InstaNet: Object Classification Applied to Instagram Image Streams

Clifford Huang
Stanford University
chuang8@stanford.edu

Mikhail Sushkov
Stanford University
msushkov@stanford.edu

Abstract

The growing popularity of social media has created a new medium for advertisers to promote their products. To reach the modern-day consumer, advertisers have turned toward referral marketing with top social media influencers acting as brand ambassadors. The current method of manually finding these individuals does not scale well to the growing demand of companies discovering the power of social media. We propose using convolutional neural networks (CNNs) for this task, focusing on Instagram. Our approach involves finetuning various pretrained models using new training sets of ImageNet and ImageNet + Instagram, and a validation set obtained through crowdsourcing labels for Instagram photos. Additionally, we experiment with data augmentations and network parameters. Our end-to-end system produces a list of top Instagram users judged to be the best brand ambassadors for products, given a set of predefined categories.

1. Problem Statement

At a high level this is an extension of the object classification problem: given a product category, can we identify the images representing said category and can we use this information to identify the most influential brand ambassadors for said category?

2. Introduction

Social media has transformed many aspects of human life in the past decade. In particular it has changed the way consumers think about what they buy, and with that, how advertisers promote products. The ubiquitous presence of social networks in daily life has caused modern advertisers to rely on social media influencers to promote their products [1]. In particular, Instagram has become a key platform for this type of advertising. To capitalize on this new channel of advertisement, companies such as Brandnew allow advertisers to conduct Instagram-specific campaigns by discovering the top social influencers who will likely be top brand ambassadors [5].

Presently, these ambassadors are discovered by humans; this is a costly and time-consuming manual process. It will not scale to the influx of companies when this emerging form of marketing becomes prevalent. Rather, we propose a CNN-based object classification system that automatically generates the set of brand ambassadors per product category given the Instagram image streams of the queries that best represent a particular product category. We predefine a set of 10 product categories (e.g. soccer ball, skateboard) and identify the influential users that are likely to advertise for a brand in a given category based on their post history. Given one of those 10 categories, if a user is influential (determined by our own metrics using Instagram user metadata) and has also posted a significant amount of photos in that category, then the system's output when querying for that category will include this user as a potential brand ambassador. The output will then be sorted, again using our own defined metric, by the likelihood of being an influential brand ambassador.

Given queries defining a product category, henceforth #*query*, obtaining candidate images for that category involves taking images from Instagram that have been tagged with #*query*. For instance, a bicycle retail chain that wishes to find influential bicycle enthusiasts to help advertise their new product will be interested in the product category *bicycle* and the queries #*bicycle*, #*bike*, #*biking*, etc. A quick look at the resulting #*query* streams will show these tags are very noisy; the majority of photos tagged with #*query* will not actually contain the product defined by #*query* (further detailed in the Data section below). Thus, the main challenge in building this system is to reliably classify whether a particular image actually contains an object in some category. That is, we are interested in precision and recall values in addition object classification's standard metric of accuracy.

Recent work demonstrates that CNNs are able to achieve super-human accuracy levels on standard object classification benchmarks such as ILSVRC [11], making them the expected choice for our system. Our system makes use of several CNN models, trained using the Caffe framework [8].

We compare the performance of our own model (defined below) to 3 pre-trained models (CaffeNet, GoogLeNet, Network in Network; discussed below) trained on different datasets (defined below) evaluated on human-labeled data. The final system then uses the best model to identify the top influencers for a given product. This system will only output the top influencers for its predefined product categories. To make the system more extensible, we apply transfer learning on top of the best previously-trained model.

3. Data

3.1. Product Categories

Our 10 initial product categories are: bicycle, broccoli, Christmas stocking, harp, paintball marker, Persian cat, sewing machine, skateboard, soccer ball, and tennis ball. The figure below shows an example image for each of our product categories.

Figure 1. 10 Initial Product Categories

3.2. Instagram Data

3.2.1 Instagram Stream #*query* Images

We use the Instagram API [3] to pull a variable number of images for a given hashtag. The most significant difference between ImageNet and Instagram images is that Instagram images do not follow the same constraints as ImageNet [10]. It is completely possible for an Instagram user to upload an image whose contents does not have any of the #*query* terms that were used to find said image. Furthermore, it is completely possible for an Instagram image to only contain products belonging to a different category altogether. Since Instagram breaks the assumptions of the ImageNet dataset, we henceforth refer to the unfiltered stream of #*query* as *noisy*, and the human-labeled stream as *validated*. In the following figure we show an example of the noisy data using the stream response of #bicycle.

Figure 2. Example *#bicycle* Images

From the sample images shown in Figure 2 it is clear that the labels on Instagram do not necessarily correlate to what is shown in the image. This noise makes a human annotator's job tedious to sort through, but is a perfect application for ConvNets.

3.2.2 Instagram User & Metadata

In addition to Instagram photos, we also pull metadata associated with each user that has produced a photo in our set of pulled images. For a given user, this metadata is the number of followers and an (optional) form of contact, i.e. email. For each user's image, we keep track of the number of likes, number of comments, and image creation date. These user and image attributes are used in the evaluation metric and beyond; they play no role in the classification predictions of our system.

3.3. Datasets

Given the initial 10 product categories, we obtain labeled training data for each category and define three training datasets. To eliminate the impact of image size, we follow the groundwork set by Wu et al. and choose to reshape all images to size 256x256 [13] before cropping. The first training dataset uses solely the ImageNet categories for particular objects, with roughly 1000 images per category. The second training set is an augmentation of the first: in addition to the ImageNet images it uses noisy Instagram stream responses for each category's #*query*. There are roughly 1600-2300 noisy images per #*query* response.

We add a Noise category of roughly 5000 images gathered from Instagram's most popular and random hashtags, i.e. #*yolo*, to represent the negative class, taking inspiration from Bourdev et al. [6].

Finally, we use an ensemble of Fiverr [4] labelers to label 1600 images per product category. This labeled simulation of Instagram image streams will be our test set.

3.4. Data Augmentations

For both data sets, we further apply three sets of data augmentations to produce a total of 6 distinct training sets. Caffe applies mirroring and cropping automatically, so we

refer to these as the Standard set. The Standard set is the first set of augmentations we apply. The second set of augmentations concatenates the first set with the augmentations used in the homeworks – random contrasts and random tints. The third set concatenates the second set with Instagram filters. We expect the Instagram filters to align ImageNet training images with our test set of Instagram images, where filters are often used. Figure 3 shows an example of our data augmentations for a sample image in the "bicycle" class.

Figure 3. Bicycle Image and Its Augmentations

4. Background Literature Survey

In this section we discuss the papers most influential to our final designs. Given the novelty of the Instagram dataset and the recency of applying CNNs to computer vision problems, there is unfortunately no literature exactly related to this project. However, this project generalizes to object classification with CNNs, for which there is a great deal of prior work. Primarily, we wish to focus on the works of Szegedy et al. [9], Kandaswamy et al. [7], and Yosinski et al [12].

In Szegedy et al.'s paper, the main conclusion is that accurate models are inherently tied to deep architectures, with their GoogLeNet model being 27 layers deep with pooling included. Additionally, this work notes the effectiveness of using small filters (1x1) in the deeper layers, as well as using ReLU and pooling layers throughout the network. Given GoogLeNet's state-of-the-art results, we take Szegedy et al.'s methodologies and apply them to a model of our own creation (discussed below).

The next influential pieces of background literature are the works of Kandaswamy et al. and Yosinski et al. These papers illustrate how it is possible to reuse the knowledge of a network trained on source problem S to improve the performance of a target problem T. Their discussions on transfer learning play an important role in how we finetune pretrained models to predict well on a new set of image categories. Specifically, Kandaswamy suggests that transfer-learned models need to reduce the pre-training learning rates by a particular order of magnitude, and increase the fine-tuning learning rates by an order as well. Whereas Kandaswamy uses 0.01x and 10x, respectively, we found 0.1x and 10x to be more effective in our task. Yosinski also suggests an alternative solution, where the learning rates of the pre-trained layers are "frozen" and only the learning rate of the last layer is tweaked. We discuss the results of these two methods in the Results & Analysis section below.

5. Models

We experiment with 4 different CNN models to perform 11-class classification on our 10 object classes and noise class.

5.1. Models Trained from Scratch

5.1.1 CNN1

The first model we try is a baseline CNN designed from scratch, referred to as CNN1. Figure 4 shows CNN1's results on the test set.

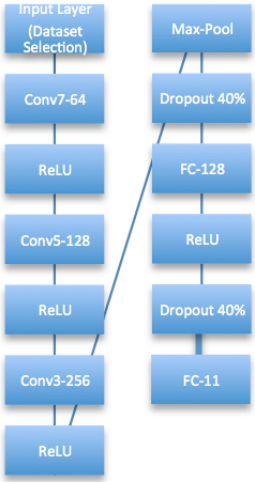

Figure 4. Architecture of CNN1

5.2. Pre-trained Models

From the results of the papers in the Background Literature Survey section, we expect transfer learning to be a faster alternative to training our own model(s) without impacting prediction accuracy. We experiment with 3 different pre-trained models (discussed below) available in Caffe's Model Zoo [2]. We chose these particular models because of their state-of-the-art results.

We modify each model slightly by changing the number of units in the last softmax layer from 1000 (used in the ILSVRC classification tasks) to 11, to account for our 11 classes. We initialize the weights of the other layers to the weights of the pretrained models, and initialize the weights of the last layer randomly. For each of the models below we follow 2 training schemes. The first follows Yosinski et al. by freezing all layers except the last (the one we changed); it

then trains the network to learn the weights of the last layer only. The second scheme, influenced by Kandaswamy et al., freezes none of the layers. Rather, it reduces the global learning rate by a factor of 100 and increases learning rate of the last layer by a factor of 10. All changes are relative to the original learning rates.

5.2.1 CaffeNet

The first pretrained model is the "Reference CaffeNet" from the Caffe Model Zoo [2]. This model is nearly identical to AlexNet, the ILSVRC2012 classification task winner [cite alexnet]. With minor tweaks it achieves a top-1 accuracy of 57.4% on the ILSVRC2012 dataset.

5.2.2 GoogLeNet

The second pretrained model is the Caffe replication of Google's GoogLeNet [citation], the winner of the ILSVRC2014 classification task. GoogLeNet achieves a top-1 accuracy of 68.7% on the ILSVRC2012 dataset. Besides its high accuracy, another advantage of GoogLeNet is the reduction in the number of parameters compared to AlexNet despite its larger depth: 5 million parameters compared to AlexNet's 60 million [cite googlenet and alexnet papers].

5.2.3 Network in Network

The last pretrained model we use is the Network in Network (NIN) architecture [citation] trained on ImageNet data. NIN achieves a top-1 accuracy of 59.36% on the ILSVRC12 dataset; its advantage is an 8-fold reduction in the number of parameters compared to AlexNet.

6. Evaluation Metrics

6.1. Object Classification Evaluation

We evaluate the correctness of our models using accuracy, precision, and recall. We use accuracy because it is the standard metric of object classification; its value allows us to compare the prediction ability of our model against the works of others and to tune the hyperparameters of our models. Its use, however, ends here. Given the nature of our problem, we are primarily interested in predicting true positives. That is, given the #*query* defining a product category, we wish to return *only* and *all* images with the product; hence, the true positives. While accuracy scores increase by correctly predicting images that do not show the product (the negative class), this is not important to our system's final output.

Before we rationalize using precision and recall, let us first define terminology using the product category *bicycle*:

- True Positive (TP) - The ensemble of annotators predicts bicycle and bicycle is one of the CNN's top-*k* predictions.
- False Positive (FP) - The ensemble predicts bicycle is not in the image and the CNN predicts bicycle.
- True Negative (TN) - The ensemble predicts bicycle is not in the image and the CNN predicts the Noise class.
- False Negative (FN) - The ensemble predicts bicycle and the CNN predicts the Noise class.

Precision, the measure of $\frac{TP}{TP+FP}$, represents how many of our returned images are relevant to the product category. A low precision value means that we are not filtering the stream for the product category well, meaning that human annotators must spend significant time filtering the CNN's predicted images. Thus, precision is proportional to how well this idea scales.

Recall is the measure of $\frac{TP}{TP+FN}$ and it represents how many images of a product category that the CNN is actually able to find. Initially, we expected low recall values to undermine our project; if we are not able to find the images representing a product category, then CNNs are not a good replacement for human annotation. However, upon analyzing the false negatives, we reassess low recall to be not as detrimental as low precision. As shown in Figure 5, the product is an insignificant part of false negative images.

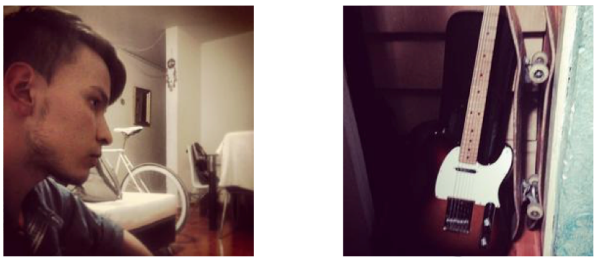

Figure 5. Examples of False Negative Images

In terms of our problem statement, false negatives and low recall are acceptable because we are interested in whether a user will be a good ambassador, not if a user happened to upload an image containing a bicycle. Hence, we maximize precision while keeping accuracy and recall above self-defined bounds.

6.2. Brand Ambassador Evaluation

The metric to evaluate the quality of candidate brand ambassadors is difficult to define without any feedback from the advertisers. It also has no relation to the subject material of this class so it will not be discussed in detail. The current metric is:

$$\frac{\#Followers+100*log(\#Likes+1.4*\#Comments)}{k}$$

Note that it is inversely proportional to the index k of the product in the CNN's list of predictions. This is intentional because the product should ideally be the focal point of the image in order to represent the product category well.

Figure 6. A Harp at different values of k

In the figure above, Harp is the primary prediction of the left-hand image while other categories (Piano, Cello, Stage, etc.) have a higher k value than Harp in the right-hand image. Inverse proportionality captures why the left-hand image's user is the more logical choice for brand ambassador.

7. Experiments & Results

There were 3 main questions we wanted to answer. First, which CNN architecture gives the best results on Instagram test data? Second, which set of data augmentations works best? Third, which training data (ImageNet only, or Instagram + ImageNet) gives best results on Instagram test data? To answer all of these questions we used a subset of a human-labeled test set of 1700 Instagram images per category.

Figure 11 shows the results for our own CNN model. Figure 7 shows the accuracy, precision, and recall as a function of k for untrained CaffeNet, GoogLeNet, and NIN models: for each of these 3, the test set accuracy, precision, and recall are evaluated using the 1000 ILSVRC12 classes, and the results are reported for the 8 classes out of our 10 that overlap with those 1000. Figure 8 shows results for all 3 models after transfer learning on only the last layer, and freezing the rest. The learning rate used to train the last layer was equal to the original learning rate used to train the GoogleNet model. Figure 9 shows results for all 3 models after transfer learning on all the layers, but with the global learning rate reduced by 100x and the learning rate for the last layer increased by 10x. Figure 10 shows the results of varying the training data, for each of the 3 sets of data augmentations (standard, standard + tint + contrast, standard + tint + constart + Instagram filters), for both just ImageNet training data and the combined ImageNet and noisy Instagram training data.

Model	k	Accuracy	Precision	Recall
CaffeNet	1	0.7050	0.4411	0.2736
	10	0.78	0.5389	0.5094
	20	0.76	0.5025	0.5280
GoogLeNet	1	0.7550	0.4561	0.4149
	10	0.7700	0.4757	0.5907
	20	0.7350	0.4345	0.6008
Network in Network	1	0.7050	0.3758	0.2714
	10	0.7650	0.5140	0.4954
	20	0.7700	0.4944	0.5720

Figure 7. Accuracy, precision, and recall in untuned models as a function of k.

Model	k	Accuracy	Precision	Recall
CaffeNet	1	0.6720	0.5300	0.4523
	2	0.6800	0.4984	0.7308
GoogLeNet	1	0.7560	0.6208	0.6169
	2	0.6920	0.5072	0.7717
Network in Network	1	0.7360	0.6501	0.5277
	2	0.6880	0.4968	0.7064

Figure 8. Accuracy, precision, and recall in finetuned models, with only the last layer trained and the rest frozen.

Model	k	Accuracy	Precision	Recall
CaffeNet	2	0.7360	0.5425	0.7622
GoogLeNet	2	0.7120	0.5169	0.7610
Network in Network	2	0.7080	0.5282	0.7099

Figure 9. Accuracy, precision, and recall in finetuned models, with all the layers trained.

Data Augmentations	Data Set	k	Accuracy	Precision	Recall
Standard	ImageNet	2	0.7120	0.5169	0.7610
	ImageNet + Instagram	2	0.7080	0.5282	0.7099
Standard + Tint + Contrast	ImageNet	2	0.7160	0.5165	0.7840
	ImageNet + Instagram	2	0.4720	0.4283	0.6872
Standard + Tint + Contrast + IG Filters	ImageNet	2	0.7120	0.5289	0.7962
	ImageNet + Instagram	2	0.4840	0.4339	0.6930

Figure 10. Accuracy, precision, and recall in finetuned models, with all the layers trained, as a function of the training data.

Model	k	Accuracy	Precision	Recall
CNN1	2	0.5840	0.1021	0.1148

Figure 11. Results for CNN1.

The following plot demonstrates the accuracy, precision, and recall of our 10 product categoreies on one of our best models, NIN, with top-1 accuracy and trained by freezing all layers except the last. One reason we choose to demonstrate this model is because of its high precision values. As can be seen in some cases, i.e. harp and soccer ball, the model has achieved precision of 1.0. While it is not perfect in every case, we find it to be the best model overall.

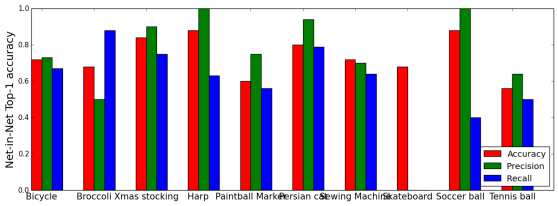

Figure 12. Results for our best model (NIN with all but the last layer frozen) for each of the 10 categories.

8. Discussion

8.1. Error Analysis

Figure 13. Examples of False Positive Images

Figures 5 and 13 are prime examples of false negative and false positive images respectively. The left hand image of Figure 5 shows a common error of our system. When the product either blends in with the background or is obstructed (this particular example illustrates both of these bad situations), it is difficult for the model to find the product. The right hand image of the same figure shows a different weakness in our system; when a product is not the main subject of an image, it will not appear as one of the top-k choices. In this image, the skateboard is easily overshadowed by the guitar. While these issues are weaknesses of a general object classification system, we see them as a blessing in disguise. As discussed in the the Evaluation Metrics section, these images are rather unexciting ways to showcase the bicycle and skateboard products. Thus, it is acceptable to leave the owners of these images out of our list of brand ambassadors.

The left hand image of Figure 13 is an example of the most harmful type of image to our system. In this image, we wrongly predict a Harp to be in the image when a human annotator is able to clearly spot that Harp does not exist in said image. These images are the worst for our system because they defeat the purpose of applying CNNs to our problem statement; if human annotators are required to filter the output of our CNNs, then CNNs face the same issues of scale as current methods. The right hand image of this figure is an interesting case. Our best model believes there is a Persian cat in this image, while humans faced with the same task believe there is not. Without being an expert in felines, it is difficult to say whether the animal in this image is truly a Persian cat, or simply a cat. This means there are two possible issues. The first is that our CNN will predict the closest label; if the cat category does not exist, the CNN is more likely to predict Persian cat than Noise. The second possibility is that there may be instances where our CNN is more capable than human annotation; its classification is correct whereas the annotators are incorrect. Regardless, we feel this error is on the same level as false negative errors and not as impactful as the first case of false positives.

8.2. Brand Ambassadors & Viability of This Approach

For brevity, Figure 14 only shows the top ambassadors for 5 of our initial 10 product categories.

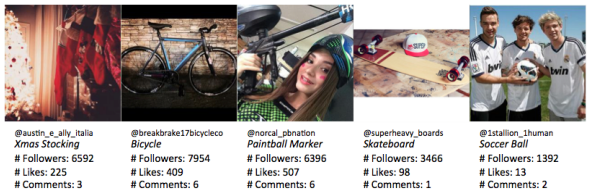

Figure 14. Candidate Brand Ambassadors

From the images, one is able to see that the product is cast in a visually appealing manner. From the metadata associated with each image, one is also able to note that the image owners are quite infuential. Another interesting note about the top influencers is that an ample number are already associated with a storefront or indie-name brand. It can be up to the big-name brands to decide whether to ally with these users or to filter them from the list of influencers completely. From these results, CNNs seem to be an effective method in scaling the growing trend of social media marketing.

References

[1] http://instabrand.com/.

[2] https://github.com/bvlc/caffe/wiki/model-zoo.

[3] https://instagram.com/developer/.

[4] https://www.fiverr.com/.

[5] http://www.brandnew.io/.

[6] L. D. Bourdev, F. Yang, and R. Fergus. Deep poselets for human detection. *CoRR*, abs/1407.0717, 2014.

[7] L. A. Chetak Kandaswamy, Luis Silva. Report: Improving CNN by Reusing Features Trained with Transductive Transfer Setting. 2014.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. 2014.

[11] C. S. Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[12] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[13] Y. Y. L. W. T. T. Zifeng Wu, Yongzhen Huang. Early Hierarchical Contexts Learned by Convolutional Networks for Image Segmentation. 2014.