CIS400/401 Final Report:
Active Learning for Image Classification
Dept. of CIS - Senior Design 2015 - 2016 *

Pratyusha Gupta, Ella Polo, Lauren Reeder, Alex Wissmann
University of Pennsylvania
Philadelphia, PA

Abstract

Personal financial management is a critical step to help alleviate poverty in developing countries. Our goal is to allow anyone without technical skills to be able to harness the power of machine learning, specifically active learning. We decided to focus our efforts on image classification. We built a user-friendly web application that uses active learning to quickly classify any set of images that a user uploads to the application. The application works as follows:

- First, the user uploads images to the app, defines categories, and begins labeling them on a speedy, user-friendly interface.
- We feed these labeled images to a Support Vector Machine and train the model in batches.
- After a certain number of images have been labeled, we present the images for labeling in order of descending ambiguity (ascending distance to the decision surface of the model); this process is called active learning. By focusing on labeling the most ambiguous images, the model is able to reach its peak accuracy more quickly.
- Finally, when the user decides to finish labeling train the model, the application returns the complete, classified dataset to the user, along with an explanation and summary statistics of the model’s results.

We believe this application can be valuable for non-technical users wanting to quickly classify a set of images. To demonstrate our app’s viability, we have chosen the use case of assisting remote fashion stylists. Our application can help stylists by allowing them to classify articles or styles of clothing to assess their clients’ preferences and needs more efficiently. Unfortunately, there are few applications currently available that cater to this need in a specialized manner. TrackIt is a mobile application that integrates speech-to-text and natural language processing technologies to help users track cash transactions in an easy, fast, and unobtru-

*Advisor: Mitch Marcus
sive manner by providing accurate offline speech to text and seamless functionality in offline environments. TrackIt is the first application that uses these technologies to solve this market need. Stringent evaluation criteria were used in the technological development of the system and to test market interest. In both areas, TrackIt exceeded expectations and has the potential to successfully integrate financial budgeting and literacy into the daily lives of our target population.

1 INTRODUCTION

Our goal was to build a tool to help nontechnical users benefit from some of the gains that machine learning models have made over the past years. Instead of having the user simply upload their data into a black box, we involved them in the model building process through the active learning. We decided to focus on image classification and built a user friend web application with an SVM classifier on the backend. Additionally, we helped guide users through the process of classifying their dataset with active learning.

![Figure 1: Web Application Interface](image)

We originally decided to incorporate active learning in order to train the model more quickly. Active learning is a machine learning technique where the model selects the data points most informative to it during the labeling process. We decided to use active learning as part of the application but disguised its use from the users. Active learning was a question of theoretical interest to us. We found research that had demonstrated that active learning was valuable in the short run in order to marginally speed up training times on various models (Fails and Olsen). We hoped to demonstrate the usefulness of this technique for image classification, especially when labeling for training purposes is a laborious, expensive task.

To motivate the design and demonstrate the impact of our application, we decided to focus on a specific use case - Fashion Styling. Currently, companies such as Stitch Fix, The Trunk Club, Le Tote, and others provide recommendations and/or samples to their clients by processing batches of photos from their client’s wardrobe and Pinterest boards to make personalized recommendations.

We chose this use case because of the impact our categorization application can have on the efficiency and accuracy of the stylists in this industry. The current workflow is that once a client uploads their wardrobe, a stylist manually classifies types of clothing, bucketing the shirts, tops, pants, shoes, etc. The stylist will then look through these images to identify an overarching style using their professional knowledge and experience. They make recommendations for new clothing pieces that would fit in the client’s current wardrobe and fill any gaps based on the their existing wardrobe pieces. Our application cuts down substantially on the initial processing time, which helps categorize the types of clothing as well as determine basic
styles. This frees the stylists to focus on creating accurate recommendations with all of the data and classifications easily available to them. This improves the efficiency of the overall process and helps make more accurate recommendations by maintaining a consistent framework for clothing items and styles.

2 APPROACH

Our approach was to build a user friendly web application that would allow users to easily upload images for classification with our backend algorithms. We chose to write our backend in Python, which led us to create our application using Flask, a Python-based web development framework. Instead of creating our own machine learning algorithm, we used Scikit’s Machine Learning libraries for our classification models. We faced a difficult decision when deciding on which machine learning model to use. Image classification is a difficult problem and, depending on the images, modern machine learning techniques such as neural networks are well-suited and achieve high accuracy. However, we decided that a lightweight model would enable fast, real-time interaction with users. To this end, we decided to use a simple SVM model to classify the images. We chose the histogram of oriented gradients feature descriptor to break down images into feature vectors. The histogram of oriented gradients function traces the general shapes (directions of gradients) in the image, ignoring color, to create a feature vector. Thus, the tasks best tackled with our application involve classifying based on the shapes of the objects featured in the image rather than their color or texture.

In order to implement the active learning component, we calculated the distance of each image from the decision boundary of the SVM. For multi-class SVMs (the user can select an arbitrary number of categories on the upload page), there are decision boundaries between each pair of classes, so to calculate the distance from the aggregated decision boundary we sum the distance to each of the surfaces. We get these pre-computed distances from SciKit, which provides a function that outputs distance to the decision surface given an unlabeled feature vector. The SVM generally trains very quickly, which, when combined with the ease and speed of the distance calculation, made the SVM an obvious choice for classifying our images despite lower accuracy than more complicated algorithms. After training the model, we compute the distances to the decision surface for the remaining unlabeled data points, then feed the images closest to the decision surface to the user for labeling.

As the user labels images, we train the models in batches on constantly running background threads. We use the latest trained model to compute the distances to the decision surface for active learning. One issue we faced was how
often to refresh the model given a labeled image from the user i.e. how big each batch should be. If we ran the SVM upon receiving each labeled image, it would have been too slow to use in real time. After some experimentation, we decided to run the SVM and recompute the ambiguity of unlabeled images for every 20 images labeled by the user. We chose this batch size because we found that it neither slowed down the labeling nor significantly reduced the effectiveness of the active learning. This means that only the first twenty images presented to the users on our application are random, for after the first twenty the first model is trained and active learning determines labeling order after that point.

The final component of our application is integration with Amazon’s Mechanical Turk, a crowdsourcing platform for tasks more cheaply accomplished by humans than by machines. We decided to incorporate this feature as an alternative to the user labeling the images manually, which depending on the image could be a time-intensive task. This feature is not related to the active learning component of our application, but was incorporated to further the goal of allowing non-technical users to obtain labels for images quickly and easily. To classify images using Mechanical Turk, we first upload the users images to S3. If the user decides to post the image set to Mechanical Turk, we use a Python library to retrieve urls for each image from S3. These, along with the user-defined labels, are then formatted into a REST request sent to Mechanical Turk using their API. The results from the crowdworkers are fetched back upon completion, again using the API, and are displayed in the same way as results from our active learning model on the web application. While this might have been a very useful feature for the application, we unfortunately discovered that Amazon does not support batch jobs for request through the API, so each image must be posted to the platform as a separate HIT (Human Intelligence Task).

3 RESULTS

Figure 3: Results of Active Learning compared to Random Learning

Our system’s key advance over the classical machine learning techniques was the incorporation of active learning in order to accelerate the learning process. An early paper mentioned that they had seen similar gains through using active learning to reduce training times by reducing the number of labels necessary to reach peak accuracy (Zhang, Lin and Zhang). We were able to replicate these gains. We trained our model in batches of 94 images using both the active learning method of labeling and random selection labeling to obtain the data in Figure 3. We found that our active learning method had a much faster learning rate than the random learning control group. Although the final
accuracy is largely the same for active learning, the number of labelled instances is dramatically reduced when compared to the random learner. Both of the accuracies peak at about 80%, which is lower than ideal but a tradeoff with the ease and congruency of the SVM with our active learning algorithm.

Another component of our system is the labeling interface. This process began with a batch uploader, which loads the images onto S3 for storage at the beginning of the process. The user can then generate their own categories for labelling, which allows the flexibility to define the names and number of categories tailored to each uploaded dataset. Once the labelling process begins, our application has integrated keyboard shortcuts that are mapped to each category for classification in order to speed up the tedious labelling process. The key for the user labeling interface was to make the process as seamless as possible for a non-technical user to upload their data, label it, and interpret the results. To test the extent to which we achieved our goals of a fast, user-friendly application, we repeatedly demod the application to students, who labeled images and gave us feedback about the interface. Based on feedback from users, we are confident that the labeling process is easy and the interface is intuitive.

For result interpretation, we show the user the images that they have already labeled by hand in addition to predictions for the next twenty images in the dataset in order to give them an idea of the accuracy of the model through a simple visualization. We also provide the user with a link to a file containing the rest of the images in their dataset along with associated labels.

We used the Amazon Mechanical Turk (AMT) sandbox to demonstrate the posting of tasks on AMT. This is essentially a non-production version of the AMT site meant for testing. We can verify that the task has been posted by looking through the sandbox for our task. The only difference between the sandbox and the live version of AMT is the word sandbox in the request URL to the API. We do not have data on the accuracy of the crowd in labeling these images because we did not have funds with which to pay actual crowdworkers.

4 ETHICAL / PRIVACY CONCERNS

We do not believe our project created or solved any significant ethical or privacy concerns. Some existing concerns, such as the privacy of data uploaded to the cloud are still a concern with our project. Our web application stores the user uploaded images in Amazon S3, which means Amazon as a third party has access to the uploaded data by definition. This data should be secure on S3, but we have no guarantee other than Amazon’s for our stored data.

If our product is used to classify sensitive data, such as medical images, the error rate is too high for any reasonable predictions. This might not be apparent to a non-technical user, and could be remedied by a more thorough explanation of the results if we were to put our application into product. Another source of confusion could arise from the twenty predicted images that are displayed after the manual clas-
sification is done. The purpose of the twenty predicted images is to give an idea of the accuracy of the predictions on the next twenty images in the dataset. This may be misleading, as all 20 might be correct, although the overall accuracy may not be as high as it appears here. This would raise issues if assumptions were made about the accuracy of the model based on the small prediction sample shown in this part of our application.

5 DISCUSSION

We built a very user-friendly application capable of classifying images with reasonable accuracy while incorporating active learning. The application flows from uploading images to results smoothly and robustly. The user can efficiently upload and label any dataset, with an arbitrary number of categories. At the end of the process, the user can see a subset of their images on the screen along with labels, so that they can visually assess the accuracy of the model in addition to the data we return to them.

We were able to successfully integrate with Mechanical Turk in order to have the crowd label images individually. Because of the lack of a batch upload feature, however, this alternative for labeling is not scalable at all. However, we did gain familiarity with the Amazon Mechanical Turk API and are confident about the promise of the crowdsourcing platform for tasks which are larger in nature and therefore more suited to being posted as individual HITs.

The largest limitation of our project as it exists now is the accuracy of the classifier, which is a result of our choice of an SVM as the underlying model. We used the simple SVM as the backend classification model because it was fast and made it easy for us to use active learning through a quick identification of which points are closest to the decision boundary. This is a very simple model, which we would like to expand in future iterations to more cutting edge models. Our goal of making these models more accessible becomes more valuable as the models we make accessible become more difficult to implement on your own, to the point where our application may become useful even for technical users who do not want to have to do the additional work of incorporating active learning to more complex models. We would like to move from the SVM to neural nets, as these would improve the accuracy as well as be able to make more robust predictions. The challenge in the future would be the incorporation of active learning into these more complicated models, as the calculation of ambiguity becomes more difficult.

Another feature that this project requires for future feasibility is a more robust education component for the non-technical users. This means more descriptive instructions on the upload page, as well as another component of the results page. Ideally, this future addition to the results page would include a detailed account of the accuracy of the model as well as an explanation of how it works. We could also allow users to identify mislabeled images in order to retrain the model even after the results page is displayed. We should also consider making the mechanics of the active learning component of the site more transparent to the users, in the hopes of inspiring them about the use of this
machine learning technique and the promise it may hold to them in their future labeling tasks.

5.1 BIBLIOGRAPHY

Fails, Jerry Alan, and Dan R. Olsen, Jr. "Interactive Machine Learning." ACM. N.p., 12 Jan. 2003. Web. 7 Oct. 2015. In a perpetual machine learning situation, it is important to build strong classifiers that can reduce the need for specialized programming burden. The current model for performing this task first selects features, then trains and classifies repeatedly. However, by removing the feature selection phase and focusing on manual feedback from the user during training, classification becomes more accurate.

Zhang, Lei, Fuzong Lin, and Bo Zhang. "Support Vector Machine Learning for Image Retrieval." IEEE Explore. N.p., 10 Oct. 2001. Web. 7 Oct. 2015. When classifying images, it is important to articulate an effective relevance feedback component. This can be done through a query refinement scheme where users mark relevant or not relevant. This is an example of pool based active learning, where there is a pool of unmarked data, and the machine can request a users mark for a specific piece of the data. The paper proposes a support vector based active learner with the express goals of a) learn target concepts accurately and b) grasp concepts quickly.