DS200 Final Project Report: Image Classification

Vilde Roland Arntzen and Parker David Nelson

University of California, Berkeley

1 Introduction

Classification seems to be a trivial task, but it can be found at the core of nearly every discipline – a good doctor can classify different diseases and treat them accordingly, good photographers are able to determine a beautiful shot from a bad one, and the ability of scientists to classify different phenomena determines their expertise. Giving computers the ability to classify is an immensely powerful concept already seen in a wide range of applications – from autonomous driving to cancer screening and facial recognition. This project examines how the task can be solved using the traditional machine learning algorithms – logistic regression, k-nearest neighbor, classification tree, random forest and a support vector machine. The classification problem in this context is framed as a 20-class, single-label classification which predicts the content of an image to be one of the 20 classes.

2 Data

The given dataset consists of 1501 images of 20 different classes. Each image is represented as a three dimensional array in which the first two dimensions represents the row and column pixels and the third dimension represents color with three values corresponding to red, green and blue (RGB) color intensity, respectively. The dataset contains images of different sizes and color scales. Most images were colored while some were grayscaled. Two samples of the images are displayed in Figure 1 and 2 with the corresponding classes comet and airplane, respectively. Further investigation of the data was done during the phase of data cleaning and exploratory data analysis.

3 Exploratory Data Analysis and Feature Extraction

Exploratory data analysis (EDA) was performed to build a deeper understanding of the data. The main goal of the EDA was to find properties in the data that differed between image classes such that the different classes could be separated easily in the classification process.

Fig. 1. Image from the dataset belonging to the comet class.

Fig. 2. Image from the dataset belonging to the airplane class.

3.1 Data Cleaning, Summary, Transformation and Visualization

The data was cleaned to resolve inconsistencies and confirm a consistent format of the dataset. As mentioned, the dataset contained images of different sizes and of different color scales. To ensure consistency among the images, they were converted into the RGB color scale. The images were kept at their original, inconsistent sizes for the scalar features because they did not depend on the size of the image. However, to increase the speed of the calculations, the images were downsized by 25 percent for the EDA section of the project and then returned to their original size once the features and tuning were completed. The matrix based features, on the other hand, needed to have a consistent size to be added to the feature matrix. The images were downsized to 50 by 50 pixels for these features. 50 by 50 was chosen because it was the lowest pixelation that still allowed for the human eye to classify the images, as seen in Figure 3 and 4. Also, the images were converted to grayscale because the features were derived off the Harris Corners and Canny Edges algorithms where color wasn't essential anymore.

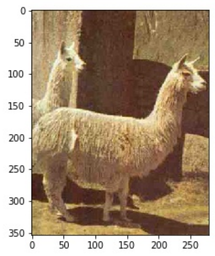
Fig. 3. Image from the dataset belonging to the llama class, not converted.

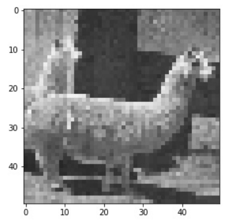
Fig. 4. Image from the dataset belonging to the llama class, converted.

The data was summarized into values such as the mean, median and standard deviation of properties before being visualized to inspect how the distribution of each value varies between image classes.

The EDA resulted in some particularly interesting results, such as the distribution of red channel intensity in images displayed in Figure 5. Figure 5 shows that the distribution of the mean of red channel intensity for the comet class is lower than all the other classes' distributions. Further investigation showed that the distribution of the mean in the green and blue channel intensities also deviates for the comet class. As one can see in Figure 1, the comet picture is characterized by a darker background with an area of higher luminance–representing the comet. When the images are displayed from the comet class, one could easily find that this is typical for the comet images. This property is captured in the mean value of channel intensity, which differentiates the comet from the other classes.

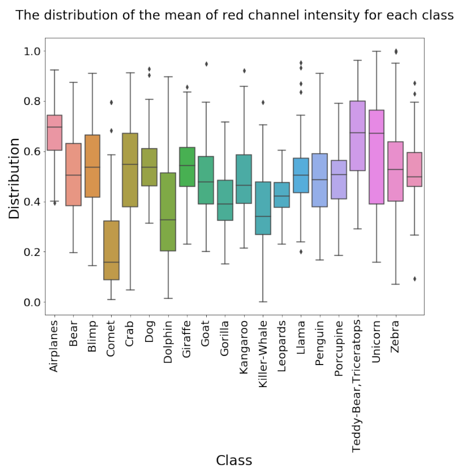

Fig. 5. Shows the distribution of the mean of red channel intensities for each class.

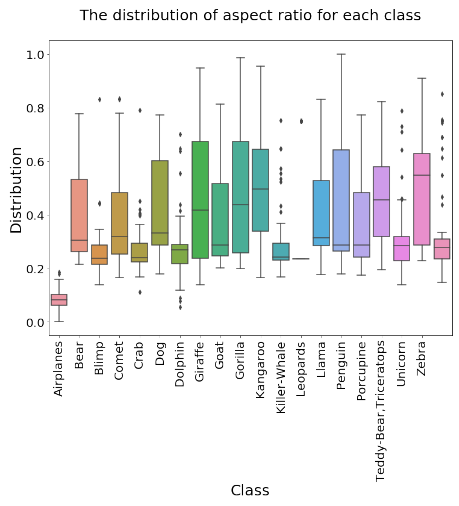

Fig. 6. Shows the distribution of the aspect ratio for each class.

The aspect ratio distributions can be looked at in a similar way to the mean color intensity distributions. The distribution for the airplane class differs greatly from the distributions of the other classes. The airplane images are often longer in the horizontal direction than the vertical direction as illustrated in Figure 2. When looking at the different images of airplanes, this seems to be typical for the class and is due to the long horizontal side, and shorter vertical side of an airplane. Because the aspect ratio equals the height divided by the length of an image, this will result in a lower aspect ratio for the images, which again gives a lower distribution of aspect ratio for airplanes than for other image classes as showed in the first box distribution visualized in Figure 6. This feature is therefore good at separating the airplane class from the other 19 classes.

Other methods that were used to extract image properties include canny edges detection and harris corner detection which detect edges and corners within an image, respectively. These are matrix-based which means that they return a matrix which capture the corners and edges of an image, and can be displayed as an image itself. Before exploring these methods the intuition was that some of the classes' distribution of detected edges and corners would be much larger than other classes because of the difference in physical appearance between classes. An image of an airplane will, for example, contain more corners than an image of a comet. However, it was shown that the distribution of the mean of harris corners detected is quite similar for the different classes. As opposed to the hypothesis, the distribution of the mean harris corners detected is in fact greater for comets than it is for airplanes. The harris corner detection is sensitive to low contrast and non-obvious edges [3]. Thus, the contrast of dark background to high luminance in the comet images might have a greater impact on the harris corners detected than the actual physical shape reflected in corners of an airplane because airplane images often have less contrast than comet images.

Each pixel of the harris corners and canny edges matrices were added as separate features in the feature dataframe. The accuracy increased only on the training set and decreased on the test set, which shows that the model was significantly overfitted. The matrix based features also added over 5000 new columns to the dataframe as well, which dominated the other features. We also tried taking the mean of each column and recording it as its own feature in the feature set. The intuition behind adding this many new features was to see if there was a linear relationship between the number of edges and corners to each type of image. It was found that this significantly decreased the test accuracy for the models, so only the aggregated features were included in the final model.

3.2 Feature selection

The examination of image properties during EDA resulted in the features displayed in Table 1. For the classification tree and random forest models the sklearn function for feature importance confirmed that both the aspect ratio and red channel intensity are important features in separating classes from each other, confirming the interpretation of the distributions explained in the EDA section. The other features were not as important, but still contributed to the overall accuracy. The matrix based features did not contribute nearly as much as was previously thought. This may be because the aggregated scalar features had already captured much of the variance between the different classes. For example, several of our scalar features included the mean and standard deviation of the harris corners that were detected in the image.

Feature Name	**Description**
Pixel Size	Returns the pixel size of the image
Mean Red Channel	Returns the mean of the red channel for the image
Mean Green Channel	Returns the mean of the green channel for the image
Mean Blue Channel	Returns the mean of the Blue channel for the image
Aspect Ratio	Proportion of width and height for each image
Median	Calculates the median of the image
Std.	Calculates the standard deviation of the image
Harris Detect	Returns the proportion of harris corners detected
Mean Harris	Returns the mean of harris corners detected
Std. Harris	Returns the std of harris corners detected
Std. Red Channel	Returns the std of the red channel for the image
Std. Green Channel	Returns the std of the green channel for the image
Std. Blue Channel	Returns the std of the blue channel for the image
Mean Canny	Returns the mean of the detected canny edges
Std. Canny	Returns the standard deviation of the detected canny edges
Full Canny	Computes the full canny edges of an image and outputs each value in its own column
Full Harris	Computes the full harris corners of an image and outputs each value in its own column

Table 1. Description of the features selected for classification.

3.3 Scaling the selected features

The features displayed in table 1 are all on different scales; each one corresponds to different measurements and will be unfairly weighted in the models. The wide variations of values affects the objective function of many machine learning algorithms resulting in giving some features a higher weighting in the classification process than others. For example, the k-nearest neighbor algorithm uses euclidean distance to calculate the distance between two points. If one feature has a broad range of values, the distance will be governed by this particular feature. To avoid this, the features need to be scaled using a normalization technique so that they will contribute to the final distance proportionally.

Before the data was fed into the models, eight scaling methods were applied to the features and compared in an attempt to optimize the models' accuracy. The sklearn.preprocessing package in Python was used to transform the features into a representation that was more suitable for the classifiers. The range of different normalizations used to improve the models are displayed in Table 2. The QuantileTransformer with the normal output distribution outperformed all of the other scalers. The variance of these different scaling methods outlined the need to do more analysis in the area of scaling the features properly, but it seemed out of scope for the purposes of this paper.

4 Classification Models

Five traditional machine learning models–logistic regression, k-nearest neighbors (KNN), decision tree, random forest, and support vector machine(SVM) – were

Feature Transformation	**Best Testing Accuracy Found**
MinMAXScaler	∼ 33%
StandardScaler	∼ 39%
MaxAbsScaler	∼ 32%
RobustScaler	∼ 33%
PowerTransformer	∼ 39%
QuantileTransformer(Uniform)	∼ 37%
QuantileTransformer(Normal)	39.78 %
Normalizer	∼ 29%

Table 2. Sklearn transformers used to scale the selected features, with corresponding accuracy on the test set.

used for the classification task. The models were implemented using the Python library scikit-learn where each model was trained on 90% and evaluated on 10% of the training data. For suitable models, a random seed was specified to ensure reproducibility of the models and consistency in validation measures.

4.1 Hyperparameter Optimization

To control the learning process of each classifier and optimize its performance, hyperparameter selection was performed for each separate model using scikit-learn's functionality for grid search. The grid search does an exhaustive search through a manually specified subset of each hyperparameter space to find the best parameters based on a performance measure [2]. Optimizing the hyperparameters increased the average accuracy of the models by roughly 9%. This was a significant improvement for all of the models.

Table 3. Accuracy results on the unseen validation set

Model	**Accuracy**
Logistic Regression	38.97 %
K-Nearest Neighbors	34.59 %
Classification Tree	34.11 %
Random Forest	37.31 %
Support Vector Machine	38.18 %
Combination Model	38.41 %

Table 4. 5-fold cross validation accuracy results on the unseen test set.

4.2 Classifier Performance Assessment

To validate each of the models, 5-fold cross validation was performed to get a more consistent measure of the model's accuracy. With the decided hyperparameters, the logistic regression model proved to have the highest test accuracy. The reason this model prevailed over the others seemed to be in the way the data was scaled. When other scalers were used, the logistic regression model was the one of the least accurate. According to the sklearn documentation, it "reduces the impact of (marginal) outliers" and spreads out the most frequent values in the given set of features by transforming the dataset to a normal distribution [4]. This makes sense because logistic regression minimizes least squares loss and the data became more consistent. It was surprising that the combination model wasn't the most accurate because it combined all the other models. This may be because the other models weren't very accurate to begin with and the majority 'voted' incorrectly. The models aren't very accurate due to the inherent nature of the data; each image class is not the exact same. For instance, dogs can be extremely different in color and shape; yet each one should still be classified as a dog. When there are 20 different highly variable classes of images it becomes difficult to find concrete differences between without having a larger data collection.

4.3 Neural Networks

Convolutional Neural Networks are the industry standard for image classification. They can achieve a very high accuracy for image classification. A neural network was created in an attempt to showcase the high accuracy of the industry standard; however, it turned out to not be as accurate as our linear model due to time constraints.

The image data was cleaned to become a constant shape of 250 by 250 pixels and converted to grayscale. Upon examining the images, this seemed to be reasonable as one could still classify each image correctly. The neural network was created using the keras library[1] and trained with the same dataset as the other models. The network was created through experimenting with different layer frameworks and hyperparameters. Further research is definitely necessary in this area to achieve a higher accuracy for this classification problem.

5 Conclusion

Traditional machine learning algorithms are not very accurate in multi-class image classification. The value in using these algorithms in this context is exploring their different strengths an weaknesses on a large and complex dataset. EDA, data transformation methods, and different forms of visualizations aided selecting useful features that separate different image classes. The aspect ratio and red channel intensity features were shown to be important when separating the image classes. In addition, five traditional machine learning algorithms were explored using the Python package sklearn. The models were tuned using grid search to optimize hyperparameters, then compared to find the most accurate model that resulted in a final accuracy of 38.97% on the unseen test set, calculated using 5-fold cross validation. To increase the accuracy for the set, more work needs to be done in analyzing the normalization methods and researching more accurate neural network frameworks.

References

1. Chollet, F., et al.: Keras. https://keras.io (2015)
2. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification pp. 5–8 (2003)
3. Lee, C.Y., Wang, H.J., Chen, C.M., Chuang, C.C., Chang, Y.C., Chou, N.S.: A modified harris corner detection for breast ir image. Mathematical Problems in Engineering Volume 2014, pp. 1–2 (2014)
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011)