Diverse Feature Extraction and Majority-Vote Model Predictions for Visual Category Recognition

Background

In this project we attempt several methods to categorize objects from images in the Caltech-101 dataset into their respective object categories. We attempt several different classification algorithms, feature extraction sets, and data analytics methods in our search for the best possible classifier. We attempted to use Matlab's bagOfFeatures extractor in combination with SURF Feature extraction, and a dimensionality reduction using PCA, but we refined the featureset to an un-pruned bagOfFeatures extraction. For our classifiers, we trained 22 basic classifiers, including KNN, Ensemble, and SVM, but later refined our selection of classifiers to 7 high-performing models. From these 7 models we attempted to further improve on the classification of objects by creating a meta-model, which is a weighted majority-vote system. We had the 7 models collectively vote on a solution, and weighed their votes by their previously determined accuracy.

1. **Related works:**

There have been several approaches to this problem to recognize the object categories. Hao Zhang, Alexander Berg, Michael Maire, and Jitendra Malik have used Discriminative Nearest Neighbor Classification algorithm to visualize the object categories (Citation 1). Zhang et. al. utilizes a framework of measuring similarities between testing data and category prototypes rather than defining differences based on a list of features. However, we have decided to use feature extractions in this. We came up with this idea from the matlab video **[citation]. [NAME]** has mentioned how to use feature extractions, and then use different models and algorithms to find an accuracy. [name] has also trained the Neural Network by using a pre-trained model (from). However, we are decided not to use it at this time for the sake of simplicity.

Alexander Berg, Tamara Berg, and Jitendra Malik have used Low Distortion Correspondences for shape matching and Object Recognition. They have used Caltech-101 dataset to recognize the object categories. They have used certain type of algorithms to compare two different shapes of the images. For Object Recognition, they have implemented a nearest neighbor framework to get a distance and predict the accuracy.

They have use cost function, which was obtained from the similarity of corresponding geometric blur point descriptors as well as the geometric distortion between pairs of corresponding feature points. The algorithms, they have used, handles the outliers. Then, they have used certain algorithms to compare two different shapes of the images. For Object Recognition, they have defined the distance between exemplar and query, which is matching the cost between different points. They have implemented a nearest neighbor framework to get a distance.

While we did not directly refer to previous academic work when constructing our weighted majority-vote meta-learning algorithm, there is at least one other author, Littlestone, N., who has implemented this

solution. The fact that there was only one author found on a quick google search for weighted majority-votes suggest that this approach, while novel and valid, is likely not one of the best solutions to this problem. We discuss these results in our conclusion section.

Method

There are several models and algorithms that we used for object recognition such as SVM, KNN and Ensemble. Using different functions for each models enriches the variety of the models and increases the probability of finding a model with a better accuracy, and training multiple models is time-inexpensive for a small dataset like Caltech-101. In order to find a model which has the highest accuracy we first used a Matlab built-in function named bagOfFeatures (BoF) to extract features from the Caltech-101 dataset. With the extracted features, we are able to construct a model-friendly table with the features as classifiers and appropriate class labels. This was accomplished by converting the table of features extracted by BoF into doubles (between 0,1). We used this data to train a variety of models one-by-one, and by comparing the models we used, we can find one with higher accuracy in recognizing the objects.

Other than these models mentioned above, we conducted more experiments using 22 of Matlab's built in classifiers, and trained them on the features extracted from BagOfFeatures and SURF. We had considered attempting a feature extraction based on a comparison to a categorical average (see figure **X.X**), and some other custom feature extraction, but we decided against this feature extraction exploration for the sake of further experimentation with modeling. In the meantime, combining extracted features using these above functions can increase the number of features which will enrich the diversity of classifiers. To utilize the useful classifiers and reduce dimensions, we attempted to apply PCA at various stages throughout our experimentation.

There are several choices we made along the way. Compared with Caltech-256 dataset, which includes 256 image categories, we choose Caltech-101 dataset since this set is smaller making it faster to run when conducting experiments. This allowed us to perform more trial-and-error experimentation and see larger changes in the resulting accuracy predictions. When it comes to the predictive power of our selected models, we were unable to predict which model works best for our dataset before experimentation, but across multiple trial-and-error attempts using our various methods we found that there were particular models that emerged as "high-performers". For the feature extractors, we are now using bagsOfFeatures because it is easy to get started and gives us a basic understanding of how feature extraction works. We attempted to perform other feature extractions and combine them into a superset of features, but we were unable to accomplish this with the time allotted. We can expect that some of the extracted feature sets would work better than the others, and that certain combinations of feature extracted sets may have the potential work better than either the individual sets alone or a combination of all possible feature sets. We also expected that PCA would greatly assist in reducing dimensionality and overfitting when we combined multiple feature sets together, but since we did not end up doing this, we only applied PCA to our one bagOfFeatures feature set.

Our finalized method selected the 7 historically highest-performing classification algorithms for a weighted majority vote meta-model. The 7 chosen were Linear Discriminant, Linear SVM, Quadratic SVM, Cubic SVM, Fine KNN, CosineKNN, and EnsembleSSD, with each of their votes weighted by their accuracy on training data. We split the data into a set of 22 training data per object category with 5-fold cross validation for the 7 classification models, and 9 testing data for the majority-vote meta-model.

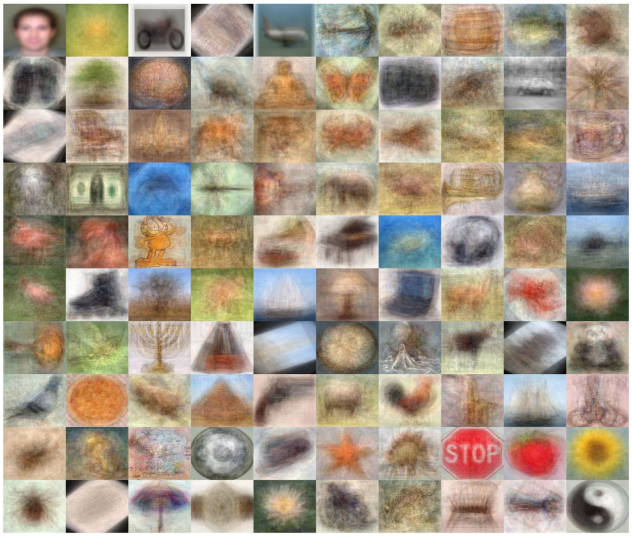

Figure X: Averaged images for each object category. Note that many object categories are unique and still clearly distinguishable despite having been combined with a number of other sources.

Plan and Experiment

***Datasets* -** The Caltech-101 and Caltech-256 datasets consist of images from 101 and 256 image categories respectively, each with one additional category of random images which is considered “background” and does not represent a human-recognizable object category. This makes the total number of classes 102 and 257, respectively. For this midterm report, we only experiment with the Caltech-101 dataset for the sake of calculation speed and simplicity, though we expect that our methods will be easily applicable to the 256 dataset once we have a high-performance method for object recognition. It is recognized that for both of the datasets, the images have the target object located toward the center of the image, which is referred to as photographer bias. This bias makes it much easier for certain object recognition algorithms to predict an object’s class. Previous data analysts have used this dataset to make predictions, and their performance is graphed in **Figure X.x**. In an attempt to compete with these benchmarks, we have extracted 250 features from this dataset and used multiple prediction models to predict the class of a test dataset.

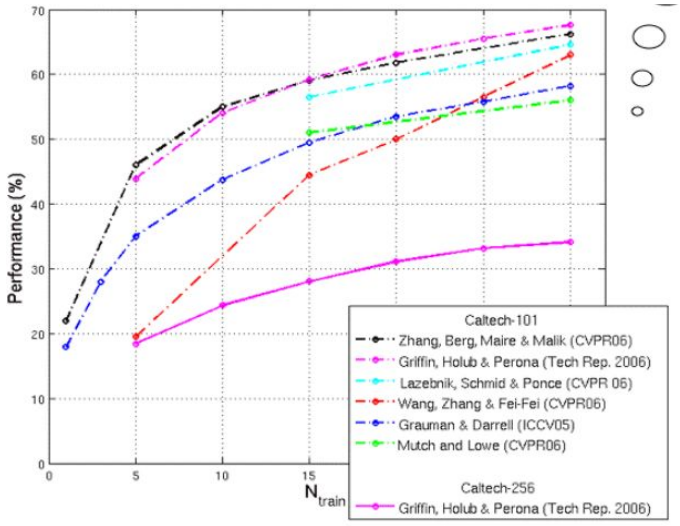

Figure X : Caltech-101 and Caltech-256 Performance Benchmarks

Hypotheses - In this project we wish to investigate the predictive power of various data analysis models when provided with various sets of image features. Some of papers we have researched indicate that this type of analysis does not perform as well as properly trained Convolutional Neural Networks, or that it is more appropriate to create a "prototype" from training data and compare distances against given testing data. However, we hypothesize that by using a combination of feature extraction techniques with dimensionality reduction through principal component analysis, that we will be able to train a model which has high predictive power. We will be interested in learning how well different feature extraction techniques perform on certain object categories, as well as how the predictive power of our models changes as we combine multiple feature extraction sets together and reduce the feature set's dimensionality through PCA. Additionally we hypothesize that a majority-vote of the most accurate prediction models would further increase the overall accuracy of predictions.

Experimental Design - The experimental design for testing our hypothesis that combining extracted feature sets and reducing dimensionality through PCA is mostly a matter of trial-and-error. If there are 15 valid feature extraction methods, we are not sure if simply combining all extracted features from all 15 methods and then reducing dimensionality with PCA will lead to the most accurate model predictions, or if there will be a set of two or three feature extraction methods which combined yield the most accurate model predictions.

On the hypothesis of majority-vote, we will want to experiment with many feature extraction techniques and prediction models to find models techniques which yield a diverse result of predictions. In our preliminary examinations, we have found that many of the high-performing models have similar confusion matrixes (**See Figure X.X),** in that they incorrectly mistake one image category for another category with regularity (indicated by brighter red squares in the confusion matrix). As such, if multiple models incorrectly predict the object category of an image in the same fashion, a majority vote will not be beneficial. In order to successfully make a majority vote prediction, we will need to design an algorithm which takes this information into account.

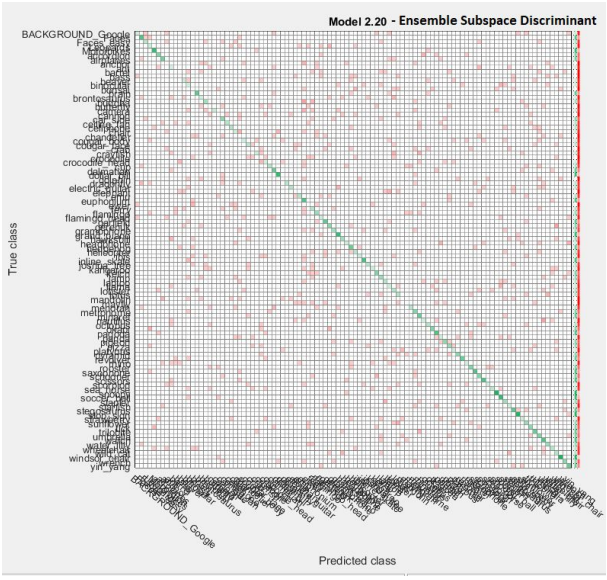

Figure X. Confusion matrixes of Ensemble model with subspace discriminant algorithm. Note the green diagonal of correct predictions with sparse red dots of high-density misclassifications.

Results

We have run the Caltech-101 dataset using Matlab with several constructed models mentioned above and the one with the highest accuracy rate is the Ensemble model with subspace discriminant algorithm. With 22 training images per class and 9 per class for testing (Total of 3162 images), Ensemble achieved a 30.6% accuracy in predicting the object category, though there were other predictive models which also achieved notable results. Linear, Quadratic, and Cubic SVMs were able to predict object categories correctly 28.6%, 29.1%, and 30.0% of the time, respectively (See figure **X.X** for more results). When applying PCA to reduce the dimensionality, we found that 95% of the variance of the data was contained in 210 out of 250 features. This is an unexpectedly large number of features, and we found that when we trained our models against the reduced dimensionality featureset, that the performance dropped by about 20% across all models.

When training these models using 5-fold cross validation with 22 images, their performance was 0.2625 0.2518 0.2545 0.2678 0.1787 0.2264 0.2669 for Linear Discriminant, Linear SVM, Quadratic SVM, Cubic SVM, Fine KNN, CosineKNN, and EnsembleSSD respectively. We then aggregated the predictions of these 7 models, and simply weighted their votes by their accuracy. This produced a performance boost of 5% over the highest-performing model (Ensemble) and a 36% performance over the worst performing model (Fine KNN). The end result is that we were able to predict the object's category correctly 28.1% of the time!

Figure X: Suite of predictive model accuracies given bagOfFeatures extracted feature set.

Conclusions

By conducting the experiments for object recognition, we have had many useful lessons that can be applied to future work or learning, Since we are mainly using Matlab for conducting the projects, we all are more experienced in using Matlab and learned more functions and libraries which are built in Matlab. In the model selection part, we not only used the built-in model, but also did some brainstorming about the models we learned which helps us have a more solid understanding of the algorithms and models.

We noted in particular that our majority-vote meta-model did not outperform default classification algorithms. In particular, it can be noted that with 31 data points per category, that Ensemble achieved a 30.6% prediction rate, while our overall meta-model performed at 28.1% accuracy. While we were unable to find the optimal training/testing sizes for our meta-model (on account of it taking 25 minutes to train each run), we still believe that we can draw the conclusion that a naively weighted majority-vote system is likely not optimal. The rationale for this conclusion is illustrated below, in figure **X.X,** where both the classification algorithms and our majority-vote model require a testing set in order to validate their accuracy. While it may not be entirely invalid to reuse the same testing set at both locations since we are weighting model's votes naively on their overall accuracy, we wanted to attempt to out-perform models by treating the system as atomic, with one input training set (that is internally split into a training/testing set for the 7 models) and one testing/validation set.

Additionally, we chose to treat the system as atomic and not reuse testing data because wanted to have a more complex weighting system based off of individual category precision. If we had done this, then reusing the testing data at both locations would be invalid, because the weights would be determined at such a granular object category level that it would positively skew accuracy ratings. To explain this system further: We had planned to weight an algorithm's vote based off its true-positive rate for its vote (see the confusion matrix in figure **X.X**). That is, if an algorithm predicts "Stop Sign", we weight its vote by

its true positive rate predicting “Stop sign” as determined by the 9 testing data passed to it, instead of its overall accuracy for all object categories.

While our naively weighted majority vote model was unsuccessful at outperforming classification algorithms of the same data training size, we believe that a more complexly weighted model with an optimized training/testing size balance could out-perform these models. However, we did notice that many of the classification algorithms classified data in a similar manner. That is, they had similar confusion matrices and often mis-predicted a particular datapoint in a similar fashion. For example two models might simultaneously mis-predict penguin as umbrella, rather than one model mis-predicting penguin as umbrella with another mis-predicting penguin as stop sign. Additionally, these models were trained under their default configurations and were not optimized, so it would not be fair to compare an optimized meta-model with them. We believe that an optimized classification algorithm would still perform better than our meta-model, both in terms of accuracy, but also in terms of the amount of computational time required to train the full model.

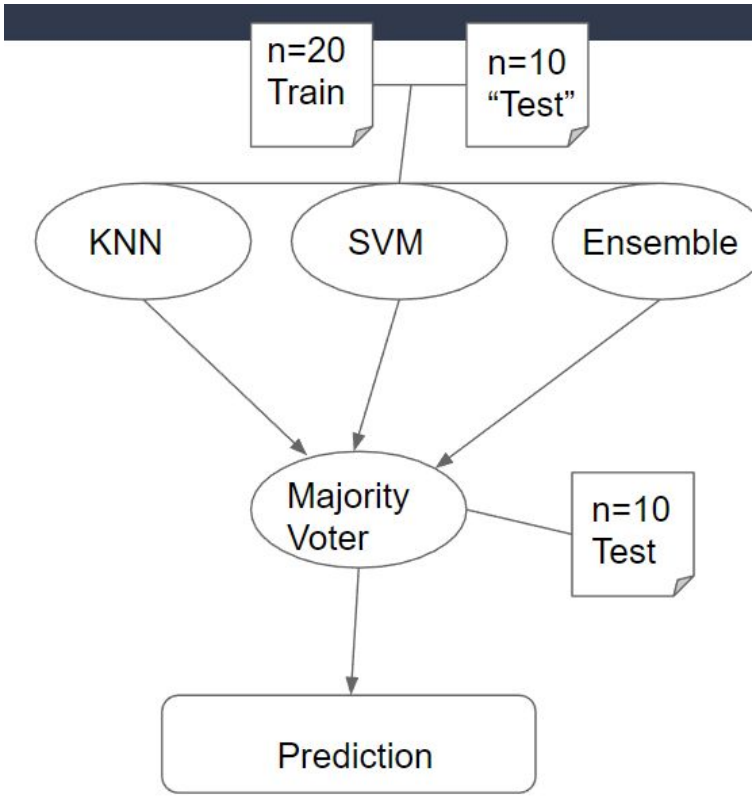

Figure X: A model of the majority vote meta-model. Note that both the voting models and the majority vote system as a whole require a (unique) testing set.

References

[1] Hao Zhang, Alexander C. Berg, Michael Maire & Jitendra Malik. Univ. of California: SVM-KNN: Discriminative
Nearest Neighbor Classification for Visual Category Recognition. Berkeley, CA. Retrieved from http://www.vision.caltech.edu/Image_Datasets/Caltech101/nhz_cvpr06.pdf.

[2] Johanna Pingel & Avinash Nehemiah. (2017, March). MathWorks: Object Recognition: Deep Learning and
Machine Learning for Computer Vision [Video File], Retrieved from
https://www.mathworks.com/videos/objectrecognition-
deep-learning-and-machine-learning-for-computer-vision-121144.html.
[3] Alexander C. Berg, Tamara L. Berg & Jitendra Malik. Univ. of California: Shape Matching
and Object Recognition using Low Distortion Correspondences. Berkely, CA. Retrieved from
http://acberg.com/papers/berg_correspondence_cvpr.pdf.
[4] Griffin G. Holub AD. & Perona P. The Caltech 256. Retrieved from
http://www.vision.caltech.edu/Image_Datasets/Caltech256/#Contact.
[5] https://en.wikipedia.org/wiki/Weighted_majority_algorithm_(machine_learning)