# GrokNet: Unified Computer Vision Model Trunk and Embeddings For Commerce

Sean Bell, Yiqun Liu, Sami Alsheikh, Yina Tang, Ed Pizzi,
M. Henning, Karun Singh, Omkar Parkhi, Fedor Borisyuk
Facebook Inc.

## ABSTRACT

In this paper, we present *GrokNet*, a deployed image recognition system for commerce applications. *GrokNet* leverages a multi-task learning approach to train a single computer vision trunk. We achieve a 2.1x improvement in exact product match accuracy when compared to the previous state-of-the-art Facebook product recognition system. We achieve this by training on 7 datasets across several commerce verticals, using 80 categorical loss functions and 3 embedding losses. We share our experience of combining diverse sources with wide-ranging label semantics and image statistics, including learning from human annotations, user-generated tags, and noisy search engine interaction data. *GrokNet* has demonstrated gains in production applications and operates at Facebook scale.

## CCS CONCEPTS

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

## KEYWORDS

Image classification, e-commerce image understanding, multi-task learning, embedding, deep learning

**ACM Reference Format:**
Sean Bell, Yiqun Liu, Sami Alsheikh, Yina Tang, Ed Pizzi, and M. Henning, Karun Singh, Omkar Parkhi, Fedor Borisyuk. 2020. GrokNet: Unified Computer Vision Model Trunk and Embeddings For Commerce. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 22–27, 2020, San Diego, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Facebook Marketplace[1] plays an important role for many people across the globe to provide a platform for selling and buying products from Facebook members and businesses. Millions of products are deployed for sale everyday and Marketplace provides ecosystem for buyers to explore, find and buy what they need. Usually product descriptions include limited information such as title, short textual description, price, location, set of images, and rarely – color, material, brand, or year of production. Since every listing has photos, we can infer and augment missing information to improve the Marketplace user experience. We use image classifiers to augment product descriptions with predicted attributes, categories, and search queries, thereby improving the ability to find and browse products. In our previous e-commerce image recognition system (MSURU [26]), we used search log interaction data to train these image classifiers with large-scale weakly-supervised data.

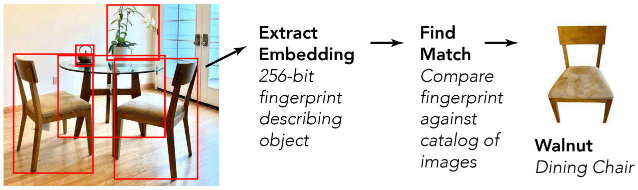

**Figure 1: Product Recognition Service, useful for many applications: product tagging, visual search, "shop the look," clustering, de-duplication, ranking, and recommendations.**

In this paper we present *GrokNet*, a unified computer vision model, which incorporates a diverse set of loss functions, optimizing jointly for exact product recognition accuracy and various classification tasks. *GrokNet* is trained on human annotations, user-generated tags, and noisy search engine interaction data. Our final trained system analyzes images to predict the following:

- *Object category:* "bar stool," "scarf," "area rug," ...
- *Home attributes:* object color, material, decor style,
- *Fashion attributes:* style, color, material, sleeve length, ...
- *Vehicle attributes:* make, model, external color, decade,
- *Search queries:* text phrases likely used by users to find the product on Marketplace Search,
- *Image embedding:* 256-bit hash used to recognize exact products, find and rank similar products, improve search quality.

Figures 1 and 2 show some applications for product recognition.

Several challenges arose as we built *GrokNet*. The outputs of our image recognition systems are used across various set of applications in Facebook Marketplace including Feed, Search, Visually Similar Product Recommendations, and Marketplace Catalog. Some applications like Search or Feed optimizes for increasing interactions of buyers and sellers constrained on relevance of presented results, while other applications like Marketplace Catalog optimizes for finding duplicate product listings and requires exact product recognition. Therefore, *GrokNet* was **required to serve various sets of optimization targets** for different applications, which constituted a challenge during development and deployment.

In practice, problem of building multi-task computer vision trunk is challenged by the **problem of combining different datasets** ranging from high quality human rater annotations to user-generated

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

*KDD '20, August 22–27, 2020, San Diego, CA, USA*
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

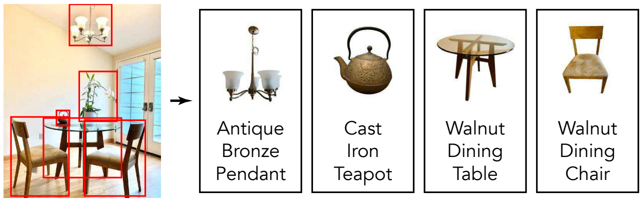

**Figure 2: Recognizing all products in a scene photo.**

tags, and noisy search log interaction data. Additionally, different datasets have various image properties from high quality imagery of product catalogs to lower quality mobile photos. Datasets also vary in sizes with biggest dataset coming from search log interaction data of 56M images to the smaller datasets of human annotated images with size in tens of thousands (described in §3.1).

Finally, with a single model, we aimed to **reduce maintenance and computational cost and improve coverage** by removing the need of separate models per vertical applications. From past experience we observed that strongly-supervised and vertical-specific models can yield higher accuracy than general models. However, specialized models only provide an output for in-domain products, which is a problem for ranking or search applications that require high coverage. In contrast, *GrokNet* is a single unified model with full coverage across all products.

## 2 RELATED WORK

*Visual recognition in the industry.* In recent years, visual recognition technologies have been applied across the industry, including systems developed by Facebook [26], Ebay [33], Microsoft [14], Amazon [30], Pinterest [36], Alibaba [37] and Google [2]. As described above, this paper presents a novel computer vision model for fine-grained product recognition, trained with a massive multi-task objective, achieving 2.1x improvement over prior systems.

*Deep metric learning.* Deep metric learning aims to learn a discriminative feature space such that similar examples are nearby in the space compared to dissimilar examples. This allows for open-vocabulary recognition of new concepts not present in the training set (using nearest-neighbor search). Deep metric learning is typically supervised by pairwise or triplet losses, which directly supervise distances between elements in the training set.

A recent thread of research has explored using modified classification losses to supervise deep metric learning [35]. These approaches have been particularly successful in the face recognition domain, where they typically outperform pairwise or triplet approaches [9, 17, 28, 29]. Similar techniques have been applied to instance-level image identification for unsupervised feature learning [31]. Classification-based approaches maintain a weight vector per labeled entity in the training set at training time (which is removed for inference), which constrains the number of identities that can be used for training. Techniques like negative sampling and noise-contrastive estimation can extend this limit for softmax cross-entropy losses [31], however training-time model storage remains linear in the number of classes.

We combine both pairwise and classification-based losses into a single model. We adapt ArcFace [9], a classification-based metric loss, to visual product recognition. We combine it with a pairwise loss to supervise a single embedding space. This allows training on more classes than is practical using ArcFace alone, while benefiting from the improved retrieval performance that ArcFace provides. We further extend ArcFace loss to the multi-label case.

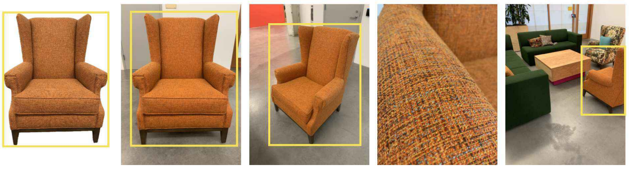

**Figure 3: Product Identities Data Collection. Annotators are presented with a set of images that potentially contain the same product identity. Their task is to draw tight bounding boxes (shown in yellow) around all instances of the shown product. Note that no box is drawn in the fourth image, since it is a zoom and cannot be confidently associated with the shown product. With this setup, we are able to collect a dataset that contains multiple unique depictions of the exact same product identity.**

*Learning from noisy data.* There is a line of work that demonstrated the power of using *large-scale, weakly supervised* image datasets for effective visual learning. For example, Joulin *et al.* [15] trained convolutional networks to predict words on a collection of 100 million Flickr photos, [25] trained on the JFT-300M dataset of 300 million weakly supervised images. Within the search system of Pinterest [34], the authors use millions of images with 20$k$ classes to train image classifiers using user annotated data from pin-boards. At Facebook, Mahajan *et al.* [19] trained on hashtags from billions of images, and Tang *et al.* [26] (MSURU) trained on Marketplace search interaction logs, both achieving gains in accuracy from large scale noisy data.

*Multi-task learning.* There has been multiple works exploring joining multiple tasks into one computer vision model [16, 18, 21, 23, 36], where authors explore synergy across different datasets to improve overall performance. In prior works [6, 16] authors investigated to balance multiple loss objectives to optimize model accuracy. The work of Zhai *et al.* [36] investigated joining multiple datasets in the production setting of Pinterest deploying a unified embedding to three user-facing applications.

## 3 MODELING

In this section, we describe how we combine several disparate large datasets with different data types to train a single unified model, how we combine many different loss functions with multi-task learning, and finally, how we trained and compressed the resulting float embedding by 50x, from 400D floats to 256-bit hashes.

### 3.1 Training Data

One of the goals of this project was to solve a large number of computer vision tasks with a single model, by training on both existing datasets as well as many new ones. For this project, we combine 7 different datasets with wide-ranging label semantics and image statistics. Our combined dataset contains 89 million public images from Facebook Marketplace. Our diverse image statistics include user photos of Marketplace products for sale ("seller photos"),

| Dataset | Num. Images | Label Types | Pairwise Loss | ArcFace | Multi-Label ArcFace | Softmax | Multi-Label Softmax |
|---|---|---|---|---|---|---|---|
| Object Categories | ~4,400,000 | Single-label | | | | ✓ | |
| Home Products | ~390,000 | Product IDs | ✓ | ✓ | | | |
| Home Attributes | ~1,500,000 | Multi-label | | | | | ✓ |
| Fashion Products | ~1,100,000 | Product IDs | ✓ | | | | |
| Fashion Attributes | ~66,000 | Single-label | | | | ✓ | |
| Vehicles | ~25,000,000 | Single-label, Product IDs | ✓ | ✓ | | ✓ | |
| Search Queries | ~56,000,000 | Multi-label | | | ✓ | | ✓ |

**Table 1: *GrokNet* training data (total: 89M images, 83 losses, see §3.1). Right 5 columns: which datasets affect which losses.**

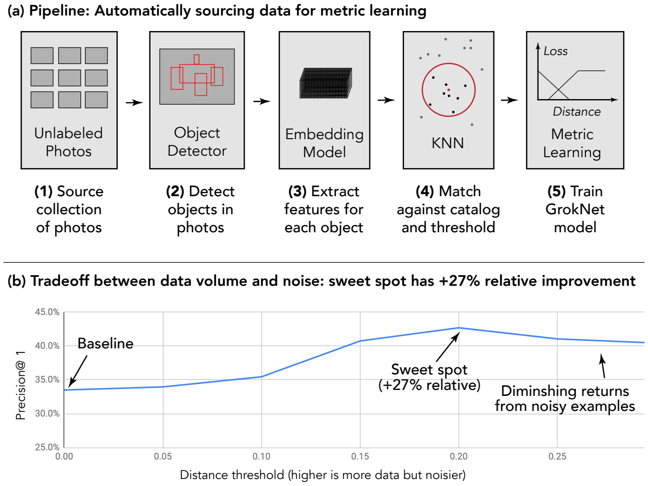

**Figure 4: Weakly supervised data augmentation – we use a prior embedding model to generate additional noisy Product ID training data. See §3.1.4.**

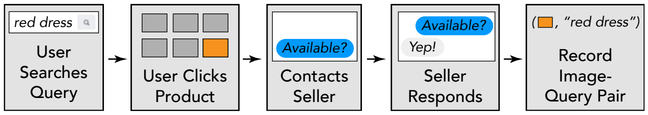

**Figure 5: Marketplace Search Queries. We create a dataset of image-query pairs by filtering for the following sequence of events: (1) a user searches for a query, (2) clicks on a product, (3) messages a Marketplace seller, and (4) the seller responds back. If all 4 events happen within a short time, it is highly likely that the user found what they were looking for, and thus the query is relevant. As described in §3.1.5, we do not have access to message contents and only know the fact that users interacted with each other.**

professionally photographed home and fashion scenes ("scene photos"), and catalog-like photos displaying products usually against a simple background ("iconic photos"). We train both on human-provided annotations and weak signals. We describe the datasets in more detail in Table 1 and in the sections below.

*3.1.1 Object Categories.* We use object categories from an internal human-annotated dataset of Marketplace images, with one of 566 labels such as "chair", "bracelet", and "bicycle."

*3.1.2 Attributes (Fashion, Home, and Vehicles).* We collect multi-label annotations for attributes on home and fashion products, and use public post metadata for vehicles. For home products, we label attribute types such as color (e.g., "beige") and material (e.g., "metal"). For fashion products, we label data for 68 attribute types across 9 object categories (818 total), with attribute types varying from high-level concepts (style, color, material) to fine-grained details (dress waistline, shirt embellishments). For vehicles, we use data for color, make, model, decade, and vehicle type (e.g., "sedan").

*3.1.3 Product Identities (Fashion, Home, and Vehicles).* In addition to the object categories and attributes, we leverage product identity (ID) datasets for each of the fashion, home, and vehicle verticals. These 3 datasets have several images of the same product in varying contexts and from varying viewpoints. Different images of a car will have the same ID if they share the the exact same make, model, color, and decade (e.g., "Gray Toyota Camry 2000s"). Different images of a fashion or home product will have the same ID if they are the exact same variation of the same product (matching color, dimensions, etc.). Figure 3 shows the annotation task used to clean this data and collect tight bounding boxes. For vehicles, we use an in-house object detector to provide high-confidence "car" detections for bounding boxes instead of human annotations.

*3.1.4 Weakly supervised data augmentation.* Product IDs are the most difficult data to collect, since there are millions of possible IDs and it is not straightforward to ask humans to label. Here we present a technique to automatically generate additional Product ID examples using our model as a feedback loop. As illustrated in Figure 4(a), we gather a large collection of unlabeled photos, detect object boxes in each photo using our in-house object detector, and compute embedding features for each box. These boxes are candidates that might be useful training data. We then match each candidate to our list of known products, take the closest match (1-nearest neighbor), and consider it to be a correct match if the distance is below a threshold. This becomes additional data to augment our training set with no human-labeling cost.

In Figure 4(b), we experiment with different embedding distance thresholds with separate held-out experiments. As we increase the threshold, we collect more matches, but at the detriment of adding noise and reducing precision of the training set. We find a sweet spot around a distance threshold of 0.2, where we get a +27% relative improvement in Precision@1, compared to not using this data. Metrics such as Precision@1 are described later in §5.1.2.

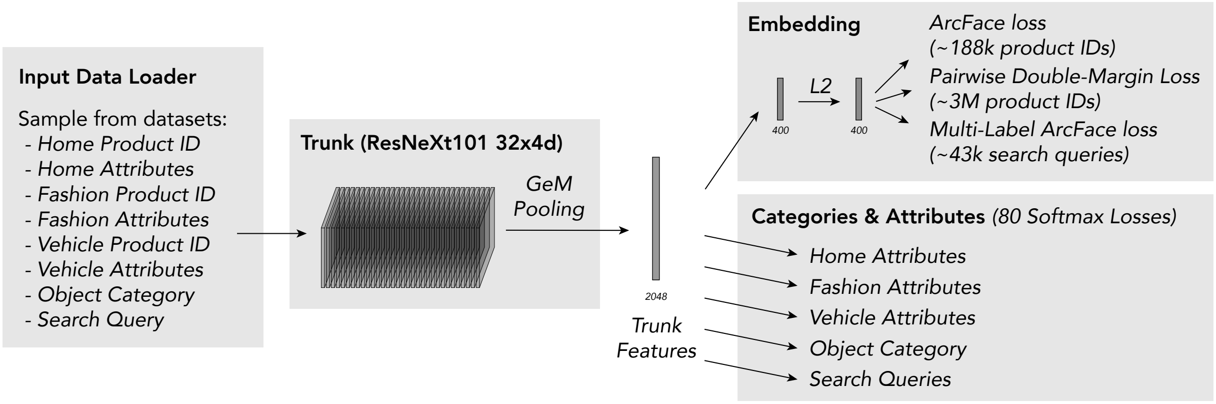

**Figure 6: *GrokNet* training architecture: 7 datasets, 83 loss functions (80 categorical losses + 3 embedding losses). The data loader samples a fixed number of images per batch from each dataset, and the losses are combined with a weighted sum. At inference time, we further compress the embedding (not shown here) from 400 dimensions to 256 bits (see §4.0.1).**

*3.1.5 Marketplace Search Queries.* Here we describe how we create a dataset of image-query pairs from search logs. Facebook Marketplace is a global-scale marketplace of items for sale, where buyers can list items for sale along with photos, price, description, and other metadata. Users can browse and search for products using Marketplace Feed and Search. Once a buyer has found a product, they can choose to message buyers about the product, such as asking questions about the product or its availability, or arranging a sale. We do not have access to message contents and only know the fact that users interacted with each other. We consider this as a proxy for an *add-to-cart* action on an e-commerce website. For this project, we use anonymized search log data from Marketplace Search to create a dataset of image-query pairs. Figure 5 describes how we create our dataset, following the same technique as MSURU [26]. We further extend MSURU data collection with *dataset cleaning* methods (described later in §5.2), which automatically reject irrelevant image-query pairs and reduce noise in the training set. The end result is a dataset of 56M images, with each image having a list of text queries estimated to be relevant for that image. We keep the top 45k most common queries, which improves precision and ensures that we have at least 300 images per query.

## 3.2 Trunk Architecture

*GrokNet* is a large-scale unification of several datasets and machine learning tasks – in total we have 7 datasets (§3.1) and 83 different loss functions (§3.3), as shown in Figure 6. In this section, we describe the underlying convolutional neural network model that forms the "trunk" of the model. We build our system as a distributed PyTorch [1] workflow in the FBLearner framework [11].

The trunk model for *GrokNet* uses ResNeXt-101 32×4d, which has 101 layers, 32 groups, and group width 4 (8B multiply-add FLOPs, 43M parameters) [32]. We initialize weights from [19], which was pre-trained on 3.5B images and 17k hashtags. We then fine-tune on our datasets using Distributed Data Parallel GPU training on 8-GPU hosts, across 12 hosts (96 total GPUs).

*3.2.1 GeM Pooling.* At the top of the trunk, we replace average pooling with generalized mean (GeM) pooling [4, 10, 22, 27], which is a parameterized pooling operation that is equivalent to average pooling for $p = 1$, and max pooling for $p = \infty$. Intuitively, this allows the embedding to concentrate more of the network's attention to salient parts of the image for each feature. We follow the method of [22], and learn the pooling parameter $p$ directly for our experiments. After training, our final model converges to a value $p \approx 3$. In separate held-out experiments on a single dataset, we found a +26% relative improvement in Precision@1 compared to average pooling.

## 3.3 Loss Functions

*GrokNet* unifies several distinct tasks into a single architecture (Figure 6), combining several loss functions and loss function types in a weighted sum. To train the 80 category and attribute heads, we use Softmax and Multi-label Softmax [19]. To train the unified embedding head, we use 3 metric learning losses operating over the same space—ArcFace [9], Multi-label ArcFace, and Pairwise Embedding Loss. The latter two are new extensions on past work, and we describe all losses in detail below.

*3.3.1 Softmax Losses.* We add categorical labels to our model using softmax with cross-entropy Loss, as is standard in the literature [12]. These labels are described in §3.1 and include object categories, home attributes, fashion attributes, and vehicle attributes. We group together categories/attributes that are mutually exclusive with respect to each other—for example, "object category" is a single group, "dress color" is another group. There are 80 groups and thus 80 softmaxes. For multi-label datasets, we use multi-label cross entropy, where each positive target is set to be $1/k$ if there are $k$ positive labels for the image [19]. Since there are so many different losses, most gradients will be zero in most iterations.

*3.3.2 Multi-Label ArcFace Loss.* ArcFace loss [9] is a modified classification objective originally introduced for face recognition. ArcFace loss expects a single label per training example. However our Marketplace Search Queries dataset (§3.1.5) often associates each product image with multiple search queries. To address this, we extend ArcFace loss to allow for *multiple labels per image.*

Multi-Label ArcFace uses cosine similarity between embeddings $x_i$ and "class center" vectors $w_j$ for each class, where each image is pulled towards multiple class centers (vs. a single center in ArcFace).

| | Weight | | Precision@1 |
|---|---|---|---|
| | CE Loss | M-AF | MSURU-700k |
| MSURU baseline [26] | 1 | 0 | - |
| Multi-Arcface Loss | 0 | 1 | -1.84% |
| Weighted Combined Loss | 1 | 10 | **+11.15%** |

**Table 2: Precision@1 trained on the *Marketplace Search Queries* dataset (§3.1.5) after 100K iterations. A combined loss with Multi-label ArcFace (weight 10) and softmax cross entropy (weight 1) outperforms cross entropy loss and Multi-label ArcFace loss respectively.**

| Arcface | Weight | | Precision@1 |
|---|---|---|---|
| Type | CE Loss | (M-)AF | MSURU-700k |
| Single-label (rarest) | 1 | 10 | - |
| Single-label (commonest) | 1 | 10 | -1.79% |
| Multi-label | 1 | 10 | **+1.68%** |

**Table 3: Precision@1 trained on the *Marketplace Search Queries dataset* (§3.1.5) after 20K iterations. A combined loss with Multi-label ArcFace (weight 10) and softmax cross entropy (weight 1) outperforms combined loss with single-label (choose the rarest label) Arcface with the same weights.**

We compute class scores as $z_{i,j} = \cos(\theta_{i,j}) = \frac{x_i^T w_j}{\|x_i\| \|w_j\|}$. We then add an angular margin $m$ to all of the image's labeled classes. The class scores are scaled by a constant factor $s$, before computing a softmax cross-entropy loss with respect to a uniform distribution over the labeled classes $Y_i$. We define Multi-Label ArcFace loss as:

$$\mathcal{L}_{M-AF}(x_i, Y_i) = -\frac{1}{|Y_i|} \sum_{j \in Y_i} \log \frac{e^{s\cos(m+\theta_{i,j})}}{\sum_{j \in Y_i} e^{s\cos(m+\theta_{i,j})} + \sum_{k \in C \setminus Y_i} e^{s\cos(\theta_{i,k})}}$$

where $Y_i = \{y_{ij}\}_{j=1}^{N_i}$ is the set of $N_i$ labeled classes for image $i$.

For the *Marketplace Search Queries* dataset (3.1.5), which is noisy and multi-labeled, we apply a weighted combination of multi-label Arcface loss and softmax cross-entropy loss. We find that this combined loss outperforms our previous production system MSURU [26] by +11.15% Precision@1 (Table 2). MSURU uses only multi-label softmax cross entropy loss. Our evaluation uses the MSURU-700k test set described in §5.1.2.

We also experimented with a single label Arcface loss (using the most rare/common label in the case of multiple labels per image) in the combined loss. We show that, with the same weight distribution, using multi-label Arcface in the combined loss performs +1.68% better, when training on the *Marketplace Search Queries* dataset alone (Table 3).

### *3.3.3 Pairwise Double-Margin Loss.*

ArcFace has very high accuracy for exact product recognition on our data. However it limits the number of products we can include in our training set, since each product requires its own class vector. To resolve this, we include all products using a pairwise loss (described here), and only supervise a subset of products using ArcFace loss. Note that both losses supervise the *same embedding space* with a weighted sum, as shown in Figure 6.

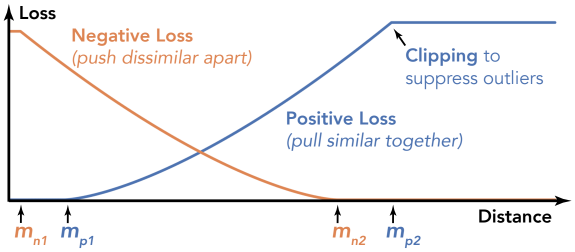

**Figure 7: Pairwise Double-Margin Loss.**

For our pairwise loss, we use a modified form of Contrastive Loss [3, 13, 30], with the following differences: (1) we use an extra margin on both positive and negatives to suppress outliers, (2) we use a power parameter $\alpha$ to adjust the balance between hard and easy examples, and (3) we train on every pair within the batch to substantially increase the number of pairs. We define the loss as:

$$\mathcal{L}_{PW} = \frac{1}{N_p \alpha} \sum_{i,j} \text{clamp}\left(\text{D}(x_i, x_j) - m_{p1}, 0, m_{p2} - m_{p1}\right)^{\alpha} (y_{ij}) + \frac{w_n}{N_n \alpha} \sum_{i,j} \text{clamp}\left(m_{n2} - \text{D}(x_i, x_j), 0, m_{n2} - m_{n1}\right)^{\alpha} (1 - y_{ij})$$

where $x_i$ are embedding points, $y_{ij} \in \{0, 1\}$ indicates whether each pair of indices $(i, j)$ is a positive (1) or negative (0) pair, $\text{D}(x_i, x_j) = ||x_i - x_j||_2^2$ is the squared distance between two points, $N_p$ is the number of positives, $N_n$ is the number of negatives, $w_n$ is a weight to control relative influence of positive and negative, $\alpha$ is a loss power to increase the weight of harder examples, $m_{p1}$, $m_{p2}$ are positive double-margins, and $m_{p1}$, $m_{p2}$ are negative double-margins.

As illustrated in Figure 7, we include an extra margin to provide resilience to outliers. If there is a false positive in our dataset, then we will be adding a high-strength force pulling two very different parts of the embedding together, distorting the space. The extra margin protects against this. Further, we adjust the shape of the curve with $\alpha$. What matters for gradient descent is the gradient, not the absolute value, so higher values of $\alpha$ put a higher weight on harder examples with larger loss, and a value of $\alpha = 1$ weights them equally. Alternatively, we can see $\alpha$ as weighting each example with a magnitude proportional to its own loss value.

We compute the loss efficiently by first pre-computing a full pairwise distance matrix. Every example $x$ with a Product ID contributes to the loss. So, if there are $B$ examples with Product IDs per batch, then we have $B^2$ total pairs and $N_p + N_n = B^2$. Any examples without Product IDs (for example, categorical data) are excluded from the loss. This gives us quadratically more examples ($B^2$ vs $B$) compared to some prior approaches [3, 7], which use each example in the batch once.

Based on experimentation on our datasets, we use the following values: loss power $\alpha = 1.5$, margins $m_{p1} = 0.1$, $m_{p2} = 0.7$, $m_{n1} = 0$, $m_{n2} = 0.7$, and negative weight $w_n = 10000$.

### *3.3.4 Distributed Pairwise Computation.*

We further extended our loss by computing distances over all possible pairs across all GPUs and all hosts. This requires a very large distributed "all gather" over

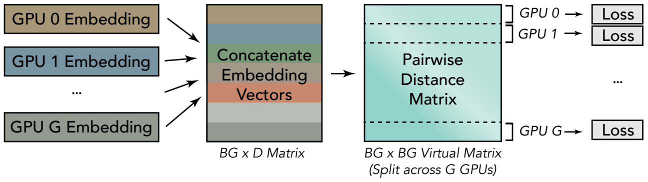

**Figure 8: Distributed pairwise embedding computation.** $D$**: embedding dimension,** $B$ **batchsize,** $G$**: number of GPUs.**

96 GPUs. With a global pairwise distance matrix, we get 96 times more distance matrix entries compared to computing within a single GPU ($96^2$ vs 96), and resulting in a +9.8% relative improvement in Precision@1. As illustrated in Figure 8, we do not actually materialize the full pairwise matrix, rather each GPU contains a slice to compare its set of local vectors with all other vectors.

*3.3.5 Combining Losses and Datasets Together.* As shown in Table 1, we have different datasets supervise different losses due to constraints in either memory or dataset format. For ArcFace, we can only include about 188k products into memory since there are so many other components to the model also held in memory—we select the 188k products with the most images per product. All Product IDs are included in the Pairwise Loss. For the Search Query data, which has multiple images per label, we include it both in the embedding via a Multi-Label ArcFace Loss, and as a separate Softmax prediction head. Tradeoffs are discussed more in §5.3.

## 4 TRAINING AND DEPLOYMENT

We train with SGD for 100,000 iterations on 96 NVIDIA V100 GPUs with a batch size of 46 per GPU, learning rate $0.012 \times 12 = 0.0144$, momentum 0.9, and weight decay 1e-4. This is the largest configuration that would fit in memory. Batch norm statistics are synchronized across all GPUs, giving a +4% relative improvement, which is important given our relatively small batch size.

*4.0.1 Compressing Embeddings 50x to 256-bit.* After training our continuous embedding (400 dimensions), we quantize to a 256 bit hash using Catalyzer [24]. We change the configuration to use a residual block, which we find slightly outperforms the original multi-layer perceptron architecture from [24]. We further tried many configurations (2674 snapshots in total), including changing the weights for the entropic regularizer, learning rates, bias, and did not find a better result compared to the default parameters from [24]. Despite requiring 50x less storage and compute at runtime, our binary accuracy is equivalent to the original raw embedding on the Furniture test set (40.3% binary vs 40.1% continuous). Figure 9 shows a histogram of distances on our furniture test set, before and after compression – we can see that Catalyzer keeps the overlap between positive and negative consistent.

*4.0.2 Calibration.* Once we are done training, the softmax heads output scores in the range [0, 1] and sum to 1, so they behave like a probability, but are not actual probability estimates. We can transform the scores using a mapping from raw score to calibrated probabilities, so that for predictions with score $S$, we expect that approximately $S$% are correct [8]. This process is called "calibration," and we apply this technique to our attribute predictions, using isotonic regression on held-out data.

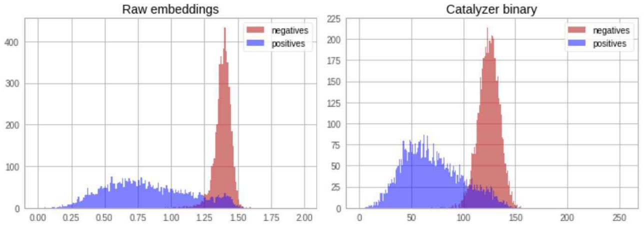

**Figure 9: Compressing embeddings 50x using Catalyzer. We plot histograms of positive (blue) and negative (red) distances. Left: 400D, right: 256-bit. We can see that the relative ranking is maintained before and after compression.**

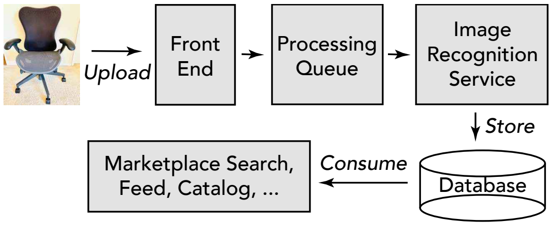
Upload
Front End
Processing Queue
Image Recognition Service
Store
Database
Consume
Marketplace Search, Feed, Catalog, ...

**Figure 10:** ***GrokNet*** **Image Recognition Service. Images are uploaded to a queue and scheduled to run on prediction machines which feed resized images through our convolutional networks. We store outputs in distributed key-value stores, and use them for applications such as Marketplace Search, Feed, Catalogue which consume our signals within seconds of new listings being created.**

*4.0.3 Query Score Thresholds.* Unfortunately we can't easily calibrate our query prediction head, since we don't know which of the 45k queries are truly valid for each image, and this would not be feasible to label with any reasonable scale. It would require 45k binary labels per image. Instead, we estimate a single threshold per query, using a combination of KNN search and k-means clustering. We start with a class-balanced test dataset by selecting $N$ positive samples for each query. Then these positive examples are used as KNN seed to retrieve $2N$ highest ranked neighbors within the same dataset. We assume that the predicted probability for the $2N$ highest ranked neighbors follows a bi-modal distribution where the positive samples form one modality and the negative ones form the other. We separately run one-dimensional k-means clustering among the $2N$ predicted scores for each search query and use the cluster centroids to determine the final threshold considering precision/recall trade-offs. We used $N = 200$ to determine the thresholds.

In production, when we run each image through the model, we compare all 45k query scores against their corresponding 45k thresholds, and keeping all queries above their thresholds and clipping to the 10 highest-scoring queries.

*4.0.4 Service Architecture. GrokNet* is deployed in production and is designed to operate on product images uploaded daily at Facebook scale in a real-time fashion, as shown in Figure 10.

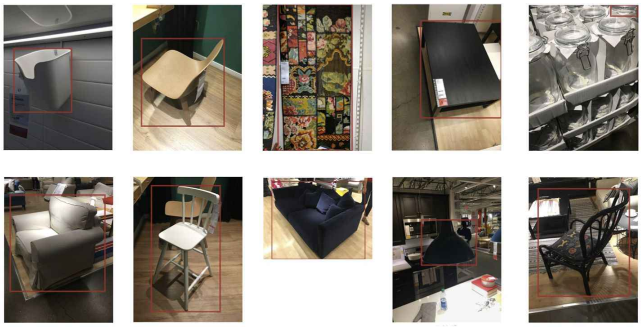

**Figure 11: Examples from our *Furniture Mobile Photos* test set showing many challenging angles.**

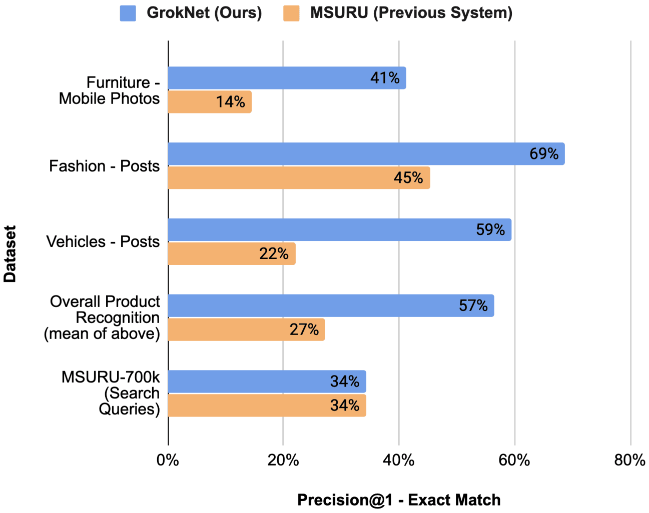

**Figure 12: Precision@1 across 4 fine-grained datasets (§5.1.1), demonstrating that *GrokNet* outperforms our previous system by 2.1x on exact product recognition tasks.**

# 5 EXPERIMENTS

## 5.1 Evaluation

In this section, we describe our evaluation data and metrics, detailing how we benchmarked our system against prior production systems at Facebook. We provide evaluation results for both hold-out training data (Table 1) and challenging mobile photographs.

*5.1.1 Data.* We benchmark our performance on several datasets:

*Furniture - Mobile Photos.* Our team manually took 2k mobile photos of furniture in stores, with human-annotated ground truth Product IDs and bounding boxes (Figure 11). During evaluation, these crops are queried against 206k images with 62k Product IDs.

*Fashion - Posts.* We use 40k public posts tagged with fashion products. We ask annotators to validate the tags and draw tight bounding boxes around the product both in the post's image and the iconic image. During evaluation, every crop is used to query against all other crops (excluding the query).

*Vehicles - Marketplace Posts.* We hold out 189k images from the Vehicle Product IDs dataset described in §3.1, and search all crops against all other crops.

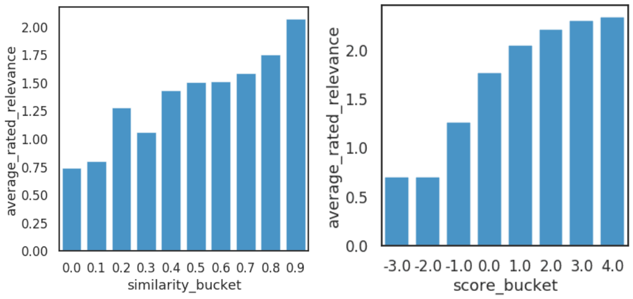

**Figure 13: Evaluation of data cleaning methods (rejecting irrelevant query-image pairs from the training set, §3.1.5, §5.2). Vertical axis: average human-rated relevance. Horizontal axis: predicted similarity by the model. An ideal method has monotonically increasing relevance from left to right. The two plots are different methods for data cleaning and we use both, left: Siamese network, right: Query-specific SVMs.**

| Number of Photos | Noisy Data % | Precision@1 Rated-20k | Precision@1 MSURU-700k |
|---|---|---|---|
| 59M | 37.2% | - | - |
| 59M | 27.1% | +1.4% | -0.3% |
| 56M | 0.0% | **+6.3%** | **+9.1%** |

**Table 4: Precision@1 relative improvement on Models trained with clean and partly-noisy data. This experiment uses the output of the last pooling layer in the trunk (sometimes called "pool5"). Precision@1 scores are relative %.**

*Search Queries - MSURU-700k.* We hold out 700k images from the Marketplace Search Queries dataset described in §3.1.5. We include all 45k search queries and use at most 15 photos for each query.

*5.1.2 Metrics.* We benchmark our model against our previous production system (MSURU) across the above four challenging fine-grained datasets. Both models use the same trunk architecture, produce 256-bit hashes, and measure hamming distance. For the product recognition datasets above (Furniture, Fashion, Vehicles), we measure Precision@1, the percentage of the time we retrieve the exact same Product ID in the top-1 position. For search query evaluation (MSURU-700k), we consider a result correct if the top-1 retrieved photo and query photo share at least one text query.

As shown in Figure 12, *GrokNet* is 2.1x more accurate than MSURU when averaging across our 3 product recognition datasets. For search query prediction using our embedding (last dataset in the plot), performance is the same as MSURU, despite training on many conflicting tasks (*GrokNet*) vs. a single task (MSURU).

## 5.2 Dataset Cleaning to Improve Precision

In §3.1.5, we described our method for creating the *Marketplace Search Queries* dataset. In this section, we describe how we further improve the precision of these queries via *dataset cleaning* methods, which automatically identify irrelevant queries and discard them from the dataset. We trained a Siamese network model to predict *{query, image}* similarity and SVM models for each search query. To train the Siamese network that predicts cosine similarity between the image and text, we used about 100M clicked *{query, image}* pairs as positives, and used an existing 256-bit image hash as one stream of input and internal 100-float word2vec [20] text embeddings trained from product titles as the other stream of input. To train the query-specific SVM models that predicts a similarity score between image and text, we only use the existing 256-bit image hash as input features, and trained them for all text queries with more than 25 clicked photos. For both models, we used random negatives as negative examples. We then remove the low-scored pairs from our dataset based on predictions from both methods.

We evaluate the effectiveness of the automated dataset cleaning with (a) human-rater evaluation and (b) KNN retrieval performance of the model trained from Search query data with different levels of noise. We collected ~3,500 human rated relevance scores for both the Siamese-network method and the Query-specific SVM method and found that they strongly correlate with the predicted model scores, as shown in Figure 13. The Siamese network scores are split into uniform buckets between [0,1] and same for Query-specific SVM model scores between [-3, 5]. The relevance is rated at 5 different levels between *0-off-topic*, *1-somewhat*, *2-reasonable*, *3-primary* and *4-vital*. Based on Figure 13, we can see that by selecting the right prediction thresholds, 0.95 for Siamese network and 1 for Query-specific SVM model, we are able to select training data that are of at least *2-reasonable* relevance.

To quantify the benefit of *dataset cleaning*, we train models on different versions of the MSURU dataset, replacing parts of the clean data with the discarded data that had low similarity scores predicted by either model, while keeping the same number of unique {image, query} pairs. In addition to evaluating the results on MSURU-700k, we collect another evaluation set, Rated-20k. It covers ~7k queries and contains human-rated {image, query} pairs that are at least of *2-reasonable* relevance, as defined in the paragraph above. Table 4 shows that on the Rated-20k dataset with ground truth labels, we can improve Precision@1 by +6.3% relative; on the weakly-supervised MSURU-700k dataset, we can improve by +9.1% relative.

## 5.3 Balancing Across 7 Datasets

When combining many different verticals and tasks into the same network, there is an explosion of possibilities in number of configurations to try. If each task has $n$ parameters with $m$ settings, then there are $(7n)^m$ total configurations to try with 7 datasets. We scoped down the design space by performing two-pair dataset ablations to understand the relative balance of two datasets. Figure 14 illustrates such an experiment: "what fraction of each training batch should be fashion, and what fraction should be vehicles?" We train the two-dataset system jointly and then separately evaluate Precision@1 on fashion and vehicle product recognition . We can see that in this case, we want to evenly balance these datasets. We applied similar experiments to pairwise-balance all 7 datasets. If we noticed that for some datasets, accuracy is not sensitive to batch size (*i.e.*, it performs the same no matter how it is configured), then this suggests that we can scope down that dataset's contribution to each batch, and use those slots for a more challenging task. We applied this insight with large-scale experiments to arrive at our final batch size distribution: search queries: 20, furniture products: 12, vehicle products: 4, fashion products: 4, object category: 4, fashion attributes: 2, home attributes: 2.

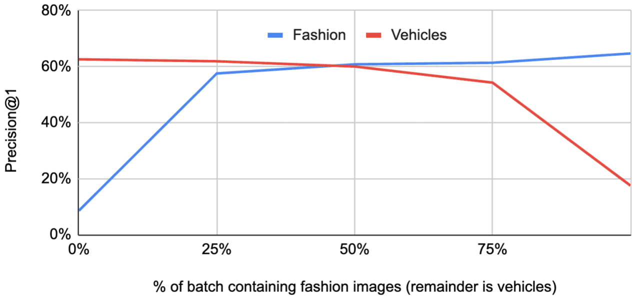

**Figure 14: Tradeoff between datasets. In this experiment, we train only with fashion and vehicle training data, and evaluate only on fashion and vehicle product recognition. We vary the fraction of the batch that contains each dataset, and normalize the losses by the number of examples so that they have equal contribution to the gradient when training.**

## 5.4 A/B Experiments

We confirmed the effectiveness of *GrokNet* by running production A/B test experiments and integrating with downstream applications. By leveraging *GrokNet* trunk to train new models and/or leveraging *GrokNet* predictions directly (embedding, categories, attributes, search queries) for Product Understanding, we observed improvement in several applications for Commerce.

**Marketplace Catalogue:** within the Marketplace ecosystem, many users are selling used items and thus are very likely to be selling exactly the same products but photographed and described differently. This creates a new large-scale deduplication and canonicalization challenge. With deduped listings, we can share properties across listings such as: likely queries each product can be retrieved by, or engagement counter features such as click-through-rate of the canonical product. One of the approaches we took is to develop a product similarity neural network (based on Siamese networks [5]) that predicts if given two products are duplicate. When using our *GrokNet* embedding as an input feature, we achieved a +76% relative improvement in precision (at 50% recall) compared to MSURU. Our image embeddings are a good fit for this task since (a) our training objective was well aligned with the target task of exact similarity and (b) every post has an image but not all of them have text or use the same language.

**Marketplace Search:** we have run A/B tests to confirm improvement of the relevance of Marketplace text-based search. As result of classification based on *GrokNet* we produce a set of *search queries*, which likely would be used by users to retrieve the product in Marketplace Search engine. These predicted queries are automatically ingested into inverted index of Marketplace search engine, and later can be matched to text-based searches from users. We keep the top 10 most confident query predictions. We replaced previous version based on MSURU [26] with *GrokNet* model *search queries* and observed +9% relative improvement in search relevance quality. We measure search relevance the percentage of search sessions with no incorrectly surfaced product among top ten ranked results. Due to the fine granularity of optimization targets for *GrokNet*, such as exact product recognition, we were able to identify difference between visually similar products with higher precision. In particular we noticed that the improvement on Marketplace Vehicles dataset was more than double the precision (vehicles was one of the most incorrectly classified category reported in search feedback), which helped to reduce incorrect search query tag predictions for products and thus improve search relevance.

After *GrokNet* predictions are calculated after inference, we store *{query, confidence}* pairs, where *query* is a text string, and *confidence* represents the likelihood of such *query* describing the image. In our previous Image Recognition model (MSURU [26]), we used global threshold to remove non-confident query predictions. During data analysis we noticed that when we apply global threshold to query predictions we kept some of incorrectly predicted query tags, and identified that to improve the precision of query predictions we need specific per-query thresholds. However, the challenge was that we don't have any human annotated data usually used to calibrate the threshold to find the balance between precision and recall, and all of our data is noisy engagement interactions from search log. We solved this using the per query threshold selection method described in §4.0.3. We on-boarded per query selection threshold to Marketplace Search and ran A/B test experiments, using predicted search queries for retrieval of products by matching the text user query. We observed over +8% relative improvement in search relevance due to per-query threshold selection.

## 6 CONCLUSION

We presented approaches for building an accurate image product recognition system called *GrokNet*, resulting in a unified computer vision model that incorporates a diverse set of 83 loss functions, optimizing jointly for exact product recognition accuracy and various classification tasks over 7 commerce datasets. We shared innovative ideas for handling *large weakly-supervised* training data using cleaning and how to improve accuracy by model results post processing, and provided practical advice on how to develop, deploy, and integrate modern state-of-the-art image product recognition system to applications operating at Facebook scale.

## 7 ACKNOWLEDGEMENTS

The authors would like to thank Cristina Scheau, Manohar Paluri, Matan Levi, Animesh Sinha, Maha El Choubassi, Jun Mei, Lumin Zhang, Varun Nasery, Siddarth Malreddy, Priyanka Kukreja, Alex Berg, Tamara Berg, Anuj Madan, Iwona Bialynicka-Birula, Rui Li, Jonathan Tan, Jason Liao, Tong Zhou, Jiang Han, Jon Guerin, Quinn Sloan, Charles Bai, Michelle Cheung, Kavita Bala, and others who contributed, supported and collaborated with us throughout.

## REFERENCES

[1] 2016. PyTorch. http://pytorch.org/
[2] 2017. Google Lens. https://lens.google.com/
[3] Sean Bell and Kavita Bala. 2015. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. Graph.* (2015).
[4] Maxim Berman, Hervé Jégou, Vedaldi Andrea, Iasonas Kokkinos, and Matthijs Douze. 2019. MultiGrain: a unified image embedding for classes and instances. *arXiv e-prints* (Feb 2019).
[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In *NIPS*.
[6] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In *ICML*.
[7] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *CVPR*.
[8] Chuan, Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *JMLR*.
[9] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In *CVPR*.
[10] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge J. Belongie. 2009. Integral Channel Features. In *BMVC*.
[11] Jeffrey Dunn. 2016. Introducing FBLearner Flow: Facebook's AI backbone. https://code.fb.com/core-data/introducing-fblearner-flow-facebook-s-ai-backbone/
[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.
[13] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*.
[14] 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*.
[15] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. 2016. Learning visual features from large weakly supervised data. In *ECCV*.
[16] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. *CoRR* abs/1705.07115 (2017).
[17] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *CVPR*.
[18] Wenjie Luo, Bin Yang, and Raquel Urtasun. 2018. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net. In *CVPR*.
[19] 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*.
[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.
[21] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-Stitch Networks for Multi-task Learning. In *CVPR*.
[22] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. 2019. Fine-Tuning CNN Image Retrieval with No Human Annotation. (2019).
[23] Zhongzheng Ren and Yong Jae Lee. 2017. Cross-Domain Self-supervised Multitask Feature Learning using Synthetic Imagery. arXiv:cs.CV/1711.09082
[24] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. 2019. Spreading vectors for similarity search. In *ICLR*.
[25] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*.
[26] 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 *KDD*.
[27] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2016. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*.
[28] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. NormFace: $L_2$ Hypersphere Embedding for Face Verification. In *Multimedia Conference, MM*.
[29] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. 2018. CosFace: Large Margin Cosine Loss for Deep Face Recognition. In *CVPR*.
[30] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. 2017. Sampling Matters in Deep Embedding Learning. In *ICCV*.
[31] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. 2018. Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. *CoRR* (2018).
[32] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2016. Aggregated Residual Transformations for Deep Neural Networks. *CVPR* (2016).
[33] Fan Yang, Ajinkya Kale, Yury Bubnov, Leon Stein, Qiaosong Wang, M. Hadi Kiapour, and Robinson Piramuthu. 2017. Visual Search at eBay. In *KDD*.
[34] 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*.
[35] Andrew Zhai and Hao-Yu Wu. 2019. Classification is a strong baseline for deep metric learning. In *BMVC*.
[36] Andrew Zhai, Hao-Yu Wu, Eric Tzeng, Dong Huk Park, and Charles Rosenberg. 2019. Learning a Unified Embedding for Visual Search at Pinterest. In *KDD*.
[37] Yanhao Zhang, Pan Pan, Yun Zheng, Kang Zhao, Yingya Zhang, Xiaofeng Ren, and Rong Jin. 2018. Visual Search at Alibaba. In *KDD*.