Image Classification
A Short Survey on Commonly Used Techniques

Sihao Chen*[1], Cem Koc*[1]

Abstract— **This report examines the performance of some of the most widely available image processing techniques and classifiers for general multi-class classification on a dataset containing 20 labels. We have trained both neural net and non-neural net based classifiers. The images are processed and featurized using a variety of image processing techniques after doing an exhaustive exploratory data analysis. Afterwards, these feature vectors are used as inputs to the classifiers which are trained using 5-fold cross validation. Although neural nets have become the industry standard for tackling this task, we show that with feature engineering one can achieve sensible accuracies using conventional classifiers. Among the non-neural net based classifiers that fit well to our training set, gradient boosting trees (using XGBoost) has attained the highest testing accuracy (51%) with logistic regression and SVM coming in second (46%). Remaining non-neural classifiers: K-nearest neighbors, decision tree classifier and random forests, perform significantly worse. Finally, we further confirm that neural nets achieve significantly better than all of our classifiers at 83%.**

I. INTRODUCTION

The task of image classification has rightfully drew the attention of many researchers and industry professionals alike for decades. Since images are such a rich medium, the real-world applications of classification tasks have made significant contributions to technology and the way we live today from cancer diagnoses, to detecting lanes for autonomous vehicles and to tracking animals in their habitats. Since the problem domain is so rich, so is the techniques to achieve it. In an effort to explain the wide-variety of tools and techniques available for image classification we feel compelled to write this report to summarize a very short sample of supervised techniques available to the modern-day researchers, students and industry.

[1]Sihao Chen and Cem Koc are with the Department of Electrical Engineering and Computer Sciences, `sihao@berkeley.edu`, `cemkoc@berkeley.edu` University of California, Berkeley, CA 94720 USA.

*These authors contributed equally to this work.

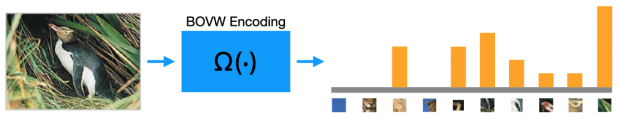

Fig. 1. Bag of Visual Words encoding for an image with dense spatial features represented as a histogram from a (partial) visual vocabulary.

The structure of the report is as follows:
In section II we introduce the dataset, pre and post cleaning procedures and the exploratory data analysis performed.
In section III we introduce the feature engineering and regularization steps performed. We introduce the hand engineered features and the Bag of Visual Words (BOVW) featurization steps. Furthermore, we discuss our methods for training the non-neural (conventional) classifiers as well as the neural net based classifier. Here we also talk about the results achieved using our neural net classifier.
In section IV we describe results achieved in all of our conventional classifiers.
And finally in section V we summarize our report and talk about the future steps.

II. DATASET

In this section we dive deeper into the dataset and our exploratory data analysis as well as our data cleaning methods.

A. *Dataset Morphology*

There are two datasets: a labeled dataset for training and validation and an unlabeled dataset for testing. The labeled dataset is located in folder "20_categories_training". Images in each category are located in a folder with label as its name. The unlabeled dataset is located in folder "20_Validation".

The dataset for training and validation consists of 1,501 images and 20 different types. Images are of different sizes (i.e., numbers of pixels). There are 1,485 RGB images and 16 grayscale images. RGB images are represented as 3-d arrays with the first two dimensions corresponding to the row and column pixels and third dimension to the color, and each value corresponds to a red, green, or blue (RGB) color intensity between 0 and $2^8 - 1$. The third dimension size is always 3. Grayscale images are represented as 2-d arrays with the dimensions corresponding to the row and column pixels. Each value corresponds to intensity between 0 and $2^8 - 1$.

B. *Data Cleaning and Exploratory Data Analysis*

We want to train and evaluate our classifiers using the provided dataset. We cannot use the unlabeled dataset to test the accuracy of our classifiers. Also, cross-validation error alone is not enough to compare the models, as we tune the hyperparameters on the dataset. Thus, we split the labeled dataset randomly to a training set, which contains 80% of the labeled images, and a testing set, which contains 20% of the labeled images.

In figure 2, we use a boxplot and a histogram to visualize the distribution of size, width, and height of the images. The sizes of images have a large variance. The largest image is over 500 times as large as the smallest one. The medium size of the images is around $300,000$ and the first quartile is slightly smaller than $200,000$ ($\approx 256 \times 256 \times 3$).

As in figure 3, we use a histogram to visualize the distribution of average pixel intensities of images over all channels. We found that the distribution is unimodal, and the mode is around 120. The distribution is similar to normal distribution, but is slightly right skewed.

We use the barplot in figure 4 to visualize the distribution of class frequencies. We found class 9 (gorilla), class 12 (leopards), and class 14 (penguin) are the most frequent class, while class 2 (bear) and class 10 (kangaroo) are the least frequent classes.

III. METHODS

A. *Feature Engineering and Regularization*

The dataset contains images with different encodings, specifically RGB and greyscale images. In an effort to standardize we convert the greyscale images to RGB images by duplicating its intensities in each channel. The shapes and values of some features heavily depend on image size, and we do not want the sizes of our images to decide them sometimes (not always). We resize images before extracting these features. We choose (256, 256) as the destination image size, smaller than around 75% of the images.

We classify features into two types: *global features*, which quantify an image as a whole, or *local features*, which quantify local regions in an image [1]. Global features have the ability to generalize an entire object with a fixed-sized single vector. Consequently, they can be used directly in our classifiers. Local features, on the other hand, are computed at multiple points in the image and are consequently more robust to occlusion and clutter. However, the shape of these local features vary with the sizes and shapes of the images themselves. Moreover, their sizes grow with the number of artifacts in the images therefore we discovered that it is impractical to use them as-is for inputs to classifiers because the features' dimensionality will make the classifiers prone to overfitting and sensitive to input noise. In order to efficiently grab local features that are size and rotation invariant, we instead utilize an encoding called Bag of Visual Words (BOVW) and collect additional feature information this way [2], [3]. Bag of Visual Words is a commonly used technique inspired from the bag of words model often found in the natural language processing (NLP) literature. Simply put, it aims to encode context aware, discrete visual features from images which then could be used to build a "visual vocabulary" to describe the image as a histogram of the learned features. We discuss the BOVW featurization process in III-A.9 in greater detail.

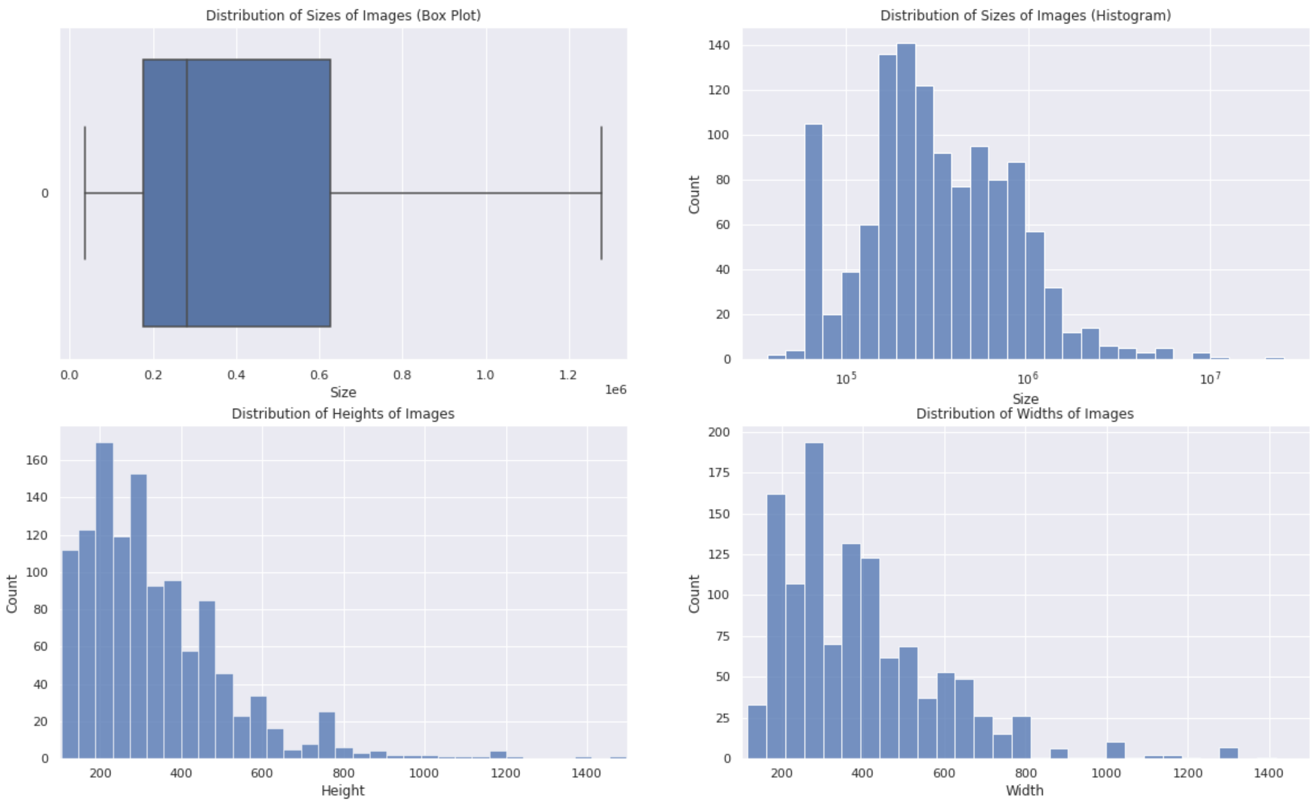

Fig. 2. Distribution of size, height, and width of the images

We extract the following features from each image:

1) A scalar representing the pixel size of the image.
2) A scalar representing the aspect ratio of the image.
3) A scalar representing the average of the red-channel intensity for the images.
4) A scalar representing the average of the green-channel intensity for the images.
5) A scalar representing the average of the blue-channel intensity for the images.
6) A scalar representing the standard deviation of the red-channel intensity for the images.
7) A scalar representing the standard deviation of the green-channel intensity for the images.
8) A scalar representing the standard deviation of the blue-channel intensity for the images.
9) A scalar representing the proportion of the pixels where red-channel intensity is higher than green- and blue-channel intensities for the images.
10) A scalar representing the proportion of the pixels where green-channel intensity is higher than red- and blue-channel intensities for the images.
11) A scalar representing the proportion of the pixels where blue-channel intensity is higher than red- and green-channel intensities for the images.
12) A scalar representing the mean value of Harris Corner Detection Results on the image [4].
13) A scalar representing the standard deviation of Harris Corner Detection Results on the image.
14) A scalar representing the number of corners returned by Shi-Tomasi method [5].

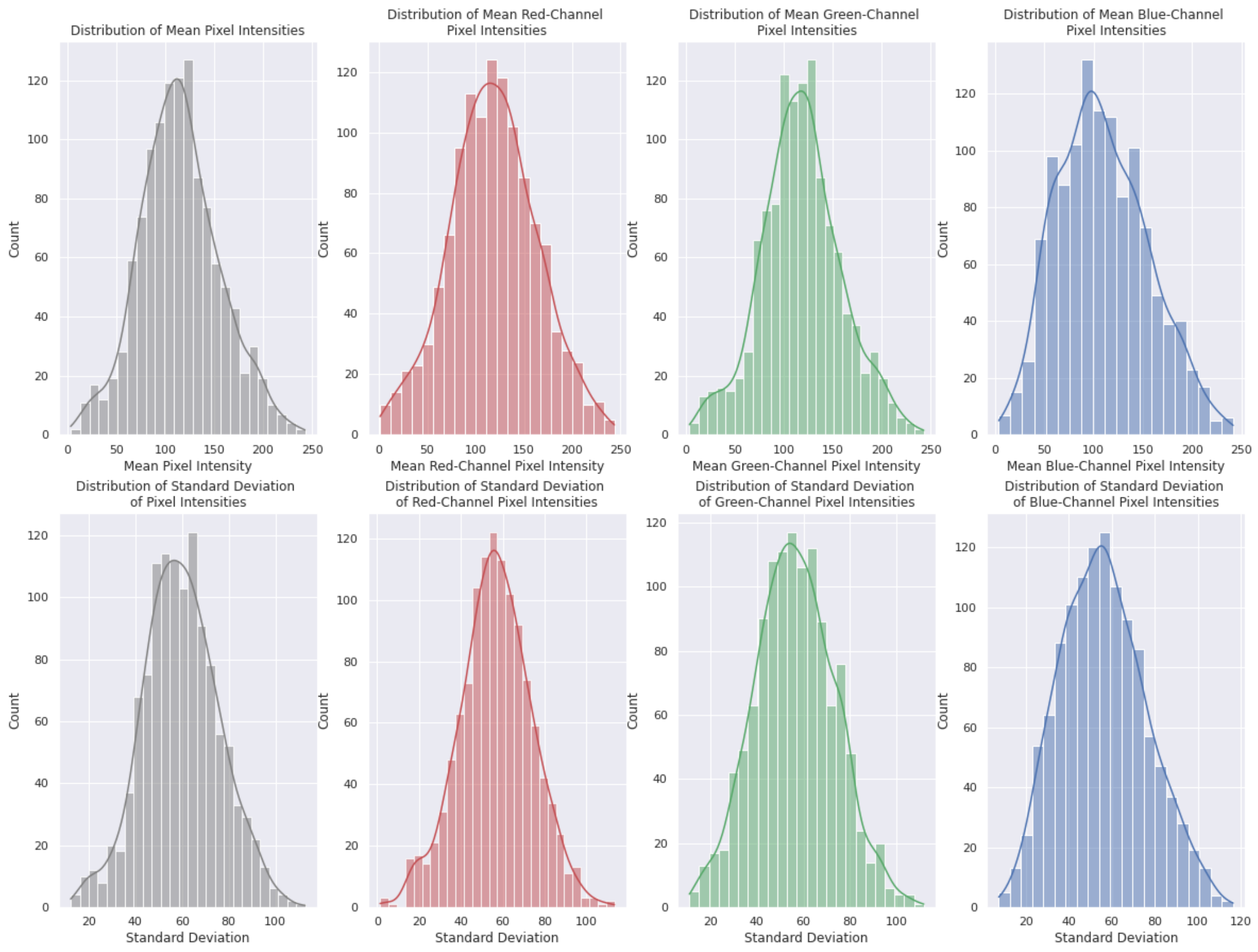

Fig. 3. Summary of Pixel Intensities

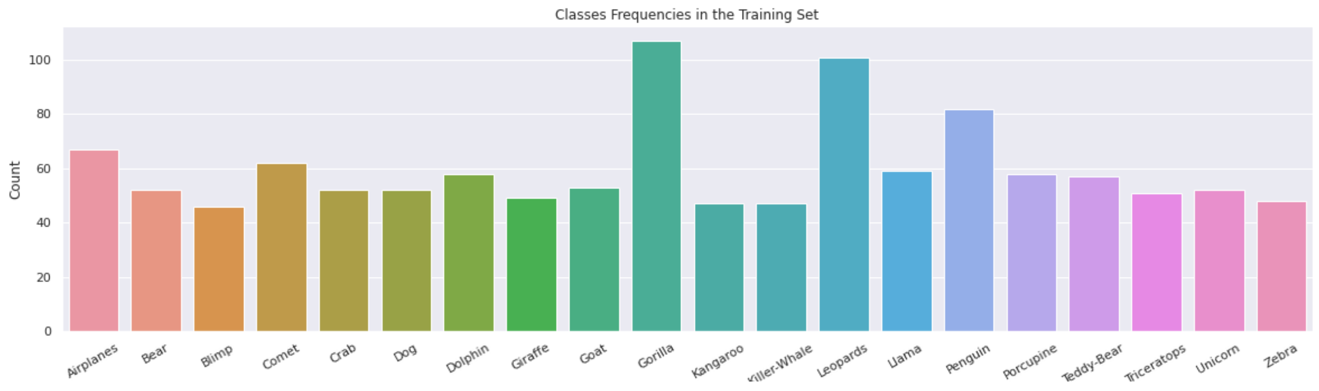

Fig. 4. Class frequencies in the original training set (Best viewed in color)

15) A 64-component vector representing the color histograms of the image.
16) A 7-component vector representing the Hu moments of the image [6].
17) A 14-component vector representing the Haralick Texture of the image [7].
18) A 49-component vector representing the Zernike moments of the image [8].
19) A 162-component vector representing the threshold adjacency statistics (TAS) of the image [9].
20) A 250-component vector representing the histogram of SIFT descriptors [10] after being processed by Bag of Visual Words [2].

1) Pixel Size, Intensity, and Aspect Ratio: We use pixel size, aspect ratio, and mean value and standard deviation of pixel intensities in three channels as features. In addition, for each channel, we calculate the proportion in the image where intensity in this channel is higher than intensities in other channels. This could reflect the relative importance of a channel despite different brightness of images.

We use the box plots in figure 5 to show the pixel sizes, average pixel intensity of red channel, and aspect ratio of images in each class. We can see that most pictures in classes like airplanes and leopards have fewer pictures compared to others. This phenomenon could be accidental, but if the images in the training and testing set are from the same source, these features could help us with the classification. On the other hand, We can see that pictures belonging to the comet and killer whale categories have a low red-channel intensity on average. This could be explained by the dark sky and dark skin of the whales. Similarly, The large aspect ratio of images labeled as planes could be caused by the shape of the planes. These features are useful even outside our dataset.

2) Harris Corner Detection: We apply Harris Corner Detection [4] to the images. The initial results of Harris Corner Detection are grayscale images where intensities are scores to determine whether a pixel is a corner or not. We extract the mean and standard deviation of these scores on each image.

3) Shi-Tomasi Corner Detection: Shi-Tomasi Corner Detection [5] is a small modification on Harris Corner Detection that made the results better. Instead of returning an image, it returns a set of locations of detected keypoints. We use the number of keypoints detected by this method as a feature.

4) Color Histogram: A color histogram represents the distribution of colors in an image. It not only can be visualized as a graph that gives a high-level intuition of the intensity distribution, but also can function as a feature in machine learning. We split the colors according to the combination of intensities in three channels into $4^3 = 64$ bins. For each image, there are 64 columns representing the proportion of each color in the image.

5) Hu Moments: Hu Moments [6] of an image are a set of seven numbers calculated using central moments that are invariant to image transformations. The first six moments have been proved to be invariant to translation, scale, and rotation, and reflection. While the seventh moment's sign changes for image reflection. We directly use Hu Moments as seven columns in our design matrix.

6) Haralick Texture: Haralick texture features [7] are 14 global texture features based on the adjacency matrix. This matrix is square with dimension N_g, where N_g is the number of gray levels in the image. Element $[i, j]$ of the matrix is generated by counting the number of times a pixel with value i is adjacent to a pixel with value j and then dividing the entire matrix by the total number of such comparisons made. We use the implementation in Mahotas [11], where 13 Haralick texture features are returned. This add 13 columns to our design matrix.

7) Zernike Moments: Zernike Moments [8] are a global measure of how the mass is distributed. The magnitudes of Zernike moments were used as features. This provided 49 descriptive features for each image.

8) Threshold Adjacency Statistics: Threshold adjacency statistics (TAS) [9] threshold the image and to count the number of above threshold pixels with a given number of above threshold pixels adjacent. For each RGB image, TAS returns a vector of 162 numbers. We found that the adding TAS into our design matrix causes overfitting. So we do not include it in the training process.

9) Scale Invariant Feature Transform: Scale Invariant Feature Transform (SIFT) [10] is a method

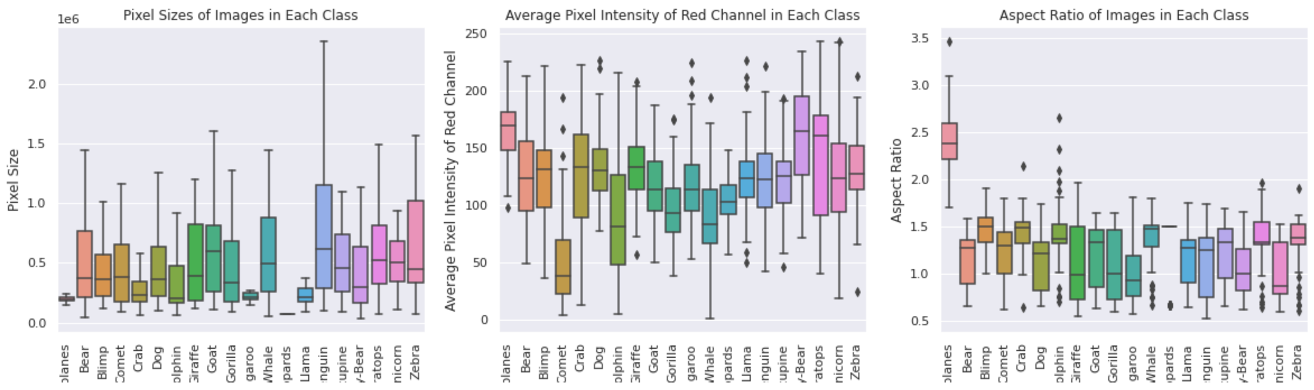

Fig. 5. Pixel size, average pixel intensity, and aspect ratio in each class

to extract keypoints and their descriptors from images. The descriptor of an image is a $k \times 128$ matrix, where k is the number of detected keypoints. We apply SIFT to resized images, and use Bag of Visual Words [2] to extract a fixed number of scalars from every image as features. We treat an image as a bag of words (BOW) and use the descriptors to construct vocabularies and represent each image as a histogram of discrete feature frequencies that are in the image as shown in 1.

The steps to generate BOVW features are as follows. We first build a visual dictionary by concatenating the SIFT descriptors of all images along the first dimension. Then, we make clusters from the descriptors using K-Means. The center of each cluster will is used as the visual dictionary's vocabularies. We try different numbers of clusters and found that the classifiers perform well when the number of clusters is 250. Finally, for each image, we make a frequency histogram of the frequency of the vocabularies in the image. Those histograms are our bag of visual words (BOVW). The histogram is represented by 250 scalars, which are added to our design matrix.

After generating our features, we standardize them by subtracting the mean value of each column and dividing them by their standard deviation. We drop the columns that have zero standard deviation.

B. Training Conventional Classifiers

We train the following conventional classifiers on our training set: logistic regression, k-nearest neighbors (kNN), classification tree, random forests, support vector machines (SVM), and gradient boosting classification trees.

In an effort to train and measure the efficacy of our classifiers, we use 5-fold cross validation to train and validate each classifier on the featurized dataset using all the features generated in the previous step.

After each iteration of the cross validation, we measure the training error and the validation error on the trained model and added these values to a list. The scores in our results indicate the average of training error and average of validation error of our classifiers as measured at the end of the cross validation.

1) Logistic Regression: We build a multi-class logistic regression model using the one-vs-rest (OvR) scheme with L2 penalty and a intercept term. There is one output value associated with each class. The class that has the highest output value is the predicted label.

2) K-Nearest Neighbors: A kNN classifier stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

3) Decision Tree Classifier: A decision tree classifier learns from data to approximate a sine curve with a set of if-then-else decision rules. During the learning process, a decision tree recursively partitions the space by minimizing the impurity. We use Gini as our impurity. We split the leave node until all leaves are pure, which makes the training error always euqal to 0.

4) Support Vector Machine: We implement a support vector machine using the polynomial RBF kernel. Support vector machines are flexible in the way that one can use multiple kernels to represent the features in higher dimensional space on which we can fit a linear classifier.

5) Random Forest Classifier: Random forest classifiers are a part of a family of models called ”ensemble” models. In short, random forests are able to address the largest shortcoming of decision trees by limiting overfitting through training an ensemble of trees on boostrap samples of the training data and averaging their predictions through a voting process in the case of multi-class classification task at hand. With the insight that each decision tree in our forest is trained on an individual bootstrap sample of our dataset therefore the average of their uncorrelated errors has a mean of zero (for a suitably large number of trees). Random forests are therefore able to limit overfitting through reducing variance while managing to keep the bias the same.

6) Gradient Boosting Decision Trees: Previously we discuss the power of ensemble models, specifically in their ability to limit overfitting through lowering model variance. In this section we also discuss a specific classifier that gotten considerable attention since its widely used implementation in 2016 by Chen et al. [12] called XGBoost which underneath uses the idea of greedy function approximation presented first by Friedman et al. in [13]. Concisely, the goal is to introduce gradient boosting to ensemble of decision trees similar to random forests which are unable to use traditional optimization methods in Euclidean space due to an absence of a global loss function that can be minimized.

C. Training Neural Net Classifier

We train a neural net with PyTorch [14] on a single NVIDIA GeForce GTX 1070 using the previously published VGG-16 architecture [15] with the modification of changing the output classes to 20 to fit our task. As previously described, the initial training set contained a total of 1501 images which proved to be too little for the training and validation losses to converge. For the loss function we use the cross entropy loss as described in equation (1) for a binary classification problem. For the fully connected layers we use the ReLU activation described in [16].

$$R(\theta) \doteq -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(X_i^T\theta)) + (1-y_i)\log(1-\sigma(X_i^T\theta))\right) \quad (1)$$

For neural networks it is essential for training and validation losses to converge, this is one of the reasons why neural nets tend to use mini-batches of images to push through the network for the forward and backward propagation steps. In order to tackle this challenge we sample images from the publicly available ImageNet dataset [17]. By using an automated script to download the 20 categories from ImageNet we expand our ”pre-training” set to $20,242$ images. The image counts for each class categories can be seen in Fig. 6

Our process for training the neural network is as follows:

- Pre-process the augmented images by resizing them to 224×224 size.
- Normalize the training dataset to achieve zero mean and unit variance.
- Train the VGG-16 model using the augmented dataset with mini batch size of 64 in 20 epochs using Adam optimizer.
- Save this model to a file.

In general, neural net based classifiers out-performing any other classifier with pre-training on an augmented dataset has been widely known [18]. However, feature engineering is not to be looked down upon as given by Pal et al. [19] where they highlight the importance of combining a new pre-processing procedure on the images called ”ZCA” that attempts to normalize the images by projecting it into principal component space.

IV. RESULTS

A. Logistic Regression

We use grid search to find the best value of inverse of regularization strength $C = 0.2$. The accuracy achieved 47% on the validation set and 47% on the test set.

B. K-Nearest Neighbors (kNN)

We use grid search to find the best number of neighbors: 18. The accuracy achieved 47% on the validation set and 33% on the test set.

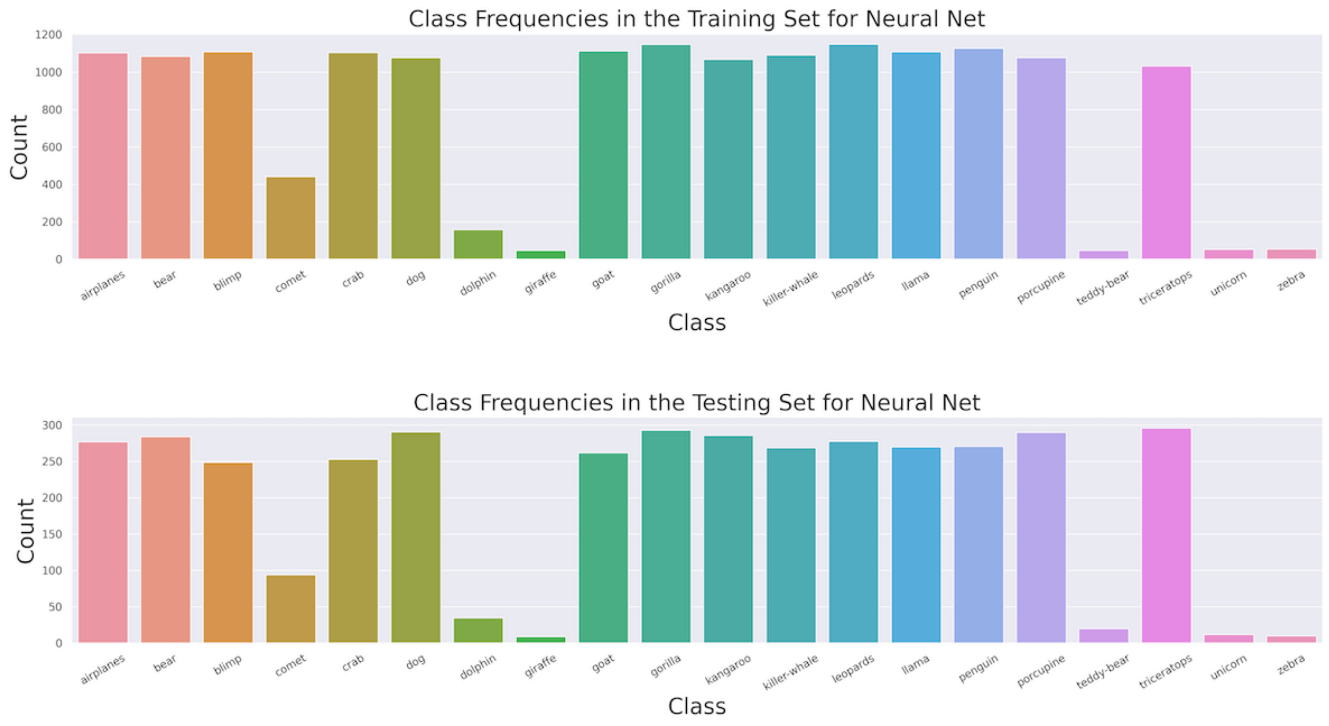

Fig. 6. Image counts for each of the 20 classes in the augmented dataset used for pre-training the neural network. (Best viewed in color)

C. Decision Tree Classifier

We train a decision tree classifier until the leaf nodes are pure nodes. The accuracy achieved 30% on the validation set and 32% on the test set.

D. Support Vector Machine

We use 4 different kernels: ”RBF”, ”linear”, ”polynomial degree=2”, ”polynomial degree=3” and have observed that RBF kernel performed the best on both validation and test sets with a slight regularization $C = 3$. We use grid search to find the best regularization term. Our final SVM classifier achieved 46% on the validation set and 45% on the test set.

E. Random Forest Classifier

Similarly in our previous other classifiers, we use grid search to pick the best parameters of the random forest classifier with the two important parameters to search being: maximum depth and number of estimators (decision trees). Since each tree in the forest can be independently trained, we leverage this by training in two separate cores of our CPU in parallel. Our final classifier achieves 43% on our validation set and 46% on the test set.

F. Gradient Boosting Decision Trees

Using the XGBoost library we train a model on our training set using 5-fold cross validation and our final model achieves 90% on our validation set and 51% on our test set.

Feature selection is an important step when it comes to classification on such high dimensional data therefore we print analyze the relative importance of each feature using gains. The 5 features that have the highest gains are: The 43rd component of color histogram, the average pixel intensity of green channel, the 45th component of color histogram, the size of the image, and the 132nd component of BoVW vector. On average, our scalar-based features have a higher gain than the scalar components in vector-based features. In addition, the gain of aspect ratio ranks 9th over the gains of 373 features, which helps us confirm that it is a useful feature.

G. Neural Network

Using the saved model file we re-train the neural net on our original dataset of 1200 images and compute the accuracy on 301 images which roughly corresponds to 20% of the entire dataset available. We report that the accuracy achieved by our neural net model is around $80-84\%$ on the test set which is significantly higher than the accuracy Gradient Boosting Tree achieves on the test dataset which is around $50-55\%$.

V. Conclusion

Multi-class classification of images has become a vital task in machine learning and computer vision research with wide ranging applications in the industry. In this report we show that image features like pixel channel intensities, aspect ratio, and image size can be useful features for image classification tasks. Moreover, relying on discrete feature encoding techniques such as BOVW can help capture contextual, locally aware image descriptions in the absence of convolutional neural nets. Through visualizing the features in different categories we are able to select the best features that help describe the images. We also show that gradient boosting decision trees can achieve a higher accuracy in image classification task than many other traditional machine learning algorithms. However, even with careful feature engineering, we conclude that accuracies of traditional techniques are still significantly worse than the accuracy the neural net based classifier achieves on the training data.

References

[1] D. A. Lisin, M. A. Mattar, M. B. Blaschko, E. G. Learned-Miller, and M. C. Benfield, "Combining Local and Global Image Features for Object Class Recognition," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pp. 47–47, Sept. 2005. ISSN: 2160-7516.

[2] Sivic and Zisserman, "Video Google: a text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1470–1477 vol.2, Oct. 2003.

[3] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Learning representations by predicting bags of visual words," 2020.

[4] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference*, pp. 23.1–23.6, Alvety Vision Club, 1988. doi:10.5244/C.2.23.

[5] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto, "Making good features track better," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pp. 178–183, 1998.

[6] Ming-Kuei Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, Feb. 1962. Conference Name: IRE Transactions on Information Theory.

[7] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 610–621, Nov. 1973. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[8] M. R. Teague, "Image analysis via the general theory of moments*," *JOSA*, vol. 70, pp. 920–930, Aug. 1980. Publisher: Optical Society of America.

[9] N. A. Hamilton, R. S. Pantelic, K. Hanson, and R. D. Teasdale, "Fast automated cell phenotype image classification," *BMC Bioinformatics*, vol. 8, p. 110, Mar. 2007.

[10] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.

[11] L. P. Coelho, "Mahotas: Open source software for scriptable computer vision," *Journal of Open Research Software*, vol. 1, p. e3, July 2013. arXiv: 1211.4907.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *Ann. Statist.*, vol. 29, pp. 1189–1232, 10 2001.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 807–814, Omnipress, 2010.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[18] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," *CoRR*, vol. abs/1310.1531, 2013.

[19] K. K. Pal and K. S. Sudeep, "Preprocessing for image classification by convolutional neural networks," in *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pp. 1778–1781, 2016.