MSURU: Large Scale E-commerce Image Classification With Weakly Supervised Search Data

Yina Tang
Facebook Inc.

Fedor Borisyuk
Facebook Inc.

Siddarth Malreddy
Facebook Inc.

Yixuan Li
Facebook Inc.

Yiqun Liu
Facebook Inc.

Sergey Kirshner
Facebook Inc.

ABSTRACT

In this paper we present a deployed image recognition system used in a large scale commerce search engine, which we call *MSURU*. It is designed to process product images uploaded daily to Facebook Marketplace. Social commerce is a growing area within Facebook and understanding visual representations of product content is important for search and recommendation applications on Marketplace. In this paper, we present techniques we used to develop efficient large-scale image classifiers using weakly supervised search log data. We perform extensive evaluation of presented techniques, explain practical experience of developing large-scale classification systems and discuss challenges we faced. Our system, *MSURU* out-performed current state of the art system developed at Facebook [23] by 16% in e-commerce domain. *MSURU* is deployed to production with significant improvements in search success rate and active interactions on Facebook Marketplace.

CCS CONCEPTS

• **Computing methodologies** → **Image representations**; • **Information systems** → *Online shopping*; *Image search*.

KEYWORDS

Image classification, e-commerce image understanding

ACM Reference Format:
Yina Tang, Fedor Borisyuk, Siddarth Malreddy, Yixuan Li, Yiqun Liu, and Sergey Kirshner. 2019. MSURU: Large Scale E-commerce Image Classification With Weakly Supervised Search Data. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330696

1 INTRODUCTION

Facebook Marketplace[1] is a social shopping platform enabling hundreds of millions of people to buy and sell items within their community and businesses. The products for sale are posted by the sellers as listings consisting of the title, price, category (e.g., men's clothing and shoes), images of the product, the seller's city, and sometimes additional description and attributes like size, brand, or material. (See Figure 1 Center for an example.) Buyers often search for products as a part of their shopping discovery, and they expect the shopping search engine to return all relevant listings. Ranking relevant search results is a difficult problem given the inherent ambiguity of what the searcher finds relevant in the context of the query. Also, there can be a large number of listings in the buyer's area, many of them with a lifespan of only days if not hours. The problem is further compounded by the fact that unlike the traditional e-commerce setting in which listings are linked to a catalog with an abundance of information about the product, we often only have the title, image, price, category, and the location.

As a single ranking function for relevance is usually expensive to evaluate, a common approach is to break up ranking into separate retrieval and ranking stages, first determining *whether* the candidate is relevant by a computationally cheap matching of the query/context to the index terms extracted offline from the candidates, and then deciding *how* relevant it is by scoring the listings only from a relatively small set of candidates which passed the retrieval stage. A simple retrieval system would only consider the listings with terms matching those of the query. While this approach would work relatively well when the information about the product is populated with relevant terms, it is often not the case in our setting as the listings commonly contain only basic information. For example, in purely textual retrieval, a query like *shoes for men* (Figure 1 Left) would not match the listing with the only text J&M as its title[2] (Figure 1 Center) even though based on the picture of the product, it is clearly of men's shoes. In these cases, matching the query to the information extracted from the listing's images is crucial to improving the recall of the potentially relevant products.

In this paper, we present a deployed image understanding system called *MSURU* which generates terms for the images in Facebook Marketplace listings and therefore enables Facebook Marketplace Search retrieval by using image concepts rather than just text. (See Figure 1 Right for an illustration.) To do this, *MSURU* learns a mapping between an image and a set of queries. Training examples are extracted from search queries and the images in the search result listings determined by user interactions from hundreds of millions of Marketplace Search sessions, truly leveraging Facebook's scale. This does not require additional curation by human raters. *MSURU* outperforms the state of the art image classification system developed at Facebook by 16%. Furthermore, its deployment led to more than 6% search success rate increase and more than 1% increase in user interactions in Facebook Marketplace Search.

The rest of the paper is organized as follows: §2 provides an overview of the related work. §3 describes our approach for building

[1] http://www.facebook.com/marketplace

[2] J&M is a clothing apparel brand. https://www.johnstonmurphy.com

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330696

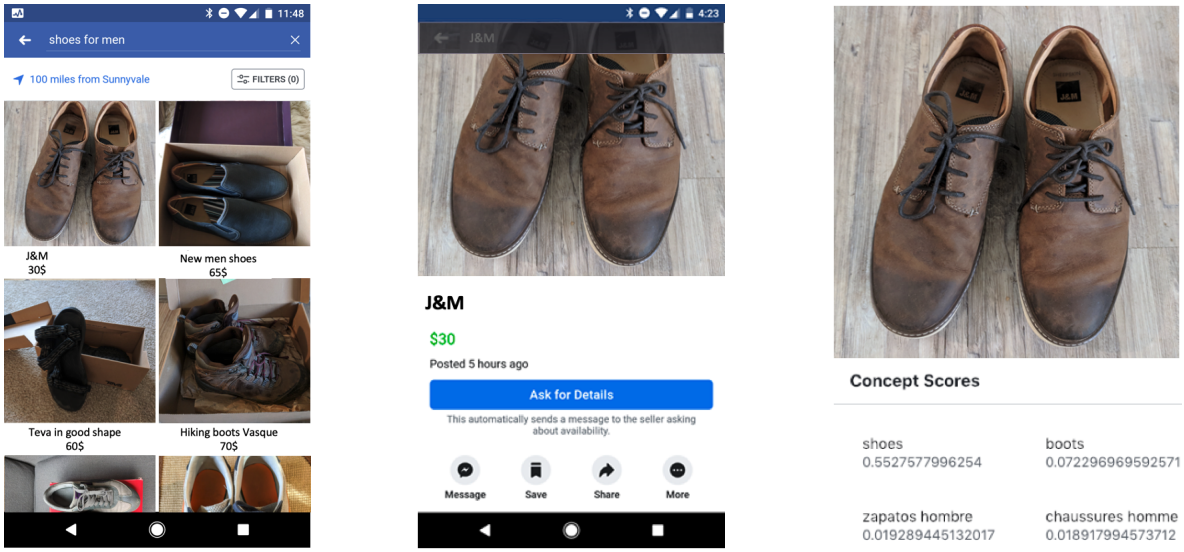

Figure 1: Left: A screenshot of a Facebook Marketplace Search results page. Center: A product detail page. Right: An example of classification applied to one of the products using *MSURU* produces set of likely queries.

large scale image classifier for Marketplace, and §4 describes the design and architecture of *MSURU*. §5 describes the experimental evaluation and the lessons learned during development. Finally, §6 describes the deployment and §7 concludes the paper.

2 RELATED WORK

Recent advances in using convolutional neural networks (CNN) for learning visual representations have led to improved precision of search ranking results. For example, Etsy [22] used a pretrained CNN to extract visual embeddings for product images, which improved the precision of their search ranking. Much of work in image understanding depend on the availability of human annotated data such as ImageNet [12]. The dataset curation process usually involves (1) identifying a set of predefined categories and (2) having human rater annotate through millions of images until enough positive examples are collected for each category. The training schema using such data is also known as *fully supervised* visual learning. Despite the effectiveness, there are several drawbacks of using such approach for large-scale applications. For example:

- Usually in such human rated datasets, categories rarely have common intersection with real visual distribution in Web applications such as Marketplace.
- Data manually collected by human raters has significant cost to collect and maintain.
- Human rater data does not scale to the growing needs and evolving nature of Marketplace due to its small scale and slow adaptation (there are a lot more categories in Marketplace than any existing image classification hierarchy).

Given the limitations, it becomes desirable to have scalable approaches to data collection and classifier training that would adapt to application needs. One apparent approach is to use *web scraped, noisy data* from Flikr [33] or search engines for image classification training. There have been efforts to train image classifiers using web noisy data [4, 8, 9, 13, 20, 26, 27, 30, 35, 41]. However, such data is often limited by the quality of the existing ranking systems of scraped web engines, and therefore classifiers trained on these data cannot provide higher quality than the given system.

On the other hand there are several cases from industry that demonstrated the power of using *large-scale, weakly supervised* image datasets for effective visual learning. For example, prior study [19] trained convolutional networks to predict words on a collection of 100 million Flickr photos. Sun *et al.* [31] train convolutional networks on the JFT-300M dataset of 300 million weakly supervised images. Within the search system of Pinterest in [39], the authors use millions of images with 20k classes to train image classifiers using user annotated data from pin-boards. Recently Zhai *et al.* proposed an extension of the approach of [39] in [40] by using a softmax classification approach to deep metric learning to produce high quality binary embeddings. In another work, Mahajan *et al.* [23] trained convolutional networks to predict hashtags on billions of Instagram images, achieving state-of-art performance on many computer vision tasks. Authors of [23] confirmed that bigger datasets help to reduce the influence of noise that comes in the dataset, where even with 25% of noise, the accuracy of the classifier decreased only by 2%. Therefore these results suggest that label noise may be a limited issue if neural networks are trained on sufficiently large datasets. Several other works have trained models on web-scale image data for other purposes, such as face recognition [32] and similarity search [18].

In the search domain, using weakly supervised learning can be challenging due to the uniqueness of search data. Different from [16, 29, 37], where the authors trained image embeddings to power visual search, in Facebook Marketplace Search the query data is primarily text based. In the search bar, people issue queries, where each query can be considered as a question, and then they find answer for the query by interacting with search results. This engagement is proactive (initiated by the searcher) and voluntary (searcher chooses whether to engage with the search result or not), which makes the connection very strong. This provides us

a great opportunity to use search data to power product image understanding. In this paper we investigated the power of search data for image classification in the commerce domain. To the best of our knowledge, we are the first to report results on using large-scale, weakly supervised e-commerce web search data with text queries to train convolutional neural networks.

Due to the increased label space for search data and possible duplication of training labels, in this paper we also investigate using sampled softmax approaches. Some of the efforts on sampled softmax include [2, 3] and are targeted to accelerate the training of the image classifiers. We proposed data collection and training techniques, which helped us improve image embedding performance by 16% in e-commerce domain compared to state of the art generic image classifier in [23].

3 APPROACH

On Facebook Marketplace, users are categorized into two types - seller and buyers. Sellers post listings of products they want to sell, and buyers search for and interact with products they want to buy. We use the public information provided by both types of users to prepare a weakly labeled dataset which we then use to train our models. Since this information is provided by millions of users on Facebook Marketplace, we can construct datasets with millions of images within a short time window. Using this mechanism, we collected millions of data points with thousands of labels for our training.

We frame our problem as a multi-class classification task and fine-tune a pre-trained convolutional neural network (CNN) based on ResNeXt [36] described in detail in section §3.3. Some Facebook Marketplace images are high-quality photographs, most product images are taken by less powerful cameras. We trained *MSURU* image classifier using millions of images in the e-commerce domain using search query data to capture data distribution of Facebook Marketplace. This resulted in gains in offline measurements reported in §5 as well as online search success rate increase reported in §6. Our production *MSURU* image classification model is trained on a sample of $10k$ frequent search queries and $40M$ user interacted product images. A product image is assumed to be relevant to a query if user interactions happened, and the search query then becomes its label. We provide more details on the data collection and preparation in the following sections §3.1 and §3.2.

3.1 Data Collection

As part of the sale process on Marketplace, the seller posts a listing for an item and provides a title and description to the listing, among other things like price, location, etc. Images are also attached to the listing, and the seller can choose one image as the primary image for the item, which becomes the cover photo of the listing.

Buyers can use the search bar to look for products using Marketplace Search. Given the buyer's search query, the Marketplace search engine returns relevant products from the inventory and displays them to buyers on the landing page as shown in Figure 1. The interface of the landing page displays several details including the title, price and primary image of each listing. This is the only information available to the buyer about the products on that page. The buyer can choose to interact with the search results in a number of ways. A *click* will result in a product details page with more information about the item, where the buyer can view the product description and browse through all the photos attached to the product item. On this page, the buyer can also choose to message the seller. We refer to this interaction as *contact*. After the message is sent, both parties can choose to further connect with each other on the sale details. We define the additional interactions, which are equivalent of *add-to-cart* on e-commerce websites, as *in-conversation*. This information is logged based on occurrence without any reference to the message content. If none of these actions happen on the search results page, we refer to that interaction as *no-click*. So each search session might result in a number of *no-click*s, *click*s, *contact*s and *in-conversation*s. These interactions are recorded in search logs and we use them to obtain training data for our models.

User interactions serve as a signal linking the query to the image. If the buyer *click*ed on the product, it is a weak signal that the product is related to the query, because buyers knowingly or unknowingly click on both relevant and irrelevant products as a way to explore the inventory. As a result, the total number of *click*s on products is much greater than other interactions. This makes the click data a very noisy signal for relevance between the image and the query. We observed that when a buyer *contact*s the seller it usually indicates stronger query-product relevance. When the buyer performs the *in-conversation* interaction, we get the strongest signal for query-product relevance. *In-conversation* interaction is even more sparse compared to *contact* interaction. These three interactions provide positive signals about query-product relevance. On the other hand, there might be *no click*s on the search result for reasons other than relevance, such as high price or long distance. Therefore we don't use *no-click* products as negative examples for the query during training.

3.2 Data Preparation

We have used search log data from Facebook's Marketplace Search to observe effectiveness of proposed image classification technology in this paper. Facebook's Marketplace Search log data is anonymized, has a specified retention time, and is deleted after retention time expires. Let's call the event of user query at specific time as *search query session*. The logged data contains timestamp, session identifier, textual query, product identifier, and how the product was engaged (*click, contact, no-click, in-conversation*). In the remainder of this section we use *click* as an example interaction, but the same steps were followed to get data from all of the interactions.

From the search logs, we extract the tuple (query, product id) for each product that was clicked for that particular query. Since a particular product might have been clicked for more than one query, we frame the problem as a multi-class classification problem. So each product will be linked to one or more queries. We end up with tuples of the form (list of queries, product id) where the product ids in the set are unique. As detailed above, each product may contain multiple images. Since we are interested in training a classifier on images, we can either choose to use the primary image, a subset of all the images, or the entire set of images as training data. In our experiments, we choose to use the entire set of images as training data in order to maximize the variety of images available for training. We assign the same list of queries attached to the product to all of the images attached to the product.

We treat each distinct query as one of the entries in the label set. Our dataset has a long tail distribution with the vast majority of labels having very few occurrences in the dataset. In order to stabilize training and decrease the dimensionality of the label set, we remove all labels which have fewer than K occurrences in the dataset. We vary K in our experiments as detailed in section §5.6.

Image deduplication: When we split the dataset into training and testing data, we need to make sure that the same image is not present in both. Simply removing duplicated images by their unique ids is not very effective because same image might be uploaded multiple times by the same seller and sellers often use a downloaded stock photo of the product. To address this problem, we first compute a compressed embedding vector extracted from the last feature layer of ResNet trained on ImageNet data for all images, and then remove duplicated images that have the same compressed embedding.

In the following section we go into more detail about the structure of the *MSURU* image classification model.

3.3 Image Classification Model

Model Architecture: We use residual networks with grouped convolutional layers, called ResNeXt [36]. Our experiments use ResNeXt-101 32×4d, which has 101 layers, 32 groups, and group widths C=4. In our experiments, we initialize the weight parameters using the convolutional network from [23] – which was pre-trained on 3.5B Instagram images over 17k hashtags – and then fine-tune on our Marketplace search query data.

Training Details: Our models are trained by synchronous stochastic gradient descent (SGD) on 160 GPUs across 20 machines with mini-batches of 5,120 images. Each GPU processes 32 images at a time. For SGD, we use Nesterov momentum [25] of 0.9 and weight decay of 0.0001. Weights of the final fully connected layer are sampled from a zero-mean Gaussian with standard deviation 0.01. We adopt the cosine learning rate as in [21], with initial learning rate 0.0005625 per GPU. We train the model for a total of 48 epochs, with epoch size 6,759,000. Similar to [23], we measure the training length by the total number of images processed, which is 324M. We use standard image rescaling and data augmentation as described in [15]. These settings are consistent across fine-tuning experiments. We used the Caffe2 framework [7] to train our *MSURU* image classifier.

3.4 Randomized Softmax

We have a multi-class and multi-label problem where ith training example consists of a small set of target classes T_i out of a large universe $|D|$ of possible classes. One of the main difficulties in training is that the computational complexity and memory requirement involved in computing the target class probability and loss function grow linearly with respect to $|D|$, the number of possible classes. To address this issue, instead of considering all $|D|$ possible classes in the forward pass, we only consider $sampling\_ratio * |D|$ classes. Specifically, for the ith training example, only the target classes T_i and a small subset S_i of randomly sampled negative classes are used for optimization, i.e. $C_i := T_i \cup S_i$ where $|C_i| = sampling\ ratio * |D|$. The proposed approach is based on earlier work [17, 40].

We use *Cross Entropy Loss* as loss function. Let x_j^i and y_j^i denote the logit and label of the jth class for the ith training example, then the loss is

$$loss_i = -\sum_j^{T_i} log \frac{e^{x_j^i}}{\sum_k^{C_i} e^{x_k^i}} = -\sum_j^{T_i} log S_j^i$$

where $S_j^i := \frac{e^{x_j^i}}{\sum_k^{C_i} e^{x_k^i}}$ is the **Randomized Softmax** for jth class, a *scaled* version of the *full softmax*. It is not hard to show that the gradient of $loss_i\ w.r.t\ x_j^i$ is

$$\frac{\partial loss_i}{\partial x_j^i} = \begin{cases} S_j^i \cdot |T_i| - y_j^i & j \in C_i \\ 0 & j \notin C_i \end{cases} \quad (1)$$

which can be interpreted as "dropping out" the logits that are not in C_i.

4 SYSTEM ARCHITECTURE

We now describe the system architecture of *MSURU*, Facebook's large-scale image classification system for Marketplace. *MSURU* is deployed in production and is designed to operate on product images uploaded daily at Facebook scale in a realtime fashion. *MSURU* is deployed and powers image understanding for Marketplace Search at Facebook. Figure 2 outlines the high-level architecture of *MSURU* with three subsystems of Offline modeling, Image classification service and Marketplace Search application.

Users of Marketplace Search enter a search query to receive a set of search results, denoted as step 1 on the Figure 2. Retrieval and ranking of search results happens within the Unicorn search system [11], and user interactions with the search results are recorded into the search log. Historical log data may include query and search results, also referred as product documents, that the user interacted with. The user interactions provide implicit feedback about the relevance of the search results, and provide an abundant source of data to train *MSURU* image classification models.

Search log data is aggregated in Presto [34] distributed SQL query engine for big data, and then consumed by *MSURU* offline modeling subsystem (step 6 in Figure 2). We select a subset of queries from the search log that will be used for model training and select the images associated with user interactions to train on. Then the model is trained using FBLearner Flow, Facebook's AI backbone [14], using distributed set of machines with GPU. We perform evaluation of the model using a set of automated offline metrics and human rater evaluation, and deploy the model in production into Visual models storage (step 9 in Figure 2) if the model meets decided quality bar. Once the MSURU model is deployed it can be applied to newly uploaded images of Marketplace products.

When sellers on Facebook Marketplace create a new product item, the *MSURU* image classification system automatically schedules the processing of product images. *MSURU* image classification service utilizes a pull-based mechanism where an image uploaded by a client application (step 11 in Figure 2) is added to a distributed processing queue. The inference machines in *MSURU* pull the enqueued jobs when resources are available and process them asynchronously (step 12 and 13 in Figure 2). On the inference machines the image is resized to 400px in the larger dimension and fed through the classification model which outputs classification scores for predicted queries, as well as image embeddings. Consuming processes can register callbacks when enqueuing jobs, which *MSURU*

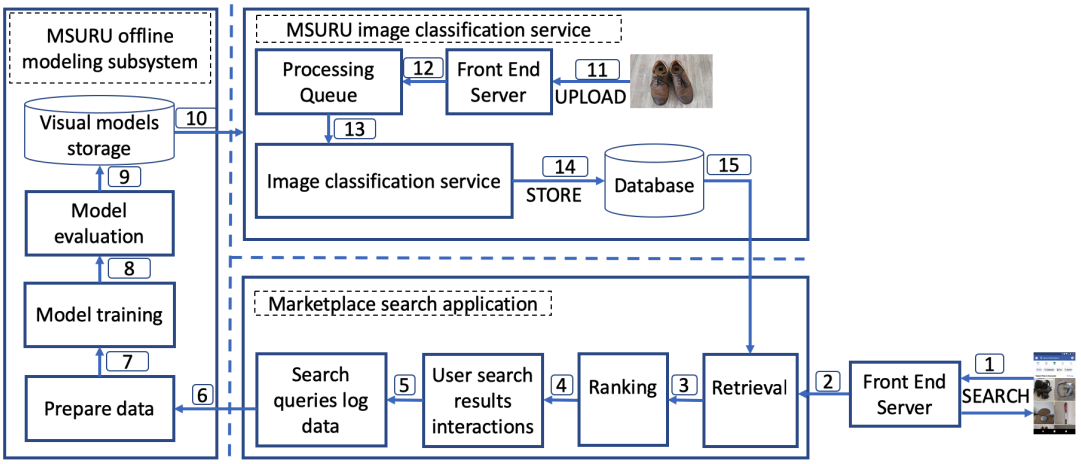

Figure 2: Architecture of *MSURU*, Facebook's scalable image classification system.

invokes right after each job is processed for immediate usage of results by downstream applications. The processing queue is optimized for high throughput and availability, and utilizes RocksDB [5] for persistence. This pull-based asynchronous architecture provides various benefits including better load-balancing, rate-limiting in scenarios of unexpected spikes in requests (for example, surges in product uploads on Facebook), and the ability to optimize the system for throughput.

Within the image classification system we run a set of models, where *MSURU* is applied on top of the generic image classifier model. We chose to fine-tune several ResNeXt blocks of the generic image classification model to save on computation capacity. At runtime, the image is passed through a set of convolutional blocks of generic image classification model, and then through a few ResNeXt block of *MSURU*, the Marketplace specific image classifier.

Results from the image classification are stored in a distributed database and include predicted queries with confidence scores and MSURU convolutional image embedding. We have applied quantization to the embedding produced by *MSURU* described in detail in [28] and compress the embedding into a binary vector to save storage space. From the database, image classification results are automatically downloaded by the search system and can be consumed within seconds after the listing is created.

In section §5, we show various experiments that helped us to maximize the quality of the *MSURU* image classification.

5 EXPERIMENTS

In order to build a high quality image classifcation model out of the data we collected based on different user interaction signals, there are a lot of questions that need to be answered. For training data, ideally we want to build a dataset that covers all existing unique products, with enough images for each product to train the model, and with all ground-truth images. However, search queries have a long tail distribution, have visually overlapping product categories (such as *bikes* vs *road bikes*), and user actions can be noisy. For model architecture, ideally we want to fine-tune the entire model to better transfer to new domains, but we are constrained by the computational cost, so we can only fine-tune the last few feature extraction blocks of the model.

In this section, we present an extensive study of building the best *MSURU* model, examining this from a few different aspects. First we define the metrics used to determine the model quality, then we describe the datasets used for training and evaluation, followed by our experiment results. We started from training on datasets with different label quality in §5.3. To understand the effect of the data distribution, we experimented with square-root data sampling in §5.4. To understand the limit of model convergence w.r.t the number of labels, we experimented with clustered and duplicated query labels in §5.5. In §5.6, we investigated the possibility of training on larger label space with high quality data. To speed up the training process and for better generalization, we experimented with randomized softmax in §5.7. In §5.8, we investigated how much improvement we can achieve by fine-tuning more feature extraction blocks.

After the initial parameter search we finalized our model training details as described in section 3.3. All models reported here are trained with the same parameters unless stated otherwise. We follow the standard practice of training and evaluating our models on separate holdout datasets.

5.1 Metrics

We use two types of offline metrics to evaluate *MSURU* model for classification and embedding quality respectively.

Model classification quality: we use mean average precision (mAP) and ranking average precision (rankingAP) as offline evaluation metrics during model training process. mAP is a general indicator of overall classification accuracy (the higher the better). RankingAP indicates the quality of top predicted labels, which is important to us as we can only save limited number of labels in production (also the higher the better).

Image embedding quality: we use the results of the k-nearest-neighbor (kNN) retrieval and calculate the mean similarity (mRel) score for top k results based on the human rated relevance between the query and neighbor images. A higher mRel score indicates closer proximity of similar photos in the embedding space, thus higher quality of embedding. The human rated relevance is based on a 0 to 4 scale, as explained in Table 1. We also compute precision@k (P@k) based on the kNN retrieval results, which is defined as the average

percentage of neighbor images in the top k results that share at least one class label with the query image. This offline evaluation metric is commonly used together with human evaluation [24, 39]. Similar to [40][3], we compute coverage to be 1 if there is at least one neighbor image in the top k results that shares at least one class label with the query image, 0 otherwise, then we average the scores for all query images to get coverage@k (C@k). Computing P@10 and C@10 scores do not require human labeling, but depends on the label quality of the query image, which could be noisy itself. Mean relevance score is unbiased and reflects visual and semantic similarity between photos, but can be affected by the quality of human rater evaluation.

Table 1: Human rated relevance scale between two photos

Score	Relevance	Explanation
4	Vital	Exactly the same photos, or one photo is a crop of the other at the pixel level
3	Primary	Semantically and visually similar
2	Reasonable	Semantically the same, but not visually similar
1	Somewhat	Semantically similar to some extent, and not visually similar
0	Off-topic	Completely not related to each other

5.2 Training and Test Datasets

Marketplace Search data is collected in the following way: for each user-created "for sale" post on Marketplace we use the search queries that resulted in one of the user actions A ∈ (*click, contact, in-conversation*) on the Marketplace post as the labels. We then attach these labels to all photos in the "for sale" post on Marketplace. Note that A_i implies A_{i-1}, and the volume of A_i compared to A_{i-1} also decreases as it goes towards final sale. This allows us to collect different weakly-supervised datasets with different levels of quality. While there are orders of magnitude less *in-conversation* product photos than *click*ed product photos, the search queries associated with *in-conversation* photos are much more likely to be true labels and thus result in a cleaner dataset.

For our experiments we created a set of different datasets for training. They are denoted as *interaction-dataset size-label size*, e.g. ICVS-47M-10k means the photos are *in-conversation* product photos with 47M photos across 10k labels. Since the label space between datasets can be different, mAP, rankingAP are reported on separate test sets where applicable. For k-Nearest Neighbor retrieval, we use the same candidate pool of 6.7M photos. We compute the 5 nearest neighbor photos from this pool for 200 randomly selected source photos, and compute the mRel metric based on the corresponding 1000 rating scores following the scale in Table 1. Although the highest possible score is 4, because we heavily de-duplicate the dataset with the compressed ResNet image embedding the highest possible relevance is 3 for most query-neighbor image pairs, as the probability of an existing cropped photo is very low. To compute P@k and C@k based on nearest neighbors, we sampled 6500 query images from the candidate pool, computed 10 neighbors per query image and aggregated the results.

[3]In [40], the same metric is called **recall** instead of **coverage** in this paper.

5.3 Label quality

In this experiment, we compare the models trained on data filtered by different user interactions, while keeping the minimum number of training photos the same for each data source. As described in previous sections we believe that search data collected based on *in-conversation* logs are of higher quality than those collected from contact or click engagement actions. Since the latter two have much higher data volume, if we use the same minimum number of photos as the cutoff for valid labels we can get a much larger label space. We also experimented with curriculum training [1]. We fine-tuned the pre-trained model on these three datasets one by one, in two directions, one in increasing noise level and the other vice versa.

From the results in Table 2, we can see that as the classifier shifted from a general image domain to commerce, the image embedding quality improved a lot. Between the model trained on the *ICVS-47M-10k* dataset and the pre-trained baseline model from [23], although the label space reduced from 17k to 10k, the mRel score increased from 1.319 to 1.528. Therefore, our *MSURU* classifier out-performed current state of the art system developed at Facebook [23] by 16% in commerce domain.

From Table 2, we can also see that as we change the user interaction type for datasets *Contact-117M-27k* and *Click-217M-56k*, as label space increases the embedding quality is improved. This is reflected in both P@10 and C@10. For results from curriculum training from Table 3, the model trained on three datasets in the *Click->Contact->ICVS* direction improved P@10 by 5%, compared to the model trained on the *ICVS-47M-10k* dataset. We hypothesize that *Click->Contact->ICVS* is better because of combination of curriculum training on larger datasets and higher quality of *ICVS* dataset or fine-tuning helps to improve embedding quality. On the other hand, the mRel score does not directly correlate with P@10 and C@10. We believe that it is more affected by the dataset label quality, since using *Click* dataset alone or part of curriculum training both impaired mRel (making mRel lower in both cases).

Based on these findings, we mainly focused on *in-conversation* datasets for our subsequent experiments .

Table 2: Model performance comparison on datasets of different quality, collected from *Click, Contact* and *in-conversation* product photos. A threshold of 1k photos is applied to all labels.

Fine-tune Dataset	P@10(%)	C@10(%)	mRel
(baseline model [23])	10.68	38.96	1.319
ICVS-47M-10k	14.04	46.67	1.528
Contact-117M-27k	14.18	46.66	1.532
Click-217M-56k	14.34	46.55	1.513

5.4 Data Resampling

The query frequencies naturally follow a Zipfian distribution. For example, the most frequent query (*iphone*) in our *ICVS-10M-10k* dataset appears more than three thousand times as often as the least frequent query (*alpina alloy wheels*) in the same dataset. Training

Table 3: Model performance comparison on datasets of different quality using curriculum training in two directions. A threshold of 1k photos is applied to all labels.

Fine-tune Dataset	P@10(%)	C@10(%)	mRel
ICVS→Contact→Click	14.46	46.54	1.487
Click→Contact→ICVS	14.83	47.81	1.490

on the natural distribution may not be optimal and therefore we consider an alternative way to sample: *square root sampling*. This sampling strategy has also been used for the training on Instagram images, as in [23]. To perform square root resampling, we compute a replication factor $r(q)$ for each query q: $r(q) = \max(1, \phi(t/f(q)))$, where $\phi(x) = \sqrt{x}$, f(q) denotes the number of examples for a search query in our dataset, and t is a hyper parameter that denotes the threshold below which replication happens for the query. Given an image I with (possibly multiple) queries $\{q_i\}$ associated, the image-level replication factor for I is computed as $r(I) = \max_i r(q_i)$. For a set of n unique images, a list of training images is constructed by computing the replication factor for each image, duplicating the image the prescribed number of times, and then randomly permuting the list.[4] The threshold t is selected such that the final list has a target length matching the desired training schedule length. In our case, t is chosen so that the overall training data size is roughly doubled.

In Table 4, we compare the performance using the original *ICVS-10M-10k* data vs. resampled dataset. We report the classification metrics on the original test set. As shown in Table 4, resampling effectively improves the mAP by 0.5%, albeit decreasing the rankingAP from 48.10% to 43.04%. For the embedding quality, both P@10 and C@10 improved by 1.4%, while mRel is about the same. Overall we believe that resampling helps to improve the quality of embedding due to better representation of less frequent classes.

Table 4: Fine-tuning performance comparison before and after square root resampling. Metrics are described in §5.1.

Dataset (ICVS)	mAP (%)	rank AP(%)	P@ 10(%)	C@ 10(%)	mRel
10M-10k	21.68	48.10	12.26	42.89	1.383
resampled	22.15	43.04	12.44	43.52	1.373

5.5 Clustered and duplicated query labels

To investigate the impact on model quality with different number of labels, we use query clustering to decrease the number of labels in the *ICVS-10M-10k* dataset and label duplication to increase the number of labels in the dataset. By clustering queries we can potentially merge queries of identical meaning and also reduce the granularity of product categorization, while by duplication we can investigate the limit of duplication within which the network can still learn a good representation.

Since the queries come in different languages, we utilize the multi-lingual embedding [10], which is computed for all Facebook search queries. The multi-lingual embedding is trained so that similar concepts in different languages are close to each other in the embedding space. We apply K-means clustering on these multi-lingual embeddings to form C clusters. We use different values of $C \in (642, 1.3k, 2.6k, 5k)$ as shown in Table 5.

To increase the number of labels, we split each label into K labels. For example, for the case when $K = 2$, say we have image I_0 with label *chair* and image I_1 also with same label *chair*. We split the labels so that image I_0 has label $chair_0$ and image I_1 has label $chair_1$. So the number of data points remains the same while the label space is doubled. This way we amplify the label duplication we observed in the original queries. We use different values of $K \in (2, 4, 8)$ as shown in Table 5.

Since the label spaces are different in this experiment, we compare P@10, C@10 and mRel scores from kNN retrieval between different models. From Table 5, we can see that decreasing the number of labels by clustering decreases the quality of the embeddings, reflected in all three metrics. This proves that using the original queries as labels is better than query clustering.

On the other hand, increasing the number of labels by duplication doesn't have such a clear trend. Even a $4x$ increase in the amount of duplicate labels gives a model which performs at par with the baseline model. Therefore, we conclude that label duplication does not decrease image embedding quality.

Table 5: kNN performance comparison between models trained with varied label spaces but the same images, with both clustered and duplicated query labels. Metrics are described in §5.1.

Dataset (ICVS)	Technique	P@10 (%)	C@10 (%)	mRel
ICVS-10M	Clustering C=642	11.66	41.77	1.348
ICVS-10M	Clustering C=1.3k	11.79	41.48	1.365
ICVS-10M	Clustering C=2.6k	11.80	42.13	1.368
ICVS-10M	Clustering C=5k	11.89	42.23	1.370
ICVS-10M	**Baseline**	12.40	43.77	1.383
ICVS-10M	Duplication K=2	12.23	43.22	1.391
ICVS-10M	Duplication K=4	12.91	43.75	1.378
ICVS-10M	Duplication K=8	13.13	44.62	1.384

5.6 Expand coverage of search queries

In this experiment, we compare the models trained on *in-conversation* datasets, but increased the label space to incorporate search queries with fewer photos. This is to explore the limit of the label space where we can still achieve similar embedding quality given the cleanest data we have.

From the results in Table 6, we can see that as the label space becomes larger we observe increase in P@10 and C@10, indicating better embedding quality. This also means the network can generalize the pre-trained features to fine-tuning datasets of larger label space. In addition, the mean relevance score (mRel) tends to increase with more training data and labels but somewhat insignificantly. This might be caused by the fact that the extra 37k labels have much less average photos per label compared to the 10k labels sampled from most popular queries.

[4]When an image with multiple queries is replicated $r(I)$ times, each individual query q_i is removed as needed such that q_i is only replicated $r(q_i)$ times.

Table 6: Model performance comparison with training data collected from *in-conversation* photos, but varied label space given different minimum photo threshold for each query. Metrics are described in §5.1.

Dataset (ICVS)	Threshold	P@10 (%)	C@10 (%)	mRel
36M-10k	1000	12.50	43.23	1.354
40M-20k	500	12.47	43.99	1.368
44M-47k	200	12.75	44.35	1.372

5.7 Random Softmax for Large Label Space and Duplicated Labels

We apply the *Random Softmax* methodology described in §3.4 to *ICVS-10M-10k* dataset (see §3.1), trained using model architecture discussed in §3.3. We present our findings in Table 7, showing that with *sampling ratio* 0.8, we achieve better results in all Precision@K, Coverage@K and mean relevance (human rated) metrics; Specifically, we reach Precision@10 +10.7%, Coverage@10 +5%, mean relevance is slightly higher than baseline, which is the full softmax model. Note that in practice, although random softmax required less time to train for each epoch, it takes more epochs to converge to the optimal model than full softmax model. One possible explanation is that in *Random Softmax*, we only update a subset of logit classes in each iteration, thus it takes more iterations for the model to converge.

We achieve significant improvement in model quality using *Random Softmax*. Our intuition for improvement of *Random Softmax* comes from the fact that it helps to reduce model overfitting by forcing model to learn robust representations of target classes by randomly dropping percentage of nodes in softmax layer. During our experimentation we tried to use original drop out (so it could also drop any softmax node including true labels), but it hurts accuracy of the classifier significantly because of not having the right learning true label targets. At runtime inference we keep all the softmax inputs.

Table 7: Precision@K, Coverage@K and mean relevance (humam rated) comparison on different random softmax sampling ratio trained on *ICVS-10M-10k* dataset. Number of epochs of training is reported along the sampling ratio.

Sampling Ratio / #Epochs	P@10 (%)	C@10 (%)	mRel
Full Softmax (B/L) / 10	12.40	43.77	1.383
0.9 / 15	13.81	46.32	1.334
0.8 / 20	13.73	46.01	1.395
0.7 / 20	13.91	46.03	1.353

5.8 Deeper model fine-tuning

In this experiment, we explored fine-tuning different numbers of feature extraction blocks in the ResNeXt-101 architecture on the *ICVS-10M-10k* dataset to get an idea of how much improvement we can achieve. While we expect the model performance to improve, it is also important to consider the computation cost. To learn the performance upper bound with this fine-tuning approach, we also fine-tuned all blocks in one variant.

From the results in Table 8, we can see that mAP improved as much as 23.6% between full fine-tuning and 1 block fine tuning, while mAP of 3 blocks fine-tuning can also improve over baseline by 16.9%. For kNN retrieval performance, P@10 and C@10 both improved by more than 20% for models with deeper fine-tuning compared to single block fine-tuning. However, the difference in mRel is much smaller – within 5% for full fine-tuning compared to single block fine-tuning. Although the optimal number of fine-tuning blocks may depend on the image data domain, for our use case fine-tuning 3 blocks is a good trade-off between model quality and deployment cost.

In the next section we describe the process of *MSURU* deployment.

Table 8: Model performance comparison between fine-tuning different numbers of feature extraction blocks in the ResNeXt-101 32×4d architecture on the ICVS-10M-10k dataset, using the model training setup described in §3.3. Metrics are described in §5.1.

Fine-tune blocks	mAP (%)	rank AP(%)	P@ 10(%)	C@ 10(%)	mRel
1	21.67	48.13	12.40	43.77	1.383
3	24.58	52.23	19.07	53.00	1.439
5	25.33	52.76	19.97	54.02	1.449
32	26.79	54.61	21.84	55.49	1.463

6 DEPLOYMENT

MSURU service is deployed within Facebook at scale, offers a cloud API for image classification from images attached to Marketplace products, and processes a large volume of product images uploaded to Facebook.

MSURU image classification system trained on the *ICVS-47M-10k* dataset was integrated into Marketplace Search within Facebook. We developed several retrieval and ranking features based on outputs of image classification including:

- Textual Query to Image relevance using Siamese networks [6], where the inputs to the model are the embedding vectors of the query and the image of the product result. The objective of training is to minimize classification loss. Query and image embedding representations are passed through separate deep neural network towers and processed by several layers to produce a binary signal, where a positive result denotes a match and negative one denotes a non-match described in more detail in [38]. Query to image relevance via Siamese networks helped us to increase user interactions by over 1% relative.
- Retrieval of products using inverted index match of textual query to image tags produced by *MSURU*. As result of classification *MSURU* system assigns set of possible queries, which can be used to refer the product. We kept the top 10 most confident classification assignments. Once *MSURU* query tags are assigned, textual query entered by user could be matched to *MSURU* tags during the retrieval within Marketplace Search. This technique helped us to increase search success rate by more than 6% relative and user interactions by over 1% relative.

To understand the impact of *MSURU* we did data analysis and share our findings here. In addition to increase in user interactions, ranking features using Siamese networks helped us to improve quality of search ranking measured by human evaluation by matching more relevant products to a given user query by over 9% relative. This is because Siamese network helped us to capture relationship between product images and queries, and helped down-rank products dissimilar to the query.

We observed that retrieval technique using *MSURU* were the most effective when products do not have a detailed description. *MSURU* classification tags for retrieval helped to associate the missing information back to the product. We were also able to return more search results for queries due to ability to tag the product correctly using image classification by *MSURU* by over 8% relative.

7 CONCLUSION

In this paper we presented approaches for building efficient models for large scale image classification for Marketplace. We presented techniques to train accurate large scale image classifiers with weakly supervised search query data. We performed thorough evaluation, demonstrated the efficiency of our system and shared our practical experience of implementation and usage of the proposed techniques. We deployed *MSURU* with significant gains in image embedding quality, outperforming state of the art system developed in Facebook by 16% in the commerce domain, and improved user search experience on Marketplace. In future work we aim to extend our learnings to other domains of applications in Facebook using search data.

8 ACKNOWLEDGMENTS

The authors would like to thank Cristina Scheau, Manohar Paluri, Hervé Jégou, Dhruv Mahajan, Dmitry Pleshkov, Jon Guerin, Kai Zhang, Ananth Subramaniam, Vignesh Ramanathan, Xin-Jing Wang, Rama Kovvuri, Polina Kuznetsova, Jun Mei, Priyanka Kukreja, Alex Marrs, Anirudh Sudarshan, Nikhil Garg, Hong Yan, Amit Nithianandan, Praveen Rokkam, Chunye Wang, Rena Patel, Rui Li, Tracey Yao, Taroon Mandhana and others who contributed, supported and collaborated with us during the development and deployment of our system.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning *(ICML)*.
[2] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19, 4 (2008), 713–722.
[3] Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick Training of Probabilistic Neural Nets by Importance Sampling.. In *AISTATS*. 1–9.
[4] Alessandro Bergamo and Lorenzo Torresani. 2010. Exploiting Weakly-labeled Web Images to Improve Object Classification: A Domain Adaptation Approach. In *NIPS*.
[5] Dhruba Borthakur. 2013. Under the Hood: Building and open-sourcing RocksDB. https://code.facebook.com/posts/666746063357648/under-the-hood-building-and-open-sourcing-rocksdb/.
[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. [n. d.]. Signature Verification Using a "Siamese" Time Delay Neural Network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS, 1993)*.
[7] Caffe2. 2017. https://caffe2.ai/.
[8] Xinlei Chen and Abhinav Gupta. 2015. Webly Supervised Learning of Convolutional Networks *(ICCV)*.
[9] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. 2013. NEIL: Extracting Visual Knowledge from Web Data *(ICCV)*.
[10] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word Translation Without Parallel Data. *CoRR* (2017).
[11] Michael Curtiss, Iain Becker, Tudor Bosman, Sergey Doroshenko, Lucian Grijincu, Tom Jackson, Sandhya Kunnatur, Soren Lassen, Philip Pronin, Sriram Sankar, Guanghao Shen, Gintaras Woss, Chao Yang, and Ning Zhang. 2013. Unicorn: A System for Searching the Social Graph. *VLDB* (2013).
[12] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*.
[13] Santosh Kumar Divvala, Ali Farhadi, and Carlos Guestrin. 2014. Learning Everything about Anything: Webly-Supervised Visual Concept Learning. *CVPR* (2014).
[14] Jeffrey Dunn. 2016. Introducing FBLearner Flow: Facebook?s AI backbone. hhttps://code.fb.com/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/
[15] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *CoRR* abs/1706.02677 (2017).
[16] Houdong Hu, Yan Wang, Linjun Yang, Pavel Komlev, Li Huang, Xi Stephen Chen, Jiapei Huang, Ye Wu, Meenaz Merchant, and Arun Sacheti. 2018. Web-scale responsive visual search at Bing. In *KDD*.
[17] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. *ACL* (2015).
[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734* (2017).
[19] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. 2016. Learning visual features from large weakly supervised data. In *ECCV*.
[20] Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. 2016. The Unreasonable Effectiveness of Noisy Data for Fine-Grained Recognition. In *ECCV*.
[21] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic Gradient Descent with Restarts. *CoRR* abs/1608.03983 (2016).
[22] Corey Lynch, Kamelia Aryafar, and Josh Attenberg. 2016. Images Don't Lie: Transferring Deep Visual Semantic Features to Large-Scale Multimodal Learning to Rank. In *KDD*. 541–548.
[23] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the Limits of Weakly Supervised Pretraining. In *ECCV, 2018*. 185–201.
[24] Henning Müller, Wolfgang Müller, David McG. Squire, Stéphane Marchand-Maillet, and Thierry Pun. 2001. Performance Evaluation in Content-based Image Retrieval: Overview and Proposals. *Pattern Recogn. Lett.* (2001).
[25] Y. Nesterov. 1983. A method of solving a convex programming problem with convergence rate O(1/k2). *In Soviet Mathematics Doklady* 27 (1983).
[26] Li Niu, Qingtao Tang, Ashok Veeraraghavan, and Ashutosh Sabharwal. 2018. Learning From Noisy Web Data With Category-Level Supervision. In *CVPR*.
[27] Scott E. Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training Deep Neural Networks on Noisy Labels with Bootstrapping. *CoRR* (2014).
[28] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2018. A neural network catalyzer for multi-dimensional similarity search. *CoRR* abs/1806.03198 (2018).
[29] Devashish Shankar, Sujay Narumanchi, HA Ananya, Pramod Kompalli, and Krishnendu Chaudhury. 2017. Deep learning based large scale visual recommendation and search for e-commerce. *arXiv preprint arXiv:1703.02344* (2017).
[30] Sainbayar Sukhbaatar and Rob Fergus. 2014. Learning from Noisy Labels with Deep Neural Networks. *CoRR* (2014).
[31] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*.
[32] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2015. Web-scale training for face identification. In *CVPR*.
[33] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. YFCC100M: The New Data in Multimedia Research. *Commun. ACM* (2016).
[34] Martin Traverso. 2013. Presto: Interacting with petabytes of data at Facebook. https://www.facebook.com/notes/facebook-engineering/presto-interacting-with-petabytes-of-data-at-facebook/10151786197628920/.
[35] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. [n. d.]. Learning From Massive Noisy Labeled Data for Image Classification. In *CVPR, 2015*.
[36] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated Residual Transformations for Deep Neural Networks. *CVPR* (2017).
[37] Fan Yang, Ajinkya Kale, Yury Bubnov, Leon Stein, Qiaosong Wang, Hadi Kiapour, and Robinson Piramuthu. 2017. Visual search at Ebay. In *KDD*.
[38] Shengqi Yang, Cristina Scheau, and Fei Yang. 2017. Facebook's Photo Search. https://code.fb.com/ml-applications/under-the-hood-photo-search/
[39] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. 2017. Visual Discovery at Pinterest. In *WWW*.
[40] Andrew Zhai and Hao-Yu Wu. 2018. Making Classification Competitive for Deep Metric Learning. *CoRR* (2018).
[41] Bohan Zhuang, Lingqiao Liu, Yao Li, Chunhua Shen, and Ian Reid. [n. d.]. Attend in Groups: A Weakly-Supervised Deep Learning Framework for Learning From Web Data. In *CVPR, 2017*.