CS200A Final Project Report: Visualizing the World

December 8, 2018

Tianxiao Hu, Sheng Shen

Introduction

Image classification is increasingly attracting attentions. In this project, we leveraged 1,501 images in 20 classes for exploring this problem from the view of machine learning and deep learning. Specifically, we tried to apply the manual crafted feature in various sklearn-based models (such as LR, SVM, KNN and Random Forest). Moreover, we detailed the feature selection process with visualizing its distribution over each class. We finally introduce a CNN-based neural network for comparison, where we find an interesting correlation between regularization and the model's generalizability.

Data Cleaning

We were given 1501 pictures containing 20 classes of objects, most of which are RGB images. For gray scale images, we turned them into RGB images during preprocessing. We found files in `airplanes` folder were corrupted so we replaced the folder with correct data. We also noticed that images have both ".jpg" and ".png" file extensions, with different size for each picture.

We use `scikit-learn` [4] in our implementation. Other packages and version info are: `skimage=0.14.0, opencv=3.4.3, tensorflow=1.9`

Feature Engineering and EDA

We carefully selected 19 features. Descriptions are listed in Table 1.

Feature Name	Description
size	picture size (width $\times$ height for grayscale image, width $\times$ height $\times$ 3 for RGB image)
avg_red	mean value of red channel
avg_green	mean value of green channel
avg_blue	mean value of blue channel
std_red	standard derivation of red channel
std_green	standard derivation of green channel
std_blue	standard derivation of blue channel
avg_gray	mean value of grayscale
aspect_ratio	aspect ratio of the image, i.e., the height divided by the width of the image
harris	amount of corners detected by Harris corner detector [2]
dog	the differences of images processed by two Gaussian filters with different variance
avg_y	mean value of luminance (Y)
avg_cb	mean value of blue chroma component (Cb)
avg_cr	mean value of red chroma component (Cr)
std_y	standard derivation of luminance (Y)
std_cb	standard derivation of blue chroma component (Cb)
std_cr	standard derivation of red chroma component (Cr)
avg_hog	mean value of Histogram of Oriented Gradients (HOG) [1]
std_hog	standard derivation of Histogram of Oriented Gradients (HOG)

Table 1: Features used in classification and description.

During EDA process, we found two features appeared to be quite interesting: "avg_y" and "aspect_ratio":

"avg_y" feature of *comet* class has a very different distribution compared with other classes. The average of "avg_y" of *comet* is less than other classes. That's because comet images always have dark backgrounds, resulting in low luminance.

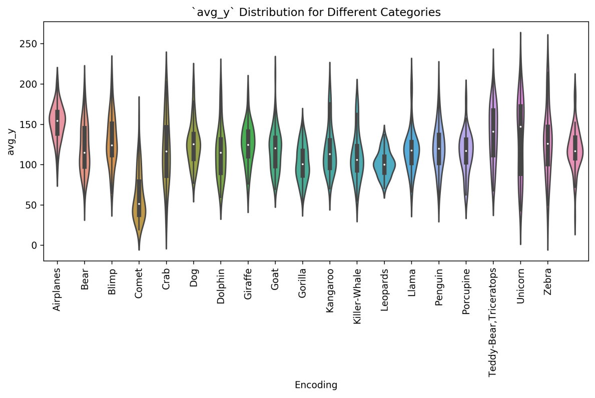

Figure 1: EDA of "avg_y"

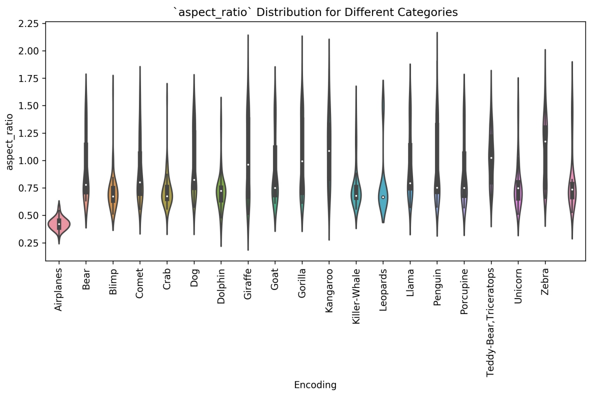

Figure 2: EDA of "aspect_ratio"

The distribution of "aspect_ratio" of *airplanes* class is also different from other distributions. That's because airplanes always spread horizontally in training images, thus images containing airplanes are more likely to be flat rectangle pictures with low "aspect_ratio".

Before EDA process, I thought "harris" can be a useful feature. For images containing *crabs* and *zebra* can be detected with more harris corners than other classes. However, after visualization, the distributions of different classes are quite similar.

Model Training

We tried four models on the dataset, also tuned parameters to get best performance. 90% of our data was used as training set and 10% as validation set. In this section, we will walk through each model and dig into the details. It's noteworthy that we fixed all random seeds in training process for easy reproduction.

Logistic Regression

We started with the simplest model. To prevent overfitting, we added regularization term for logistic regression. We searched L1 and L2 regularization and corresponding hyperparameter. For L2 regularization, we also tested models with different multiclass options ("ocr" and "multinomial").

The best logistic regression model achieved 43.71% accuracy on validation set and 42.44% on training set. The hyperparameter set was $C = 10$ and $penalty$ ='l1'.

Figure 3 is a graph showing performance of logistic regression models with different hyperparameter. We can see that hyperparameter C strongly affects the accuracy. For our best model, the accuracy on validation test is quite similar to training set, showing there is not overfitting.

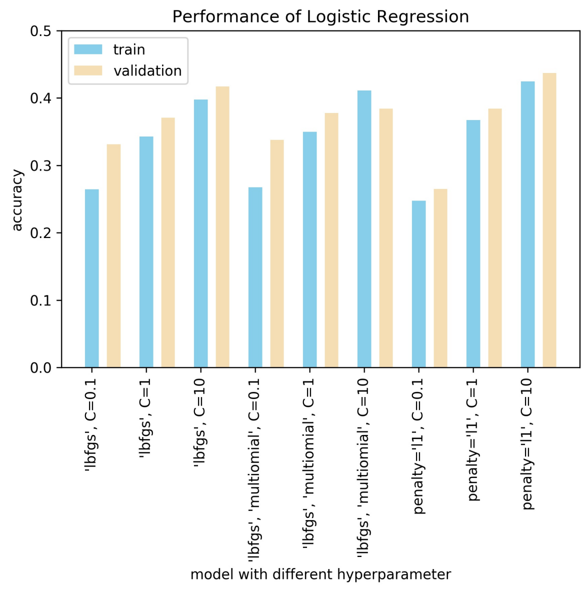

Figure 3: Performance of Logistic Regression

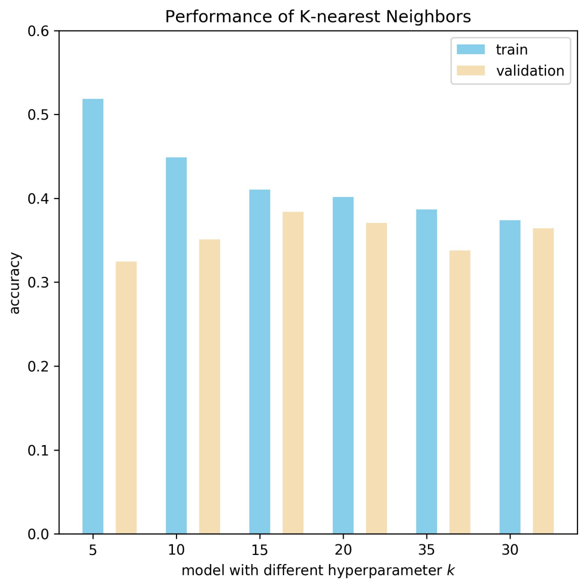

Figure 4: Performance of K-nearest Neighbors

K-nearest Neighbors

KNN is a non-parametric method for classification. We can classify a image by a majority vote of its neighbors in feature vector space. The image will be assigned to the class which is most common among the image's k nearest neighbors. We tried different parameter of k to search for best model.

The best k-nearest neighbors model achieved 41.03% accuracy on validation set and 38.41% on training set, with parameter $k = 5$.

Figure 4 is a graph showing performance of k-nearest neighbors models with different parameter k. As k becomes large, the accuracy of training set decreases, showing that it's easy to overfit when k is small. This may be caused by lack of data. However, the accuracy on validation test doesn't fluctuate as much as training accuracy when k changes.

Random Forest

Random forests are an ensemble method for classification by constructing a forest of decision trees during training. The output of random forest model is the mode of the classes of the individual decision trees. We tried models with different parameter 'n_estimators' and 'max_depth'. The first parameter represents the tree number in random forest and the second stand for maximum depth in an individual tree.

The best random forest model achieved 48.67% accuracy on validation set and 41.06% on training set. The best parameter set is $n\_estimators = 100, max\_depth = 6$.

Figure 5 showing performance of k-nearest neighbors models with different parameters. As 'max_depth' increases, the accuracy of training set goes up significantly, indicating that random forest can easily overfit.

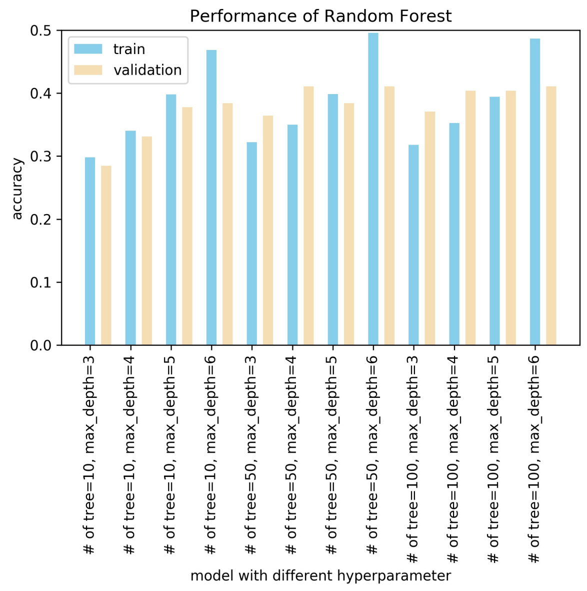

Figure 5: Performance of Random Forest

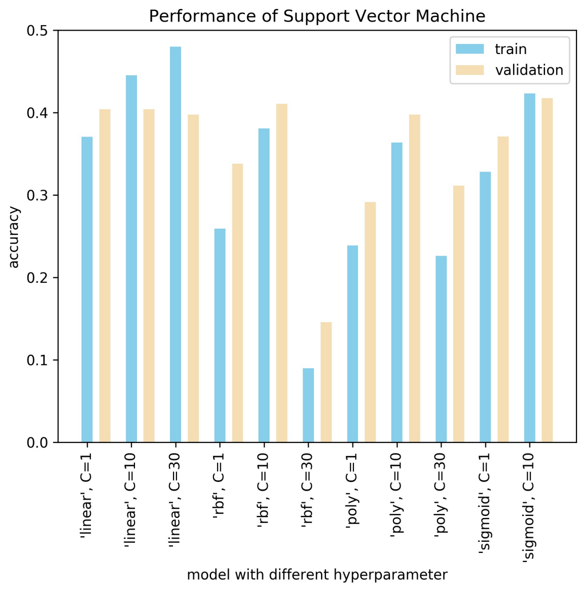

Figure 6: Performance of Support Vector Machine

SVM

We also employed support vector machine model on our dataset. We tried 4 kernel function: 'lilnear', 'rbf', 'poly' and 'sigmoid' with different penalty terms.

The best support vector machine model achieved 41.72% accuracy on validation set and 42.30% on training set, using 'sigmoid' kernel anc $C = 100$.

Figure 6 showing performance of support vector machine models with different parameter set. For very large penalty parameter C, the optimization will choose a smaller-margin hyperplane to do a better job of getting all the training points classified correctly.

Tensorflow

Following the baisc idea of [3], we adopted the deep learning method for image classification with tensorflow. Specifically, we propose to use a CNN-based neural network (3 layer of convolution layer to extract the feature directly from the 3 channels of each image and 3 layer of fully connect network to aggregate the features from the previous layers.) We also resize each image to (224, 224) and scale down each pixel in the range of

[0, 1] for numeric stability. The best performance of TF-based model was 41.44% on training and 39.06% on validation in terms of accuracy with train/validation split ratio as 9:1.

One thing we find interesting with training of the huge amount of parameters-based NN model is that the regularization seems to help a lot in the generalizability of this model. With the same model structure we described above, we trained the NN with adding 0.0001 scaled regularization to the last 3 fully connected layers and vice versa. We can notice from the following plots, which indicate with regularization the model turns to perform uniformly on both training and validation set. While the model without last three regularization-based layer turns out to enlarge the discrepancy between training and validation, which is known as the 'overfitting' issue we have learned from the class. One possible reason for that is the regularization constrained the scale of parameters and make the model more invariant towards small fluctuation of features.

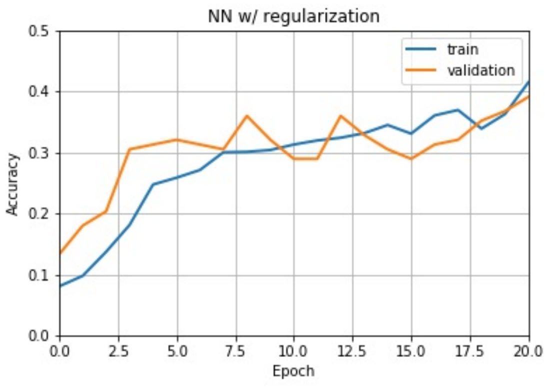

Figure 7: Performance of NN with regularization

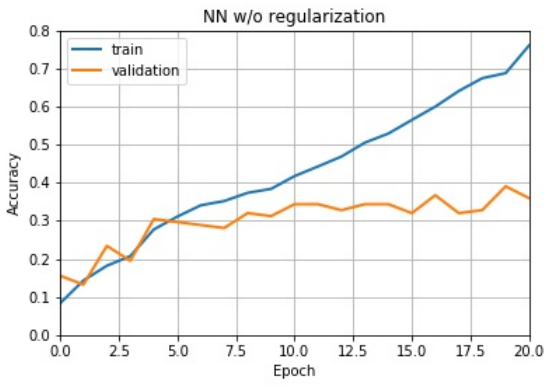

Figure 8: Performance of NN without regularization

Moreover, for comparison with the ML-based model, we can notice the advantage of NN is that the larger amount of parameters actually makes it capable of fitting any kinds of function, but also accompanies the potential risk of more likely to 'overfitting'. On the other hand, ML-based model turns out to be more stable (small variations) but less capable of fitting dataset. One possible future direction is to extract the feature from the last layer of NN and feed that into ML-based model, which might thus combine the advantage of both models.

Conclusion

The multi-class classification of images is possible through machine learning and computer vision algorithms. One of the biggest pain point we drew from the project is the wide varieties of possible data (inequality of labelled training data will easily lead to overfitting in that class). Therefore, in order to further create a more accurate system, there needs to be a large and continuously growing data source. Moreover, we discover the different advantages of ML and DL-based models and point out a possible direction for future system design in image detection.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[2] Christopher G. Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.