DS200 Final Project - Image Classiﬁcation

Jiujiu Lou, Tianrui Huang

University of California, Berkeley

[Abstract 2](#_Toc27159968)

[1. Introduction 2](#_Toc27159969)

[2. Data Description and Preprocessing 2](#_Toc27159970)

[2.1 Data Description 2](#_Toc27159971)

[2.2 EDA and data visualization 4](#_Toc27159972)

[Mean Features 5](#_Toc27159973)

[Variance Features 6](#_Toc27159974)

[Explore Correlation 7](#_Toc27159975)

[Quantile Features 8](#_Toc27159976)

[Edge-related Features 8](#_Toc27159977)

[2.3 Data Preprocessing 9](#_Toc27159978)

[3. Models 10](#_Toc27159979)

[3.1 Logistic Regression 10](#_Toc27159980)

[3.2 K-Nearest Neighbor 11](#_Toc27159981)

[3.3 Decision Tree and Random Forest 12](#_Toc27159982)

[3.4 Support Vector Machine 13](#_Toc27159983)

[4. Result Analysis 15](#_Toc27159984)

[4.1 Features of Interest 15](#_Toc27159985)

[4.2 Ineffective Features 15](#_Toc27159986)

[4.3 Final Model Selection 16](#_Toc27159987)

[5. Discoveries 17](#_Toc27159988)

[5.1 Interesting Findings 17](#_Toc27159989)

[5.2 Limitations 17](#_Toc27159990)

[5.3 Future Works 17](#_Toc27159991)

[Reference 18](#_Toc27159992)

## Abstract

Image classification is a long discussed but still challenging topic in digital image processing. Traditional classification models often perform poorly on image classification due to variations in angles and colors. In this project, we aim to develop an image classifier with efficient features extracted from the given images with labels.

In the first part, we performed exploratory data analysis on colors and sizes of images. Then we preprocessed data by trimming and resizing the input images. We also introduced Laplacian operators to generate grayscale images that outline edges. Then we selected useful features including mean values, variance, quantiles of three color-channels and the grey edge images.

In the second part, five different models were trained and evaluated using 5-fold Cross Validation (CV) with the features aforementioned. Our models included Logistic regression, K-Nearest Neighbor (KNN), Decision Tree, Random Forest, and Support Vector Machine (SVM). Among these models, the logistic regression model obtained a higher CV accuracy, as well as a more consistent performance regardless of penalty strength.

Lastly, we summarized important breakthroughs, possible limitations of our project, and discussed possible future works to improve our model.

## 1. Introduction

Image classification is one of the basic problems in image processing, and concerns wide applications such as abnormal cells identification in healthcare industry and vehicles detection in self-driving system. Given a set of labelled images, our goal is to train a model that recognizes the features of an input picture, and then predicts which class the picture belongs to.

Many classifiers based on machine learning algorithms have been developed for the long-discussed topic. The most commonly seen ones are logistic regression, KNN, classification trees, random forests and SVM. These methods are proven effective in many other cases, but need further discussion and improvement when applied to image processing, as it is challenging to extract effective features from various images. In this project, we explored effective features with EDA, trained and compared various models, and evaluated the effectiveness of the models.

## 2. Data Description and Preprocessing

### 2.1 Data Description

Our training set included 1500 labeled images from 20 different classes. The original raw data were enclosed in folders and we read those images and used pathstrings to give labels to each image. All images were in RGB format, with 16 special cases of grayscale images.

Encoded images have the following straightforward numerical mapping:

|  |  |  |  |
| --- | --- | --- | --- |
| Encode | Label | Encode | Label |
| 0 | Airplanes | 10 | Kangaroo |
| 1 | Bear | 11 | Killer-Whale |
| 2 | Blimp | 12 | Leopards |
| 3 | Comet | 13 | Llama |
| 4 | Crab | 14 | Penguin |
| 5 | Dog | 15 | Porcupine |
| 6 | Dolphin | 16 | Teddy-Bear |
| 7 | Giraffe | 17 | Triceratops |
| 8 | Goat | 18 | Unicorn |
| 9 | Gorilla | 19 | Zebra |

Table 1. Image Encodings

We made a bar-plot to visualize number of pictures in different classes. Most of the classes contain less than 80 images, except for Gorilla [9], Leopards [12], and Penguin [14].

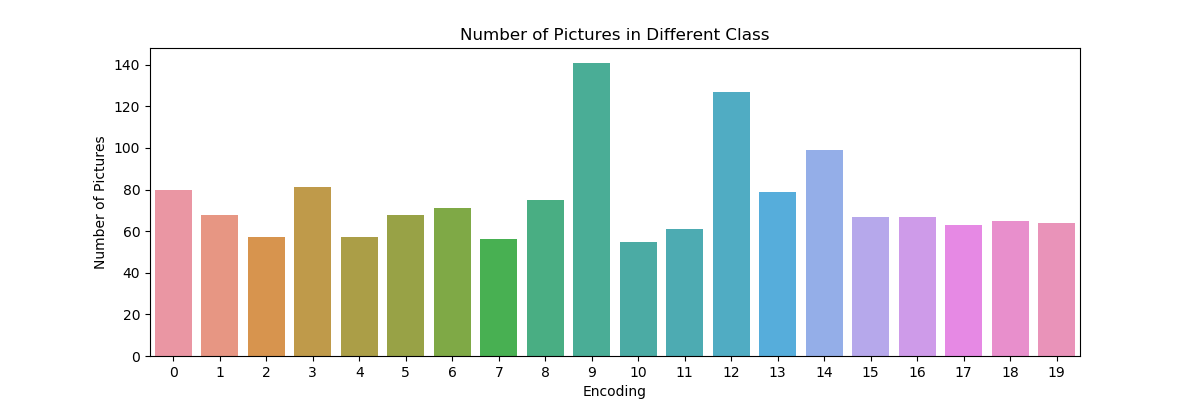


Fig 1. Number of Pictures in Different Class

### 2.2 EDA and data visualization

Initial exploratory analysis started with visualization of images. At first glance, we noticed some images have empty white space, and varying image sizes. This prompt us to generate ideas for initial data preprocessing, such as cropping and resizing our images.

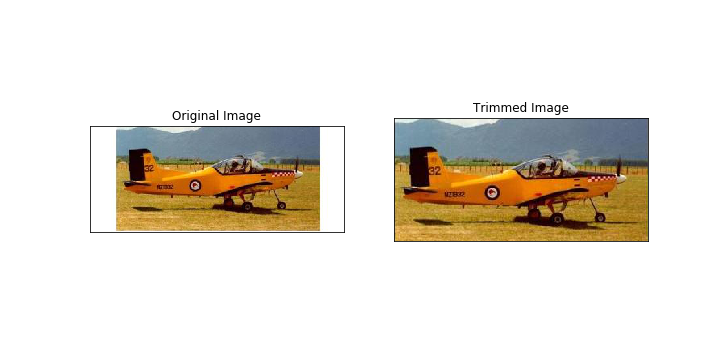


Fig 2. Cropping Image

After treating our original images with initial background cropping, we visualized the first two features that could be interesting: image size and aspect ratio.

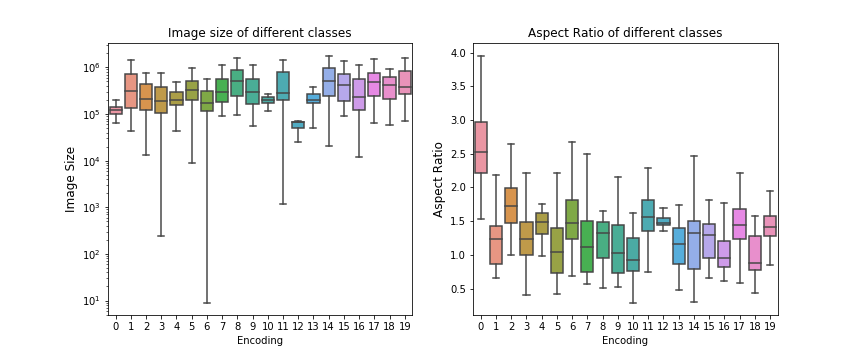


Fig 3. Boxplot of Figure Size and Aspect Ratio

Image size and aspect ratio varied dramatically between classes. For instance, overall, aspect ratios of airplane [0] images were much larger than other classes, which suggested that it can serve as an important feature for classification.

Since both image size and aspect ratio vary widely, to maintain data input consistency, we then decided to resize our images to a baseline shape. We finally settled on choosing 400x300 as our standard image size based on the original images’ distributions.

#### Mean Features

Then we tried to capture some scalar features such as channel specific means across classes to check if those potential features make sense.

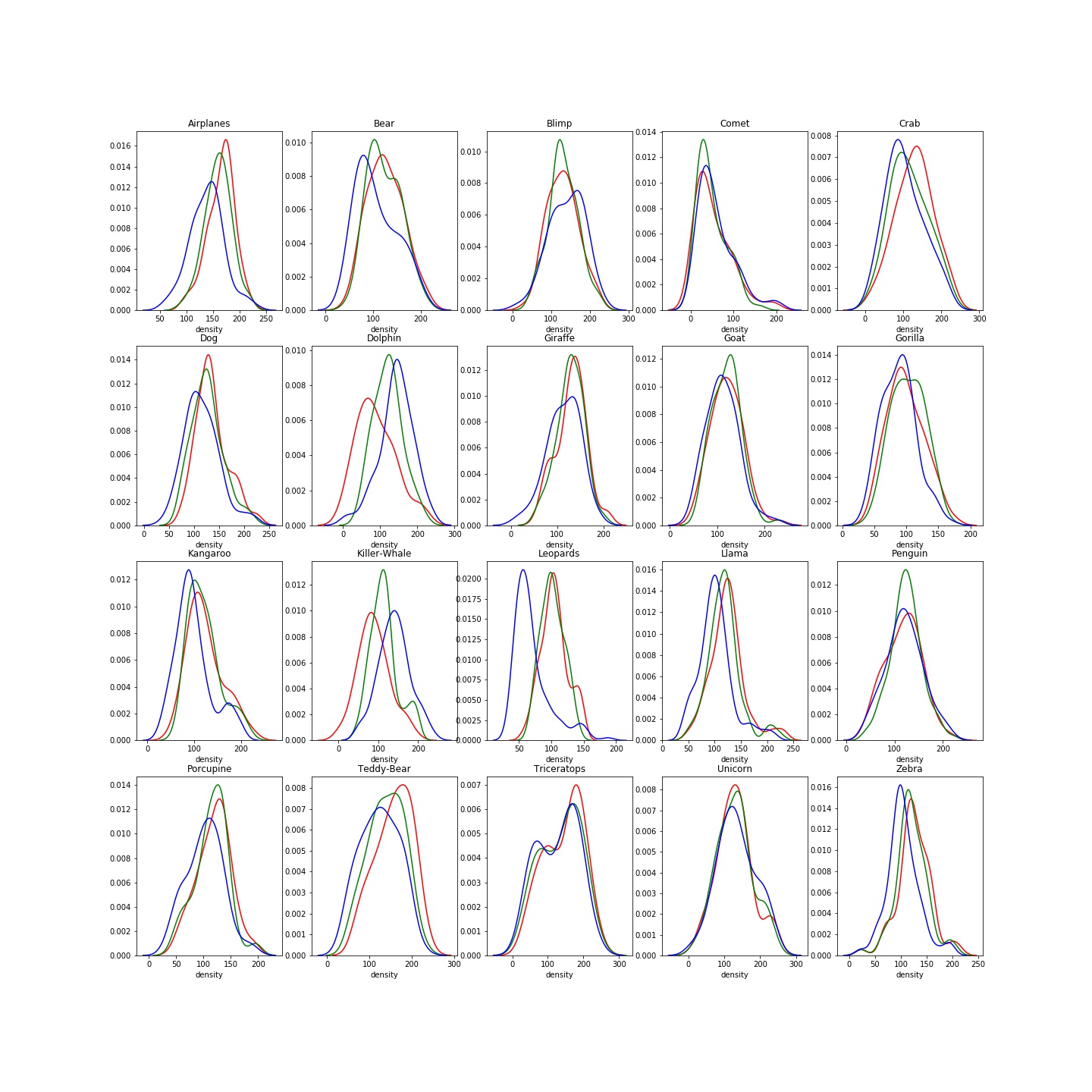


Fig 5. Distribution Plot of RGB-channel Average across classes

We can see each class had its own distribution of colored means. For instance, Leopards images had lower blue-channel mean compared to red and green-channel, which reflected that the dominating color of leopard is orange (red+green).

#### Variance Features

In addition to mean values, variances of a particular channel may also vary across classes. Putting distributions in one plot provides a straightforward illustration of this idea.

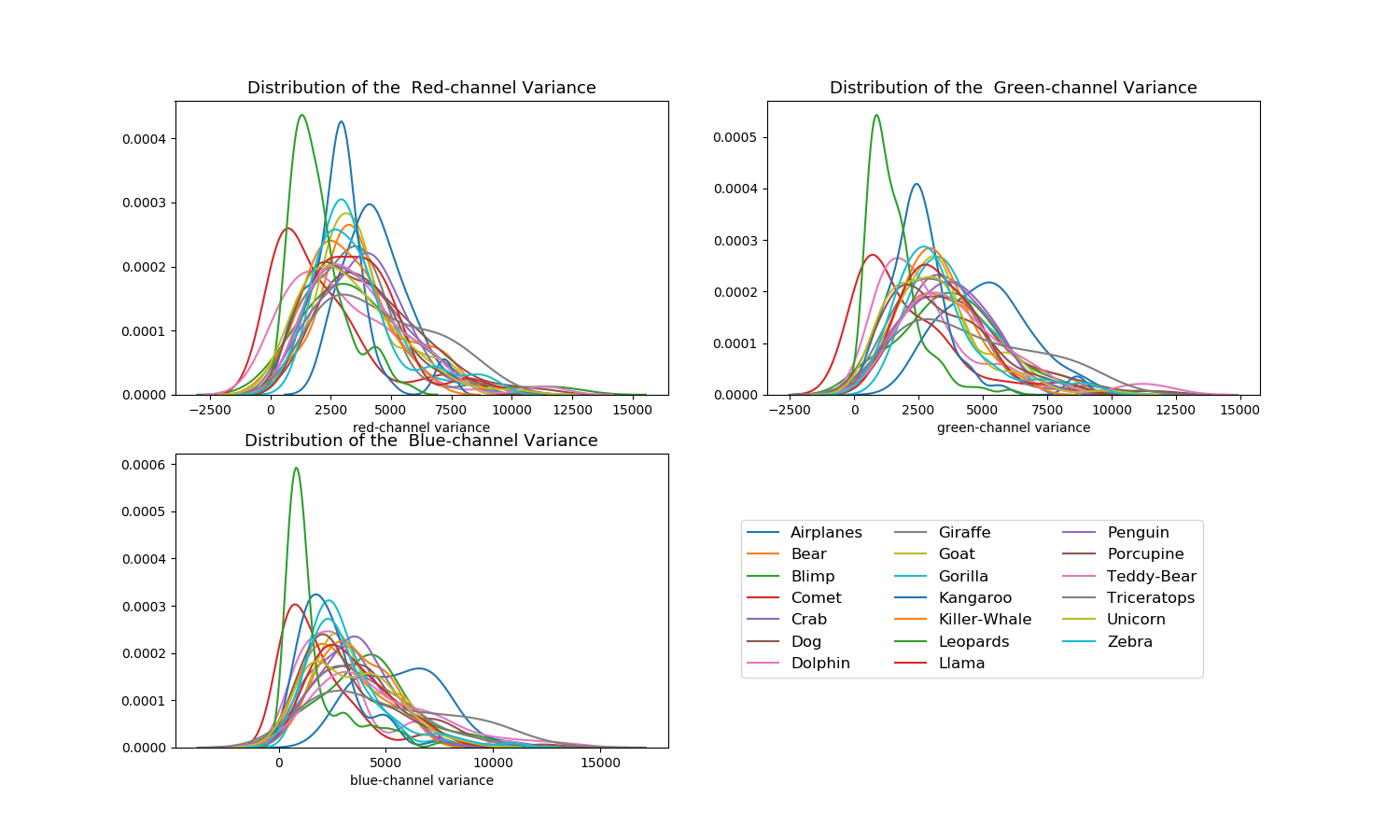


Fig 6. RGB Variance Distribution of different classes

#### 

#### Explore Correlation

Furthermore, we were interested in identifying possible correlations between our features and understand potential clusters that would aid our classification.

Though no clear cluster were identified, we observed that the green-channel and red-channel average values were somehow correlated, which should be considered in feature selection.

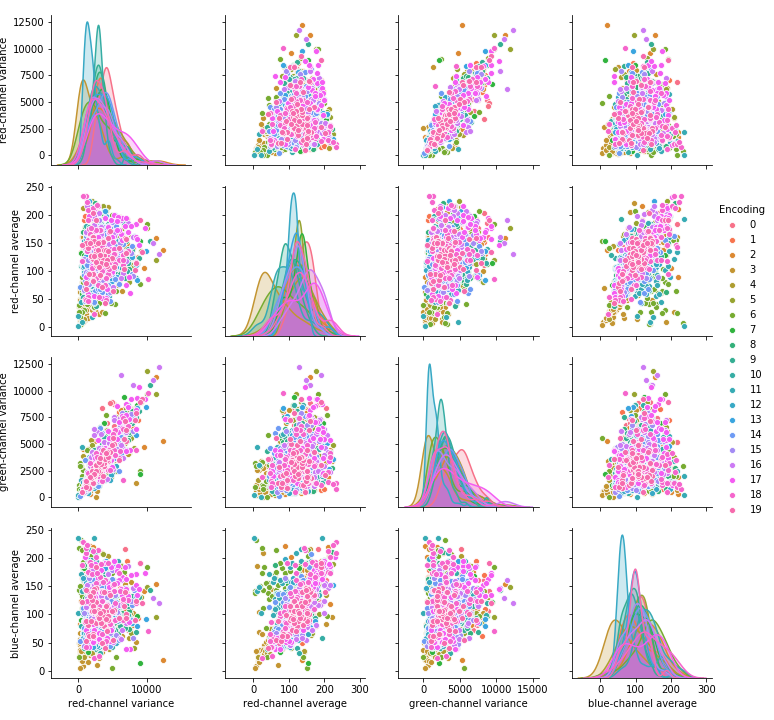


Fig 8. Pairwise Plot between Colored Features

#### 

#### Quantile Features

To obtain vectorized features and make efficient use of our image data, we also took quantile values of each channel into consideration. For each class, we calculated the average quantile values, and then plotted a line connecting them to explore this trend. Surprisingly, each class had rather distinct quantiles distributions, indicating the feasibility of taking quantile features into model training.

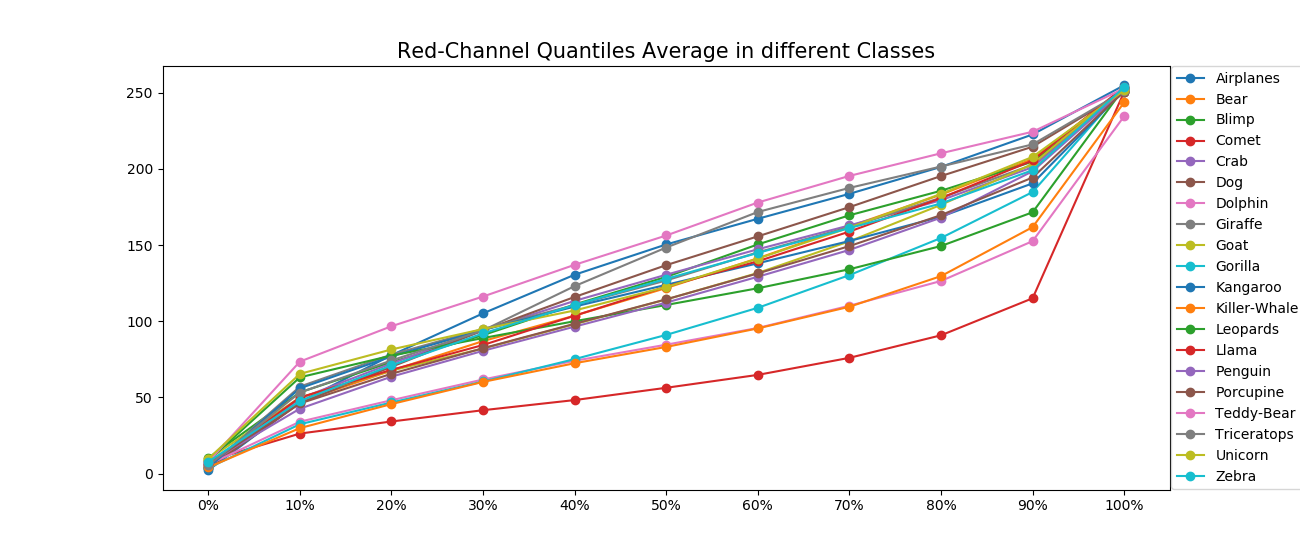


Fig 7. Red-channel Quantiles in different classes

#### Edge-related Features

Lastly, we took a look at additional features beside color based. We first converted original RGB image to grayscale and then use openCV laplacian operator to obtain its edges.

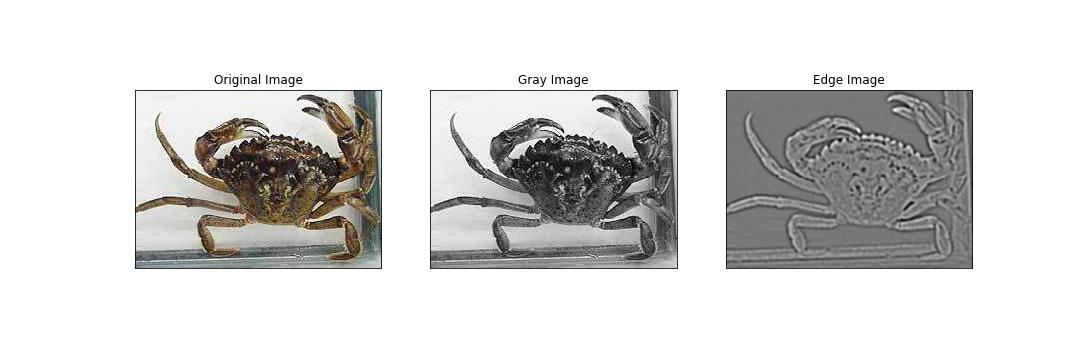


Fig 9. Converting to Edges

Then, similar to single color cases, we extracted the mean, variance as well as quantiles from these images that represented edges.

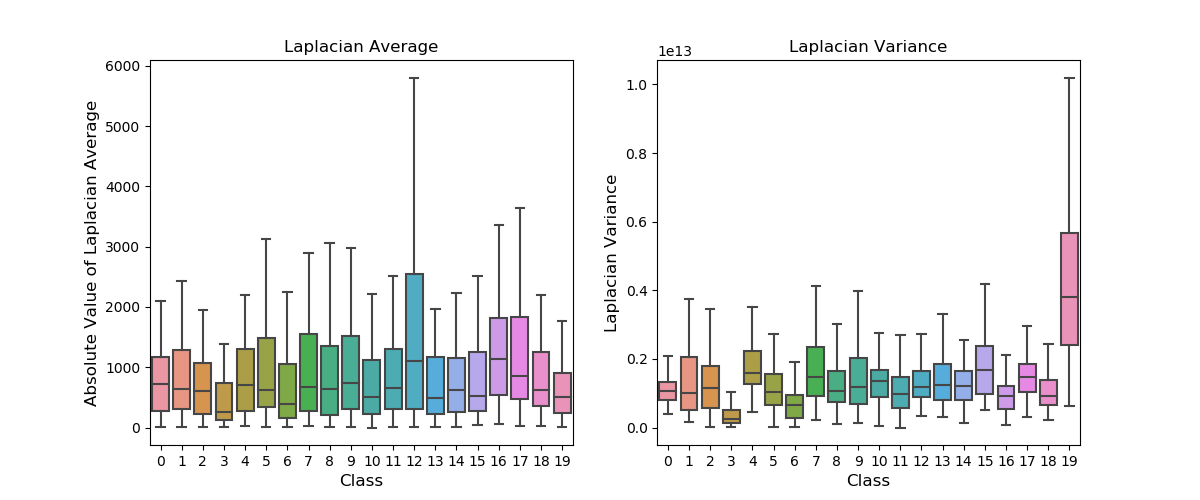


Fig 10. Average Value and Variance of Converted Images

### 

### 2.3 Data Preprocessing

As mentioned above in section 2.2, the main features selected are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature ID | 1 | 2 | 3,4,5 | 6,7,8 | 9 | 10 |
| Feature Description | Image size | Image aspect ratio | RGB mean | RGB variance | Laplacian mean | Laplacian variance |
| Feature ID | 11-21 | 22-32 | 33-43 | 44-54 |  |  |
| Feature Description | Red channel quantile | Green channel quantiles | Blue channel quantiles | Laplacian quantiles |  |  |

Table 2. Feature ID Matching

The steps taken to obtain those features are detailed below:

1. Trim all images off white/ black borders
2. Obtain scalar features that rely on original image (size, aspect ratio)
3. Resize images to maintain standard (resized to 400x300)
4. Obtain both scalar and vector features for RGB channels (mean, variance and quantiles)
5. Convert trimmed images to grayscale
6. Apply laplacian operator on grayscale images to find edges
7. Obtain laplacians related features (mean, variance and quantiles)
8. Standardize all features for model stability and even weighting on model coefficients

## 3. Models

There are plenty of model choices for classification tasks. Existing packages from scikit-learn were utilized in this project. Each model was ran on preprocessed data using 5-fold cross validation and confidence intervals drawn based on CV accuracy scores. Hyperparameters were also tuned based on the CV scores.

### 3.1 Logistic Regression

Logistic regression is perhaps the most natural for classification. There are a few variations possible. First, we can choose the scheme of one-over-rest or multinomial. In case with one-over-rest, we evaluate each category separately as a binary classification problem and select the one with the highest binary probability (Cheng 2006). For the multinomial case, we make use of softmax function and optimize over all class probabilities together. Turns out, both methods work relatively well with our dataset. Another hyperparameter that would be interesting to tune is the regularizer strength of our model. As shown below, for both logistic regression schemes, we visualized the model average accuracy over regularizer strength. We saw the best performance on test set at 0.3994 with a multinomial scheme and the penalty strength at c = 1.

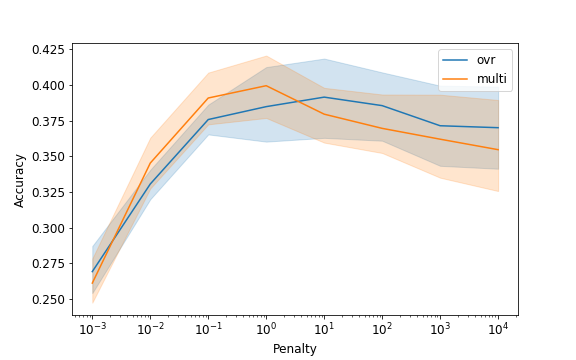


Fig 11. Logistic Regression CV Accuracy

We then took a look at feature coefficients from the logistic model with highest accuracy to check if there were potentially redundant features or those with little significance.

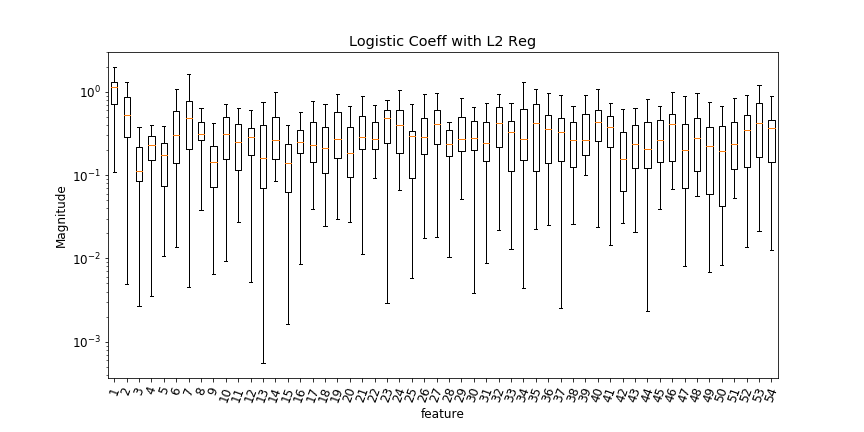


Fig 12. Logistic Regression Coefficients

The most significant feature in our model turned out to be image size, labeled 1 here. While all features had relatively high significance, there was a recurring pattern where the first entry of each quantile value had low impact on our final result. This makes sense as the first quantile value has the tendency to be consistently 0 and has little information contained in them.

### 3.2 K-Nearest Neighbor

We then explored K-Nearest Neighbor (KNN) as a method for classification. A main advantage of the KNN algorithm is that it performs well with multi-modal classes because the basis of its decision is based on a small neighborhood of similar objects (Kim 2012).

Here, we explored the feature space and identified categories based on feature distances between unknown and known examples. We chose the distance metrics as well as the number of neighbors we check at evaluation time as hyperparameters for tuning. The best performing model was one with distance-based metrics and neighbor count at 13, whereby we obtained an average CV accuracy score of 0.3370.

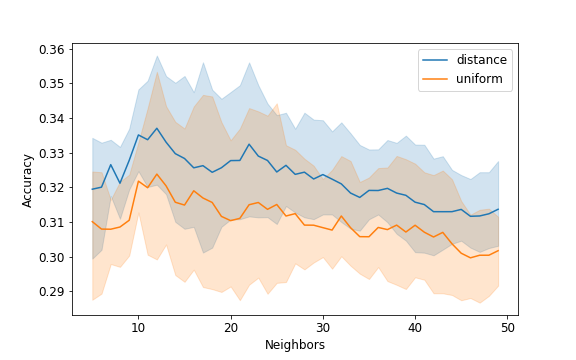


Fig 13. KNN CV Accuracy

From the figure, we noticed that a more intuitive distance-based KNN, where closer neighbors take on larger weights at prediction time, did outperform uniform KNN. The results also seemed to suggest that in this particular case, the best model was built when neighbor count was somewhere between 10 and 20.

### 3.3 Decision Tree and Random Forest

Another class of methods for classification is tree-based. This method had the potential to be very effective in this case, as the features showed little continuity in themselves (Akar 2012). We explored max depth as a parameter to bound the complexity of our tree models. We also tried to fit random forest to see if it helps to reduce variance on validation set.

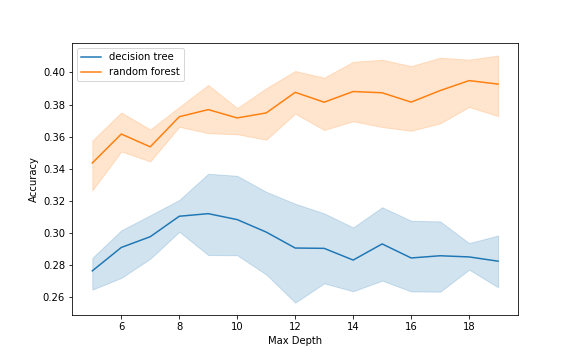


Fig 14. Tree-based Model CV Accuracy

We observed that random forest did perform better consistently due to its ability to reduce variance by combining multiple trees. And the best model was obtained with random forest at max depth of 18 and accuracy score of 0.3948.

### 3.4 Support Vector Machine

Lastly, we tried to use support vector machine (SVM) for classification. SVM performs well on datasets that have many attributes, even when there are only a few cases that are available for the training process (Kim 2012). Similar to logistic regression, a one-over-rest scheme was used to identify each label: essentially finding a plane between data points for each category. This worked surprisingly well, obtaining a score around the same as logistic regression models. We also tried to tune the hyperparameter of penalty regularizer strength here.

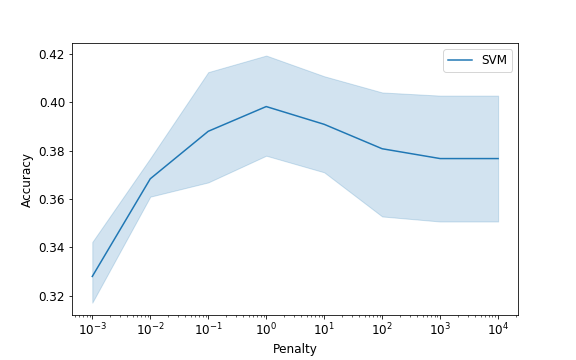


Fig 15. SVM CV Accuracy

We were able to obtain comparable accuracy to logistic regression at 0.3982. We also tried to identify the relevant coefficients again by checking its magnitude.

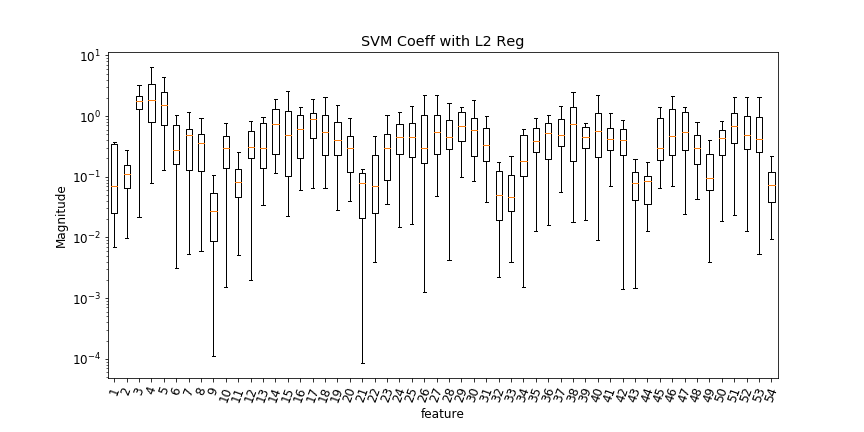


Fig 16. SVM Coefficients

There were some disagreements in valuing various features compared to logistic regression, mainly that RGB mean values were given very high weights, and previously significant feature of image size was no longer that important. But there were similar trends as well. For instance, the first and last quantile values continued to be of little significance.

## 4. Result Analysis

### 4.1 Features of Interest

The features derived from laplacians, the edge images, were surprisingly effective. Prior to adding those features, our model accuracy flattened around 0.3 whereas after adding the mean, variance and quantiles from laplacians, we were able to boost the accuracy score up to around 0.4.

Another surprisingly effective feature was image size. Even though intuitively speaking, image size should have little to do with image content, the boxplot (fig. 3) on image size across categories showed significant variations in its distributions. Furthermore, coefficients (fig. 12) from our best performing logistic model seemed to suggest that image size is a significant predictor as well by having the largest coefficient.

### 4.2 Ineffective Features

Previously, we tried to implement encoded feature such as a shrunken version of the original image and then stretched it out to be a vector input. However, not only did it take significant computation power to run but it also improved model in minisicual ways.

To give a more concrete example, the following steps were performed to obtain a 20x15 shrunken image and then transformed to a 300x1 vector as input into our model. The resulting features did little to improve model performance.

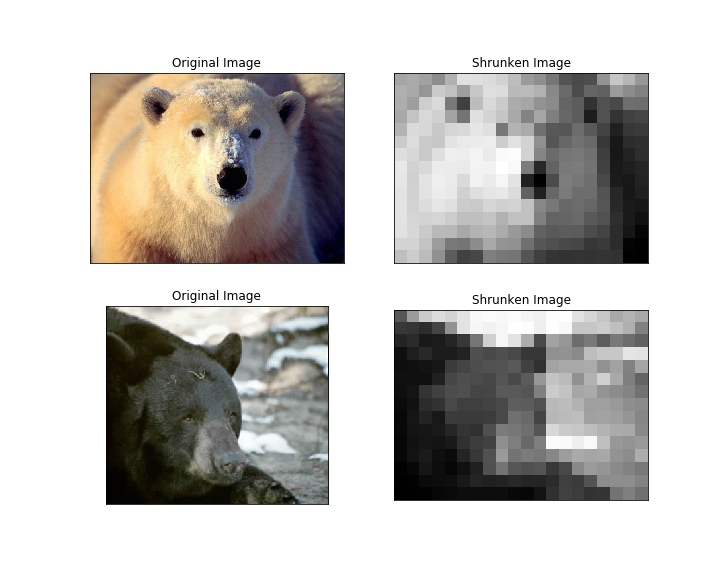


Fig 17. Shrunken Image Comparison

As illustrated, even though the two images belong to the same category, the resulting mapping had little in common. Because our subject can appear in different lighting, position, orientation, and etc., it is not surprising that having a stretched out vector of shrunken image simply added noise to our input.

### 4.3 Final Model Selection

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Logistic Regression | KNN | Decision Tree | Random Forest | SVM |
| Max CV Accuracy | 0.3994 | 0.3370 | 0.3135 | 0.3948 | 0.3982 |
| Hyper-parameters | Scheme = multinomial,  Penalty strength c = 1 | Metric = distance,  N\_neighbor = 12 | max\_depth = 9 | max\_depth = 18 | Penalty strength c = 1 |

Table 3. Max Accuracy Model for Each Method

As illustrated in the table above, we can pick the best model out of all by examining its cross validation accuracies. However, as detailed in plots of section 3, for each method, we plotted the confidence interval of their accuracy scores (ex fig. 11). This is important as a larger confidence interval could mean varying model performance on our final test set. Ideally we want to pick a model that not only has higher average accuracy but also small confidence interval. Therefore, by comparing these two criteria, we selected logistic regression with penalty c=1, and multinomial scheme as our final model at test time.

## 5. Discoveries

### 5.1 Interesting Findings

1. With less strict penalties, CV accuracies tend to diverge as the feature space become less restricted. This can be observed in (fig. 15) of section 3, where larger c values resulted in broader confidence interval bands.
2. We can find edges in image by calculating second derivatives along both the X and Y directions. This was done by utilizing the Laplacian operator provided in OpenCV.

### 5.2 Limitations

1. In feature extracting part, we mainly considered the color-related features. Though we visualized the edges of the image with Laplacian factor, we only used the quantiles, mean and variance values in model training. Due to the limitation of scale, we didn’t analyze the location of edges.
2. There could be feature-to-feature correlations and may negatively impact our test set accuracy. One possible fix is to use dimensionality reduction techniques such as Principal Component Analysis before applying models.

### 5.3 Future Works

1. Feature Encoding with Convolutional Neural Networks (CNN):

One decided disadvantage in our implementation was that we did not make use of the spatial information inherent in images, as we only looked at pixel value information. Therefore, one way to mitigate this issue is to use feature encoding from convolutional neural networks (CNN). By applying CNN first, we could extract higher level features from images first which make use of spatial information of images. With those higher level encoded features, we can then extract scalar variables for predictions.

1. We could also make use of identifying features in images. By using feature detection algorithms such as Oriented FAST and Rotated BRIEF (ORB).
2. We can augment our training dataset by rotating, cropping and shifting our original images to create more training examples for our model.

## Reference

[1] Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary R. Bradski: ORB: An efficient alternative to SIFT or SURF. ICCV 2011: 2564-2571.

[2] Foody, Giles M., and Ajay Mathur. "A relative evaluation of multiclass image classification by support vector machines." IEEE Transactions on geoscience and remote sensing 42, no. 6 (2004): 1335-1343.

[3] Akar, Özlem, and Oğuz Güngör. "Classification of multispectral images using Random Forest algorithm." Journal of Geodesy and Geoinformation 1, no. 2 (2012): 105-112.

[4] Kim¹, J. I. N. H. O., B. S. Kim, and Silvio Savarese. "Comparing image classification methods: K-nearest-neighbor and support-vector-machines." In Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics, vol. 1001, pp. 48109-2122. 2012.

[5] Cheng, Qi, Pramod K. Varshney, and Manoj K. Arora. "Logistic regression for feature selection and soft classification of remote sensing data." IEEE Geoscience and Remote Sensing Letters 3, no. 4 (2006): 491-494.