DS200A Final Project: Image Detection

December 13, 2019

Emeri Zhang, Nicolas Kardous

Abstract:

The purpose of the project is to make use of everything we learned throughout the semester. The project goes through the entire data lifecycle of an image classification problem. Specifically, we are building different models to make a prediction on what an image is of. We are training our model for 20 different images (classes) that ranges from animals to planes. The entire process involves data input and data processing, exploratory data analysis and feature extraction, classifier training, and a classifier performance assessment. We are building six different models that include support vector machine, k-nearest neighbor, logistic regression, classification trees, random forest and convolutional neural networks. We found that our SVM model performed the best with an accuracy of 0.4286, and then our Random Forest with an accuracy of 0.4252. We concluded that corner features and key points were good features, while some models like BRIEF performed better than ORB.

Introduction:

In this project, we were tasked with building an image classifier for a training dataset with 1501 images of 20 classes, then running our predictor on a testing dataset of 716 images. After reading images into a dataframe and crafting features, we conducted exploratory data analysis on the feature frame, visualizing distributions over each class. We then explored various machine learning techniques and assessed predictor performance with scikit learn, including Logistic Regression, kNN, classification trees, random forest, and support vector machines. Finally, we built a convolutional neural network CNN with keras.

Implementation:

Library versions: cv2 version 3.4.2, scikitlearn 0.21.3, pandas 0.25.3

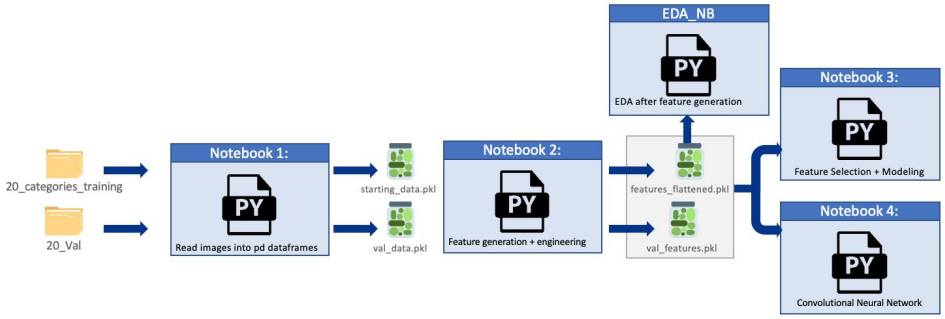


Figure 1: File Paths to show the different notebooks, Notebook 1 is where the images are read into a datagrame, Notebook 2 is for feature generation and engineering, EDA_NB is for Explanatory Data Analysis, Notebook 3 is for Feature Selection and Modelling, and Notebook 4 is for Neural Networks

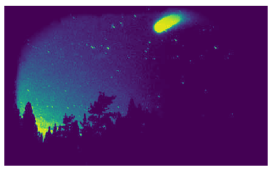
Data Ingestion and Preprocessing

We were provided a dataset of 1501 images of 20 classes. We read these into a dataframe of shape 1501, 2 with column **image** containing a 3-d array, with dimensions row pixels, column pixels, RGB color intensity (between 0 and 28 − 1). Column **label** is the class label encoded alphabetically, with the following mapping:

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: Shows the different images, and the label number for each class of image

We note 15 images are grayscale and lack RGB channels and thus we see a dimension mismatch with representation in a 2-D array. We note 12 of these images are class 3, or comet, making up 15% of the comet training images -- too many to drop. We impute missing RBG channels with cv2.



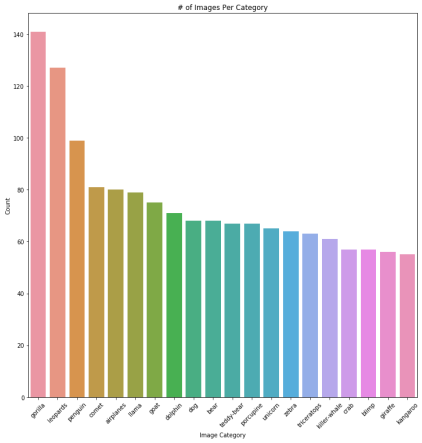
Before: missing RBG

After: imputed

Figure 2: Image on the left shows original comet image missing RBG channel, and image on the right shows the comet image imputed

Feature Engineering:

As a cursory exploration, we view the 1) image count over classes to identify any imbalances, and 2) average image size (# of pixels). We note 1) gorillas, leopards, and penguins dominate the set, while remaining classes have roughly uniform distribution, 2) penguins, bears, and zebras images are largest



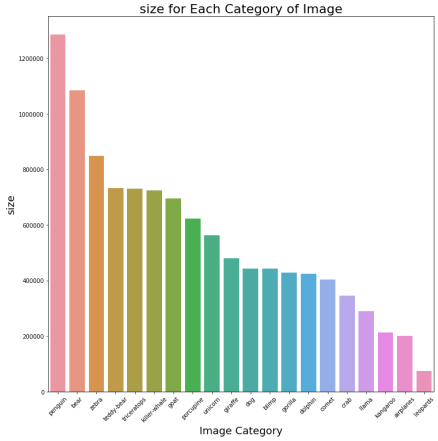


Figure 3: Histogram on the left shows the number of images in each category, and image on the right shows the size of the image in each category

Based on literature and documentation on openCV, we selected 26 features, listed below.

Feature Name	Description
size	Total pixel size
aspect_ratio	aspect ratio of the image
r_mean	Mean for the red channel intensity
r_std	Standard deviation pixel value for red channels
g_mean	Mean for the green channel intensity
g_std	Standard deviation pixel value for green channels
b_mean	Mean for the blue channel intensity
b_std	Standard deviation pixel value for blue channels
luminance_mean	Mean of luminance (Y)
luminance_std	SD of luminance (Y)
cb_mean	Mean of blue chroma component (Cb)
cb_std	SD of blue chroma component (Cb)
cr_mean	Mean of red chroma component (Cr)
cr_std	SD of red chroma component (Cr)
fast_corner	FAST Algorithm for Corner Detection, returns # of key points
brief	BRIEF (Binary Robust Independent Elementary Features), returns # of key points
orb	ORB (Oriented FAST and Rotated BRIEF), returns # of key points
sift_kp	SIFT (Scale Invariant Feature Transform), returns # of key points
surf_kp	SURF (Speeded-Up Robust Features), returns # of key points
canny_edges	Canny edge detector that returns the edge gradient; sum
prewitt_h	Prewitt operator for finding edges in the horizontal direction; sum
prewitt_v	Prewitt operator for finding edges in the vertical direction; sum
binarize	Binarization of grayscale image, returns matrix of 0 or 1; mean
harris	Harris corner detector returns a matrix for the corners extracted; mean
shi_tomasi	Shi Tomasi corner detector returns a matrix for the corners extracted; mean
label	Class labels, encoded 0-19

These features fall in 3 major domains: 1) color channels 2) corner and edge detection 3) key point detection

Table 2: Feature metadata, shows the 25 different features extracted and color coded on the domain it falls under

Color Channels: Upon a quick skim through the image files, we note some classes have different prevalent colors i.e. comets lack RGB, dolphins and whales are more dominated by the blue of water, unicorns have a more "sparkly" property. Thus, we decided to calculate the mean and standard deviation for channel values, and do the same for Y (luminance), Cb (chroma blue), and Cr (chroma red). Note we also binarized a grayscale version of the image, in case it is useful to eliminate the concept of color altogether and revert to grayscale only.

Corner and Edge Detection: When we perceive images, our brain processes colors and shapes. A first instinct was to understand how to distinguish boundaries on images, and after reading literature, we understood that from a computer's perception of an image, an edge is where there is a change in color, and corners are point-like features of images that are specific points of interest. There are many corner and edge detection algorithms available, and we chose a few commonly used ones from openCV. Below, we show canny edges with various sigma values, and we chose 2.

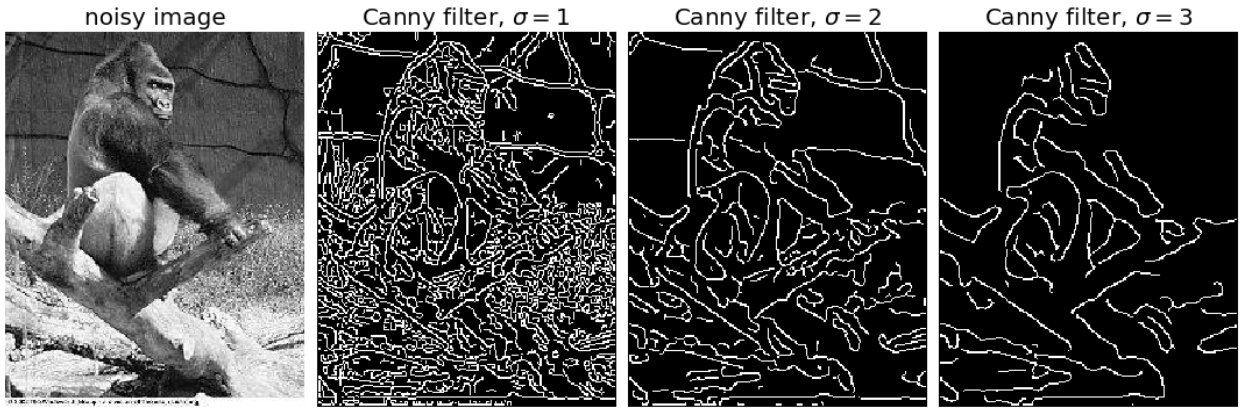


Figure 4: Canny images shown for different sigma values

For edge detection features, we took the sum of flattened arrays due to the sparsity of the feature matrix. For corner detection features, we chose mean due to denser feature matrices.

Key Point Detection: Besides corners, other interesting points exist as spatial locations or points on an image, which might define areas that help a specific class stand out and thus be more easily identifiable. Below we show an original test image, followed by plots of key points from BRIEF, ORB, and SIFT key point detection algorithms.

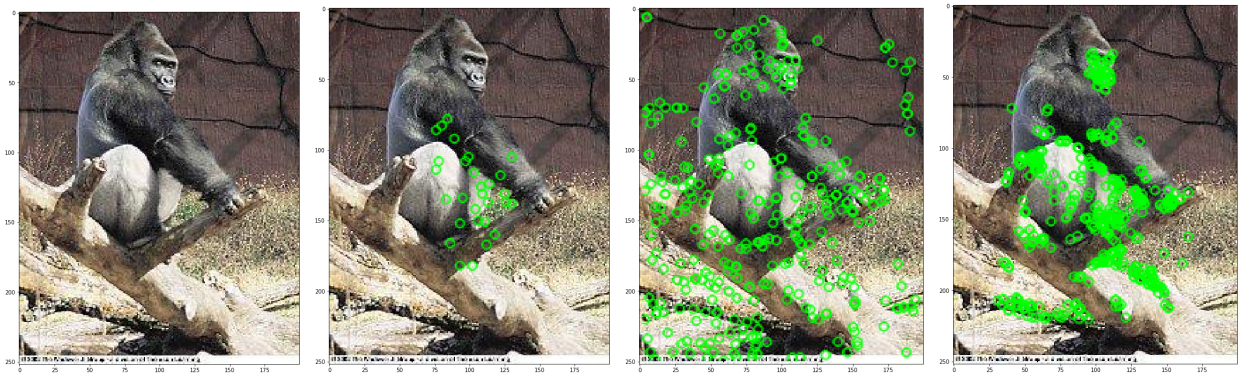


Figure 5: First image shows the original test image, the second image is the key points from BRIEF, third image is the key points from ORB, and fourth image is the key points for SIFT

For key point detection features, we returned the length of the key points array, yielding a scalar number as the feature.

Note: Several feature detection algorithms yielded multi-index arrays of varying sizes, which is not handled when modeling in scikitlearn. To ensure feature matrices align for all images in the learning set, we flattened the features and summarized as best fit.

EDA:

After generating the feature frame, we created visualizations to explore features across the 20 classes.

1) We plotted seaborn barplot of the **means in descending order** to get a general summarized idea of which classes have smaller or larger feature values.
2) We created boxplots to **identify quartiles and observe any outliers** across classes.
3) We created violin plots **exploring distribution and variance and skew across classes**.

Some interesting findings are summarized below:

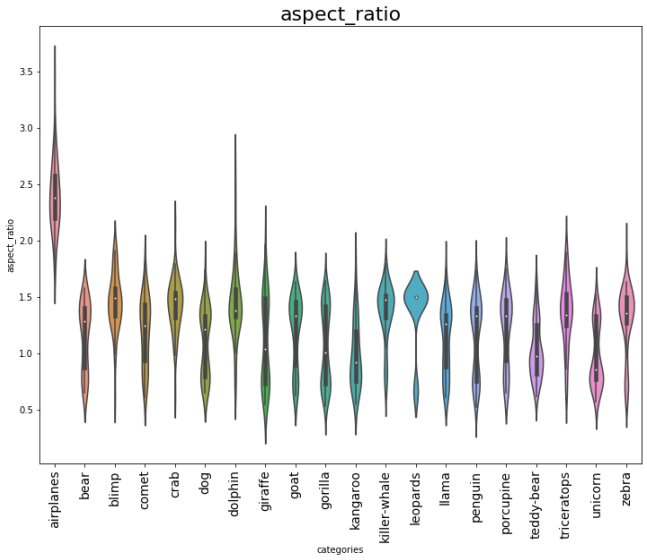


Figure 6: Aspect ratio across the different label

The image above depicts that airplanes have a higher aspect ratio compared to the aspect ratios of the other labels.

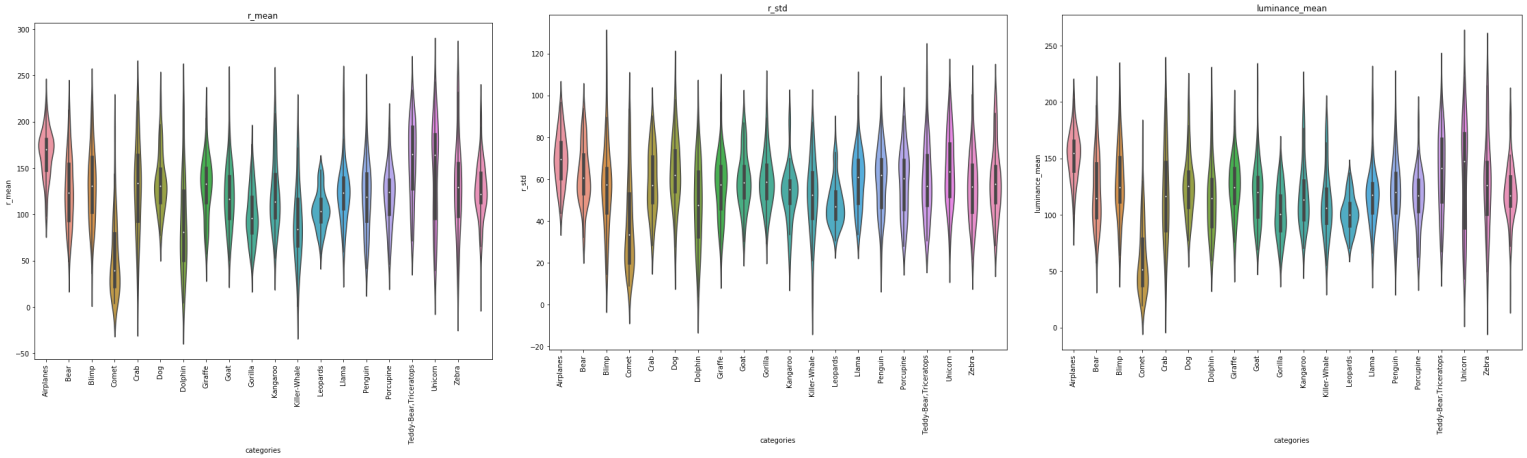


Figure 7: First image is the red channel mean across all labels, second image is red channel std across all labels, and third image is the luminance mean across all labels

The three images above show that comets (fourth violin plot from the left) have a low **r_mean**, low **r_std**, low **luminance_mean**, and also a low g_mean.

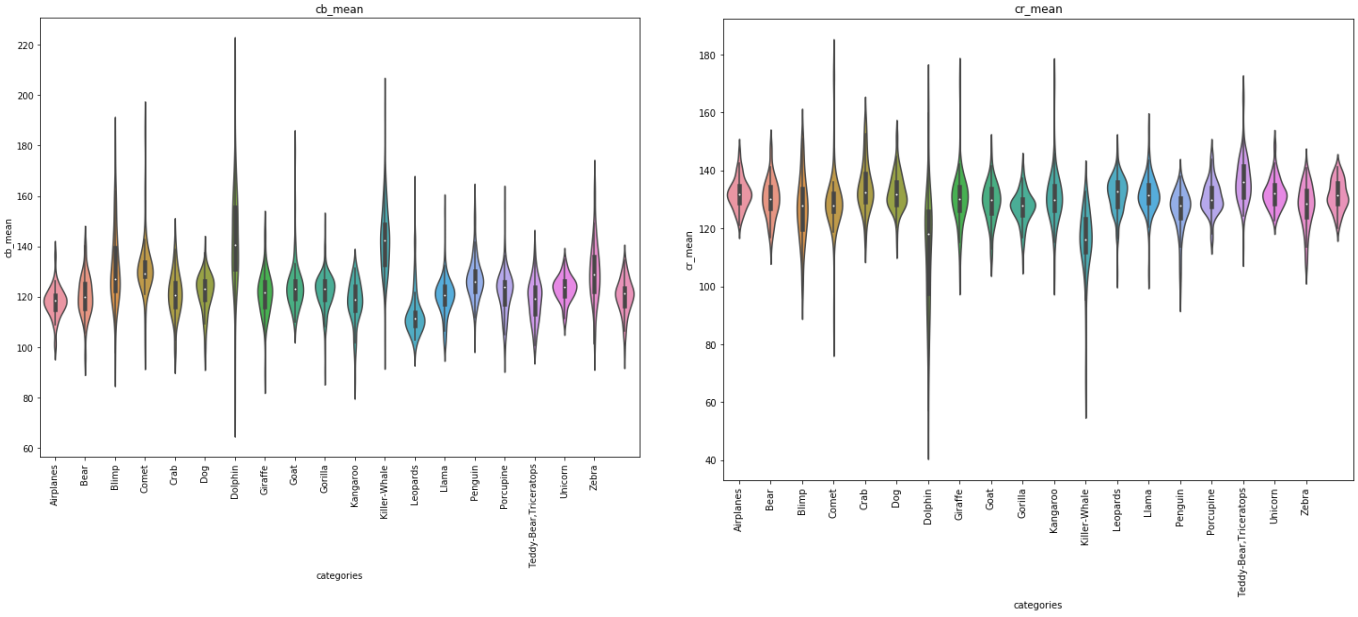


Figure 8: First image is the blue chroma component mean across all labels, second image is the red chroma mean across all labels

Three images above show that comets (fourth violin plot from the left) have a low r_mean, low r_std, low luminance_mean, and also a low g_mean. It is also found that comets and leopards also have a low b_mean and a low b_std. Dolphins and killer-whales tend to have higher cb-mean and lower cr-means. This makes sense because dolphins and whales are aquatic creatures, and these images should have a heavier blue tone because they are taken in the water.

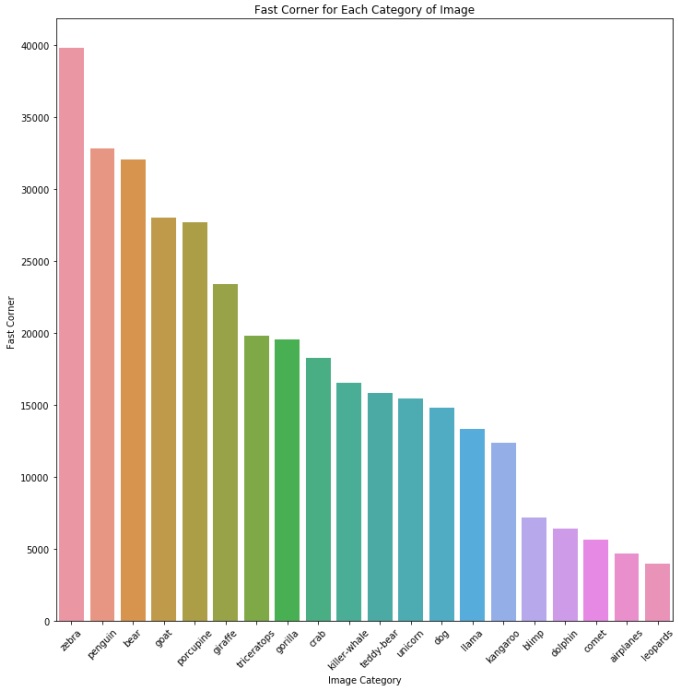


Figure 9: Number of keypoints across all labels from Fast_Corner method

The image above shows the number of key points for each label from using the FAST algorithm in corner detection. There is a clear variation among the different labels.

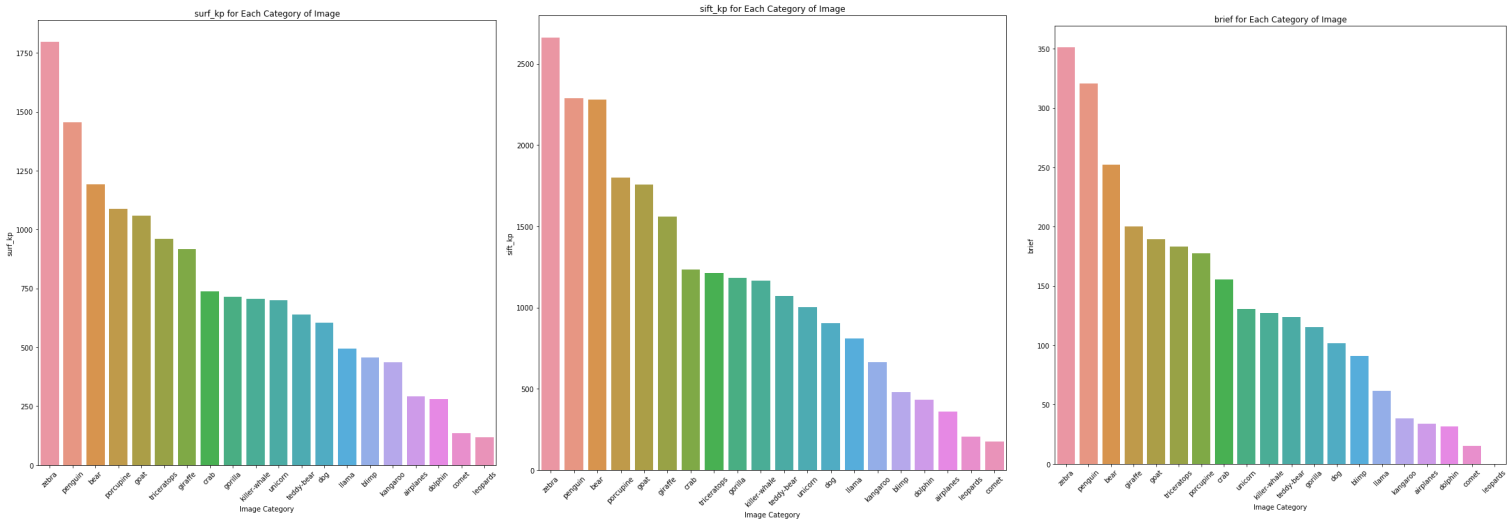


Figure 10: First image shows the number of keypoints returned using SURF method. Second image shows number of keypoints returned using SIFT method. Third image shows number of keypoints returned using BRIEF method

The image above shows the number of key points for each label from using the SIFT, SURF and BRIEF feature. Similar to the Fast_corner feature, there is a clear trend where each label returns a specific range of keypoints.

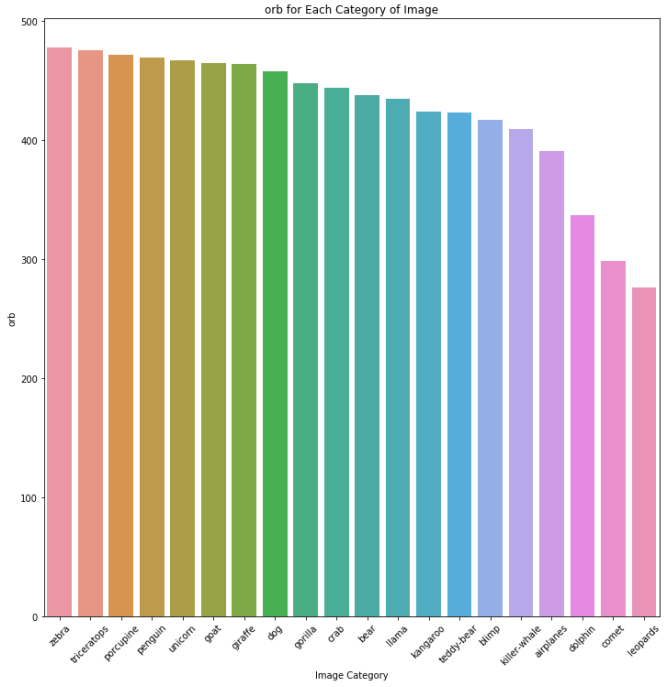


Figure 11: Number of keypoints returned across all labels, from the ORB method

The image above shows the ORB feature that returns the number of keypoints for each of the labels. ORB is a more efficient alternative to SIFT or SURF for detecting keypoints, and we were expecting the histogram to follow a similar trend to the SIFT, SURF and BRIEF features. However, these trends seem to be more incremental and less distinguishable. There are fewer keypoints, which can be seen from the figures under Keypoint Detection.

Summary of Results:

```
SVM Model accuracy is:  0.42857142857142855
KNN Model accuracy is:  0.38205980066445183
Decision Tree Model accuracy is:  0.30564784053156147
RFM Model accuracy is:  0.42524916943521596
Logistic Model accuracy is:  0.4152823920265781
```

Figure 12: Accuracy values across all the models

Discussion:

The most interesting features that our group noticed was Fast_corner, SIFT, and BRIEF. We believe this because when graphing the histograms across all the labels, we notice a clear variance among the different labels. Not only that, across Fast_corner, SIFT and BRIEF, all labels follows a similar distribution. For example, zebra, bear and penguins always return the highest number of keypoints, and comets, leopards and airplanes always return the lowest number of keypoints. This also makes sense, because from our classification tree model, the highest accuracy received after using cross validation was found when utilizing all the features up to BRIEF. Moreover, another feature that we found very interesting is the harris feature and the shi_tomasi feature. This is because across all the models, the accuracy has increased to the maximum when including the harris and shi_tomasi features. Additionally, from the Random Forest model, after cross validation, the highest accuracy was found for including features up to the Harris feature. Finally, the last useful feature we found was the canny_edges feature that returns the sum of all the edge gradients. This is because including this feature improved the accuracy across all our models, and the SVM model had the highest accuracy for all the features up to the canny_edges feature.

As mentioned above, we thought ORB feature would be useful because it returns the number of keypoints. However, from figure 11, we see that there is not a great amount of variation with the number of keypoints that the orb feature returns. Additionally, when going through cross validation and computing the accuracies for the successive features, we found the minimum increase in accuracy for the logistic regression model were from including the features prewiit_h and prewitt_v. I believe this makes sense because these two features returns the sum of a flattened matrix, and these sum values might not be too different across the different labels. In addition, from the classification tree and SVM models, it is found that the minimum increase in accuracy is found when the mean luminance and std_luminance features are included. Finally, from the random forest model, the minimum increase in accuracy is found when the ORB feature is added. From the histogram above, we have shown this as there are incremental differences in the number of keypoints returned across the labels.

The models we have utilized are logistic regression, K nearest neighbor, Classification tree, random forest, and support vector machine. The reason why we used these models is because we are training our models on classification. For example, we have 20 different labels, which our classification model will choose from. Our most effective models turned out to be the SVM model, the Random Forest model, and the Logistic Regression model, respectively.

With regards to the SVM, we chose to use a linear kernel function. It would make sense why the SVM model performed well, because it is able to create 20 different linear hyperplanes for the 20 different labels. SVM models work relatively well when there is a clear margin of separation between classes. Additionally, SVM is effective in high dimension spaces and is relatively more memory efficient.

Random Forests had the second highest accuracy because it is a rather greedy method. Greedy in the sense that the models make the most optimal decision on each step of the tree. Benefits to the random forest is that it is invariant to feature scaling and translation. It also does automatic feature selection, there are non-linear decision boundaries without complicated feature engineering, and it does not overfit as often as other nonlinear models. The biggest benefit to the Random Forest is that it is an ensemble method, and takes the average across thousands of individual trees.

Because random forest utilizes ensemble learning, we could see why it performed better than the decision tree classifier. For example, as the decision tree classifier is also a greedy method, it would choose the most optimal steps at each node of the tree to have the lowest impurity cost. Due to this, the decision tree classifier would overfit on the training set, and thus not perform as well when predicting on the testing set. On the other hand, because random forest takes the average across many trees, it ultimately reduces the variance of the model.

The reason why the KNN accuracy is not as high as the others is because the model suffers from the curse of dimensionality. KNN works well with a small number of features, but struggles to predict as there are more features. KNN also requires to choose the optimal number of neighbors, it is sensitive to outliers as it chooses the neighbor based on the distance criteria, and requires an initial parameter of choosing the optimal number of neighbors.

Evaluation:

Overall, our group has been very satisfied with our approach in exploratory data analysis and feature extraction. We were able to find up to 25 features, which was more than enough to give us an accuracy above 35%. Additionally, in the EDA, the plots we used very well depicted a clear difference among the labels. For example, the SIFT, SURF and ORB methods showed a clear variation in the number of keypoints across the different labels. This made us realize that keypoints and corner detection are really important features in image classification.

One surprising discovery that we found was that there was less variation in the key points across the labels from the ORB method. This was interesting for us because the ORB method is supposed to be a more efficient method compared to SIFT or SURF. One limitation in our approach was that for some of our features that were matrices, we transformed them to a scalar by either summing all the terms in the matrix, or finding the meaning of them. For example, the Harris and Shi_tomasi methods returns a matrix of corner detection, and we chose to find the mean of the matrix. However, two different labels could return similar means, but have vastly different corner detections. Thus, next time, if had smaller matrices that were the same size across all the observations, we could make each scalar in the matrix into a feature.

In the future, our group hopes to better improve the quality of the dataset, as there were some issues of mislabeling. As neural networks are the industry standard in image classification, our group also wants to consider different forms of neural networks. For example, in the future, we are considering implementing GAN (Generative Adversarial Networks) and RNN (Recurrent neural networks) with LSTM (Long Short Term memory).

Sources:

"1.10. Decision Trees¶." *Scikit*, Scitkit-Learn, 2019, scikit-learn.org/stable/modules/tree.html.

"1.4. Support Vector Machines¶." *Scikit*, Scitkit-Learn, 2019, scikit-learn.org/stable/modules/svm.html.

"3.2.4.3.1. Sklearn.ensemble.RandomForestClassifier¶." *Scikit*, Scikit Learn, 2019, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

"3.2.4.3.1. Sklearn.ensemble.RandomForestClassifier¶." *Scikit*, Scitkit-Learn, 2019, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

Agarwal, Yash. "Create Your First Image Recognition Classifier Using CNN, Keras and Tensorflow Backend." *Medium*, Nybles, 24 May 2019, medium.com/nybles/create-your-first-image-recognition-classifier-using-cnn-keras-and-tensorflow-backend-6eaab98d14dd.

"Feature Detection and Description¶." *Feature Detection and Description - OpenCV 3.0.0-Dev Documentation*, Open CV, 10 Nov. 2014, docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html#py-table-of-content-feature2d.

Grogan, Michael. "Image Recognition with Keras: Convolutional Neural Networks." *Medium*, Towards Data Science, 26 Oct. 2019, towardsdatascience.com/image-recognition-with-keras-convolutional-neural-networks-e2af10a10114.

Gupta, Vikas. "Image Classification Using Convolutional Neural Networks in Keras." *Learn OpenCV*, 29 Nov. 2017, www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/.

Keras-Team. "Error When Checking Model Target: Expected activation_2 to Have Shape (None, 10) but Got Array with Shape (3, 1) · Issue #3109 · Keras-Team/Keras." *GitHub*, Github, github.com/keras-team/keras/issues/3109.

McCaffrey01/02/2019, James. "Image Classification Using Keras." *Visual Studio Magazine*, 2 Jan. 2019, visualstudiomagazine.com/articles/2018/12/01/image-classification-keras.aspx.

rohanpillai20. "rohanpillai20/Image-Classification-by-Keras-and-Tensorflow." *GitHub*, Githubb, github.com/rohanpillai20/Image-Classification-by-Keras-and-Tensorflow/blob/master/Using%20Keras/Training.py.

"Sklearn.linear_model.LogisticRegression¶." *Scikit*, Scikit-Learn, 2019, scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

"Sklearn.neighbors.KNeighborsClassifier¶." *Scikit*, Scitkit-Learn, 2019, scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

Thoma, Martin, and Martin ThomaMartin Thoma. "Are There Any Image Classification Algorithms Which Are Not Neural Networks?" *Data Science Stack Exchange*, 1 Sept. 1966, datascience.stackexchange.com/questions/13231/are-there-any-image-classification-algorithms-which-are-not-neural-networks.

Zhou, Victor. “Training a Convolutional Neural Network from Scratch.” *Medium*, Towards Data Science, 6 June 2019, towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754.

Zhou, Xuan, and Jiajun Wang. “Feature Selection for Image Classification Based on a New Ranking Criterion.” *Journal of Computer and Communications*, vol. 03, no. 03, 2015, pp. 74–79., doi:10.4236/jcc.2015.33013.