Image-based Product Recommendation System with Convolutional Neural Networks

Luyang Chen
Stanford University
450 Serra Mall, Stanford, CA
lych@stanford.edu

Fan Yang
Stanford University
450 Serra Mall, Stanford, CA
fanfyang@stanford.edu

Heqing Yang
Stanford University
450 Serra Mall, Stanford, CA
heqing@stanford.edu

Abstract

Most on-line shopping search engines are still largely depend on knowledge base and use key word matching as their search strategy to find the most likely product that consumers want to buy. This is inefficient in a way that the description of products can vary a lot from the seller's side to the buyer's side.

In this paper, we present a smart search engine for on-line shopping. Basically it uses images as its input, and tries to understand the information about products from these images. We first use a neural network to classify the input image as one of the product categories. Then use another neural network to model the similarity score between pair images, which will be used for selecting the closest product in our e-item database. We use Jaccard similarity to calculate the similarity score for training data. We collect product information data (including image, class label etc.) from Amazon to learn these models. Specifically, our dataset contains information about 3.5 million products with image, and there are 20 categories in total. Our method achieves a classification accuracy of 0.5. Finally we are able to recommend products with similarity higher than 0.5, and offer fast and accurate on-line shopping support.

1. Introduction

The on-line retail ecosystem is fast evolving and on-line shopping is unavoidable growing around the world. A digital analytics firm eMarketer shows that on-line retail sales will continue double and account for more than 12% of global sales by 2019. As reported in the result of the Nielsen Global Connected Commerce Survey (2015)[1], 63% of respondents who shopped or purchased the travel products or services category, for example, in the past six months say they looked up the product on-line.

[1]The Nielsen Global Connected Commerce Survey was conducted between August and October 2015 and polled more than 13,000 consumers in 26 countries throughout Asia-Pacific, Europe, Latin America, the Middle East, Africa and North America.

However, the explosive growth in the amount of available digital information has created the challenge of information overload for on-line shoppers, which inhibits timely access to items of interest on the Internet. This has increased the demand for recommendation systems. Though almost every e-commerce company nowadays has its own recommendation system that can be used to provide all sorts of suggestions, they are mostly text-based and usually rely on knowledge base and use key word matching system. This requires on-line shoppers to provide descriptions of products, which can vary a lot from the sellers' side to the buyers' side.

With the rapid development of neural network these recent years, we can now change the traditional search paradigms from text description to visual discovery. A snapshot of a product tells a detailed story of its appearance, usage, brand and so on. While a few pioneering works about image-based search have been applied, the application of image matching using artificial intelligence in the on-line shopping field remains largely unexplored. Based on this idea, here we build a smart recommendation system, which takes images of objects instead of description text as its input.

The input to our algorithm is an image of any object that the customer wants to buy. We then use a Convolutional Neural Network(CNN) model to classify the category that this object probably belongs to, and use the input vector of the last fully connected layer as a feature vector to feed in a similarity calculation CNN model to find the closest products in our database. More concretely, the two functionalities that we want to achieve in the recommendation system are:

1. Classification: given a photo of the product taken by the customer, find the category that this product most likely belong to. We have 20 categories in our data set in total. The details of these categories are shown

in the **Datasets and Features** section. For example, a image of iPhone(s) will be classified as “Cell Phones & Accessories”.

2. Recommendation: given the features of the photo and the category that this product belongs to, calculate similarity scores and find the most similar products in our database. Ideally, people looking for iPhones should be recommended iPhones.

2. Related Work

Paper [6] presented an idea of combining image recommendation and image recommendation decades ago. In this project, we use Amazon product dataset, which is used to build typical recommender system using collaborative filtering in [4] and [8]. In the field of image recommendation, [5] tends to recommend images using Tuned perceptual retrieval(PR), complementary nearest neighbor consensus (CNNC), Gaussian mixture models (GMM), Markov chain (MCL), and Texture agnostic retrieval (TAR) etc. CNNC, GMM, TAR, and PR are easy to train, but CNNC and GMM are hard to test while PR, GMM, and TAR are hard to generalize. Also, since data consists of images, the neural network should be a worth trying method.

Paper [7] presented AlexNet model that can classify images into 1000 different categories. In adddition, paper [9] presented VGG neural network that classify images in ImageNet Challenge 2014. In our first part of project, we use both models to classify the categories of the products. However, both papers did not present a method for image recommendation.

Although there are papers that studies image similarity such as [12] and [11], most of them are based on category similarity, i.e. products are regarded as similar if they are in the same category. However, products that come from the same category can still vary a lot. Thus, one reliable strategy is to first classify target image into a certain category and then recommend images from this classified category.

In paper [13], they considered using neural network to calculate the similarities within category. However, the paper only consider ConvNet, DeepRanking etc. Since we have larger dataset, deeper convolutional neural network such as AlexNet and VGG should outperform naive ConvNets. The idea could be also found in [3] and [10].

Paper [1] is also focusing on learning similarity using CNN. However, it considers more on the case that multi-product contained in a single image. In our project, we assume that users are looking for a product and so image would only contains one product.

Before we recommend, we need to answer What is the measurement of similarity. The most nature answer is either cosine similarity or L_2 norm similarity. Another way to measure the similarity is by introducing semantic information. The paper [2] indicates that visual similarity and semantic similarity are correlated. Thus, we introduce a new model to calculate similarities between images based on semantic information. Paper [15] and [14] share the same idea as we do here.

3. Approach

There are two major problems that we want to solve in our project. First, determine the category that a given image belongs to; second, find and recommend the most similar products according to the given image. Since our project is mainly based on convolutional neural network, we would first introduce common used convolutional neural network layers.

3.1. CNN Layers

The most important step of CNN is Convolutional(Conv) layer. As we can see from Figure 1 that conv layer would translate small rectangle of input layer into a number of output layer using matrix multiplication.

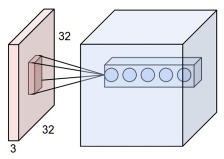

Figure 1. conv layer

Pooling layers are similar to convolutional layers except that it would use non-parameter method to transform small rectangle into a number. Max pooling are commonly used in CNN, which would output the maximum number in the rectangle of input layer.

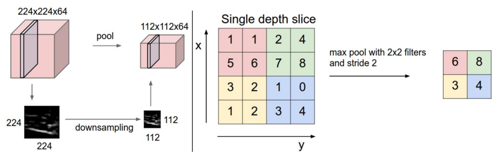

Figure 2. pooling layer

3.2. Classification

In this step, we would like to classify an input image into one of the 20 categories. We construct AlexNet and VGG model for the classification task and compare them with SVM model as a baseline model.

- **Support Vector Machine**: a linear classification model, used as a baseline model here. This model is basically a fully connected layer. We use Multi-class Support Vector Machine (SVM) loss plus a L_2

norm term as the loss function. For an image i, we use the RGB pixels as input features $x_i \in \mathbb{R}^d$, where $d = 224 \times 224$. We calculate the class scores for $n = 20$ classes through a linear transformation

$$s = Wx_i + b \tag{1}$$

where $W \in \mathbb{R}^{n \times d}$ is the weight matrix and $b \in \mathbb{R}^n$ is the bias term. The SVM loss is given by

$$L_{SVM}(W, b; x_i) = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{2}$$

where y_i is the label for the true class.

- **AlexNet**: a deep convolutional neural network classification model proposed by [7]. As we can see, (Figure 3) AlexNet model first contains 2 convolutional layers with max pooling and batch normalization; then there are 3 convolutional layers with separated feature; one max pooling before three fully connected layers.

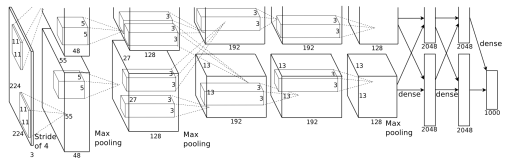

Figure 3. AlexNet model

The original model was trained to classify images in the ImageNet LSVRC-2010 content, where there were 1000 categories. Since our problem only contains 20 categories, we change the last fully connected layer to 4096×20. To save time, we use the pre-trained weights of the first 5 neurons and train the last three fully connected layers.

- **VGG**: a deep convolutional neural network classification model proposed by [9]. As showed below (Figure 4), VGG contains 13 convolutional layers with max pooling every 2 or 3 convolutional layers; then 3 fully connected layers and softmax as the final layer.

The original model was trained to classify images in the ImageNet ILSVRC-2014 content, where there were 1000 categories. We change the last fully connected layer to 4096×20. We also utilize the pre-trained weights as initialization of parameters and to train the last three fully connected layers. We also add batch normalization layers after the activation functions in the first two fully connected layers.

3.3. Recommendation

For the recommendation step, we use the last fully connected layer in our classification model as feature vectors

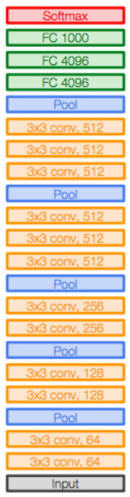

Figure 4. VGG model

of images. For any images in the dataset, there will be one corresponding feature vector. And this feature vector will be the input for our recommendation model. The work flow of this step is shown in the following bullets.

- Feature extraction: the classification model is used to identify which category the target image belongs to. Then we extract the input from last fully connected layer of classification model as features.

- Input of the model: the feature vector of the target image extracted in the above.

- Similarity calculation: using different measures to calculate similarity scores between feature vector of target image and feature vectors of all images in the target category to measure similarity between image pairs. We have tried L_2 distance, cosine distance and neural network models to compute the similarity scores. For two different images i and j, the L_2 distance score is defined as

$$s_{L_2} = \|v_i - v_j\|_2 \tag{3}$$

where $v_i, v_j \in \mathbb{R}^l$ are the two corresponding feature vectors, and $l = 4096$ is the length of feature vectors. The smaller the score s_{L_2} is, the more similar the two images are.

The cosine distance score is defined as

$$s_{cosine} = \frac{v_i^\top v_j}{\|v_i\|\|v_j\|} \tag{4}$$

The larger the score s_{cosine} is, the more similar the two images are.

The data-driven approach to calculate the similarity score is to train the following 3-layer neural network:

$$\begin{aligned} h_1 =& f(v \cdot W_1 + b_1) \\ h_2 =& f(h_1 \cdot W_2 + b_2) \\ s_{model} =& \text{sigmoid}(h_2 \cdot W3 + b_3) \end{aligned} \tag{5}$$

where $v = [v_1, v_2] \in \mathbb{R}^{l \times 2}$ is obtained by concatenating two feature vectors. $f(x) = \max(0.01x, x)$ is the leaky ReLU function. The first layer can be treated as a 1-d convolution layer with Leaky ReLU as the activation function and $W_1 \in \mathbb{R}^2$ and $b_1 \in \mathbb{R}$ as parameters. The second layer is a fully-connected layer with Leaky ReLU as the activation function and $W_2 \in \mathbb{R}^l$ and $b_2 \in \mathbb{R}$ as parameters. The output layer is a linear transformation with the sigmoid function as activation function. The larger the score s_{model} is, the more similar the two images are. There is no easy way to define similarity score purely based on the image pixels. Fortunately, the input images has a corresponding title describing the product. To characterize how similar two images are, we use the Jaccard similarity of two sets of tokens in the titles of two images as the similarity of two images. The Jaccard similarity of two set A and B is defined as

$$s_{Jaccard} = \frac{|A \cap B|}{|A \cup B|} \tag{6}$$

which is a number between 0 and 1. This is also the reason that we use the sigmoid function as the activation function for the last layer. We train this model by minimizing the L_2 loss $\|s_{model} - s_{Jaccard}\|_2^2$.

- Output: top k images (products) that are most similar to the target image.

4. Dataset and Features

For building the recommendation system, we use Amazon product image data, spanning May 1996 July 2014, which includes 9.4 million products. Excluding the ones that lack images, we collected a dataset of 3.5 million products, with 20 categories in total. Figure 5 shows the distribution plot of all the labels of the dataset. The detailed information of each image contains:

- **asin** - ID of the product, e.g. 0000031852
- **title** - name of the product
- **price** - price in US dollars (at time of crawl)
- **imUrl** - url of the product image
- **related** - related products (also_bought, also_viewed, bought_together, buy_after_viewing)
- **salesRank** - sales rank information
- **brand** - brand name
- **categories** - list of categories the product belongs to

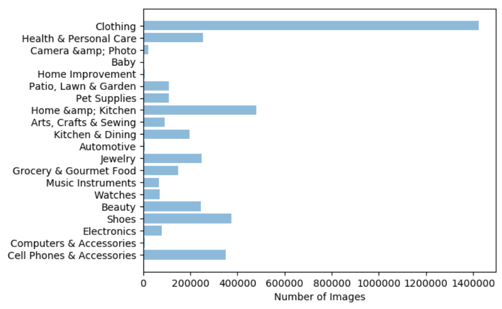

Figure 5. Label distribution of the dataset.

Considering the imbalance across different classes and due to the limitation of the machine memory, we randomly sample 500 images in each class and collect 10000 images for the classification task. Then we split the dataset into 7:2:1 for training, validation and testing respectively. Each image in the dataset has 300×300 pixels. We use raw pixels of images as the input for our classification neural network model. Examples of the data are shown in Figure 6. For the convenience of tuning hyper-parameters, we resize images into $224 \times 224 \times 3$ using "scipy.misc" for VGG, and resize into 227×227 for AlexNet.

Figure 6. Examples of the data. These are three products from category "Computers & Accessories".

5. Experiment

5.1. Data preprocessing

The raw images need pre-processing before being used as inputs of the classification models. An original image is first resized into the standard input size of either VGG model (224×224) or AlexNet model (227×227). Then it is demeaned in each channel (Figure 7).

For the recommendation model, the ground truth similarity is defined using the Jaccard similarity (6) of sets of tokens in the **title** of two images. (**title** information is attached to each image in the dataset.) However, we care more about

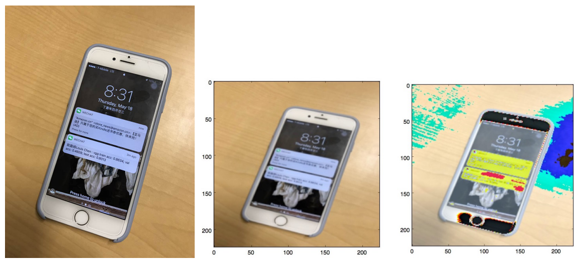
Figure 7. Preprocessing of the input image. The left is the original input image (960 × 1280 pixels). The middle is the resized image (224 × 224 pixels). The right is the demeaned image.

the pair of images that are similar. If we use all the data, the majority of the pairs will have similarity scores close to 0. Therefore, instead of using all the pairs, we only consider pairs within the same category. Moreover, we particularly want to find image pairs that have relatively large Jaccard similarity. We find those pairs with similarity scores above 0.5 and there are around 800 such pairs. We also sample 1000 pairs with similarity scores equal to 0. We use these pairs as training examples.

5.2. Evaluation

We split the dataset into $7 : 2 : 1$ for training, validation and test respectively. To evaluate the models, we run our models on the test dataset and compare the output with the ground truth.

For classification problem, we evaluate the model by calculating classification accuracy:

$$\text{Accuracy} = \frac{\#\text{correctly classified images}}{\#\text{images in validation dataset}}. \quad (7)$$

For the recommendation task, we evaluate the model using the root mean square error (RSME).

$$\text{RSME} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_{model} - s_{Jaccard})^2} \quad (8)$$

5.3. Classification

For the classification task, we have trained two Convolutional Neural Networks (VGG16 and AlexNet) to classify the categories of products images versus our baseline model – linear classification model (SVM model).

Table 1 shows our best accuracy on training data, validation data and test data for these three models respectively. For SVM model, we use learning rate 0.0005, regularization coefficients 0.001. For AlexNet, we use mini-batch size 128, regularization coefficient 0.01, learning rate 0.001 and dropout 0.6[2]. For VGG model, we use mini-batch size 100, regularization coefficient 0.01, learning rate 0.0007 and dropout 0.5. We can see from the result that our models suffer from the over-fitting problem. The training accuracy of both AlexNet model and VGG model are almost 1.5 times of the accuracy for their validation set and test set. That is why we need relatively higher regularization coefficients(0.01). But as we increases the regularization coefficient, the test accuracy does not go up any more. For the same reason, the dropout fractions we choose for these two models are also relatively large (0.6 and 0.5 respectively). However, we still suffer from the over-fitting to some extent.

Model	training accuracy	validation accuracy	test accuracy
SVM (baseline)	0.2616	0.1807	0.2679
AlexNet	0.6484	0.4064	0.3946
VGG	0.8769	0.5110	0.5010

Table 1. Model accuracy results of AlexNet and VGG compared with baseline

5.4. Recommendation

For the recommendation task, we trained a neural network model described in section 3.2. We evaluate the model using RMSE. Table 2 shows our best errors on training data, validation data and test data. The model is trained with regularization coefficient 0.02, learning rate 0.001 and dropout 0.9. To avoid over-fitting, we have a add the regularization term and choose a relatively large coefficient. Here the dropout fraction is relatively small (0.1) because the number of images within one category is limited, unlike the situation in classification task where we have image data in all 20 categories.

training error	validation error	test error
0.1318	0.1448	0.1524

Table 2. RMSE of the neural network model for recommendation

There is no baseline similarity scores for two metrics L_2 distance and cosine distance. We cannot provide a RMSE for these two metrics.

Not many e-commercial platforms have the feature of searching items by images. By now, we find that the Amazon mobile App has such feature. Thus we compared our recommendation results with theirs. Figure 8 shows two examples of outputs using our recommendation system and the Amazon App. We use the VGG model to predict the

[2]A dropout coefficient of 0.6 in our model means at each layer, 0.4 fraction of the neurons' values are set to 0. We denote 0.4 in this example as dropout fraction. This applies to all the other dropout coefficient in our report.

category and extract features. Our recommendation is based on the cosine similarity score, since we find out it outperforms the L_2 similarity. The left column shows the input images that the user took. The middle column shows the top four similar products that our system recommends. The right column shows the top four similar products that the Amazon App recommends.

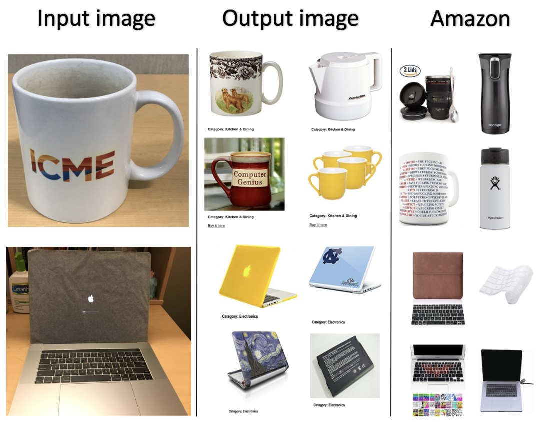

Figure 8. Examples of our recommendation system results, compared with Amazon mobile App search results.

As we can see from the example results, one of the input images is a mug. Our model recommends similar mugs or a mug-shape-like pot; while the Amazon App recommends either mugs or bottles. Results like this example give similar recommendations between our model and Amazon's. The other input image is a laptop. Our model suggests we might be searching for a Mac laptop (shown in the first two images); while the Amazon App mostly recommends keyboards/keyboard protectors. Results like this example imply that our model may understand the content of images better.

6. Conclusion

In this project we build a smart shopping recommender for image search. We tried out different neural network models for image classification and different ways to quantify the similarity between two images. We are able to achieve a classification accuracy of 0.5 and recommend products with similarity score higher than 0.5. There is over-fitting issue in our model, which can be one of the things to do in future work.

As shown in the Dataset and Features section, though we have a huge data set, due to the limitation on time and machine memory, we only used 10,000 out of 3.5 million images. In the next step, we can try to train our model on a larger amount of data using batches. This can potentially increase the accuracy of the model.

Currently we only use 20 categories when doing classification. However, products within category varies a lot, which explains our low accuracy in classification. We would try to find a more specific category information and train our model on it.

Besides, We would also like to try deeper neural networks such as ResNet.

7. Appendices

7.1. UI

We build a user interactive App for our recommendation system. On this UI, we can upload images or take photos and get the recommendation from our models. Figure 9 shows the interface of the UI given a watch image example.

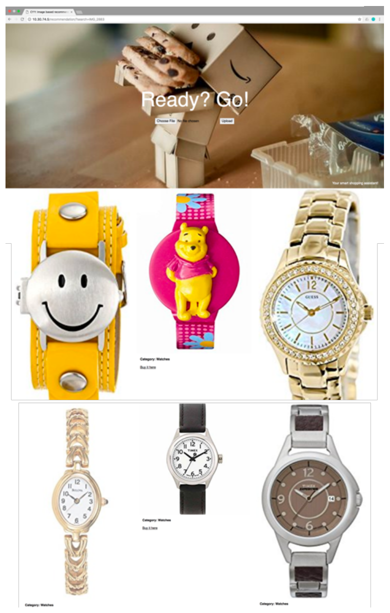

Figure 9. Examples of our recommendation system results, compared with Amazon mobile App search results.

References

[1] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98, 2015.

[2] T. Deselaers and V. Ferrari. Visual and semantic similarity in imagenet. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1777–1784. IEEE, 2011.

[3] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[4] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference*

on World Wide Web, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.

[5] V. Jagadeesh, R. Piramuthu, A. Bhardwaj, W. Di, and N. Sundaresan. Large scale visual recommendations from street fashion images. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1925–1934. ACM, 2014.

[6] I. Kanellopoulos and G. Wilkinson. Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, 18(4):711–725, 1997.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] M. Tan, S.-P. Yuan, and Y.-X. Su. A learning-based approach to text image retrieval: using cnn features and improved similarity metrics. *arXiv preprint arXiv:1703.08013*, 2017.

[11] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus. Learning invariance through imitation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2729–2736. IEEE, 2011.

[12] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr groups using stochastic intersection kernel machines. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 428–435. IEEE, 2009.

[13] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[14] J. Yang, J. Fan, D. Hubball, Y. Gao, H. Luo, W. Ribarsky, and M. Ward. Semantic image browser: Bridging information visualization with automated intelligent image analysis. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 191–198. IEEE, 2006.

[15] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.