Timothy Delille, Terrance Wang

Professor Joshua Hug, Deborah Nolan

Principles and Techniques of Data Science

December 13, 2019

Image Classification Model Training Based on Multiple Classification Algorithms

Abstract

This paper discusses the workflow to solve an image classification problem that aims to train and evaluate several predictors for images that could each fall into one of 20 specified categories.

Introduction

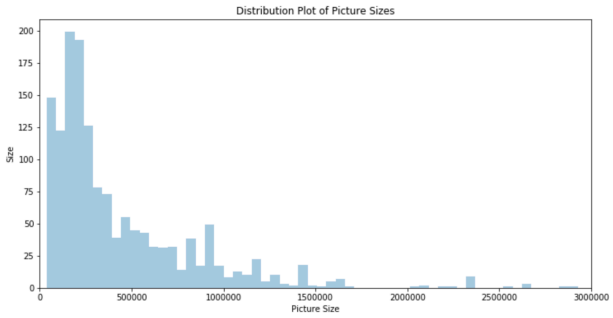
From a learning set containing 2921 images of 20 different types, we aim to train and evaluate several predictors for the class of an image and then evaluate these predictors using a test set to determine the best predictor. The 20 types are the following: ‘airplanes’, ‘bear’, ‘blimp’, ‘comet’, ‘crab’, ‘dog’, ‘dolphin’, ‘giraffe’, ‘goat’, ‘gorilla’, ‘kangaroo’, ‘killer-whale’, ‘leopards’, ‘llama’, ‘penguin’, ‘porcupine’, ‘teddy-bear’, ‘triceratops’, ‘unicorn’, ‘zebra’, encoded from 0 to 19 in that particular order.

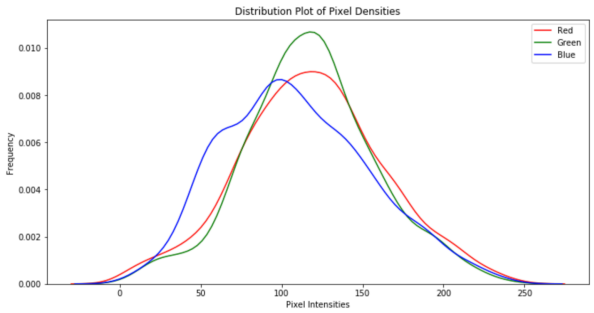
The learning set consists of a total of 2,921 images with different sizes. Each image is represented as 3-d array with the first two dimensions corresponding to the row and column pixels and third dimension to the color. The third dimension size is always 3, and each value corresponds to a RGB color intensity between 0 and 28 – 1. To orga-

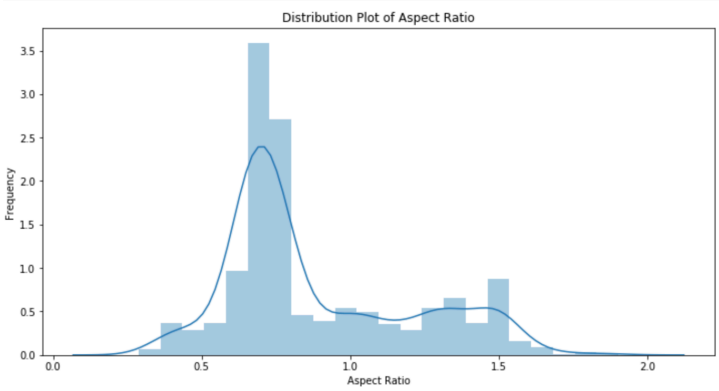
nize the image representations from all 20 categories in one data-frame, a table is created with a column ‘Pictures’ containing the 3-d array representation of each image and a column ‘Encoding’ containing the encoded category of each image.

Exploratory Data Analysis

For EDA, we plot the distribution plots for sizes, pixel densities and aspect ratios of the pictures. As we don’t want our classifier to be dependent on the size of the image or the aspect ratio (which seemingly have no link to the actual class), we won’t further use these features. However, pixel intensities can be helpful to differentiate some classes, as we will later see.

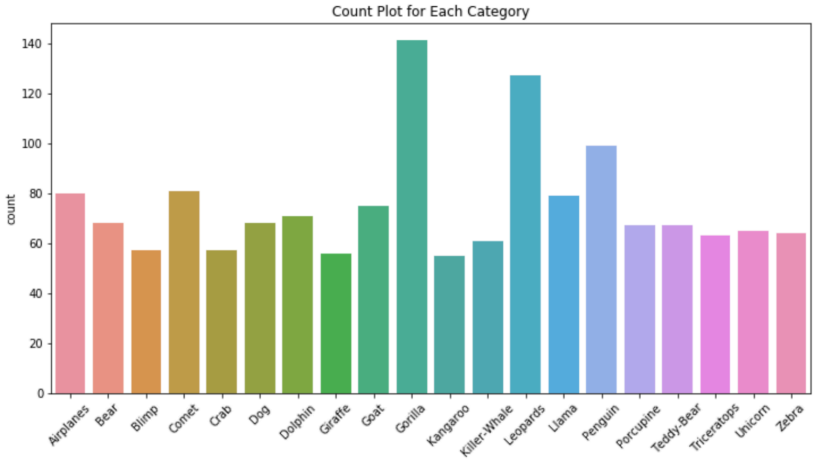




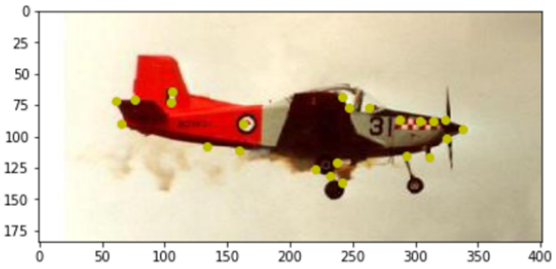


We can see that the dataset isn’t fully balanced, some classes are highly represented (gorillas or leopards for example). We could downsample those classes to avoid intro-

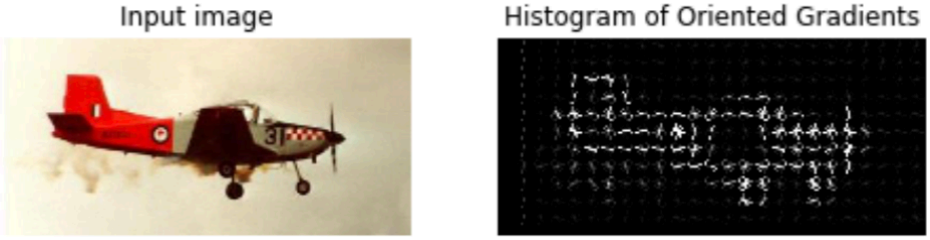
ducing bias, but seeing as image classification needs lots of data, we chose to keep it that way.



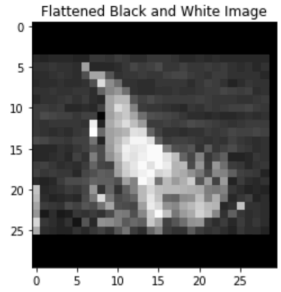
Firstly, we naively tried to use fancy features such as Harris Corner Detection, Oriented Gradient Histograms or PCA, which happened to not be very helpful compared to the computational complexity it introduced. Harris Corner Detection outputs point at which the algorithm detects corners, which is highly dependent on the background, the relatively poor efficiency of the algorithm but not so much on the actual class:



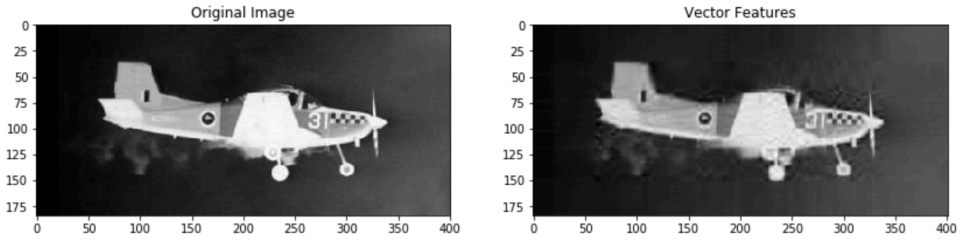
Oriented gradients happened to be pretty helpful since they carry way more information than corners and can be interpreted as the output of a convolutional layer:



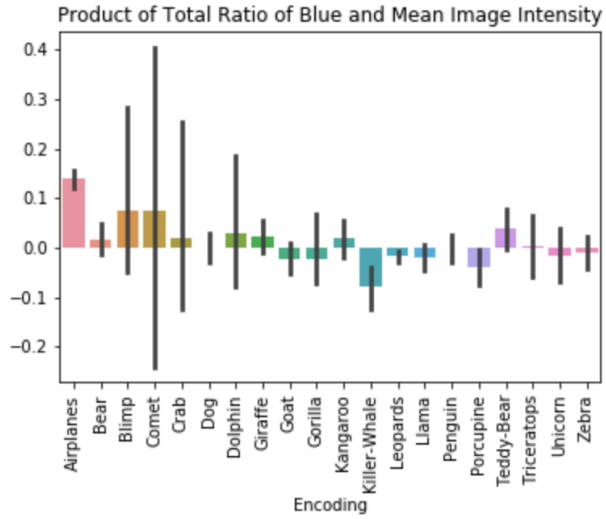
Moreover, we found that using a 40x40 black and white copy of the original image was quite helpful. However, when training solely on this feature, the classifier did not go past 0.1% accuracy.

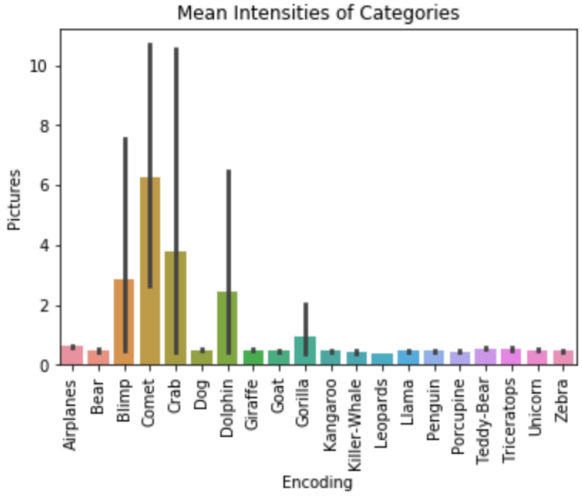


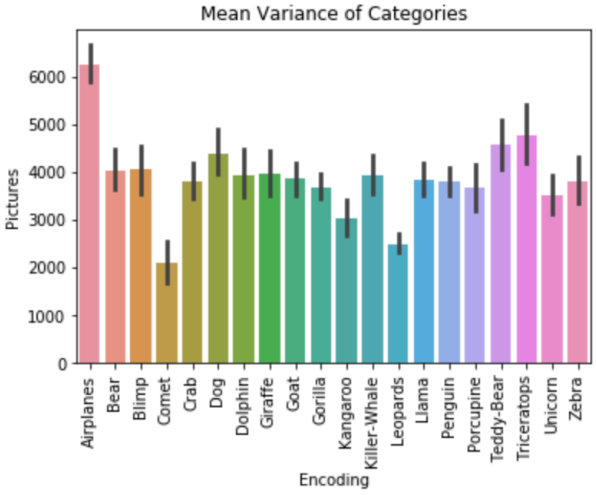
Furthermore, we found that computing principal components was as inefficient as it was computationally costly. It is hard to plot the distribution of vector based features among the classes so we judged whether a feature was relevant or not when it was discarded during training. This process was automated through a function that built the data frame and trained a model for all features of interest.

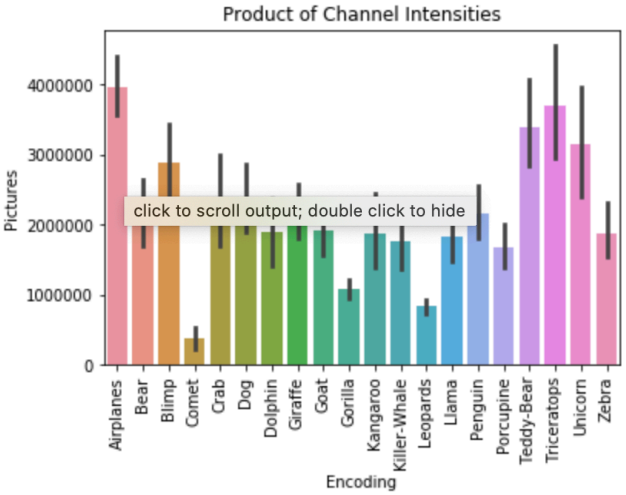


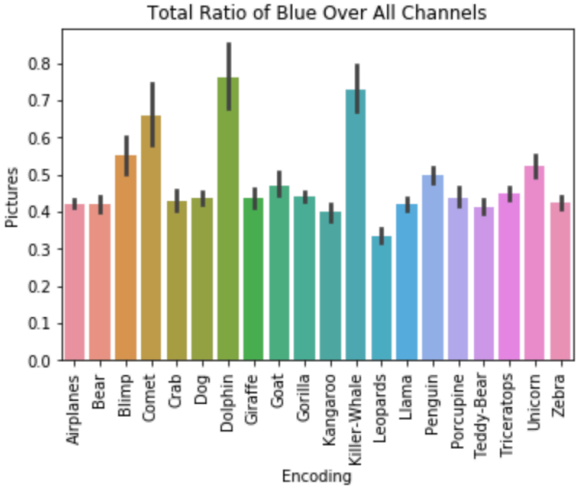
For effective features, we find that the mean intensities of images are particularly useful in identifying categories such as Blimp, Comet, Crab, Dolphin or Gorilla. Low image variance and low products of channel intensities suggest high likelihood of the Comet, Kangaroo and Leopards categories. High ratios of blue over all channels serves well in classifying images in categories of Blimp, Comet, Dolphin and Killer-Whale. These correlations can be observed from the bar plots for these features.


Product of Total Ratio of Blue and Mean Image Intensity
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
Airplanes
Bear
Blimp
Comet
Crab
Dog
Dolphin
Giraffe
Goat
Gorilla
Kangaroo
Killer-Whale
Leopards
Llama
Penguin
Porcupine
Teddy-Bear
Triceratops
Unicorn
Zebra
Encoding


Mean Intensities of Categories
Pictures
10
8
6
4
2
0
Airplanes
Bear
Blimp
Comet
Crab
Dog
Dolphin
Giraffe
Goat
Gorilla
Kangaroo
Killer-Whale
Leopards
Llama
Penguin
Porcupine
Teddy-Bear
Triceratops
Unicorn
Zebra
Encoding


Mean Variance of Categories
Pictures
6000
5000
4000
3000
2000
1000
0
Airplanes
Bear
Blimp
Comet
Crab
Dog
Dolphin
Giraffe
Goat
Gorilla
Kangaroo
Killer-Whale
Leopards
Llama
Penguin
Porcupine
Teddy-Bear
Triceratops
Unicorn
Zebra
Encoding


Product of Channel Intensities
Pictures
4000000
3000000
2000000
1000000
0
click to scroll output; double click to hide
Airplanes
Bear
Blimp
Comet
Crab
Dog
Dolphin
Giraffe
Goat
Gorilla
Kangaroo
Killer-Whale
Leopards
Llama
Penguin
Porcupine
Teddy-Bear
Triceratops
Unicorn
Zebra
Encoding


Total Ratio of Blue Over All Channels
Pictures
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Airplanes
Bear
Blimp
Comet
Crab
Dog
Dolphin
Giraffe
Goat
Gorilla
Kangaroo
Killer-Whale
Leopards
Llama
Penguin
Porcupine
Teddy-Bear
Triceratops
Unicorn
Zebra
Encoding

The products of total ratio of blue and mean image intensity also allow us to differentiate classes in a better way than some of the previous scalar features. Intuitively, we thought that images with high intensity such as comets or lots of blue (like whales) would have a higher “blue to other channels” ratio. However, a drawback is the high intra-class variance.

The last feature we used was: using all the previously computed feature to generate 20 clusters using a mixture of gaussians model. This feature boosted our accuracy by 10% and unsurprisingly helped the decision tree / random forest classifiers extensively.

We performed a train-test split on the labelled dataset for validating our trained models using 5-fold cross-validation. For models that necessitated tuning hyper-parameters, we could have used nested cross validation to avoid overfitting to our testing data while both selecting a parameter and evaluating the model. We concluded that logistic regression performs better without a penalty or with a slight L2 penalty and that 1-Nearest Neighbors tend to perform better, as the theory suggests. The logistic regression and random forest models gives accuracy rates around 30%, while the classification tree and SVM models give around 20% accuracy and the KNN model only gives accuracy rates around 15%. For future work, the accuracy rate can definitely be improved by using more effective combinations of features and training from larger and more diverse datasets.

The logistic regression implementation of the scikit-learn module doesn’t seem to perform one-hot vector encoding for the label vector when doing “one versus rest” classification. This leads to some bias for the classes with the higher label, so we tried imple-

menting logistic regression but it ended up not training properly. One interesting thing we found was that, by scaling the features, we could make the loss function less narrow, thus less prone to make SGD diverge. However, scaling the features often led to bad performance for our Random Forest algorithm, which can be explained by the homogenization of entropy in the dataset when we scale the features, thus making it harder for the classifier to choose which feature to split on.

Finally, we combined our top performing classifiers (logistic regression + random forest) into a meta-classifier (logistic regression) to compute the final classes. However, as this turned out to be less accurate than our random forest classifier (though certainly less variant), we sticked to the random forest for our final prediction.