CS229 PROJECT REPORT

Predicting Instagram tags with and without data

Shreyash Pandey - shreyash@stanford.edu
Abhijeet Phatak - aphatak@stanford.edu

Abstract

There are around 30,000 human-distinguishable basic object classes and many more fine grained ones. A major barrier to progress in computer based visual recognition is thus collecting training data for many classes. To counter this problem, a technique known as Zero Shot Learning (ZSL) has recently been introduced through which one is able to detect classes which were not part of the training set. In this project, we have analyzed two techniques within this area, describing the algorithms, strengths and weaknesses. To compare them with fully supervised image classification, we picked the task of Instagram hash-tag prediction, and developed an end-to-end data collection, cleaning and training regime for deep CNN architectures.

1 Introduction

One of the major bottlenecks in recognizing objects from images is that the number of different classes that the image could comprise of are huge in number. Data collection and annotation process for all those classes can be too inefficient and unfeasible. The other way to recognize those objects is to design algorithms that simulate how humans overcome this issue. A human being can detect the object in question even though it may be the first time they are seeing it. We are able to perform this inference by drawing information about that object from a different source (like text) and then using that to attempt to identify the object. This method is essentially what is used in practice to detect unseen classes and is referred to as Zero Shot Learning (ZSL).

A ZSL model typically utilizes information from text corpora, images and their labels and maps them to a common semantic space. Such a semantic space could either be a word space or an attribute space. Attribute space is defined using attributes(usually binary) such as *'hasFur'*, *'hasTail'*, *'isBrown'* etc. and is usually not preferred since manually tagging images with such attributes is not scalable and is inefficient. In case of a word space, where the labels are already mapped to that space, a mapping is learnt from the data to project images into that space. During test time, the input image is mapped in the semantic space and then a nearest neighbour search or some other similarity metric is used to select the closest unseen class.

As part of our project, we have decided to implement and compare two ZSL methods, with a specific application in mind - predicting common Instagram tags for images. The parsed tags (English words) are unseen classes that our ZSL tags map to. To compare them to fully supervised methods, we have also developed a cascade of data collection, data cleaning and training a deep CNN architecture.

We start this report by covering the basic aspects of all the common ZSL techniques, followed by a comparison between the two methods, then describing in detail the task at hand and the data preparation methods that we used, and finally concluding by comparing ZSL to fully supervised methods. We also include parallel work on Flask server that acts as a front-end for our hash-tag generation module.

2 Related Work

The recent survey paper titled, "Zero-Shot Learning - The Good, the Bad and the Ugly" was a great starting point for us as it provided a comprehensive overview of different techniques that have been tried in ZSL literature [1]. The general approaches that recognize unseen classes in images consist of knowledge transfer between visual and semantic spaces. This is done by ensuring that there is compatibility (linear or non-linear) between the two spaces. Methods that learn non-linear compatibility between the two spaces outperform methods that learn linear compatibility. Hybrid models are the ones that express images and semantic class embeddings as a mixture of seen class proportions. Following the advice of this paper, we decided to implement two hybrid models for our task, mostly because they are intuitive and simple to understand, and give decent results as well.

3 Zero Shot Learning Methods

3.1 ConSE

The first technique, known as ConSE or "Zero-Shot Learning by Convex Combination of Semantic Embeddings" (ConSE), employs a straightforward approach to dealing with ZSL tasks [2]. It uses a classifier trained on ImageNet to obtain semantic embedding of images by a convex combination of class label embedding vectors from the training set. The intuition is that the semantic embedding of an unseen image would be close to a weighted combination of the most likely seen classes:

$$f(x) = \frac{1}{Z}\sum_{t=1}^{T} p(\hat{y_0}(\mathbf{x}, t \mid \mathbf{x}).s(\hat{y_0}(\mathbf{x}, t)) \qquad \text{where } Z = \sum_{t=1}^{T} p(\hat{y_0}(\mathbf{x}, t \mid \mathbf{x})), \tag{1}$$

T here is a hyperparameter, $\hat{y_0}(\mathbf{x}, t)$ gives the t^{th} most probable label and $s(y)$ gives the semantic embedding of an image y. Finally, the prediction is obtained by finding the class nearest to the obtained semantic embedding.

We implemented this in PyTorch [3], and we used gensim for 300 dimensional Word2Vec features. For our experiments, we used the Word2Vec features trained on Google News because it had the largest vocabulary (3M). To perform the nearest neighbour search in the semantic word space, we used the cosine similarity metric. Also, because ImageNet classes mostly consist of multiple words per class name, we averaged the word vectors as recommended by the authors of the original paper.

3.2 HierSE

The next paper we studied was an extension of ConSE, "Zero-shot Image Tagging by Hierarchical Semantic Embedding" (HierSE) [4]. It improves upon ConSE and obtains better semantic embedding by extracting hierarchical structure defined in the WordNet. This is to ensure that labels with low/no occurrence in the vocabulary, which are of particular interest in ZSL, get reliable embedding vectors. It also creates its semantic space from Flickr tags as opposed to Google News in ConSE. This is based on the intuition that Flickr might be a better source since their tags better capture the label's visual context. The embedding vectors now are obtained by :

$$\begin{gathered} f(x) = \frac{1}{Z}\sum_{t=1}^{T} p(\hat{y_0}(\mathbf{x}, t \mid \mathbf{x}) s_{hi}(\hat{y_0}(\mathbf{x}, t)) \quad \text{where } s_{hi}(y) = \frac{1}{Z_{hi}} \sum_{y' \in y \cup super(y)} w(y' \mid y) s(y) \\ Z_{hi} = \sum_{y' \in y \cup super(y)} w(y' \mid y), \end{gathered} \tag{2}$$

Here $super(y)$ refers to the ancestors of a label obtained using WordNet and $w(y' \mid y)$ is a weight subject to exponential decay with respect to the minimal path length from y to y'. Prediction is performed in a similar manner as described above. The code [5] for HierSE is available online in Caffe and TensorFlow. We used the TensorFlow code for testing purposes.

Figure 1: Results of our PyTorch ConSE implementation for Hash-tag generation using ZSL. We took a set of 76 unseen (instagram relevant) tags. We then map 76 tags to popular hashtags as shown below.

4 Supervised Hashtag Prediction

The idea was to compare ZSL methods to fully supervised methods on a real world image classification task, and we chose Instagram hashtag prediction as it is relevant and exciting. The challenge is that there is no publicly available dataset that has images and hash-tag labels, which is why we had to develop the entire cascade on our own.

4.1 Data Collection

One option was to collect data directly for hash-tags, but this would pose a serious concern: word vectors would not be available for semantically non-sensical hashtags that are prevalent in social media. Hence, we decided to collect data for common English words that map to popular Insta hashtags, as this would help us have a fair comparison between ZSL and fully supervised methods. We surveyed and collected 76 partially exhaustive labels that were mapped to popular hash-tags using an API from `www.all-hashtag.com` which maps English words to Instagram hashtags. Once we had this list, we used DuckDuckGo image search engine to download 200 images per label [6]. Unlike Google search that blocks queries if there are too many of them, DuckDuckGo doesn't keep track, and hence it was ideal for our data collection system.

4.2 Data Cleaning

One quick look at the data downloaded through DuckDuckgo showed that there were plenty of noisy images that would hamper any classifier's accuracy. For example, some search queries had images with too many watermarks, ads or text on top of images. To remove such outliers, we employed the following procedure:

- Feature Extraction: Deep Convolutional Neural Networks have revolutionized feature extraction for image data. Instead of going ahead with traditional SIFT or SURF features, we decided to extract 4096 dimensional feature vectors as outputs from FC7 layer of pre-trained **AlexNet[7] CNN architecture**. This is bound to capture the semantic juice present in the image, and is ideal for the downstream task of outlier detection.

- Dimensionality Reduction: To perform outlier detection in a reasonable amount of time, and to remove correlations among different features, we decided to have a dimensionality reduction module in the pipeline where we employ **PCA** to reduce dimensions of the feature vector from 4096 to 128. We used scikit-learn [8] for PCA.

- Outlier Detection: We used **K-Means** as a clustering technique to form 5 clusters for images of each class. Images that were too far away from the centroids were bound to be noisy, and this was confirmed when we saw the results. We chose an arbitary threshold of two standard deviations from the mean ($\mu + 2\sigma$), and almost all the images with distance from centroid greater than this threshold turned out to be noisy. On an average, we were able to remove around 18 noisy images per class using this outlier detection cascade. The entire cascade is shown in Figure 2(b). We used scikit-learn [8] for K-means and other calculations.

4.3 Supervised Learning

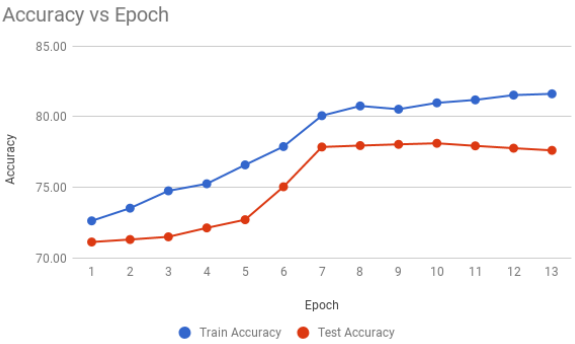
We split our dataset into train and dev sets in 4:1 ratio. The next step was to train a classifier on the clean data that we had accumulated. We decided to implement and compare the following two supervised methods:

- **Bag of Visual Words SVM**: We extract local features from the training image sets using SIFT. This essentially converts the image into a feature vector. The next step is codebook generation. KMeans (K=100) clustering over these local features gives us the centroids (vocabulary), and then each image is represented as a frequency distribution over this vocabulary. A linear SVM classifier is trained over this representation of images. In our experiments, we used an adapted version of MATLAB Bag of Visual Words Model in the Computer Vision Toolbox to implement this baseline model [9].

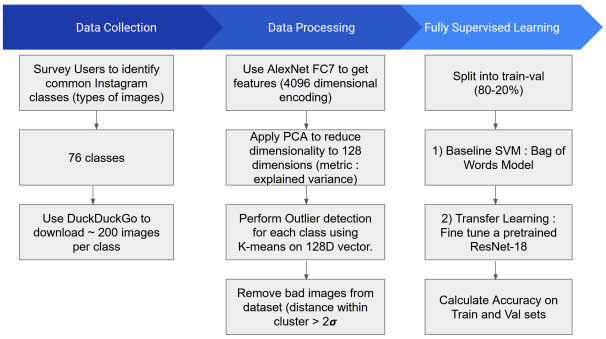
- **ResNet-18**: We then went ahead with ResNet [10] CNN architecture as it gives the state-of-the-art results on recognition tasks. Instead of initializing weights of our network randomly, we decided to finetune a pre-trained ResNet-18 (trained on ImageNet). This was done to ensure that the network doesn't overfit on the training data (as it is too few in number), and also, to ensure that the network trains quickly. This form of transfer learning is pervasive in Computer Vision, and can be thought of as learning the weights on highly informative features extracted through the initial layers of CNN. We used cross-entropy loss that is the standard for classification tasks. We also made use of the dropout technique as a form of regularization, where some neuron connections are randomly dropped with some probability (0.7 in our case). This ensures that the network doesn't overfit on the training data, especially when it's small (as ours is). With 13 epochs of training, we observed that the network obtained a training accuracy of 81% and a dev accuracy of 77%.

Figure 2: Learning Curve and the End-to-End Pipeline for Supervised Learning

(a) Learning Curve for ResNet CNN on our dataset



(b) Supervised Hashtag Prediction Pipeline



5 Hashtag Prediction

Using the API from all-hashtag.com, we maintained a deterministic mapping of 76 labels to popular Instagram hashtags. To generate hashtags from labels, for both supervised and ZSL methods, we follow the following procedure:

- ZSL methods are bound to have a low top-1 prediction accuracy, because it is especially difficult to discriminate between semantically similar classes such as "fashion" and "outfit". So we need to include all top-5 or top-10 predictions to generate the hashtags. Using cosine similarity in the word space as weights, we randomly sample hashtags from the mapping to generate hashtags for our test images.
- Because fully supervised method has a top-1 accuracy of 77% and a top-2 accuracy of 84%, we can safely ignore all the predictions after this, especially to avoid false alarms. Using probabilities of predictions as weights, we randomly sample hashtags from the mapping to generate hashtags for our test images.

6 Experiments and Results

Since the objective of our project is to compare ZSL with data-driven approaches, it would be fair to obtain both objective accuracies as numbers as well as subjective results as the quality of hashtags obtained.

- Objective Comparison: Top-1, Top-5 and Top-10 accuracy metrics can be used to compare ZSL and supervised methods. Note that Top-k accuracy is the fraction of times correct label is present in the top k predictions of the model. This is heavily biased towards the supervised methods, as some of the tags are semantically similar (such as "outfit" and "fashion") and it is especially difficult for ZSL methods to discriminate between the two. As can be seen from the results below, ZSL falls far behind, resulting in almost 35% difference in top-5 accuracy.

- Subjective Comparison: Comparing the hash-tags generated makes a lot more sense, but we don't have hash-tag ground-truths available. We decided to generate hashtags and survey the quality of results ourselves. Surprisingly, ZSL provides accurate hash-tags that are similar to ResNet, 7 out of 10 times.

Table 1: Objective and Subjective Comparisons

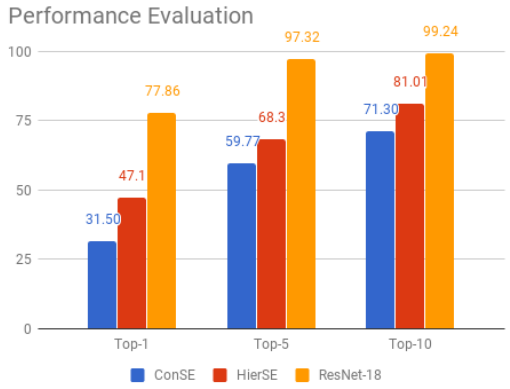
Model	**Top-1**
Supervised SVM	19
Supervised ResNet-18	77
ZSL ConSE	32
ZSL HierSE	47

(a) Objective Comparison : Top-1 Accuracies on our dataset

Model	**Hashtags**
ResNet-18	#ootd#fashion#trendy
ConSE	#outfit#whatiwore#dress

(b) Subjective Comparison : Hashtags Comparison for Image in Fig. 1

Figure 3: Comparison of Top-1,5,10 accuracy for ConSE, HierSE and ResNet-18



7 Conclusion

Based on our experiments, it turns out that data collection and cleaning, albeit computationally expensive and time consuming, leads to much better objective results. However, Zero Shot Learning classifiers give impressive subjective results, and they do it without any data, which is an exciting direction for future work. Developing an end-to-end system that recognizes image tags was a great learning experience, and we had a great time applying concepts that we learnt in class. With the advent of deep learning, traditional methods such bag of visual words and SVM have been outclassed, as verified by our experiments. But deep networks require large amount of data, which can be a bottleneck at times. Zero Shot Learning tries to mimick how humans learn, which is getting closer to a general purpose AI, and hence is an exciting field to keep track of.

8 Flask Server for Front End

We have deployed our code as a web-service using Flask[11]. Flask is a powerful web-development micro-framework in python that has support for debugging, templating, RESTful-API-like model and thus facilitates smooth deployment. The idea was to solve a real-world problem and to be able to reach many people. We expect that our improved model in the future can be actually used by many people on social-media. If given an opportunity, we would also like to deploy our project on an actual server provided by Stanford and also explore what kinds of problems can be solved by our project. We have uploaded a small demonstration of the same on Youtube (`https://www.youtube.com/watch?v=MB_hdmHf_uQ`).

9 Future Work

The next step could be to improve the accuracy of our Zero Shot Learning classifier and then comparing it to the fully supervised methods. We could then look into the more general problem of Generalized Zero Shot Learning, wherein, the test classes include training labels as well.

10 Contributions

Both the team members contributed equally to the project. Till the milestone, work was divided between the two of us. Shreyash chose the programming framework to be used for this project. He wrote most of the code for finetuning the ResNet-18 and ConSE models including word2vec [12] feature extraction, nearest neighbour algorithm. Abhijeet scraped the data (images) and processed through the cascade (feature extraction, PCA for dimensionality reduction, k-means, outlier image removal). MALTAB code for Bag of Visual Words was closely adapted from the documentation. Flask application for the web interface was made. Both of us worked equally for making the poster and reports.

11 Code

Our code for all the implemented methods including has been uploaded to github (`https://github.com/abhijit-15/zeroshotlearning`).

References

[1] Yongqin Xian et al. "Zero-shot learning-A comprehensive evaluation of the good, the bad and the ugly". In: *arXiv preprint arXiv:1707.00600* (2017).

[2] Mohammad Norouzi et al. "Zero-shot learning by convex combination of semantic embeddings". In: *arXiv preprint arXiv:1312.5650* (2013).

[3] *Pytorch - Deep Learning and GPU Computations.* URL: `https://www.pytorch.org`.

[4] Xirong Li et al. "Zero-shot image tagging by hierarchical semantic embedding". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2015, pp. 879–882.

[5] *Hierarchial Semantic Embedding.* URL: `https://github.com/li-xirong/hierse`.

[6] *Our scraper was adapted from.* URL: `https://github.com/deepanprabhu/duckduckgo-images-api`.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[8] *scikit-learn : Machine Learning in Python.* URL: `http://scikit-learn.org/stable/`.

[9] *MATLAB bag of visual words.* URL: `https://www.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html`.

[10] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[11] *Flask - microframework for Python based on Werkzeug, Jinja 2 and good intentions.* URL: `http://flask.pocoo.org/`.

[12] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).