DS4021 Final Project Report
Brian Hockett, Eva Winston, Karen Guzman

Introduction

Image classification is an important task in machine learning with a variety of real-world applications. On a large scale, image classification powers systems from autonomous vehicles and self-checkout retail technologies to agricultural monitoring tools capable of detecting crop health [1]. While most image classification work focuses on traditional photographic images, human-drawn sketches introduce a unique challenge: they are visually simple, are primarily blank space, and usually created by non-artists using touch-based devices, requiring models to extract meaning from minimal line information rather than detailed pixel patterns [2]. Sketch-based image recognition is useful in several areas, such as search-by-drawing interfaces, educational tools, and accessibility technologies [3]. However, because sketches lack the rich pixel information of photographs, it is less clear how traditional machine learning models interpret them. Our project is motivated by this gap in the classification of sketch-based data; we aim to better understand how models interpret low-detail visual data and what factors may influence classification performance. Convolutional neural networks (CNN) have emerged as a powerful framework for recognizing these sketches due to their ability to learn hierarchical spatial features [2]. However, we wanted to explore how different machine learning approaches perform on human-drawn images, without relying on convolutional neural networks. Using Google's "Quick, Draw!" dataset of human-drawn images, we selected 4 object classes and trained three models: Random Forest, Histogram Gradient Boosting, and a Multilayer Perceptron Neural Network. By applying techniques learned throughout the course, we aim to answer the following question: How do Random Forest, Gradient Boosting, and Neural Network models compare in their ability to classify human-drawn objects?

Methods

The dataset we used for this project was Google's "Quick, Draw!" dataset, containing 28x28 grayscale bitmap images of a variety of human-drawn objects, stored as NumPy arrays. For the purposes of this project, we selected the images of 4 object types. Each array contains 784 values corresponding to the pixel intensities of the flattened 28x28 image. To create our structured dataset, we loaded these arrays into a pandas DataFrame, where each row represented a single image. Next, we uniformly standardize the data, dividing each value by 255, to move the range of values from $[0, 255] \rightarrow [0, 1]$. The label corresponding to the image was then appended as an additional column. Due to the size of the data and the constraint on training time and resources, we reduced the data to include just 15% of the images, prior to splitting into training and testing sets. This resulted in a training set with 68,250 images, and a testing set with 17,063.

For all 3 models, the only preprocessing step necessary was label encoding, which mapped the labels from strings (ex. Airplane) to integers (ex. 0). The exact methodology for conducting this mapping varied from model to model. Likewise, for all 3 models, we made use

of Stratified K-Fold cross validation with 5 folds to tune the models. For the histogram gradient boosting model, the learning rate and max depth hyperparameters were tuned. For the neural network model, the learning rate, weight decay, dropout, and layer sizes were tuned. For the random forest model, the number of trees in the forest, maximum depth, minimum number of samples required to split an internal node, and minimum number of samples required at a leaf node were tuned. For evaluation of the model performance, we used the cross-validation accuracy of the model, which is the mean accuracy on the validation set across all folds.

Results

All 3 of the classification models performed very well in the cross-validation stage, each having a cross-validation accuracy greater than 94%. As seen in the chart below, our Multilayer Perceptron (MLP) Neural Network model performed the best among the three approaches.

Cross-Validation Performance (Accuracy)		
Random Forest	**Gradient Boosting**	**Neural Network**
94.41%	95.44%	95.91%

As a result, the Neural Network model was then trained on the entirety of the training set, using the optimal hyperparameters found via cross-validation. Evaluating this model on the testing set yielded a very high accuracy of 95.79%. This far exceeds our initial goal of ~80% that we set out to accomplish in this project. We also examined the resulting confusion matrix and found that the model performed very well on all 4 classes, despite the imbalance in the class frequencies.

Discussion

The Neural Network model performing best is intuitive because the images were small and simple, and MLPs are adept at memorizing essential patterns, such as strokes in a doodle. This model may have performed better than our Gradient Boosting and Random Forest models because it had an enhanced understanding of the spatial structure of each image, whereas the tree-based models treat each pixel as a completely independent feature.

Despite these encouraging findings, limitations of this project should be noted. Google's "Quick, Draw!" dataset contains hundreds of object types, but our analysis only used four classes. As mentioned in the methods section, a 15% subset of images from these classes was used due to constraints on the types of models being used, as well as computational and time resources available for model training. Additionally, sketch data is inherently noisy due to a variety of quality and styles. Because the dataset consists of real users playing a timed game, many drawings are incomplete, overly simplistic, or rushed scribbles, which introduces noise that may affect model performance. Our dataset contains only digital sketches made on trackpads or touchscreen devices, which may not fully represent how sketches are produced in other real-world contexts, i.e., via a stylus or other drawing tools. Future analysis could be expanded by increasing the number of classes, using more images per class, or testing alternative models such as Convolutional Neural Networks.

References

[1] OpenCV, “Image Classification,” 2023. [Online]. Available: https://opencv.org/blog/image-classification/

[2] S. Saxena, A. Upadhyay, and Y. Varshney, “Sketch Recognition via CNNs,” Stanford CS231n Course Project Report, 2017. [Online]. Available: https://cs231n.stanford.edu/reports/2017/pdfs/420.pdf

[3] M. Eitz, J. Hays, and M. Alexa, “How Do Humans Sketch Objects?,” *ACM Transactions on Graphics (SIGGRAPH)*, vol. 31, no. 4, pp. 1–10, 2012. [Online]. Available: https://www.researchgate.net/publication/254461838_How_Do_Humans_Sketch_Objects