Product Type Classification for Ecommerce with a Convolutional Neural Network

Laura Chambers

Abstract

The COVID-19 pandemic accelerated ecommerce, with Canadian retail ecommerce sales growing 70.5% in 2020 [1]. Many brick-and-mortar retailers have been pushed online or risk going out of business, enabled by platforms like Shopify. One of the first barriers retailers face moving online is digitizing their products. RendAR makes it fast and easy for retailers to capture product photos and data (i.e., mass and dimensions for shipping) and automatically add them to their Shopify store. The RendAR system includes a physical capture rig, an iPhone app, cloud-based image processing, and integration with the Shopify API.

This paper expands on the product digitization capabilities of RendAR. A convolutional neural network (CNN) using the ResNet architecture is employed to automatically populate the Product Type data field in Shopify. As a starting point, the classifier recognizes four categories of footwear: Shoes, Boots, Sandals, and Slippers. The CNN has 18 layers and was trained with four epochs, a batch size of 64, and a cyclic learning rate in the range of 1.74E-03 to 1.74E-02. The CNN was trained using the UT-Zappos50K dataset, achieving an accuracy of 96% percent. To improve the model further, additional hyperparameter tuning should be done. The dataset could also be augmented to balance the relative proportion of images belonging to each class.

Introduction

The COVID-19 pandemic accelerated ecommerce. In 2020, ecommerce retail sales in Canada increased 70.5% in response to physical distancing and store closures [1]. Brick-and-mortar retailers have been pushed to move their stores online or risk going out business. As a result, merchants have become increasingly reliant on ecommerce software platforms like Shopify, which powers over 1 million businesses worldwide [2].

While Shopify makes it easier than ever to set up and manage an online store, merchants still face challenges. This is especially true for retailers without technical skills and resources. One of the first barriers retailers face is digitizing their products. To sell products online, retailers much capture digital media and data to represent their products to customers.

“On white” product photos are the standard form of digital media used in ecommerce. As shown in Figure 1, on-white product photos are typically high quality images with low noise, high resolution, and white backgrounds. In term of product data, there are up to 36 data fields per product that merchants must manually collect and populate in their Shopify backend. Currently, retailers learn to capture product images and data themselves, which is time consuming, repetitive, and tedious, or hire someone to do it for them, which is expensive.

Figure 1: Example on-white product photos

RendAR lets retailers easily capture consistent, high-quality product images and data and automatically add them to their Shopify store. The RendAR system includes a capture rig and an iPhone app that communicate via Bluetooth to capture on-white product images from multiple angles, and mass and dimensions for shipping. This data is then sent to a cloud server for image processing. The cloud server interfaces with the Shopify API to create a new product page in the retailer’s online store.

Cloud-based image processing is used to enhance images and populate additional Shopify data fields for the product, eliminating the need for manual data entry. The goal of this paper is to build and train a model to classify the Product Type data field using a convolutional neural network (CNN). Product Type is used to manage and organize products within an online catalogue. As a starting point, the model classifies footwear based on four classes: Shoes, Boots, Sandals, and Slippers. The model is trained using the UT-Zappos50K dataset.

Background Review

Global spending by retailers on artificial intelligence is projected to grow from $3.6 billion in 2019 to $12 billion by 2023 [3]. To date, product recognition has been proposed for a number of retail applications including automated self-checkout [4], shelf-restocking [5], and assistance for visually-impaired customers [6]. Little work has been done in the specific application of product recognition for ecommerce automation.

There are two categories of product recognition: coarse-grain and fine-grain. Coarse-grain classification differentiates between objects with obvious differences. Fine-grained classification differentiates between objects with subtle differences. For example, shoes have fine-grain differences such as rounded versus pointed toe [7].

There are five steps in the product recognition process [8]. First, images are captured. Next, images are pre-processed to enhance image quality, reduce noise, resize, remove the background, etc. The resulting images are analyzed for feature extraction. Once features are mapped to the feature space, a decision rule is implemented to classify features. Finally, the product recognition pipeline outputs the class of the retail product.

Classic methods of feature extraction in computer vision include the Scale Invariant Feature Transform (SIFT) [9], speeded up robust features (SURF) [10], and binary robust invariant scalable keypoints (BRISK) [11]. SIFT was first introduced in 1999 by David Lowe. It has several advantages, including rotation and translation invariance [8]. In 2006, Bay, Tuytelaars, and Van Gool introduced SURF, which improves on the calculation speed and efficiency of SIFT [10]. Leutenegger, Chli, and Siegwart published BRISK in 2011, which is an order of magnitude faster than SURF [11].

Following feature extraction, features can be represented as PGIST (pyramid of Gist), PHOW (pyramid histogram of words), PHOG (pyramid histogram of gradient) and PLBP (pyramid of local binary pattern) [7,12] and classified with state vector machines (SVMs) [12]. A combination of PHOW, PHOG, PLBP, PGIST, and SMV achieved a product recognition accuracy of 86.6% on the PI 100 dataset [12].

More recently, deep learning via convolutional neural networks has become a preferred method of image recognition and object detection [8]. LeCun et al. first proposed the CNN, LeNet, to classify images in 1988 [13]. Since then, many networks have been developed based on LeNet, including AlexNet, GoogLeNet, VGG, and ResNet [8]. A ResNet-based CNN with an additional Local-Concepts-Accumulation (LCA) layer achieved product classification accuracies of 72.3% on the Grozi-120 dataset, 92.2% on CAPG-PG, and 100% on DM4VM [14].

This paper focuses on a constrained, fine-grained product recognition problem to validate a practical use case for novel ecommerce automation using existing AI methods.

Application/Dataset

The dataset used to create the RendAR Product Type classifier is the UT-Zappos50K dataset publicly available from the University of Texas. The UT-Zappos50K dataset contains 50,025 images of shoes collected from Zappos.com, an American online shoe and clothing retailer.

The dataset is divided into four main categories: 1) shoes, 2) sandals, 3) slippers, and 4) boots. Table 1 shows the breakdown of classes and the number and percentage of images in each class. Each major category is further subdivided based on functional categories and brands and labelled with eight additional metadata, including material and gender. Subcategories and metadata are not used for the purposes of this paper.

Table 1: UT-Zappos50K Dataset

Class	Number of Images	Percentage of Dataset
Shoes	30,170	60%
Sandals	5,742	11%
Slippers	1,284	3%
Boots	12,879	26%
Total	50,075	100%

Figure 2 shows a sample image from each major category in the dataset. The dataset is comprised of on-white product photos of left-footed items of footwear, comparable to the kind of photos captured by RendAR. There is one photo per product. Products are oriented at an angle, with the outer side of the footwear facing the camera. The dataset includes footwear for men, women, boys and girls. Each image is an RBG photo with a resolution of 136 x 136 pixels.

Figure 2: Sample image of each major category: shoe, sandal, slipper, boot (left to right)

This dataset was chosen because the images are comparable to images captured by the RendAR system. Therefore, a model trained on this dataset is likely to be generalizable to automating product digitization with RendAR.

Proposed Scheme/Algorithms

The UT-Zappos50K is a labelled dataset suitable for supervised learning. The proposed supervised learning approach is a convolutional neural network (CNN) based on the ResNet architecture. CNNs are beneficial for image processing because they leverage the spatial correlation of pixels. As a result, they require fewer parameters compared to fully-connected networks, which reduces the risk of overfitting.

ResNet is a deep residual network that was developed to solve the diminishing gradient problem. Figure 3 shows ResNet architectures ranging from 18 to 152 layers taken from PyTorch.org. Specifically, ResNet18 was chosen for this application because it enables transfer learning while requiring less training time and fewer parameters than ResNet architectures with more layers. ResNet has been pre-trained on millions of images via ImageNet. The goal of the proposed method is to train ResNet18 on the UT-Zappos50K dataset such that the weights are fine-tuned to recognize the desired shoe types.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3: ResNet architectures

The CNN will be implemented using the fastai library, which is a deep learning library built on top of PyTorch. Figure 4 outlines the CNN implementation steps. First, an `ImageDataBunch` object is initialized to prepare the data. Preparation includes resizing images to 224 x 224 pixels, which is standard for ResNet, and splitting the data into training (80%) and testing sets (10%). Next, the `cnn_learner()` method is used to instantiate the CNN. The CNN is configured to normalize the data as required by ResNet. Finally, the CNN in trained using the `fit_one_cycle()` method, which uses cyclical learning rates to improve training speed and accuracy [15]. The network is unfrozen, meaning all weights in the network are updatable.

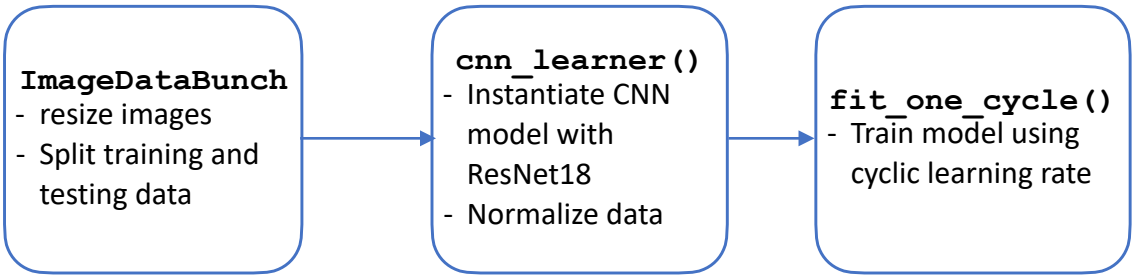

Figure 4: Implementation of CNN

Experiments and Results

Data Preparation

The dataset was downloaded and loaded into an `ImageDataBunch` object. Images were resized from 136x136 to 224x224 to match the standard input size for ResNet. The dataset was split into training (80%=40,020 images) and testing (20%=10,005 images) sets. Images in the UT-Zappos50K dataset are of comparable quality and orientation as images captured by RendAR; therefore, augmentation was deemed unnecessary.

Initial Performance

The ResNet18 CNN was instantiated and tested before training on the dataset to set an accuracy benchmark for performance. Figure 7 shows the initial confusion matrix. Performance is poor, with many misclassifications. Shoes had the highest accuracy and Boots and Sandals had no hits. Common misclassifications were Slippers as Shoes.

Training

To train the model, first the `lr_find()` method was used to find the optimal range of learning rate. Figure 5 shows a plot of the output of the learning rate finder. The optimal range of learning rate was chosen from the midpoint of the steepest section of the plot, from 1.74E-03 to 1.74E-02. This learning rate range was subsequently used for training the CNN with the `fit_one_cycle()` method.

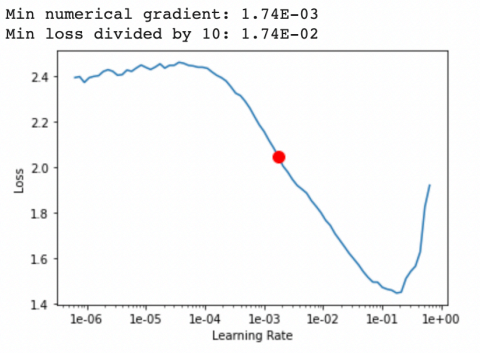

Figure 5: Learning rate optimization

The model was trained with 4 epochs and a batch size of 64. The rate of convergence and accuracy of the model are dependent on the learning rate and the number of epochs. Fewer epochs are required with a faster learning rate; however, the model may not converge if the learning rate is too high. Table 2 summarizes the results of training. The model has an error rate of 4%, meaning an accuracy of 96%. The comparable losses between training and validation suggest that the model has not overfit the data; however, the difference increased in epoch 4.

Table 2: CNN Training Results

Epoch	Training Loss	Validation Loss	Error Rate
0	0.485053	0.473556	0.135932
1	0.216397	0.198500	0.06567
2	0.135579	0.124017	0.044778
3	0.072440	0.115347	0.040080

Evaluation

Figure 6 shows the confusion matrices for the trained model compared to the initial untrained model. The accuracy of predictions for each class is 96.8% for Boots, 91.3% for Sandals, 97.3% for Shoes, and 77% for Slippers. Figure 7 shows examples of predictions with the highest losses. To improve accuracy, hyperparameters including learning rate, batch size, the number of layers, and the number of epochs should be fine-tuned. In addition, the dataset may need augmentation to equalize the distribution of images among the classes, particularly for Slippers.

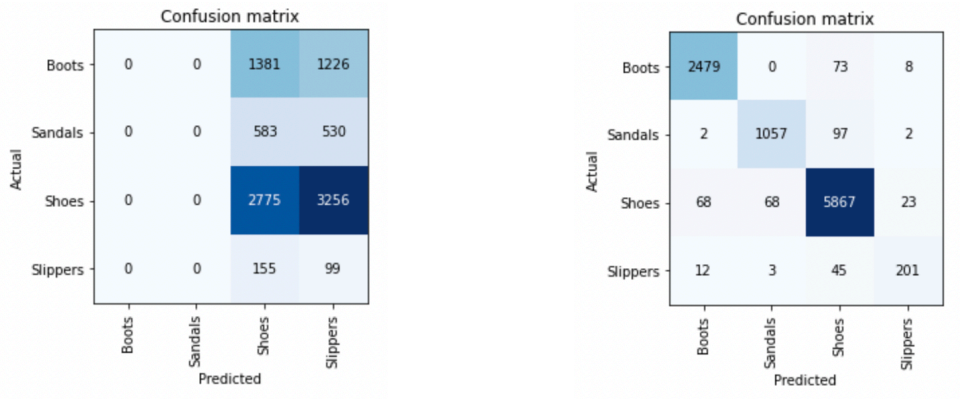

Figure 6: Untrained (left) vs. trained (right) model confusion matrices

Figure 7: Example of predictions with top losses (Prediction/Actual/Loss/Probability)

References

[1] "Retail Trade, December 2020," 19 February 2021. [Online]. Available: https://www150.statcan.gc.ca/n1/daily-quotidien/210219/dq210219a-eng.htm.

[2] P. Heaven, "Posthaste: Crushed by debt and hamstrung by COVID, small business owners watch their retirement go up in smoke," Financial Post, 18 March 2021. [Online]. Available: https://financialpost.com/executive/executive-summary/posthaste-crushed-by-debt-and-hamstrung-by-covid-small-business-owners-are-watching-their-retirement-go-up-in-smoke.

[3] "AI spending by retailers to reach $12 billion," 29 April 2019. [Online]. Available: https://www.juniperresearch.com/press/ai-spending-by-retailers-reach-12-billion-2023.

[4] B.-F. Wu, W.-J. Tseng, Y.-S. Chen, S.-J. Yao and P.-J. Chang, "An intelligent self-checkout system for smart retail," in *2016 International Conference on System Science and Engineering (ICSSE)*, Puli, Taiwan, 2016.

[5] R. Moorthy, S. Behera, S. Verma, S. Bhargave and P. Ramanathan, "Applying Image Processing for Detecting On-Shelf Availability and Product Positioning in Retail Stores," in *Proceedings of the 3rd International Symposium on Women in Computing and Informatics*, Kochi, India, 2015.

[6] D. López-de-Ipiña, T. Lorido and U. López, "Indoor Navigation and Product Recognition for Blind People Assisted Shopping," in *Proceedings of the 2011 International Workshop on Ambient Assisted Living,*, Torremolinos, Spain, 2011.

[7] W. Wang, Y. Cui, G. Li, C. Jiang and S. Deng, "A self-attention-based destruction and construction learning finegrained image classification method for retail product recognition," *Neural Computing and Applications,* vol. 32, no. 18, pp. 14613-14622, 2020.

[8] Y. Wei, S. Tran, S. Xu, B. Kang and M. Springer, "Deep Learning for Retail Product Recognition: Challenges and Techniques," *Computational Intelligence and Neuroscience,* vol. 2020, pp. 8875910-8875910, 2020.

[9] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of the International Conference on Computer Vision*, 1999.

[10] H. Bay, T. Tuytelaars and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision - ECCV 2006*, 2006.

[11] Leutenegger S, Chli M, Siegwart RY. “BRISK: Binary Robust invariant scalable keypoints,” In *2011 International Conference on Computer Vision*. IEEE; 2011:2548-2555, 2011.

[12] J. Shi-jie, K. Xiang-wei and M. Hong, "Automatic Product Image Classification with Multiple Support Vector Machine Classifiers," *Shanghai jiao tong da xue xue bao,* vol. 16, no. 4, pp. 391-394, 2011.

[13] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, p. 2278–2324, 1998.

[14] M. M. Srivastava, "Bag of Tricks for Retail Product Image Classification," *ParallelDots, Inc.,* pp. 1-4, 2020.

[15] K. Mavropalias, “Understanding Fastai's fit_one_cycle method,” *IconOf.com*, 19-Feb-2019. [Online]. Available: https://iconof.com/1cycle-learning-rate-policy/. [Accessed: 25-Apr-2021].