A Machine Learning Approach to Classifying Billiards Balls

October 9, 2024

1 Introduction

This report is about recognizing billiard balls from images. In our guild, there is a need for software that can analyze billiard games from video footage. Currently, our billiard statistics software only registers the outcomes, without insights into the gameplay itself. A system capable of recognizing the balls on the table could generate various player statistics, such as the success rate of long shots or games lost due to potting the eight ball prematurely. As a first step towards developing this system, this project focuses specifically on labeling individual billiard balls from clear, square-shaped images.

This report is structured as follows: First, the problem is defined, including an overview of the dataset and the machine learning approach. The methodology section details the data preprocessing, model selection, and the training process. This is followed by a discussion of the results, including evaluation metrics and insights gained from the experiments. Finally, conclusions and potential areas for improvement are outlined.

2 Problem Formulation

Our dataset of billiards balls contains 960 high-quality 3552x3552 pixel, 24-bit RGB images of billiards balls, 60 for each of the 16 different categories. Each image only depicts one ball. The data is therefore categorical with 16 different classes, one representing each type of ball.

We have cropped and resized the images to 64 by 64 pixels. We have also scaled the color 8-bit color channels, whose values range from 0 to 256, to floating-point values between 0 and 1. After preprocessing the image dataset, each image will constitute a 3×4096 feature matrix of floating-point values. A machine learning model, specifically a SVM classifier, is trained using this labeled dataset. This means we have a supervised machine learning task, with the goal of accurately predicting the labels of input images.

3 Method

A large literature of image classification projects exist. In this report we implement two approaches for image classification: a convolutional neural network (CNN) and a support vector machine classifier (SVC). For the SVC approach we have chosen to largely follow the method described in [1]. The CNN approach largely follows tensorflow's computer vision tutorial, which we also used for the dataset loading and preprocessing [2].

3.1 Dataset

The dataset of billiards ball images was created specifically for the goal of predicting the correct labels of billiards balls at our table location. We took 60 photos of each of the 16 balls from various angles, resulting in 960 total images. The images taken were 3552 by 3552 pixel jpg images with a bit depth of 24. The dataset was split into training, validation and test datasets. The final training dataset had 720 images, validation dataset 80 images and test dataset 160 images, corresponding roughly to a 75-8-17 split. We chose this split to have most of the data be used in training, while setting aside separate datasets for validation to prevent overfitting, and testing to evaluate the final performance of the model. The reasoning for a commonly used 75-10-15 split is explained in [3]. The training, validation and test datasets were randomly separated using scikit-learn's train_test_split-function.

3.2 Data Preprocessing and Feature Selection

Our initial dataset consists of unnecessarily large RGB images. Our preprocessing has three steps: Cropping the images, resizing them to 64x64 pixels and scaling the 8-bit integer RGB values down to a single floating-point number between 0 and 1. This is done to reduce the size of inputs to the model from an integer matrix of size $3 \times (3552^2)$, to a more manageable 3×4096 floating-point values. The color value scaling was done because machine learning methods generally work better with small input values. [4]. Now each feature is a 3 by 4096 matrix of floating-point numbers between 0 and 1. The dimensions of the matrix represent the red, green and blue color channels of the image. The features are labeled with a number between 0 and 15, representing the ground truth, which is the number of the billiards ball in the image. The cue ball is number 0.

With the Support Vector Classifier (explained in section 3.4), the data was additionally preprocessed with Principal Component Analysis (PCA) [5]. PCA reduced the size of the features by 75% from $64 \times 64 \times 3$ to $64 \times 16 \times 3$. The PCA was implemented using the PCA class of scikit-learn's decomposition module [6].

3.3 Data Augmentation

Since our initial dataset is somewhat small, we have extended it using dataset augmentation to increase the quantity and diversity of the training set. We have augmented the training dataset by applying random rotations, image flips and brightness adjustments to the preprocessed training dataset images. The augmentation was performed using keras' layers utility.

3.4 Model

SVM Classifier

For this task of recognizing billiard balls from images, we have selected the Support Vector Classifier (SVC) as our initial ML model. SVC is ideal for handling high-dimensional data, like images, by finding a hyperplane that maximises the margin between different classes. In this case, the classes are different billiard balls. SVC works by identifying the optimal weight vector and bias term to separate these classes. If the data isn't linearly separable, kernel functions such as the Radial Basis Function (RBF) can be used to map the data to a higher-dimensional space, allowing more accurate classification. [1], [7]

$$h(x) = sign(wx + b) \tag{1}$$

In this case, $h(x)$ assigns a class label to the input data point x, returning $+1$ when $wx + b$ is greater than or equal to 0, and -1 when it is less than 0.

SVC was chosen because of its ability to efficiently manage image-based tasks with numerous pixel features, and its effectiveness in separating classes through margin maximization. It's also robust against overfitting and performs well with smaller datasets like the couple thousand images used in this project, making it a practical and reliable choice. [1], [7]

Convolutional Neural Network

For this project, we also explored the use of Convolutional Neural Networks (CNN), another powerful machine learning model designed specifically for image-based tasks. CNNs are highly effective in recognizing patterns within images by taking advantage of their unique architecture, which is well-suited for extracting spatial hierarchies from visual data [8].

A CNN works by passing the input image through several layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters that move across the image to detect local features such as edges or textures. These features are then aggregated in the pooling layers, which reduce the spatial dimensions of the data, preserving important information while minimizing computational complexity. The final fully connected layers perform the actual classification by combining the learned features into predictions for each class [8].

CNN were chosen because they can automatically learn relevant features from images, minimizing the need for manual feature engineering. They are also robust when dealing with complex images, making them an excellent choice for the billiard ball classification task. Like Support Vector

Classifiers (SVCs), CNNs can handle high-dimensional data, but their architecture is particularly well-suited for spatially structured inputs such as image [8].

We implemented CNN using the same dataset of 960 labeled images, processed to 64x64 pixels, and trained the model on the training split. To improve the model's generalization, data augmentation techniques, such as random rotations and flips, were applied. PCA was not applied in the use of CNN.

3.5 Loss Function

SVM Hinge Loss

For this project, we have selected the hinge loss function as our initial loss function, which is commonly used with Support Vector Machines (SVMs). Hinge loss encourages the model to maximize the margin between different classes, penalizing predictions that fall too close to the decision boundary or are misclassified. This ensures that the SVC model not only classifies the billiard balls correctly but does so with confidence, reducing the likelihood of errors. [1], [9]

$$L(y, f(x)) = max(0, 1 - y \cdot f(x)) \tag{2}$$

Here, y denotes the actual class label (+1 or -1), while $f(x)$ refers to the classifier's decision function for the given data point x.

Hinge loss is computationally efficient and aligns perfectly with SVC's margin-maximizing objective, making it an ideal choice. By penalizing near-boundary predictions, hinge loss ensures that the model maintains clear and accurate boundaries between billiard ball classes, improving overall classification performance. [1], [9]

CNN Sparse Categorical Cross Entropy

For our Convolutional Neural Network (CNN), we used the Sparse Multiclass Cross-Entropy Loss function to optimize the model's performance in the classification of billiard balls. This loss function is specifically designed for multiclass classification tasks where each image corresponds to exactly one class, making it well-suited for our project of recognizing a single billiard ball from each image [10].

The Sparse Multiclass Cross-Entropy Loss works by comparing the true class label with the predicted probability for that class. The function takes the negative log of the predicted probability assigned to the true class, and then averages this value across the dataset. Mathematically, it is expressed as:

$$L(y_{\text{true}}, y_{\text{pred}}) = -\sum(y_{\text{true}} \times \log(y_{\text{pred}})) \tag{3}$$

Here, y_{true} represents the true label (an integer value between 0 and the number of classes), while y_{pred} is the predicted probability for that class. This formulation ensures that the model is penalized more when it assigns a lower probability to the correct class[10].

One advantage of the sparse version of cross-entropy loss is its computational efficiency, as it only requires the probability for the true class rather than for all possible classes. However, it is somewhat less robust to class imbalance than other loss functions, which could be considered in future iterations of the project. Overall, Sparse Multiclass Cross-Entropy Loss allowed us to efficiently train the CNN while maintaining a high level of prediction accuracy.

4 Results

5 Conclusion

6 Aknowledgements

Use of generative AI

Parts of this text were generated using ChatGPT, but all prompts were crafted and texts edited and reviewed by students. Additionally, ChatGPT was used for assistance in data processing and coding.

References

[1] unknown, "A Machine Learning Approach to Classifying Bangla Handwritten Characters," Sep. 2023.

[2] "Computer vision with TensorFlow — TensorFlow Core," Accessed: Sep. 20, 2024. [Online]. Available: https://www.tensorflow.org/tutorials/images.

[3] V. R. Joseph, "Optimal Ratio for Data Splitting," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, pp. 531–538, Aug. 2022, ISSN: 1932-1864, 1932-1872. DOI: 10.1002/sam.11583. arXiv: 2202.03326 [cs, stat]. Accessed: Sep. 19, 2024. [Online]. Available: http://arxiv.org/abs/2202.03326.

[4] "Importance of Feature Scaling," scikit-learn, Accessed: Sep. 20, 2024. [Online]. Available: https://scikit-learn/stable/auto_examples/preprocessing/plot_scaling_importance.html.

[5] S. Wold and K. E. P. Geladi, "Principal Component Analysis,"

[6] "PCA," scikit-learn, Accessed: Oct. 9, 2024. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html.

[7] W. S. Noble, "What is a support vector machine?" *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, Dec. 2006, ISSN: 1087-0156, 1546-1696. DOI: 10.1038/nbt1206-1565. Accessed: Sep. 20, 2024. [Online]. Available: https://www.nature.com/articles/nbt1206-1565.

[8] K. O'Shea and R. Nash. "An Introduction to Convolutional Neural Networks." arXiv: 1511.08458 [cs], Accessed: Oct. 8, 2024. [Online]. Available: http://arxiv.org/abs/1511.08458, pre-published.

[9] P. L. Bartlett and M. H. Wegkamp, "Classification with a Reject Option using a Hinge Loss," Aug. 2008.

[10] C. Jeeva. "Loss Functions in Neural Networks," Scaler Topics, Accessed: Oct. 8, 2024. [Online]. Available: https://www.scaler.com/topics/loss-functions-in-neural-networks/.

A preprocessing.pdf