like HOG proved more effective than edge detection, with HOG filters and SVM achieving the highest scores.

Keywords: Rice Image, Image Classification, Feature Extraction.

1 Introduction

Examining and elaborating on the influence of tree depth within classification algorithms in machine learning, this discussion aims to explore how varying the depth impacts several critical dimensions. These dimensions include the time required for model training and prediction, the space or memory consumption of the model, the computational complexity required, and the overall accuracy and predictive performance of the model. Specifically, the focus will be on understanding the nuanced trade-offs between enhancing model complexity through deeper trees against the potential for increased computational demands and overfitting, and how these factors interact to affect the efficiency and effectiveness of classification tasks in machine learning environments.

1.1 Literature Review

As part of our machine learning course, our group embarked on a journey to identify and analyze image datasets with potential applications in various industries. The selection process involved detailed discussions of each dataset's characteristics and potential utility.

Acne, Computer Vision Dataset: Available on Kaggle, this dataset comprises over 2,000 images of human faces with diverse skin conditions, meticulously labeled to identify various types of acne and skin imperfections. The images vary significantly in terms of lighting, background, and facial expressions, presenting a complex chal-

lenge for any facial recognition algorithm. The potential applications of this dataset extend beyond medical diagnostics to include enhancements in the beauty and photography industries, where precision in identifying and retouching skin imperfections can significantly enhance product and service quality.

Fruit Quality Classification Dataset: Hosted on Kaggle this dataset features almost 20,000 images of fruits captured from different angles. Each image is tagged with labels indicating various quality parameters such as ripeness, color uniformity, and presence of visual defects. This dataset poses unique challenges due to the variability in fruit appearance based on angle, maturity, and type, making it an excellent candidate for developing robust automated quality control systems for the agricultural and retail sectors. By automating the quality assessment process, such systems aim to improve consistency and reduce waste in food production and distribution.

Rice Image Dataset:

Ultimately, after thorough consideration, we selected the Rice Image Dataset for our project. This dataset, featuring 75,000 images across five different rice varieties (Arborio, Basmati, Ipsilon, Jasmine, and Karacadağ), offers an extensive range of images categorized by 12 morphological features, 4 shape descriptors, and 90 color attributes. The comprehensive nature of this dataset allows for deep exploration into the effectiveness of various preprocessing techniques that utilize morphological classification—a method pivotal for enhancing the signal-to-noise ratio in image processing.

Morphological classification in this context involves techniques that focus on structural elements within the images to improve the clarity and relevance of the data fed into classification algorithms. This approach is crucial for minimizing noise and enhancing the accuracy of subsequent analyses.

2 Methods / Algorithms

2.1 Data Preparation

2.2 Feature Extraction

The 3 techniques that I found helpful for our algorithms were HOG, Edge Detection and Gabor Texture Analysis. Each of these techniques analyze the unique features of an image that when passing through an algorithm can help with training for model strength and accuracy.

HOG or Histogram of Oriented Gradients is a feature descriptor great for image processing or image classification by capturing the shape and appearance of the object in an image into an array of numeric values. Our rice images were pictures of a grain with a black background. With most of the image being black, most of the hog array was also 0 because there is no gradient of color at the point of the image axes. The other values in the array are the points on the image axis where the rice grain is present, giving us edges and overall object detection.

Edge Detection is another technique that was used because with each image in our dataset only being one grain per image, it was a way to get an overall understand-

ing of what shape and unique cuts the grain would have. The advantage of this technique is it will result with a series of unique features for each rice type so when testing comes it will help with the model accuracy for each rice type because it returns the object shape. Additionally, to complete edge detection you can either use Sobel or Canny. Sobel allows you to see both the x axis edge detection and y axis edge detection individually or look at XY edge detection together. I found that being able to see both axis edge detection allowed us to understand the placement of the rice in the image in a computer vision view. Canny, which was the other function that allowed for edge detection. We prefer this function because it additionally returns an array of the edge detection and the computer vision version. The Sobel version didn't capture all the edges of the rice in the image compared to the Canny version.

Gabor texture analysis was the most interesting feature extraction technique because it was one that we never worked with before and allows us to see other possibilities that image data can be returned in an array. It works by taking the training data of x and y, testing data of x with Frequency and Thetas. The frequency is the number of oscillations (movement back and forth at a regular speed) within the filter and theta is the orientation of the filters. The features extraction is data that can be used to characterize the image and provide a representation of the texture properties at different scales and orientation.

2.3 Classification Methods

Support Vector Machine

Support Vector Machine is a supervised machine learning algorithm that is used for classification. It's a common use algorithm because it is effective for high dimensional data even in cases where the number of dimensions is greater than the number of samples, and versatile because it has different kernel functions that can be specific for decision function. The kernels allow the data to be linear or non-linear, but it has a chance of overfitting data. The Kernel's functions that are available are Linear, Polynomial, Radius Bias or Gaussian and Sigmoid. Linear Kernel uses the equation seen in Figure 2 and the simplest kernel, it represents the linear decision boundary and works best when the data is linear separable. Polynomial Kernel maps the data into higher-dimensional spaces using a polynomial function see in Figure 1 and is great for when the decision boundary is non-linear but still smooth. Radial Bias Kernel transforms the data into infinite-dimensional space and is effective for capturing complex non-linear decision boundaries when there is no prior knowledge of the data distribution. Radial Bias completes this process by using the equation found in Figure 3. Lastly, Sigmoid Kernel is useful for binary classification and completes this using the equation found in figure 4.

$$K(x_i, x_j) = \tanh(\alpha x_i^T \cdot x_j + c)$$

Fig. 1. Sigmoid Function Kernel

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Fig. 2. Radial Bias Kernel Function

$$K(x_i, x_j) = x_i^T \cdot x_j$$

Fig. 3. Linear Kernel Function

$$K(x_i, x_j) = (x_i^T \cdot x_j + c)^d$$

Fig. 4. Polynomial Kernel Function

Decision Trees

Decision Trees are a useful and versatile classification method. It is best used in supervised learning, where the data already has labels that can be used during the training process of the algorithm. It utilizes a hierarchical tree-like structure, built upon nodes. Each node represents a point where the data splits on a value from a single feature. It looks at the data based on the criteria specified at that node and divides the data accordingly. Each feature can only be used as decision criteria once to prevent an overdependence on a select number of features. Because of this, however, the more features a dataset has, the more complex the decision tree is.

With our images, each image got processed into a numerical format. This means each image either was saved as grayscale pixel values, or a series of values generated from our feature extraction techniques. We then passed this data into our Decision Tree Classifier as the independent variable, using the labels (name of the rice of the select image) as the dependent variable, or in other words, the classes/types that the tree will be predicting the rice grain belongs to.

Neuron Network

The model starts by looking at the images to find basic shapes using a filter called Conv2D with 32 pieces that are 3x3 in size. This filter helps it see simple things in the

pictures. Then, it uses a MaxPooling2D layer to make the picture simpler and smaller, making it easier for the model to understand. This process happens again, but now with more complicated filters: one with 64 pieces and another with 128 pieces, each followed by MaxPooling2D. This makes the model see more and more complex things in the pictures. After that, the model flattens the information, getting it ready to decide what type of rice it is seeing. A layer with 128 parts helps it make sense of this information, and a dropout layer with a 50% rate helps prevent it from thinking too much about the same things and overfitting. Finally, there's a layer with 5 parts that says which of the five types of rice the model thinks it is.

Random Forest

A Random Forest algorithm is an "ensemble" method that works by training and combining the outputs from multiple Decision Trees. Each of these decision trees are trained using random subsets of the training data, and the output is obtained by finding the majority output for all the decision trees together. Because Random Forests works as a group of Decision Trees, the preprocessing for the images was the same as for the Decision Tree Classifiers.

3 Experiments/Results

3.1 SVM

To run a support vector machine algorithm the image data is converted to an array. Without a feature extraction method, it would be the grayscale pixel values for each image. However, since we ran it through feature extraction methods, each value in the array contained the results of each feature extraction method ran on the image. The labels of each rice image were also saved in a separate array. Looking at Gabor Feature Extraction data, it collected the texture of the image, making it great for rice due to the different textures that can be picked up from the image. The SVM model that produced the best results was the Linear Model kernel. The labels are connected to the image array and they can say that the relationship between image array x and labels y are linearly related. Looking at HOG Feature extraction, it is the best SVM model overall but we can notice that every kernel type returns similar values. This set of SVM algorithm results of hog feature extraction raise questions about the impact of the kernels while the other feature extraction methods.

| SVM Algorithm Gabor Feature Extraction | | | | |
|--|--|--|--|--|
| Kernel Type | Linear | Poly | RBF | Sigmoid |
| Confusion Matrix | <code>[[295 0 4 1 0] [1298 1 0 0] [6 4281 0 9] [3 0 0297 0] [4 0 20 0 276]]</code> | <code>[[272 0 15 9 4] [1299 0 0 0] [6 14241 1 38] [5 0 3292 0] [11 1 20 0 268]]</code> | <code>[[270 0 14 10 6] [1299 0 0 0] [6 13234 1 46] [6 0 2292 0] [9 1 19 0 271]]</code> | <code>[[0300 0 0 0] [0 2298 0 0] [0274 26 0 0] [0300 0 0 0] [0287 13 0 0]]</code> |
| Accuracy | 0.96 | 0.91 | 0.91 | 0.02 |
| Precision | 1 | 1 | 1 | 0 |
| Recall | 0.96 | 0.91 | 0.91 | 0.02 |
| F1 Score | 0.96 | 0.91 | 0.91 | 0.02 |

Fig. 5. SVM Algorithm Gabor Feature Extraction

| SVM Algorithm HOG Feature Extraction | | | | |
|--------------------------------------|--|---|---|--|
| Kernel Type | Linear | Poly | RBF | Sigmoid |
| Confusion Matrix | <code>[[289 0 1 10 0] [2298 0 0 0] [0 0293 0 7] [4 0 0296 0] [0 0 6 0 294]]</code> | <code>[[290 0 1 9 0] [1297 2 0 0] [0 0297 0 3] [4 0 0296 0] [0 0 6 0 294]]</code> | <code>[[290 0 1 9 0] [1297 2 0 0] [0 0296 0 4] [4 0 0296 0] [0 0 6 0 294]]</code> | <code>[[284 0 2 14 0] [2298 0 0 0] [0 0293 0 7] [4 0 0296 0] [0 0 6 0 294]]</code> |
| Accuracy | 0.98 | 0.98 | 0.98 | 0.98 |
| Precision | 0.98 | 0.98 | 0.98 | 0.98 |
| Recall | 0.98 | 0.98 | 0.98 | 0.98 |
| F1 Score | 0.98 | 0.98 | 0.98 | 0.98 |

Fig. 6. SVM Algorithm HOG Feature Extraction

3.2 Decision Tree

For the Decision Tree Algorithm, I had to convert the image data into an array. Without a feature extraction method, it would be the grayscale pixel values for each image. However, since we ran it through feature extraction methods, each value in the array contained the results of each feature extraction method ran on the image. The labels of each rice image were also saved in a separate array. Once these processes were complete, we fit the Decision Tree Classifier on the array of values from feature extraction and the label names. We created predictions based on the decision tree and compared it to the actual test data, yielding our evaluation metrics. From running the data, the

HOG filter had the best prediction accuracy while Canny Edge Detection had the worst. Sobel and Canny, being our edge detection methods, performed the worst overall, yielding the lowest score on the evaluation metrics. I think this is because some of the rice types shared strong similarities on the perimeter, so edge detection was more likely to make a wrong prediction. Apart from Canny, all the feature extraction methods overall predicted our data well.

| | Gabor | Sobel (x-axis) | Sobel (y-axis) |
|------------------|---|---|---|
| Confusion matrix | [237 0 18 30 15] [0 295 4 0 1] [15 3 264 4 14] [18 0 5 277 0] [17 1 23 1 258] | [206 9 23 58 4] [4 267 12 4 13] [20 8 235 6 31] [21 4 3 270 2] [3 5 38 0 254] | [225 6 18 48 3] [6 265 14 4 11] [19 11 230 3 37] [46 1 5 246 2] [5 9 51 2 233] |
| Accuracy | 0.89 | 0.82 | 0.8 |
| Precision | 0.89 | 0.82 | 0.8 |
| Recall | 0.89 | 0.82 | 0.8 |
| F1 score | 0.89 | 0.82 | 0.8 |

Fig. 7. Decision Tree Algorithm Results Part 1

| | Sobel (x & y axes) | Canny | HOG |
|------------------|--|---|---|
| Confusion matrix | [225 4 32 33 6] [10 253 15 2 20] [32 7 208 7 46] [38 0 3 257 2] [10 15 49 1 225] | [147 14 29 99 11] [14 248 1 34 3] [39 13 133 20 95] [94 21 5 178 2] [23 5 110 10 152] | [258 2 16 23 1] [4 291 0 0 5] [10 4 265 0 21] [19 0 0 280 1] [2 0 23 0 275] |
| Accuracy | 0.78 | 0.57 | 0.91 |
| Precision | 0.78 | 0.57 | 0.91 |
| Recall | 0.78 | 0.57 | 0.91 |
| F1 score | 0.78 | 0.57 | 0.91 |

Fig. 8. Decision Tree Algorithm Results Part 2

3.3 Random Forest

The comparative results for Random Forest bore resemblance to the results for Decision trees, while also giving greater accuracy and precision than Decision Trees across the board. Given the nature of Random Forest as an ensemble of Decision

Trees, this was somewhat expected. Canny edge detection comparatively performed the worst. Sobel x-axis edge detection yielded the best results compared to the other Sobel features. Curiously, Sobel x-axis edge detection also gave the best results overall—this differs from Decision Trees, in which Sobel was outcompeted by HOG.

| | Gabor | Sobel (x-axis) | Sobel (y-axis) |
|------------------|--|---|---|
| Confusion matrix | [275 0 8 17 0] [1 298 1 0 0] [0 3 287 0 10] [12 0 0 288 0] [0 0 5 0 295] | [283 0 4 13 0] [1 299 0 0 0] [1 4 281 0 14] [9 0 0 291 0] [0 0 7 0 293] | [283 0 3 14 0] [1 298 0 0 1] [0 7 271 0 22] [7 0 0 293 0] [0 0 8 0 292] |
| Accuracy | 0.962 | 0.965 | 0.958 |
| Precision | 0.962 | 0.965 | 0.958 |
| Recall | 0.962 | 0.965 | 0.958 |
| F1 score | 0.962 | 0.965 | 0.958 |

Fig. 9. Random Forest Algorithm Results Part 1

| | Sobel (x & y axes) | Canny | HOG |
|------------------|---|---|--|
| Confusion matrix | [285 0 3 12 0] [2 297 0 0 1] [1 11 248 0 40] [11 0 0 289 0] [1 0 8 0 291] | [202 2 9 85 2] [16 268 0 16 0] [20 0 205 5 70] [29 6 0 265 0] [4 0 103 0 193] | [278 0 13 8 1] [1 295 4 0 0] [2 0 289 0 9] [7 0 2 291 0] [1 0 9 0 290] |
| Accuracy | 0.94 | 0.755 | 0.962 |
| Precision | 0.942 | 0.761 | 0.963 |
| Recall | 0.94 | 0.755 | 0.962 |
| F1 score | 0.94 | 0.755 | 0.962 |

Fig. 10. Random Forest Algorithm Results Part 2

3.4 Neural Networks

The application of different image preprocessing techniques, specifically Gabor, HOG, and Sobel filters, has demonstrated significant efficacy in enhancing the performance of our neural network model for rice grain classification. Across all three methods, the network achieved notably high metrics, reflecting robust model accuracy and consistency. The accuracy, precision, recall, and F1 score for the models prepro-

cessed with Gabor filters were uniformly 0.97, showcasing excellent model performance. With HOG and Sobel filters, these metrics slightly improved to 0.98, indicating a marginal enhancement in the model's ability to classify rice grains accurately. These results underscore the importance of tailored preprocessing in improving the precision and reliability of image-based classification models. The Sobel filter proved to be slightly more effective in delineating finer details in the rice images, which likely contributed to the slight increase in the performance metrics. Each preprocessing method contributed to a high degree of model accuracy and generalization, demonstrating their respective strengths in capturing essential features for classification tasks.

| | Gabor | HOG | Sobel (x & y axes) |
|------------------|--|---|--|
| Confusion matrix | [2768 0 0 9 86] [0 2806 0 57 0] [8 0 2848 7 0] [4 128 6 2725 0] [156 0 0 0 2707] | [2811 0 0 8 44] [0 2815 0 48 0] [1 0 2854 8 0] [0 118 5 2740 0] [89 0 0 0 2774] | [2732 0 0 20 111] [0 2805 0 58 0] [0 0 2859 4 0] [0 66 5 2781 0] [19 0 0 0 2844] |
| Accuracy | 0.97 | 0.98 | 0.98 |
| Precision | 0.97 | 0.98 | 0.98 |
| Recall | 0.97 | 0.98 | 0.98 |
| F1 score | 0.97 | 0.98 | 0.98 |

Fig. 11. Neural Network Algorithms Results

4 Conclusion

In conclusion, the feature extraction methods worked better than the edge detection methods. The best grouping of feature extraction and classification algorithms is HOG filters and SVM with a 0.98 score across all. Each classification algorithm took a long time to run as there were a lot of values for them to process. The time mostly paid off however as overall, the algorithms yielded good predictive results. This proves the predictive efficacy of classification algorithms over supervised learning datasets.

For future works, it would be interesting to see how the classification algorithms would run with multiple rice images within one image. In other words, seeing how it reacts to multiple rice instances within one image would be intuitive. The dataset was large and that impaired our run times so I think deciding to work with a smaller dataset in the beginning would be helpful for analysis.

References

1. <https://www.kaggle.com/datasets/imtkaggleteam/acne-computer-vision/data>
2. <https://www.kaggle.com/datasets/ryandpark/fruit-quality-classification>
3. <https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset/data>